Bottom-up/Top-Down Image Parsing by Attribute Graph Grammar

Feng Han and Song-Chun Zhu
Departments of Computer Science and Statistics
University of California, Los Angeles
Los Angeles, CA 90095
hanf@cs.ucla.edu, sczhu@stat.ucla.edu

Abstract

In this paper, we present an attribute graph grammar for image parsing on scenes with man-made objects, such as buildings, hallways, kitchens, and living rooms. We choose one class of primitives – 3D planar rectangles projected on images, and six graph grammar production rules. Each production rule not only expands a node into its components, but also includes a number of equations that constrain the attributes of a parent node and those of its children. Thus our graph grammar is context sensitive. The grammar rules are used recursively to produce a large number of objects and patterns in images and thus the whole graph grammar is a type of generative model. The inference algorithm integrates bottom-up rectangle detection which activates top-down prediction using the grammar rules. The final results are validated in a Bayesian framework. The output of the inference is a hierarchical parsing graph with objects, surfaces, rectangles, and their spatial relations. In the inference, the acceptance of a grammar rule means a recognition of an object, and actions are taken to pass the 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.

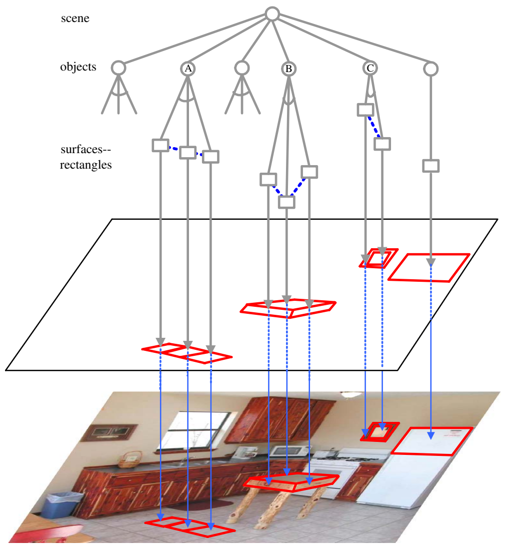


Figure 1. An example of generative scene modeling with graph grammar. A kitchen scene consists of a number of objects and each object is generated by a number of rectangles (transformed) using one of the four graph grammar rules. The grammar rules specify the spatial relations between the constituent rectangles.

1. Introduction

In real world images, especially man-made scenes, such as buildings, offices, and living spaces, a large number of complex patterns (objects) are composed of a small set of primitives using few operational relations. This is very similar to language where a huge set of sentences can be generated with a relatively small vocabulary and a few grammar rules in a hierarchy from words, phrases, clauses, and to sentences. The objective of image parsing is to decompose an image into its constituent components (objects, parts, surfaces, primitives) in a hierarchical structure represented by a parsing graph. Fig. 1 shows a parsing graph for a kitchen scene using the rectangle primitives. We only show a subset of the rectangles for clarity. Its vertical edges show the decomposition of scene and objects, and its horizontal links specify the spatial relations between objects and components. Thus a parsing graph is more general than a parsing tree in language.

In the literature, the study of syntactic pattern recognition was pioneered by Fu et al [4, 8] in the 1970-80s. Unfortunately, its applicability was very much limited by two difficulties. (i) The primitive patterns (terminators in their visual languages) could not be computed reliably from real images. Thus their parsing algorithms achieved little success in real images. (ii) The string grammar (finite state automation) and stochastic context free grammars (SCFG) in early work are not strong enough to express complex visual patterns.

In recent years, two new developments are encouraging and thus motivate our research. Firstly, more powerful grammars, such as attribute graph grammars [1] and context sensitive graph grammars [6] have been developed in visual diagrams parsing, such as music notes and diagrams for relational databases. These grammars allow attributes to be passed among the nodes in the parsing process in three directions: top-down, bottom-up, and horizontal context. Such mechanisms are much desired in visual inference. Secondly, more powerful inference tools, such as the data-driven Markov chain Monte Carlo (DDMCMC) [10, 7], were developed for passing the hypotheses up and down the parsing tree (in SCFG) probabilistically in a principled way. This makes it possible to apply the grammar parsing algorithms to real images.

In this paper, we study an attribute graph grammar for image parsing and focus on man-made scenes, such as buildings, hallways, kitchens, living rooms, etc. We choose one primitive (terminator) – 3D planar rectangles projected on images as shown in Fig. 2. We choose six graph grammar production rules as displayed in Fig. 3. 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 both attributed and context sensitive. These rules can be used recursively to generate a large set of complex configurations – called the "language" of the grammar.

The paper is mainly focused on the top-down and bottom-up inference. Our algorithm proceeds in two phases.

Phase I is bottom-up computation. 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 [9]. Thus the line segments are divided into three line sets illustrated in Fig. 4. Then we generate a number of rectangle hypotheses in a method similar to RANSAC [3], which are weighted by their goodness of fitting to the input image. Thus an excessive number of rectangles are proposed as **bottom-up particles**, and some of them may overlap or conflict with each other. These bottom-up particles are tested and pursued one by one in a procedure similar to the matching pursuit algorithm [5] used in wavelets. With a high stopping threshold, we obtain a number of "strong" hypotheses (rectangles) as initialization.

Phase II integrates top-down/bottom-up inference. Each rectangle in the current state of solution matches to a production rule (rule 5) with attributes passed to the non-terminal node. These non-terminal nodes are in turn matched to other production rules (rule 2,3,4,6) which then propose a number of **top-down particles**. Some of these top-down particles may have existed in the bottom-up particles with weak scores. Then each rule is evaluated in a Bayesian framework. The acceptance of a production rule means a recognition of an object, and actions are taken to pass the 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.

In most of the images, the parsing graph has 2-3 layers with fewer than 20 nodes, so the computation can be done by AI search algorithms, such as best first search. The top-down and bottom-up computing is illustrated in Fig. 5 for the kitchen scene. 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.

Although this paper is focused on rectangular scenes, we argue that the method could be extended to other classes of objects. To our best knowledge, this paper is perhaps the first work applying attributed context sensitive graph grammars to parse real world scenes.

The remaining of the paper is organized as follows. We first present the graph grammar representation in Section 2, and then the top-down/bottom-up inference in Section 3. We show some results in Section 4 and conclude the paper with a discussion of future work in Section 5.

2 Graph grammar representation

We briefly introduce the attribute graph grammar, then present our primitives and grammar rules for the rectangle scene, and formulate the problem in Bayesian inference.

2.1 Attribute graph grammar

We denote an attribute graph grammar by a 4-tuple

$$\mathcal{G} = (V_N, V_t, \mathcal{R}, \Sigma).$$

V_N is a set of non-terminal nodes, often denoted by capital letters A, A_1, A_2, etc. V_t is a set of terminal nodes denoted by lower case letters a, b, c, a_1, a_2, etc. Both non-terminal and terminal nodes have a vector of attributes denoted by

$X(A)$ and $x(a)$ respectively. Within the attribute space, the terminal and non-terminal set could be very large. $\mathcal{R}$ is a set of production rules

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

Each production rule specifies how a non-terminal node in V_N is expanded in $V_N \cup V_T$, and is associated with a number of constraint equations. For example, the following is a rule that expands one node A into two nodes $A_1, A_2 \in V_N$.

$$r: \ A \to (A_1, A_2). \tag{1}$$

The associated equations are constraints on the attributes.

$$g_i(X(A)) = f_i(X(A_1), X(A_2)), i = 1, 2, ..., n(r).$$

$g_i()$ and $f_i()$ are usually projection functions that take some elements from the attribute vectors. For example, 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,

$$X_1 = X_{11}.$$

Other rules may instantiate a non-terminal node to a terminal node

$$r: \ A \to a,$$

with constraints

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

In a parsing process, sometimes we know the attributes of a child node X_{11} and then pass it to X_1 as 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".

Finally Σ is a set of instances (or configurations) that can be produced by the production rules starting from a root node $S \in V_N$,

$$\Sigma(\mathcal{G}) = \{(C, X(C)) : S \overset{r_{c(1)}, \ldots, r_{c(k)}}{\longrightarrow} C\}.$$

Each configuration C has attributes $X(C)$ and corresponds to a specific scene structure. The grammar rules are applied in a certain order and form the parsing graph $\mathbf{G}$,

$$\mathbf{G}(C) = (r_{c(1)}, ..., r_{c(k)}).$$

Fig.1 illustrates a parsing graph for a kitchen scene. Most grammars are ambiguous, and thus a configuration C may have more than one parsing graphs. In such case, we should choose the shortest or most probable one. We will discuss this in the Bayesian formulation below. In the following, we explain the primitive and production rule of our grammar for rectangle scenes.

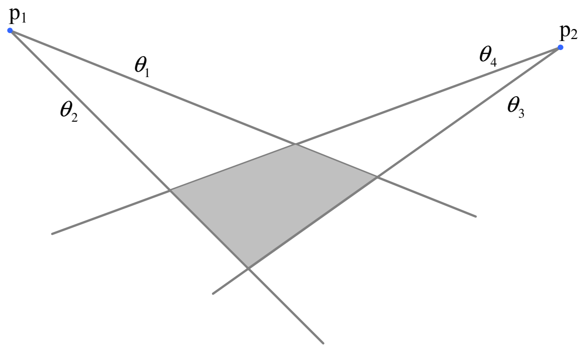


Figure 2. A planar rectangle (ABCD) is described by 8 variables. The two vanish points $p_1(x, y)$ and $p_2(x, y)$ and four directions θ_1, θ_2, θ_3 and θ_4 for the four boundaries.

2.2 A primitive — rectangles

In this paper, we use only one class of primitives – 3D planar rectangles projected on images illustrated in Fig. 2. The two pairs of parallel line segments in 3D may form two vanishing points p_1, p_2 after projected to the image plane.

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

There are many equivalent ways to define the attributes $x(a)$ for a rectangle. We choose the two vanishing points $p_1(x, y)$ and $p_2(x, y)$, and the four directions, $\theta_1, \ \theta_2, \ \theta_3, \ \theta_4$, of the four lines corresponding to the four boundaries of the rectangle. Thus a rectangle is described by 8-variables,

$$x(a) = (p_1(x, y), p_2(x, y), \theta_1, \theta_2, \theta_3, \theta_4).$$

We choose the attributes in a way that they can be shared with the non-terminal nodes. V_T is a rather large set because of the large space of rectangles denoted by Ω_a. One may denote some special rectangles with smaller number of variables.

2.3 Six production rules

In our graph grammar for rectangle scenes, there are only two types of non-terminal nodes – the root node S for the scene and a node A for objects.

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

The scene node generates n independent objects. It may also include a label ℓ for the scene category, but we do not deal with scene classification in this paper. The object node A can be instantiated (assigned) to a rectangle (rule r_5), otherwise it will be used recursively by other four production

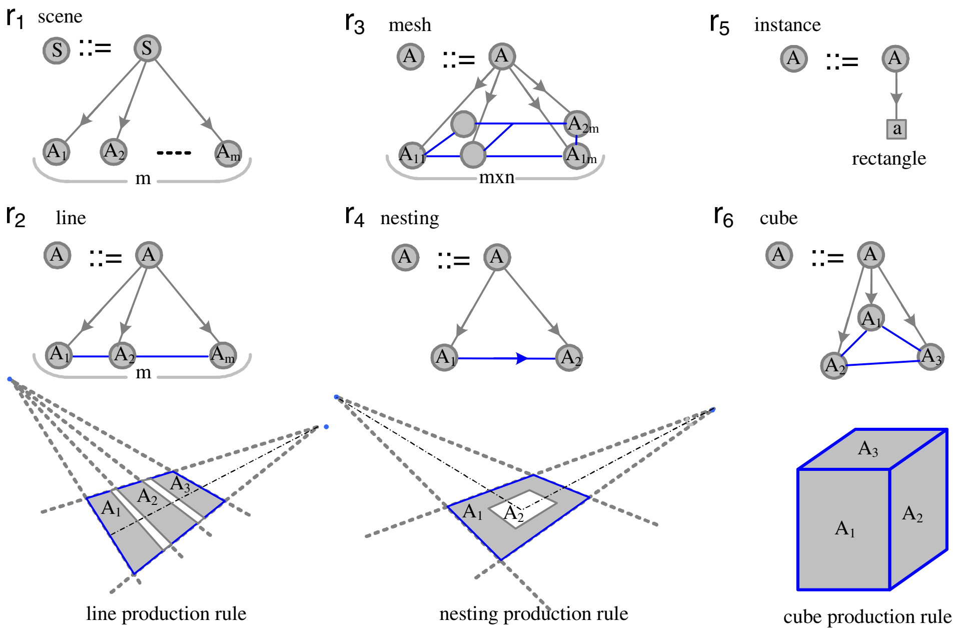


Figure 3. Six attribute grammar rules that can be used recursively. Attributes will be passed between a node to its children and the horizontal lines shows constraints on attributes. See text for explanation.

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,

$$\mathcal{R} = \{r_1, r_2, r_3, r_4, r_5, r_6\}.$$

Each of the production rules produces a structure and is associated with a number of constraints on attributes. 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.

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

Thus accepting a production rule is recognizing one of the five patterns in inference. 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.

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

In the following, we explain some of the production rules. The simplest rule is r_5.

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

This rule terminates the recursion and assigns a rectangle and its attributes to the non-terminal node A. Therefore $X_o(A) = x(a)$ is the constraint equation.

The second simplest one is the line production rule r_2 we choose $m = 3$ for simplicity.

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

As illustrated in Fig. 3 (bottom-left corner), $X_o(A)$ has 8 attributes describing the bounding box (blue) of A_1, A_2, A_3. Given $X_o(A)$, the three rectangles 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. As one can see, the use of grammar rule reduces the description length.

For space limit, we omit the tedious specification of the constraint equations for all production rules.

2.4 Bayesian formulation and Probability models

Let $\mathbf{I}$ be an input image, our goal is to compute parsing graph $\mathbf{G}$ as our recognition of the scene using grammar $\mathcal{G}$. The terminal nodes in $\mathbf{G}$, i.e. all rectangles, compose a 2D configuration $C = C(G)$ – a reduced representation of $\mathbf{G}$.

In a Bayesian framework this is to optimize a posterior probability,

$$\mathbf{G}^* = \arg\max p(\mathbf{I}|C(\mathbf{G}))p(\mathbf{G}). \qquad (2)$$

Let $\Delta_N(\mathbf{G})$ and $\Delta_T(\mathbf{G})$ be the set of non-terminal nodes and terminal nodes in the parsing graph $\mathbf{G}$ respectively. The prior probability is the product of the probabilities for all the rules it used.

$$p(\mathbf{G}) = \prod_{A\in\Delta_N(\mathbf{G})} p(X(A_1), ..., X(A_m)|X(A)). \quad (3)$$

One may easily calculate these probabilities by computing the coding length L

$$\log_2 p(X(A_1), ..., X(A_m)|X(A)) = L(X(A_1), ..., X(A_m)|X(A)).$$

L is the number of bits coding the attributes of the children $A_1, ...A_m$ given the attributes of their parent A. In our experiments, we compute the degree-of-freedom and calculate the bits after discretization of the attribute spaces.

$C(\mathbf{G})$ is a set of rectangles produced by the parsing graph. For example, Fig. 8 shows the computed rectangles for four scenes. These rectangles are a sketch representation of the image. Therefore, we adopt a likelihood model used in the primal sketch [2]. We avoid a region based model for the following two reasons. (i) The rectangles are occluding each other. We have to infer an partial order between the rectangles so that the region based model can be applied properly. This is an extra computational burden. (ii) The intensity inside rectangle could be non-trivial and includes shading effects, see the kitchen scene. Thus it is rather complex for region based model. In contrast, the primal sketch model [2] only needs to record the image primitives along the line segments (5 pixels perpendicular to the line segment), and the rest of the image is filled in with simple texture.

In the primal sketch model, the image lattice Λ is divided into two parts: the structural part along the boundaries of rectangles and the textural part for the remaining areas. $\Lambda = \Lambda_{\rm sk} \cup \Lambda_{\rm nsk}$. We divide the line segments of the rectangles in $C(\mathbf{G})$ into short segments (say 7 pixels long with 5 pixel width). We denote them by $\Lambda_{\rm sk,k}, k = 1, 2, ..., N$, thus $\Lambda_{\rm sk}$ consists of all such 5×7 image patches,

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

Each patch is fitted to an image primitive B_k from a dictionary Δ_B with Gaussian residue n.

$$\mathbf{I}(x, y) = B_k(x, y) + n(x, y),\ (x, y) \in \Lambda_{\rm sk,k}. \quad (4)$$

Thus the sketchable part follows a likelihood model

$$p(\mathbf{I}_{\rm sk,k}|C; \Delta_B) \propto \exp\{-\sum_{(x,y)\in\Lambda_{\rm sk,k}} \frac{(\mathbf{I}(x, y) - B_k(x, y))^2}{2\sigma^2}\}. \quad (5)$$

The textural part follows Markov random field models $p(\mathbf{I}_{\rm nsk}|\mathbf{I}_{\rm sk}; \beta)$ with parameters β conditioning on $\mathbf{I}_{\rm sk}$. So,

$$p(\mathbf{I}|C(\mathbf{G})) = p(\mathbf{I}_{\rm nsk}|\mathbf{I}_{\rm sk}; \beta) \prod_{k=1}^{N} p(\mathbf{I}_{\rm sk,k}|C; \Delta_B) \quad (6)$$

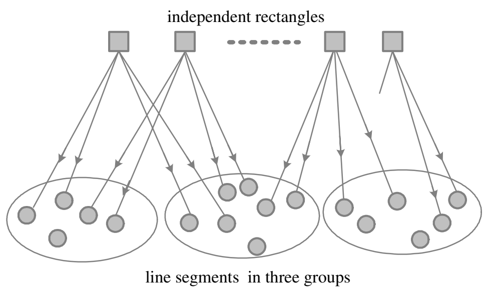


Figure 4. A simplified two layer model for bottom-up rectangle detection. Each rectangle consists of 2 pairs of nearly parallel line segments, and the rectangles are treated independently.

We refer to [2] for details of the primal sketch model.

3 Inference algorithm

Given our objective to compute the best parsing graph $\mathbf{G}$ for an input image, the computation algorithm must achieve three difficult tasks: (1). constructing the parsing graph, whose structure, like parse trees in natural languages, is not pre-determined but depends on the input image; (2). estimating the attributes of graph nodes (label and model parameters); and (3). computing the spatial relations between nodes on the same level.

There are several ways to infer the best parsing graph and Markov Chain Monte Carlo (MCMC) methods have been used in [10, 7]. In this paper, as our domain is limited to rectangle scenes and the parsing graph is not too big, the AI best first search algorithm can be directly applied to maximize the posterior probability in a steepest ascent way.

Our algorithm consists of two phases. In the first phase, a number of independent rectangle proposals are generated from the input image and an initialization for the parsing graph is achieved in a greedy way. In the second phase, these rectangle proposals are used with the grammars rules to efficiently construct the parsing graph in a bottom-up/top-down mechanism.

3.1 Generate bottom-up rectangle proposals and initialization

We start with edge detection and edge tracing to get a number of long contours, which are cut into a number of straight line segments by polygon-approximation. Then we run the vanishing point estimation algorithm in [9] to group all the line segments into three groups corresponding to the principal directions. Given the three orthogonal groups of

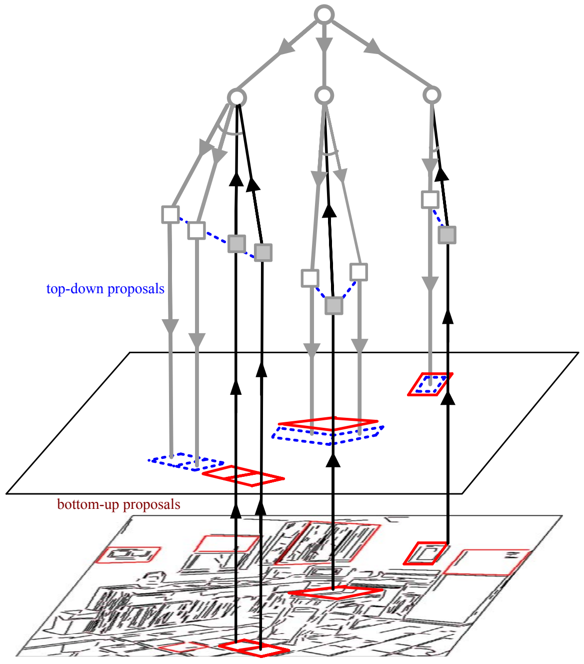


Figure 5. Bottom-up detection of rectangles instantiate some terminal nodes (shaded squares), which in turn activate graph grammar rules. Using context information, these rules generate top-down proposals for new rectangles which will be validated from the data (edge map).

line segments, we generate the rectangle proposals in the spirit of RANSAC [3]. We exhaustively draw two pairs of line segments from two out of the three line groups, and compute their intersection points as shown in Figure 4. In case the selected four lines indeed delineate a rectangle, we treat this as a valid proposal. However, these proposals are not equally good. So we weight them by their fitness to the given image. These weights can help us to sample the better proposals into the computation process first, and thus achieve speeding up.

Now we have an excessive number of weighted **bottom-up particles** for rectangle proposals. Some of them are conflicting with each other (e.g. two rectangle proposals share two or more edge segments) and only one of them should appear. We mark this conflicting relation among all the proposals and when one proposal is accepted, this can be used to shut down those conflicting with it.

Then these bottom-up particles are tested and pursued one by one in a procedure like the matching pursuit algorithm [5] used in wavelets. With a high stopping threshold, we obtain a number of "strong" hypotheses (rectangles) as initialization $\mathbf{G}^0$.

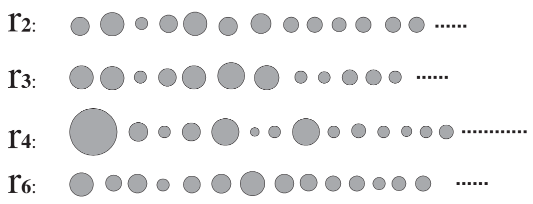


Figure 6. The proposed production rules are organized in four candidate rows, each has a posterior probability change it produces on current parsing graph as its importance.

3.2 Bottom-up and top-down construction of parsing graph with graph grammar

After the initialization, the following two steps are iteratively performed to construct the parsing graph by maximizing its posterior probability in a steepest ascend manner:

Step1: Each newly added rectangle in current state $\mathbf{G}^t$ matches to active one production rule (rule 5) with attributes passed to the non-terminal node. Then these non-terminal nodes in turn match to activate other production rules (rule 2,3,4,6) and so on, which propose a number of top-down particles. One example is shown in Figure 5. In this example, the four red rectangles instantiate four terminal nodes which activate four production rule 5. The left two further activate one production rule 2, which generates two new blue rectangles in a line with the two exiting red ones. The middle one further actives one production rule 6 to generate two new blue rectangles to form a cube aspect, while the left one activates one production rule 4 to generate one new blue rectangle to form a nested group.

All of these activated rules are put into four rows as shown in Figure 6. Each rule is weighted by the posterior probability change it achieves when being applied to the current parsing graph, which can be easily computed since each rule only affects the parsing graph locally. In addition, this posterior probability change depends on the state of the parsing graph. Therefore, we should on-line update the weights for all the rules in the four rows.

Step2: Select the rule with the biggest weight from all the rules in the four rows and accept it if its weight is positive. If accepted, it means a recognition of an object, and actions are taken to pass the attributes between a node and its parent through the constraint equations associated with this production rule. At the same time, this rule is removed from the rule rows. If not accepted, the whole process stops.

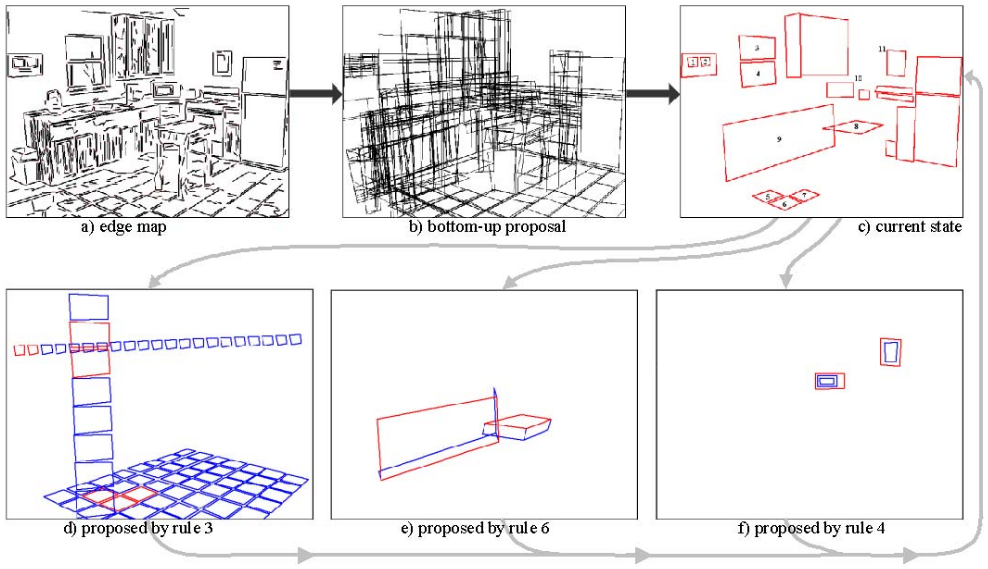


Figure 7. Running example for the kitchen scene. We first compute the edge map in (a), and generate a large number of rectangle hypotheses in (b). Some of them are validated by the generative model and we obtain the current state in (c). All the rectangles in (c) are used to activate the graph grammar rules. (d-e-f) are the top-down hypotheses (shown in blue) which are then feed to the generative model for validation iteratively. Not all top-down proposals are accepted.

4. Experiments

We run our algorithm on four typical scenes with rectangle structures. One running example in Figure 7 shows details of the algorithm. The final experimental results are shown in Figure 8. Some rectangles are missing due to the strong occlusion. In addition, it clearly shows that the high level knowledge introduced by the grammar greatly improves the results.

5. Discussion

In this paper, we study an attribute graph grammar for image parsing. The paper makes two main contributions to the vision literature. Firstly, it uses attributed graph grammar which is context sensitive for scene representation. 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. For future work, we should study general frameworks for learning the graph grammars and for systematic parsing algorithm with large set of graph grammar rules.

Acknowledgements

This work was supported in part by National Science Foundation grant IIS-0413214.

References

[1] S. Baumann. A simplified attribute graph grammar for high-level music recognition. *Third International Conference on Document Analysis and Recognition*, 1995.

[2] S. Z. C. Guo and Y. Wu. A mathematical theory of primal sketch and sketchability. *ICCV*, Nice, France, 2001.

[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*, 24:381–395, 1981.

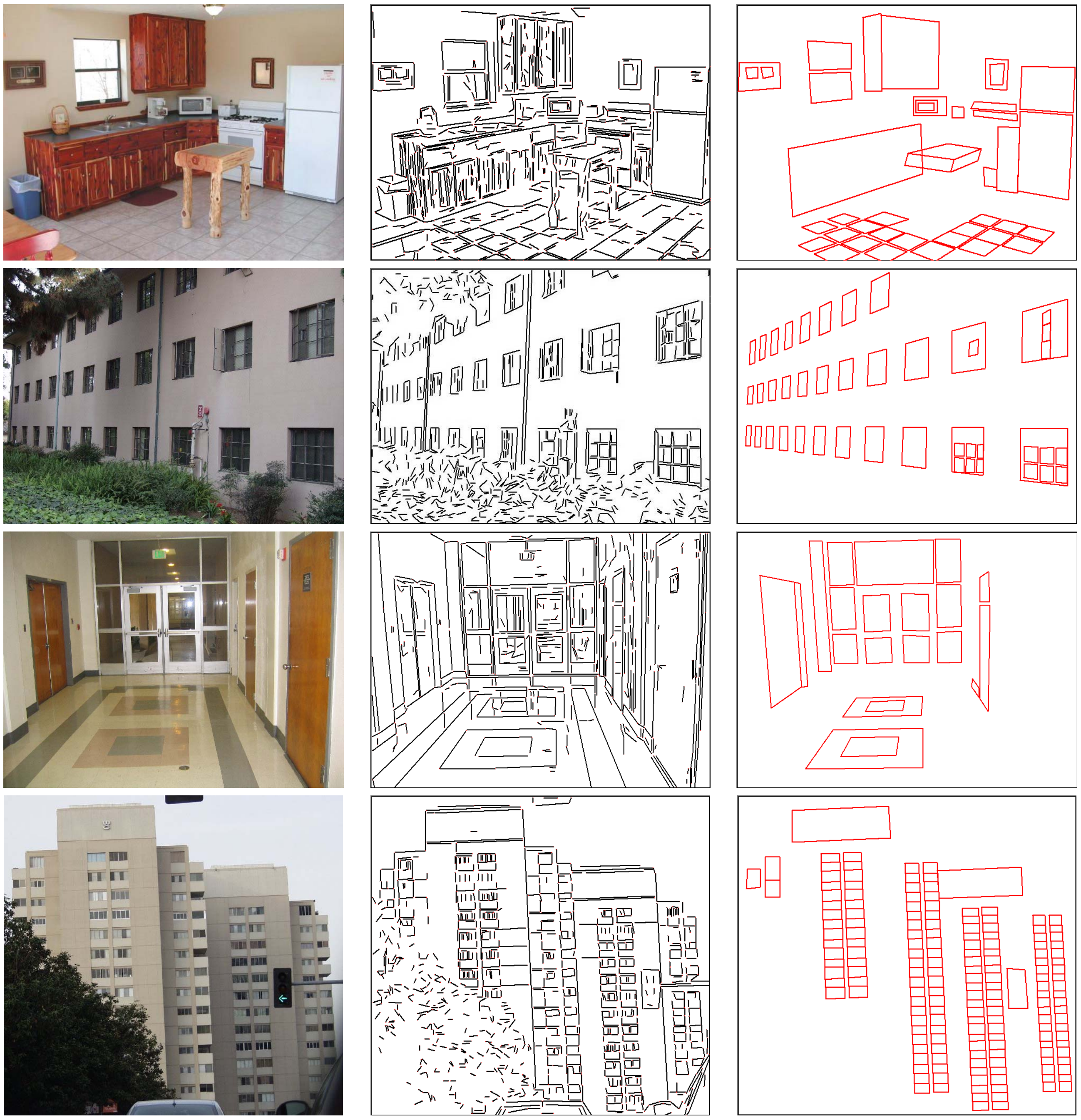
[4] K. Fu. *Syntactic Pattern Recognition and Applications*. Prentice Hall, 1981.

[5] S. Mallat and Z. Zhang. Matching pursuit with time-frequency dictionaries. *IEEE Trans. on Signal Processing*, 41(12):3397–3415, 1993.

[6] J. Rekers and A. Schurr. Defining and parsing visual languages with layered graph grammars. *J. Visual Language and Computing*, 9 1996.

[7] Z. Tu, X. Chen, A. Yuille, and S. Zhu. Image parsing: unifying segmentation, detection, and recognition. *ICCV*, 2003.

[8] F. You and K. Fu. A syntactic approach to shape recognition using attributed grammars. *IEEE Trans. on SMC*, 9:334–345, 1979.



(a) input image (b) edge map (c) rectangles detected and grouped.

Figure 8. Some experiments. The rectangles in (c) are detected from the edge maps in (b). Some rectangles are missing in the result, which are most attributed to occlusion.

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

[10] S. Zhu, R. Zhang, and Z. W. Tu. Integrating top-down/bottom-up for object recognition by ddmcmc. *CVPR*, 2000.