Foundations and Trends® in
Computer Graphics and Vision
Vol. 2, No. 4 (2006) 259–362
© 2007 S.-C. Zhu and D. Mumford
DOI: 10.1561/0600000018

A Stochastic Grammar of Images

Song-Chun Zhu[1,*] **and David Mumford**[2]

[1] *University of California, Los Angeles, USA, sczhu@stat.ucla.edu*
[2] *Brown University, USA, David_Mumford@brown.edu*

Abstract

This exploratory paper quests for a stochastic and context sensitive grammar of images. The grammar should achieve the following four objectives and thus serves as a unified framework of representation, learning, and recognition for a large number of object categories. (i) The grammar represents both the hierarchical decompositions from scenes, to objects, parts, primitives and pixels by terminal and non-terminal nodes and the contexts for spatial and functional relations by horizontal links between the nodes. It formulates each object category as the set of all possible valid configurations produced by the grammar. (ii) The grammar is embodied in a simple And–Or graph representation where each Or-node points to alternative sub-configurations and an And-node is decomposed into a number of components. This representation supports recursive top-down/bottom-up procedures for image parsing under the Bayesian framework and make it convenient to scale up in complexity. Given an input image, the image parsing task constructs a most probable parse graph on-the-fly as the output interpretation and this parse graph is a subgraph of the And–Or graph after

[*] Song-Chun Zhu is also affiliated with the Lotus Hill Research Institute, China.

making choice on the Or-nodes. (iii) A probabilistic model is defined on this And–Or graph representation to account for the natural occurrence frequency of objects and parts as well as their relations. This model is learned from a relatively small training set per category and then sampled to synthesize a large number of configurations to cover novel object instances in the test set. This generalization capability is mostly missing in discriminative machine learning methods and can largely improve recognition performance in experiments. (iv) To fill the well-known semantic gap between symbols and raw signals, the grammar includes a series of visual dictionaries and organizes them through graph composition. At the bottom-level the dictionary is a set of image primitives each having a number of anchor points with open bonds to link with other primitives. These primitives can be combined to form larger and larger graph structures for parts and objects. The ambiguities in inferring local primitives shall be resolved through top-down computation using larger structures. Finally these primitives forms a primal sketch representation which will generate the input image with every pixels explained. The proposal grammar integrates three prominent representations in the literature: stochastic grammars for composition, Markov (or graphical) models for contexts, and sparse coding with primitives (wavelets). It also combines the structure-based and appearance based methods in the vision literature. Finally the paper presents three case studies to illustrate the proposed grammar.

1

Introduction

1.1 The Hibernation and Resurgence of Image Grammars

Understanding the contents of images has always been the core problem in computer vision with early work dated back to Fu [22], Riseman [33], Ohta and Kanade [54, 55] in the 1960–1970s. By analogy to natural language understanding, the task of image parsing [72], as Figure 1.1 illustrates, is to compute a parse graph as the most probable interpretation of an input image. This parse graph includes a tree structured decomposition for the contents of the scene, from scene labels, to objects, parts, primitives, so that all pixels are explained, and a number of spatial and functional relations between nodes for contexts at all levels of the hierarchy.

People who worked on image parsing in the 1960–1970s were, obviously, ahead of their time. In Kanade's own words, they had only 64K memory to work with at that time. Indeed, his paper with Ohta [55] was merely 4-page long! The image parsing efforts and structured methods encountered overwhelming difficulties in the 1970s and since then entered a hibernation state for a quarter of a century. The syntactic and grammar work have been mostly studied in the backstage as we

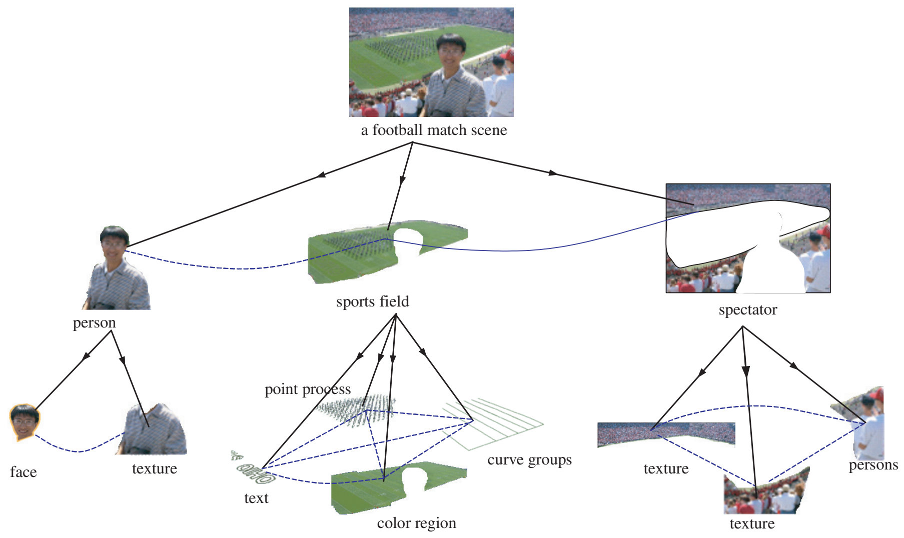


Fig. 1.1 Illustrating the task of image parsing. The parse graph includes a tree structured decomposition in vertical arrows and a number of spatial and functional relations in horizontal arrows. From [72].

shall review in later section. These difficulties remain challenging even today.

Problem 1: There is an enormous amount of visual knowledge about the real world scenes that has to be represented in the computer in order to make robust inference. For example, there are at least $3,000$ object categories[1] and many categories have wide intra-category structural variations. The key questions are: how does one define an object category, say a car or a jacket? and how does one represent these categories in a consistent framework?

The visual knowledge is behind our vivid dreams and imaginations as well as the top-down computation. It was known that there are far more downward fibers than upward fibers in the visual pathways of primate animals. For example, it is reported in [65] that only 5%–10% of the input to the geniculate relay cells derives from the retina. The

[1] This number comes from Biederman who adopted a method used by pollsters. Take an English dictionary, open some pages at random, and count the number of nouns which are object categories at a page and then times the number of pages of the dictionary proportionally.

rest derives from local inhibitory inputs and descending inputs from layer 6 of the visual cortex. The weakness in knowledge representation and top-down inference is, in our opinion, the main obstacle in the road toward robust and large scale vision systems.

Problem 2: The computational complexity is huge.[2] A simple glance of Figure 1.1 reveals that an input image may contain a large number of objects. Human vision is known [70] to simultaneously activate the computation at all levels from scene classification to edge detection — all occurs in a very short time $\leq 400\,\text{ms}$, and to adopt multiple visual routines [76] to achieve robust computation. In contrast, most pattern recognition or machine learning algorithms are feedforward and computer vision systems rarely possess enough visual knowledge for reasoning.

The key questions are: how does one achieve robust computation that can be scaled to thousands of categories? and how does one coordinate these bottom-up and top-down procedures? To achieve scalable computation, the vision algorithm must be based on simple procedures and structures that are common to all categories.

Problem 3: The most obvious reason that sent the image parsing work to dormant status was the so-called semantic gap between the raw pixels and the symbolic token representation in early syntactic and structured methods. That is, one cannot reliably compute the symbols from raw images. This has motivated the shift of focus to appearance based methods in the past 20 years, such as PCA [75], AAM [12], and appearance based recognition [51], image pyramids [69] and wavelets [15], and machine learning methods [21, 63, 78] in the past decade.

Though the appearance based methods and machine learning algorithms have made remarkable progress, they have intrinsic problems that could be complemented by structure based methods. For example, they require too many training examples due to the lack the compositional and generative structures. They are often over-fit to specific training set and can hardly generalize to novel instances or configurations especially for categories that have large intra-class variations.

[2] The NP-completeness is no longer an appropriate measure of complexity, because even many simplified vision problems are known to be NP-hard.

After all these developments, the recent vision literature has observed a pleasing trend for returning to the grammatical and compositional methods, for example, the work in the groups of Ahuja [71], Geman [27, 36], Dickinson [14, 40], Pollak [79], Buhmann [57] and Zhu [9, 32, 44, 59, 72, 74, 85, 86]. The return of grammar is in response to the limitations of the appearance based and machine learning methods when they are scaled up.

The return of grammar is powered by progresses in several aspects, which were not available in the 1970s. (i) A consistent mathematical and statistical framework to integrate various image models, such as Markov (graphical) models [90], sparse coding [56], and stochastic context free grammar [10]. (ii) More realistic appearance models for the image primitives to connect the symbols to pixels. (iii) More powerful algorithms including discriminative classification and generative methods, such as the Data-Driven Markov China Monte Carlo (DDMCMC) [73]. (iv) Huge number of realistic training and testing images [87].

1.2 Objectives

This exploratory paper will review the issues and recent progress in developing image grammars, and introduce a stochastic and context sensitive grammar as a unified framework for representation, learning, and recognition. This framework integrates many existing models and algorithms in the literature and addresses the problems raised in the previous subsection. This image grammar should achieve the following four objectives.

Objective 1: *A common framework for visual knowledge representation and object categorization.* Grammars, studied mostly in language [1, 26], are known for their expressive power in generating a very large set of configurations or instances, i.e., their language, by composing a relatively much smaller set of words, i.e., shared and reusable elements, using production rules. Hierarchic and structural composition is the key concept behind grammars in contrast to enumerating all possible configurations.

In this paper, we embody the image grammar in an And–Or graph representation[3] where each Or-node points to alternative subconfigurations and an And-node is decomposed into a number of components. This And–Or graph represents both the hierarchical decompositions from scenes, to objects, parts, primitives and pixels by terminal and non-terminal nodes and the contexts for spatial and functional relations by horizontal links between the nodes. It is an alternate way of representing production rules and it contains all possible parse trees. Then we will define a probabilistic model for the And–Or graph which can be learned from examples using maximum likelihood estimation. Therefore, all the structural and contextual information are represented in the And–Or graph (and equivalently the grammar). This also resolve the object categorization problem. We can define each object category as *the set of all valid configurations which are produced by the grammar, with its probability learned to reproduce natural frequency of instances occurring in the observed ensemble.*

As we will show in later section, this probability model integrates popular generative models, such as sparse coding (wavelet coding) and stochastic context free grammars (SCFG), with descriptive models, such as Markov random fields and graphical models. The former represents the generative hierarchy for reconfigurability while the latter models context.

Objective 2: *Scalable and recursive top-down/bottom-up computation.* The And–Or graph representation has recursive structures with two types of nodes. It can be easily scalable up in the number of nodes and object categories. For example, suppose an Or-node represents an object, say car, it then has a number of children nodes for different views (front, side, back etc.) of cars. By adding a new child node, we can augment to new views. This representation supports recursive top-down/bottom-up procedures for image parsing and make it convenient to scale up in complexity.

Figure 1.2 shows a parsing graph under construction at a time step. This simple grammar is one of our case study in later section uses one

[3] The And–Or graph was previously used by Pearl in [58] for heuristic searches. In our work, we use it in a very different purpose and should not be confused with Pearl's work.

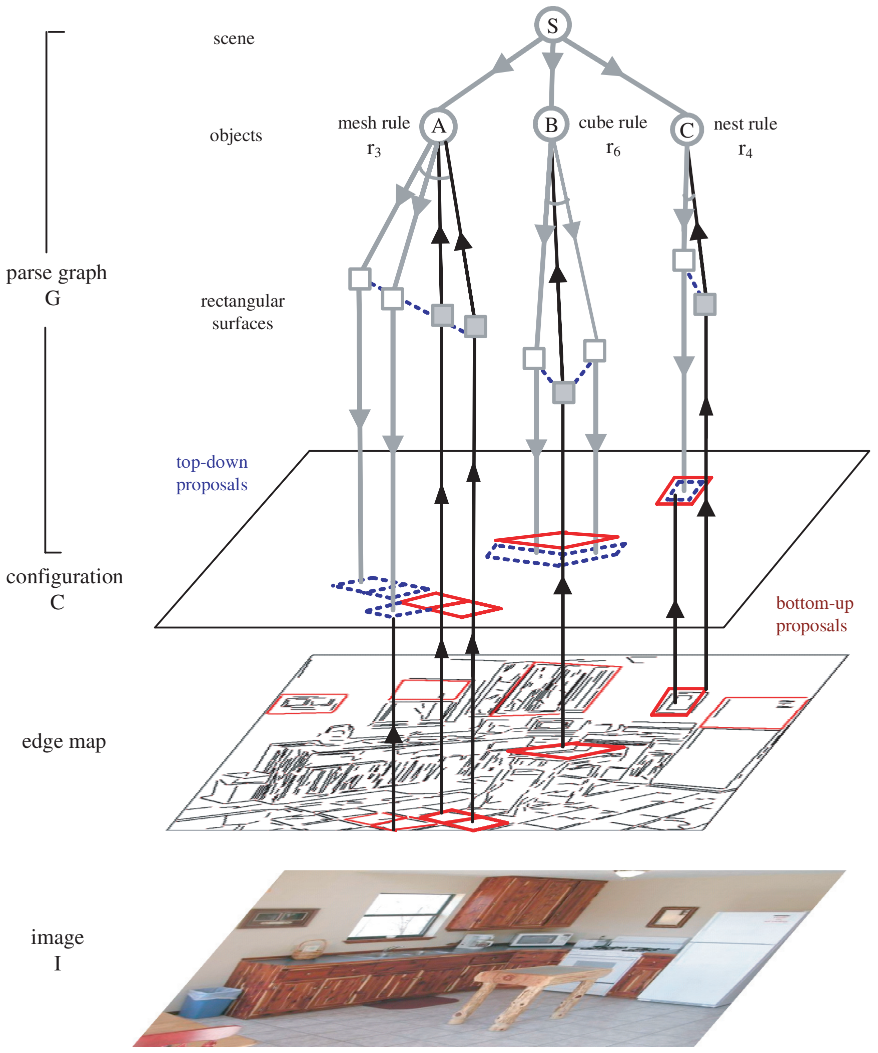


Fig. 1.2 Illustrating the recursive bottom-up/top-down computation processes in image parsing. The detection of rectangles (in red) instantiates some non-terminal nodes shown as upward arrows. They in turn activate graph grammar rules for grouping larger structures in nodes A, B, and C, respectively. These rules generate top-down prediction of rectangles (in blue). The predictions are validated from the image under the Bayesian posterior probability. Modified from [59].

primitive: rectangular surfaces projected onto the image plane. The grammar rules represents various organization, such as alignments of the rectangles in mesh, linear, nesting, cubic structures. In the kitchen scene, the four rectangles (in red) accepted through bottom-up process and they activate the production rules represented by the non-terminal nodes A, B, and C, respectively. Which then predict a number of candidates (in blue) in top-down search. The solid upward arrows show the bottom-up binding, while the downward arrows show the top-down prediction. As the ROC curves in Figure 9.5 shows in later section, the top-down prediction largely improves the recognition rate of the rectangles, as certain rectangles can only be hallucinated through top-down process due to occlusion and severe image degradation.

Given an input image, the image parsing task constructs a most probable parse graph on-the-fly as the output interpretation and this parse graph is a subgraph of the And–Or graph after making choices on the Or-nodes.

As we shall discuss in later section, the computational algorithm maintains the same data structures for each of the And-nodes and Or-nodes in the And–Or graph and adopt the same computational procedure: (i) bottom-up detecting and binding using a cascade of features; and (ii) top-down on-line template composition and matching. To implement the system, we only need to write one common class (in C++ programming) for all the nodes, and different objects and parts are realized as instances of this class. These nodes use different bottom-up features/tests and the top-down templates during the computational process. The features and templates are learned off-line through training images and loaded into the instances of the C++ class during the computational process. This recursive algorithm has the potential to be implemented in a massively parallel machine where each unit has the same data structures and functions described above.

Objective 3: *Small sample learning and generalization.* The probabilistic model defined on this And–Or graph representation can be learned from a relatively small training set per category and then sampled through Monte Carlo simulation to synthesize a large number of configurations. This is in fact an extension to the traditional texture synthesis experiment by the minimax entropy principle [90], where new texture samples are synthesized which are different from the observed texture but are perceptually equivalent to the observed texture. The minimax entropy learning scheme is extended to the And–Or graph models in [59], which can generate novel configurations through composition to cover unforeseen object instances in the test set. This generalization capability is mostly missed in discriminative machine learning methods.

In the experiments reported in [44, 59], they seek for the minimum number of distinct training samples needed for each category, usually in the range of 20–50. They prune some redundant examples which can be derived through other examples by composition. Then they found that the generated samples can largely improve the object recognition performance. For example, a 15% recognition rate is reported in [44].

Objective 4: *Mapping the visual vocabulary to fill the semantic gap.* To fill the well-known semantic gap between symbols and pixels, the grammar includes a series of visual dictionaries for visual concepts at all levels. There are two key observations for these dictionaries.

1. The elements of the dictionaries are organized through graph composition. At the bottom-level the dictionary is a set of image primitives each having a number of anchor points in a small graph with open bonds to link with other primitives. These primitives can be combined to form larger and larger graph structures for parts and objects, in a way similar to Lego pieces that kids play with.[4]
2. Vision is distinct from other sensors, like speech in the aspect that objects can appear at arbitrary scales. As a result, the instances of each node can occur at any sizes. The non-terminal nodes at all levels of the And–Or graph can terminate directly as image primitives. Thus one has to account for the transitions between instances of the same node over scales. This is the topics studied in the perceptual scale space theory [80].

[4] Note that Lego pieces are well designed to have standardized teeth to fit each other, this is not true in the image primitives. The latter are more flexible.

Though there are variations in the literature for what the low level primitives should be, the differences are really minor between what people called textons, texels, primitives, patches, and fragments. The ambiguities in inferring these local primitives shall be resolved through top-down computation using larger structures.

Finally the primitives are connected to form a primal sketch graph representation [31] which will generate the input image with every pixels explained. This closes the semantic gap.

1.3 Overview of the Image Grammar

In this subsection, we overview the basic concepts in the image grammar. We divided it into two parts: (i) representation and data structures, (ii) Image annotation dataset to learn the grammar, and the learning and computing issues.

1.3.1 Overview of the Representational Concepts and Data Structures

We use Figure 1.3 as an example to review the representational concepts in the following:

1. *An And–Or graph.* Figure 1.3(a) shows a simple example of an And–Or graph. An And–Or graph includes three types of nodes: And-nodes (solid circles), Or-nodes (dashed circles), and terminal nodes (squares). An And-node represents a decomposition of an entity into its parts. It corresponds to the grammar rules, for example,

   $$A \to BCD, \quad H \to NO.$$

   The horizontal links between the children of an And-node represent relations and constraints. The Or-nodes act as "switches" for alternative sub-structures, and stands for labels of classification at various levels, such as scene category, object classes, and parts etc. It corresponds to production rules like,

   $$B \to E \mid F, \quad C \to G \mid H \mid I.$$

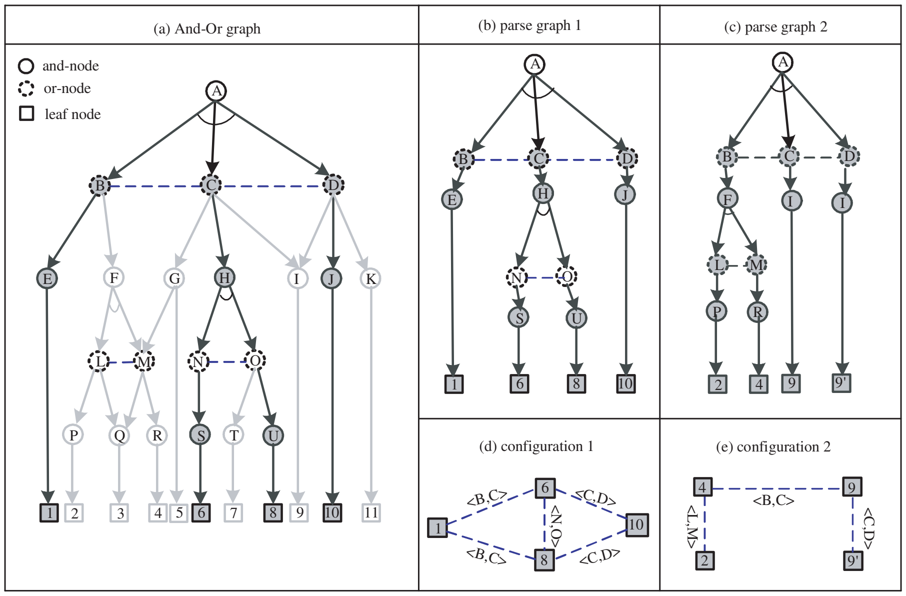


Fig. 1.3 Illustrating the And–Or graph representation. (a) An And–Or graph embodies the grammar productions rules and contexts. It contains many parse graphs, one of which is shown in bold arrows. (b) and (c) are two distinct parse graphs by selecting the switches at related Or-nodes. (d) and (e) are two graphical configurations produced by the two parse graphs, respectively. The links of these configurations are inherited from the And–Or graph relations. Modified from [59].

Due to this recursive definition, one may merge the And–Or graphs for many objects or scene categories into a larger graph. In theory, all scene and object categories can be represented by one huge And–Or graph, as it is the case for natural language. The nodes in an And–Or graph may share common parts, for example, both cars and trucks have rubber wheels as parts, and both clock and pictures have frames.

2. *A parse graph*, as shown in Figure 1.1, is a hierarchic generative interpretation of a specific image. A parse graph is augmented from a parse tree, mostly used in natural or programming language by adding a number of relations, shown as side links, among the nodes. A parse graph is derived from the And–Or graph by selecting the switches or classification labels at related Or-nodes. Figures 1.3(b) and 1.3(c)

are two instances of the parse graph from the And–Or graph in Figure 1.3(a). The part shared by two node may have different instances, for example, node I is a child of both nodes C and D. Thus we have two instances for node 9.

3. *A configuration* is a planar attribute graph formed by linking the open bonds of the primitives in the image plane. Figures 1.3(d) and 1.3(e) are two configurations produced by the parse graphs in Figures 1.3(b) and 1.3(c), respectively. Intuitively, when the parse graph collapses, it produces a planar configuration. A configuration inherits the relations from its ancestor nodes, and can be viewed as a Markov networks (or deformable templates [19]) with reconfigurable neighborhood. We introduce a mixed random field model [20] to represent the configurations. The mixed random field extends conventional Markov random field models by allowing address variables and handles non-local connections caused by occlusions. In this generative model, a configuration corresponds to a primal sketch graph [31].
4. *The visual vocabulary.* Due to scaling property, the terminal nodes could appear at all levels of the And–Or graph. Each terminal node takes instances from certain set. The set is called a dictionary and contains image patches of various complexities. The elements in the set may be indexed by variables such as its type, geometric transformations, deformations, appearance changes etc. Each patch is augmented with anchor points and open bond to connect with other patches.
5. *The language* of a grammar is the set of all possible valid configurations produced by the grammar. In stochastic grammar, each configuration is associated with a probability. As the And–Or graph is directed and recursive, the sub-graph underneath any node A can be considered a sub-grammar for the concept represented by node A. Thus a sub-language for node A is the set of all valid configurations produced by the And–Or graph rooted at A. For example, if A is an object category, say a car, then this sub-language defines all the valid

configurations of car. In an exiting case, the sub-language of a terminal node contains only the atomic configurations and thus is called a dictionary.

In comparison, an element in a dictionary is an atomic structure and an element in a language is a composite structure (or configuration) made of a number of atomic structures. A configuration of node A in zoomed-out view loses its resolution and details, and becomes an atomic element in the dictionary of node A. For example, a car viewed in close distance is a configuration consisting of many parts and primitives. But in far distance, a car is represented by a small image patch as a whole and is not decomposable. This is a special property of the image grammar. The perceptual transition over scales is studied in [80, 84].

1.3.2 Overview of the Dataset and Learning

Now we briefly overview the learning and computing issues with stochastic image grammars.

A foremost question that one may ask is: how do you build this grammar and where is the dataset? Collecting the dataset for learning and training is perhaps more challenging than the learning task itself.

Although fully automated learning is most ideal, for example, let a computer program watch Disney cartoon or Hollywood movies and hope it figures out all the object categories and relations. But purely unsupervised learning is less practical for learning the structured compositional models at present for two reasons. (i) Visual learning must be guided by objectives and purposes of vision, not purely based on statistical information. Ideally one has to integrate this automatic learning process with autonomous robot and AI reasoning at the higher level. Before the robotics and AI systems are ready, we should guide the learning process with some human supervision. For example, what are important structures and what are decorative stuff. (ii) In almost all the unsupervised learning methods, the trainers still have to select their data carefully to contrast the involved concepts. For example, to learn the concept that a car has doors, we must select images of cars with doors both open and closed. Otherwise the concept of door cannot be learned.

We propose to learn the image grammar in a semi-automatic way. We shall start with a supervised learning with manually annotated images and objects to produce the parse graphs. We use this dataset to initiate the process and then shift to weakly supervised learning. This initial dataset is still very large if we target thousands of object categories.

To make the large scale grammar learning framework practical, the first author founded an independent non-profit research institute which started to operate in the summer of 2005.[5] It has a full time annotation team for parsing the image structures and a development team for the annotation tools and database construction. Each image or object is parsed, semi-automatically, into a *parse graph* where the relations are specified and objects are names using the wordnet standard. Figure 1.4 lists an inventory of the current ground truth dataset parsed at LHI. It has now over 500,000 images (or video frames) parsed, covering 280 object categories. Figure 1.5 shows two examples — the parse trees of cat and car. For clarity we only show the parse trees with naming of the nodes. Beyond the object parsing, there are many scene images annotated with the objects and their spatial relations labeled. As stated in a report [87], this ground truth annotation is aimed at broader scope and more hierarchic structures than other datasets collected in various groups, such as Berkeley [4, 50], Caltech [16, 29], and MIT [62].

With this annotated dataset, we can construct the And–Or graph for object and scene categories and learn the probability model on the And–Or graphs. These learning steps are guided by a minimax entropy learning scheme [90] and maximum likelihood estimation. It is divided into three parts:

1. Learning the probabilities at the Or-node so that the configurations generated account for the natural co-occurrence frequency. This is typical in stochastic context free grammars [10].
2. Learning and pursuing the Markov models on the horizontal links and relations to account for the spatial relations, as well

[5] It is called the Lotus Hill Research Institute (LHI) in China (www.lotushill.org).

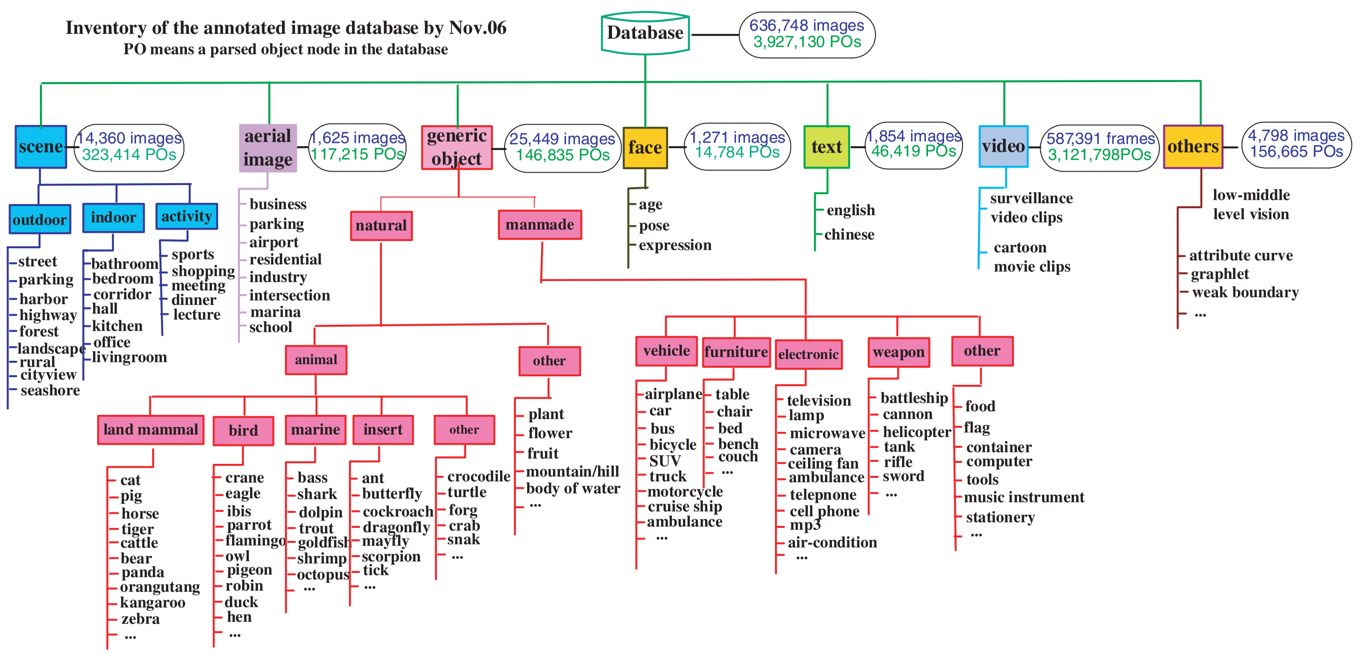


Fig. 1.4 Inventory of the current human annotated image database from Lotus Hill Research Institute for learning and testing. From [87]. A large set of human annotated images and video ground truth is available at the website www.imageparsing.com.

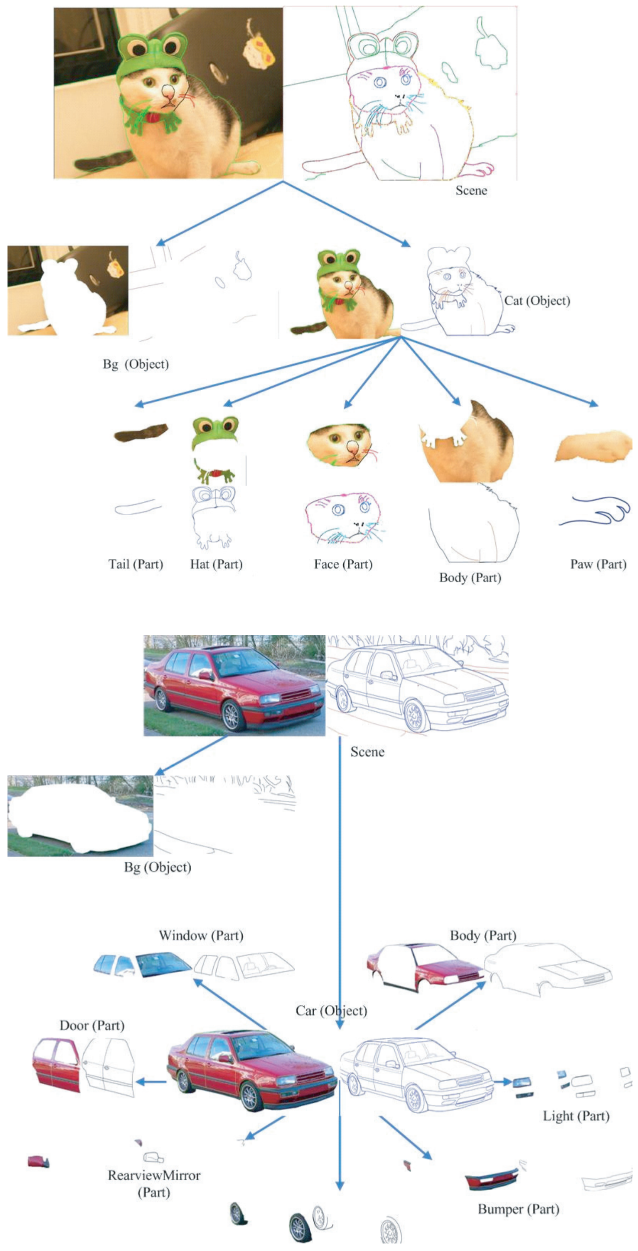


Fig. 1.5 Two examples of the parse trees (cat and car) in the Lotus Hill Research Institute image corpus. From [87].

as consistency of appearance between nodes in the And–Or graphs. This is similar to the learning of Markov random fields [90], except that we are dealing with a dynamic graphical configuration instead of a fixed neighborhood.

3. Learning the And–Or graph structures and dictionaries. The terminal nodes are learned through clustering and the non-terminal nodes are learned through binding. We only briefly discuss this issue in this paper as the current literature has not made significant progress in this part.

The proposed stochastic context sensitive grammar (SCSG) combines the reconfigurability of SCFG with the contextual constraints of graphical (MRF) models, and has the following properties: (a) Compositional power for representing large intra-class structural variations. The grammar can generate a huge number of configurations (i.e., its language) for scenes and objects by composing a relatively much smaller vocabulary. All are represented in graphical configurations. The language of the grammar is the set of all valid configurations of a category, such as furniture, clothes, vehicles, etc. Thus it has enormous expressive power. (b) Recursive structures for scalable computing. The grammar is embodied into an And–Or graph which has recursive structure. The latter is easy to scale in terms of increasing the number of object categories or augmenting more levels (e.g., scene nodes). Consequently the inference algorithms is also recursively defined. We only need to write general top-down and bottom-up functions for a common And–Or node, and re-use the code for all nodes in the And–Or graph. (c) Small sample for effective learning. Due to explicit composition and part-sharing between categories, the state spaces for all object categories are decomposed into products of subspaces of lower dimensions for the vocabulary and relations. Thus we need relatively smaller number of training examples (20–100 instances) for each category. In recent experiments (see Figure 2.6), we can sample the learned object model to generate novel object configurations for generalization, and observe remarkable (over 15% improvement in object category) recognition tasks.

The rest of the paper is organized in the following way. We first discuss in Chapter 2 the background of stochastic grammar, its formulation, the new issues of image grammar in contrast to language grammar, and previous work on image grammar. Then we present the grammar and And–Or graph representation in Chapters 3–6 sequentially: the visual grammar, the relations and configurations, the parse graphs, and finally the And–Or graph. The learning algorithm and results are discussed in Chapter 7, which is followed by the top-down/bottom-up inference algorithm in Chapter 8, and three case studies in Chapter 9. Finally, we raise a number of unsolved problems in Chapter 10 to conclude the paper.

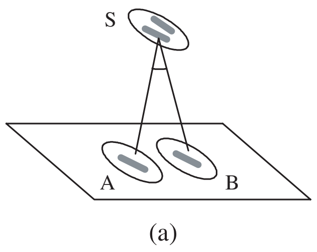
2 Background

2.1 The Origin of Grammars

The origin of grammar in real-world signals, either language or vision, is that certain parts of a signal s tend to occur together more frequently than by chance. Such co-occurring elements can be grouped together forming a higher order part of the signal and this process can be repeated to form increasingly larger parts. Because of their higher probability, these parts are found to re-occur in other similar signals, so they form a vocabulary of "reusable" parts. A basic statistical measure, which indicates whether something is a good part, is a quantity which measures, in bits, the strength of binding of two parts $s|_A$ and $s|_B$ of the signal s:

$$\log_2\left(\frac{p(s|_{A\cup B})}{(p(s|_A)\cdot p(s|_B))}\right). \tag{2.1}$$

Two parts of a signal are bound if the probability of their co-occurrence is significantly greater than the probability if their occurence was independent. The classic example which goes back to Laplace is the sequence of 14 letters "CONSTANTINOPLE": these occur much more frequently in normal text than in random sequences of the 26 letters



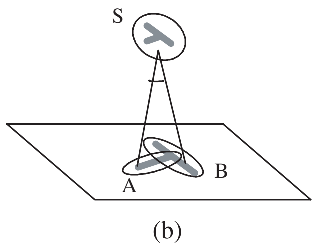


Fig. 2.1 (a) Two parallel lines form a reusable part containing as its constituents the two lines, (b) A T-junction is another reusable part formed from two lines.

in which the letters are chosen independently, even with their standard frequencies. In this example, the composite part is a word, its constituents are letters. A more elaborate example from vision is shown in Figure 2.1. On the left, this illustrates how nearby lines tend to be parallel more often than at other mutual orientations, hence a pair of parallel lines forms a reusable part. On the right, we see how another frequent configuration is when the two lines are roughly perpendicular and touch forming a "T-junction."

The set of reusable parts that one identifies in some class of signals, e.g., in images, is called the *vocabulary* for this class of signals. Each such reusable part has a name or label. In language, a noun phrase, whose label is "NP" is a common reusable part, an element of the linguistic vocabulary. In vision, a face is a clear candidate for such a very high-level reusable part. The set of such parts which one encounters in analyzing statistically a specific signal is called the *parse graph* of the signal. Abstractly, one first associates to a signal $s : D \to I$ the set of subsets $\{A_i\}$ of D such that $s|_{A_i}$ is a reusable part. Then these subsets are made into the vertices or nodes $\langle A_i \rangle$ of the parse graph. In the graph, the proper inclusion of one subset in another, $A_i \subsetneqq A_j$, is shown by a "vertical" directed edge $\langle A_j \rangle \to \langle A_i \rangle$. For simplicity, we prune redundant edges in this graph, adding edges only when $A_i \subsetneqq A_j$ and there is no A_k such that $A_i \subsetneqq A_k \subsetneqq A_j$.

In the ideal situation, the parse graph is a tree with the whole signal at the top and the domain D (the letters of the text or the pixels of the image) at the bottom. Moreover, each node $\langle A_i \rangle$ should be the disjoint union of its children, the parts $\{A_j | A_j \subsetneqq A_i\}$. This is the case for the simple parse trees of Figure 2.1 or in most sentences, such as the ones shown below in Figure 2.6.

2.2 The Traditional Formulation of Grammar

The formal idea of grammars goes back to Panini's Sanskit grammar in the first millenium BCE, but its modern formalization can be attributed to Chomsky [11]. Here one finds the definition making a grammar into a 4-tuple $\mathcal{G} = (V_N, V_T, \mathrm{R}, S)$, where V_N is a finite set of non-terminal nodes, V_T a finite set of terminal nodes, $S \in V_N$ is a start symbol at the root, and R is a set of production rules,

$$\mathrm{R} = \{\gamma : \alpha \to \beta\}. \tag{2.2}$$

One requires that $\alpha, \beta \in (V_N \cup V_T)^+$ are strings of terminal or non-terminal symbols, with α including at least a non-terminal symbol.[1] Chomsky classified languages into four types according to the form of their production rules. A type 3 grammar has rules $A \to aB$ or $A \to a$, where $a \in V_T$ and $A, B \in V_N$. It is also called a finite state or regular grammar. A type 2 grammar has rules $A \to \beta$ and is called a context free grammar. A type 1 grammar is context sensitive with rules $\xi A \eta \to \xi \beta \eta$ where a non-terminal node A is rewritten by β in the context of two strings ξ and η. The type 0 grammar is called a phrase structure or free grammar with no constraint on α and β.

The set of all possible strings of terminals ω derived from a grammar $\mathcal{G}$ is called its *language*, denoted by

$$\mathbf{L}(\mathcal{G}) = \left\{\omega : \; S \overset{\mathrm{R}^*}{\Longrightarrow} \omega, \, \omega \in V_T^*\right\}. \tag{2.3}$$

R^* means a sequence of production rules deriving ω from S, i.e.,

$$S \overset{\gamma_1, \gamma_2, \ldots, \gamma_{n(\omega)}}{\Longrightarrow} \omega \tag{2.4}$$

If the grammar is of type 1, 2, or 3, then given a sequence of rules generating the terminal string ω, we obtain a parse tree for ω, denoted by

$$\mathbf{pt}(\omega) = (\gamma_1, \gamma_2, \ldots, \gamma_{n(\omega)}), \tag{2.5}$$

[1] V^* means a string consisting of $n \geq 0$ symbols from V, and V^+ means a string with $n \geq 1$ symbols from V.

if each production rule creates one node labeled by its head A and a set of vertical arrows between A and each symbol in the string β. To relate this to the general setup of the previous section, note that each node has a set of ultimate descendents in the string ω. This is to be a reusable part. If we give this part the label $A \in V_N$, we see that the tree can equally well be generated by taking these parts as nodes and putting in vertical arrows when one part contains another with no intermediate part. Thus the standard Chomskian formulation is a special case of our general setup.

As is illustrated in Figure 2.4, the virtue of the grammar lies in its expressive power of generating a very large set of valid sentences (or strings), i.e., its language, through a relatively much smaller vocabulary V_T, V_N and production rules R. Generally speaking, the following inequality is often true in practice,

$$|\mathbf{L}(\mathcal{G})| \gg |V_n|, |V_T|, |\mathrm{R}|. \tag{2.6}$$

In images, V_T can be pixels, but here we will find it more convenient to make it correspond to a simple set of local structures in the image, textons, and other image primitives [30, 31]. Then V_N will be reusable parts and objects in the image, and a production rule $A \to \beta$ is a template which enables you to expand A. Then the $\mathbf{L}(\mathcal{G})$ will be the set of all valid object *configurations*, i.e., scenes. The grammar rules represent both structural regularity and flexibility. The structural regularity is enforced by the template which decomposes an entity A, such as object into certain elements in β. The structural flexibility is reflected by the fact that each structure A has many alternative decompositions.

In this paper, we will find it convenient to describe the entire grammar by one universal *And–Or tree*, which contains all parsings as subtrees. In this tree, the Or-nodes are labeled by $V_N \cup V_T$ and the And-nodes are labeled by production rules R. We generate this tree recursively, starting by taking start symbol as a root which is an Or-node. We proceed as follows: wherever we have an Or-node with non-terminal label A, we consider all rules which have A on the left and create children which are And-nodes labeled by the corresponding rules. These in turn expand to a set of Or-nodes labeled by the symbols on the right of the rule. An Or-node labeled by a non-terminal does

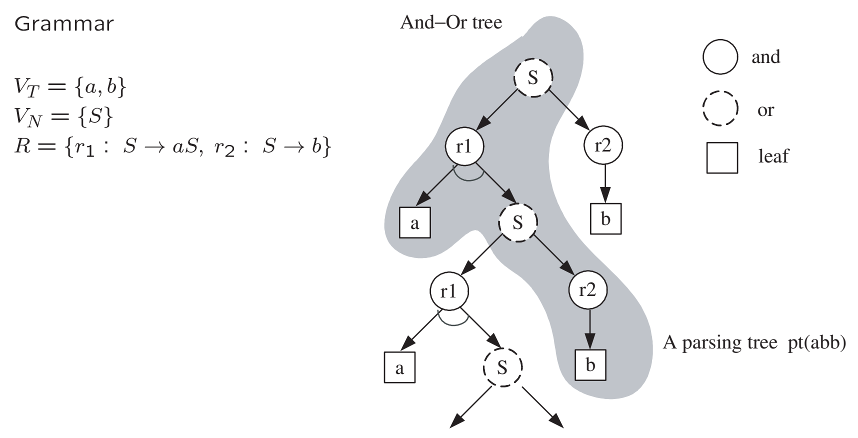


Fig. 2.2 A very simple grammar, its universal And–Or tree and a specific parse tree in shadow.

not expand further. Clearly, all specific parse trees will be contained in the universal And–Or tree by selecting specific children for each Or-node reached when descending the tree. This tree is often infinite. An example is shown in Figure 2.2.

A vision example of an And–Or tree, using the reusable parts in Figure 2.1, is shown in Figure 2.3. A, B, C are non-terminal nodes and

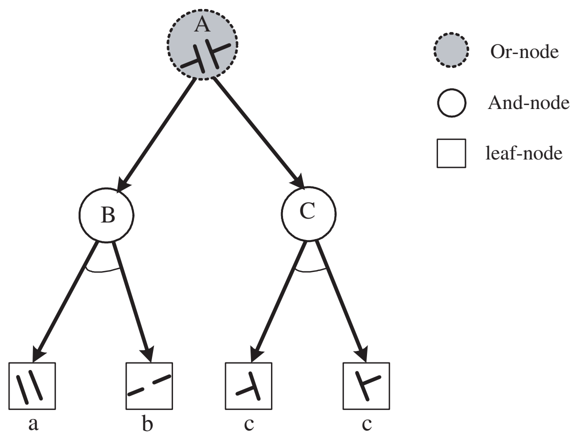


Fig. 2.3 An example of binding elements a, b, c into a larger structures A in two alternative ways, represented by an And–Or tree.

a, b, c are terminal or leaf nodes. B, C are the two ambiguous ways to interpret A. B represents an occlusion configuration with two layers while C represents a butting/alignment configuration at one layer. The node A in Figure 2.3 is a frequently observed local structure in natural images when a long bar (e.g., a tree trunk) occludes a surface boundary (e.g., a fence).

The expressive power of an And–Or tree is illustrated in Figure 2.4. On the left is an And-node A which has two components B and C. Both B and C are Or-nodes with three alternatives shown by the six leaf nodes. The 6 leaf nodes can compose a set of configurations for node A, which is called the "language" of A – denoted by $\mathbf{L}(A)$. Some of the valid configurations are shown at the bottom. The power of composition is crucial for representing visual concepts which have varying structures, for example, if A is an object category, such as car or chair, then $\mathbf{L}(A)$ is a set of valid designs of cars or chairs. The expressive power of the And–Or tree rooted at A is reflected in the ratio of the total number of configurations that it can compose over the number of nodes in the And–Or tree. For example, Figure 2.4(b) shows two levels of And-nodes and two levels of Or-nodes. Both have branch factor

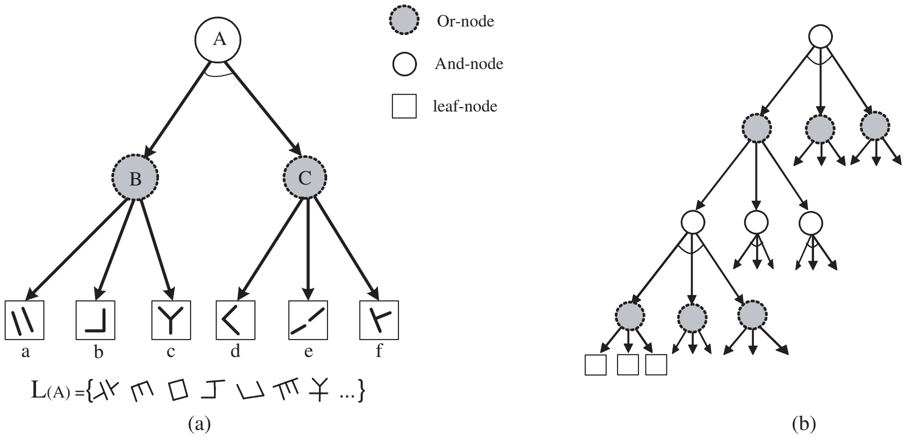


Fig. 2.4 (a) An And-node A is composed of two Or-nodes B and C, each of which includes three alternative leaf nodes. The 6 leaf nodes can compose a set of configurations for node A, which is called the "language" of A. (b) An And–Or tree (5-level branch number = 3) with 10 And-nodes, 30 Or-nodes, and 81 leaf nodes, can produce $3^{12} = 531,441$ possible configurations.

$b = 3$. This tree has a total of 10 And-nodes, 30 Or-nodes, and 81 leaf nodes, the number of possible structures is $(3 \times 3^3)^3 = 531{,}441$, though some structures may be repeated.

In Section 2.6, we shall discuss three major differences between vision grammars and language grammars.

2.3 Overlapping Reusable Parts

As mentioned, in good cases, there are no overlapping reusable parts in the base signal and each part is the disjoint union of its children. But this need not be the case. If two reusable parts do overlap, typically this leads to parse structures with a diamond in them, Figure 2.5 is an example. Many sentences, for example, are ambiguous and admit two reasonable parses. If there exists a string $\omega \in \mathbf{L}(\mathcal{G})$ that has more than one parse tree, then $\mathcal{G}$ is said to be an *ambiguous grammar*. For example, Figure 2.6 shows two parse trees for a classic ambiguous sentence (discussed in [26]). Note that in the first parse, the reusable part “saw the man” is singled out as a verb phrase or VP; in the second, one finds instead the noun phrase (NP) “the man with the telescope.” Thus the base sentence has two distinct reusable parts which overlap in “the man.” Fixing a specific parse eliminates this complication. In context, the sentence is always spoken with only one of these meanings, so one parse is right, one is wrong, one reusable part is accepted, one is rejected. If we reject one, the remaining parts do not overlap.

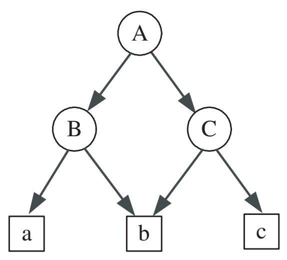


Fig. 2.5 Parts sharing and the diamond structure in And–Or graphs.

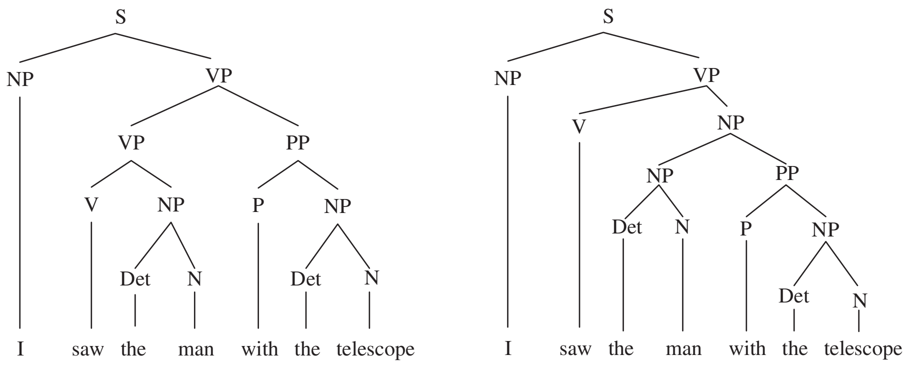

S
NP
VP
VP
PP
V
NP
P
NP
Det
N
Det
N
I saw the man with the telescope
S
NP
VP
V
NP
NP
PP
Det
N
P
NP
Det
N
I saw the man with the telescope

Fig. 2.6 An example of ambiguous sentence with two parse trees. The non-terminal nodes S, V, NP, VP denotes sentence, verbal, noun phrase, and verbal phrase, respectively. Note that if the two parses are merged, we obtain a graph, not a tree, with a "diamond" in it as above.

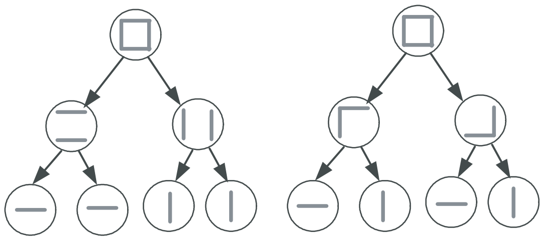
The above is, however, only the simplest case where reusable parts overlap. Taking vision, there seem to occur an overlap in four ways.

1. Ambiguous scenes where distinct parses suggest themselves.
2. High level patterns which incorporate multiple partial patterns.
3. "Joints" between two high level parts where some sharing of pixels or edges occurs.
4. Occlusion where a background object is completed behind a foreground object, so the two objects overlap.

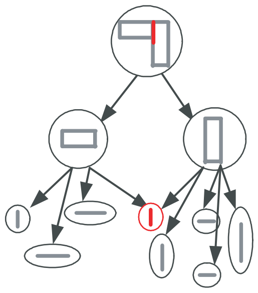
A common cause of ambiguity in images is when there is an accidental match of color across the edge of an object. An example is shown in Figure 2.7(a): the man's face has similar color to the background and, in fact, the segmenter decided the man had a pinnocio-like nose. The true background and the false head with large nose overlap. As in the linguistic examples, there is only "true" parse and the large nose part should be rejected.

An example of the second is given by a square (or by many alphanumeric characters). A square may be broken up into two pairs of parallel lines. A pair of parallel lines is a common reusable part in its own right, so we may parse the square as having two child nodes, each

(a)



(b)



(c)

(d)

Fig. 2.7 Four types of images in which “reusable parts” overlap. (a) The pinnocio nose is a part of the background whose gray level is close to the face, so it can be grouped with the face or the background. This algorithm chose the wrong parse. (b) The square can be parsed in two different ways depending on which partial patterns are singled out. Neither parse is wrong but the mid-level units overlap. (c) The two halves of a butt joint have a common small edge. (d) The reconstructed complete sky, trees and field overlap with the face.

such a pair. But the square is also built up from 4 line pairs meeting in a right angle. Such pairs of lines also form common reusable parts. The two resulting parses are shown in Figure 2.7(b). One “solution” to this issue is to choose, once and for all, one of these as the preferred parse for a square. In analyzing the image, both parses may occur but, in order to give the whole the “square” label, one parse is chosen and the other parts representing partial structures are rejected.

“Joints” will be studied below: often two parts of the image are combined in characteristic geometric ways. For example, two thin rectangles may butt against each other and then form a compound part. But clearly, they share a small line segment which is common to both their boundaries: see Figure 2.7(c). If the parsing begins at the pixel level, such sharing between adjacent parts is almost inevitable. The simplest way to restore the tree-like nature of the parse seems to be to duplicate the overlapping part. For example, an edge is often part of the structure on each side and it seems very natural to allocate to the edge two nodes — the edge attached to side 1 and the edge attached to side 2.

The most vision-specific case of overlap is caused by occlusion. Occlusion is seen in virtually every image. It can be modeled by what the second author has called the 2.1D sketch. Mentally, humans (and presumably other visual animals) are quite aware that two complete objects exist in space but that certain parts of the two objects project to the same image pixels, with only one being visible. Here we consciously form duplicate image planes carrying the two objects: this is crucial when we actually want to use our priors to reconstruct as much as possible of the occluded object. It seems clear that the right parse for such objects should add extra leaves at the bottom to represent the occluded object. The new leaves carry colors, textures etc. extrapolated from the visible parts of the object. Their occluded boundaries were what the gestalt school called *amodal* contours. The gestalt school demonstrated that people often make very precise predictions for such amodal contours.

Below we will assume that the reusable parts do not overlap so that inclusion gives us a tree-like parse structure. This simplifies immensely the computational algorithms. Future work may require dealing with diamonds more carefully (REF Geman).

2.4 Stochastic Grammar

To connect with real-world signals, we must augment grammars with a set of probabilities $\mathcal{P}$ as a fifth component. For example, in a stochastic context free grammar (SCFG) — the most common stochastic grammar in the literature, suppose $A \in V_N$ has a number of alternative rewriting rules,

$$A \to \beta_1 \,|\, \beta_2 \,|\, \cdots \,|\, \beta_{n(A)}, \quad \gamma_i : A \to \beta_i. \tag{2.7}$$

Each production rule is associated with a probability $p(\gamma_i) = p(A \to \beta_i)$ such that:

$$\sum_{i=1}^{n(A)} p(A \to \beta_i) = 1. \tag{2.8}$$

This corresponds to what is called a random branching process in statistics [2]. Similarly a stochastic regular grammar corresponds to a Markov chain process.

The probability of a parse tree is defined as the product,

$$p(\mathbf{pt}(\omega)) = \prod_{j=1}^{n(\omega)} p(\gamma_j). \tag{2.9}$$

The probability for a string (in language) or configuration (in image) $\omega \in \mathbf{L}(\mathcal{G})$ sums over the probabilities of all its possible parse trees.

$$p(\omega) = \sum_{\mathbf{pt}(\omega)} p(\mathbf{pt}(\omega)). \tag{2.10}$$

Therefore a stochastic grammar $\mathcal{G} = (V_N, V_T, \mathrm{R}, S, \mathcal{P})$ produces a probability distribution on its language

$$\mathbf{L}(\mathcal{G}) = \Big\{(\omega, p(\omega)) : \ S \overset{\mathrm{R}^*}{\Longrightarrow} \omega, \ \omega \in V_T^*\Big\}. \tag{2.11}$$

A stochastic grammar is said to be *consistent* if $\sum_{\omega \in \mathbf{L}(\mathcal{G})} p(\omega) = 1$. This is not necessarily true even when Equation (2.8) is satisfied for each non-terminal node $A \in V_N$. The complication is caused by cases when there is a positive probability that the parse tree may not end in a finite number of steps. For example, if we have a production rule that expands A to AA or terminates to a, respectively,

$$A \to AA \,|\, a \quad \text{with prob. } \ \rho \,|(1-\rho)$$

If $\rho > \frac{1}{2}$, then node A expands faster than it terminates, and it keeps replicating. This poses some constraints for designing the set of probabilities $\mathcal{P}$.

The set of probabilities $\mathcal{P}$ can be learned in a supervised way from a set of observed parse trees $\{\mathbf{pt}_m, m = 1, 2, \ldots, M\}$ by maximum likelihood estimation,

$$\mathcal{P}^* = \arg\max \prod_{m=1}^{M} p(\mathbf{pt}_i). \tag{2.12}$$

The solution is quite intuitive: the probability for each non-terminal node A in (2.7) is

$$p(A \to \beta_i) = \frac{\#(A \to \beta_i)}{\sum_{j=1}^{n(A)} \#(A \to \beta_j)}. \tag{2.13}$$

In the above equation, $\#(A \to \beta_i)$ is the number of times a rule $A \to \beta_i$ is used in all the M parse trees. In an unsupervised learning case, when the observation is a set of strings without parse trees, one can still follow the ML-estimation above with an EM-algorithm. It was shown in [10] that the ML-estimation of $\mathcal{P}$ can rule out infinite expansion and produce a consistent grammar.

In Figure 2.3, one can augment the two parses by probabilities ρ and $1 - \rho$, respectively. We write this as a stochastic production rule:

$$\mathrm{A} \to \mathrm{a} \cdot \mathrm{b} \,|\, \mathrm{c} \cdot \mathrm{c}; \quad \rho | (1 - \rho). \tag{2.14}$$

Here "|" means an alternative choice and is represented by an "Or-node." "·" means composition and is represented by an "And-node" with an arc underneath. One may guess that the interpretation B has a higher probability than C, i.e., $\rho > 1 - \rho$ in natural images.

2.5 Stochastic Grammar with Context

In the rest of this paper, we shall use an And–Or tree defined by a stochastic grammar but we will augment it to an And–Or graph by adding relations and contexts as horizontal links. The resulting probabilistic models are defined on the And–Or graph to represent a stochastic context *sensitive* grammar for images.

A simple example of this in language, due to Mark, Miller and Grenander augments the stochastic grammar models with word co-occurrence probabilities. Let $\omega = (\omega_1, \omega_2, \ldots, \omega_n)$ be a sentence with n words, then bi-gram statistics counts the frequency $h(\omega_i,\ \omega_{i+1})$ and all word pairs, and therefore leads to a simple Markov chain model for the string ω:

$$p(\omega) = h(\omega_1)\prod_{i=1}^{n-1} h(\omega_{i+1}|\omega_i). \qquad (2.15)$$

In [48], a probabilistic model was proposed to integrate parse tree model in (2.9) and the bi-gram model in (2.15) for the terminal string, by adding factors $h^*(\omega_i,\omega_{i+1})$ and re-normalizing the probability:

$$p(\mathbf{pt}(\omega)) = \frac{1}{Z}h^*(\omega_1)\prod_{i=1}^{n-1} h^*(\omega_{i+1},\omega_i)\cdot\prod_{j=1}^{n(\omega)} p(\gamma_j). \qquad (2.16)$$

The factors are chosen so that the marginal probability on word pairs matches the given bi-gram model. Note that one can always rewrite the probability in a Gibbs form for the whole parse tree and strings,

$$p(\mathbf{pt}(\omega);\Theta) = \frac{1}{Z}\exp\left\{-\sum_{j=1}^{n(\omega)}\lambda(\gamma_j) - \sum_{i=1}^{n-1}\lambda(\omega_{i+1},\omega_i)\right\}, \qquad (2.17)$$

where $\lambda(\gamma_j) = -\log p(\gamma_j)$ and $\lambda(\omega_{i+1}|\omega_i) = -\log h^*(\omega_{i+1}|\omega_i)$ are parameters included in Θ. Thus the existence of the h^* is a consequence of the existence of exponential models matching given expectations.

However, the left-to-right sequence of words may not express the strongest contextual effects. There are non-local relations as the arrows in Figure 2.8 show. First interjections mess up phrases in language. The italicized words in the sentence split the text flow. Thus the "next" relation in the bi-gram is not deterministically decided by the word order but has to be inferred. Second the word "what" is both the object of the verb "said" and the subject of the verb "is." It connects the

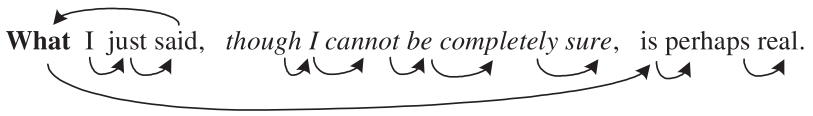

What I just said, though I cannot be completely sure, is perhaps real.

Fig. 2.8 An English sentence with non-local "next" relations shown by the arrows and the word "what" is a joint to link two clauses.

two clauses together. Quite generally, all pronouns indicate long range dependencies, link two reusable parts and carry context from one part of an utterance or text to another. In images one shall see many different types of joints that combine parts of objects, such as butting, hinge, and various alignments that similarly link two reusable parts. As we shall discuss in a later section, each node may have many types of relations in the way it interacts with other nodes. These relations are often hidden or cannot be deterministically decided and thus we shall represent these potential connections through some “address variables” associated with each node. The value of an address variable in a node ω_i is an index toward another node ω_j, and the node pair (ω_i, ω_j) observes a certain relation. These address variables have to be computed along with the parse tree in inference.

In vision, these non-local relations occur much more frequently. These relationships represent the spatial context at all levels of vision from pixels and primitives to parts, objects and scenes, and lead to various graphical models, such as Markov random fields. Gestalt organizations are popular examples in the middle level and low-level vision. For example, whenever a foreground object occludes part of a background object, with this background object being visible on both sides of the foreground one, these two visible parts of the background object constrain each other. Other non-local connections may reflect functional relations, such as object X is “supporting” object Y.

2.6 Three New Issues in Image Grammars in Contrast to Language

As we have seen already, an image grammar should include two aspects: (i) The hierarchic structures (the grammar $\mathcal{G}$) which generate a large set of valid image *configurations* (i.e., the language $\mathbf{L}(\mathcal{G})$). This is especially important for modeling object categories with large intra-class structural variabilities. (ii) And the context information which makes sure that the components in a configuration observe good spatial relationships between object parts, for example, relative positions, ratio of sizes, and consistency of colors. Both aspects encode important parts of our visual knowledge.

Going from 1D language grammars to 2D image grammars is non-trivial and requires a major leap in technology. Perhaps more important than anything else, one faces enormous complexity, although the principles are still simple. The following section summarizes three major differences (and difficulties) between the language grammars and image grammars.

The first huge problem is the loss of the left-to-right ordering in language. In language, every production rule $A \to \beta$ is assumed to generate a linearly ordered sequence of nodes β and following this down to the leaves, we get a linearly ordered sequence of terminal words. In vision, we have to replace the implicit links of words to their left and right neighbors by the edges of a more complex "region adjacency graph" or RAG. To make this precise, let the domain D of an image I have a decomposition $D = \cup_{k \in S} R_k$ into disjoint regions. Then we make an RAG with nodes $\langle R_i \rangle$ and edges $\langle R_k \rangle$ — $\langle R_l \rangle$ whenever R_k and R_l are adjacent. This means we must *explicitly add horizontal edges to our parse tree to represent adjacency.* In a production rule $A \to \beta$, we no longer assume the nodes of β are linearly ordered. Instead, we should make β into a *configuration*, that is, a set of nodes from $V_N \cup V_T$ plus horizontal edges representing adjacency. We shall make this precise below.

Ideas to deal with the loss of left-to-right ordering have been proposed by the K. S. Fu school of "syntactic pattern recognition" under the names "web grammars" and "plex grammars" [22], by Grenander in his pattern theory [28], and more recently by graph grammars for diagram interpretation in computer science [60]. These ideas have not received enough attention in vision. We need to study the much richer spatial relations for how object and parts are connected. Making matters more complex, due to occlusions and other non-local groupings, non-adjacent spatial relations often have to be added in the course of parsing.

One immediate consequence of the lack of natural ordering is that a region has very ambiguous production rules. Let A be a region and a an atomic region, and let the production rules be $A \to aA \,|\, a$. A linear region $\omega = (a, a, a, \ldots, a)$ has a unique parse graph in left-to-right ordering. With the order removed, it has a combinatorial number of parse

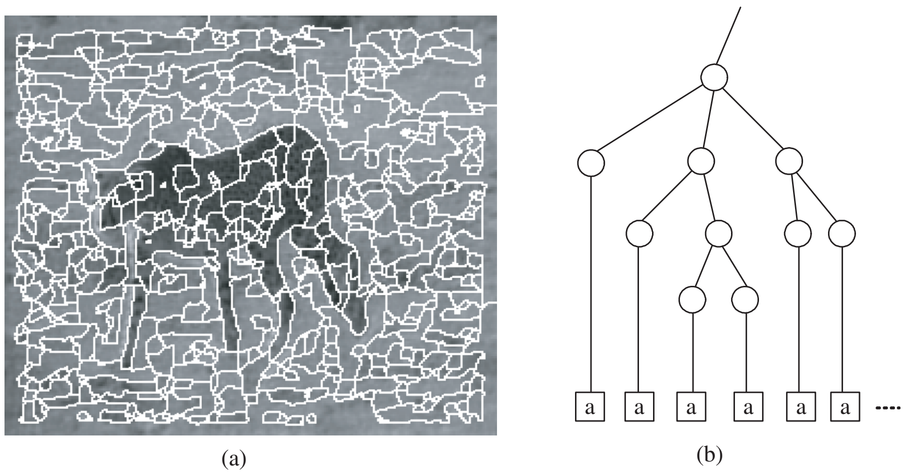


Fig. 2.9 A cheetah and the background after local segmentation: both can be described by an RAG. Without the left-to-right order, if the regions are to be merged one at a time, they have a combinatorially explosive number of parse trees.

trees. Figure 2.9 shows an example of parsing an image with a cheetah. It becomes infeasible to estimate the probability $p(\omega)$ by summing over all these parse trees in (2.10).

Therefore we must avoid these recursively defined grammar rules $A \to aA$, and treat the grouping of atomic regions into one large region A as a single computational step, such as the grouping and partitioning in a graph space [3]. Thus the probability $p(\omega)$ is assigned to each object as a whole instead of the production rules. In the literature, there are a number of hierarchic representations by an adaptive image pyramid, for example, the work by Rosenfeld and Hong in the early 80s [34], and the multi-scale segmentation by Galun et al. [23]. Though generic elements are grouped in these works, there are no explicit grammar rules. We shall distinguish such multi-scale pyramid representation from parse trees.

The second issue, unseen in language grammar, is the issue of image scaling [45, 80, 82]. It is a unique property of vision that objects appear at arbitrary scales in an image when the 3D object lies nearer or farther from the camera. You cannot hear or read an English sentence at multiple scales, but the image grammar must be a multi-resolution

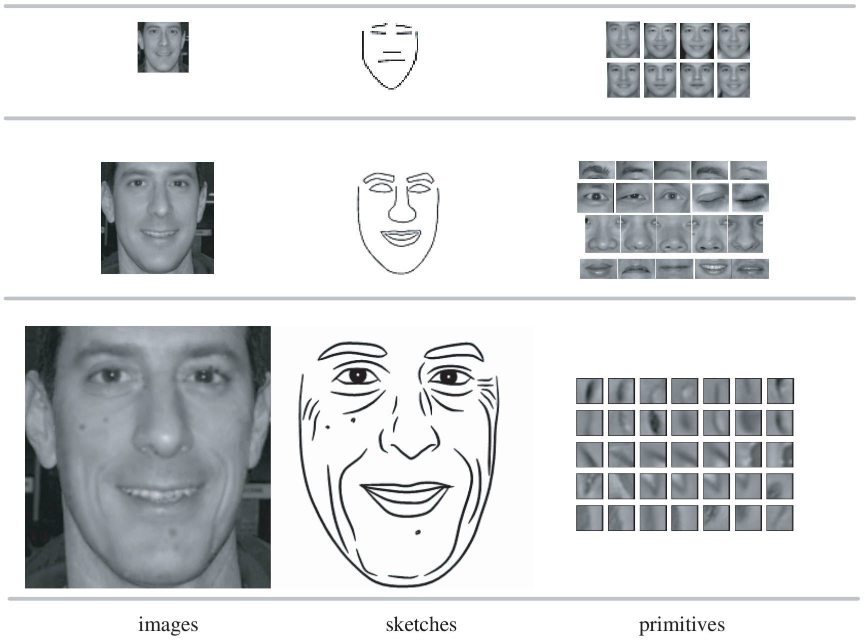


Fig. 2.10 A face appears at three resolutions is represented by graph configurations in three scales. The right column shows the primitives used at the three levels.

representation. This implies that the parse tree can terminate immediately at any node because no more detail is visible.

Figure 2.10 shows a human face in three levels from [85]. The left column shows face images at three resolutions, the middle column shows three configurations (graphs) of increasing detail, and the right column shows the dictionaries (terminals) used at each resolution, respectively. At a low resolution, a face is represented by patches as a whole (for example, by principle component analysis), at a middle resolution, it is represented by a number of parts, and at a higher resolution, the face is represented by a sketch graph using smaller image primitives. The sketch graphs shown in the middle of Figure 2.10 expands with increasing resolution. One can account for this by adding some termination rules to each non-terminal node, e.g., each non-terminal node may exit the production for a low resolution case.

$$\forall A \in V_N, \quad A \to \beta_1 \,|\cdots|\beta_{n(A)}\,|\,t_1\,|\,t_2\,|, \tag{2.18}$$

where $t_1, t_2, \in V_T$ are image primitives or image templates for A at certain scales. For example, if A is a car, then t_1, t_2 are typical views (small patches) of the car at low resolution. As they are in low resolution, the parts of the cars are not very distinguishable and thus are not represented separately. The decompositions $\beta_i, i = 1, 2, \ldots, n(A)$ represent the production rules for higher resolutions, so this new issue does not complicate the grammar design, except that one must learn the image primitives at multiple scales in developing the visual vocabulary.

The third issue with image grammars is that natural images contain a much wider spectrum of quite irregular local patterns than in speech signals. Images not only have very regular and highly structured objects which could be composed by production rules, they also contain very stochastic patterns, such as clutter and texture which are better represented by Markov random field models. In fact, the spectrum is continuous. The structured and textured patterns can transfer from one to the other through continuous scaling [80, 84]. The two categories of models ought to be integrated more intimately and melded into a common model. This raises numerous challenges in modeling and learning at all levels of vision. For example, how do we decide when we should develop a image primitive (texton) for a specific element or use a texture description (for example, a Markov Random Field)? How do we decide when we should group objects in a scene by a production rule or by a Markov random field for context?

2.7 Previous Work in Image Grammars

There are four streams of research on image grammars in the vision literature.

The first stream is *syntactic pattern recognition* by K. S. Fu and his school in the late 1970s to early 1980s [22]. Fu depicted an ambitious program for scene understanding using grammars. A block world example is illustrated in Figure 2.11. Similar image understanding systems were also studied in the 1970–1980s [33, 54] The hierarchical representation on the right is exactly the sort of parse graph that we are pursuing today. The vertical arrows show the decomposition of the scene and objects, and the horizontal arrows display some relations, such as

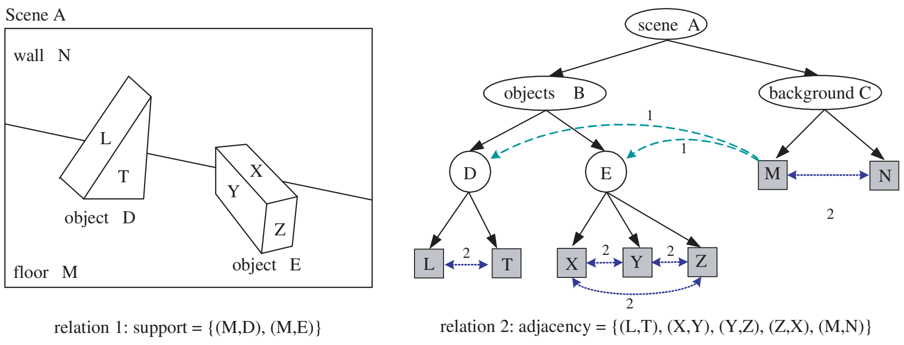

Scene A
wall N
L
T
object D
X
Y
Z
object E
floor M
scene A
objects B
background C
D
E
M
N
L
T
X
Y
Z
1
1
2
relation 1: support = {(M,D), (M,E)}
relation 2: adjacency = {(L,T), (X,Y), (Y,Z), (Z,X), (M,N)}

Fig. 2.11 A parser tree for a block world from [22]. The ellipses represents non-terminal nodes and the squares are for terminal nodes. The parse tree is augmented into a parse graph with horizontal connections for relations, such as one object supporting the other, or two adjacent objects sharing a boundary.

support and adjacency. Fu and collaborators applied stochastic grammars to simple objects (such as diagrams) and shape contours (such as outline of a chromosome). Most of the work remained in 1D structures, although the ideas of web grammars and plex grammars were also studied. This stream was disrupted in the 1980s and suffered from the lack of an image vocabulary that is realistic enough to express real-world objects and scenes, and reliably detectable from images. This remains a challenge today, though much progress has been made recently in appearance based methods, such as PCAs, image primitives, [31], code books [17], fragments and patches [38, 77]. It is worth mentioning that many of these works on patches and fragments do not provide a formalism for composition and that they lack the bond structures studied in this paper.

The second stream are the *medial axis* techniques for analyzing 2D shapes. For animate objects represented by simple closed contours, Blum argued in 1973 [8] that medial axes are an intuitive and effective representation of a shape, in contrast to boundary fragments. Leyton proposed a process grammar approach to these in 1988 [43]. He argued that any shape is a record of motion history, and developed a grammar for the procedure for how a shape grows from a simple object, say a small circle. A shape grammar for shape matching and recognition via medial axes was then developed by Zhu and Yuille in 1996 [91].

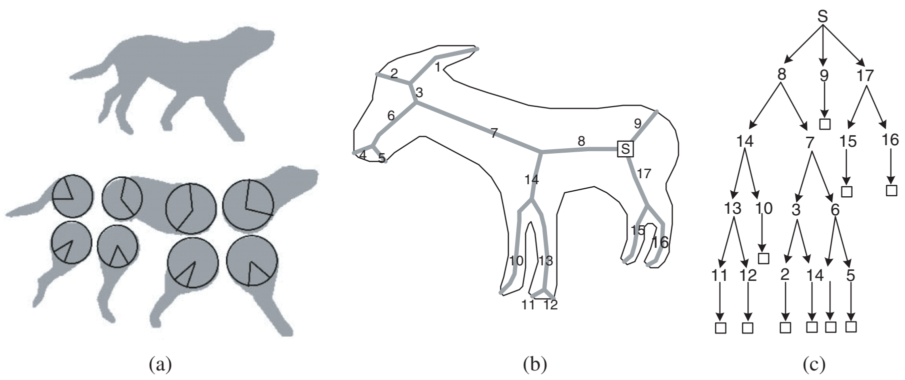


Fig. 2.12 (a) A dog and its decomposition into parts using the medial axis algorithm of [91]. (b) The shock graph of a goat with its shock tree in (c) adopted from [68]. The root of the tree is the node at the "hip" of the goat marked by a square.

An example is shown on the left in Figure 2.12. The dog should be read as a node A in the parse tree and the fragments below it as the child nodes for a production rule that expands the dog into its limbs, trunk, head, and tail. The circles are the maximal circles on which the medial axis is based and allow one to create horizontal arrows between the parts, so that the production yields not merely a set of parts but a configuration.

A formal shock graph was studied by Zucker's school including Dickinson [40], Kimia [67], Siddiqi et al. [41, 64, 68]. They reverse Leyton's growth process by collapsing the shape using the distance transform. The singularities in the process create "shocks," for example, when two sides of the leg of a dog collapse into an axis. Thus different sections of their skeleton are characterized by the types of singularity and record the temporal record of the shape's collapse. Figure 2.12 shows on the right the shock graph of a goat from [68]. The vertical arrows in their shock tree are very different from those in the parse tree. In the shock tree the child nodes are a younger generation that grow from the parent nodes, thus the two graphs have quite different interpretations.

The third stream can be seen as a number of works branching out from the school of *pattern theory*. Grenander [28] defined a regular pattern on a set of graphs which are made from some primitives which he called "generators." Each generator is like a terminal element and has a number of attributes and "bonds" to connect with other generators. Geman and collaborators [6, 27, 36] proposed a more ambitious formulation for compositionality which is quite similar to that developed in this paper. Moreover, they seek to create not only computer vision systems but models of cortical vision mechanisms in animals. In sharp contrast to our approach, they make the overlapping of their reusable parts into a central element of their formalism. This overlapping is used to allow parts to compute their "binding strength" depending on any and all features of this overlap. It is also the key, in their system, to synchronizing the activity of the neurons expressing the higher order parts. As a proof of concept, they applied the compositional system to handwritten upper case letter recognition and to licence plate reading [36]. The work in this paper belongs to this approach, cf. an attribute grammar to parse images of the man-made world [32], and a context sensitive grammar for representing and recognizing human clothes [9]. These will be reviewed in later sections.

Finally, the sparse image coding model can be viewed as an attribute SCFG. In sparse coding [56, 69], an image is made of a number of n independent image bases, and there are a few types of image bases, such as Gabor cosine, Gabor sine, and Laplacian of Gaussian etc. These bases have attributes $\theta = (x, y, \tau, \sigma, \alpha)$ for locations, orientations, scales and contrasts, respectively. This can be expressed as an SCFG. Let S denote a scene, A an image base, and a, b, c the different bases.

$$\begin{aligned}
&S \to A^n, \quad n \sim p(n) \propto e^{-\lambda_o n},\\
&A \to a(\theta)\,|\,b(\theta)\,|\,c(\theta), \quad \theta \sim p(\theta) \propto e^{-\lambda|\alpha|},
\end{aligned}$$

where $p(\theta)$ is uniform for location, orientation and scale. Crouse et al. [13] introduce a Markov tree hierarchy for the image bases and this produces an SCFG.

3

Visual Vocabulary

3.1 The Hierarchic Visual Vocabulary — The "Lego Land"

In English dictionaries, a word not only has a few attributes, such as meanings, number, tense, and part of speech, but also a number of ways to connect with other words in a context. Sometimes the connections are so strong that compound words are created, for example, the word "apple" can be bound with "pine" or "Fuji" to the left, or "pie" and "cart" to the right. For slightly weaker connections, phrases are used, for instance, the work "make" can be connected with "something" using the prepositions "of" or "from," or connected with "somebody" through the prepositions "at" or "against." Figure 3.1 illustrates a word with attributes and a number of "bonds" to connect with other words. Thus a word is very much like a piece of Legos for building toy objects.

The bonds exist more explicitly and are much more necessary in the 2D image domain. We define the visual vocabulary in the following.

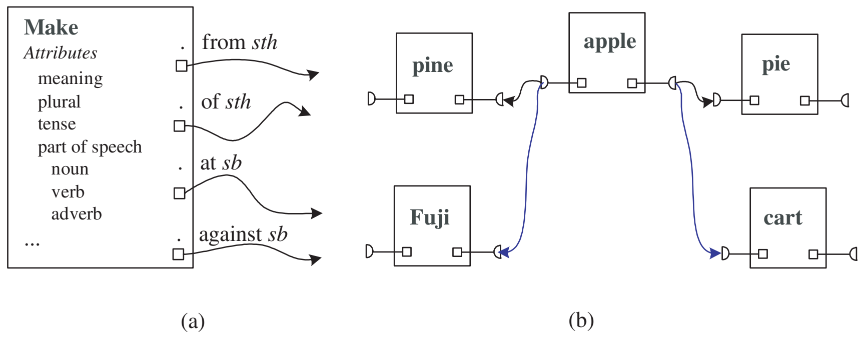


Fig. 3.1 In an English dictionary, each word has a number of attributes and some conventional ways to connect to other words. In the first example, the word "make" can be connected to "something" or "somebody." The word "apple" has strong bonds with other words to make compound words "pine-apple," "Fuji-apple," "apple-pie," "apple-cart."

Definition 3.1 Visual vocabulary. The visual vocabulary is a set of pairs, each consisting of an image function $\Phi_i(x, y; \alpha_i)$ and a set of $d(i)$ bonds (i.e., its degree), to be eventually connected with other elements, which are denoted by a vector $\beta_i = (\beta_{i,1}, \ldots, \beta_{i,d(i)})$. We think of $\beta_{i,k}$ as an address variable or pointer. α_i is a vector of attributes for (a) a geometric transformation, e.g., the central position, scale, orientation and plastic deformation, and (b) appearance, such as intensity contrast, profile or surface albedo. In particular, α_i determines a domain $\Lambda_i(\alpha_i)$ and Φ_i is then defined for $(x, y) \in \Lambda_i$ with values in R (a gray-valued template) or R^3 (a color template). Often each $\beta_{i,k}$ is associated with a subset of the boundary of $\Lambda_i(\alpha_i)$. The whole vocabulary is thus a set:

$$\Delta = \{(\Phi_i(x, y; \alpha_i), \beta_i) : (x, y) \in \Lambda_i(\alpha_i) \subset \Lambda\}, \tag{3.1}$$

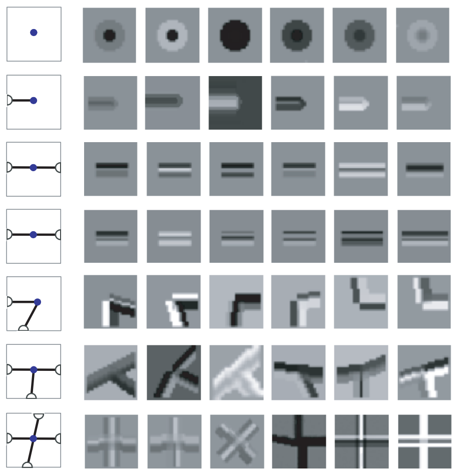
where i indexes the type of the primitives.

The conventional wavelets, Gabor image bases, image patches, and image fragments are possible examples of this visual vocabulary except that they do not have bonds. As an image grammar must adopt a multi-resolution representation, the elements in its vocabulary represent visual concepts at all levels of abstraction and complexity. In the following, we introduce some examples of the visual vocabulary at the low, middle, and high levels, respectively.

3.2 Image Primitives

In the 1960s–1970s, Julesz conjectured that textons (blobs, bars, terminators, crosses) are the atomic elements in the early stage of visual perception for local structures [37]. He found in texture discrimination experiments that the human visual system seem to detect these elements with a parallel computing mechanism. Marr extended Julesz's texton concept to image primitives which he called "symbolic tokens" in his primal sketch representation [49]. An essential criterion in selecting a dictionary in low level vision is to ensure that they are parsimonious and sufficient in representing real-world images, and more importantly they should have the necessary structures to allow composition into higher level parts. In this subsection, we review a dictionary of image primitives proposed in Guo et al. [31] as a formal mathematical model of the primal sketch. Many other studies have come up with similar lists, including studies which are based on the statistical analysis of small image patches from large databases [35, 42, 66].

Illustrated in Figure 3.2(a), an image primitive is a small image patch with a degree d connections or bonds which are illustrated by the half circles. The primitives are called blobs, terminators, edges or ridges, "L"-junctions, "T"-junctions, and cross junctions for $d = 0, 1, 2, 3, 4$, respectively. Each primitive has a number of attributes for its geometry and appearance. The geometric attributes include position, orientation, scale, and relative positions of the bonds with respect to the center. The appearance is described by the intensity profiles around the center and along the directions perpendicular to the line-segment connecting the center and the bonds. For instance, a $d = 2$ primitive could be called a step edge, a ridge/bar, or double edge depending on its intensity profile. Each bond of the primitive is like an arm or hand. When the bonds of two primitives are joined by matching the two half circles, we say they are connected. Figure 3.2(b) illustrates how a "T"-shape is composed through 3 terminators, 3 bars, and 1 "T"-junction.


(a)

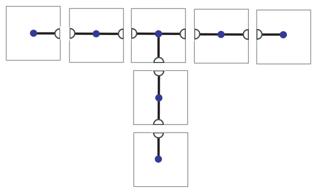

(b)

Fig. 3.2 Low level visual vocabulary — image primitives. (a) Some examples of image primitives: blobs, terminators, edges, ridges, "L"-junctions, "T"-junction, and cross junction etc. These primitives are the elements for composing a bigger graph structure at the upper level of the hierarchy. (b) is an example of composing a big "T"-shape image using 7 primitives. From [30].

In the following, we show how these primitives can be used to represent images. We start with a toy image in Figure 3.3 to illustrate the model and a real image in Figure 3.4.

In Figure 3.3, the boundaries of the two rectangles are covered by 4 "T"-junctions, 8 "L"-junctions, and 20 step edges. We denote the domain covered by an image primitive Φ_i^{sk} by $\Lambda_{\text{sk},i}$, and the pixels covered by these primitives, which are called the "sketchable part" in [31], are denoted by

$$\Lambda_{\text{sk}} = \bigcup_{i=1}^{n_{\text{sk}}} \Lambda_{\text{sk},i}. \tag{3.2}$$

The image $\mathbf{I}$ on Λ_{sk} is denoted by $\mathbf{I}_{\text{sk}}$ and is modeled by the image primitives through their intensity profiles. Let ϵ be the residual noise.

$$\mathbf{I}_{\text{sk}}(x,y) = \Phi_i^{\text{sk}}(x,y;\alpha_i,\beta_i) + \epsilon(x,y), \quad (x,y) \in \Lambda_{\text{sk},i}, \;\; i = 1,2,\ldots,n_{\text{sk}}. \tag{3.3}$$

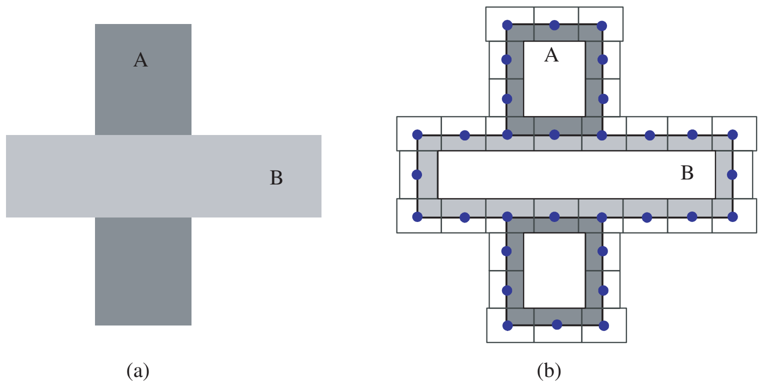


Fig. 3.3 An illustrative example for composing primitives into a graph configuration. (a) is a simple image, and (b) is a number of primitives represented by rectangles which cover the structured parts of the image. The remaining part of the image can be reconstructed through simple heat diffusion.

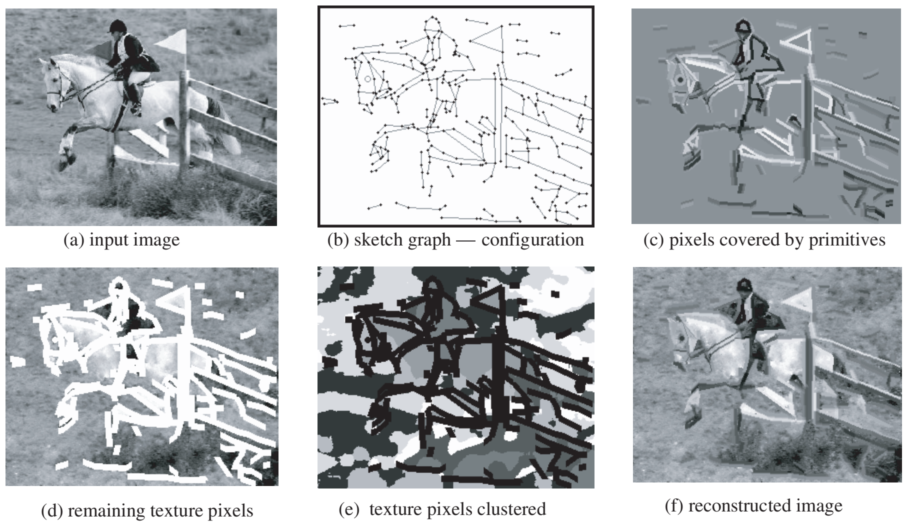

(a) input image
(b) sketch graph — configuration
(c) pixels covered by primitives
(d) remaining texture pixels
(e) texture pixels clustered
(f) reconstructed image

Fig. 3.4 An example of the primal sketch model. (a) An input image **I**. (b) The sketch graph – configuration computed from the image **I**. (c) The pixels in the sketchable part Λ_{sk}. (d) The remaining non-sketchable portion are textures, which are segmented into a small number of homogeneous regions in (e). (f) The final synthesized image integrating seamlessly the structures and textures. From [31].

The remaining pixels are flat or stochastic texture areas, called non-sketchable, and are clustered into a few homogeneous texture areas

$$\Lambda_{\mathrm{nsk}} = \Lambda \backslash \Lambda_{\mathrm{sk}} = \bigcup_{j=1}^{n_{\mathrm{nsk}}} \Lambda_{\mathrm{nsk},j}. \tag{3.4}$$

They can be reconstructed through Markov random field models conditional on $\mathbf{I}_{\mathrm{sk}}$,

$$\mathbf{I}_{\mathrm{nsk},j} \,|\, \mathbf{I}_{\mathrm{sk}} \sim p(\mathbf{I}_{\mathrm{nsk}} \,|\, \mathbf{I}_{\mathrm{sk}}; \Theta_j). \tag{3.5}$$

Θ_j is a vector-valued parameter for the Gibbs model, for example, the FRAME model [90].

Figure 3.4 shows a real example of the primal sketch model using primitives. The input image has 300×240 pixels, of which $18,185$ pixels (around 25%) are considered sketchable. The sketch graph has 275 edges/ridges (primitives with degree $d = 2$) and 152 other primitives for "vertices" of the graph. Their attributes are coded by $1,421$ bytes. The non-sketchable pixels are represented by 455 parameters or less. The parameters are 5 filters for 7 texture regions and each pools a 1D histogram of filter responses into 13 bins. Together with the codes for the region boundaries, total coding length for the textures is $1,628$ bytes. The total coding length for the synthesized image in Figure 3.4(f) is $3,049$ bytes or 0.04 byte per pixel. It should be noted that the coding length is roughly computed here by treating the primitives as being independent. If one accounts for the dependence in the graph and applies some arithmetic compression schemes, a higher compression rate can be achieved.

To summarize, we have demonstrated that image primitives can compose a planar attribute graph configuration to generate the structured part of the image. These primitives are transformed, warped, and aligned to each other to have a tight fit. Adjacent primitives are connected through their bonds. The explicit use of bonds distinguishes the image primitives from other basic image representations, such as wavelets and sparse image coding [47, 56] mentioned before, and other image patches and fragments in the recent vision literature [77]. The bonds encode the topological information, in addition to the geometry and appearance, and enable the composition of bigger and bigger structures in the hierarchy.

3.3 Basic Geometric Groupings

If by analogy, image primitives are like English letters or phonemes, then one wonders what are the visual words and visual phrases. This is the central question addressed by the gestalt school of psychophysicists [39, 88]. One may summarize their work by saying that the geometric relations of alignment, parallelism and symmetry, especially as created by occlusions, are the driving forces behind the grouping of lower level parts into larger parts. A set of these composite parts is shown in Figure 3.5 and briefly described in the caption.

It is important to realize that these groupings occur at every scale. Many of them occur in local groupings containing as few as 2–8 image primitives as in the previous section. We will call these "graphlets" [83]. But extended curves, parallels and symmetric structures may span the whole image. Notably, symmetry is always a larger scale feature but one occurring very often in nature (e.g., in faces) and which is highly detectable by people even in cluttered scenes. Parallel lines also occur

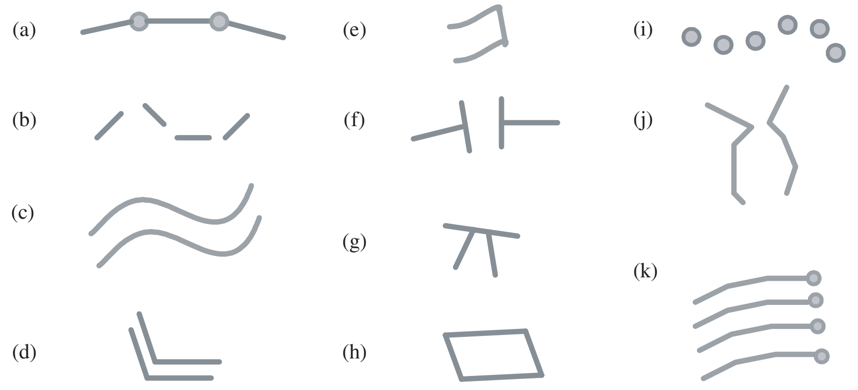


Fig. 3.5 Middle level visual vocabulary: common groupings found in images. (a) extended curves, (b) curves with breaks and imperfect alignment, (c) parallel curves, (d) parallels continuing past corners, (e) ends of bars formed by parallels and corners, (f) curves continuing across paired T-junctions (the most frequent indication of occlusion), (g) a bar occluded by some edge, (h) a square, (i) a curve created by repetition of discrete similar elements, (j) symmetric curves, and (k) parallel lines ending at terminators forming a curve.

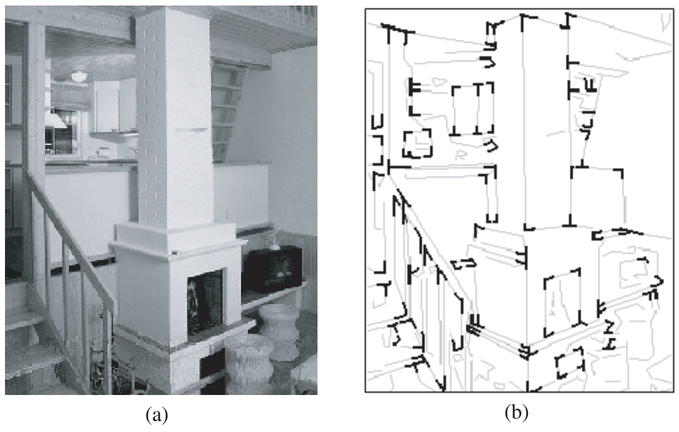


Fig. 3.6 An example of graphlets in natural image. The graphlets are highlighted in the primal sketch. These graphlets can be viewed as larger pieces of lego. From [24].

frequently in nature, e.g., in tree trunks. The occlusion clue shown in Figure 3.5 is especially important because it is not only common but is the strongest clue in a static 2D image to the 3D structure of the scene. Moreover, it implies the existence of an "amodal" or occluded contour representing the continuation of the left and right edges behind the central bar. This necessitates a special purpose algorithm to be discussed below. Figure 3.6 shows an image with its primal sketch on the right side with its graphlets shown in dark line segments.

These graphlets are learned through clustering and binding the image primitives in a way discussed in Equation (2.1). Each cluster in this space is an equivalence class subject to an affine transform, some deformation, as well as minor topological editing. These graphlets are generic 2D patterns, and some of them could be interpreted as object parts.

3.4 Parts and Objects

If one is only interested in certain object categories segmented from the background, such as bicycles, cars, ipods, chairs, clothes, the dictionary will be object parts. Although these object parts are significant within each category or reusable by a few categories, their overall frequency

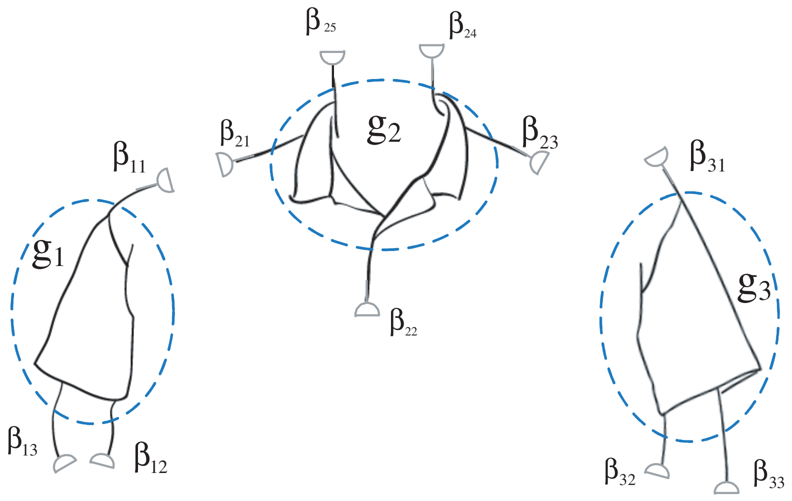


Fig. 3.7 High level visual vocabulary — the objects and parts. We show an example of upper body clothes made of three parts: a collar, a left and a right short sleeves. Each part is again represented by a graph with bonds. A vocabulary of part for human clothes is shown in Figure 3.8. From [9].

is low and they are often rare events in a big database of real-world images. Thus the object parts are less significant as contributors to lowering image entropy than the graphlets presented above, and the latter are, in turn, less entropically significant than the image primitives at the low level.

We take one complex object category — clothes as an example. Figure 3.7 shows how a shirt is composed of three parts: a collar, a left, and a right short sleeves. In this figure, each part is represented by an attribute graph with open bonds, like the graphlets. For example, the collar part has 5 bonds, and the two short sleaves have 3 bonds to be connected with the arms and collar. By decomposing a number of instances in the clothes category together with upper body and shoes, one can obtain a dictionary of parts. Figure 3.8 shows some examples for each category.

Thus we denote the dictionary by

$$\Delta_{\text{cloth}} = \{(\Phi_i^{\text{cloth}}(x,y;\alpha_i),\beta_i) \,:\, \forall i, \alpha_i, \beta_i.\} \tag{3.6}$$

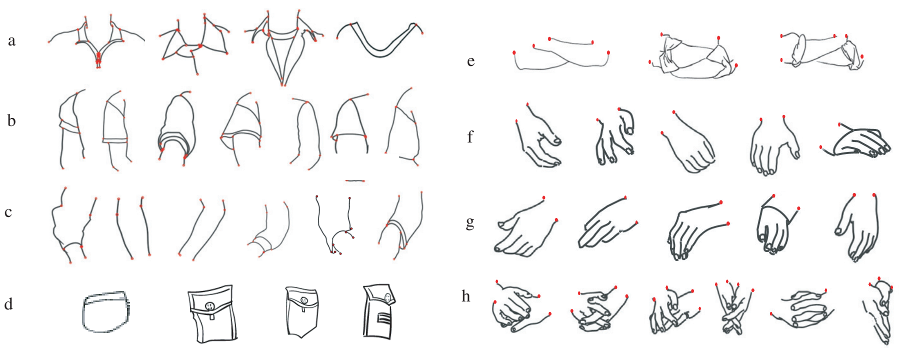


Fig. 3.8 The dictionary of object parts for cloth and body components. Each element is a small graph composed of primitives and graphlets and has open-bonds for collecting with other parts. Modified from [9].

As before, Φ_i^{cloth} is an image patch defined in a domain Λ_i^{cloth} which does not have to be compact or connected. α_i controls the geometric and photometric attributes, and $\beta_i = (\beta_{i1}, \beta_{i2}, \ldots, \beta_{id(i)})$ is a set of open bonds. These bonds shall be represented as address variables that point to other bonds. Some upper-cloth examples that are synthesized by these parts are shown in Figure 9.7.

In fact, the object parts defined above are not so much different from the dictionaries of image primitives or graphlets, except that they are bigger and more structured. Indeed they form a continuous spectrum for the vision vocabulary from low to high levels of vision.

By analogy, each part is like a class in object oriented programming, such as C++. The inner structures of the class are encapsulated, only the bonds are visible to other classes. These bonds are used for communication between different object instances.

In the literature, Biederman [5] proposes a set of "geons" as 3D object elements, which are generalized cylinders for representing 3D man-made objects. In practice, it is very difficult to compute these generalized cylinders from images. In comparison, we adopt a view based representation for the primitives, graphlets, and parts which can be inferred relatively reliably.

4
Relations and Configurations

While the hierarchical visual vocabulary represents the vertical compositional structures, the relations in this section represent the horizontal links for contextual information between nodes in the hierarchy at all levels. The vocabulary and relations are the ingredients for constructing a large number of image configurations at various level of abstractions. The set of valid configurations constitutes the language of an image grammar.

4.1 Relations

We start with a set of nodes $V = \{A_i : i = 1,2,\ldots,n\}$ where $A_i = (\Phi_i(x,y;\alpha_i),\beta_i) \in \Delta$ is an entity representing an image primitive, a grouping, or an object part as defined in the previous section. A number of spatial and functional relations must be defined between the nodes in V to form a graph with colored edges where the color indexes the type of relation.

Definition 4.1 Attributed Relation. A binary relation defined on an arbitrary set S is a subset of the product set $S \times S$

$$\{(s,t)\} \subset S \times S. \tag{4.1}$$

An attributed binary relation is augmented with a vector of attributes γ and ρ,

$$E = \{(s,t;\gamma,\rho) : s,t \in S\}, \tag{4.2}$$

where $\gamma = \gamma(s,t)$ represents the structure that binds s and t, and $\rho = \rho(s,t)$ is a real number measuring the compatibility between s and t. Then $\langle S,E\rangle$ is a graph expressing the relation E on S. A k-way attributed relation is defined in a similar way as a subset of S^k.

There are three types of relations of increasing abstraction for the horizontal links and context. The first type is the bond type that connects image primitives into bigger and bigger graphs. The second type includes various joints and grouping rules for organizing the parts and objects in a planar layout. The third type is the functional and semantic relation between objects in a scene.

Relation type 1: *Bonds and connections.* For a set of nodes $V = \{A_i : i = 1,2,\ldots,n\}$ defined above, each node $A_i \in V$ has a number of open bonds $\{\beta_{ij} : j = 1,2,\ldots,n(i)\}$ shown by the half disks in the previous section. We collect all these bonds as a set,

$$S_{\text{bond}} = \{\beta_{ij} : i = 1,2,\ldots,n,\ j = 1,2,\ldots,n(i)\}. \tag{4.3}$$

Two bonds β_{ij} and β_{kl} are said to be connected if they are aligned in position and orientation. Therefore the bonding relation is a set of pairs of bonds with attributes:

$$E_{\text{bond}}(S) = \{(\beta_{ij},\beta_{kl}\,;\gamma,\rho)\}, \tag{4.4}$$

where $\gamma = (x,y,\theta)$ denote the position and orientation of the bond. The latter is the tangent direction at the bond for the two connected primitives. ρ is a function to check the consistency of intensity profile or color between two connected primitives.

The trivial example is the image lattice. The primitives A_i, $i = 1,\ldots,|\Lambda|$ are the pixels. Each pixel has 4 bonds $\beta_{ij},\ j = 1,2,3,4$. Then $E_{\text{bond}}(S)$ is the set of 4-nearest neighbor connections. In this case, $\gamma = nil$ is empty, and ρ is a pair-clique function for the intensities at pixels i and j. Figures 3.5 and 3.7 show more examples of bonds for composing graphlets from primitives, and composing clothes from parts. Very often people use graphical models, such as templates, with fixed structures where the bonds are decided deterministically and thus become transparent. In the next subsection, we shall define the bonds as random variables to reconfigure the graph structures.

Relation type 2: Joints and junctions. When image primitives are connected into larger parts, some spatial and functional relations must be found. Besides its open bonds to connect with others, usually its immediate neighbors, a part may be bound with other parts in various ways. The gestalt groupings discussed in the previous section are the best examples: parts can be linked over possibly large distances by being collinear, parallel, or symmetric. To identify this groupings, connections must be created flagging this non-accidental relationship. Figure 4.1 displays some typical relations of this type between object parts.

Some of these relations also contribute to 3D interpretations. For example, an ellipse is a part that has multiple possible compositions. If it is recognized as a bike wheel, its center can function as an axis and thus can be connected to the tip of a bar (see the rightmost of Figure 4.1). It could also be the rim of a tea cup, and then the two ends of its long axis will be joined to a pair of parallel lines to form a cylinder. In Figure 2.8, we discussed a phenomenon occurred in language where the word "what" is shared by two clauses. Similarly we have many such joints in images, such as hinge joints, and butting joints.

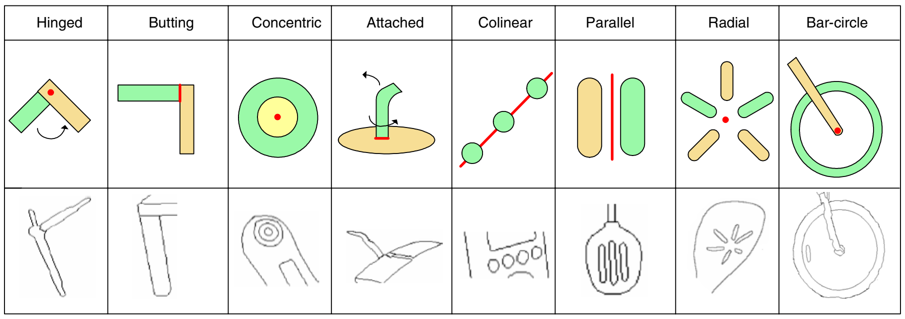


Fig. 4.1 Examples of spatial relations for binding object parts. The red dots or lines are the attributes $\gamma(s,t)$ of joint relation (s,t) which form the "glue" in this relation. From [59].

As Figure 4.1 shows, two parts can be hinged at a point. For example, two hands of a clock have a common axis. For a set of parts in an image $S = V$, the hinge relation is a set

$$E_{\text{hinge}}(S) = \{(A_i, A_j\,; \gamma(A_i, A_j), \rho(A_i, A_j))\}. \tag{4.5}$$

Here γ is the hinge point and $\rho = nil$. In a butting relation, $\gamma(A_i, A_j)$ represents the line segment(s) shared by the two parts. The line segment is shown in red in Figure 4.1. Sometimes, two parts may share two line segments. For example the handle of a teapot or cup share two line segments with the body.

Relation type 3: *Object interactions and semantics.* When letters are grouped into words, semantic meanings emerge. When parts are grouped into objects, semantic relations are created for their interactions. Very often these relations are directed. For example, the occluding relation is a viewpoint dependent binary relation between object or surfaces, and it is important for figure-ground segregation. A view point independent relation is a supporting relation. A simple example is shown in Figure 2.11. Let $S = V$ be a set of objects,

$$\begin{aligned} E_{\text{supp}} &= \{\langle \text{M}, \text{D}\rangle, \langle \text{M}, \text{E}\rangle\}, \\ E_{\text{occld}} &= \{\langle D, M\rangle, \langle E, M\rangle, \langle D, N\rangle, \langle E, N\rangle\}. \end{aligned} \tag{4.6}$$

The $\langle\ \rangle$ represents directed relation and the attributes γ, ρ are omitted. There are other functional relations among objects in a scene. For example, a person A is eating an apple $B\, E_{\text{edible}}(S) = \{\langle A, B\rangle\}$, and a person is riding a bike $E_{\text{ride}}(S) = \{\langle A, C\rangle\}$. These directed relations usually are partially ordered.

It is worth mentioning that the relations are dense at low level, such as the bonds, in the sense that the size $|E(S)|$ is in the order of $|S|$, and that they become very sparse (or rare) and diverse at high level. At the high level, we may find many interesting relations but each relation may only have a few occurrences in the image.

4.2 Configurations

So far, we have introduced the visual dictionaries and relations at various levels of abstractions. The two components are integrated into what we call the visual configuration in the following.

Definition 4.2 Configuration. A configuration $\mathcal{C}$ is a spatial layout of entities in a scene at certain level of abstraction. It is a one layer graph, often flattened from hierarchic representation,

$$\mathcal{C} = \langle V, E \rangle. \tag{4.7}$$

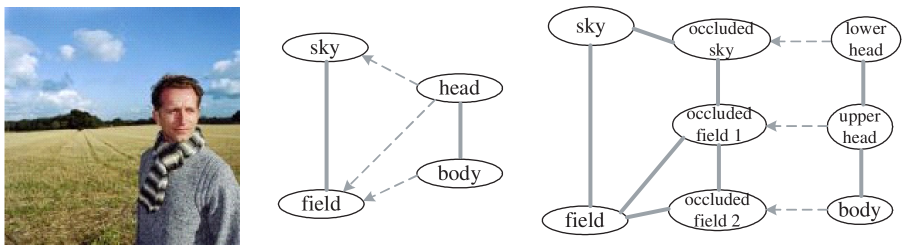
$V = \{A_i, i = 1, 2, \ldots, n\}$ is a set of attributed image structures at the same semantic level, such as primitives, parts, or objects and E is a relation. If V is a set of sketches and $E = E_{\text{bonds}}$, then $\mathcal{C}$ is a primal sketch configuration. If E is a union of several relations $E = E_{r_1} \cup \cdots \cup E_{r_k}$, which often occurs at the object level, then $\mathcal{C}$ is called a "*mixed configuration.*" For a generative model, the image on a lattice is the ultimate "*terminal configuration,*" and its primal sketch is called the "*pre-terminal configuration.*" Note that E will close some of the bonds in V and leave others open; thus we may speak of the open bonds in a configuration.

We briefly present examples of configurations at three levels.

First, for early vision, the scene configuration $\mathcal{C}$ is a primal sketch graph where V is a set of image primitives with bonds and $E = E_{\text{bonds}}$ is the bond relation. For example, Figure 3.3(b) illustrates a configuration for a simple image in Figures 3.3(a), and 3.4(b) is a configuration for the image in Figure 3.4(a). These configurations are attributed graphs because each primitive v_i is associated with variables α_i for its geometric properties and photometric appearance. The primal sketch graph is a parsimonious "token" representation in Marr's words [49], and thus it is a crucial stage connecting the raw image signal and the symbolic representation above it. It can reconstruct the original image with perceptually equivalent texture appearance.

Second, for the parts to object level, Figure 9.7 displays three possible upper body configurations composed of a number of clothes' parts shown in Figure 3.8. In these examples, each configuration $\mathcal{C}$ is a graph with vertices being 6–7 parts and $E = E_{\text{bond}}$ is a set of bonds connecting the parts, as it was shown in Figure 3.7.

Third, Figures 4.2(a) and 4.2(b) illustrate a scene configuration at the highest level of abstraction. V is a set of objects, and E included



(a) image (b) layer 1 configuration (c) layer 2 configuration

Fig. 4.2 An illustration of scene configuration. (a) is a scene of a man in a field. (b) is the graph for the highest level configuration $\mathcal{C} = \langle V, E \rangle$, V is the set of 4 objects $\{sky,\ field,\ head,\ body\}$ and $E = E_{\text{adj}} \cup E_{\text{occlude}}$ includes two relations: "adjacency" (solid lines) and"occlusion" (dotted arrows). (c) is the configuration at an intermediate level in which the occlusion relation is unpacked: now the dotted arrows indicate two *identical* sets of pixels but on separate layers.

two relations an "adjacency" relation in solid lines

$$E_{\text{adj}} = \{(sky, field), (head, body)\}, \tag{4.8}$$

and a directed "occlusion" relation in dotted arrows,

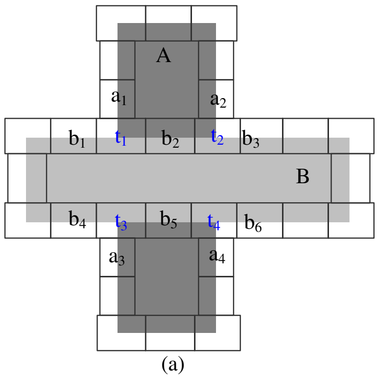
$$E_{\text{contain}} = \{\langle head, sky \rangle, \langle head, field \rangle, \langle body, field \rangle\}. \tag{4.9}$$

In summary, the image grammar which shall be presented in the next section is also called a "layered grammar." That is, it can generate configurations as its "language" at different levels of detail.

4.3 The Reconfigurable Graphs

In vision, the configurations are inferred from images. For example, in a Bayesian framework, the graph $\mathcal{C} = \langle V, E \rangle$ will not be pre-determined but reconfigurable on-the-fly. That is, the set of vertices may change, so does the set of edges (relations). Therefore, the configurations must be made flexible to meet the demand of various visual tasks. Figure 4.3 shows such an example.

On the left of the figure is a primal sketch configuration $\mathcal{C}_{\text{sk}}$ for the simple image shown in Figure 3.3. This is a planar graph with 4 "T"-junctions. In this configuration two adjacent primitives are connected by the bond relation E_{bond}. The four "T"-junctions are then



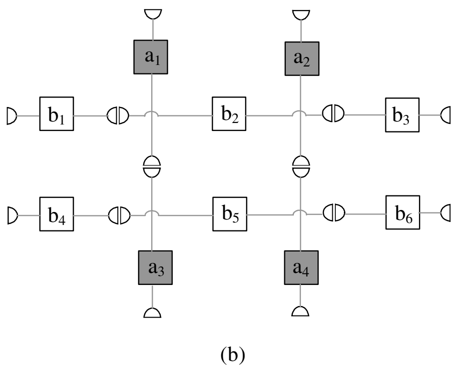


Fig. 4.3 (a) A primal sketch configuration for a simple image. It has four primitives for "T"-junctions — t_1, t_2, t_3, t_4. It is a planar graph formed by bonding the adjacent primitives. (b) A layered (2.1D sketch) representation with two occluding surfaces. The four "T"-junctions are broken. The bonds are reorganized. a_1 is bonded with a_3, and a_2 is bonded with a_4.

broken in the right configuration, which is called the 2.1D sketch [53] and denoted by $\mathcal{C}_{2.1\text{sk}}$. The bonds are reorganized with a_1 being connected with a_3 and a_2 with a_4. $\mathcal{C}_{2.1\text{sk}}$ includes two disjoint subgraphs for the two rectangles in two layers. From this example, we can see that both the vertices and the bonds must be treated as random variables. Figure 4.4 shows a real application of this sort of reconfiguration in computing a 2.1D sketch from a 2D primal sketch. This example is from [25]. It decomposes an input image in Figure 4.4(a) into three layers in Figures 4.4(d)–(f), found after reconfiguring the bonds by completing the contours (red line segments in Figures 4.4(b) and 4.4(c)) behind and filling-in the occluded areas using the Markov random field region descriptor in the primal sketch model. From the point of view of parse structures, we need to add new nodes to represent the extra layers present behind the observed surfaces together with "occluded by" relations. This is illustrated in Figure 4.2(c). This is a configuration which has duplicated three regions to represent missing parts of the background layer.

A mathematical model for the reconfigurable graph is called the *mixed Markov model* in [20]. In a mixed Markov model, the bonds are treated as nodes. Therefore, the vertex set V of a configuration

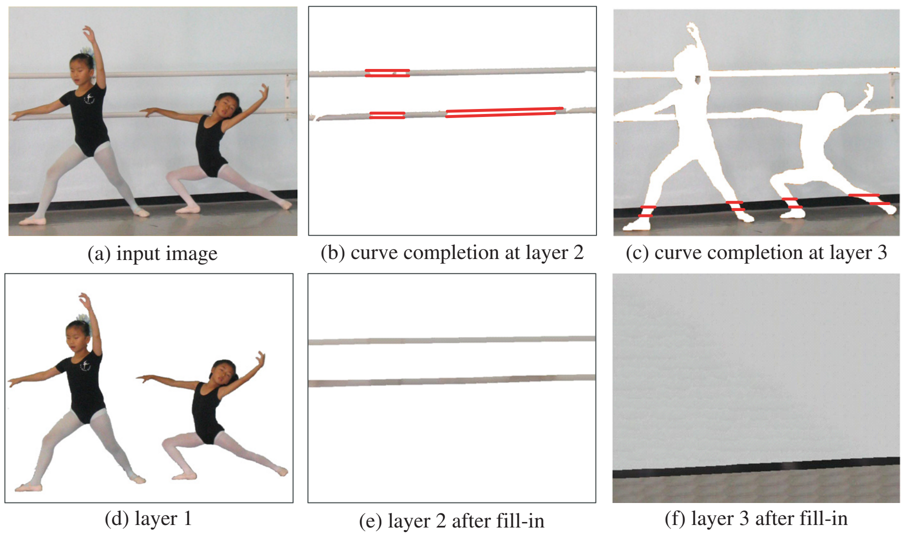

(a) input image
(b) curve completion at layer 2
(c) curve completion at layer 3
(d) layer 1
(e) layer 2 after fill-in
(f) layer 3 after fill-in

Fig. 4.4 From a 2D sketch to a 2.1D layered representation by reconfiguring the bond relations. (a) is an input image from which a 2D sketch is computed. This is transferred to a 2.1D sketch representations with three layers shown in (d), (e), and (f), respectively. The inference process reconfigures the bonds of the image primitives shown in red in (b) and (c). From [25].

has two type of nodes — $V = V_x \cup V_a$. V_x include the usual nodes for image entities, and V_a is a set of address nodes, for example, the bonds. The latter are like the pointers in the C language. These address nodes reconfigure the graphical structure and realize non-local relations. It was shown that a probability model defined on such reconfigurable graphs still observes a suitable form of he Hammersley-Clifford theorem and can be simulated by Gibbs sampler.

By analogy to language, the bonds in this example correspond to the arrows in the English sentence discussed in Figure 2.8 for non-local context. As there are many possible (bond, joint, functional, and semantic) relations, each image entity (primitives, parts, objects) may have many random variables as the "pointers." Many of them could be empty, and will be instantiated in the inference process. This is similar to the words "apple" and "make" in Figure 3.1.

5
Parse Graph for Objects and Scenes

In this chapter, we define parse graphs as image interpretations. Then we will show in the next chapter that these parse graphs are generated as instances by an And–Or graph. The latter is a general representation that embeds the image grammar.

Recall that in Section 2.2 a language grammar is a 4-tuple $\mathcal{G} = (V_N, V_T, \mathrm{R}, S)$, and that a sentence ω is derived (or generated) by a sequence of production rules from a starting symbol S,

$$S \overset{\gamma_1,\gamma_2,\ldots,\gamma_{n(\omega)}}{\Longrightarrow} \omega. \tag{5.1}$$

These production rules form a parse tree for ω,

$$\mathbf{pt}(\omega) = (\gamma_1, \gamma_2, \ldots, \gamma_{n(\omega)}). \tag{5.2}$$

For example, Figure 2.6 shows two possible parse trees for a sentence "I saw the man with the telescope."

This grammar is a generative model, and the inference is an inverse process that computes a parse tree for a given sentence as its interpretation or one of its best interpretations. Back to image grammars, a configuration $\mathcal{C}$ is a flat attributed graph corresponding to a sentence ω, and a parse tree **pt** is augmented to a parse graph **pg** by adding horizontal links for various relations. In previous chapter, Figure 2.11(b) has shown a parse graph for a block work scene, and Figure 1.1 has shown a parse graph for a football match scene.

In the following, we define a parse graph as an interpretation of image.

Definition 5.1 Parse graph. A *parse graph* **pg** consists of a hierarchic parse tree (defining "vertical" edges) and a number of relations E (defining "horizontal edges"):

$$\mathbf{pg} = (\mathbf{pt}, E). \tag{5.3}$$

The parse tree **pt** is also an And-tree whose non-terminal nodes are all And-nodes. The decomposition of each And-node A into its parts is given by a production rule which now produces not a string but a configuration:

$$\gamma : A \to \mathcal{C} = \langle V, E \rangle. \tag{5.4}$$

A production should also associate the open bonds of A with open bonds in $\mathcal{C}$. The whole parse tree is a sequence of production rules

$$\mathbf{pt}(\omega) = (\gamma_1, \gamma_2, \ldots, \gamma_n). \tag{5.5}$$

The horizontal links E consists of a number of directed or undirected relations among the terminal or non-terminal nodes, such as bonds, junctions, functional and semantic relations,

$$E = E_{r_1} \cup E_{r_2} \cup \cdots \cup E_{r_k}. \tag{5.6}$$

A parse graph **pg**, when collapsed, produces a series of flat configurations at each level of abstraction/detail,

$$\mathbf{pg} \Longrightarrow \mathcal{C}. \tag{5.7}$$

Depending on the type of relation, there may be special rules for producing relations at a lower level from higher level relations in the collapsing process. The finest configuration is the image itself in which every pixel is explained by the parse graph. The next finest configuration is the primal sketch graph.

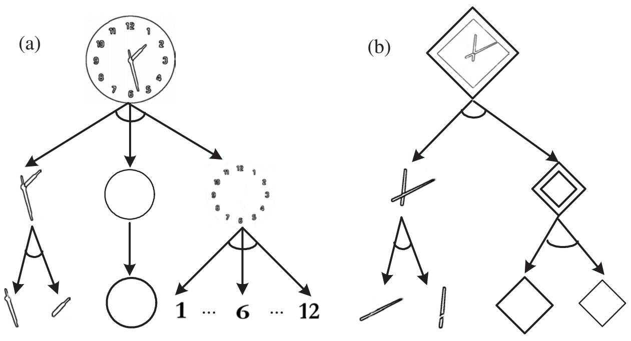


Fig. 5.1 Two parse graph examples for clocks which are generated from the And–Or-graph in Figure 6.1. From [86].

The parse graph, augmented with spatial context and possible functional relations, is a comprehensive interpretation of the observed image **I**. The task of image parsing is to compute the parse graph from input image(s). In the Bayesian framework, this is to either maximize the posterior probability for an optimal solution,

$$\mathbf{pg}^* = \arg\max p(\mathbf{pg}|\mathbf{I}), \tag{5.8}$$

or sampling the posterior probability for a set of distinct solutions,

$$\{\mathbf{pg}_i : i = 1,2,\ldots,K\} \sim p(\mathbf{pg}|\mathbf{I}). \tag{5.9}$$

Object instances in the same category may have very different configurations and thus distinct parse graphs. Figure 5.1 displays two parse graphs for two clock instances. It has three levels and the components are connected through three types of relations: the hinge joint to connect clock hands, a co-centric relation to align the frames, and a radial relation to align the numbers.

As it was mentioned in Section 2.6, objects appear at arbitrary scales in images. As shown in Figure 2.10, a face can be decomposed into facial elements at higher resolution, and it may terminate as a whole face for low resolution. Therefore, one remarkable property that distinguishes an image parse graph is that a parse graph may stop at any level of abstraction, while the the parse tree in language must stop at the word level. This is the reason for defining visual vocabulary at multiple levels of resolution, and defining the image grammar as a layered grammar.

6

Knowledge Representation with And–Or Graph

This chapter addresses the central theme of the paper — developing a consistent representation framework for the vast amount of visual knowledge at all levels of abstraction. The proposed representation is the And–Or graph embedding image grammars. The And–Or graph representation was first explicitly used in [9] for representing and recognizing a complex object category of clothes.

6.1 And–Or Graph

While a parse graph is an interpretation of a specific image, an And–Or graph embeds the whole image grammar and contains all the valid parse graphs. Before introducing the And–Or graph, we revisit the origin of grammar and its Chomsky formulation in Sections 2.1 and 2.2.

First, we know each production rule in the SCFG can be written as

$$\mathrm{A} \to \beta_1 \,|\, \beta_2 \cdots |\, \beta_{\mathrm{n(A)}}, \quad \text{with } \mathrm{A} \in \mathrm{V_N},\ \beta \in (\mathrm{V_N} \cup \mathrm{V_T})^+. \qquad (6.1)$$

Therefore each non-terminal node A can be represented by an Or-node with $n(A)$ alternative structures, each of which is an And-node composed of a number of substructures. For example, the following rule is represented by a two level And–Or tree in Figure 2.3.

$$\mathrm{A} \rightarrow \mathrm{a} \cdot \mathrm{b} \,|\, \mathrm{c} \cdot \mathrm{c}; \quad \rho|(1-\rho). \tag{6.2}$$

The two alternatives branches at the Or-node are assigned probabilities $(\rho, 1-\rho)$. Thus an SCFG can be understood as an And–Or tree.

Second, we have shown in Figure 2.4 that a small And–Or tree can produce a combinatorial number of configurations — called its language. To represent contextual information in the following, we augment the And–Or tree into an And–Or graph producing a context sensitive image grammar.

In a previous survey paper [89], the first author showed that any visual pattern can be conceptualized as a statistical ensemble that observes a certain statistical description. For a complex object pattern, its statistical ensemble must include a large number of distinct configurations. Thus our objective is to define an And–Or graph, thus its image grammar, such that its language, i.e., the set of valid configurations that it produces, reproduces the ensemble of instances for the visual pattern.

An And–Or graph augments an And–Or tree with two new features.

1. Horizontal lines are indicate to show relations, bonds, junctions, and semantic relations.
2. Relations at all levels are augmented on the And–Or graph to represent hard (compatibility) or soft (statistical) constraints.
3. The children of an Or-node may share Or-node children. It represents a reusable part shared by several production rules. The sharing of nodes reduces the complexity of the representation and thus the size of dictionary. Other possible sharings may be useful: see, for example, Section 2.3.

In Chapter 1, Figure 1.3(a) has shown a simple example of an And–Or graph. An And–Or graph includes three types of nodes: And-nodes (solid circles), Or-nodes (dashed circles), and terminal nodes (squares). The Or-nodes have labels for classification at various levels, such as scene category, object classes, and parts etc. Due to this recursive definition, one may merge the And–Or graphs for many objects or scene categories into a larger graph. In theory, the whole natural image ensemble can be represented by a huge And–Or graph, as it is for language.

By assigning values to these labels on the Or-node, one obtains an And-graph — i.e., a parse graph. The bold arrows and shaded nodes in Figure 1.3(a) constitute a parse graph **pg** embedded in the And–Or graph. This parse graph is shown in Figure 1.3(b) and produces a configuration shown in Figure 1.3(d). It has four terminal nodes (for primitives, parts, or objects): $1, 6, 8, 10$ and the edges are inherited from their parent relations. Both nodes 8 and 10 have a common ancestor node C. Therefore the relation $\langle B, C\rangle$ is propagated to $\langle 1, 6\rangle$ and $\langle 1, 8\rangle$. For example, if $\langle B, C\rangle$ includes three bonds, two bonds may be inherited by $\langle 1, 8\rangle$ and one by $\langle 1, 6\rangle$. Similarly the links $\langle 6, 10\rangle$ and $\langle 8, 10\rangle$ are inherited from $\langle C, D\rangle$.

Figure 1.3(c) is a second parse graph and it produces a configuration in Figure 1.3(e). It has 4 terminal nodes $2, 4, 9, 9'$. The node 9 is a reusable part shared by nodes C and D. It is worth mentioning that a shared node may appear as multiple instances.

Definition 6.1 And–Or Graph. An And–Or graph is a 6-tuple for representing an image grammar $\mathcal{G}$.

$$G_{\text{and-or}} = \langle S, V_N, V_T, \mathcal{R}, \Sigma, \mathcal{P}\rangle. \tag{6.3}$$

S is a root node for a scene or object category, $V_N = V^{\text{and}} \cup V^{\text{or}}$ is a set of non-terminal nodes including an And-node set V^{and} and an Or-node set V^{or}. The And-nodes plus the graph formed by their children are the productions and the Or-nodes are the vocabulary items. V_T is a set of terminal nodes for primitives, parts, and objects (note that an object at low resolution may terminate without decomposition directly), $\mathcal{R}$ is a number of relations between the nodes, Σ is the set of all valid configurations derivable from the grammar, i.e., its language, and $\mathcal{P}$ is the probability model defined on the And–Or graph.

The following is more detailed explanation of the components in the And–Or graph.

1. *The Non-terminal nodes* include both And-nodes and Or-nodes $V_N = V^{\text{and}} \cup V^{\text{or}}$,

   $$V^{\text{and}} = \{u_1, \ldots, u_{m(u)}\}, \quad V^{\text{or}} = \{v_1, \ldots, v_{m(v)}\}. \qquad (6.4)$$

   An Or-node $v \in V^{\text{or}}$ is a switch pointing to a number of possible And-nodes, the productions whose head is v.

   $$v \to u_1 \,|\, u_2 \cdots |\, u_{n(v)}, \quad u_1, \ldots, u_n \in V^{\text{and}}. \qquad (6.5)$$

   We define a switch variable $\omega(v)$ for $v \in V$, that takes an integer value to index the child node.

   $$\omega(v) \in \{\emptyset, 1, 2, \ldots, n(v)\}. \qquad (6.6)$$

   By choosing the switch variables in the Or-nodes, one obtains a parse graph from the And–Or graph. The switch variable is set to empty $\omega(v) = \emptyset$ if v is not part of the parse graph. In fact the assignments of Or-nodes at various levels of the And–Or graph corresponds to scene classification and object recognition. In practice, when an Or-node has a large $n(v)$, i.e., too fat, one may replace it by a small Or-tree that has $n(v)$ leaves. We omit the discussion of such cases for clarity.

   An And-node $u \in V^{\text{and}}$ either terminates as a template $t \in V_T$ or it can be decomposed into a set of Or-nodes. In the latter case, the relations between these child nodes are specified by some relations $r_1, \ldots, r_{k(u)} \in \mathcal{R}$ shown by the dashed horizontal lines in Figure 1.3. We adopt the symbol :: for representing the relations associated with the production rule or the And-node.

   $$u \to t \in V_T; \text{ or}$$
   $$u \to \mathcal{C} = (v_1, \ldots, v_{n(v)}) :: (r_1, \ldots, r_{k(v)}), \quad v_i \in V, \; r_j \in \mathcal{R}.$$

   The termination rule reflects the multi-scale representation. That is, the node u may be instantiated by a template at a relatively lower image resolution.

2. *The Terminal node set* $V_T = \{t_1, \ldots, t_{m(T)}\}$ is a set of instances from the image dictionary Δ. Usually it is a graphical template $(\Phi(x, y; \alpha), \beta)$ with attributes α and open bonds β. Usually, each $t \in V_T$ is a sketch graph, such as the image primitives.
3. The Configurations which are produced from the root node S are the language of the grammar: $G_{\text{and}-\text{or}}$,

$$\mathbf{L}(G_{\text{and}-\text{or}}) = \Sigma = \left\{\mathcal{C}_k : S \overset{G_{\text{and}-\text{or}}}{\Longrightarrow} \mathcal{C}_k \, k = 1, 2, \ldots, N\right\}. \quad (6.7)$$

Each configuration $\mathcal{C} \in \Sigma$ is a composite template, for example, the cloth shown in Figure 3.7. The And–Or graph in Figure 1.3(a) contains a combinatorial number of valid configurations, e.g.,

$$\Sigma = \{(1,6,8,10), (2,4,9,9), (1,5,11), (2,4,6,7,9), \ldots\}. \quad (6.8)$$

The first two configurations are shown on the right side of Figure 1.3.
4. *The relation set* $\mathcal{R}$ pools over all the relations between nodes at all levels.

$$\mathcal{R} = \bigcup_m E_m = \{e_{st} = (v_s, v_t; \gamma_{st}, \rho_{st})\}. \quad (6.9)$$

These relations become the pair-cliques in the composite graphical template. When a node v_s is split later, the link e_{st} may be split as well or may descend to specific pairs of children. For example, in Figure 1.3 node C is split into two leaf nodes 6 and 8, then the relation (B, C) is split into two subsets between $(1, 6)$ and $(1, 8)$.
5. $\mathcal{P}$ is a probability model defined on the And–Or graph. It includes many local probabilities - one at each Or-node to account for the relative frequency of each alternative, and local energies associated with each link $e \in \mathcal{R}$. The former is like the SCFG and the latter is like the Markov random fields or graphical models. We will discuss the probability component in the next subsection.

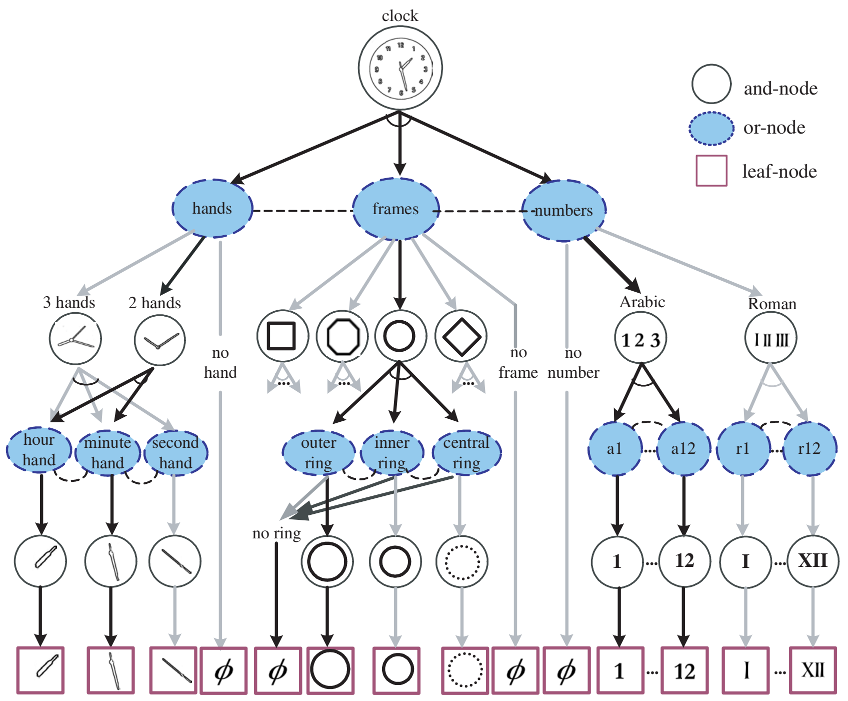


Fig. 6.1 An And–Or graph example for the object category — clock. It has two parse graphs shown in Figure 5.1, one of which is illustrated in dark arrows. Some leaf nodes are omitted from the graph for clarity. From [86].

Before concluding this section, we show an And–Or graph for a clock category [86] in Figure 6.1. Figure 6.1 has shown two parse graphs as instances of this And–Or graph. The dark bold arrows in Figure 6.1 are the parse tree shown in Figure 5.1(a).

Another And–Or example is shown in Figure 9.6. It is a subgraph extracted, for reason of clarity, from a big And–Or graph for the upper body of human figure [9]. Figure 9.7 displays three cloth configurations produced by this And–Or graph.

In summary, an And–Or graph $G_{\text{and-or}}$ defines a *context sensitive graph grammar* with V_T being its vocabulary, V_N the production rules, Σ its language, $\mathcal{R}$ the contexts. $G_{\text{and-or}}$ contains all the possible parse graphs which in turn produce a combinatorial number of configurations. Again, the number of configurations is far larger than the vocabulary, i.e.,

$$|V_N \cup V_T| \ll |\Sigma|. \tag{6.10}$$

This is a crucial aspect for representing the large intra-category structural variations.

Our next task is to define a probability model on $G_{\text{and}-\text{or}}$ to make it a stochastic grammar.

6.2 Stochastic Models on the And–Or Graph

The probability model for the And–Or graph $G_{\text{and}-\text{or}}$ must integrate the Markov tree model (SCFG) for the Or-nodes and the graphical (Markov) models for the And-nodes. Together a probability model is defined on the parse graphs. The objective of this probability model is to match the frequency of parse graphs in an observed training set (supervised learning will be discussed in the next section).

Just as the language model in Equation (2.17) defined probabilities on each parse tree $\mathbf{pt}(\omega)$ of each sentence ω, the new model should define probabilities on each parse graphs $\mathbf{pg}$. As $\mathbf{pg}$ produces a final configuration $\mathcal{C}$ deterministically when it is collapsed, thus $p(\mathbf{pg};\Theta)$ produces a marginal probability on the final configurations with Θ being its parameters. A configuration $\mathcal{C}$ is assumed to be directly observable, i.e., the input, and parse graph $\mathbf{pg}$ are hidden variables and have to be inferred.

By definition IV, a parse graph $\mathbf{pg}$ is a parse tree $\mathbf{pt}$ augmented with relations E,

$$\mathbf{pg} = (\mathbf{pt}, E). \tag{6.11}$$

For notational convenience, we denote the following components in $\mathbf{pg}$.

- $T(\mathbf{pg}) = \{t_1, \ldots, t_{n(\mathbf{pg})}\}$ is the set of leaf nodes in $\mathbf{pg}$. For example, $T(\mathbf{pg}) = \{1, 6, 8, 10\}$ for the parse graph shown by the dark arrows in Figure 1.3. In applications, $T(\mathbf{pg})$ is often the pre-terminal nodes with each $t \in T(\mathbf{pg})$ being an image primitive in the primal sketch.

- $V^{\text{or}}(\mathbf{pg})$ is the set of non-empty Or-nodes (switches) that are used $\mathbf{pg}$. For instance, $V^{\text{or}}(\mathbf{pg}) = \{B, C, D, N, O\}$. These switch variables selected the path to decide the parse tree $\mathbf{pt} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$.
- $E(\mathbf{pg})$ is the set of links in $\mathbf{pg}$.

The probability for $\mathbf{pg}$ is of the following Gibbs form, similar to Equation (2.17),

$$p(\mathbf{pg}; \Theta, \mathcal{R}, \Delta) = \frac{1}{Z(\Theta)} \exp\{-\mathcal{E}(\mathbf{pg})\}, \tag{6.12}$$

where $\mathcal{E}(\mathbf{pg})$ is the total energy,

$$\begin{aligned}\mathcal{E}(\mathbf{pg}) = &\sum_{v \in V^{\text{or}}(\mathbf{pg})} \lambda_v(\omega(v)) + \sum_{t \in T(\mathbf{pg}) \cup V^{\text{and}}(\mathbf{pg})} \lambda_t(\alpha(t)) \\ &+ \sum_{(i,j) \in E(\mathbf{pg})} \lambda_{ij}(v_i, v_j, \gamma_{ij}, \rho_{ij}).\end{aligned} \tag{6.13}$$

The model is specified by a number of parameters Θ, the relations set $\mathcal{R}$, and the vocabulary Δ. The first term in the energy is the same as the SCFG. It assigns different weights $\lambda_v()$ to the switch variables $\omega(v)$ at the or-nodes v. The weight should account for how frequently a child node appears. Removing the 2nd and 3rd terms, this reduces to an SCFG in Equation (2.9). The second and third terms are typical singleton and pair-clique energy for graphical models. The second term is defined on the geometric and appearance attributes of the image primitives. The third term models the compatibility constraint, such as the spatial and appearance constraint between the primitives, graphlets, parts, and objects.

This model can be derived from a maximum entropy principle under two types of constraints on the statistics of training image ensembles. One is to match the frequency at each Or-node, just like the SCFG, and the other is to match the statistics, such as histograms or co-occurrence frequency as in standard graphical models. Θ is the set of parameters in the energy,

$$\Theta = \{\lambda_v(), \lambda_t(), \lambda_{ij}();\ \forall v \in V^{\text{or}}, \forall t \in V_T, \forall (i,j) \in \mathcal{R}\}. \tag{6.14}$$

Each $\lambda()$ above is a potential function, not a scalar, and is represented by a vector through discretizing the function in a non-parametric way, as it was done in the FRAME model for texture [90]. Δ is the vocabulary for the generative model. The partition function is summed over all parse graph in the And–Or graph $G_{\text{and}-\text{or}}$ or the grammar $\mathcal{G}$.

$$Z = Z(\Theta) = \sum_{\mathbf{pg}} \exp\{-\mathcal{E}(\mathbf{pg})\}. \tag{6.15}$$

7

Learning and Estimation with And–Or Graph

Suppose we have a training set sampled from an underlying distribution f governing the objects.

$$D^{\text{obs}} = \{(\mathbf{I}_i^{\text{obs}}, \mathbf{pg}_i^{\text{obs}}) : \ i = 1, 2, \ldots, N\} \sim f(\mathbf{I}, \mathbf{pg}). \tag{7.1}$$

The parse graphs $\mathbf{pg}_i^{\text{obs}}$ are from the groundtruth database [87] or considered missing in unsupervised case. The objective is to learn a model p which approaches f by minimizing a Kullback–Leibler divergence,

$$\begin{aligned} p^* &= \arg\min KL(f||p) \\ &= \arg\min \sum_{\mathbf{pg} \in \Omega_{\mathbf{pg}}} \int_{\Omega_{\mathbf{I}}} f(\mathbf{I}, \mathbf{pg}) \log \frac{f(\mathbf{I}, \mathbf{pg})}{p(\mathbf{I}, \mathbf{pg}; \Theta, \mathcal{R}, \Delta)} d\mathbf{I}. \end{aligned} \tag{7.2}$$

This is equivalent to the ML estimate for the optimal vocabulary Δ, relation $\mathcal{R}$, and parameter Θ, as it was formulated in [59]

$$(\Delta, \mathcal{R}, \Theta)^* = \arg\max \sum_{i=1}^{N} \log p(\mathbf{I}_i^{\text{obs}}, \mathbf{pg}_i^{\text{obs}}; \Theta, \mathcal{R}, \Delta) - \ell(V_T, V_N, N), \tag{7.3}$$

where $\ell(V_T, V_N, N)$ is a term that shall balance the model complexity w.r.t. sample size N but also account for the semantic significance of each elements for the vision purpose (human guided here). The latter is often reflected by utility or cost functions in Bayesian decision theory.

Learning the probability model includes three phases and all three phases follow the same principle above [59].

1. Estimating the parameters Θ from training date D^{obs} for given $\mathcal{R}$ and Δ,
2. Learning and pursuing the relation set $\mathcal{R}$ for nodes in $\mathcal{G}$ given Δ,
3. Discovering and binding the vocabulary Δ and hierarchic And–Or tree automatically.

In the following we briefly discuss the first two phases. There is no significant work done for the third phase yet.

7.1 Maximum Likelihood Learning of Θ

For a given And–Or graph hierarchy and relations, the estimation of Θ follows the MLE learning process. Let $\mathcal{L}(\Theta) = \sum_{i=1}^{N} \log p(\mathbf{I}_i^{\text{obs}}, \mathbf{pg}_i^{\text{obs}}; \Theta, \mathcal{R}, \Delta)$ be the log-likelihood, by setting $\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta} = 0$, we have the following three learning steps.

1. Learning the λ_v at each Or-node $v \in V^{\text{or}}$ accounts for the frequency of each alternative choice. The switch variable at v has $n(v)$ choices $\omega(v) \in \{\emptyset, 1, 2, \ldots, n(v)\}$ and it is $\emptyset$ when v is not included in the **pg**. We compute the histogram,

$$\mathbf{h}_v^{\text{obs}}(\omega(v) = i) = \frac{\#(\omega(v) = i)}{\sum_{j=1}^{n(v)} \#(\omega(v) = j)}, \quad i = 1, 2, \ldots, n(v). \tag{7.4}$$

$\#(\omega(v) = i)$ is the number of times that node v appears with $\omega(v) = i$ in all the parse graphs in $\Omega_{\mathbf{pg}}^{\text{obs}}$. Thus,

$$\lambda_v(\omega(v) = i) = -\log \mathbf{h}_v^{\text{obs}}(\omega(v) = i), \quad \forall v \in V^{\text{or}}. \tag{7.5}$$

2. Learning the potential function $\lambda_t()$ at the terminal node $t \in V_T$. $\frac{\partial \ell(\Theta)}{\partial \lambda_t} = 0$ leads to the statistical constraints,

$$E_{p(\mathbf{pg}; \Theta, \mathcal{R}, \Delta)}[\mathbf{h}(\alpha(t)] = \mathbf{h}_t^{\text{obs}}, \quad \forall t \in V_T. \tag{7.6}$$

In the above equation, $\alpha(t)$ are the attributes of t and $\mathbf{h}(\alpha(t))$ is a statistical measure of the attributes, such as the histogram. $\mathbf{h}_t^{\text{obs}}$ is the observed histogram pooled over all the occurrences of t in $\Omega_{\mathbf{pg}}^{\text{obs}}$.

$$\mathbf{h}_t^{\text{obs}}(z) = \frac{1}{\#t} \sum_t \mathbf{1}\left(z - \frac{\epsilon}{2} < \alpha(t) \le z + \frac{\epsilon}{2}\right). \tag{7.7}$$

$\#t$ is the total number of times, a terminal node t appears in the data $\Omega_{\mathbf{pg}}^{\text{obs}}$. z indexes the bins in the histogram and ϵ is the length of a bin.

3. Learning the potential function $\lambda_{ij}()$ for each pair relation $(i,j) \in \mathcal{R}$. $\frac{\partial \ell(\Theta)}{\partial \lambda_{ij}} = 0$ leads to the following implicit function,

$$E_{p(\mathbf{pg};\Theta,\mathcal{R},\Delta)}[\mathbf{h}(v_i,v_j)] = \mathbf{h}_{ij}^{\text{obs}}, \quad \forall (i,j) \in \mathcal{R}. \tag{7.8}$$

Again, $\mathbf{h}(v_i,v_j)$ is a statistic on v_i,v_j, for example, a histogram on the relative size, position, and orientation, appearance etc. $\mathbf{h}_{ij}^{\text{obs}}$ is the histogram summed over all the occurrence of (v_i,v_j) in D^{obs}.

The equations (7.5), (7.6), and (7.8) are the constraints for deriving the Gibbs model $p(\mathbf{pg};\Theta,\mathcal{R},\Delta)$ in Equation (6.12) through the maximum entropy principle.

Due to the coupling of the energy terms, both Equations (7.6) and (7.8) are solved iteratively through a gradient method. In a general case, we follow the stochastic gradient method adopted in learning the FRAME model [90], which approximates the expectations $E_p[\mathbf{h}(\alpha(t))]$ in Equation (7.6) and $E_p[\mathbf{h}(v_i,v_j)]$ in (7.8) by sample means from a set of synthesized examples. This is the method of analysis-by-synthesis adopted in our texture modeling paper [90]. At the end of this chapter, we show the sampling and synthesis experiments on two object categories — clock and bike in Figures 7.1 and 7.2.

7.2 Learning and Pursuing the Relation Set

Besides the learning of parameters Θ, we can also augment the relation sets $\mathcal{R}$ in an And–Or Graph, and thus pursue the energy terms

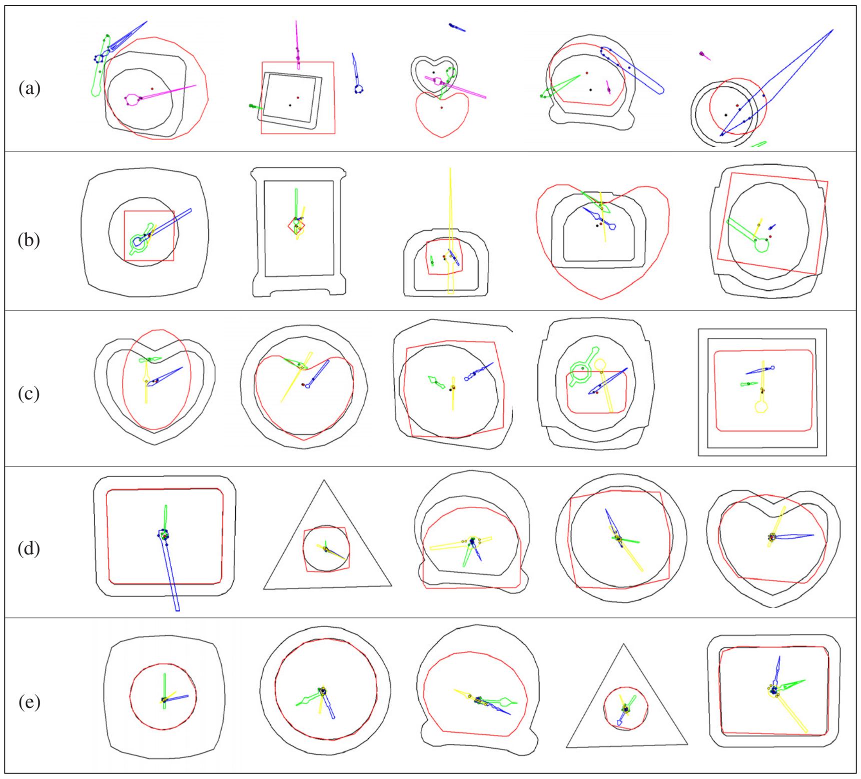


Fig. 7.1 Learning the And–Or graph parameters for the clock category. (a) Sampled clock examples (synthesis) based on SCFG (Markov tree) that accounts for the frequency of occurrence. (b–e) Synthesis examples at four incremental stages of the minimax entropy pursuit process. (b) Matching the relation positions between parts, (c) further matching the relative scales, (d) further pursuing the hinge relation, (e) further matching the containing relation. From [59].

in $\sum_{(i,j)\in E(\mathbf{pg})}\lambda_{ij}(v_i,v_j)$ in the same way as pursuing the filters and statistics in texture modeling by the minimax entropy principle [90].

Suppose we start with an empty relation set $\mathcal{R}=\emptyset$ and thus $p=p(\mathbf{pg};\lambda,\emptyset,\Delta)$ is an SCFG model. The learning procedure is a greedy pursuit. In each step, we add a relation e_+ to $\mathcal{R}$ and thus augment model $p(\mathbf{pg};\Theta,\mathcal{R},\Delta)$ to $p_+(\mathbf{pg};\Theta,\mathcal{R}_+,\Delta)$, where $\mathcal{R}_+=\mathcal{R}\cup\{e_+\}$.

e_+ is selected from a large pool $\Delta_{\mathcal{R}}$ so as to maximally reduce KL-divergence,

$$e_+ = \arg\max \mathrm{KL}(f||p) - \mathrm{KL}(f||p_+) = \arg\max \mathrm{KL}(p_+||p), \qquad (7.9)$$

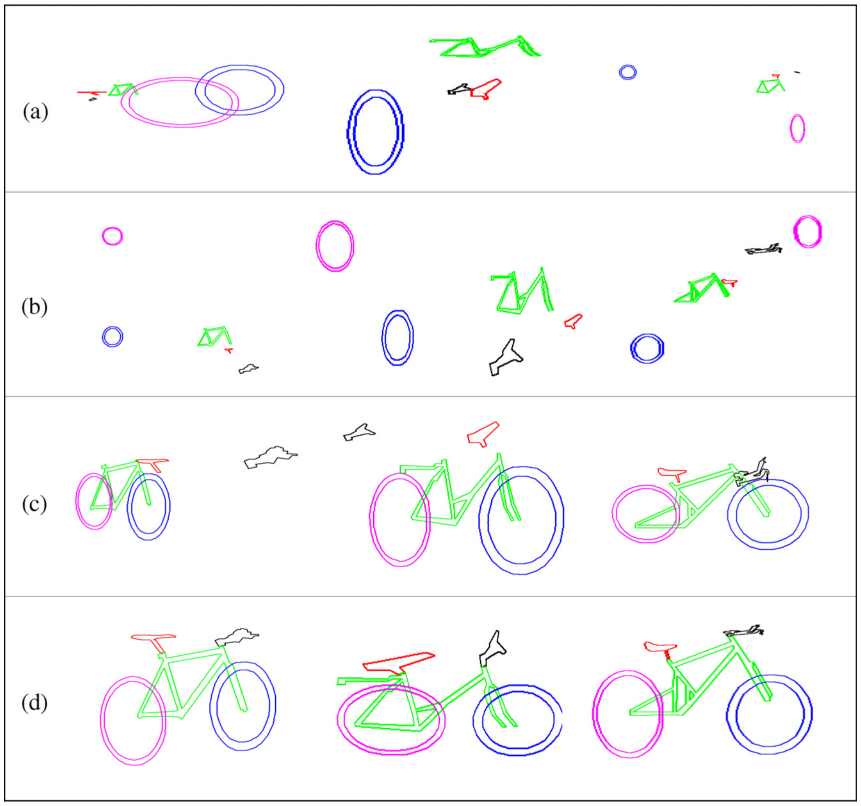


Fig. 7.2 Random sampling and synthesis of the bike category. From [59].

Thus we denote the information gain of e_+ by

$$\delta(e_+) \stackrel{\text{def}}{=} \mathrm{KL}(p_+||p) \approx f^{\text{obs}}(e_+) d_{\text{manh}}(\mathbf{h}^{\text{obs}}(e_+), \mathbf{h}_p^{\text{syn}}(e_+)). \qquad (7.10)$$

In the above formula, $f^{\text{obs}}(e_+)$ is the frequency that relation e_+ is observed in the training data, $\mathbf{h}^{\text{obs}}(e_+)$ is the histogram for relation e_+ over training data D^{obs}, and $\mathbf{h}_p^{\text{syn}}(e_+)$ is the histogram for relation e_+ over the synthesized parse graphs according to the current model p. $d_{\text{manh}}()$ is the Manhanonabis distance between the two histograms.

Intuitively, $\delta(e_+)$ is large if e_+ occurs frequently and tells a large difference between the histograms of the observed and the synthesized parse graphs. Large information gain means a significant relation e_+.

Algorithm 7.1. Learning Θ by Stochastic Gradients

```
Input: $D^{\mathrm{obs}} = \{\mathbf{pg}_i^{\mathrm{obs}}; i = 1, 2, \ldots, M\}$.
 1. Compute histograms $\mathbf{h}_v^{\mathrm{obs}}, \mathbf{h}_t^{\mathrm{obs}}, \mathbf{h}_{ij}^{\mathrm{obs}}$ from $D^{\mathrm{obs}}$ for all feature/relations.
 2. Learn the parameters $\lambda_v$ at the Or-nodes by Equation (7.5).
 3. Repeat (outer loop)
 4.     Sample a set of parse graphs from the current model
            $p(\mathbf{pg}; \Theta, \mathcal{R}, \Delta)\ D^{\mathrm{syn}} = \{\mathbf{pg}_i^{\mathrm{syn}};\ i = 1, 2, ..., M'\}$
 5.     Compute histograms $\mathbf{h}_t^{\mathrm{syn}}, \mathbf{h}_{ij}^{\mathrm{syn}}$ from $D^{\mathrm{syn}}$ for all feature/relations
 6.     Select a feature/relation that maximizes the difference between obs. vs syn. histograms.
 7.     Set $\lambda = 0$ for the newly selected feature/relation.
 8.     Repeat (inner loop)
 9.         Update the parameters with stepsize $\eta$
                $\delta\lambda_t = \eta_t\,(\mathbf{h}_t^{\mathrm{syn}} - \mathbf{h}_t^{\mathrm{obs}}), \quad \forall t \in V_T,$
                $\delta\lambda_{ij} = \eta_{ij}\,(\mathbf{h}_{ij}^{\mathrm{syn}} - \mathbf{h}_{ij}^{\mathrm{obs}}), \quad \forall (i,j) \in \mathcal{R}.$
            Sample a set of parse graphs and update the histograms.
10.     Until $|\mathbf{h}_t^{\mathrm{syn}} - \mathbf{h}_t^{\mathrm{obs}}| \le \epsilon$ and $|\mathbf{h}_{ij}^{\mathrm{syn}} - \mathbf{h}_{ij}^{\mathrm{obs}}| \le \epsilon$ for the selected feature/relations.
11. Until $|\mathbf{h}_t^{\mathrm{syn}} - \mathbf{h}_t^{\mathrm{obs}}| \le \epsilon$ and $|\mathbf{h}_{ij}^{\mathrm{syn}} - \mathbf{h}_{ij}^{\mathrm{obs}}| \le \epsilon$ for all features and relations.
Equations (7.6) and (7.8) are then satisfied to certain precision.
```

7.3 Summary of the Learning Algorithm

In summary, the learning algorithm starts with an SCFG (Markov tree) and a number of observed parse graphs for training D^{obs}. It first learns the SCFG model by counting the occurrence frequency at the Or-nodes. Then by sampling this SCFG, it synthesizes a set of instances D^{syn}. The sampled instances in D^{syn} will have the proper components but often have wrong spatial relations among the parts as there are no relations specified in SCFG. Then the algorithm chooses a relation that has the most different statistics (histogram) over some measurement between the sets D^{obs} and D^{syn}. The model is then learned to reproduce the observed statistics over the chosen relation. A new set of synthesized instances is sampled. This iterative process continues until no more significant differences are observed between the observed and synthesized sets.

Remark 1. At the initial step, the synthesized parse graphs will match the frequency counts on all Or-nodes first, but the synthesized parse graphs and their configurations will not look realistic. Parts of the objects will be in wrong positions and have wrong relations. The iterative steps will make improvements. Ideally, if the features and statistical constraints selected in Equations (7.6) and (7.8) are sufficient, then the synthesized configurations

$$\Omega_{\mathcal{C}}^{\text{syn}} = \{\mathcal{C}_i^{\text{syn}} : \mathbf{pg}_i^{\text{syn}} \longrightarrow \mathcal{C}_i^{\text{obs}},\ i = 1, 2, \ldots, M'\}. \tag{7.11}$$

should resemble the observed configurations. This is what people did in texture synthesis.

Remark 2. Note that in the above learning process, a parse graph $\mathbf{pg}_i^{\text{obs}}$ contributes to some parameters only when the corresponding nodes and relations are present in $\mathbf{pg}_i^{\text{obs}}$.

7.4 Experiments on Learning and Sampling

In [89], the first author showed a range of image synthesis experiments by sampling the image model (ensembles) for various visual patterns, such as textures, texton processes, shape contours, face etc. to verify the learned model in the spirit of analysis-by-synthesis. In this subsection, we show synthesis results in sampling the probabilistic ensemble (or the language) defined by the grammar, i.e., sampling the typical configurations from the probabilistic model defined on the And–Or graph.

$$\mathcal{C} \sim \mathbf{L}(G_{\text{and-or}}) = \left\{(\mathcal{C}_k, p(\mathcal{C}_k)) : S \overset{G_{\text{and-or}}}{\Longrightarrow} \mathcal{C}_k\right\}. \tag{7.12}$$

This is equivalent to first sampling the parse graphs,

$$\mathbf{pg}; \sim\ p(\mathbf{pg}; \Theta, \Delta), \tag{7.13}$$

and then producing the configurations,

$$\mathbf{pg} \rightarrow \mathcal{C}. \tag{7.14}$$

Figure 7.1 illustrates the synthesis process for a clock category whose And–Or graph is shown previously in Figure 6.1. The experiment is from (Porway, Yao and Zhu) [59]. Each row in Figure 7.1 shows five typical examples from the synthesis set $\Omega^{\mathrm{syn}}_{\mathbf{pg}}$ in different iterations. In the first row, the clocks are sampled from the SCFG (Markov tree) in a window. These examples have valid parts for clocks shown in different colors, but there are no spatial relations or features to constrain the attributes of the component or layouts. Thus the instances look quite wrong. In the second row, the relative positions of the components (in terms of their centers) are considered. After matching the statistics of the synthesized and observed sets, the sampled instances look more reasonable. In the third, fourth, and fifth rows, the statistics on the relative scale, the hinge relation between clock hands, and a containing relation are added one by one. The synthesized instances become more realistic configurations.

Figure 7.2 shows the same random sampling and synthesis experiment on another object category — bike. With more spatial relations included and statistics matched, the sampled bikes from the learning models become more realistic from (a) to (d).

The synthesis process produces novel configurations not seen in the observed set and also demonstrates that the spatial relations captured by the And–Or graph will provide information for top-down prediction of object components. Figure 9.9 shall show an example of top-down prediction and hallucination of occluded parts using the learned bike model above.

In a recent experiment on a recognition task with 33 object categories [44], Lin et al. used the synthesized samples to augment the training set and showed that the generalized examples can improve the recognition performance by 15% in comparison to the expertiments without synthesized examples.

8

Recursive Top-Down/Bottom-Up Algorithm for Image Parsing

This chapter briefly reviews an inference algorithm with three case studies of image parsing using grammars by the author and collaborators. The first case is a generic grammar for man-made world scenes. The compositional objects include buildings (indoor or outdoor) and furniture [32]. The second is a more restrictive grammar for human clothes and upper body [9]. The third case [86] applies the grammar for recognizing five object categories — clock, bike, computer (screen and keyboard), cup/bowl, teapot. In both cases, the inference is performed under the Bayesian framework. Given an input image $\mathbf{I}$ as the terminal configuration, we compute a parse graph $\mathbf{pg}$ that maximizes a posterior probability

$$\mathbf{pg}^* = \arg\max_{\mathbf{pg}} p(\mathbf{I}|\mathbf{pg}; \Delta_{\text{sk}}) p(\mathbf{pg}; \Theta, \Delta). \tag{8.1}$$

The likelihood model is based on the primal sketch in Section 3.2, and the prior is defined by the grammar model in Equation (6.12).

In the following, we briefly review the computing procedures, and refer to the original papers [32] and [9] for more details.

The And–Or graph is defined recursively, as is the inference algorithm. This recursive property largely simplifies the algorithm design and makes it easily scalable to arbitrarily large number of object categories.

Consider an arbitrary And-node A in an And–Or graph. A may correspond to an object or a part. Without loss of generality, we assume it can be either terminated into one of n leaves at low resolution or decomposed into $n(A) = 3$ parts,

$$A \to A_1 \cdot A_2 \cdot A_3 \,|\, t_1 \,|\, \cdots \,|\, t_n. \tag{8.2}$$

This recursive unit is shown in Figure 8.1.

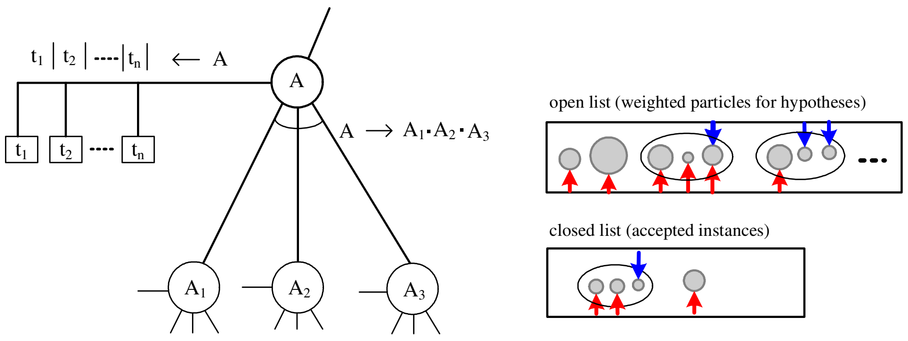
In this figure, each such unit is associated with data structures which are widely used in heuristic searches in artificial intelligence [58].

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

Thus the inference algorithm consists of two basic processes that compute and maintain the Open and Closed lists for each unit A.

The bottom-up process creates the particles in the Open lists in two methods.

(i) Generating hypotheses for A directly from images. Such bottom-up processes include detection algorithms such as



t1 | t2 | ----| tn | ← A
A
t1
t2
tn
A → A1·A2·A3
A1
A2
A3
open list (weighted particles for hypotheses)
closed list (accepted instances)

Fig. 8.1 Data structure for the recursive inference algorithm on the And–Or graph. See text for interpretation.

Adaboosting [21, 78], Hough transform etc. for detecting the various terminals $t_1, \ldots, t_n$ without identifying the parts. The detection process tests some image features. These particles are shown in Figure 8.1 by single circles with bottom-up arrows. The weight of a detected hypothesis (indexed by i) is the logarithm of some local marginal posterior probability ratio given a small image patch Λ^i,

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

$\bar{A}$ means competitive 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 $\mathbf{I}_{\lambda^i}$. For example, in face detection by Adaboosting [78], the strong classifier can be reformulated as a posterior probability ratio of face vs. non-face [21, 63].

(ii) Generating hypotheses for A by binding a number of k $(1 \leq K \leq n(A))$ parts from the existing Open and Closed lists of its children $A_1, A_2, \ldots, A_{n(A)}$. The binding process will test the relationships between these child nodes for compatibility and quickly rule out the obviously incompatible compositions. In Figure 8.1, these hypotheses are illustrated by a big ellipse containing $n(A) = 3$ small circles for its children. The upward arrows show existing parts in the Open or Closed lists of the child nodes, and the downward arrows show the missing parts that need to be validated in the top-down process. The weight of a bound 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

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

where $\bar{A}$ means competitive hypothesis. $p(A^i_1, A^i_2|A^i)$ is reduced to tests of compatibility between A^i_1 and A^i_2 for computational efficiency. It leaves the computation of searching for A^i_3 as well as fitting the image area $\mathbf{I}_{\Lambda_A}$ to the top-down process.

The top-down process validates the bottom-up hypotheses in all the Open lists, following the Bayesian posterior probability. It also needs to maintain the weights of the Open lists.

(i) Given a hypothesis A^i with weight $\hat{\omega}^i_A$, the top-down process validates it by computing the true posterior probability ratio ω^i_A stated above. If A^i is accepted into the Closed list of A. This corresponds to a move from the current parse graph $\mathbf{pg}$ to a new parse graph $\mathbf{pg}_+$. The latter includes a new node A^i – either as a leaf node or as a non-terminal node with children $A^i_1, \ldots, A^i_{n(A)}$. 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 deletes it (putting it back to the Open list) or disassembles it into independent parts.

(ii) Maintaining the weights of the particles in the OPEN Lists after adding (or removing) a node A^i from the parse graph. 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 in the parse graph, all the other hypotheses whose domains overlap with that of A will have to update their weights.

The acceptance of a node can be done by a greedy algorithm that maximizes the posterior probability. Each time 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.

Otherwise, one may use a stochastic algorithm with reversible jumps. According to the terminology of data driven Markov chain

Monte Carlo (DDMCMC) [73, 74], one may view the approximative weight $\hat{\omega}_A^i$ as a logarithm of the proposal probability ratio. The acceptance probability, in the Metropolis–Hastings method [46], is thus

$$\begin{aligned} a(\mathbf{pg} \to \mathbf{pg}_+) &= \min\left(1, \frac{q(\mathbf{pg}_+ \to \mathbf{pg})}{q(\mathbf{pg} \to \mathbf{pg}_+)} \cdot \frac{p(\mathbf{pg}_+|\mathbf{I})}{p(\mathbf{pg}|\mathbf{I})}\right) \\ &= \min\left(1, \frac{q_+(A^i)}{q(A^i)} \exp\{\omega_A^i - \hat{\omega}_A^i\}\right), \end{aligned}$$

where $q_+(A^i)$ (or $q(A^i)$) is the proposal probability for selecting A^i to be disassembled from $\mathbf{pg}_+$ (to be added to $\mathbf{pg}$).

For the stochastic algorithm, its initial stage is often deterministic when the particle weights are very large and the acceptance probability is always 1.

We summarize the inference algorithm in the following:

Algorithm 8.1. Image Parsing by Top-down/Bottom-up Inference

Input: an image $\mathbf{I}$ and an And–Or graph.
Output: a parse graph $\mathbf{pg}$ with initial $\mathbf{pg} = \emptyset$.

```
1. Repeat
2. Schedule the next visit note $A$
3. Call the Bottom — Up(A) process to update $A$'s Open lists
4. (i) Detecting terminal instances for $A$ from images
5. (ii) Binding non-terminal instances for $A$ from its children's
   Open or Closed lists.
6. Call the 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.
```

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. For some visual patterns, like human faces in Figure 2.10, it is perhaps more effective to detect the whole face and then locate the facial components. For other visual patterns, like the cheetah image in Figure 2.9, it is more effective to work in a bottom-up fashion. More objects, like the two examples in the following two subsections, need to alternate between the bottom-up and top-down processes.

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. A special case is studied in [7] for coarse-to-fine testing. Many popular algorithms in AI heuristic search [58] or the matching pursuit [47] can be considered deterministic versions of the above algorithm. In DDMCMC [73, 92], the algorithm always performs all the necessary bottom-up tests before running the top-down process. As does the feedforward neural networks [61]. This may not be the optimal schedule.

9

Three Case Studies of Image Grammar

9.1 Case Study I: Parsing the Perspective Man-Made World by Han and Zhu

In this case, the grammar has one class of primitives as the terminal nodes (i.e., V_T), which are 3D planar rectangles projected on images. Obviously rectangles are the most common elements in man-made scenes, such as buildings, hallways, kitchens, living rooms, etc. Each rectangle $a \in V_T$ is made of two pairs of parallel line segments in 3D space, which may intersect at two vanishing points through projection. The grammar has only two types of non-terminal nodes (i.e., V_N) — the root node S for the scene and a node A for any composite objects. The grammar has six production rules as shown in Figure 9.1. The scene node S generates m independent objects (rule r_1). An object node A can be instantiated (assigned) to a rectangle (rule r_5), or be used recursively by the other four production rules: r_2 — the line production rule that aligns a number of rectangles in one row, r_3 — the mesh production rule that arranges a number of rectangles in a matrix, r_4 — the nesting production rule that has one rectangle containing the other, and r_6 — the cube production rule that aligns three rectangle

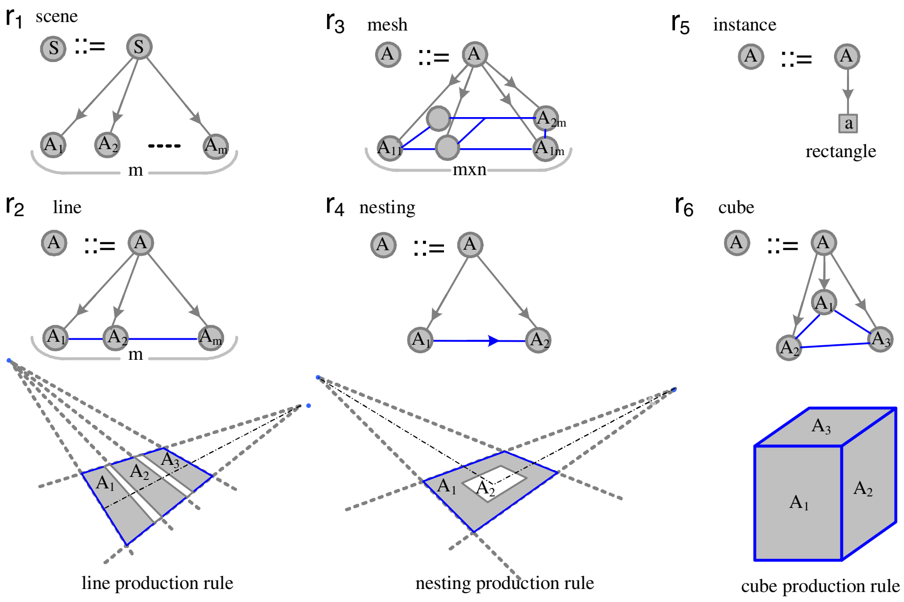


Fig. 9.1 Six attribute grammar rules for generic man-made world scenes. This grammar features a single class of primitives — rectangle and four generic organizations — line, mesh, cube, and nesting. Attributes will be passed between a node to its children and the horizontal lines show constraints on attributes. See text for explanation.

into a solid shape. The unknown numbers m and n can be represented by the Or-nodes for different combinations.

Each production rule is associated with a number of equations that constrain the attributes of a parent node and those of its children. These rules can be used recursively to generate a large set of complex configurations. Figure 9.2 shows two typical parsing configurations — (b) a floor pattern and (d) a toolbox pattern, and their corresponding parse graphs in (a) and (c), respectively.

The parsing algorithm adopts a greedy method following the general description of Algorithm 8.1. For each of the 5 rules $r_2, \ldots, r_6$, it maintains an Open list and a Closed list. In an initial phase, it detects an excessive number of rectangles in by a bottom-up rectangle detection process and thus fill the Open list for rule r_5. Each particle consists of two pairs of parallel line segments.

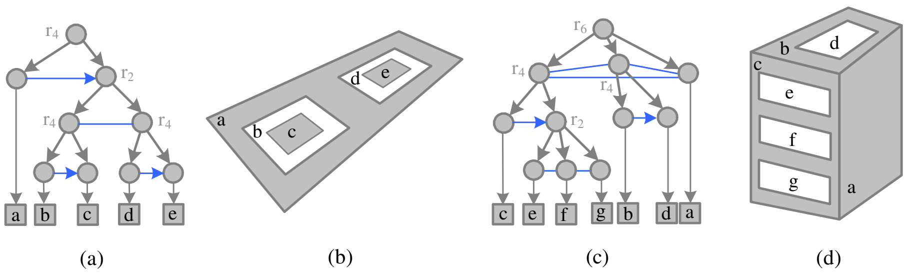


Fig. 9.2 Two examples of rectangle object configurations (b) and (d) and their corresponding parse graphs (a) and (c). The production rules are shown as non-terminal nodes.

The top-down and bottom-up computation has been illustrated in Figure 1.2 for a kitchen scene. Figure 1.2 shows a parse graph under construction at a time step, the four rectangles (in red) are the accepted rectangles in the Closed list for r_5. They activated a number of candidates for larger groups using the production rules r_3, r_4, r_6, respectively, and three of these candidates are then accepted as non-terminal nodes A, B, and C, respectively. The solid upward arrows show the bottom-up binding, while the downward arrows show the top-down prediction.

Figure 9.3 shows the five Open lists for the candidate sets of the five rules. At each step the parsing algorithm will choose the candidate with the largest weight from the five particle sets and add a new non-terminal node to the parse graph. If the particle is in the r_5 Open list, it means accepting a new rectangle. Otherwise the algorithm creates a non-terminal node and inserts the missing children in this particle into their respective Open lists for future tests.

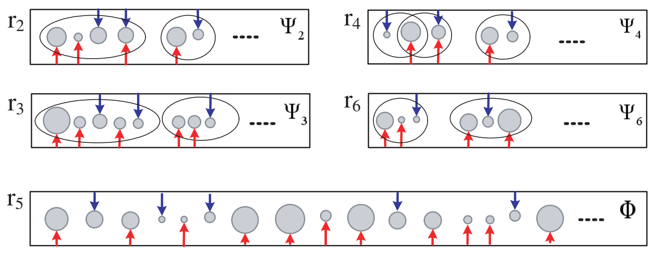


Fig. 9.3 Illustration for the open lists of the five rules.

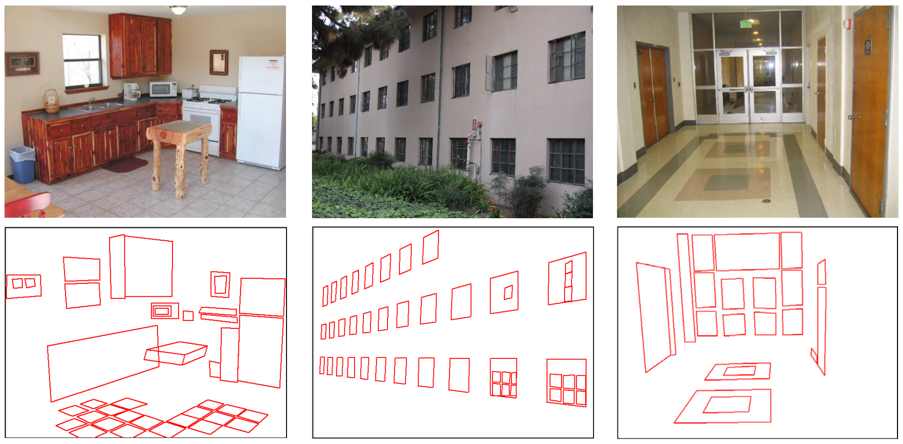


Fig. 9.4 Some experimental results. The first row shows the input images. The second row shows the computed rectangle configurations. From [32].

Figure 9.4 shows three examples of the inference algorithm. The computed configuration $\mathcal{C}$ for each image consists of a number of rectangles arranged in generic structures. More discussions and experiments are referred to [32].

Figure 9.5 shows two ROC curves for performance comparison in detecting the rectangles in 25 images against human annotated groundtruth. One curve shows the detection rate (vertical axis) over the number of false alarms per image (horizontal axis) for pure bottom-up method. The other curve is for the methods integrating bottom-up and top-down. From these ROC curves, we can clearly see the dramatic improvement by using top-down mechanism over the traditionally bottom-up mechanism only. Intuitively, some rectangles are nearly impossible to detect using the bottom-up methods and can only be recovered through the context information using the grammar rules.

9.2 Case Study II: Human Cloth Modeling and Inference by Chen, Xu, and Zhu

The second example, taken from [9], represents and computes clothes by And–Or graph. Unlike the rigid rectangle objects in the first example,

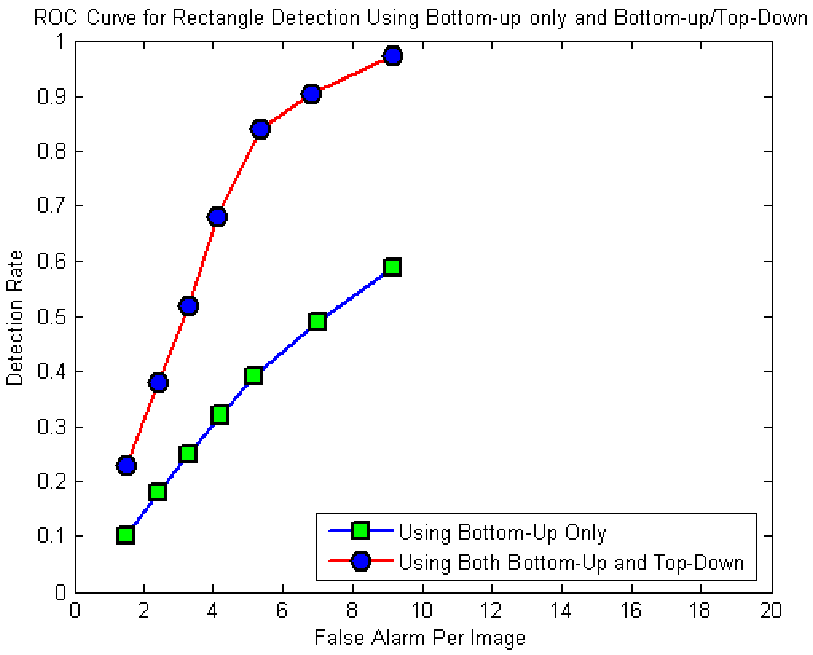


Fig. 9.5 ROC curves for the rectangle detection results by using bottom-up only and, using both bottom-up and top-down. From [32].

human clothes are very flexible objects with large intra-category structural variations.

The authors in [9] took 50 training images of college students sitting in a high chair with good light conditions and uniform background to reduce occlusion and control illumination. An artist was asked to draw sketches as consistent as possible on these images. From the sketches, they manually separate a layer of sketches corresponding to shading folds and textures (e.g., shoe lace, text printed on T-shirt), and then decompose the remaining structures into a number of parts: hair, face, collar, shoulder, upper and lower arms, cuff, hands, pants, shoes, and pockets. Some of the examples are shown in Figure 3.8. The largest two categories are hands and shoes. The hands have many possible configurations — separate or held/crossed. The 50 pairs of hands collected are not necessarily exhaustive. However, an interesting observation in the experiment is that human vision is not very sensitive to the precise

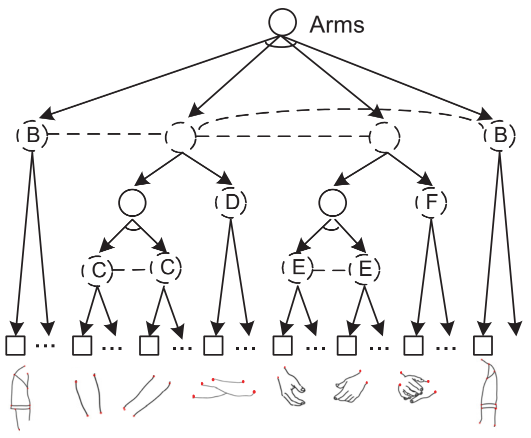


Fig. 9.6 The And–Or graph for arms as a part of the overall And–Or graph.

hand gesture/poses. If a test image has a hand configuration outside of our training category, the algorithm will find a closest match and simply paste the part at the hand position without noticeable difference. Therefore complex parts, such as hands and shoes, can be treated less precisely.

With these categories, an And–Or graph is constructed manually to account for the variability of configurations. A portion of the And–Or graph for arms and hands is shown in Figure 9.6. Intuitively, this And–Or graph is like a "mother template" and it can produce a large set of configurations including configurations not seen in the training set. Figure 9.7 displays three configurations produced by this And–Or graph.

This And–Or graph is then used for drawing clothes from images using a version of algorithm II. The algorithm makes use of the bottom-up process for detecting parts that are most discriminable, such as face, skin color, shoulder. Then it activates top-down searches for predicted parts based on the context information encoded in the And–Or graph. Figure 9.8 shows three results of the computed configurations. These graphical sketches are quite nice for they are generated by rearranging the artist's parts. Such results have potential applications in digital arts and cartoon animations.

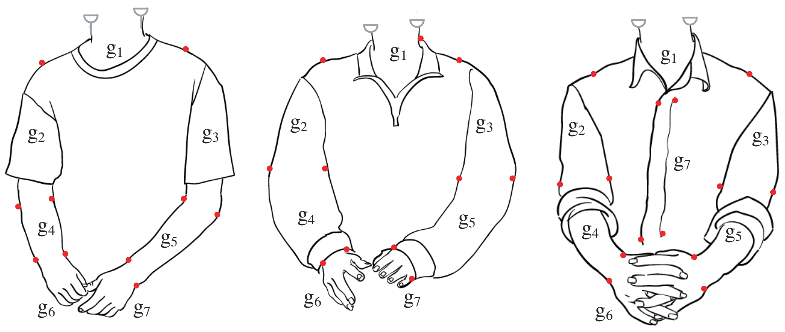


Fig. 9.7 Three novel configurations composed of 6,5,7 sub-templates in the categories, respectively. The bonds are shown by the red dots.

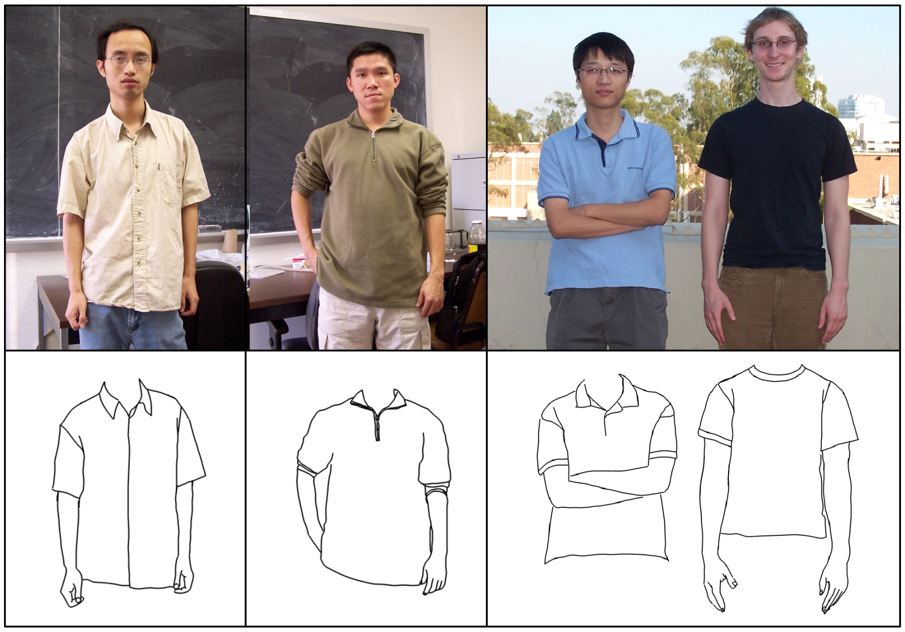


Fig. 9.8 Experiment on inferring upper body with clothes from images. From [9].

9.3 Case Study III: Recognition on Object Categories by Xu, Lin, and Zhu

The third example, taken from [86], applies the top-down/bottom-up inference to five object categories — clock, bike, computer (screen and keyboard), cup/bowl, and teapot. The five categories are selected from a large scale ground truth database from the Lotus Hill Institute. The database includes more than 500,000 objects over 200 categories parsed in And–Or graphs [87]. The probabilistic models are learned for these And–Or graphs using the MLE learning presented in the previous section. The clock and bike sampling results were shown in Figures 7.1 and 7.2.

The And–Or graphs together with their probabilistic models represent the prior knowledge above the five categories for top-down inference. Figure 9.9 shows an example of inferring a partially occluded bicycle from clutter.

In Figure 9.9, the first row shows the input image, an edge map, and bottom-up detection of the two wheels using Hough transform. The

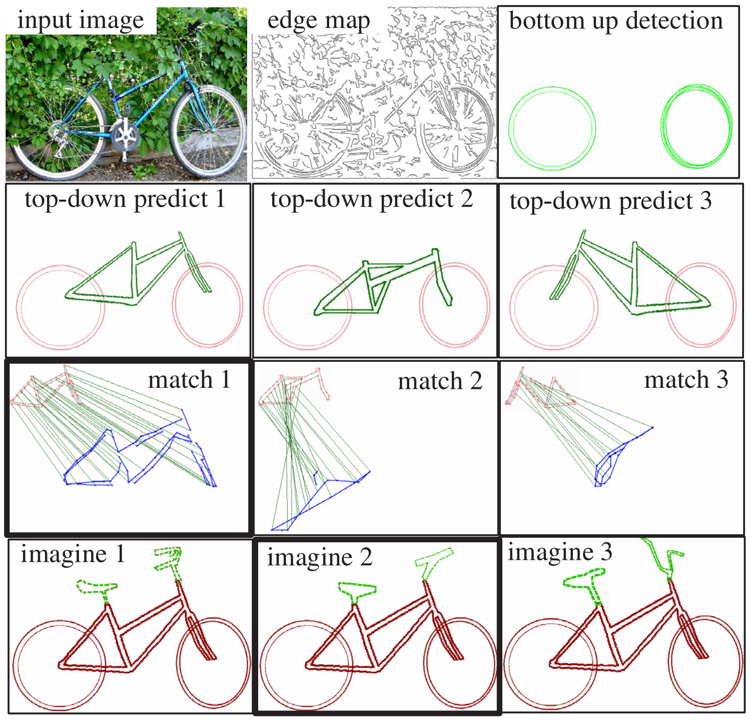


Fig. 9.9 The top-down influence in inferring a partially occluded bike from clutter. From [86].

Hough transform method is adopted to detect parts like circles, ellipses, and triangles. The second row shows some top-down predictions of bike frame based on the two wheels. The transform parameters of the bike frame are sampled from the learned MRF model. As we cannot tell the front wheel from the rear at this moment, the frames are sampled for both directions. We only show three samples for clarification. The third row shows the template matching process that matching the predicted frames (in red) to the edges (in blue) in the image. The one with minimum matching cost is selected. The fourth row shows the top-down hallucinations (imaginations) for the seat and handlebar (in green). As these two parts are occluded. The three sets of hallucinated parts are randomly sampled from the And–Or graph model, in the same way as random sampling of the whole bike.

Finally, we show a few recognition examples in Figure 10.1 for the five categories. For each input image, the image on its right-side shows the recognized parts from the image in different colors. It should be mentioned that the recognition algorithm is distinct from most of the classification algorithms in the literature. It interprets the image by a parse graph which includes the classification of categories and parts on the Or-nodes, and matches the leaf templates to images, and hallucinates occluded parts.

10

Summary and Discussion

This exploratory paper is concerned with representing large scale visual knowledge in a consistent modeling, learning, and computing framework. Specifically two huge problems must be solved before a robust vision system is feasible: (i) large number (hundreds) of object and scene categories; and (ii) large intra-category structural variation. The framework proposed to tame these two problems is a stochastic graph grammar embedded in an And–Or graph, which can be learned from a large annotated dataset.

First, to represent intra-category variation, the grammar can create a large number of configurations from a relatively much smaller vocabulary. The And–Or graph acts like a reconfigurable mother template, and assembles novel configurations on-the-fly to interpret novel instances unseen before.

Second, to scale up to hundreds of categories, the And–Or graph is recursively designed. Thus one can integrate, without much overhead, all categories into one big And–Or graph. The learning and inference algorithms are designed recursively as well. This permits large scale parallel computing.

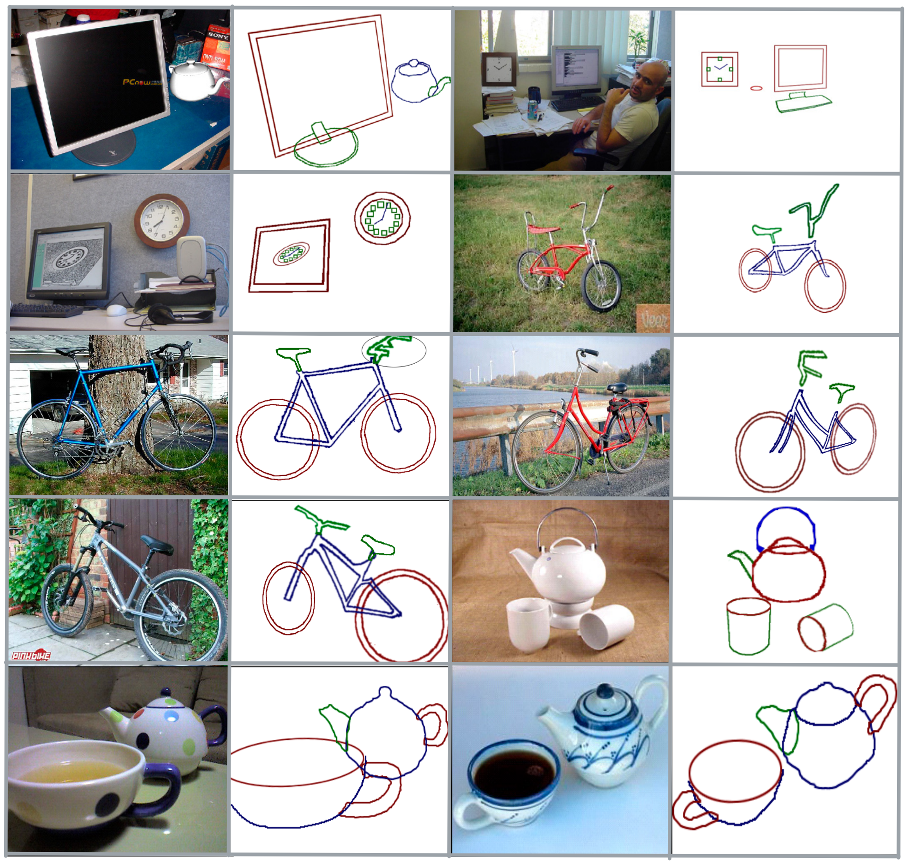


Fig. 10.1 Recognition experiments on five object categories. From [86].

There are two open issues for further study.

(i) *Learning and discovering the And–Or graph.* As it was proposed in a series of recent works [17, 52, 59, 81, 86], the objective is to map the visual vocabulary including dictionaries at all levels of abstraction and all visual aspects. This task can be formulated in theory under a common learning principle, that is to put the dictionary Δ into the maximum likelihood learning process. The various information criteria, such as the binding strength, mutual information, minimax entropy, will come naturally out of this learning process.

However, the ultimate visual vocabulary is unlikely to be learned fully automatically from statistical principles, as the determination of the vocabulary must take the purposes of vision into account. This argues for a semi-automatic method which is being carried out at the Lotus Hill Institute. Human users, guided by real life experience, psychology and vision tasks, define most of the structures, and leaving the estimation of parameters and adaptation to computers. The computers, at a more sophisticated stage, should be able to find and pursue the addition of novel elements in their dictionaries. So far, And–Or graphs have been constructed for over 200 object and scene categories, including aerial images, at the Lotus Hill Institute [87].

(ii) *Scheduling and ordering of top-down and bottom-up processes.* When we have a big And–Or graph with thousands of nodes organized hierarchically, we can imagine that the computing process is like a many-story factory with thousands of assembly lines. Intuitively, each assembly line corresponds to the Open and Closed lists of a node in the And–Or graph. With all these assembly lines sharing only one CPU (or even multiple CPUs), it is crucial to optimize the schedule to maximize the total throughput of the factory. Traditionally, vision algorithms always start with bottom-up processes to feed the assembly lines with raw materials (proposing weighted hypothesis), for example, the DDMCMC [73, 92], and feed-forward neural networks [61]. Due to the multi-resolution property, each node in the And–Or graph can be terminated immediately and thus the raw material can be sent to the assembly lines at all stories of the factory directly, instead of going up story-by-story. This strategy is supported by human vision experiments [18, 70] that show humans can detect scene and object categories as fast as we detect the low level textons and primitives.

There has been a long standing debate over the roles of top-down and bottom-up processes [76]. We believe that this debate can only be answered numerically not verbally. That is to say, we need to compute, numerically, the information gain of each operator, either top-down or bottom-up, over the ensemble of real-world images.

Acknowledgments

The authors thank Drs. Stuart Geman, Yingnian Wu, Harry Shum, Alan Yuille, and Joachim Buhmann for their extensive discussions and helpful comments. The first author also thanks many students at UCLA (Hong Chen, Jake Porway, Kent Shi, Zijian Xu) and the Lotus Hill Institute (Liang Lin, Zhenyu Yao, Tianfu Wu, Xiong Yang, et al.) for their assistance. The work is supported by a NSF grant IIS-0413214 and an ONR grant N00014-05-01-0543. The work at the Lotus Hill Institute is supported by a Chinese National 863 grant 2006AA01Z121.

References

[1] S. P. Abney, "Stochastic attribute-value grammars," *Computational Linguistics*, vol. 23, no. 4, pp. 597–618, 1997.

[2] K. Athreya and A.Vidyashankar, *Branching Processes*. Springer-Verlag, 1972.

[3] A. Barbu and S. C. Zhu, "Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities," *IEEE Transactions on PAMI*, vol. 27, no. 8, pp. 1239–1253, 2005.

[4] K. Barnard *et al.*, "Evaluation of localized semantics: Data methodology, and experiments," Tech. Report, CS, U. Arizona, 2005.

[5] I. Biederman, "Recognition-by-components: A theory of human image understanding," *Psychological Review*, vol. 94, pp. 115–147, 1987.

[6] E. Bienenstock, S. Geman, and D. Potter, "Compositionality, MDL priors, and object Recognition," in *Advances in Neural Information Processing Systems 9*, (M. Mozer, M. Jordan, and T. Petsche, eds.), MIT Press, 1998.

[7] G. Blanchard and D. Geman, "Sequential testing designs for pattern recognition," *Annals of Statistics*, vol. 33, pp. 1155–1202, June 2005.

[8] H. Blum, "Biological shape and visual science," *Journal of Theoretical Biology*, vol. 38, pp. 207–285, 1973.

[9] H. Chen, Z. J. Xu, Z. Q. Liu, and S. C. Zhu, "Composite templates for cloth modeling and sketching," in *Proceedings of IEEE Conference on Pattern Recognition and Computer Vision*, New York, June 2006.

[10] Z. Y. Chi and S. Geman, "Estimation of probabilistic context free grammar," *Computational Linguistics*, vol. 24, no. 2, pp. 299–305, 1998.

[11] N. Chomsky, *Syntactic Structures*. Mouton: The Hague, 1957.

[12] T. F. Cootes, C. J. Taylor, D. Cooper, and J. Graham, "Active appearance models–their training and applications," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[13] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet based statistical signal processing using hidden Markov models," *IEEE Transactions on Signal Processing*, vol. 46, pp. 886–902, 1998.

[14] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld, "From volumes to views: An approach to 3D object recognition," *CVGIP: Image Understanding*, vol. 55, no. 2, pp. 130–154, 1992.

[15] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechie, "Data compression and harmonic analysis," *IEEE Transactions on Information Theory*, vol. 6, pp. 2435–2476, 1998.

[16] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 100 object categories," *Workshop on Generative Model Based Vision*, 2004.

[17] L. Fei-Fei, R. Fergus, and P. Perona, "One-Shot learning of object categories," *IEEE Transactions on PAMI*, vol. 28, no. 4, pp. 594–611, 2006.

[18] L. Fei-Fei, A. Iyer, C. Koch, and P. Perona, "What do we perceive in a glance of a real-world scene?," *Journal of Vision*, vol. 7, no. 1, pp. 1–29, 2007.

[19] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computer*, vol. C-22, pp. 67–92, 1973.

[20] A. Fridman, "Mixed markov models," *Proceedings of Natural Academy of Science USA*, vol. 100, pp. 8092–8096, 2003.

[21] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Annals of Statistics*, vol. 38, no. 2, pp. 337–374, 2000.

[22] K. S. Fu, *Syntactic Pattern Recognition and Applications.* Prentice-Hall, 1982.

[23] M. Galun, E. Sharon, R. Basri, and A. Brandt, "Texture segmentation by multiscale aggregation of filter responses and shape elements," *Proceedings of ICCV, Nice*, pp. 716–723, 2003.

[24] R. X. Gao, T. F. Wu, N. Sang, and S. C. Zhu, "Bayesian inference for layered representation with mixed Markov random field," in *Proceedings of the 6th International Conference on EMMCVPR*, Ezhou, China, August 2007.

[25] R. X. Gao and S. C. Zhu, "From primal sketch to 2.1D sketch," Technical Report, Lotus Hill Institute, 2006.

[26] S. Geman and M. Johnson, "Probability and statistics in computational linguistics, a brief review," in *Int'l Encyc. of the Social and Behavioral Sciences*, (N. J. Smelser and P. B. Baltes, eds.), pp. 12075–12082, Pergamon: Oxford, 2002.

[27] S. Geman, D. Potter, and Z. Chi, "Composition systems," *Quarterly of Applied Mathematics*, vol. 60, pp. 707–736, 2002.

[28] U. Grenander, *General Pattern Theory.* Oxford University Press, 1993.

[29] G. Griffin, A. Holub, and P. Perona, "The Caltech 256," Technical Report, 2006.

[30] C. E. Guo, S. C. Zhu, and Y. N. Wu, "Modeling visual patterns by integrating descriptive and generative models," *IJCV*, vol. 53, no. 1, pp. 5–29, 2003.

[31] C. E. Guo, S. C. Zhu, and Y. N. Wu, "Primal sketch: Integrating texture and structure," in *Proceedings of International Conference on Computer Vision*, 2003.

[32] F. Han and S. C. Zhu, "Bottom-up/top-down image parsing by attribute graph grammar". *Proceedings of International Conference on Computer Vision,* Beijing, China, 2005. (A long version is under review by PAMI).

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

[34] T. Hong and A. Rosenfeld, "Compact region extraction using weighted pixel linking in a pyramid," *IEEE Transactions on PAMI*, vol. 6, pp. 222–229, 1984.

[35] J. Huang, PhD Thesis, Division of Applied Math, Brown University.

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

[37] B. Julesz, "Textons, the elements of eexture perception, and their interactions," *Nature*, vol. 290, pp. 91–97, 1981.

[38] T. Kadir and M. Brady, "Saliency, scale and image description," *International Journal of Computer Vision*, 2001.

[39] G. Kanisza, *Organization in Vision.* New York: Praeger, 1979.

[40] Y. Keselman and S. Dickinson, "Generic model abstraction from examples," *CVPR*, 2001.

[41] B. Kimia, A. Tannenbaum, and S. Zucker, "Shapes, shocks and deformations I," *Interantional Journal of Computer Vision*, vol. 15, pp. 189–224, 1995.

[42] A. B. Lee, K. S. Pedersen, and D. Mumford, "The nonlinear statistics of high-contrast patches in natural images," *IJCV*, vol. 54, no. 1/2, pp. 83–103, 2003.

[43] M. Leyton, "A process grammar for shape," *Artificial Intelligence*, vol. 34, pp. 213–247, 1988.

[44] L. Lin, S. W. Peng, and S. C. Zhu, "An empirical study of object category recognition: Sequential testing with generalized samples," in *Proceedings of International Conference on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

[45] T. Lindeberg, *Scale-Space Theory in Computer Vision.* Netherlands: Kluwer Academic Publishers, 1994.

[46] J. S. Liu, *Monte Carlo Strategies in Scientific Computing.* NY: Springer-Verlag, p. 134, 2001.

[47] S. Mallat and Z. Zhang, "Matching pursuit in a time-frequency dictionary," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3397–3415, 1993.

[48] K. Mark, M. Miller, and U. Grenander, "Constrained stochastic language models," in *Image Models (and Their Speech Model cousins)*, (S. Levinson and L. Shepp, eds.), IMA Volumes in Mathematics and its Applications, 1994.

[49] D. Marr, *Vision.* Freeman Publisher, 1983.

[50] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms," *ICCV*, 2001.

[51] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-D objects from appearance," *International Journal of Computer Vision*, vol. 14, pp. 5–24, 1995.

[52] K. Murphy, A. Torralba, and W. T. Freeman, "Graphical model for recognizing scenes and objects," *Proceedings of NIPS*, 2003.

[53] M. Nitzberg, D. Mumford, and T. Shiota, "Filtering, segmentation and depth," *Springer Lecture Notes in Computer Science*, vol. 662, 1993.

[54] Y. Ohta, *Knowledge-Based Interpretation of Outdoor Natural Color Scenes.* Pitman, 1985.

[55] Y. Ohta, T. Kanade, and T. Sakai, "An analysis system for scenes containing objects with substructures," in *Proceedings of 4th International Joint Conference on Pattern Recognition*, (Kyoto), pp. 752–754, 1978.

[56] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[57] B. Ommer and J. M. Buhmann, "Learning compositional categorization method," in *Proceedings of European Conference on Computer Vision*, 2006.

[58] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, 1984.

[59] J. Porway, Z. Y. Yao, and S. C. Zhu, "Learning an And–Or graph for modeling and recognizing object categories," Technical Report, Department of Statistics, UCLA, 2007.

[60] J. Rekers and A. Schürr, "A parsing algorithm for context sensitive graph grammars," TR-95–05, Leiden University, 1995.

[61] M. Riesenhuber and T. Poggio, "Neural mechanisms of object recognition," *Current Opinion in Neurobiology*, vol. 12, pp. 162–168, 2002.

[62] B. Russel, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: A database and web-based tool for image annotation," *MIT AI Lab Memo AIM-2005-025*, September 2005.

[63] R. E. Schapire, "The boosting approach to machine learning: An overview," *MSRI Workshop on nonlinear Estimation and Classification*, 2002.

[64] T. B. Sebastian, P. N. Klein, and B. B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Transactions on PAMI*, vol. 26, no. 5, pp. 550–571, 2004.

[65] S. M. Sherman and R. W. Guillery, "The role of thalamus in the flow of information to cortex," *Philosophical Transactions of Royal Society London (Biology)*, vol. 357, pp. 1695–1708, 2002.

[66] K. Shi and S. C. Zhu, "Visual learning with implicit and explicit manifolds," *IEEE Conference on CVPR*, June 2007.

[67] K. Siddiqi and B. B. Kimia, "Parts of visual form: Computational aspects," *IEEE Transactions on PAMI*, vol. 17, no. 3, pp. 239–251, 1995.

[68] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, "Shock graphs and shape matching," *IJCV*, vol. 35, no. 1, pp. 13–32, 1999.

[69] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multi-scale transforms," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 587–607, 1992.

[70] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, pp. 520–522, 1996.

[71] S. Todorovic and N. Ahuja, "Extracting subimages of an unknown category from a set of images," *CVPR*, 2006.

[72] Z. W. Tu, X. R. Chen, A. L. Yuille, and S. C. Zhu, "Image parsing: Unifying segmentation, detection, and recognition," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 113–140, 2005.

[73] Z. W. Tu and S. C. Zhu, "Image segmentation by data-driven Markov chain Monte Carlo," *IEEE Transactions on PAMI*, May 2002.

[74] Z. W. Tu and S. C. Zhu, "Parsing images into regions, curves and curve groups," *International Journal of Computer Vision*, vol. 69, no. 2, pp. 223–249, 2006.

[75] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, p. 1, 1991.

[76] S. Ullman, "Visual routine," *Cognition*, vol. 18, pp. 97–157, 1984.

[77] S. Ullman, E. Sali, and M. Vidal-Naquet, "A fragment-based approach to object representation and classification," in *Proceedings of 4th International Workshop on Visual Form*, Capri, Italy, 2001.

[78] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *CVPR*, pp. 511–518, 2001.

[79] W. Wang, I. Pollak, T.-S. Wong, C. A. Bouman, M. P. Harper, and J. M. Siskind, "Hierarchical stochastic image grammars for classification and segmentation," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 3033–3052, 2006.

[80] Y. Z. Wang, S. Bahrami, and S. C. Zhu, "Perceptual scale space and it applications," in *International Conference on Computer Vision*, Beijing, China, 2005.

[81] M. Weber, M. Welling, and P. Perona, "Towards automatic discovery of object categories," *IEEE Conference on CVPR*, 2000.

[82] A. P. Witkin, "Scale space filtering," *International Joint Conference on AI.* Palo Alto: Kaufman, 1983.

[83] T. F. Wu, G. S. Xia, and S. C. Zhu, "Compositional boosting for computing hierarchical image structures," *IEEE Conference on CVPR*, June 2007.

[84] Y. N. Wu, S. C. Zhu, and C. E. Guo, "From information scaling laws of natural images to regimes of statistical models," *Quarterly of Applied Mathematics*, 2007 (To appear).

[85] Z. J. Xu, H. Chen, and S. C. Zhu, "A high resolution grammatical model for face representation and sketching," in *Proceedings of IEEE Conference on CVPR*, San Diego, June 2005.

[86] Z. J. Xu, L. Lin, T. F. Wu, and S. C. Zhu, "Recursive top-down/bottom-up algorithm for object recognition," Technical Report, Lotus Hill Research Institute, 2007.

[87] Z. Y. Yao, X. Yang, and S. C. Zhu, "Introduction to a large scale general purpose groundtruth database: Methodology, annotation tools, and benchmarks," in *6th International Conference on EMMCVPR*, Ezhou, China, 2007.

[88] S. C. Zhu, "Embedding Gestalt laws in Markov random fields," *IEEE Transactions on PAMI*, vol. 21, no. 11, 1999.

[89] S. C. Zhu, “Statistical modeling and conceptualization of visual patterns,” *IEEE Transactions on PAMI*, vol. 25, no. 6, pp. 691–712, 2003.

[90] S. C. Zhu, Y. N. Wu, and D. B. Mumford, “Minimax entropy principle and its applications to texture modeling,” *Neural Computation*, vol. 9, no. 8, pp. 1627–1660, November 1997.

[91] S. C. Zhu and A. L. Yuille, “Forms: A flexible object recognition and modeling system,” *Interantional Journal of Computer Vision*, vol. 20, pp. 187–212, 1996.

[92] S. C. Zhu, R. Zhang, and Z. W. Tu, “Integrating top-down/bottom-up for object recognition by data-driven Markov chain Monte Carlo,” *CVPR*, 2000.