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 studies a simple attribute graph grammar as a generative image representation for image parsing in a Bayesian framework. This grammar has one class of primitives as its terminal nodes – 3D planar rectangles projected on images, and six production rules for the spatial layout of the rectangular objects. All the terminal and non-terminal nodes in the grammar are described by attributes for their geometric properties and image appearance. Each production rule either instantiates an non-terminal node as a rectangular primitive or expands a node into its components. A production rule is associated with a number of equations that constrain the attributes of a parent node and those of its children. This grammar, albeit simple, can produce a combinatorial number of configurations and objects in made-made scenes, such as buildings, hallways, kitchens, and living rooms etc. A configuration refers to a planar graph representation produced by a series of grammar rules in a parsing graph (augmented from a parsing tree by including horizontal constraints). A given image is then generated from a configuration by a primal sketch model [5] which is formulated as the likelihood in the Bayesian framework. The paper will be focused on designing an effective inference algorithm which computes (or construct) a hierarchical parsing graph from an input image in the process of maximizing a Bayesian posterior probability or equivalently minimizing a description length (MDL). It is worth clarifying that the parsing graph here is only a generic interpretation for the object layout and it is beyond the scope of this paper to deal with explicit scene and object recognition. The inference algorithm integrates bottom-up rectangle detection as weighted candidates (particles) which activate the grammar rules for top-down predictions of occluded or missing components. Intuitively each grammar rule maintains a list of particles as in an "assembly line". The top-down process chooses the most promising particle (with heaviest weight) at each step. The acceptance of a grammar rule means a recognition of a certain sub-configuration so that the description length is decreased in a greedy way, and it also activates a number of actions: (i) creating new "top-down" particles and insert them into the lists; (ii) reweighting some particles in the lists; (iii) passing attributes between a node and its parent through the constraint equations associated with this production rule. When an attribute is passed from a child node to a parent node, it is called bottom-up; and the opposite is called top-down. The whole procedure is, in spirit, similar to the datat-driven Markov chain Monte Carlo paradigm [20], [16], except that a greedy algorithm is adopted for simplicity.

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 patterns (objects) are composed of a few types of primitives arranged in a small set of spatial relations. This is very similar to language where a huge set of sentences can be generated with relatively small vocabulary and grammar rules in a hierarchy from words, phrases, clauses, and to sentences. In this paper, we study a simple attribute grammar as a generative image representation for image parsing in a Bayesian framework. Fig. 1 illustrates such a representation for a kitchen scene.

Given an input image, our objective is to compute a hierarchical parsing graph where each non-terminal node corresponds to a production rule. In this parsing graph, the vertical links show the decomposition of scene and objects into their components, and the horizontal (dashed) links specify the spatial relations between components through constraints on their attributes. Thus a parsing graph is more general than a parsing tree in language. The parsing graph then produces a planar configuration which is a sketch or graph representation as in the primal sketch model [5]. The parsing graph is not pre-determined but constructed "on-the-fly" from the input image with bottom-up and top-down steps which in combination maximize a Bayesian posterior probability in a greedy way.

It is worth clarifying that the parsing graph here is only a generic interpretation for the object layout and it is beyond the scope of this paper to deal with explicit scene and object recognition.

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

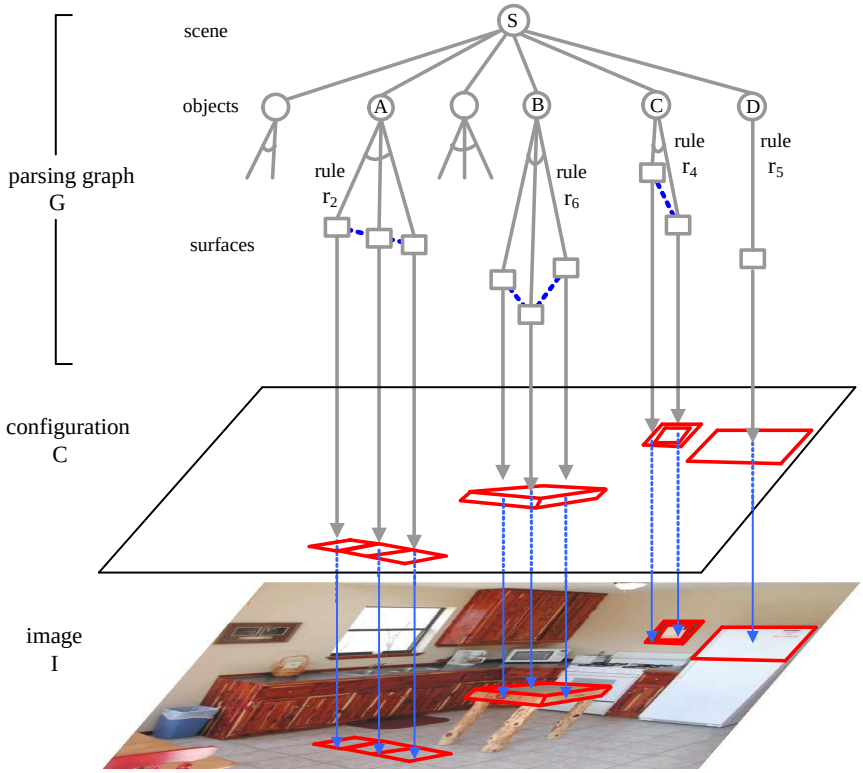


Fig. 1. An example of generative scene representation and parsing with graph grammar. The spatial layout of a kitchen scene is interpreted in a hierarchical parsing graph where each non-terminal node corresponds to a grammar rule and a terminal node is a projected rectangle. We only show part of the parsing graph for clarity. The parsing graph forms a planar configuration which in turn generates an image in a primal sketch model (Guo et al 2003). The parsing graph is inferred from an image in a top-down/bottom-up procedure by maximizing a Bayesian posterior probability.

A. Overview of the representation and model

In this paper, our grammar has one class of primitives as the terminal nodes which are 3D planar rectangles projected on images. Obviously rectangles are the most popular elements in man-made scenes, such as buildings, hallways, kitchens, living rooms etc. As shown in Fig. 3, a rectangle is specified by 8 variables called attributes. A degenerated rectangle has fewer variables. A rectangle consists of two pairs of parallel line segments which intersect at two vanishing points in the image plane. If a pair of line segments are parallel in the image plane, then their vanish point is at infinity. The grammar has six production rules as shown

in Fig. 4 and three of them are illustrated in Fig. 1. For example, a "line rule" aligns a number of rectangles in one row, a "nesting rule" has one rectangle containing the other, and a "cube rule" has three rectangles forming a rectangular box. Each production rule is associated with a number of equations that constrain the attributes of a parent node and those of its children. Thus our graph grammar is attributed. These rules can be used recursively to generate a large set of complex configurations. Fig. 2 shows two typical configurations – a floor pattern and a toolbox pattern, and their corresponding parsing graphs.

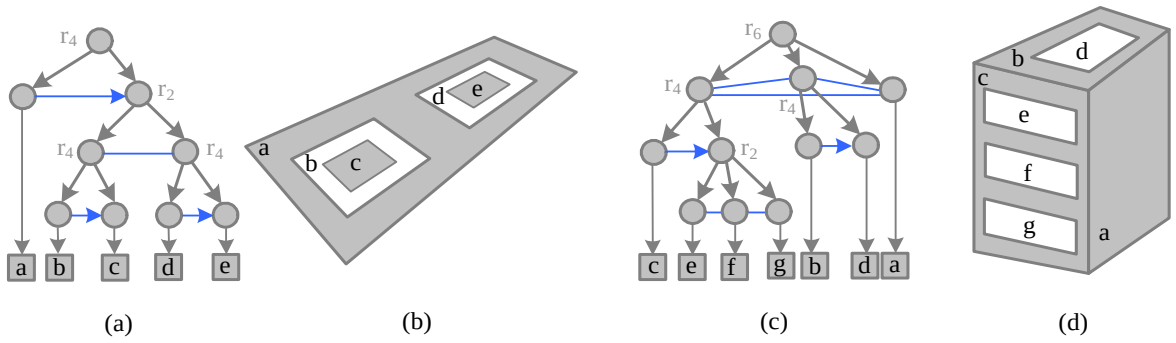


Fig. 2. Two examples of rectangle object configurations (b) and (d) and their corresponding parsing graphs (a) and (c). The production rules are showed as non-terminal nodes.

Note that the parsing graph is hierarchical, and each non-terminal node corresponds to a grammar rule. The configuration is represented in a planar sketch graph. This graph is further broken into smaller image primitives for edge elements, bars, and corners in an image primitive dictionary, which in turn, generates the image by the primal sketch model [5], [6]. Therefore our model is fully generative from the scene node to the pixels (see image reconstruction results in Figs.11 and 12) and it is divided into two levels. The first level is what this paper will be focused on and it goes from the scene node to the configuration. The second level goes from the configuration to the image pixels and is referred to a previous work on primal sketch [5], [6].

We shall formulate a prior probability model on the parsing graphs and it integrates two interesting components studied widely in the literature. One is the stochastic context free

grammar (SCFG) for the hierarchical parsing graph and the other is a Markov model on the configurations (graphs) for local spatial relations. The likelihood adopts the primial sketch model. This yields a Bayesian posterior probability (or equivalently a description length) to be optimized.

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

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

Phase I is bottom-up detection. We first compute a number of edge segments from the input image, and estimate a number of vanish points (usually three) using a method studied in [19]. Thus the line segments are divided into three line sets as illustrated in Fig. 7. Then a number of rectangle hypotheses are generated in a method similar to RANSAC [3]. We draw two pairs of line segments from two out of the three line sets, and then evaluate it by the goodness of fit to a rectangle in the edge map. Each rectangle is a weighted hypothesis called a *particle*. An excessive number of rectangles are proposed as *bottom-up particles* which may overlap or conflict with each other. The bottom-up particles are candidates for activating the instantiation rule r_5 and are sorted in decreasing order by their weights. Each rule maintains its own candidate set as Fig.9 illustrates.

Phase II initializes the terminal nodes of the objective parsing graph in a greedy way. In each step, the algorithm picks the most promising bottom-up particle (with the heaviest weight) among all the candidate and accept it if it increases the Bayesian probability or reduces the description length. Then the weights of all the candidates who are overlapping or conflicting with this accepted rectangle are updated.

Phase III integrates top-down/bottom-up inference. Each rectangle in the current state of solution matches to a production rule (say rule r_5 for a single rectangle) with attributes

passed to the non-terminal node. These non-terminal nodes are in turn matched (often partially) to other production rules (rule r_2, r_3, r_4, r_6) which then generate a number of *top-down particles* for predictions whose weights are defined based on the prior probabilities. For example, two adjacent rectangles may activate the line rule r_2 (or a mesh r_3 or a cube r_6), which then generates a number of rectangles along the aligned axis. See Fig.10 for the top-down examples in the kitchen scene. Some of these top-down particles may have existed in the candidate sets of bottom-up particles. Such particles bear both the upward and downward arrows in Fig.9.

Like the Phase II, in each step the algorithm picks the most promising particle (with the heaviest weight) among all candidate sets. Then it is accepted if it increases the Bayesian probability or reduces the description length. Thus a new non-terminal node is added to the parsing graph. It means a recognition of sub-configuration and activates the following actions (i) Creating possibly new "top-down" particles and insert them in the lists. (ii) Re-weighting some particles in the candidate sets. (iii) Passing 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. 8 for the kitchen scene. For most images, the parsing graphs have 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 context information helps the algorithm detect weak rectangles which are missing in bottom-up detection. Some "illusory rectangles" could also be hallucinated, especially due to the line and mesh grammar rules.

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 [4], [17], [18] in the 1970-80s. Its applicability has been limited by two difficulties: (i) The primitive patterns (terminators in their description languages) could not be computed reliably

from real images. Thus their parsing algorithms were mostly applied to artificial image or diagrams. (ii) The study was mostly on string grammar and stochastic context free grammars (SCFG) in early work which are less expressive. In recent years, attribute grammars [1] and context sensitive graph grammars[12] have been developed in visual diagrams parsing, such as music notes and diagrams for relational databases. These grammars are much desired in vision. In the vision literature, graph grammars are mostly studied in binary shape recognition, such as the grammars for medial axis [21] and shock graph [13].

Detecting rectangular structures in images has been well studied in the vision literature, especially for detecting building roofs in aerial images. One type of methods [8], [9], [15] detect edge segments, line primitives and then group them into rectangles. The other types of methods [23], [19] use Hough Transforms on edge maps to detect rectangles globally. Our method is a combination of various techniques as we introduced in a previous subsection. Rectangle detection will not be a main technical focus of our paper. Although this paper is focused on rectangular scenes, we argue that the method could be extended to other classes of objects.

Our work is closely related to some previous work on object recognition and image parsing by data-driven Markov chain Monte Carlo (DDMCMC) [20], [16]. The common goal is to design effective algorithms by integrating bottom-up and top-down steps for inferring hierarchical image structures. In comparison to the previous work, this paper has the following novel aspects.

1. It extends the representation with an attribute grammar, which sets the ground for recognizing objects with structural variabilities.

2. It derives a general probability model that combines both SCFG (or Markov tree) model and the Markov random field models on graphs. This model is tightly integrated with the primal sketch models to yield a full generative representation from scene to pixels.

3. It extends the bottom-up and top-down parsing mechanism for grammar rules. This also includes a new algorithm for rectangle detection.

The remaining 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 scene. R is a set of production rules for spatial relationship. P is the probability for the grammar to be discussed in the next section.

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.

$$\mathcal{R} = \{r_1, r_2, ..., r_m\} \tag{2}$$

is the 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 $A_1, A_2 \in V_N$.

$$r: \ A \rightarrow (A_1, A_2). \tag{3}$$

The associated equations are constraints on the attributes.

$$g_i(X(A)) = f_i(X(A_1), X(A_2)), i = 1, 2, ..., n(r). \tag{4}$$

$g_i()$ 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 be simply an equivalence constraint (or assignment) for passing the information between nodes A and A_1 in either directions,

$$X_1 = X_{11}. \tag{5}$$

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

$$r: \ A \rightarrow a, \tag{6}$$

with constraints

$$g_i(X(A)) = f_i(x(a)), i = 1, 2, ..., n(r).$$

Definition 2: A *parsing graph* $\mathbf{G}$ is a tree structured representation expanded from a root node S by a sequence of production rules

$$\mathbf{G} = (\gamma_1, \gamma_2, ..., \gamma_k).$$

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

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

The attributes are denoted by $X(C)$.

$C = C(\mathbf{G})$ is generated deterministically by the parsing graph. In this paper, the rectangle configuration $C(\mathbf{G})$ is represented as a planar graph.

Definition 4: A *pattern* 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

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

By analogy to grammars in natural language, Σ is called the language (i.e. all valid sentences) for grammar $\mathcal{G}$.

Fig.2 illustrates two parsing graphs for two configurations of rectangle scenes using the production rules discussed next.

Most grammars are ambiguous, and thus a configuration C may have more than one parsing graphs. Our goal is to compute the most probable parsing graph according to the Bayesian formulation in the next section. In the following, we explain the primitive and production rule of our grammar for rectangle scenes.

B. A class of primitives — rectangles

In this paper, we use only one class of primitives – planar rectangle in 3-space illustrated in Fig. 3. The two pairs of parallel line segments in 3D may intersect at two vanishing points v_1, v_2 through projection. Therefore, the terminal set is

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

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.

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

V_T is a rather large set.

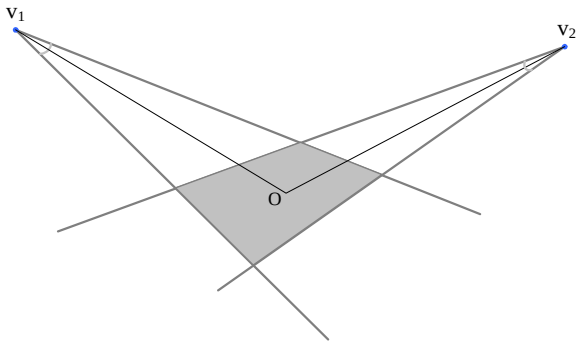


Fig. 3. A planar rectangle (shaded) is described by 8 variables. The two vanish points $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$ and four directions θ_1, θ_2, θ_3 and θ_4 at the two vanish points.

C. Six production rules

As a generic grammar for image interpretation, it has only two types of non-terminal nodes – the root node S for the scene and a node A for object.

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

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 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. 4.

The attribute $X(A) = (\ell(A), X_o(A))$ includes a label ℓ for the type of object (structure) represented by A, and $X_o(A)$ for other attributes, for example geometric properties and appearance.

$$\ell(A) \in \omega_\ell = \{\text{line}, \text{ mesh}, \text{ nest}, \text{ rect}, \text{ cube}\} \tag{11}$$

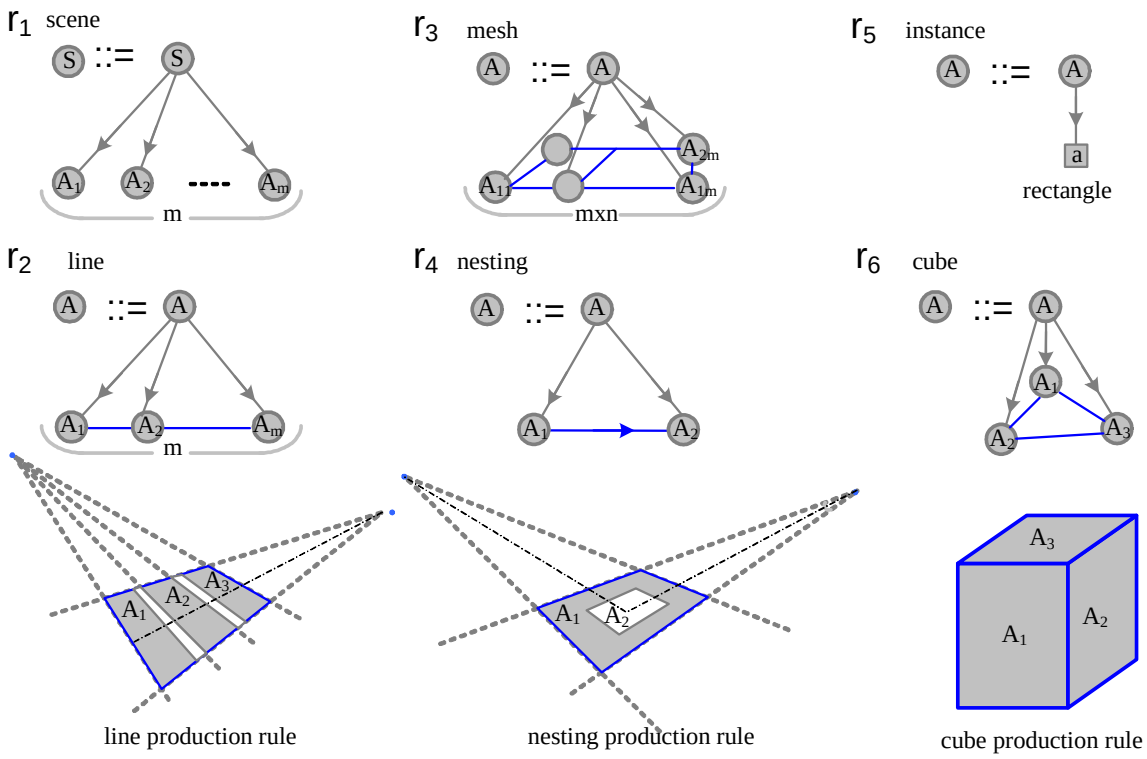


Fig. 4. Six attribute grammar rules. Attributes will be passed between a node to its children and the horizontal lines shows constraints on attributes. See text for explanation.

Thus accepting a production rule is to recognize one of the five patterns in inference. For object recognition tasks, we should further introduce different object categories. The other attributes $X_o(A)$ are different for different object types, so we have the space of A as the union of the five different subspaces.

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

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

1. For a line object of n rectangles,

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

The first eight parameters define the bounding box for the n rectangles, and the rest $2n-2$

orientations are used for the remaining directions specifying the individual rectangles in the object, as illustrated in Fig. 4 (row 3 left).

2. For a mesh object of $m \times n$ rectangles,

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

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 in the object.

3. For a nest object with 2 rectangles.

$$X_o(A) = (v_1, v_2, \theta_1, ..., \theta_4, \tau_1, ..., \tau_4)$$

4. For a cube object,

$$X_o(A) = (v_1, v_2, v_3, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6).$$

It has three vanish points and 3 pairs of orientation angles.

Remark 1: 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.

Remark 2: The rectangle elements in the above attribute description could be the bounding box for other objects to allow recursive applications of the rules.

Remarks 3: 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 of the 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 associate attributes to the the 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 directions in computation.

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

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

As illustrated in Fig. 4 (bottom-left corner), S is the bounding rectangle for A_1, A_2, A_3 and shares with them the two vanish 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

Let $\mathbf{I}$ be an input image, our goal is to compute a parsing graph $\mathbf{G}$ whose terminal nodes, i.e. all rectangles, compose a 2D planar configuration $C = C(\mathbf{G})$ where each rectangle is described as four line segments. Besides these line segments, there are other structures (object boundaries) in $\mathbf{I}$. We call them free sketches and denote by C_{free}. Together we have a primal sketch representation which generates the image $\mathbf{I}$.

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

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

$$\mathbf{G}^* = \arg\max p(\mathbf{I}|C_{\text{sk}})p(\mathbf{G})p(C_{\text{free}}). \tag{12}$$

The prior model $p(\mathbf{G})$ is the fifth component in the grammar $\mathcal{G}$. C_{free} is the nuisance variables whose prior model $p(C_{\text{free}})$ follows the primal sketch model. Now we present $p(\mathbf{G})$ and likelihood $p(\mathbf{I}|C_{\text{sk}})$ in the following two subsections.

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

A parsing graph $\mathbf{G}$ is a series of production rules $\gamma_1, ..., \gamma_k$. Let $\Delta_N(\mathbf{G})$ and $\Delta_T(\mathbf{G})$ be the sets of non-terminal nodes and terminal nodes respectively in the parsing graph $\mathbf{G}$. Each non-terminal node A in $\mathbf{G}$ has two types of variables $\ell(A)$ and $X_o(A)$. $\ell(A) \in \{\text{line, mesh, nest, rect, cube}\} = \Omega_\ell$ is a "switch" variable for selecting one of the 5 rules. We denote the probabilities for the five rules as q_ℓ which are summed to one: $\sum_{\ell \in \Omega_\ell} q_\ell = 1$. At the root S, we have a variable m for the number of objects and a prior model $p(m)$. Removing the attributes, we denote the parsing tree by

$$\mathbf{T} = (m, \{\ell(A) : A \in \Delta_N(\mathbf{G})\}).$$

The parsing graph is augmented with attributes

$$\mathbf{G} = (\mathbf{T}, X_o(\mathbf{T})),\ X_o(\mathbf{T}) = \{X_o(A) : A \in \Delta_N(\mathbf{G})\}.$$

The prior model for the parsing tree $\mathbf{T}$ follows the SCFG, and it can be written in an exponential form,

$$p(\mathbf{T}) = p(m) \prod_{A \in \Delta_N(\mathbf{G})} q_\ell = \frac{1}{Z_0} \exp\{-E_o(\mathbf{T})\} \tag{13}$$

$$E_o(\mathbf{T}) = \lambda_o m + \sum_{A \in \Delta_N(\mathbf{G})} \lambda_{\ell(A)}. \tag{14}$$

In the above equation, $\lambda_o > 0$ is a constant penalizing the complexity m, $\lambda_\ell = -\log q_\ell$.

For a given tree $\mathbf{T}$ and therefore the configuration C, we introduce soft constraints on the attributes for the spatial relationships. This leads to some typical Markov model through a

maximum entropy principle,

$$p(X_o(\mathbf{T})) = \frac{1}{Z(\mathbf{T})}\exp\{-E(X_o(\mathbf{T}))\} \quad (15)$$

$$E(X_o(\mathbf{T})) = \sum_{A\in\Delta_N(\mathbf{G})}[\phi_{\ell(A)}(X_o(A_i)) + \sum_{B\in\text{Child}(A)}\psi_\ell(A)(X_o(A), X_o(B))]. \quad (16)$$

The potential functions $\phi()$ and $\psi()$ usually takes quadratic forms to enforce some regularities, such as the aligned rectangles in a group are evenly spaced. So we have the following potential functions for the rectangle group, A, in Figure 4 (row 3 left):

$$\phi_{line}(X_o(A)) = (d(\tau_1,\tau_2) - d(\tau_3,\tau_4))^2,$$

$$\sum_{i=1}^{3}\psi_{line}(X_o(A), A_i) = \sum_{i=1}^{3}(|\theta_{3i}-\theta_3|^2 + |\theta_{4i}-\theta_4|^2),$$

where, $d()$ is a function to compute the distance between two neighboring rectangles in the line group. In addition, we may also relax the hard constraints for some attributes.

Integrating the SCFG and Markov model together, we have a joint probability on the parsing graphs.

$$p(\mathbf{G}) = \frac{1}{Z(\mathcal{G})}\exp\{-E_o(\mathbf{T}) - E(X_o(\mathbf{T})\}, \quad (17)$$

where the two energy terms are defined in eqns.(14) and (16) respectively. Note that $p(\mathbf{G})$ is not simply the product of $p(\mathbf{T})$ and $p(X_o(\mathbf{T}))$ as it may look like. The partition function $Z(\mathbf{T})$ in $p(X_o(\mathbf{T}))$ is a function of $\mathbf{T}$, and $Z(\mathcal{G}) \neq Z_o \cdot Z(\mathbf{T})$. The new partition function $Z(\mathcal{G})$ depends only on the grammar and is summed over all possible parsing graphs in $\mathcal{G}$.

$$Z(\mathcal{G}) = \sum_{\mathbf{G}}\exp\{-E_o(\mathbf{T}) - E(X_o(\mathbf{T})\}.$$

This is a crucial property of this prior model. Because of a common Z for different parsing graphs, we no longer need to worry about the partition function or ratio when we switch between different configurations in inference.

One interpretation for $p(\mathbf{G})$ is to minimize a Kullback-Leibler divergence between $p(\mathbf{G})$ and the SCFG model $p(\mathbf{T})$, i.e.

$$p^*(\mathbf{G}) = \arg\min \sum_{\mathbf{G}} p(\mathbf{G}) \log \frac{p(\mathbf{G})}{p(\mathbf{T})} \tag{18}$$

under the same soft constraints which were used for deriving the Markov model $p(X_o(\mathbf{T}))$ through maximum entropy.

Solving the constrained optimization problem above by Lagrange multipliers, we get $p(\mathbf{G})$ in equation (17). This prior model integrates the SCFG model and Markov random field model. There is a similar formulation to the constrained stochastic models in natural language [11] where people integrates SCFG with bi-gram statistics. A more advanced form is derived in a most recent work [2] where we may even allow the neighborhoods of the MRF to be variables.

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

We adopt the primal sketch model for $p(\mathbf{I}|C_{\rm sk})$, and refer to two previous papers [5], [6] for this model and algorithm. The reconstruction (synthesis) of images using a configuration is shown in the experiment section (See Figs.11 and 12). In the following we briefly introduce the model for the paper to be self-contained.

$C(\mathbf{G})$ 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.

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

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 $\Lambda_{\rm sk}$ is divided into N image primitives (patches) of 5×7 pixels along these segments.

$$\Lambda_{\rm sk} = \cup_{k=1}^{N} \Lambda_{{\rm sk},k}. \tag{19}$$

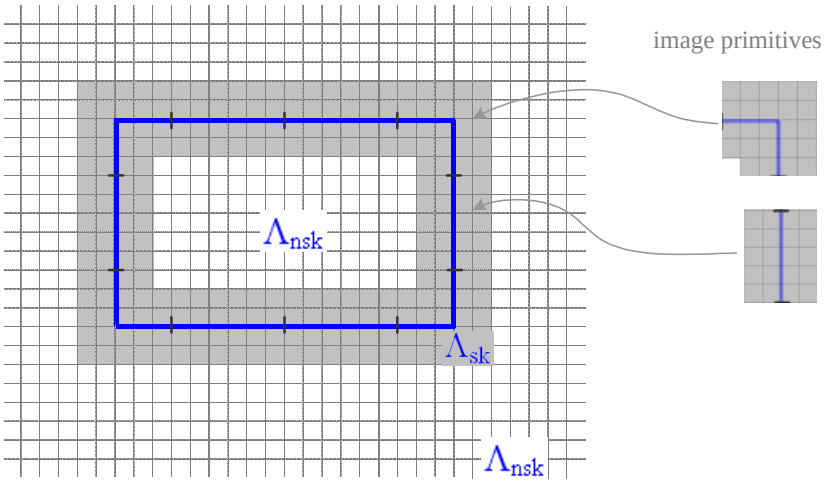


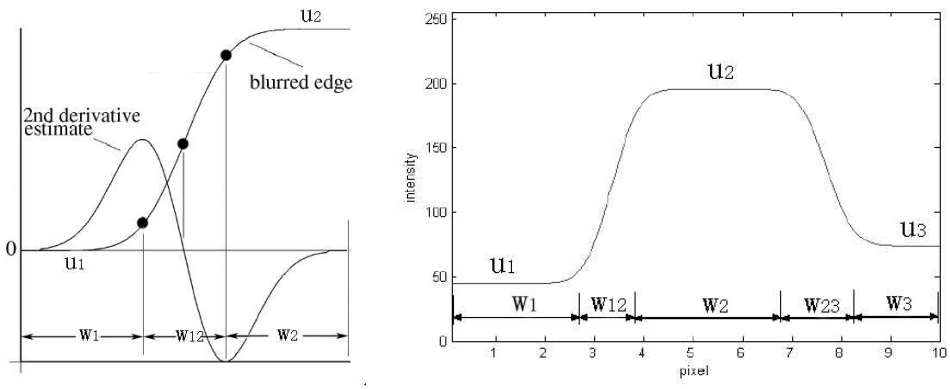
Fig. 5. Partition of image lattice Λ into two parts. The shaded pixels $\Lambda_{\rm sk}$ around the rectangles and the remaining part $\Lambda_{\rm nsk}$. The rectangles are divided into small edge and corner segments. Therefore $\Lambda_{\rm sk}$ is divided into many image primitives.

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,

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

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.

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



(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.

ridge profile is represented by seven parameters: $\Theta = (u_1, u_2, u_3, w_1, w_{12}, w_2, w_{23}, w_3)$. More detailed description is referred to [6]. Using the above edge/ridge model, the 1D intensity function of the profile for the rectangle boundaries could be fully obtained.

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

$$\mathbf{I}(x,y) = B_{t_k}(x - x_k, y - y_k; \tau_k, \sigma_k, \Theta_k) + n(x,y),\ (x,y) \in \Lambda_{\text{sk,k}},\ \forall k = 1, 2, ..., N. \qquad (20)$$

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 it is mentioned previously, rectangles are only part of the sketchable structures in the images, though they are the most popular structures in the 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 textures without prominent structures and the textures 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,

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

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

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

The probability model for the textures are the FRAME model [22] with the Lagrange parameters (vector) β_m as the learned potentials. These textures use the sketchable part $\Lambda_{\rm sk}$ as the boundary condition in calculating the filter responses.

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

$$p(\mathbf{I}|C_{\rm sk}) = \frac{1}{Z}\exp\{-\sum_{k=1}^{N}\sum_{(x,y)\in\Lambda_{\rm sk,k}} \frac{(\mathbf{I}(x,y) - B_k(x,y))^2}{2\sigma_o^2} - \sum_{m=1}^{M}\langle\beta_m, h(\mathbf{I}_{\Lambda_{\rm nsk,m}})\rangle\} \tag{21}$$

The above likelihood is based on the concept of primitives not rectangles, therefore the recognition of rectangle or bigger structures (cube, mesh etc.) only affects the likelihood locally. In other words, our parsing graph is build on the primal sketch representation. This is important in designing 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, such as 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 parsing graph $\mathbf{G}$ by maximizing the posterior probability formulated in the previous section. The computation algorithm should achieve two difficult goals: (i) Constructing the parsing 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 parsing graph.

There are several ways to infer the optimal parsing graph and Data-Drive Markov Chain Monte Carlo (DDMCMC) has been used in [20], [16]. In this paper, our domain is limited to rectangle scenes, and the parsing graph is not too big (usually ~ 20 nodes). Thus, the best first search algorithm in artificial intelligence can be directly applied to compute the parsing 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 parsing graph. The algorithm in phase II is very much like matching pursuit [10]. Finally phase III constructs the parsing 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_{sk} using the likelihood model in eqn. 21. We segment each long contour into a number of n straight line segments by polygon-approximation. In man-made scenes, the majority line segments are aligned with one of three principal di-

rections and each group of parallel lines intersect at a vanish point due to perspective projection. We define all lines ending at a vanish point to be a parallel line group. A rectangle has two pairs of parallel lines which belongs to two separate parallel line groups. We run the vanishing point estimation algorithm [19] to group all the line segments into three groups corresponding to the principal directions. With these three line groups, we generate the rectangle hypotheses like RANSAC [3]. 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 bad fitted hypothesis.

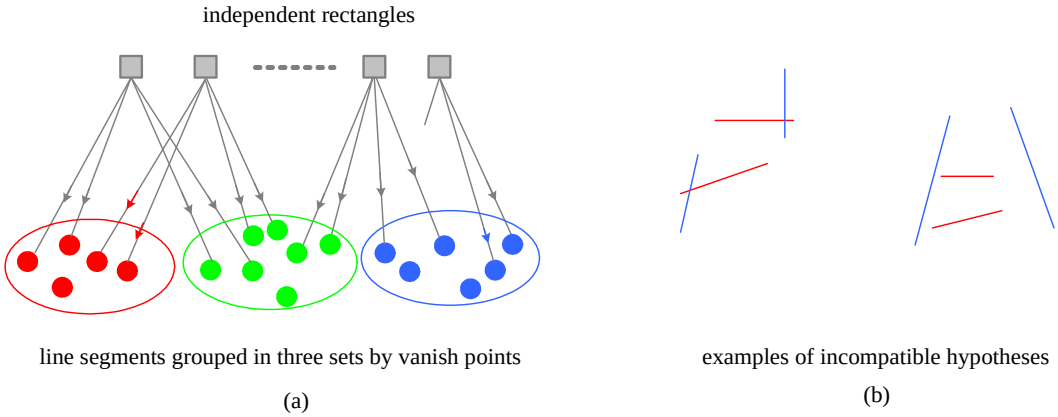


Fig. 7. Bottom-up rectangle detection. The n line segments are grouped into three sets according to their vanish points. Each rectangle consists of 2 pairs of nearly parallel line segments (represented by a small circle).

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 down graded or eliminated.

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

We adopt a simplified model which use 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 [10].

Rectangle Pursuit: initialize the terminal nodes of $\mathbf{G}$

Input candidate set $\Phi = \{\pi_1, \pi_2, ..., \pi_M\}$ from phase I.

1. Initialize parsing graph $\mathbf{G} \leftarrow \emptyset$, $m = 0$.
2. Compute weight ω_i for $\pi_i \in \Phi$, $i = 1, ..., |\Phi|$, thus obtain

   $\{(\pi_i, \omega_i) : i = 1, 2, ..., |\Phi|\}$.
3. Select a rectangle π_+ with highest weight in Φ,

   $\omega(\pi_+) = \max\{\omega(\pi) : \pi \in \Phi\}$.
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 π overlaps with π_+
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 the remaining are missing, which are the missing edges or gaps between primitives in C_{free}. Thus we define two sets

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

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. (12),

$$\omega(b_+) = \log[\frac{p(\mathbf{I}|C'_{\text{sk}})}{p(\mathbf{I}|C_{\text{sk}})} \cdot \frac{p(G')}{p(G)} \cdot \frac{p(C'_{\text{free}})}{p(C_{\text{free}})}] \tag{22}$$

Choosing a rectangle π_+ with largest weight $\omega(\pi_+) > 0$ is to increase 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 on a small domain covered by the primitives in $L_{\text{off}}(\pi_+)$. Pixels in this domain belongs to Λ_{nsk} before and are in Λ_{rmsk} after adding π_+. The likelihood does not change for any other pixels. The second term $\log \frac{p(G')}{p(G)} = -\lambda_o$ penalizes the model complexity of rectangles (see eqn 14). 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 22. 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. Such weight update was used in wavelet pursuit where it is interpreted as "lateral inhibition" in neuroscience. It is a typical step in generative models.

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

The algorithm for constructing the parsing 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 be missed in bottom-up detection. We start with an illustration of the algorithm for the kitchen scene.

In Fig. 8, 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. 8 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. 8. 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. 8 with downward

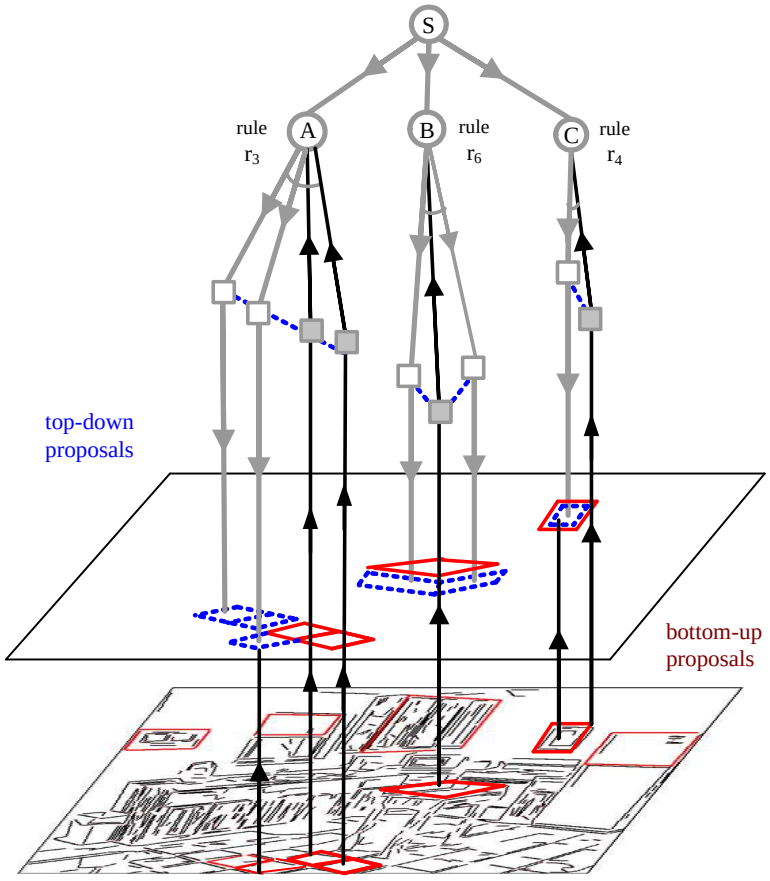


Fig. 8. Bottom-up detection of rectangles (red) instantiates some non-terminal nodes (rule r_5) in upward arrows. They in turn activate graph grammar rules r_3, r_6, r_4 for grouping larger structures in nodes A, B, C respectively. These rules generate top-down prediction of rectangles (blue). The predictions are validated from the image.

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 the bottom proposal of Π. Such nodes bear both upward and downward arrows.

Fig. 9 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

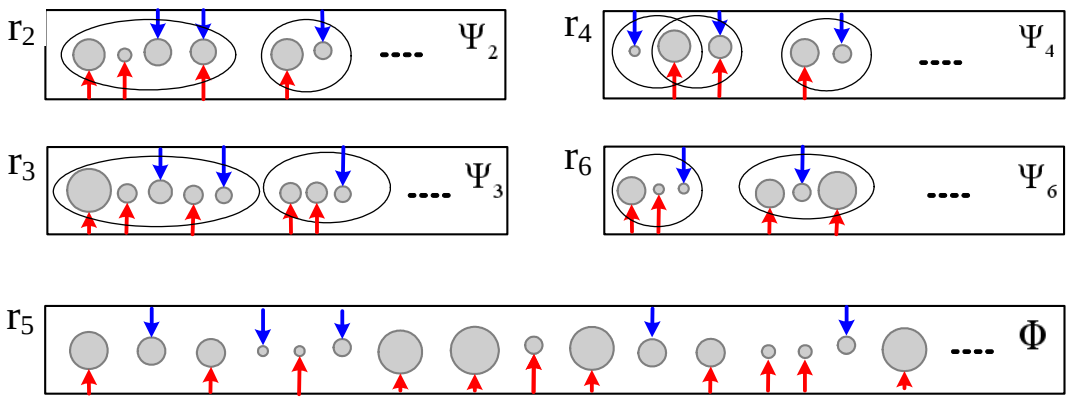


Fig. 9. Four sets of proposed candidates $\Psi_2, \Psi_3, \Psi_4, \Psi_6$ for production rules r_2, r_3, r_4, r_6 respectively and the candidate set Φ for the instantiation rule r_5. Each circle represents a rectangle π or a bounding box of for an non-terminal node. The size of the circle represents its weight $\omega(\pi)$. Each ellipses 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.

Φ by the log-posterior probability ratio.

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

$\Phi = \{(\pi_i, \omega_i) : i = 1, 2, ..., M\}$ 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 nodes A in graph $\mathbf{G}$ may participate in more than one group candidates Π's, just as a line segment may be part of multiple rectangle candidates π's. This creates overlaps between the candidates and need to be resolved in a generative model.

At each step the parsing algorithm will choose one candidate with largest weight from the five candidate sets and add a new non-terminal node to the parsing graph. If the candidate is $\pi \in \phi$, it means accepting a new rectangle. Otherwise the candidate is a larger structure Π, then it 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 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 in the following.

The algorithm for constructing the parsing 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 parsing graph $\mathbf{G}$ $\mathbf{G} \leftarrow \mathbf{G} \cup \{A_+\}$.

4. Set the parent node of A_+ to a non-terminal node which proposed A_+ in top-down

   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 Φ 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 Π overlap with A_+.

9. Repeat 2-8 until ω_+ is smaller than a threshold δ_1.

Output a parsing graph $\mathbf{G}$.

Fig. 10 shows a snapshot of one iteration of the algorithm on the kitchen scene. Fig. 10.(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 parsing 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 $B_{(j)}, j = 1, 2, ..., k_\pi$ are shown in blue.

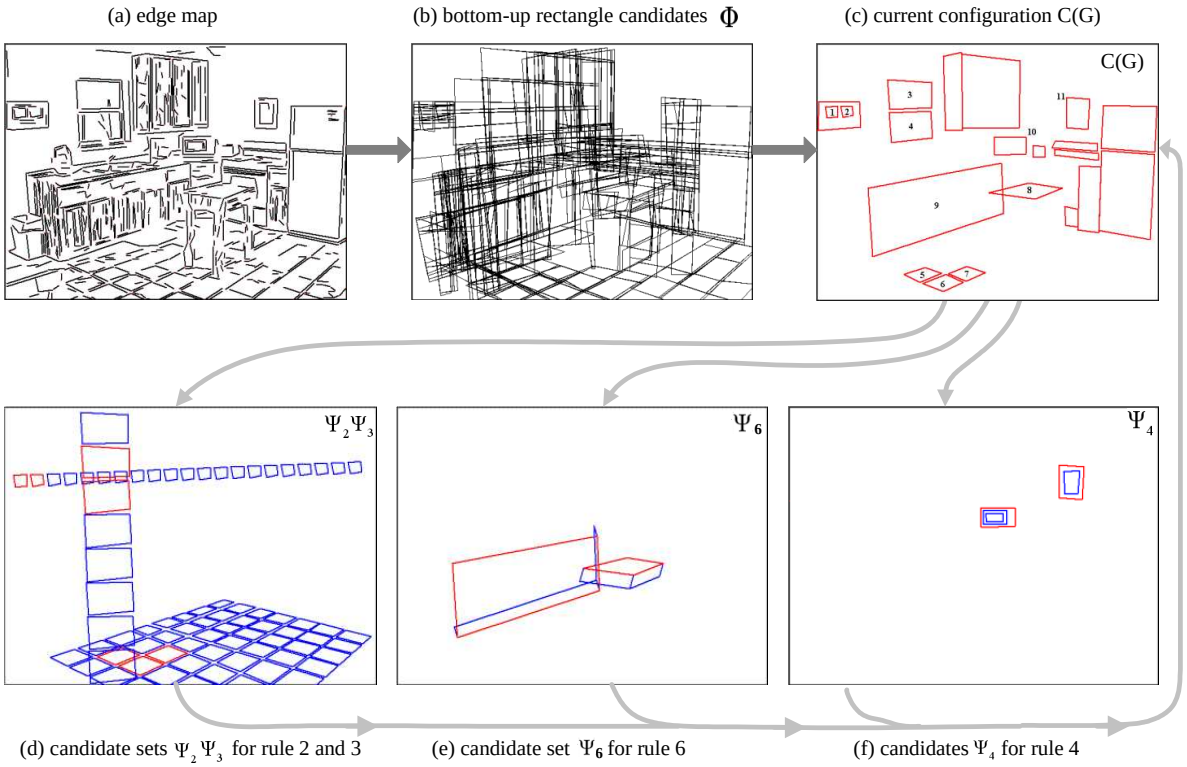


Fig. 10. A kitchen scene as running example. (a) is the edge map, (b) is a subset of Φ for rectangle candidates detected in phase I. We show a subset for clarity. (c) is the configuration $C(\mathbf{G})$ with a number of accepted rectangles in phase II. (d-f) are candidates in $\Psi_2, \Psi_3, \Psi_4, \Psi_6$ respectively. They are proposed based on the current node in $\mathbf{G}$ (i.e. shown in (b)).

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 vanish points and 4 orientation. We can simply test any two node 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.10.(c), it proposes two rectangles by the cube rule r_6 in Fig.10.(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 parsing graph is in the spirit similar to the DDMCMC algorithm[20], [16], 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 likely get local optimal solutions.

V. Experiments

We test our algorithm on a number of scenes with rectangle structures and six results are shown in Figs.11 and 12. In these two figures, the first row shows the input images, the second row shows the edge detection results, the third row are the detected and grouped rectangles in the final configurations, 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. Firstly, for the sketchable parts, we reconstruct by the image primitives after fitting some parameters for the intensity profiles. For the rest area Λ_{nsk}, we follow [5] to divide Λ_{nsk} into homogeneous texture regions by k-mean clustering method and then synthesizing each texture region by sampling the Julesz ensemble so that the synthesize image has histograms matched with 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 totally 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 shows that high level knowledge introduced by the graph grammar largely improves the results. For example, in the building scene at the third column in Fig. 11, 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, the Phase I is the most time-consuming stage and takes about 2 minutes on a 640x480 image since we have to try a lot of combinations to generate all the rectangle proposals and build up their occlusion relations. The Phase II and III are very fast and take about 1 minute totally.

VI. Discussion

In this paper, we study an attributed grammar for image parsing on man-made scene. The paper makes two main contributions to the vision literature. Firstly, it uses attributed grammar to incorporate prior knowledge. With more powerful class of grammars, we should rejuvenate the syntactic pattern recognition research originally pursued in the 70s-80s [4]. Such grammar representations are long desired for high level vision, especially scene understanding and parsing. Secondly, it integrates top-down/bottom-up procedure for computing the parsing graph with grammars.

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, especially functional objects where objects within each category exhibit wide range structural variabilities. The extended grammar will have many more production rules. (ii) At the current work, we manually set some probabilities and parameters in the energy function. These parameters should be learned automatically when we have large number of manually parsed training examples, i.e. through

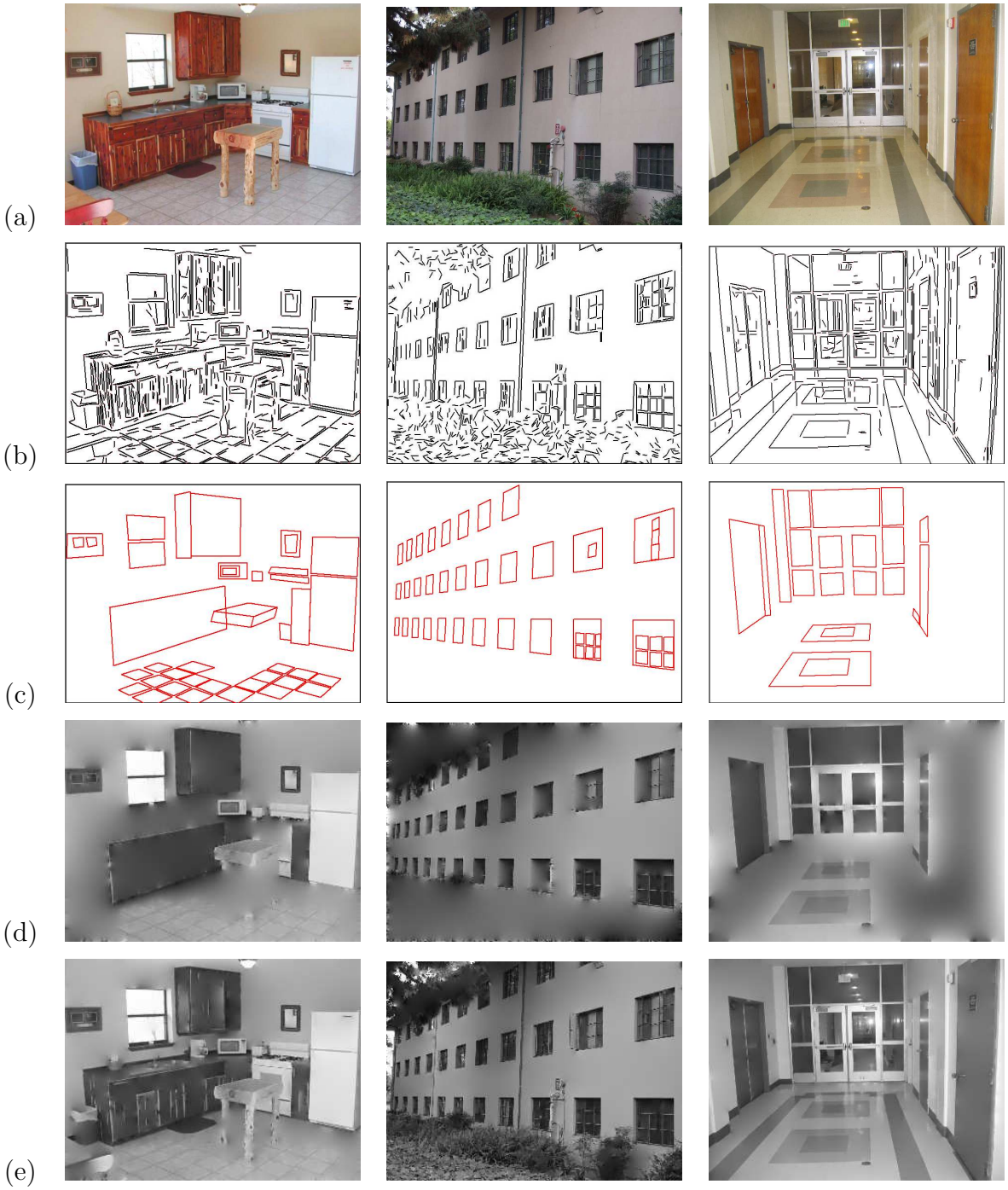


Fig. 11. Some experimental results. (a) Input image. (b) Edge map. (c) Computed rectangle configurations. (d) Reconstructed image from the primal sketch model using the rectangle configurations only. (e) Reconstructed images after adding some background sketches to the configurations.

supervised learning. We are currently collecting a large manually parsed image dataset for learning grammars. In experiments, we observe that the stoping threshold Δ_0 and Δ_1 in

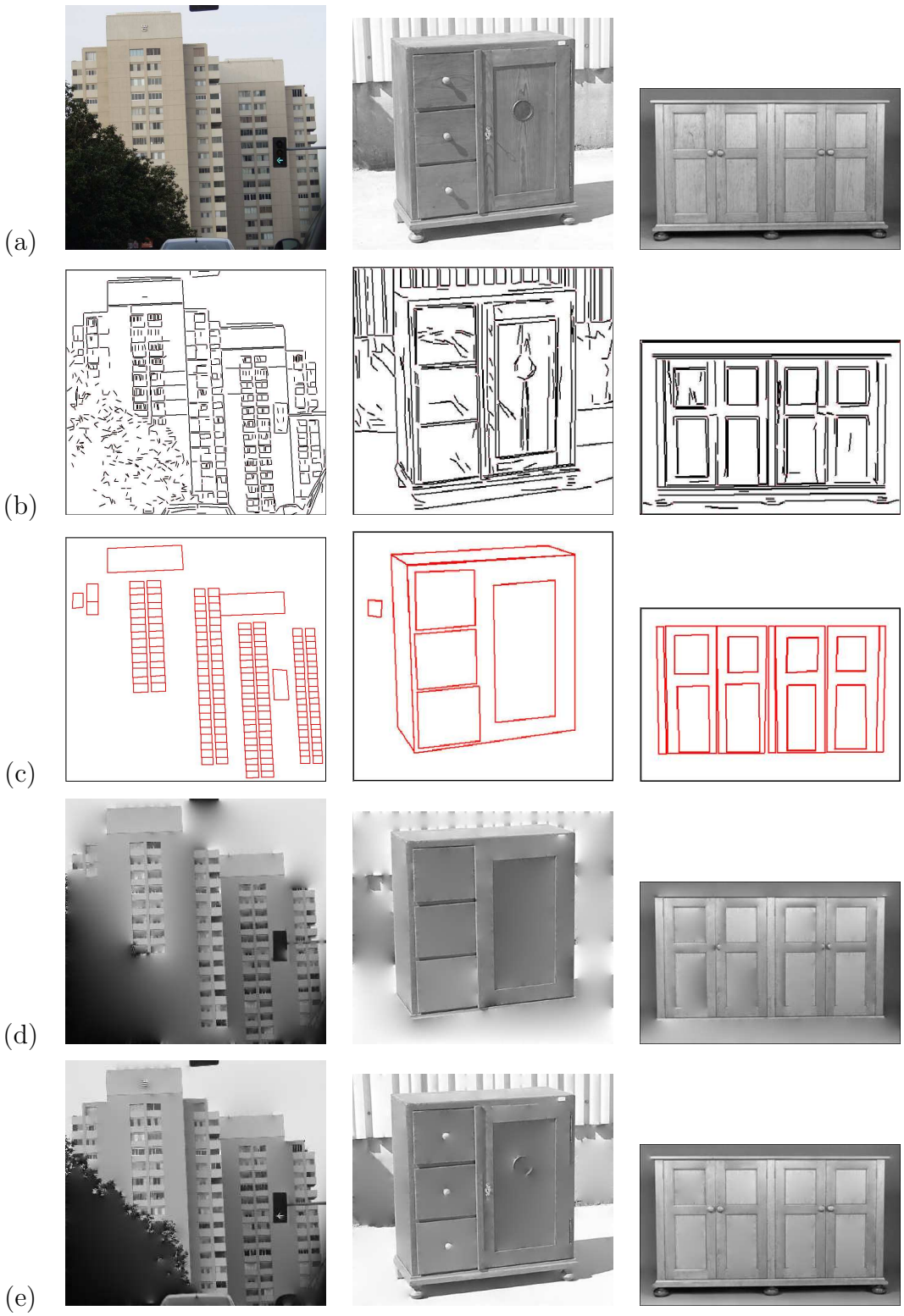


Fig. 12. More experimental results. (a) Input image. (b) Edge map. (c) Computed rectangle configurations. (d) Reconstructed image from the primal sketch model using the rectangle configurations only. (e) Reconstructed images after adding some background sketches to the configurations.

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. They are manually tuned in the experiment.

Acknowledgements

This work was supported in part by NSF grant IIS-0413214 and an ONR grant N00014-05-01-0543.

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] H. Chen, Z.J. Xu, and S.C. Zhu, "Composite templates for cloth modeling and sketching," *Technical Report*, Dept. of Statistics, UCLA, 2005.

[3] 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.

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

[5] 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.

[6] C.E. Guo, S.C. Zhu and Y.N. Wu, "Primal sketch: integrating texture and structure", *Computer Vision and Image Understanding* the Special Issue on Generative Model Based Vision, Accepted 2005.

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

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

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

[10] S. Mallat, and Z. Zhang, "Matching pursuit with time-frequency dictionaries", *IEEE Trans. on Signal Processing*, vol. 41, no. 12, 3397-3415, 1993.

[11] K. Mark and M. Miller and U. Grenander, "Constrained stochastic language models", *Image Models*

and Their Speech Model Cousins (S. E. Levinson and L. Shepp, eds, Minneapolis, IMA Volumes in Mathematics and its Applications, 1994.

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

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

[14] J.M. Siskind, J. Sherman, I. Pollak, M.P. Harper, C.A. Bouman, "Spatial random tree grammars for modeling hierarchal structure in images", *Technical Report*, 2004

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

[21] S.C. Zhu and A.L. Yuille, "FORMS: A Flexible Object Recognition and Modeling System," 20(3), pp.187-212, 1996.

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

[23] 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.