Compositional Boosting for Computing Hierarchical Image Structures

Tian-Fu Wu[1] and Gui-Song Xia[1]
[1]Lotus Hill Institute for Computer Vision and Information Science, Ezhou, 436000, China
{tfwu.lhi,gsxia.lhi}@lotushill.org

Song-Chun Zhu[1,2]
[2]Statistics Department University of California, Los Angeles, CA 90095, U.S.
sczhu@stat.ucla.edu

Abstract

In this paper, we present a compositional boosting algorithm for detecting and recognizing 17 *common image structures in low-middle level vision tasks. These structures, called "graphlets", are the most frequently occurring primitives, junctions and composite junctions in natural images, and are arranged in a 3-layer And-Or graph representation. In this hierarchic model, larger graphlets are decomposed (in And-nodes) into smaller graphlets in multiple alternative ways (at Or-nodes), and parts are shared and re-used between graphlets. Then we present a compositional boosting algorithm for computing the* 17 *graphlets categories collectively in the Bayesian framework. The algorithm runs recursively for each node* A *in the And-Or graph and iterates between two steps – bottom-up proposal and top-down validation. The bottom-up step includes two types of boosting methods. (i) Detecting instances of* A *(often in low resolutions) using Adaboosting method through a sequence of tests (weak classifiers) image feature. (ii) Proposing instances of* A *(often in high resolution) by binding existing children nodes of* A *through a sequence of compatibility tests on their attributes (e.g angles, relative size etc). The Adaboosting and binding methods generate a number of candidates for node* A *which are verified by a top-down process in a way similar to Data-Driven Markov Chain Monte Carlo [18]. Both the Adaboosting and binding methods are trained off-line for each graphlet category, and the compositional nature of the model means the algorithm is recursive and can be learned from a small training set. We apply this algorithm to a wide range of indoor and outdoor images with satisfactory results.*

1. Introduction

This paper presents a recursive algorithm called compositional boosting for detecting and recognizing 17 common image structures in low-middle level vision tasks, such as edge detection[13], segmentation[16], primal sketch[8], and junction detection [2, 12, 7, 14]. These structures are the most frequently occurring primitives, junctions and composite junctions in natural images, and we call them "graphlets" in this paper as they are represented in small graphical configurations.

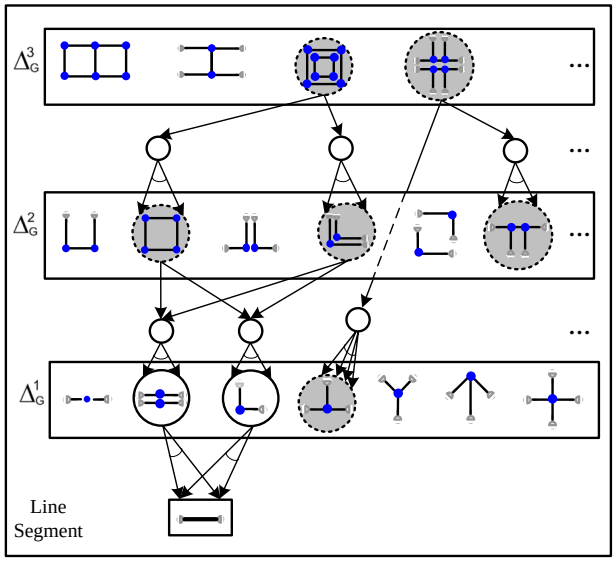

Figure 1. A hierarchic representation of 17 graphlets in a joint And-Or graph (links are reduced for clarity). Large graphlets are decomposed into smaller ones in multiple ways and share parts between them. All graphlets are composed of line segments at the lowest level.

As Figure 1 shows, these graphlets are compositional structures and can be arranged in a three-level And-Or graph representation. We only show a few of the links in Figure 1 for clarity. In this hierarchic representation, large graphlets are decomposed into smaller ones in multiple ways and share parts between them. An And-node (in solid circle) represents a decomposition and all of its children must appear together under certain spatial relations (collinear, parallel, proximity, and perpendicular), while an Or-node (in dashed circle) represents a few most plausible ways of decomposition and only one of its children may be

selected for each instance.

The study of this hierarchic representation is motivated by three objectives.

Firstly, detecting junctions is, by itself, known to be a challenging task in the vision literature [2, 12, 8, 13, 10, 7]. Local images are extremely ambiguous and yield multiple interpretations. To resolve the uncertainty, one should not only look at a larger scope, but also compute those competing configurations in an integrated manner, instead of detecting them independently. In this paper, we first identify the 17 most frequently occurring image structures and learn their binding relations in the And-Or graph. We then infer these graphlets from images in a common Bayesian framework. The final result is a sketch graph consisting of a number of graphlets which has cleaner edges and junctions.

Secondly, in a broader view, we may consider the graphlets as 17 small object categories, thus the representation and algorithm presented in this paper should reveal some desirable design principles for multi-class object recognition. More specifically, as compositionality and part-sharing [5, 3, 11] are common principles in object modeling and recognition, an effective inference algorithm must explore the compositional structures. However, in the object recognition literature, though there are recent works on joint training and classification on multiple categories, such as the JointBoosting[17], spatial boosting[1], mutual boosting [6] and other multi-class boosting method[4], these methods do not explicitly decompose objects into parts. Consequently their computation is based on raw image features, though some features are shared by different categories.

In contrast, our method in this paper exploits the explicit decomposition between the 17 classes. The algorithm runs recursively for each node A in the And-Or graph and iterate two steps – bottom-up proposal and top-down validation. The bottom-up step includes two types of boosting methods.

1. *Implicit boosting*: detecting instances of A (often in low resolutions) using Adaboosting through a sequence of tests on image features.

2. *Explicit binding*: proposing instances of A (often in high resolution) by binding existing components (i.e. children nodes of A) through a sequence of compatibility tests on their attributes, say angles, alignments, and relative size etc.

The two methods generate a number of candidates for node A which are verified by a top-down process in a way similar to DDMCMC [18]. Both the Adaboosting and binding are trained off-line for each graphlet category, and the composition and part sharing leads to smaller training sets and a recursive algorithm.

Thirdly, objects appear in multiple resolutions, and a robust algorithm must account for the scaling effects. In the literature of object detection and classification, images are often scaled to a certain regular window size, because the features in the detection algorithm are learned in that particular scale. For example, all face images must be down-scaled to around 20×20 pixels in Viola and Jones [19]. The down-scaling process loses information. In contrast, we represent each graphlet in multiple resolutions, as Figure 2 illustrates. The terminal nodes are the low resolution representation and are detected by tests on raw image features. The non-terminal nodes are the high resolution representation and are detected by tests on the attributes of their parts. The latter may be further defined in two resolutions as well.

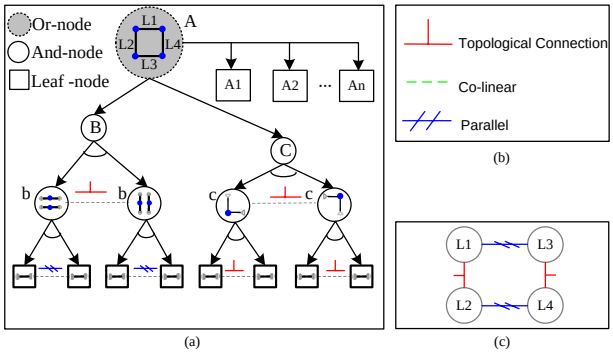

Figure 2. (a) An And-Or graph representation for one graphlet – the rectangle in two resolutions. The node A can either terminate into a leaf node (square) or have two ways of composition by nodes B and C. The dashed line between And-nodes or Leaf-nodes represent the relations of them. (b)Three generic and most prominent relations between any two line segments: co-linearity, parallelism and topological connection. (c) The relations between the 4 lines of the rectangle configuration.

The rest of the paper is arranged as follows. Section 2 presents the three-level generative model and learns a dictionary of graphlets. Section 3 proposes the compositional boosting algorithm and implements the pursuit of graphlets with experiments in Section 4. The paper is concluded with a discussion in Section 5.

2. Generative representation with graphlets

2.1. Learning the graphlets

In the first experiment, we collect a database of 200 images (many from the Corel dataset). To suit the low-middle level tasks, we choose images which contain generic structures at relatively high resolutions instead of complex objects or clutter (such as tree, texture etc). These images are manually segmented and sketched into graph representations, and we denoted these sketch graphs by

$$\text{Training data set 1} : DG = \{S_1, S_2, ..., S_{200}\}.$$

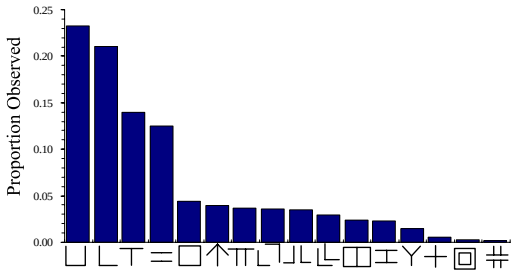

Figure 3. The normalized frequency counts (vertical axis) for the 16 most commonly observed graphlets (horizonal axis) on sketch graphs of natural images.

Starting with the edge element as the basic element, we count the frequency of all the small subgraph configurations (called graphlets) in the 200 graphs above. Some TPS (Thin Plate Spline) distance is used in the counting process. The frequency of the top 16 graphlets is plotted in Figure 3.(a). This excludes the edge element itself and the two line segment shown in Figure 1. Other graphlets are very rare (less than 0.2%) and thus are ignored.

We denote the graphlets and the frequency by

$$\text{graphlets set} : (\Delta_G, h) = \{(g_i, h(g_i)) : i = 2, 3, ..., 17\}. \tag{1}$$

We then learn the And-Or graph representation in Figure 1 automatically through a recursive binding process. Suppose our goal is to build a good coding model for the 200 sketch graphs in DG. We start with a naive model that codes each line segment independently. The learning process then explores the dependency between the line elements (alignments). Take for example the rectangle in Figure 2.(a). This rectangle A can be expressed by two possible compositions of graphlets which occur with probability ρ and $1-\rho$ respectively. Written in a stochastic production rule, it is,

$$\mathrm{A} \rightarrow \mathrm{b} \cdot \mathrm{b} \,|\, \mathrm{c} \cdot \mathrm{c}; \quad \rho|(1-\rho).$$

'|' means alternative choice and is represented by an "Or-node". '$\cdot$' means composition and is represented by an "And-node" with an arc underneath. b and c are smaller graphlets for parallel bars and L respectively.

Let $h(A)$ is the frequency of A in Eq.(1), and $p(A) = \rho p(b)p(b) + (1-\rho)p(c)p(c)$ is the accidental probability that A occurs through the b and c, then the following log-likelihood ratio measures the non-accidental statistics and thus the gain of coding length by binding the element into a graphlet A.

$$\delta\mathcal{L}(A) = \log \frac{h(A)}{p(A)}. \tag{2}$$

Multiplying it by the total frequency of A, we have the overall coding gain in adding the graphlet A in the "codebook",

$$\mathrm{Bind}(A) = h(A) \log \frac{h(A)}{p(A)} \tag{3}$$

where the larger $\mathrm{Bind}(A)$ is, the more significant A is.

In the next section, we shall use this coding length in the Bayesian framework to search for the most parsimonious representation.

Figure 1 shows the results we pursued from the image database. Currently, we treat the 17 hierarchical structures as a dictionary, denoted by Δ_G, augmenting the two-level models in low level vision to a three-level model to bridge low level and middle level representations.

$$\Delta_G = \Delta_G^1 \cup \Delta_G^2 \cup \Delta_G^3 \tag{4}$$

2.2. Generative image model with graphlets

Given an image $\mathbf{I}$ defined on an image domain Λ, we first convert it into a primal sketch representation S, following [8]. Like the manually sketched graph above, S is an attributed graph which can reconstruct the image through a dictionary Δ_{sk} of small image patches under the edges and fill in the remaining areas by texture. As discussed above, we further decompose this sketch graph into an unknown number of N graphlets through the graphlet dictionary Δ_G above.

$$G = (g_1(\beta_1), ..., G_N(\beta_N)), \tag{5}$$

where $\beta_i, i = 1, 2..., N$ are the parameters β_k (affine transformations plus deformations) for each graphlet g_i . That is, we have a decomposition

$$S = \cup_{i=1}^{N} g_i \ \cup \ g_o$$

where g_o is the remaining line segments in S.

Therefore we have a three level generative model,

$$G \stackrel{\Delta_G}{\Longrightarrow} S \stackrel{\Delta_{\mathrm{sk}}}{\Longrightarrow} \mathbf{I}. \tag{6}$$

This is described in the following joint probability

$$p(I, S, G) = p(I|S; \Delta_{sk})p(S|G; \Delta_G)p(G) \tag{7}$$

where $p(I|S; \Delta_{sk})$ follows the primal sketch model [8], and the joint probability of S and G are:

$$p(S, G; \Delta_G) \propto \exp\{-\mathcal{L}(g_o) - \mathcal{L}(G))\}, \tag{8}$$

where $\mathcal{L}(g)$ is the coding length of a given graphlet g. Each line segment in g_o is coded independently (thus expensive). The graphlets in G are also coded independent of each other.

Therefore we have a Bayesian formulation,

$$\begin{aligned}(G, S)^* &= \arg\max p(G, S|\mathbf{I}) \\ &= \arg\max p(I|S; \Delta_{sk})p(S|G; \Delta_G)p(G).\end{aligned} \tag{9}$$

Maximizing this Bayesian posterior probability leads to a improved sketch graph G which are decomposed into a number of graphlets $G = (g_1, ..., g_N)$.

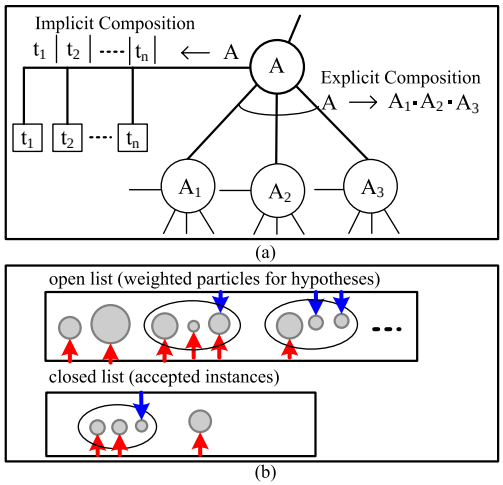

Figure 4. (a)Illustration of compositional boosting for a generic node A and (b) the data structure associated with each node for hypothesis testing. See text for detailed interpretation.

3. Compositional boosting

As the And-Or graph representation is recursive, our inference computes the graphlets in a recursive manner. Without loss of generality, we only interpret the algorithm for computing one node A in Figure 4, following the generic representation in Figure 2.

The algorithm remains two data structures for each node A:

- An **Open List**. It stores a number of weighted particles (or hypotheses) which are computed in a bottom-up process for the instances of A in the input image.
- A **Closed List**. It stores a number of instances for A which are accepted in the top-down process. These instances are nodes in the current parsing graph G.

The algorithm iterates over two processes: bottom-up proposal and top-down validation. The bottom-up process includes an Adaboosting and binding methods for generating a number of candidates for node A. These candidates are verified by a top-down process in a way similar to Data-Driven Markov Chain Monte Carlo [18].

Both the Adaboosting and binding methods in the bottom-up detection are trained off-line for each graphlet category, and the compositional model leads to small training set and recursive algorithm.

3.1. Bottom-up Proposals

The bottom-up process includes two types of boosting methods.

(i) Generating hypotheses for A directly from images. This bottom-up process uses Adaboosting [19, 9, 4, 15] for

Figure 5. Positive examples of various graphlets and negative training examples of background.

detecting the various terminals $t_1, ..., t_n$ without identifying the parts. The detection process tests some image features. This process is called *implicit testing*.

We train AdaBoost classifier [19, 9, 4] for each graphlet. Features used include statistics of gradients, both magnitude and orientation, differences between histograms of filter responses of the filter banks used by the primal sketch model, intensity edge flow and texture edge flow [16] at multiple scales over a local image patch Λ^i. There are 4158 features in total, denoted by $F(\mathbf{I}_{\Lambda^i})$. Some positive examples of graphlets and negative examples from the background are shown in Figure 5.

The particles generated by implicit testings are shown in Figure 4.(b) by single circles with bottom-up arrows. Figure 6 shows the proposal map for different kinds of graphlets.

The weight of a detected hypothesis (indexed by i) on image patch Λ^i is the logarithm of some local marginal posterior probability ratio,

$$\omega_A^i = \log \frac{p(A^i|\mathbf{I}_{\Lambda^i})}{p(\bar{A}^i|\mathbf{I}_{\Lambda^i})} \approx \log \frac{p(A^i|F(\mathbf{I}_{\Lambda^i}))}{p(\bar{A}^i|F(\mathbf{I}_{\Lambda^i}))} = \hat{\omega}_A^i. \quad (10)$$

where $\bar{A}$ represents a competing hypothesis. For computational effectiveness, the posterior probability ratio is approximated by posterior probabilities using local features $F(\mathbf{I}_{\Lambda^i})$ rather than the image $\mathbf{I}_{\Lambda^i}$.

(ii) Generating hypotheses for A by binding a number of k ($1 \leq K \leq n(A)$, $n(A) = 3$) parts $A_1, A_2, ...A_{n(A)}$. The binding process will test the relationships between these child nodes for compatibility and quickly rule out the obviously incompatible compositions. This is called *explicit testing*.

There are three relations $\{\tau_\perp, \tau_\parallel, \tau_\text{-}\}$ for perpendicularity, parallelism, and co-linearity respectively. We associate each relation type with a potential energy $(U_{\tau_\perp}, U_{\tau_\parallel}, U_{\tau_\text{-}})$.

By considering a line connecting the midpoints of the two segments, a and b, and denoting the smallest angles each segment forms with this line as θ_a and θ_b respectively.

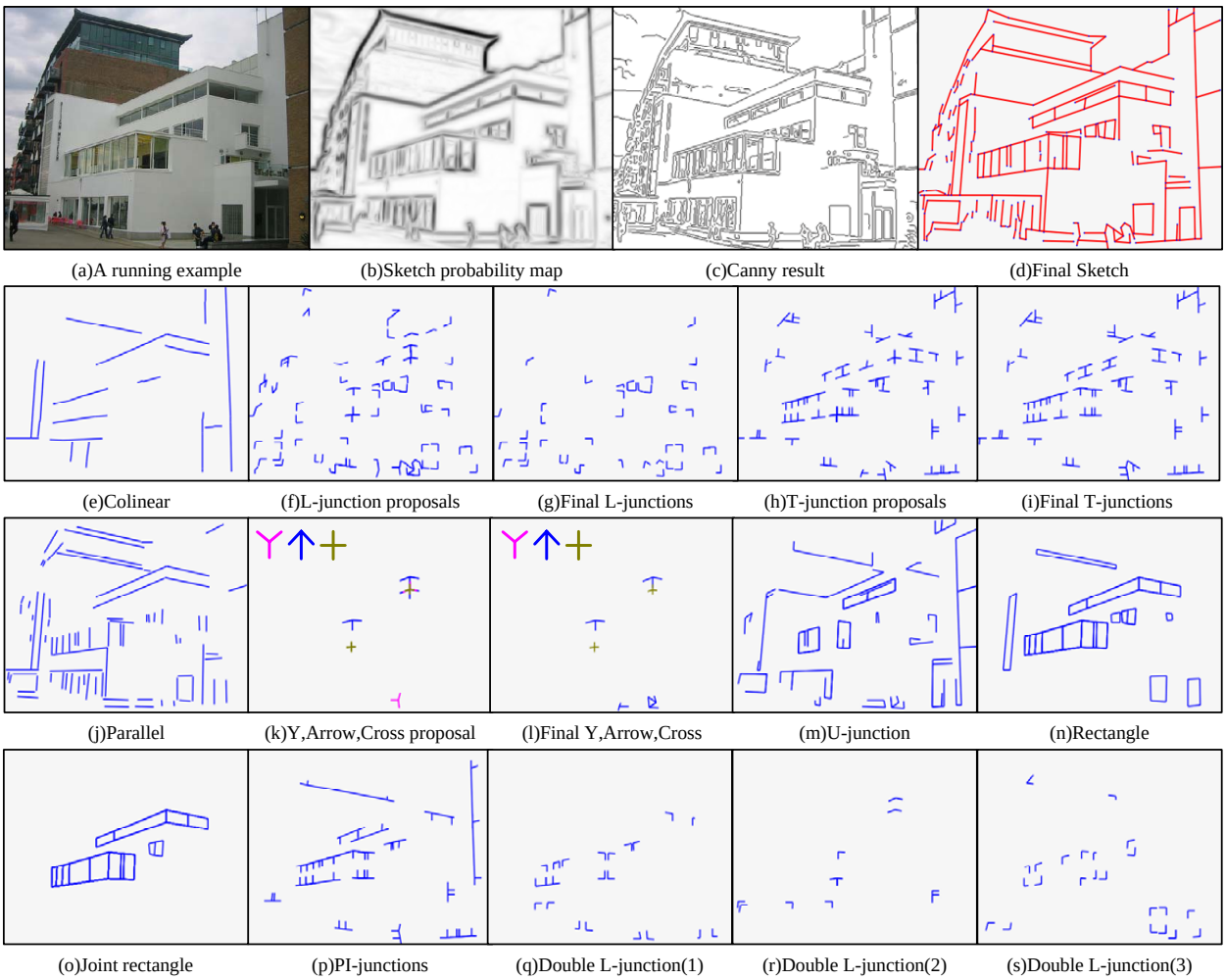

Figure 6. A running example for the computing the graphlets using compositional boosting. (e)-(l) are the bottom-up proposals (particles) for the graphlets respectively and (d) is the final sketch after verifying the graphlets. The final sketch is more concise and clean than the Canny edge map, especial on the junctions.

we define $(U_{r_\perp}, U_{r_\parallel}, U_{r_-})$ as follows

$$U_{r_-}(ab) = L(a)^2\theta_a + L(b)^2\theta_b \quad (11)$$

$$U_{r_\parallel}(ab) = (\theta_a - \theta_b)^2 \quad (12)$$

$$U_{r_\perp}(ab) = J(a)\ell_a + J(b)\ell_b \quad (13)$$

where $L(a)$, $L(b)$ are the lengths of line segments a and b respectively, ℓ_a and ℓ_b are the extended length of the two line segments that intersect one another. $J(a) = \infty$, if $\ell_a > CL(a)$ with a constant C favoring greater extensions for longer segments, and $J(a) = \frac{\ell_a}{L(a)}$ otherwise.

These potential functions are used as the attributes to test explicit binding for graphlets.

The particles generated by explicit testings are illustrated by a big ellipse containing $n(A) = 3$ small circles for its children in Figure 4.(b). Some examples are shown in Figure 6.

The weight of a binding hypothesis (indexed by i) is the logarithm of some local conditional posterior probability ratio. Suppose a particle A^i is bound from two existing parts A_1^i and A_2^i with A_3^i missing, and Λ^i is the domain containing the hypothesized A. Then the weight will be

$$\begin{aligned}\omega_A^i &= \log \frac{p(A^i|A_1^i, A_2^i, \mathbf{I}_{\Lambda^i})}{p(\bar{A}^i|A_1^i, A_2^i, \mathbf{I}_{\Lambda^i})} \\ &= \log \frac{p(A_1^i, A_2^i, \mathbf{I}_{\Lambda^i}|A^i)p(A^i)}{p(A_1^i, A_2^i, \mathbf{I}_{\Lambda^i}|\bar{A}^i)p(\bar{A}^i)} \\ &\approx \log \frac{p(A_1^i, A_2^i|A^i)p(A^i)}{p(A_1^i, A_2^i|\bar{A}^i)p(\bar{A}^i)} = \hat{\omega}_A^i. \end{aligned} \quad (14)$$

where $\bar{A}$ means competitive hypothesis. $p(A_1^i, A_2^i|A^i)$ is reduced to tests of compatibility between A_1^i and A_2^i for computational efficiency. It leaves the computation of searching

for A_3^i as well as fitting the image area $\mathbf{I}_{\Lambda_A}$ to the top-down process.

The two kinds of bottom-up tests generate hypotheses for graphlets and place them into an open list.

Results of bottom-up proposals are shown in Figure 6 and Figure 8, and as we can see in the figures, there can be more than one particle on the same patch, which is caused by the local ambiguity and needs top-down verification to be resolved.

3.2. Top-down verifications

The top-down process validates the bottom-up hypotheses in all the Open lists and accepted hypotheses are placed into a closed list, following the Bayesian posterior probability. It also needs to maintain the weights of the Open lists.

(i) Given a hypothesis A^i with weight $\hat{\omega}_A^i$, the top-down process validates it by computing the true posterior probability ratio ω_A^i stated above. If A^i is accepted, it is placed into the Closed list of A. The criterion of the acceptance is discussed below. In a reverse process, the top-down process may also select a node A in the Closed list, and then either delete it (putting it back to the Open list) or disassemble it into independent parts.

(ii) The top down process must maintain the weights of the particles in the Open Lists after adding (or removing) a node A^i. It is clear that the weight of each particle depends on the competing hypothesis. Thus for two competing hypotheses A and A' which overlap in a domain Λ_o, accepting one hypothesis will lower the weight of the other. Therefore, whenever we add or delete a node A, all the other hypotheses whose domains overlap with that of A will have to update their weights.

The acceptance of a node can be computed by a greedy algorithm that maximizes the posterior probability. At each iteration it selects the particle whose weight is the largest among all Open lists and then accepts it until the largest weight is below a threshold.

For the pursuit of graphlets, the top-down process verifies and selects a subset $G = \{g_1(\beta_1), g_2(\beta_2), ..., g_N(\beta_N)\}$ that maximizes the joint probability of Eq. 8. We do this with a process that maximizes a posterior (MAP).

$$\begin{aligned}(S^*, G^*) &= \arg\max p(G, S|I; \Delta_B, \Delta_G) \\ &= \arg\max p(G, S, I; \Delta_B, \Delta_G) \\ &= \arg\min \mathcal{L}(g_0) + \mathcal{L}(G) + \mathcal{L}(I|S) \\ &= \arg\min \mathcal{L}_{g_0} + \mathcal{L}_G + \mathcal{L}_{I|S} \qquad (15)\end{aligned}$$

Where $\mathcal{L}_{g_0}$ is the coding length of the residual of the sketch graph after encoding by G, $\mathcal{L}_G$ is coding length of the selected graphlets, and $\mathcal{L}_{I|S}$ is the coding length of the sketchable region of the image I. The encoding of this region is based on works in [8].

Compositional Boosting

Input: an image $\mathbf{I}$ and an And-Or graph.
Output: a parsing graph $\mathbf{pg}$ with initial $\mathbf{pg} = \emptyset$.
1. Repeat
2. Schedule the next node to visit A
3. Call the $\mathrm{Bottom-Up(A)}$ process to update A's Open lists
4. (i) Detect terminal instances for A from images
5. (ii) Bind non-terminal instances for A from its children's Open or Closed lists.
6. Call the $\mathrm{Top-Down(A)}$ process to update A's Closed and Open lists
7. (i) Accept hypotheses from A's Open list to its Closed list.
8. (ii) Remove (or disassemble) hypotheses from A's closed lists.
9. (iii) Update the Open lists for particles that overlap with current node.
10. Until a certain number of iteration or the largest particle weight is below a threshold.

Figure 7. Flow of compositional boosting algorithm

Results are shown in Figure 6, where we can see that some particles appear in the proposal but do not appear in the final results.

The algorithm described thus far is deterministic. As an alternative, one may use a stochastic algorithm with reversible jumps. According to the terminology of data driven Markov chain Monte Carlo (DDMCMC) [18], one may view the approximative weight $\hat{\omega}_A^i$ as a logarithm of the proposal probability ratio. For the stochastic algorithm, its initial stage is often deterministic when the particle weights are very large and the acceptance probability is always 1, so this approach is generally only valuable when ω_A^i is close to 0.

We summarize the compositional boosting algorithm as shown in Figure 7.

The key issue of the inference algorithm is to order the particles in the Open and Closed lists. In other words, the algorithm must schedule the bottom-up and top-down processes to achieve computational efficiency. The optimal schedule between bottom-up and top-down is a long standing problem in vision. A greedy way for scheduling is to measure the information gain of each step, either a bottom-up testing/binding or a top-down validation, divided by its computational complexity (CPU cycles). Then one may order these steps by the gain/cost ratio.

4. Experiments

In our second experiment, we apply the compositional boosting algorithm to compute a sketch graph S and a series of graphlets G in a wide variety of indoor and outdoor images as shown in Figure 6 and Figure 8 (Please refer to the supplemental file for an animation of the compositional boosting process and more results). As in our training set, we are targeting low-middle level generic structures and thus avoid textures and clutter.

In Figure 8, we show the results of five images. For each image, The "Canny" image is the Canny edge map which are pixel level representation and has no concepts such as corners, junctions and line segments. The "First layer" image is the detection result of the first layer graphlets including different kinds of junctions where parallel lines are in red, L-junction in green, T-junction in blue, Arrow junction in yellow and so on. The "Final sketch" image is the sketch graph G which are graph level representation and more concise than the Canny edge map. It has the concepts of different graphlets. Many high-level vision tasks can be based on this representation.

In these images, the junctions and composite junctions are much improved in comparison to the Canny edge map. It is also much improved in comparison to the primal sketch method [8].

5. Discussion

In this paper, we present a compositional boosting algorithm for detecting and recognizing 17 common image structures in low-middle level vision tasks. We conducted two sets of experiments: one on learning and binding the graphlets from sketch graph, and the other on detecting the graphlets from raw images. The key contribution of this paper is a recursive compositional boosting algorithm which explore the recursive decomposition structures in the representation, in contrast to the literature on spatial boosting [1] and JointBoosting [17], and mutual boosting[6]. It only needs a small set of training examples and is easy to scale when new nodes are added – all are desirable properties for object detection and recognition. In ongoing projects, we are applying this algorithm to object recognition by moving up the hierarchy.

6. Acknowledgements

This work is done at the Lotus Hill Research Institute and is supported by : National 863 project(No. 2006AA01Z121), National Science Foundation China(No. 60672162 and No. 60673198). The data used in this paper were provided by the Lotus Hill Annotation project, which was supported partially by a sub-award from the W.M. Keck foundation, a Microsoft gift. The authors thank Di Lu for his help in the preparation of training data.

References

[1] S. Avidan. Spatialboost: Adding spatial reasoning to adaboost. In *ECCV*, 2006. 2, 7

[2] D. J. Beymer. Finding junctions using the image gradient. In *MIT AI Memo*, 1991. 1, 2

[3] H. Chen, Z. J. Xu, Z. Q. Liu, and S. C. Zhu. Composite templates for cloth modeling and sketching. In *CVPR*, 2006. 2

[4] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 2, 4

[5] S. G. E. Bienenstock and D. Potter. Compositionality, mdl priors, and object recognition. In *Advances in Neural Information Processing Systems 9, M.Mozer, M.Jordan, T.Petsche, eds., MIT Press*, 1998. 2

[6] M. Fink and P. Perona. Mutual boosting for contextual inference. In *NIPS*, 2003. 2, 7

[7] G.Giraudon and R.Deriche. On corner and vertex detections. In *CVPR*, 1991. 1, 2

[8] C. E. Guo, S. C. Zhu, and Y. N. Wu. Towards a mathematical theory of primal sketch and sketchability. In *ICCV*, 2003. 1, 2, 3, 6, 7

[9] T. H. J. Friedman and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 38(2):337–374, 2000. 4

[10] D. Li, G. Sullivan, and K. Baker. Edge detection at junctions. In *Proc. Alvey Vision Conference*, 1989. 2

[11] B. Ommer and J. M. Buhmann. Learning compositional categorization models. In *ECCV*, 2006. 2

[12] L. Parida, D. Geiger, and R. Hummel. Junctions: Detection, classification, and reconstruction. *IEEE Trans. on PAMI*, 20(7):687–698, July 1998. 1, 2

[13] X. Ren, C. Fowlkes, and J. Malik. Familiar configuration enables figure/ground assignment in natural scenes. *Vision Science Society*, 2005. 1, 2

[14] M. A. Ruzon and C. Tomasi. Edge, junction, and corner detection using color distributions. *IEEE Trans. on PAMI*, 23(11):1281–1295, Nov. 2001. 1

[15] R. E. Schapire. The boosting approach to machine learning: an overview. In *MSRI Workshop on nonlinear Estimation and Classification*, 2002. 4

[16] B. Sumengen and B. S. Manjunath. Edgeflow-driven variational image segmentation: Theory and performance evaluation. *IEEE Trans. on PAMI*, 2005. 1, 4

[17] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR*, 2004. 2, 7

[18] Z. Tu and S. C. Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Trans. on PAMI*, 24(5):657–673, May 2002. 1, 2, 4, 6

[19] P. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 2, 4

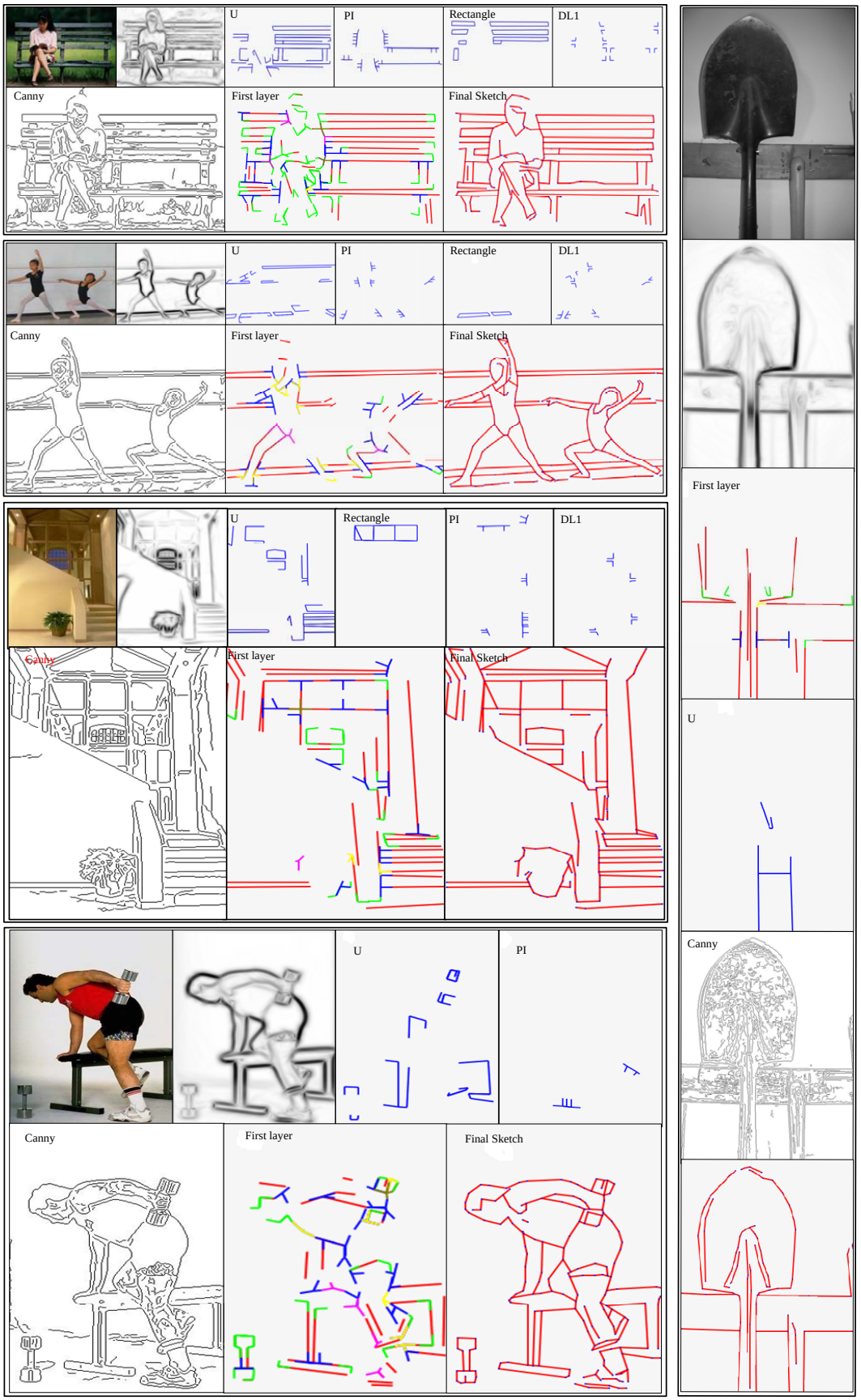

Figure 8. More experiments on computing the graphlets.