Spatial Random Trees and the Center-Surround Algorithm

I. Pollak J.M. Siskind M.P. Harper C.A. Bouman

This work was supported in part by a National Science Foundation (NSF) CAREER award CCR-0093105, a Purdue Research Foundation grant, a Xerox Foundation grant, and NSF grants 9987576 and 9980054. Part of this work was carried out while the third author was on leave at NSF. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the view of NSF.

The authors are with the School of Electrical and Computer Engineering, Purdue University, 1285 EE Building, West Lafayette, IN 47907. Corresponding author's e-mail: ipollak@ecn.purdue.edu.

Abstract

A new class of multiscale stochastic processes called spatial random trees (SRTs) is introduced and studied. As with previous multiscale stochastic processes, SRTs model multidimensional signals using random processes on trees. Our key innovation, however, is that the tree structure itself is random and is generated by a probabilistic context-free grammar (PCFG) [26]. While PCFGs have been used to model 1-D signals, the generalization to multiple dimensions is not direct because the leaves of a tree generated by a PCFG cannot be naturally mapped to a multidimensional lattice. We solve this problem by defining a new class of PCFGs which can produce trees whose leaves are naturally arranged in a multidimensional lattice. We call such trees admissible and show that each of them generates a unique multidimensional signal. Based on this framework, procedures are developed for likelihood calculation, MAP estimation of the processes, and parameter estimation. The new framework is illustrated through simple detection problems.

Abstract

A new class of multiscale stochastic processes called spatial random trees (SRTs) is introduced and studied. As with previous multiscale stochastic processes, SRTs model multidimensional signals using random processes on trees. Our key innovation, however, is that the tree structure itself is random and is generated by a probabilistic context-free grammar (PCFG) [26]. While PCFGs have been used to model 1-D signals, the generalization to multiple dimensions is not direct because the leaves of a tree generated by a PCFG cannot be naturally mapped to a multidimensional lattice. We solve this problem by defining a new class of PCFGs which can produce trees whose leaves are naturally arranged in a multidimensional lattice. We call such trees admissible and show that each of them generates a unique multidimensional signal. Based on this framework, procedures are developed for likelihood calculation, MAP estimation of the processes, and parameter estimation. The new framework is illustrated through simple detection problems.

Index Terms

Branching processes, detection, estimation, inside-outside algorithm, context-free grammars.

I. Introduction

In this work, we develop a new class of multiscale stochastic models for multidimensional signals, called *spatial random trees (SRTs)*. Similar to [5, 6, 18], our models are stochastic processes on trees with each leaf corresponding to a single sample, for example, to a 2-D image pixel or a 3-D image voxel.[1] Our key innovation, however, is that the tree structure itself is random and is generated by a *probabilistic context-free grammar* (PCFG) [26]. While PCFGs have been used to model 1-D signals [19], the generalization to multiple dimensions is not direct because the leaves of a tree generated by a PCFG cannot be naturally mapped to a multidimensional grid. We solve this problem by defining a new class of PCFGs which can produce trees whose leaves are naturally arranged in a multidimensional grid. We call such trees admissible and show that each of them generates a unique multidimensional signal.

PCFGs have been widely used in natural-language processing, for example, to model the structure of sentences [19]. The concept of a PCFG is based on the notion of branching stochastic processes which have been used in studying population dynamics since 1845 [4, 13, 14, 28]. These problems have been posed either in 1-D where the objects under consideration, for example, words in sentences, have a natural linear arrangement; or even in "0-D" where the arrangement of objects, such as molecules of

[1] More generally, leaves may correspond to other observable quantities associated with a signal. This generalization will be considered elsewhere.

different types in a population of particles, does not matter. In addition to early efforts to apply both deterministic grammars (see, e.g., [10, 25, 27], and references therein) and probabilistic grammars [10] in 2-D, probabilistic grammars have more recently been applied to such 2-D problems as optical character recognition [23] and analyzing the layout of document images [12].

These developments have motivated the formulation of SRTs–our new general framework for modeling multidimensional signals with PCFGs. This framework is described in Section III and is the central contribution of this paper. We explain in Section VI that fixed-tree multiscale stochastic models, such as [5, 6], are a special case of our model. In general, however, instead of imposing a fixed multiscale tree structure on a signal, our model is able to adapt its tree structure to data.

For the sake of clarity, we keep most of our exposition of SRTs in this paper to 2-D; we only sketch the generalizations of our definitions and results to an arbitrary number of dimensions since these generalizations are straightforward. A key reason for the applicability of our results to any number of dimensions is that the underlying tree structure is always a binary tree, regardless of the number of dimensions.

Once our framework is in place, we obtain exact recursive algorithms for computing likelihoods and for finding the MAP estimate of the tree structure and the corresponding states, as well as adapt the Baum-Welch (or EM) algorithm [3] to the training problem, i.e., the problem of searching for the parameter values for our model that maximize the likelihood function. We collectively term these resulting algorithms the *Center-Surround algorithm*. They form the second major contribution of this paper. They are a generalization of–and were inspired by–the Forward-Backward algorithm [24] for hidden Markov models and the Inside-Outside algorithm [1, 16, 19] for 1-D PCFGs.

Even though the core of this paper is theoretical, we also include simulation results since we believe that the major impact of our framework will eventually be in the area of applications. While extensive experiments with real data are certainly beyond the scope of this paper, we do include simple synthetic examples in Section V to illustrate the Center-Surround algorithm. Research efforts are currently underway to adapt our method to several practical application domains.

The organization of our paper is as follows. The background on branching processes and PCFGs in 1-D is provided in Section II. Although this section is mostly tutorial, we believe that our approach to defining a probability distribution on the set of trees associated with a branching process is simpler than previous approaches in [9, 11, 21]. In Section III, we introduce spatial random trees. Section IV discusses inference using SRTs. Specifically, Subsection IV-A introduces exact recursive algorithms for calculating the probability of a multidimensional signal; Subsection IV-B describes an exact recursive algorithm which computes the best parse for a multidimensional signal, that is, it extracts the maximum a posteriori probability tree whose leaves are signal samples; and Subsection IV-C is devoted to training our model through the EM algorithm. Section V presents our experimental examples. Section VI places our framework in the context of the existing research and describes our current research directions.

II. Background

A. Branching Processes and PCFGs

Following [9, 15, 21], we consider a population of objects (for example, particles or people's surnames or grammatical structures) of several ***terminal types*** and ***nonterminal types***. Objects of any nonterminal type are capable of reproducing whereas terminal objects are not. We assume that both the set $\mathcal{T}$ of all possible terminal types and the set $\mathcal{N}$ of all possible nonterminal types are finite sets. We consider a discrete-time process of evolution of the population of objects, with time indexed by the set of all nonnegative integers. We suppose that at each time step one object of nonterminal type j is either transformed into two objects of nonterminal types k and ℓ with probability $\mathbf{P}_{prod}(j \to k, \ell)$, or into one object of terminal type u with probability $\mathbf{P}_{prod}(j \to u)$. In keeping with the terminology from formal language theory [26], we call each allowed transformation a ***production rule***:

$$j \quad \to \quad k, \ell \quad \forall j, k, \ell \in \mathcal{N}, \tag{1}$$

$$j \quad \to \quad u \qquad \forall j \in \mathcal{N}, \ \forall u \in \mathcal{T}. \tag{2}$$

Rules (1) are called nonterminal production rules, and rules (2) are called terminal production rules. The set of all production rules is denoted $\mathcal{P}$ and the triple $(\mathcal{N}, \mathcal{T}, \mathcal{P})$ consisting of the sets of nonterminal and terminal types and the set of production rules is an example of what is called a ***context-free grammar*** in formal language theory [19]. (We assume here that $\mathcal{P}$ is uniquely determined by $\mathcal{N}$ and $\mathcal{T}$ via Eqs. (1,2).)

We use $\mathcal{P}^j$ to denote the set of all production rules which have state j in the lefthand side. The production probabilities must satisfy the following normalization equations:

$$\sum_{\Lambda \in \mathcal{P}^j} \mathbf{P}_{prod}(\Lambda) = 1, \quad \forall j \in \mathcal{N}, \tag{3}$$

which say that any object of a nonterminal type gets transformed into something (either a pair of nonterminal objects or a single terminal object) with unit probability.

We further assume that initially, at time 0, there exists just one object whose type is $j \in \mathcal{N}$ with probability $\mathbf{P}_{root}(j)$ where this initial probability distribution $\mathbf{P}_{root}$ must also satisfy a normalization property:

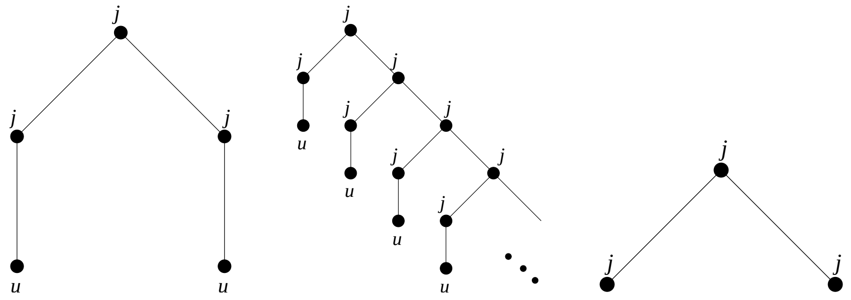
$$\sum_{j \in \mathcal{N}} \mathbf{P}_{root}(j) = 1. \tag{4}$$

This stochastic process of transformations of our objects is easily seen to be a ***branching process***, defined in, e.g., [11, 15]. (The term *multiplicative process* has also been used, e.g., in [9, 21].) The corresponding context-free grammar, equipped with probability distributions $\mathbf{P}_{root}$ and $\mathbf{P}_{prod}$, is called in natural-language processing [19] a ***probabilistic*** (or ***stochastic***) ***context-free grammar*** (PCFG or SCFG).

We consider the set of all possible genealogies, i.e., the set of all records of the transformations of the initial object (generation 0) and its descendants (generations $1, 2, \ldots$). We represent a genealogy as a (deterministic) tree, by drawing a dot with "j" next to it for every object of type j, and connecting every object to its children, that is, to the objects it gets transformed into, as in Fig. 1. We moreover observe the convention of drawing the children below their parent. In our discussion of trees, we adopt the standard terms [8] ***vertex*** which is synonymous with a dot representing one of our objects, and ***edge*** which denotes any parent-child pair. We use lowercase Greek letters to index the vertices of trees, with ρ exclusively denoting the root vertex, that is, the vertex corresponding to the original object at time 0. The type of the object corresponding to vertex α is called the ***state*** at α and is denoted by x_α. A tree is therefore a triple $(\mathcal{V}, \mathcal{E}, x)$ consisting of a set $\mathcal{V}$ of all vertices, a set $\mathcal{E}$ of all edges, and a mapping x from $\mathcal{V}$ to $\mathcal{N} \cup \mathcal{T}$ which associates a state (type) x_α to every vertex (object) α.

We conclude this subsection by describing some other terms and notational conventions pertaining to trees which will be used in the remainder of the paper. (See Table II in the Appendix for a concise summary of our notation.) We say that a tree has ***depth*** i if it consists of generations $0, 1, \ldots, i$. It is sometimes convenient to abbreviate a generic production rule as Λ (this was done in Eq. (3) above), and to denote the production rule applied at a vertex α as Λ_α. For example, referring to the tree of Fig. 1(a), Λ_ρ–meaning the production rule applied at the root vertex–is in this case simply a shorthand for $j \rightarrow j, j$.

For any vertex α that has two children, we always assume that the children are ordered, and denote the first child as $C_1(\alpha)$ and the second child as $C_2(\alpha)$. A vertex which has at least one child is called an ***internal vertex***, and a vertex with no children is called a ***leaf***. The ***yield*** of any internal vertex α, denoted $\mathcal{Y}(\alpha)$, is the set of all leaf descendants of α. Alternatively, we say that α ***dominates*** $\mathcal{Y}(\alpha)$, and write $\alpha = \mathcal{Y}^{-1}(\mathcal{Y}(\alpha))$. For a set which is not the yield of an internal vertex, the inverse yield function $\mathcal{Y}^{-1}$ is not defined. The yield of a leaf is undefined since we do not consider any vertex to be a descendant of itself. The yield of a tree is synonymous with the yield of the root of the tree.



(a) An Ω-tree of depth 2. (b) An Ω-tree of infinite depth. (c) A 1-pruned tree.

Fig. 1. (a,b) Two Ω-trees generated by applying two rules ($j \to j, j$ and $j \to u$) of the PCFG in Eqs. (1,2). (c) A 1-pruned tree generated by the same PCFG.

B. Probability Distribution on the Set of Trees

Our basic strategy for defining a probability distribution on the set of all trees is similar to [9, 11, 21]: we first define probability distributions on sets of finite-depth trees, and then extend them to the set of all trees. The set of all possible trees that can be generated by a PCFG $\mathcal{G}$ is denoted by $\Omega(\mathcal{G})$. Throughout this paper, it is always clear from context which PCFG is being discussed, and therefore we will call this set Ω for brevity. While the elements of Ω are simply trees, we will also sometimes call them ***Ω-trees***, in order to avoid confusion with ***i-pruned trees***. The process of i-pruning can be applied to an Ω-tree whose depth is strictly greater than i, and simply means retaining only generations $0, 1, \ldots, i$ of the Ω-tree and discarding the remaining generations. The resulting tree is called an i-pruned tree. The process of i-pruning can moreover be applied to any i'-pruned tree to obtain an i-pruned tree if $i' > i$. For example, the 1-pruned tree in Fig. 1(c) results if we retain only generations 0 and 1 of the Ω-tree of Fig. 1(a) or (b). The result of 0-pruning the 1-pruned tree of Fig. 1(c) would be a 0-pruned tree of depth 0 consisting of a single root vertex with state j.

The equivalence class of all Ω-trees whose i-pruning results in a given i-pruned tree T is denoted by $[T]$. If T is an Ω-tree, we denote $[T] = T$.

We use $\Pi_i(\mathcal{G})$ (or simply Π_i) to denote the set of all i-pruned trees. For example, one element of the set Π_1 for the PCFG in Eqs. (1,2) is depicted in Fig. 1(c). We use $\Phi_i(\mathcal{G})$ (or simply Φ_i) to denote the set of all Ω-trees whose depth is i. We use Ω_i to denote the set consisting of all Ω-trees of depth i or smaller, and all i-pruned trees:

$$\Omega_0 = \Pi_0; \quad \Omega_i = \bigcup_{i'=1}^{i} \Phi_{i'} \cup \Pi_i \quad \text{for } i = 1, 2, \ldots.$$

If $i' > i$, the equivalence class of all trees in $\Omega_{i'}$ whose i-pruning results in a given i-pruned tree T is denoted by $[T]_{i'}$.

We first define a probability distribution $\mathbf{P}_i$ on the set Ω_i, and then extend it to Ω. The probability of any tree $T \in \Omega_i$ is defined to be the product of the root state probability and the probabilities of all the production rules that are involved in generating T:

$$\mathbf{P}_i(T) \stackrel{\Delta}{=} \mathbf{P}_{root}(x_\rho) \prod_{\alpha \in \mathcal{V}_{int}} \mathbf{P}_{prod}(\Lambda_\alpha), \tag{5}$$

where ρ is the root vertex of T, $\mathcal{V}_{int}$ is the set of all internal vertices of T, and Λ_α is the production rule applied at α. We now need two auxiliary results which will be used throughout the remainder of the paper, and in particular, in the main propositions of this subsection. For any Ω-tree of depth i, definition (5) is consistent: $\mathbf{P}_{i'}(T) = \mathbf{P}_i(T)$ for any $i' > i$. We now show that definition (5) is moreover consistent for i-pruned trees, in the sense that the probability of any i-pruned tree T as defined by Eq. (5) is the same as the probability of the corresponding equivalence class $[T]_{i'}$, for any $i' > i$.

Lemma 1. *If $T \in \Omega_i$ and $i' > i$ then $\mathbf{P}_{i'}([T]_{i'}) = \mathbf{P}_i(T)$.*

Proof. See Appendix. ■

To show that $\mathbf{P}_i$ is indeed a probability distribution on Ω_i, we need the following lemma which says that $\mathbf{P}_{root}(j)$ is equal to the combined probability of the set of all Ω-trees of depth i or smaller whose root state is j and the set of all i-pruned trees whose root state is j.

Lemma 2. *Let Ω_i^j be the set of all trees in Ω_i whose root state is j. Then*

$$\mathbf{P}_i(\Omega_i^j) = \mathbf{P}_{root}(j). \tag{6}$$

Proof. See Appendix. ■

Proposition 1. $\mathbf{P}_i(\Omega_i) = 1$.

Proof. It follows from the definition of Ω_i^j that $\Omega_i = \bigcup_{j \in \mathcal{N}} \Omega_i^j$ and that the sets $\{\Omega_i^j\}_{j \in \mathcal{N}}$ are mutually disjoint. Therefore, by Lemma 2 and normalization equation (4), we have:

$$\mathbf{P}_i(\Omega_i) = \sum_{j \in \mathcal{N}} \mathbf{P}_i(\Omega_i^j) = \sum_{j \in \mathcal{N}} \mathbf{P}_{root}(j) = 1. \quad ■$$

Our function $\mathbf{P}_i$ is thus a probability distribution on Ω_i, for any i. We now extend it to Ω.

For any subset A of Ω_i, we let $[A]$ be the induced subset of Ω:

$$[A] = \bigcup_{T \in A} [T] \quad \text{for any } A \subset \Omega_i.$$

For any collection $\mathcal{A}$ of subsets of Ω_i, we let $[\mathcal{A}]$ be the induced collection of subsets of Ω:

$$[\mathcal{A}] = \bigcup_{A \in \mathcal{A}} \{[A]\}.$$

We let $\mathcal{A}_i$ be the power set of Ω_i, i.e., the collection of all subsets of Ω_i. The following lemma constructs an algebra of subsets of Ω and extends $\mathbf{P}_i$ to a probability measure on the algebra.

Lemma 3. *Let $\mathcal{A}_\infty = \bigcup_{i=0}^{\infty}[\mathcal{A}_i]$ be the collection of subsets of Ω induced by all the subsets of all the Ω_i's.*

(i) *Then $\mathcal{A}_\infty$ is an algebra (i.e., it is closed under set complementation and finite union).*

(ii) *For any $[A] \in \mathcal{A}_\infty$, define*

$$\widetilde{\mathbf{P}}([A]) \stackrel{\Delta}{=} \mathbf{P}_i(A),$$

where i is such that $A \in \mathcal{A}_i$. Then $\widetilde{\mathbf{P}}$ is a probability measure on $\mathcal{A}_\infty$.

Proof. See Appendix. ■

In order to get a legitimate probability measure, it only remains to extend our algebra $\mathcal{A}_\infty$ to a σ-algebra.[2] The following proposition is a direct corollary of Lemma 3 and the extension theorem (page 23 of [20]) which says that any probability measure on an algebra can be uniquely extended to the smallest σ-algebra which contains that algebra.

Proposition 2. *Let $\mathcal{A}$ be the smallest σ-algebra containing $\mathcal{A}_\infty$. There exists a unique probability measure $\mathbf{P}$ defined on $\mathcal{A}$ such that the restriction of $\mathbf{P}$ to $\mathcal{A}_\infty$ is $\widetilde{\mathbf{P}}$.* ■

This proposition says that we have defined a probability distribution $\mathbf{P}$ on Ω, with probabilities of Ω-trees and i-pruned trees given by $\mathbf{P}_i$ of Eq. (5).

III. Spatial Random Trees

A. Notation for Multidimensional Signals

While our results below generalize to an arbitrary number of dimensions, we primarily consider 2-D signals–which we call images–whose domain of definition is an $M_1 \times M_2$ rectangle, as illustrated in Fig. 2. In other words, an image $\mathbf{u}$ is just an $M_1 \times M_2$ matrix of numbers. The rectangular domain whose upper left corner is $p = (p_1, p_2)$ and whose lower right corner is $q = (q_1, q_2)$ is denoted $\square_{pq}$. If $q_1 < p_1$ or $q_2 < p_2$ then $\square_{pq}$ simply denotes an empty rectangle. For $p = (p_1, p_2)$, we write u_p and

[2] A σ-algebra is an algebra which is closed under countable union [20].

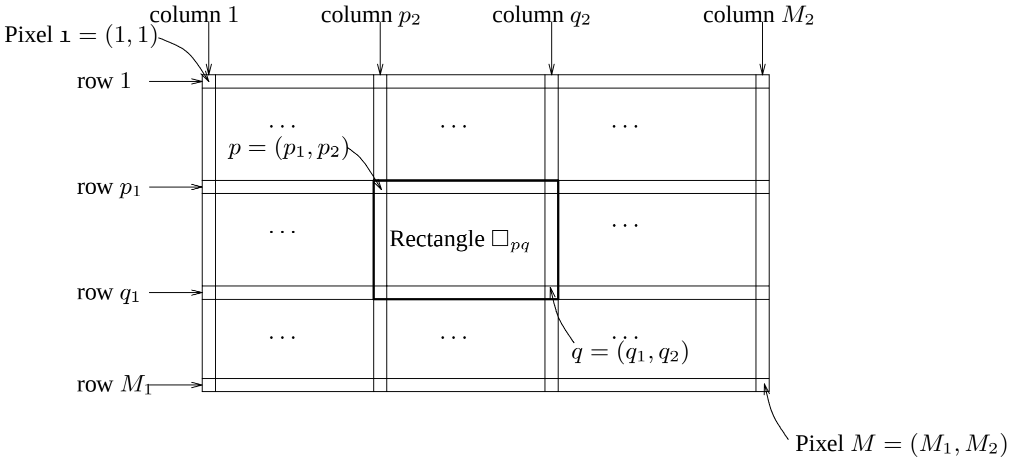


Fig. 2. An illustration of our notation for images defined on an $M_1 \times M_2$ rectangular grid.

$\Box_{pp} = \Box_p = p$ to denote the value and location, respectively, of the pixel at the intersection of row p_1 and column p_2. We abbreviate $\imath = (1, 1)$ and $M = (M_1, M_2)$, so that the whole domain of definition of image $\mathbf{u}$ is $\Box_{\imath,M}$.

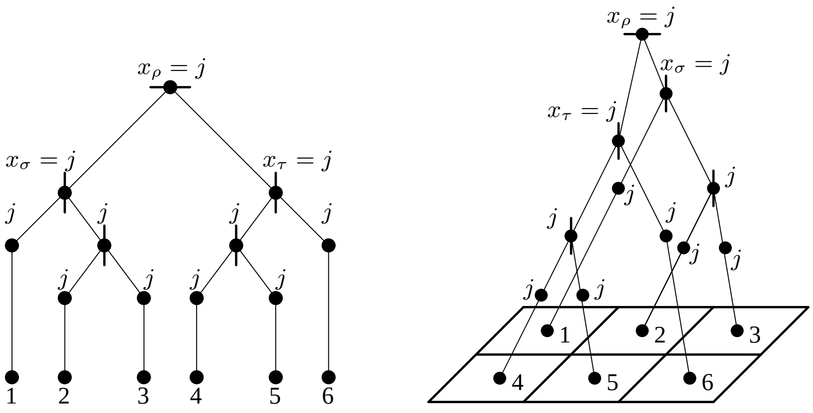
For an arbitrary number of dimensions D, our signal is a D-dimensional $M_1 \times \ldots \times M_D$ matrix, and the domain $\Box_{pq}$ is defined by its two corner points $p = (p_1, \ldots, p_D)$ and $q = (q_1, \ldots, q_D)$.

B. SRT: Definition

Finite Ω-trees illustrated in Fig. 1(a) can be used to model linearly arranged objects such as words which form sentences, each word being the state of a leaf. The set of all leaves $\mathcal{Y}(\rho)$ has a natural arrangement, from left to right. The yield of each internal vertex of such a tree is a contiguous segment in $\mathcal{Y}(\rho)$; moreover, if β and γ are the children of α then $\mathcal{Y}(\beta)$ and $\mathcal{Y}(\gamma)$ are nonoverlapping segments whose union is $\mathcal{Y}(\alpha)$.

In order to model images, the yield of an Ω-tree must be capable of representing pixels arranged in an $M_1 \times M_2$ rectangular grid. In our framework, the yield $\mathcal{Y}(\alpha)$ of each internal vertex α of the resulting tree is a nonempty rectangular region of an image, with each leaf representing a 1×1 region, i.e., a single pixel location. If β and γ are the children of α then $\mathcal{Y}(\beta)$ and $\mathcal{Y}(\gamma)$ are nonoverlapping rectangular regions whose union is $\mathcal{Y}(\alpha)$.

A key innovation that makes this possible is to have two different kinds of nonterminal production rules: horizontal, to represent top-bottom splits; and vertical, to represent left-right splits. Specifically, an application of a horizontal nonterminal production rule splits the rectangular region $\mathcal{Y}(\alpha)$ into a top subrectangle $\mathcal{Y}(\beta)$ and a bottom subrectangle $\mathcal{Y}(\gamma)$ where $\beta = C_1(\alpha)$ and $\gamma = C_2(\alpha)$. An application



(a) An Ω-tree.

(b) The same tree with the corresponding image.

Fig. 3. (a) An Ω-tree generated by our image PCFG, and (b) the same tree superimposed on the corresponding image. A short horizontal (vertical) line through a vertex signifies a horizontal (vertical) split at that vertex.

of a vertical nonterminal production rule splits the rectangular region $\mathcal{Y}(\alpha)$ into a left subrectangle $\mathcal{Y}(\beta)$ and a right subrectangle $\mathcal{Y}(\gamma)$ where, again, $\beta = C_1(\alpha)$ and $\gamma = C_2(\alpha)$. We use $\mathcal{O}$ to denote the set of possible orientations of a nonterminal production rule: $\mathcal{O} = \{h, v\}$. Our basic 1-D PCFG of Eqs. (1,2) is modified as follows:

$$j \xrightarrow{o} k,\ell \quad \forall j,k,\ell \in \mathcal{N},\ \forall o \in \mathcal{O}, \tag{7}$$

$$j \rightarrow u \qquad \forall j \in \mathcal{N},\ \forall u \in \mathcal{T}. \tag{8}$$

Our definition of a tree $T = (\mathcal{V}, \mathcal{E}, x, o)$ now includes a function o from the vertex set to the set $\mathcal{O}$ of orientations. For any vertex α with two children, o_α is the orientation of the production rule applied at α.

The generalization to D dimensions is straightforward: the set $\mathcal{O}$ then has D elements, $\mathcal{O} = \{i_1, \ldots, i_D\}$, and there are D corresponding varieties of nonterminal production rules: $j \overset{i_d}{\rightarrow} k, \ell$ for $d = 1, \ldots, D$. It is important to emphasize here that, regardless of the number of dimensions, our basic modeling structure is a binary tree, i.e., a tree where each vertex has at most two children, due to the fact that the righthand side of a nonterminal production rule always has two elements. This property is in contrast with [5,6,18] where images are modeled with quadtrees.

C. Generating Images from the PCFG of Eqs. (7,8)

We have not yet specified precisely what it means for the yield of an Ω-tree produced by our PCFG to form an $M_1 \times M_2$ rectangular grid. Indeed, each production rule only specifies the *orientation* of a split of a rectangular domain (horizontal with $o = h$ in Eq. (7) or vertical with $o = v$ in Eq. (7)), but does not specify exactly *where* the split will occur. It is moreover undesirable to fix the sizes of the rectangular regions a priori–in other words, we do not want to require the state of an internal vertex of an Ω-tree to contain information about the size of its yield. Otherwise, our model would become unmanageable since there would be an enormous number of possible states and production rules even for images of moderate size. At first glance, it may therefore seem that our PCFG is too ill-defined to unambiguously generate regularly sampled images. We now show that this is not the case. We start with an example.

Example 1. We let $\mathcal{T} = \{1, 2, 3, 4, 5, 6\}$ and construct an Ω-tree by letting the root state be j and applying the following sequence of production rules:

$j \xrightarrow{h} j, j$ at the root vertex ρ;

$j \xrightarrow{v} j, j$ both at $\sigma = C_1(\rho)$ and at $\tau = C_2(\rho)$;

$j \to 1$ at $C_1(\sigma)$; $j \xrightarrow{v} j, j$ both at $C_2(\sigma)$ and at $C_1(\tau)$; $j \to 6$ at $C_2(\tau)$;

$j \to u$ for $u = 2, 3, 4, 5$, at the remaining internal vertices.

This results in the tree depicted in Fig. 3(a). As shown in Fig. 3(b), this tree generates a 2×3 image. The yield of each internal vertex α of the tree is a nonempty rectangular region $\mathcal{Y}(\alpha)$. The root corresponds to the whole image domain–i.e., $\mathcal{Y}(\rho)$ is the whole 2×3 domain of definition of our image. Moreover, each application of a nonterminal production rule involves splitting a rectangular region $\mathcal{Y}(\alpha)$ into two other nonempty rectangular regions, $\mathcal{Y}(C_1(\alpha))$ and $\mathcal{Y}(C_2(\alpha))$. The orientation of the splitting is either horizontal or vertical, and is specified by the orientation of the nonterminal production rule.

A horizontal split at a vertex α is denoted in our diagrams with a horizontal line through the vertex. For example, the 2×3 image domain corresponding to the root vertex ρ in Fig. 3(b) is split horizontally, to result in the first child $\sigma = C_1(\rho)$ whose yield is the top 1×3 subrectangle, and the second child $\tau = C_2(\rho)$ whose yield is the bottom 1×3 subrectangle.

A vertical split at a vertex α is denoted in our diagrams with a vertical line through the vertex. For example, the 1×3 rectangle which is the yield of the first child σ of the root vertex in Fig. 3(b) is split vertically, to result in the first child $C_1(\sigma)$ whose yield is the top left pixel, and the second child $C_2(\sigma)$ whose yield is the rectangular region with pixel values 2,3. ■

It is not always the case that the yield of an Ω-tree can be bijectively mapped to an $M_1 \times M_2$ rectangular grid. For example, the tree of Fig. 4(a) does not correspond to such a grid, as illustrated in Fig. 4(b). Regardless of the locations of splits, the first row will have two “pixels” whereas the second and third row will have only one “pixel” each. We call such Ω-trees *inadmissible trees*. The trees for which there exists

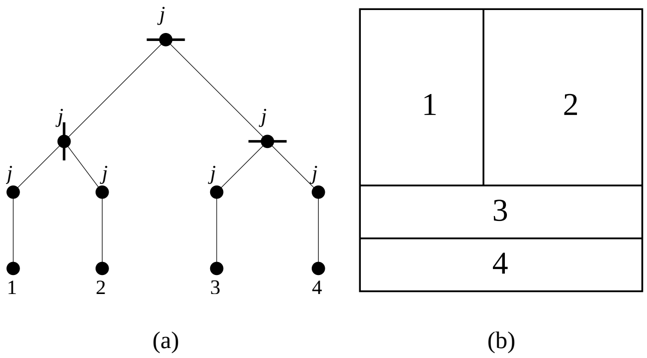


Fig. 4. (a) An Ω-tree that does not correspond to an $M_1 \times M_2$ rectangular grid. (b) Regardless of the split locations, the first row will have two "pixels" whereas the second and third rows will have only one "pixel" each.

a bijective association between the leaves and pixel locations on a rectangular grid, are called *admissible*.

Definition 1 (Admissible trees). *Let T be an Ω-tree generated by the PCFG of Eqs. (7,8). Let $\mathcal{V}_{int}$ be the set of its internal vertices, and let ρ be its root vertex. Suppose that there exist a pair of positive integers $M = (M_1, M_2)$ and a bijective function*

$$\mathfrak{F} : \mathcal{Y}(\rho) \to \square_{\mathbf{1},M}$$

which uniquely maps each leaf of the tree to a location in an $M_1 \times M_2$ grid, and which has the following property: the yield of each internal vertex of the tree is mapped to a rectangular region, i.e.,

$$\forall \alpha \in \mathcal{V}_{int} \quad \exists p, q \text{ such that } \{\mathfrak{F}(\beta) | \beta \in \mathcal{Y}(\alpha)\} = \square_{pq}. \tag{9}$$

We then say that T is an ***admissible tree*** *and $\mathfrak{F}$ is an associated* ***admissibility function****.* ■

We now show that, if the yield of an Ω-tree can be mapped to an image grid in a manner described above and illustrated in Fig. 3, such mapping is unique.

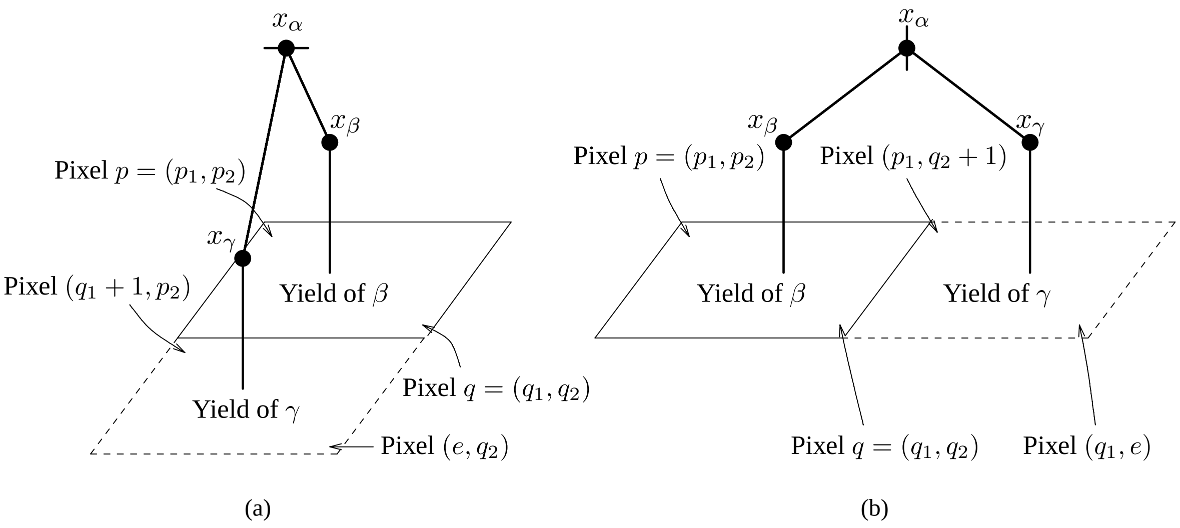
Theorem 1 (Admissibility Theorem). *If a tree T is admissible, there is a unique admissibility function for T.*

Proof. We denote the leaves of T, ordered left to right, by $\lambda_1, \dots, \lambda_I$. Our proof is a recursive procedure which constructs $\mathfrak{F}(\lambda_1), \dots, \mathfrak{F}(\lambda_I)$ from left to right. If the procedure fails, the tree is inadmissible; we show, however, that if the procedure succeeds, then it constructs a unique $\mathfrak{F}$.

For any internal vertex α, we use the following notation:

$$\mathfrak{F}(\alpha) = \{\mathfrak{F}(\beta) | \beta \in \mathcal{Y}(\alpha)\},$$

meaning that $\mathfrak{F}(\alpha)$ is the rectangular region (9) into which the yield of α is mapped by the function $\mathfrak{F}$.



(a)

(b)

Fig. 5. Illustrations for the proof of the Admissibility Theorem.

Our proof is inductive; both the base of the induction and the induction step are obtained from the following observation which immediately follows from our definition of nonterminal production rules: the leftmost leaf of the yield of an internal vertex γ must be the upper left corner of $\mathfrak{F}(\gamma)$. In other words, suppose that vertex γ has yield $\lambda_{left}, \ldots, \lambda_{right}$ and that $\mathfrak{F}(\gamma) = \Box_{pq}$. Then $\mathfrak{F}(\lambda_{left}) = \Box_p$.

This observation implies that $\mathfrak{F}(\lambda_1) = \Box_1$, which is the base of our induction.

Suppose now that $\mathfrak{F}(\lambda_1), \ldots, \mathfrak{F}(\lambda_i)$ have been uniquely determined. We now show that either $\mathfrak{F}(\lambda_{i+1})$ is undefined (in which case the tree T is inadmissible), or it can be determined uniquely.

Suppose that α is the youngest common ancestor of λ_i and λ_{i+1}, that is, α is an ancestor of both λ_i and λ_{i+1} and none of the children of α is an ancestor of both λ_i and λ_{i+1}. Let $\beta = C_1(\alpha)$ and $\gamma = C_2(\alpha)$ be the first and second child of α, respectively. By construction, the rightmost vertex of the yield of β is λ_i, and so the yield of β is a subset of $\{\lambda_1, \ldots, \lambda_i\}$. Thus $\mathfrak{F}(\beta)$ has been uniquely determined. Let us denote $\mathfrak{F}(\beta) = \Box_{pq}$. Furthermore, let m be the number of vertices in the yield of γ.

As shown in Fig. 5(a), if $x_\alpha \xrightarrow{h} x_\beta, x_\gamma$, then, by our definition of a horizontal nonterminal production rule, there must exist an integer e such that

$$\mathfrak{F}(\gamma) = \Box_{(q_1+1, p_2), (e, q_2)}, \tag{10}$$

i.e., the yield of γ must correspond to a rectangle with $q_2 - p_2 + 1$ columns. Therefore, if the number m of leaves in the yield of γ is not divisible by $q_2 - p_2 + 1$, we immediately have that the tree T is inadmissible. If, however, m is divisible by $q_2 - p_2 + 1$, we know that the leftmost leaf in the yield of γ

must correspond to the upper left corner of $\square_{(q_1+1,p_2),(e,q_2)}$:

$$\mathfrak{F}(\lambda_{i+1}) = \square_{(q_1+1,p_2)}.$$

Similarly, if $x_\alpha \overset{v}{\to} x_\beta, x_\gamma$, then, by our definition of a vertical nonterminal production rule, there must exist an integer e such that

$$\mathfrak{F}(\gamma) = \square_{(p_1,q_2+1),(q_1,e)}. \tag{11}$$

This is illustrated in Fig. 5(b). In this case, if m is not divisible by q_1-p_1+1, then the tree is inadmissible; otherwise,

$$\mathfrak{F}(\lambda_{i+1}) = \square_{(p_1,q_2+1)}.$$

This completes our induction. ■

We note that while the Admissibility Theorem is stated and proved in 2-D, it also holds for an arbitrary number of dimensions. This identification of the leaves of an admissible tree with pixel locations in an $M_1 \times M_2$ (or, more generally, $M_1 \times \ldots \times M_D$) rectangular grid justifies the terminology we used in Example 1, allowing us to identify $\mathcal{Y}(\alpha)$ with $\square_{pq}$ for any internal vertex α of an admissible tree with $\mathfrak{F}(\alpha) = \square_{pq}$.

D. Probability Model

Since our PCFG of Eqs. (7,8) is similar to Eqs. (1,2), we can define a probability distribution on the set of all Ω-trees generated by this PCFG, via a procedure similar to that of Subsection II-B.

Suppose now that we have an $M_1 \times M_2$ image $\mathbf{u} = \mathbf{u}_{\mathbf{1},M}$ and an admissible tree T. If the yield of T is $\square_{\mathbf{1},M}$ and the states of the leaves are $\mathbf{u}_{\mathbf{1},M}$, we say that the tree T ***generates*** the image $\mathbf{u}$. We define an event[3] $\Omega_{\mathbf{u}}$ to be the set of all admissible trees that generate the image $\mathbf{u}$. The term ***probability of an image*** $\mathbf{u}$, denoted $\mathbf{P}(\mathbf{u})$, is shorthand for the probability of the set $\Omega_{\mathbf{u}}$. Note that $\mathbf{P}(\mathbf{u})$ does not, in general, define a probability distribution on the set of all images since the set of all admissible trees does not, in general, have unit probability. A probability distribution $\mathbf{P}'$ on the set of images can be obtained by conditioning on the set of admissible trees:

$$\mathbf{P}'(\mathbf{u}) = \frac{\mathbf{P}(\mathbf{u})}{\sum\limits_{\mathbf{u}'} \mathbf{P}(\mathbf{u}')}. \tag{12}$$

The probability $\mathbf{P}'(\mathbf{u})$ is, in general, very difficult to compute because of the denominator in the expression (12). Fortunately, we will not need to compute it in any of the algorithms developed below.

[3] We emphasize here that our notation $\Omega_{\mathbf{u}}$ has no relation to the notation Ω_i which we used in Section II.

Definition 2. *The stochastic process defined by the PCFG $\mathcal{G}$ of Eqs. (7,8), with a probability distribution on $\Omega(\mathcal{G})$ as defined above, and with the ensuing probabilities of images, is called a* ***spatial random tree*** *(SRT).*

IV. SRTs and Inference: The Center-Surround Algorithm

Within our framework of spatial random trees, we pose three basic problems as suggested by [24]:

- **Likelihood computation.** Given a PCFG $\mathcal{G}$, find the likelihood $\mathbf{P}(\mathbf{u}|\mathcal{G})$ of the data $\mathbf{u}$.
- **MAP estimation of a tree.** Find the most probable tree for the data $\mathbf{u}$,
$$\arg\max_{T\in\Omega} \mathbf{P}(T|\Omega_{\mathbf{u}}).$$
- **ML parameter estimation.** Choose a PCFG $\mathcal{G}$ that maximizes the likelihood[4] of an observation $\mathbf{u}$ or a sequence of observations $\mathbf{u}^1,\ldots,\mathbf{u}^I$:
$$\arg\max_{\mathcal{G}} \mathbf{P}(\mathbf{u}^1,\ldots,\mathbf{u}^I|\mathcal{G}).$$

We address these three problems in subsections IV-A, IV-B, and IV-C, respectively. The algorithms that we develop are collectively termed the ***Center-Surround Algorithm*** and are inspired by the Forward-Backward algorithm [24] and the Inside-Outside algorithm [1,16,19]. The Center-Surround algorithm is based on recursive calculations involving *center* and *surround* variables which we presently define.

For any vertex δ of any admissible tree T, let T_δ be the subtree of T rooted at δ, and let $\overline{T}_\delta$ be the portion of T that is obtained by removing all descendants of δ (however, δ itself is a part of $\overline{T}_\delta$). If $T\in\Omega_{\mathbf{u}}$, $x_\delta = j$, and $\mathcal{Y}(\delta)=\square_{pq}$, then we let $c^j_{pq}(\mathbf{u},T,\delta)$ be the product of probabilities of all the production rules involved in constructing T_δ; otherwise, $c^j_{pq}(\mathbf{u},T,\delta)=0$:

$$c^j_{pq}(\mathbf{u},T,\delta) \triangleq \begin{cases} \displaystyle\prod_{\alpha\in\mathcal{V}_{\delta,int}} \mathbf{P}_{prod}(\Lambda_\alpha), & \text{if } \mathcal{Y}(\delta)=\square_{pq} \text{ and } x_\delta = j, \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where $\mathcal{V}_{\delta,int}$ is the set of all internal vertices of T_δ. For a fixed $\overline{T}_\delta$, we define the ***center variable*** $c^j_{pq}(\mathbf{u})$ as follows:

$$c^j_{pq}(\mathbf{u}) \triangleq \sum_{T\,:\,T\in\Omega_{\mathbf{u}} \text{ and } \overline{T}_\delta \text{ is fixed}} c^j_{pq}(\mathbf{u},T,\delta). \tag{14}$$

[4] Note that our definition of maximum likelihood estimation is distinct from the maximization of $\mathbf{P}'(\mathbf{u}^1,\ldots,\mathbf{u}^I|\mathcal{G})$ defined in Eq. (12).

In other words, we keep $\overline{T}_\delta$ fixed and sum over all possible subtrees T_δ with root state $x_\delta = j$. Note that $c^j_{pq}(\mathbf{u})$ is in fact the same for any $\overline{T}_\delta$ with $x_\delta = j$ because every term $c^j_{pq}(\mathbf{u}, T, \delta)$ in the summation only depends on T_δ.

It follows from these definitions (13,14) as well as from Eq. (5) and Lemma 2, that $c^j_{1,M}(\mathbf{u}, T, \rho)$ is simply the conditional probability of T given that its root state is j:

$$c^j_{1,M}(\mathbf{u}, T, \rho) = \mathbf{P}(T|\Omega^j), \quad \forall T \in \Omega_{\mathbf{u}},$$

where Ω^j is the set of all trees whose root state is j. Summing over all $T \in \Omega_{\mathbf{u}}$, we obtain an expression for the conditional probability of the image $\mathbf{u}$ in terms of a center variable:

$$c^j_{1,M}(\mathbf{u}) = \mathbf{P}(\mathbf{u}|\Omega^j). \tag{15}$$

It is easily seen that all other center variables are also conditional probabilities of events in Ω.

If $T \in \Omega_{\mathbf{u}}$, $x_\delta = j$, and $\mathcal{Y}(\delta) = \square_{pq}$, then we let $s^j_{pq}(\mathbf{u}, T, \delta)$ be the product of the probability of the root state of T and the probabilities of all the production rules which are *not* involved in constructing T_δ; otherwise, $s^j_{pq}(\mathbf{u}, T, \delta) = 0$:

$$s^j_{pq}(\mathbf{u}, T, \delta) \triangleq \begin{cases} \mathbf{P}_{root}(x_\rho) \displaystyle\prod_{\alpha \in \overline{\mathcal{V}}_{\delta,int}} \mathbf{P}_{prod}(\Lambda_\alpha) & \text{if } \mathcal{Y}(\delta) = \square_{pq} \text{ and } x_\delta = j, \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

where $\overline{\mathcal{V}}_{\delta,int}$ is the set of all internal vertices of T that do not belong to T_δ. For a fixed T_δ, we define the ***surround variable*** $s^j_{pq}(\mathbf{u})$ as follows:

$$s^j_{pq}(\mathbf{u}) \triangleq \sum_{T\,:\,T \in \Omega_{\mathbf{u}} \text{ and } T_\delta \text{ is fixed}} s^j_{pq}(\mathbf{u}, T, \delta). \tag{17}$$

An important consequence of Eqs. (13) and (16) is that, for any tree $T \in \Omega_{\mathbf{u}}$ having a vertex δ with state j which dominates $\square_{pq}$,

$$\mathbf{P}(T) = s^j_{pq}(\mathbf{u}, T, \delta) c^j_{pq}(\mathbf{u}, T, \delta).$$

We just mention here without proof that it is also possible to interpret all the surround variables as probabilities of events in Ω.

Now that we have defined the center and surround variables, we proceed to describe our three inference algorithms in the next three subsections. Recall that our notation for SRTs, deterministic trees, and images is summarized in Table II in Appendix.

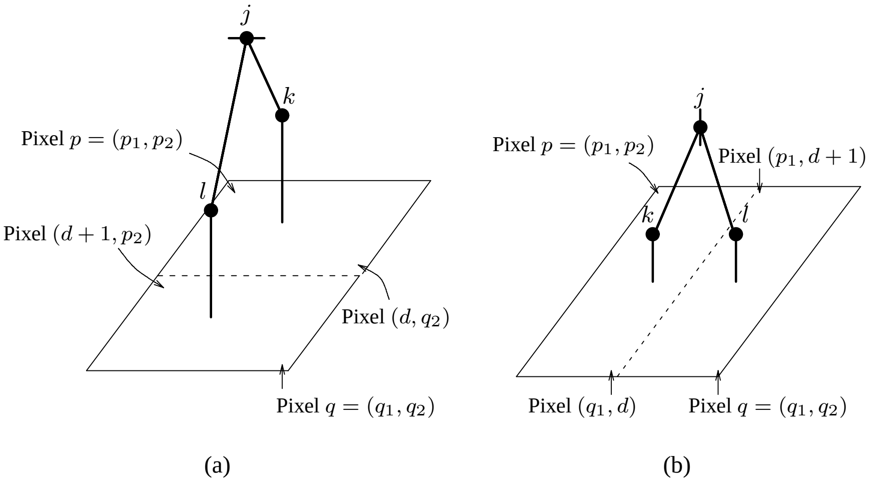


Fig. 6. Illustration of the recursive computation of the center variables: (a) horizontal split; (b) vertical split.

A. Likelihood Computation

1) Recursive Computation of the Center Variables: Eq. (15) tells us that, given a fixed PCFG, the probability of the image $\mathbf{u}$ can be easily computed if the center variables $c^j_{1,M}(\mathbf{u})$ are known for all possible root states $j \in \mathcal{N}$:

$$\mathbf{P}(\mathbf{u}) = \sum_{j\in\mathcal{N}} \mathbf{P}(\mathbf{u}|\Omega^j)\mathbf{P}(\Omega^j) = \sum_{j\in\mathcal{N}} c^j_{1,M}(\mathbf{u})\mathbf{P}_{root}(j). \tag{18}$$

Our definition of the center variables suggests that it should be possible to express $c^j_{1,M}(\mathbf{u})$ in terms of center variables defined on rectangular subdomains of $\Box_{1,M}$. The following proposition, illustrated in Fig. 6, shows that this is indeed the case: $c^j_{1,M}(\mathbf{u})$ can be computed recursively. The first term of the recursion formula (19) corresponds to summing over all possible horizontal splittings (Fig. 6(a)) and the second term corresponds to the vertical splittings (Fig. 6(b)).

Proposition 3. *For any nonempty rectangular domain $\Box_{pq} \subset \Box_{1,M}$ with $p \neq q$, and any $j \in \mathcal{N}$,*

$$\begin{aligned} c^j_{pq}(\mathbf{u}) &= \sum_{d=p_1}^{q_1-1} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} \mathbf{P}_{prod}(j \xrightarrow{h} k, \ell) c^k_{p,(d,q_2)}(\mathbf{u}) c^\ell_{(d+1,p_2),q}(\mathbf{u}) \\ &+ \sum_{d=p_2}^{q_2-1} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} \mathbf{P}_{prod}(j \xrightarrow{v} k, \ell) c^k_{p,(q_1,d)}(\mathbf{u}) c^\ell_{(p_1,d+1),q}(\mathbf{u}), \end{aligned} \tag{19}$$

where our convention is that any sum over an empty set is zero. For any $p \in \Box_{1,M}$ and any $j \in \mathcal{N}$,

$$c^j_{pp}(\mathbf{u}) = \mathbf{P}_{prod}(j \to u). \tag{20}$$

Proof. See Appendix. ■

Combining Proposition 3 with Eq. (18) gives a recursive algorithm for calculating the probability $\mathbf{P}(\mathbf{u})$ of an image $\mathbf{u}$. This algorithm computes the center variables for each nonempty subrectangle $\Box_{pq}$ (with $p \neq q$) of the rectangle $\Box_{\mathbf{1},M}$. A simple calculation shows that the total number of such subrectangles is $M_1 M_2 (M_1 + 1)(M_2 + 1)/4 = O(M_1^2 M_2^2)$. For each such subrectangle, each nonterminal production rule contributes $O(M_1 + M_2)$ computations via the sums in Eq. (19). Since there are altogether $2|\mathcal{N}|^3$ nonterminal production rules, where $|\mathcal{N}|$ is the number of nonterminal states, the overall time complexity is $O(M_1^2 M_2^2 (M_1 + M_2)|\mathcal{N}|^3)$.

It has been our experience, however, that useful probabilistic grammars typically have many nonterminal production rules with zero probabilities. For the simple examples considered in Section V, the number of nonterminal rules with nonzero probabilities is $O(|\mathcal{N}|)$, as illustrated in Table I. This can be exploited[5] to reduce the time complexity of the algorithm to $O(M_1^2 M_2^2 (M_1 + M_2)|\mathcal{N}|)$. We are currently investigating methods for further reducing the running time by restricting the set of possible trees, e.g., using methods discussed in Section VI.

We now consider the space complexity of our recursive algorithm–i.e., the amount of memory it requires. During the computation of the center variables $c^j_{\mathbf{1},M}(\mathbf{u})$ via a recursive call to Eq. (19), we need to simultaneously store $c^j_{p,q}(\mathbf{u})$ for each subrectangle $\Box_{p,q}$ of $\Box_{\mathbf{1},M}$, and for each $j \in \mathcal{N}$. Therefore, the space complexity is $O(M_1^2 M_2^2 |\mathcal{N}|)$. The sparseness of the probabilistic grammar (i.e. the fact that many production rules have zero probability) can in some cases cause many center variables to be zero. In this case, the overall memory complexity can be drastically reduced by using memory-efficient techniques such as hash tables. Restricting the set of possible trees may lead to further reductions in memory complexity.

We note that recursion (19) is easily modified for the case of D-dimensional data: the two triple summations are replaced with D triple summations, one for each split orientation. The time complexity is then $O(M_1^2 \cdot \ldots \cdot M_D^2 (M_1 + \ldots + M_D) D |\mathcal{N}|^3)$ which is $O(|\Box_{\mathbf{1},M}|^{2+\frac{1}{D}} D |\mathcal{N}|^3)$ for signal domains which are D-dimensional hypercubes, where we denote the total number of samples in the multidimensional signal by $|\Box_{\mathbf{1},M}|$. The space complexity is still $O(|\Box_{\mathbf{1},M}|^2 |\mathcal{N}|)$.

We conclude our discussion of complexity by commenting on the non-asymptotic case when D is small. This case is of particular interest since it corresponds to a variety of data such as 1-D signals ($D = 1$), planar images ($D = 2$), volumetric images and video ($D = 3$). Observe that increasing the dimensionality of the problem will, for an appropriately large M, reduce the time complexity in this case.

[5] By not performing any calculations involving zero-probability rules.

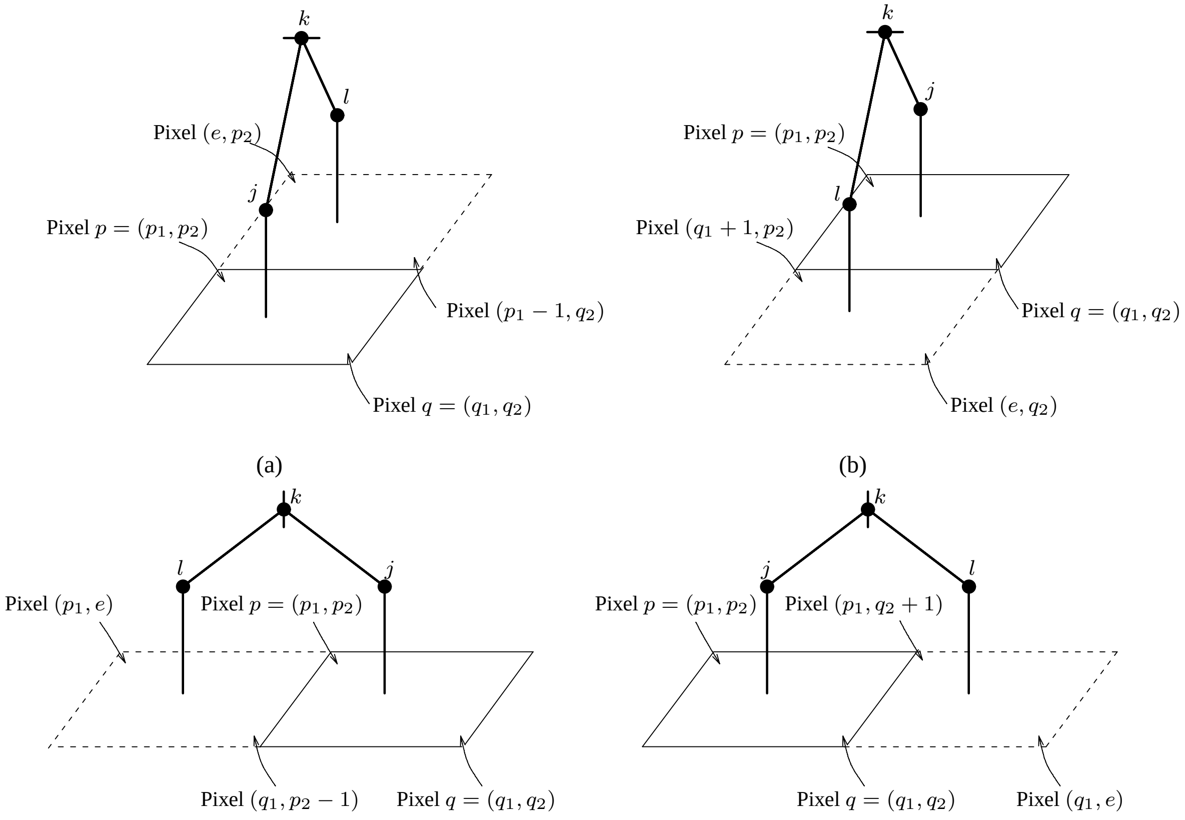


Fig. 7. Illustration of the surround recursions: (a) and (b) horizontal splits; (c) and (d) vertical splits.

This is quite atypical for algorithms that handle multidimensional data.

2) Recursive Computation of the Surround Variables: The likelihood can also be computed from the surround variables. The surround recursion uses the center variables which therefore must be precomputed. The surround recursion (22) is illustrated in Fig. 7. The four terms in Eq. (22) correspond to two different cases for a horizontal split of the parent rectangle and two cases for a vertical split.

Proposition 4. *For any $p \in \square_{\mathbf{1},M}$,*

$$\mathbf{P}(\mathbf{u}) = \sum_{j \in \mathcal{N}} s^j_{pp}(\mathbf{u}) c^j_{pp}(\mathbf{u}). \tag{21}$$

For any nonempty rectangular domain $\square_{pq} \subset \square_{\mathbf{1},M}$ with $p \neq q$, and any $j \in \mathcal{N}$,

$$\begin{aligned} s^j_{pq}(\mathbf{u}) \quad &= \quad \sum_{e=1}^{p_1-1} \sum_{k \in \mathcal{N}} \sum_{\ell \in \mathcal{N}} s^k_{(e,p_2),q}(\mathbf{u}) \mathbf{P}_{prod}(k \xrightarrow{h} \ell, j) c^\ell_{(e,p_2),(p_1-1,q_2)}(\mathbf{u}) \\ &+ \quad \sum_{e=q_1+1}^{M_1} \sum_{k \in \mathcal{N}} \sum_{\ell \in \mathcal{N}} s^k_{p,(e,q_2)}(\mathbf{u}) \mathbf{P}_{prod}(k \xrightarrow{h} j, \ell) c^\ell_{(q_1+1,p_2),(e,q_2)}(\mathbf{u}) \end{aligned}$$

$$+ \sum_{e=1}^{p_2-1} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} s^k_{(p_1,e),q}(\mathbf{u})\mathbf{P}_{prod}(k \xrightarrow{v} \ell, j) c^{\ell}_{(p_1,e),(q_1,p_2-1)}(\mathbf{u})$$

$$+ \sum_{e=q_2+1}^{M_2} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} s^k_{p,(q_1,e)}(\mathbf{u})\mathbf{P}_{prod}(k \xrightarrow{v} j, \ell) c^{\ell}_{(p_1,q_2+1),(q_1,e)}(\mathbf{u}), \qquad (22)$$

where our convention is that any sum over an empty set is zero. The base case for this recursion is:

$$s^j_{\mathbf{1},M}(\mathbf{u}) = \mathbf{P}_{root}(j). \qquad (23)$$

Proof is similar to the proof of Proposition 3 which is given in the Appendix. ■

The computational complexity of the surround recursions is similar to that of the center recursions. The latter was discussed above.

B. *MAP Tree Estimation*

In this section, we present an algorithm for extracting the most probable tree $\widehat{T}$ from $\Omega_{\mathbf{u}}$, for a given image $\mathbf{u}$. The probability of the most probable tree in $\Omega_{\mathbf{u}}$ with root state j is denoted $g^j_{pq}(\mathbf{u})$. The recursive formulas are a simple variant of the center recursion of the previous subsection (see Proposition 3 and Fig. 6), with "$\sum$" replaced by "$\max$". We therefore state them without proof. The base case is:

$$g^j_{pp}(\mathbf{u}) = \mathbf{P}_{prod}(j \to u).$$

We recursively calculate $g^j_{pq}(\mathbf{u})$ for any rectangle in terms of probabilities associated with smaller rectangles:

$$\begin{aligned}
g^{j;h}_{pq}(\mathbf{u}) &= \max_{d\in\{p_1,\ldots,q_1-1\},k\in\mathcal{N},\ell\in\mathcal{N}} \mathbf{P}_{prod}(j \xrightarrow{h} k, \ell)\, g^k_{p,(d,q_2)}(\mathbf{u})\, g^{\ell}_{(d+1,p_2),q}(\mathbf{u}), \\
g^{j;v}_{pq}(\mathbf{u}) &= \max_{d\in\{p_2,\ldots,q_2-1\},k\in\mathcal{N},\ell\in\mathcal{N}} \mathbf{P}_{prod}(j \xrightarrow{v} k, \ell)\, g^k_{p,(q_1,d)}(\mathbf{u})\, g^{\ell}_{(p_1,d+1),q}(\mathbf{u}), \\
g^j_{pq}(\mathbf{u}) &= \max(g^{j;v}_{pq}(\mathbf{u}), g^{j;h}_{pq}(\mathbf{u})).
\end{aligned}$$

We in addition store the four-tuple of parameters (o, k, l, d) which have led to the maximal g^j_{pq} where $o \in \{h, v\}$ stands for the split orientation. We call this four-tuple $f^j_{pq}(\mathbf{u})$. If $g^{j;h}_{pq}(\mathbf{u}) > g^{j;v}_{pq}(\mathbf{u})$, then

$$f^j_{pq}(\mathbf{u}) = (h,\ \arg\max_{(k,\ell,d)} \mathbf{P}_{prod}(j \xrightarrow{h} k, \ell)\, g^k_{p,(d,q_2)}(\mathbf{u})\, g^{\ell}_{(d+1,p_2),q}(\mathbf{u})).$$

Otherwise,

$$f^j_{pq}(\mathbf{u}) = (v,\ \arg\max_{(k,\ell,d)} \mathbf{P}_{prod}(j \xrightarrow{v} k, \ell)\, g^k_{p,(q_1,d)}(\mathbf{u})\, g^{\ell}_{(p_1,d+1),q}(\mathbf{u})).$$

If the maximum is not unique, we can choose an arbitrary maximizing four-tuple. The probability of the MAP tree $\widehat{T}$ is calculated from the g variables,

$$\mathbf{P}(\widehat{T}) = \max_{j\in\mathcal{N}} g^j_{\mathbf{1},M}(\mathbf{u})\mathbf{P}_{root}(j),$$

and the MAP tree itself is constructed from the list of the f variables.

Our discussion above of the computational complexity of the center recursions also applies here, as well as the various possibilities for reducing the computational complexity.

C. Parameter Estimation

Given a set $\mathbf{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^I\}$ of I independent observations, we use the expectation maximization (EM) algorithm [3] to estimate the model parameters. In this section, we denote the likelihood by $\mathbf{P}(\mathbf{U}|\mathcal{G})$, explicitly indicating its dependence on the PCFG $\mathcal{G}$. Starting with any initial PCFG $\mathcal{G}^0$, the EM algorithm generates a sequence of PCFGs $\mathcal{G}^0, \mathcal{G}^1, \mathcal{G}^2, \ldots$ which is guaranteed to climb the likelihood surface and therefore allows one to improve the PCFG parameters, starting with any initial parameter values. This result is based on the following simple proposition which we adapt from [3].

Proposition 5. *Let*

$$Q(\mathcal{G}^{n+1}, \mathcal{G}^n) \triangleq \sum_{i=1}^{I} \sum_{T\in\Omega_{\mathbf{u}^i}} \mathbf{P}(T|\mathbf{u}^i, \mathcal{G}^n) \log \mathbf{P}(T|\mathcal{G}^{n+1}). \tag{24}$$

If $Q(\mathcal{G}^{n+1}, \mathcal{G}^n) > Q(\mathcal{G}^n, \mathcal{G}^n)$ *then* $\mathbf{P}(\mathbf{U}|\mathcal{G}^{n+1}) > \mathbf{P}(\mathbf{U}|\mathcal{G}^n)$.

Proof. This proof is adapted from [3]. We use the independence of the individual observations $\mathbf{u}^i$:

$$\begin{aligned}
\log \frac{\mathbf{P}(\mathbf{U}|\mathcal{G}^{n+1})}{\mathbf{P}(\mathbf{U}|\mathcal{G}^n)} &= \sum_{i=1}^{I} \log \frac{\mathbf{P}(\mathbf{u}^i|\mathcal{G}^{n+1})}{\mathbf{P}(\mathbf{u}^i|\mathcal{G}^n)} = \sum_{i=1}^{I} \log \frac{\sum\limits_{T\in\Omega_{\mathbf{u}^i}} \mathbf{P}(T|\mathcal{G}^{n+1})}{\mathbf{P}(\mathbf{u}^i|\mathcal{G}^n)} \\
&= \sum_{i=1}^{I} \log \sum_{T\in\Omega_{\mathbf{u}^i}} \left(\frac{\mathbf{P}(T|\mathcal{G}^n)}{\mathbf{P}(\mathbf{u}^i|\mathcal{G}^n)} \cdot \frac{\mathbf{P}(T|\mathcal{G}^{n+1})}{\mathbf{P}(T|\mathcal{G}^n)} \right) \\
&= \sum_{i=1}^{I} \log \sum_{T\in\Omega_{\mathbf{u}^i}} \left(\mathbf{P}(T|\mathbf{u}^i, \mathcal{G}^n) \cdot \frac{\mathbf{P}(T|\mathcal{G}^{n+1})}{\mathbf{P}(T|\mathcal{G}^n)} \right) \\
&\geq \sum_{i=1}^{I} \sum_{T\in\Omega_{\mathbf{u}^i}} \left(\mathbf{P}(T|\mathbf{u}^i, \mathcal{G}^n) \cdot \log \frac{\mathbf{P}(T|\mathcal{G}^{n+1})}{\mathbf{P}(T|\mathcal{G}^n)} \right) = Q(\mathcal{G}^{n+1}, \mathcal{G}^n) - Q(\mathcal{G}^n, \mathcal{G}^n) > 0,
\end{aligned}$$

where we used the fact that $\log$ is a concave function and applied Jensen's inequality. ■

This proposition suggests that we can improve our estimate $\mathcal{G}^n$ by choosing a PCFG $\mathcal{G}^{n+1}$ which makes the Q function larger. Specifically, the EM algorithm chooses $\mathcal{G}^{n+1}$ so as to maximize $Q(\mathcal{G}, \mathcal{G}^n)$

over all possible probability assignments $\mathbf{P}_{root}, \mathbf{P}_{prod}$ of the PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathbf{P}_{root}, \mathbf{P}_{prod})$:

$$\mathcal{G}^{n+1} = \arg \max_{\mathbf{P}_{root}, \mathbf{P}_{prod}} Q(\mathcal{G}, \mathcal{G}^n), \quad \text{where } \mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathbf{P}_{root}, \mathbf{P}_{prod}). \tag{25}$$

We note that no optimization is performed over the sets $\mathcal{N}$, $\mathcal{T}$, and $\mathcal{P}$; however, this does not mean that these sets are always kept fixed by the EM algorithm. Some production rules which are in $\mathcal{P}$ may, in effect, be eliminated by the algorithm because their probability may evolve to zero. If these production rules happened to be terminal, it may also happen that the corresponding member of $\mathcal{T}$ is eliminated, i.e., that it is no longer part of any rule whose probability is nonzero. However, no new production rules can be created, and, because of the normalization equations, nonterminal states cannot be eliminated.

We now derive the update equations for the PCFG parameters.

Proposition 6. *Suppose we have I independent observations $\mathbf{u}^1, \ldots, \mathbf{u}^I$. Let $y^{i,n}$ be the inverse likelihood of the i-th observation after the n-th iteration of the EM algorithm:*

$$y^{i,n} \triangleq \frac{1}{\mathbf{P}(\mathbf{u}^i|\mathcal{G}^n)} = \left(\sum_{j \in \mathcal{N}} c^j_{\mathbf{1},M^i}(\mathbf{u}^i) \mathbf{P}^n_{root}(j) \right)^{-1},$$

where $M^i = (M^i_1, M^i_2)$ is the size of the image $\mathbf{u}^i$. Then the update equations for the PCFG parameters corresponding to Eq. (25) are:

$$\mathbf{P}^{n+1}_{root}(j) = \frac{1}{I} \sum_{i=1}^{I} y^{i,n} c^j_{\mathbf{1},M^i}(\mathbf{u}^i) \mathbf{P}^n_{root}(j), \tag{26}$$

$$\mathbf{P}^{n+1}_{prod}(j \xrightarrow{h} k, \ell) = \frac{\sum_{i=1}^{I} y^{i,n} \sum_{p,q} s^j_{pq}(\mathbf{u}^i) \mathbf{P}^n_{prod}(j \xrightarrow{h} k, \ell) \sum_{d=p_1}^{q_1-1} c^k_{p,(d,q_2)}(\mathbf{u}^i) c^\ell_{(d+1,p_2),q}(\mathbf{u}^i)}{\sum_{i=1}^{I} y^{i,n} \sum_{p,q} s^j_{pq}(\mathbf{u}^i) c^j_{pq}(\mathbf{u}^i)}, \tag{27}$$

$$\mathbf{P}^{n+1}_{prod}(j \xrightarrow{v} k, \ell) = \frac{\sum_{i=1}^{I} y^{i,n} \sum_{p,q} s^j_{pq}(\mathbf{u}^i) \mathbf{P}^n_{prod}(j \xrightarrow{v} k, \ell) \sum_{d=p_2}^{q_2-1} c^k_{p,(q_1,d)}(\mathbf{u}^i) c^\ell_{(p_1,d+1),q}(\mathbf{u}^i)}{\sum_{i=1}^{I} y^{i,n} \sum_{p,q} s^j_{pq}(\mathbf{u}^i) c^j_{pq}(\mathbf{u}^i)}, \tag{28}$$

$$\mathbf{P}^{n+1}_{prod}(j \to u) = \frac{\sum_{i=1}^{I} y^{i,n} \mathbf{P}^n_{prod}(j \to u) \sum_{p:u^i_{pp}=u} s^j_{pp}(\mathbf{u}^i)}{\sum_{i=1}^{I} y^{i,n} \sum_{p,q} s^j_{pq}(\mathbf{u}^i) c^j_{pq}(\mathbf{u}^i)}, \tag{29}$$

where all center and surround variables in the righthand sides are calculated using the PCFG $\mathcal{G}^n$, and where each double summation over p and q is done over all such pairs (p, q) that $\square_{p,q}$ is a nonempty rectangular subdomain of $\square_{\mathbf{1},M^i}$. As before, our convention is that any sum over an empty set is zero.

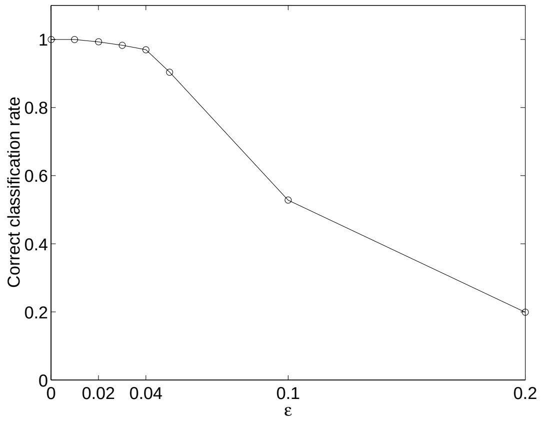


Fig. 8. The rate of correct classification of noisy digit images, as a function of the noise level ε.

Proof. See Appendix. ■

The space complexity of the EM algorithm is the same as that of the center and surround recursions. The time complexity, however, depends not only on the time complexity of the center and surround recursions but also on the number of iterations which is not predictable. The running time of the parameter estimation algorithm, however, is not as critical as for the MAP and likelihood calculation algorithms, since for any specific problem the model parameters are estimated only once, off-line.

V. Experimental Illustrations of the Algorithm

In this section, we illustrate the potential value of our methods with simple examples.

A. Experiment 1: Classification of Noisy Images

We apply our likelihood computation algorithm of Section IV-A to classifying binary images of noisy digits. Our data set consists of the ten digits from the X Windows 9x15 font whose characters are 10×7 pixel images, placed at various locations on a white 14×11 background. These images are corrupted by synthetic noise which independently flips every pixel with probability ε.

For each noise-free digit $k = 0, 1, \ldots, 9$, a probabilistic grammar $\mathcal{G}_k$ is obtained from the EM algorithm of Section IV-C, by training on a single 10×7 image of the digit. Each grammar $\mathcal{G}_k$ is then manually expanded (i.e., several new nonterminal states and nonterminal production rules are introduced) to obtain a new probabilistic grammar capable of placing the image of the digit k at any location on a white background. Using these PCFGs, a PCFG $\mathcal{G}_{k,\varepsilon}$ is obtained for each level of noise ε and each digit $k = 0, 1, \ldots, 9$, by manually modifying the terminal production rule probabilities to model the noise. For example, the PCFG for the digit zero with noise level $\varepsilon = 0.05$ is given in Table I (the production rule probabilities given here are approximate, with only two significant digits).

TABLE I

PCFG $\mathcal{G}_{0,0.05}$ FOR THE DIGIT ZERO WITH NOISE LEVEL $\varepsilon = 0.05$.

The set of nonterminal states: $\mathcal{N} = \{0, 1, \ldots, 29\}$.

The set of terminal states: $\mathcal{T} = \{b, w\}$.

Root state distribution: $\mathbf{P}_{root}(j) = \begin{cases} 0 & \text{for } j = 1, 2, \ldots, 14 \\ 1/16 & \text{otherwise.} \end{cases}$

Production rule	Probability	Production rule	Probability	Production rule	Probability	Production rule	Probability
$0 \xrightarrow{v} 5, 5$	1	$1 \xrightarrow{h} 13, 4$	0.56	$1 \xrightarrow{h} 4, 14$	0.44	$2 \xrightarrow{h} 14, 3$	0.08
$2 \xrightarrow{h} 14, 2$	0.21	$2 \xrightarrow{h} 3, 14$	0.24	$2 \xrightarrow{h} 2, 14$	0.47	$3 \xrightarrow{h} 14, 14$	0.32
$3 \xrightarrow{h} 14, 3$	0.32	$3 \xrightarrow{h} 3, 14$	0.36	$4 \xrightarrow{h} 14, 6$	0.18	$4 \xrightarrow{h} 14, 1$	0.36
$4 \xrightarrow{h} 2, 13$	0.46	$5 \xrightarrow{v} 5, 1$	0.71	$5 \xrightarrow{h} 4, 14$	0.29	$6 \xrightarrow{h} 13, 13$	0.20
$6 \xrightarrow{h} 13, 6$	0.70	$6 \xrightarrow{h} 6, 13$	0.10	$7 \xrightarrow{h} 9, 7$	0.5	$7 \xrightarrow{h} 9, 9$	0.5
$8 \xrightarrow{h} 10, 8$	0.5	$8 \xrightarrow{h} 10, 10$	0.5	$9 \xrightarrow{v} 13, 9$	0.5	$9 \xrightarrow{v} 13, 13$	0.5
$10 \xrightarrow{v} 14, 10$	0.5	$10 \xrightarrow{v} 14, 14$	0.5	$11 \xrightarrow{h} 13, 11$	0.5	$11 \xrightarrow{h} 13, 13$	0.5
$12 \xrightarrow{h} 14, 12$	0.5	$12 \xrightarrow{h} 14, 14$	0.5	$15 \xrightarrow{v} 8, 16$	0.5	$15 \xrightarrow{v} 12, 16$	0.5
$16 \xrightarrow{v} 25, 8$	0.5	$16 \xrightarrow{v} 25, 12$	0.5	$17 \xrightarrow{v} 8, 25$	0.5	$17 \xrightarrow{v} 12, 25$	0.5
$18 \xrightarrow{h} 24, 8$	0.5	$18 \xrightarrow{h} 24, 10$	0.5	$19 \xrightarrow{h} 8, 24$	0.5	$19 \xrightarrow{h} 10, 24$	0.5
$20 \xrightarrow{v} 28, 8$	0.5	$20 \xrightarrow{v} 28, 12$	0.5	$21 \xrightarrow{v} 8, 28$	0.5	$21 \xrightarrow{v} 12, 28$	0.5
$22 \xrightarrow{v} 29, 8$	0.5	$22 \xrightarrow{v} 29, 12$	0.5	$23 \xrightarrow{v} 8, 29$	0.5	$23 \xrightarrow{v} 12, 29$	0.5
$24 \xrightarrow{v} 8, 26$	0.5	$24 \xrightarrow{v} 12, 26$	0.5	$25 \xrightarrow{h} 8, 28$	0.5	$25 \xrightarrow{h} 10, 28$	0.5
$26 \xrightarrow{v} 0, 8$	0.5	$26 \xrightarrow{v} 0, 12$	0.5	$27 \xrightarrow{v} 8, 0$	0.5	$27 \xrightarrow{v} 12, 0$	0.5
$28 \xrightarrow{h} 0, 8$	0.5	$28 \xrightarrow{h} 0, 10$	0.5	$29 \xrightarrow{h} 8, 0$	0.5	$29 \xrightarrow{h} 10, 0$	0.5
$13 \rightarrow b$	0.05	$13 \rightarrow w$	0.95	$14 \rightarrow b$	0.95	$14 \rightarrow w$	0.05

We now describe the results of our classification experiments with noisy digit images: 900 experiments for each noise level $\varepsilon = 0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2$. Specifically, for each digit we consider nine different shifts: centered on the 14×11 background; shifted to the left, right, top, bottom, upper left corner, upper right corner, lower left corner, and lower right corner. For each shift of each digit, we generate 10 noisy versions with independent noise realizations for each noise level ε. Each of the 900 images is classified by calculating its likelihoods with respect to the ten PCFGs $\mathcal{G}_{0,\varepsilon}, \ldots, \mathcal{G}_{9,\varepsilon}$ and choosing the hypothesis corresponding to the largest likelihood. Implemented in C, each likelihood calculation takes about 0.3 seconds on an 800 MHz Pentium III processor–i.e., classifying each image takes about 3 seconds. The PCFGs obtained in this experiment are very sparse. For example, even though PCFG $\mathcal{G}_{0,0.05}$ (see Table I) has 30 nonterminal states, only 60 nonterminal production rules have nonzero

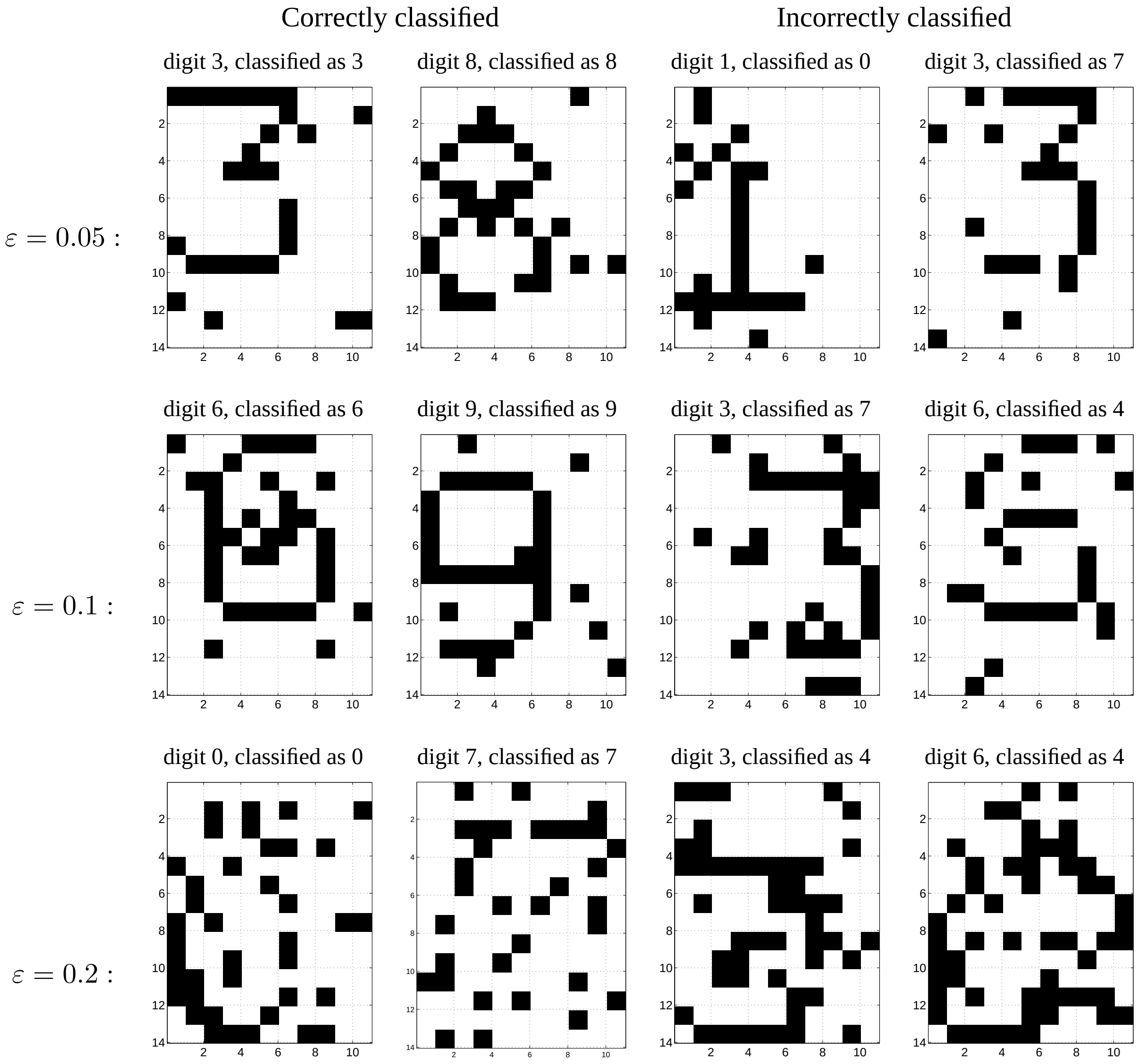
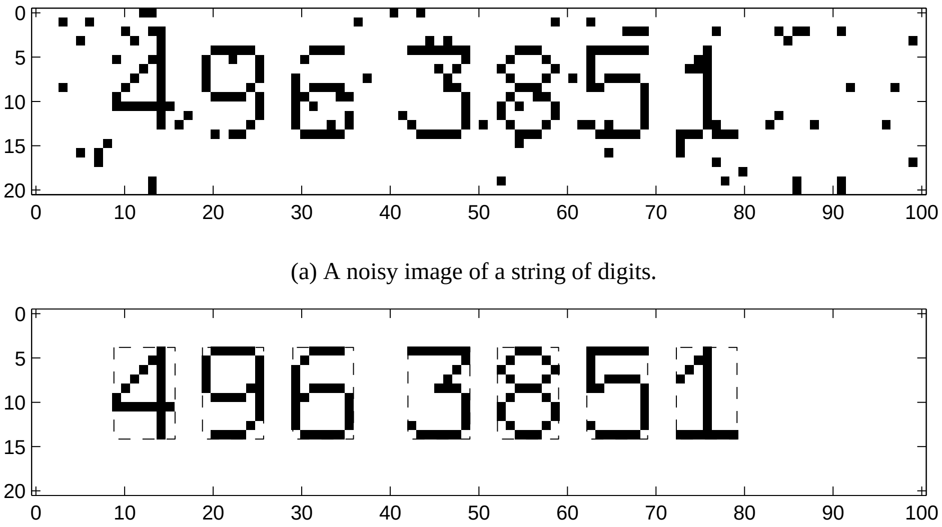


Fig. 9. Classification results for 12 images. Overall correct classification rate for $\varepsilon = 0.05$ is 90%. Overall correct classification rate for $\varepsilon = 0.1$ is 53%. Overall correct classification rate for $\varepsilon = 0.2$ is 20%.

probabilities. This makes the recursive likelihood computation algorithm almost three orders of magnitude faster than it would be if each of the $2 \cdot 30^3 = 54000$ possible nonterminal production rules had a nonzero probability. In addition, this particular experiment is of course parallelizable: the ten likelihoods could be computed in parallel. We moreover are investigating various methods–which have not been used here–of further speeding up the computations.

Our experiments are summarized in Fig. 8, which shows a plot of our estimates of the correct classification probability as a function of the noise level ε, from the noise-free case $\varepsilon = 0$ to the extremely noisy case of $\varepsilon = 0.2$. This latter case corresponds to an average of about 31 incorrect pixels per 14×11 image,



(a) A noisy image of a string of digits.

(b) The image and segmentation (dashed lines) produced by our algorithm.

Fig. 10. Given the noisy image shown in (a), our algorithm identifies each of the seven digits correctly and produces the segmentation shown in dashed lines in (b).

which, as shown in Fig. 9, makes some images unrecognizable to a human. Our estimate of the correct classification probability is simply the number of correct classifications among our 900 experiments, divided by 900. Assuming that the actual correct classification probability is P_ε, and that our experiments are independent, the standard deviation of our estimate is $\sqrt{(P_\varepsilon - P_\varepsilon^2)/900} \leq \sqrt{(1/4)/900} = 1/60$ which is sufficiently accurate for this illustrative example. The plot in Fig. 8 demonstrates the excellent performance of our algorithm and graceful degradation for very noisy images.

Several images from our experiments are shown in Fig. 9. The three rows correspond to noise levels $\varepsilon = 0.05, 0.1$, and 0.2, respectively. With $\varepsilon = 0.05$, a few of the images are difficult for a human to recognize; most of the ones with $\varepsilon = 0.2$ are unrecognizable.

Emphasizing that it is not the goal of this paper to compete with state-of-the-art character recognition algorithms (in fact, a simple algorithm, such as a matched filter, would probably perform very well in the simple example we have just described), we note nevertheless that the performance of our algorithm in this example is promising and demonstrates its viability and potential.

B. Experiment 2: Segmentation and Recognition

In this example, we use the MAP estimation algorithm of Section IV-B to extract a string of digits from a noisy image and classify these digits.

The PCFGs for all ten digits from Experiment 1 are embedded in a larger PCFG which describes strings of seven digits on a white background. Just as in Experiment 1, the PCFG is modified to account for noise. In this PCFG, there are ten special nonterminal states $digit$-0, $digit$-1, $\ldots$, $digit$-9 which are used to label the ten digits. For example, $x_\alpha = digit$-0 is interpreted to mean that digit zero is present in the image and is situated in $\mathcal{Y}(\alpha)$.

Given an image such as that of Fig. 10(a), we use our algorithm to estimate the MAP tree. For each internal vertex α of this tree such that $x_\alpha = digit\text{-}k$, we extract the rectangle $\mathcal{Y}(\alpha)$ and label it as digit k. Our algorithm therefore produces the segmentation of our image into digits and background, and recognizes each digit. For the input image of Fig. 10(a), this results in Fig. 10(b).

VI. Discussion and Conclusions

A. The Inside-Outside Algorithm

As indicated above, the Center-Surround algorithm was motivated by the Inside-Outside algorithm [1, 16, 19] for the 1-D PCFG defined by Eqs. (1,2). In fact, when the number of dimensions D is equal to one, the multidimensional grammar of Eqs. (7,8) reduces to the 1-D grammar of Eqs. (1,2), and the Center-Surround algorithm reduces to the Inside-Outside algorithm. This is also equivalent to the 2-D case of the Center-Surround algorithm for images whose height or width is one. Indeed, it is easily verified that, for $M_1 = 1$, the center recursion (19) becomes the inside recursion (Eq. (6) in [16]), the surround recursion (22) becomes the outside recursion (Eq. (8) in [16]), and the MAP estimation formulas become identical to Eqs. (20) and (21) in [16]. As we have shown in Section III, however, the leap from the 1-D framework to multiple dimensions is not a simple one: the issue arises of unambiguously associating a multidimensional signal with every tree. This issue does not arise in 1-D. Our framework handles this issue by insuring that at most one signal can correspond to a tree. This is proven in the Admissibility Theorem of Section III. On the other hand, in multiple dimensions, there exist trees which do not correspond to a signal and which we call inadmissible trees. This property is immaterial to the algorithms that comprise the Center-Surround algorithm. The existence of inadmissible trees, however, makes the problem of sampling from our model (i.e., producing a sample path which is admissible) much more difficult than sampling from the 1-D PCFG of Eqs. (1,2). This problem is open and is one of the avenues of our current research.

B. Fixed-Tree Discrete-State Multiscale Models

Another inspiration for our work is research on multiscale statistical modeling of images with quadtrees of fixed structure [5, 6, 18]–i.e., quadtrees such as the one in Fig. 11(a) for which the set of vertices and

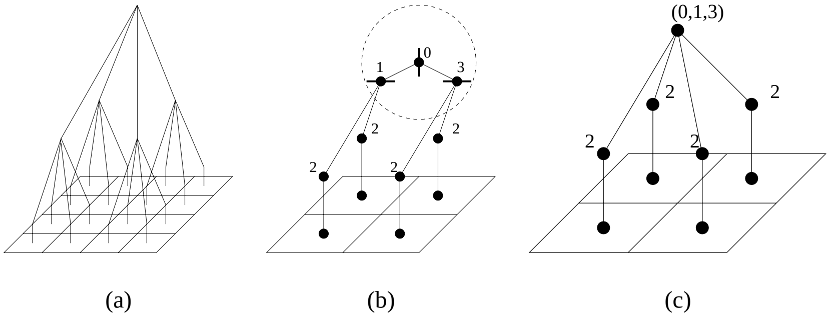


Fig. 11. (a) A quadtree for a 4×4 image. (b) An Ω-tree generated by our image PCFG. This particular Ω-tree is equivalent to the quadtree depicted in (c) which is obtained by combining the three encircled vertices of the Ω-tree into one.

edges is fixed and does not depend on the image. A fixed quadtree is thus able to model a $2^n \times 2^n$ image where n is determined by the depth of the tree. It can be seen that a quadtree structure can be achieved in our 2-D PCFG of Eqs. (7,8) by restricting it in such a way that, in each Ω-tree, horizontal and vertical productions are forced to alternate. It is easy to show that these restrictions can be made by setting the probabilities of certain production rules in Eqs. (7,8) to zero. This is illustrated with a particular tree in Fig. 11(b) which models a 2×2 image. To make this model identical to that of [5, 6], each triple of vertices $(\alpha, C_1(\alpha), C_2(\alpha))$ such that α belongs to an even-numbered generation on the tree, is replaced by a single vertex with vector state $(x_\alpha, x_{C_1(\alpha)}, x_{C_2(\alpha)})$. For example, the three upper vertices of the Ω-tree in Fig. 11(b) become a single vertex with aggregate state (0,1,3), to result in the quadtree of Fig. 11(c).

Our framework is more flexible than that of [5, 6]: instead of imposing a quadtree structure on an image, it can adapt the structure of the tree to the data. For this flexibility, we pay a computational cost. The time complexity of likelihood calculation and MAP estimation in [5, 6] is $O(M_1 M_2 |\mathcal{N}|^2)$ for an $M_1 \times M_2$ image and for a model whose set of hidden states has size $|\mathcal{N}|$. As we saw in Section IV, the corresponding complexity in our case could be much higher. Our current research agenda includes exploiting the sparsity of PCFGs to reduce computational complexity, as well as developing approximate algorithms with improved speed and memory performance.

C. Continuous-State Multiscale Models

Using quadtrees of fixed structure to develop statistical multiscale models for images has its origins in [2, 7, 18] where a continuous-state model was introduced, and the resulting scale-recursive algorithms were devised for calculating the linear least-squares estimates. We are currently pursuing the extension of our framework to the continuous-state case. In this case, discrete probability distributions $\mathbf{P}_{root}$ and $\mathbf{P}_{prod}$ are replaced with probability densities.

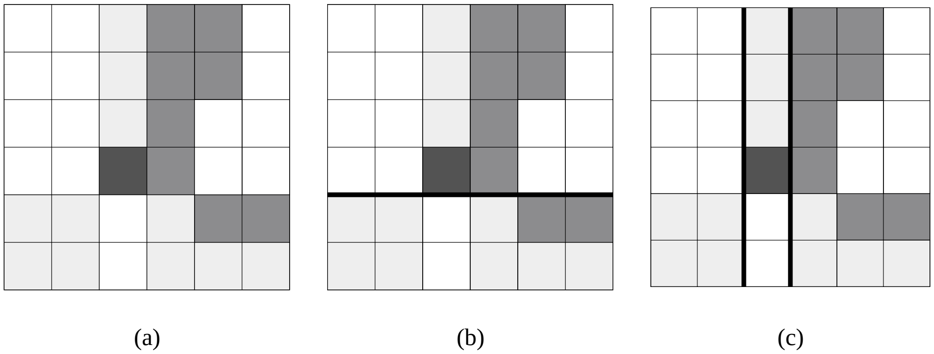
In addition to having a theoretical interest, it is suggested by previous work, such as [17], that this research topic may have significant practical implications. For example, [17] supplemented the traditional hidden Markov models with continuous random variables which model the durations of phonetic segments. These random variables were specified by a parametric family of probability density functions which were given by a small number of parameters. This greatly reduced the complexity of the model.

D. A More General Structure of Observations.

The Center-Surround algorithm was developed above using the most typical measurement scenario, namely, that every pixel value of an image or a set of images is observed. The algorithm is easily generalized to several cases where there may be some additional coarse-scale information, and where some pixel values may be unknown. We now list in detail these alternative scenarios that can be handled by our algorithm.

- It may be known a priori–for example, from a preprocessing segmentation step–that certain groups of pixels belong together. In this case, any tree which does not respect these groupings must have zero probability. For example, preprocessing may indicate that the image of Fig. 12(a) can only be split horizontally along the thick line in Fig. 12(b) but not anywhere else. We use $\mathcal{D}_{pq,h}$ to denote the list of all allowed horizontal split locations for an image subdomain $\square_{pq}$. More formally, $\mathcal{D}_{pq,h}$ is the set of all numbers d such that $\mathcal{Y}^{-1}(\square_{p,(d,q_2)})$ and $\mathcal{Y}^{-1}(\square_{(d+1,p_2),q})$ are allowed to be the children of $\mathcal{Y}^{-1}(\square_{pq})$. $\mathcal{D}_{pq,h}$ can be any (possibly empty) subset of $\{p_1,\ldots,q_1-1\}$, see Fig. 6(a). For example, $\mathcal{D}_{(1,1),(6,6),h} = \{4\}$ for Fig. 12(b).

  Similarly, $\mathcal{D}_{pq,v}$ is the list of all allowed vertical split locations for $\square_{pq}$, i.e., the set of all numbers d such that $\mathcal{Y}^{-1}(\square_{p,(q_1,d)})$ and $\mathcal{Y}^{-1}(\square_{(p_1,d+1),q})$ are allowed to be the children of $\mathcal{Y}^{-1}(\square_{pq})$. $\mathcal{D}_{pq,v}$ can be any (possibly empty) subset of $\{p_2,\ldots,q_2-1\}$, see Fig. 6(b). In the example of Fig. 12(c), $\mathcal{D}_{(1,1),(6,6),v} = \{2,3\}$.
- Some information may be available about the states of internal vertices. For example, we may know a priori the state of the ancestor α of a certain region $\square_{pq}$. More generally, we use $\mathcal{N}_{pq}$ to denote the list of all possible states for the internal vertex α whose yield is $\square_{pq}$ if such vertex exists. $\mathcal{N}_{pq}$ can be any nonempty subset of $\mathcal{N}$.
- Some information about the image pixels, on the other hand, may be missing: for example, the values for a certain group of pixels may be unobserved. More generally, we use $\mathcal{T}_p$ to denote the list of all possible states for the leaf $\square_p$. $\mathcal{T}_p$ can be any nonempty subset of $\mathcal{T}$, in particular, it may


(a)

(b)

(c)

Fig. 12. (a) A 6×6 grayscale image. (b) A single allowed horizontal split location for the 6×6 image domain, obtained through a preprocessing segmentation step. (c) Two allowed vertical split locations.

consist of a single pixel value.

We assume that the dimensions $M = (M_1, M_2)$ of the observed image are part of the observations. Our observation information in this more general case consists of M and the constraint sets $\mathcal{D}_{pq,h}$, $\mathcal{D}_{pq,v}$, $\mathcal{N}_{pq}$, and $\mathcal{T}_p$. In other words, we define our observation information $\mathbf{v}$ as follows:

$$\mathbf{v} \triangleq \{M\} \cup \{\mathcal{N}_{pq}, \mathcal{D}_{pq,h}, \mathcal{D}_{pq,v}\}_{p,q:\square_{pq} \subset \square_{\mathbf{1},M}} \cup \{\mathcal{T}_p\}_{p \in \square_{\mathbf{1},M}}. \tag{30}$$

We then define $\Omega_{\mathbf{v}}$ to be the set of all admissible Ω-trees that satisfy the constraints $\mathbf{v}$, and we use $\mathbf{P}(\mathbf{v})$ as a shorthand notation for $\mathbf{P}(\Omega_{\mathbf{v}})$.

The generalizations of the center and surround recursions, the MAP recursions, and the EM update formulas for this case, are straightforward. We just give the center recursion formulas for data $\mathbf{v}$ as an illustration. For any nonempty rectangular domain $\square_{pq} \subset \square_{\mathbf{1},M}$ with $p \neq q$, and any $j \in \mathcal{N}_{pq}$, Eq. (19) is modified as follows:

$$\begin{aligned} c_{pq}^{j}(\mathbf{v}) &= \sum_{d \in \mathcal{D}_{pq,h}} \sum_{k \in \mathcal{N}_{p,(d,q_2)}} \sum_{\ell \in \mathcal{N}_{(d+1,p_2),q}} \mathbf{P}_{prod}(j \xrightarrow{h} k, \ell) c_{p,(d,q_2)}^{k}(\mathbf{v}) c_{(d+1,p_2),q}^{\ell}(\mathbf{v}) \\ &+ \sum_{d \in \mathcal{D}_{pq,v}} \sum_{k \in \mathcal{N}_{p,(q_1,d)}} \sum_{\ell \in \mathcal{N}_{(p_1,d+1),q}} \mathbf{P}_{prod}(j \xrightarrow{v} k, \ell) c_{p,(q_1,d)}^{k}(\mathbf{v}) c_{(p_1,d+1),q}^{\ell}(\mathbf{v}), \end{aligned} \tag{31}$$

where our convention is that any sum over an empty set is zero. For any $p \in \square_{\mathbf{1},M}$ and any $j \in \mathcal{N}_{pp}$, the initialization equation is:

$$c_{pp}^{j}(\mathbf{v}) = \sum_{u \in \mathcal{T}_p} \mathbf{P}_{prod}(j \rightarrow u). \tag{32}$$

The probability of data $\mathbf{v}$ is computed from the center variables $c_{\mathbf{1},M}^{j}(\mathbf{v})$:

$$\mathbf{P}(\mathbf{v}) = \sum_{j \in \mathcal{N}_{\mathbf{1},M}} c_{\mathbf{1},M}^{j}(\mathbf{v}) \mathbf{P}_{root}(j). \tag{33}$$

The advantage of this more general formulation is that it allows one to handle a wide variety of different modes of observation. In particular, the “partially bracketed” scenario of [22] can be handled by our formulation. This can dramatically reduce the time and space complexity of the likelihood calculation and MAP estimation algorithms. As shown by [22] in the context of natural language processing and by our own preliminary experiments in the context of image classification, partial bracketing and its generalizations also lead to significant improvements in parameter estimation algorithms, both improving the quality of estimates and reducing the computational cost.

We finally remark that the case of D dimensions is handled by having D different constraint sets $\mathcal{D}_{pq,o}$, one for each split orientation o.

E. Further Generalizations of Our Model

Our definition of SRTs requires each leaf of an admissible tree to correspond to a single image pixel. As a consequence, the recursive algorithms for likelihood calculation and MAP estimation always must go down to the pixel level, even if there is no observation associated with some pixels. In certain cases, there may be aggregate information about a rectangular group of pixels. In these cases, there can be substantial computational advantages to modeling this aggregated rectangle by a single leaf vertex. We are now investigating several possible ways of generalizing SRTs to such a scenario.

The current formulation of the SRT model is, moreover, based on the use of a rectangular lattice. This is a restriction for applications in which the observations are on an arbitrary graph. For example, in some applications images may be initially segmented into regions with arbitrary shapes. Information extracted from such regions is then best represented using a sparsely connected graph structure with each vertex corresponding to a segmented region of the image and each edge connecting a pair of neighboring regions. We are currently developing methods for adapting the Center-Surround algorithm to such arbitrary graph structures so that it can be better used in applications such as image interpretation and classification.

F. Applications

Multiscale models are important both because they naturally describe many aspects of the world and also because of computational advantages (many problems can be solved more efficiently by organizing computations in a multiscale manner). This has motivated a large body of research on multiresolution representations, linear and nonlinear scale-spaces, and multiscale statistical models.

Our multiscale model is unique in that it is both statistical and adaptive (i.e., able to adjust the tree structure). We therefore anticipate that SRTs and the Center-Surround algorithm will be useful in a wide variety of applications where it is important to extract an optimal hierarchical structure of the data.

VII. Acknowledgments

We would like to thank Yan Huang, Bill Nagel, James Sherman, and Eric Théa for developing implementation of the Center-Surround algorithm and running the experiments of Section V, as well as for stimulating and helpful discussions.

Appendix

TABLE II

Model Notation for SRTs.

$\mathbf{u}$	an image
$M_1 \times M_2$	image dimensions
$\Box_{pq}$	the set of pixel locations whose upper left corner is $p = (p_1, p_2)$ and whose lower right corner is $q = (q_1, q_2)$
$\mathbf{u}_{pq}$	subimage of $\mathbf{u}$ whose upper left corner is $p = (p_1, p_2)$ and whose lower right corner is $q = (q_1, q_2)$
$\mathcal{T}$	the set of terminal states (e.g., grayscale levels)
$\mathcal{N}$	the set of nonterminal (or hidden) states
$\mathcal{P}^j$	the set of all nonterminal production rules with j in the lefthand side
$\mathbf{P}_{prod}(\Lambda)$	the probability of a production rule Λ
$\mathbf{P}_{root}(j)$	the probability that the root state is j
ρ	the root vertex of a tree
$\mathcal{Y}(\alpha)$	the yield of an internal vertex α, i.e., the set of all leaf descendants of α
Λ_α	the production rule applied at α
$C_1(\alpha)$ and $C_2(\alpha)$	the children of α
x_α	the state at α

Proof of Lemma 1

We prove this lemma for $i' = i + 1$; the case of any other i' follows by induction. Since $T \in \Pi_i$, note that the set $[T]_{i+1}$ is obtained by applying all possible production rules at all the leaves of T with nonterminal states. Let $\{\alpha_1, \ldots, \alpha_p\}$ be the set of all the leaves of T with nonterminal states, and let $j_1, \ldots, j_p$ be the corresponding states. We only prove the lemma for $p = 1$; the case $p > 1$ is similar. When $p = 1$,

$$\mathbf{P}_{i+1}([T]_{i+1}) = \sum_{\Lambda \in \mathcal{P}^{j_1}} \mathbf{P}_i(T)\mathbf{P}_{prod}(\Lambda) \overset{(3)}{=} \mathbf{P}_i(T). \quad \blacksquare$$

PROOF OF LEMMA 2

The set $\Omega_0^j = \Pi_0^j$ consists of a single 0-pruned tree of depth zero whose root state is j. This tree is, in other words, a single vertex with state j which has no children. Let us call this tree T. According to our definition (5), $\mathbf{P}_0(T) = \mathbf{P}_{root}(j)$. Now suppose $i > 0$. Note that $[T]_i$ is the set Ω_i^j since Ω_i^j is the set of all trees in Ω_i whose 0-pruning results in T. Therefore, Lemma 1 can be applied to yield: $\mathbf{P}_0(T) = \mathbf{P}_i([T]_i) = \mathbf{P}_i(\Omega_i^j)$ which, combined with $\mathbf{P}_0(T) = \mathbf{P}_{root}(j)$, implies the statement of Lemma 2. $\blacksquare$

PROOF OF LEMMA 3

(i) Proof of the fact that $\mathcal{A}_\infty$ is an algebra

For any subset A of Ω_i and any i' such that $i' > i$, we let $[A]_{i'}$ be the induced set in $\Omega_{i'}$:

$$[A]_{i'} = \bigcup_{T \in A} [T]_{i'} \qquad \text{for any } A \subset \Omega_{i'}.$$

We let $[\mathcal{A}_i]_{i'}$ be the collection of subsets of $\Omega_{i'}$ induced by $\mathcal{A}_i$:

$$[\mathcal{A}_i]_{i'} = \bigcup_{A \in \mathcal{A}_i} \{[A]_{i'}\}.$$

Note that $[[\mathcal{A}_i]_{i'}] = [\mathcal{A}_i]$.

According to our definition of $\mathcal{A}_\infty$, every element of $\mathcal{A}_\infty$ is of the form $[A]$ where $A \in \mathcal{A}_i$ for some i. We take two arbitrary elements $[A_1]$ and $[A_2]$ of $\mathcal{A}_\infty$ and assume, without loss of generality, that $A_1 \in \mathcal{A}_{i_1}$ and $A_2 \in \mathcal{A}_{i_2}$ with $i_2 \leq i_1$. Since $\mathcal{A}_{i_1}$ is the collection of all subsets of Ω_{i_1}, and since $[A_2]_{i_1}$ is a subset of Ω_{i_1}, we have: $A_1 \cup [A_2]_{i_1} \in \mathcal{A}_{i_1}$, and therefore $[A_1 \cup [A_2]_{i_1}] \in \mathcal{A}_\infty$. But $[A_1 \cup [A_2]_{i_1}] = [A_1] \cup [[A_2]_{i_1}] = [A_1] \cup [A_2]$. We have thus shown that from $[A_1], [A_2] \in \mathcal{A}_\infty$ it follows that $[A_1] \cup [A_2] \in \mathcal{A}_\infty$. Moreover, $A_1^c \in \mathcal{A}_{i_1}$ and therefore $[A_1^c] = [A_1]^c \in \mathcal{A}_\infty$. So, $\mathcal{A}_\infty$ is an algebra.

(ii) Proof of the fact that $\widetilde{\mathbf{P}}$ is a probability measure on $\mathcal{A}_\infty$

By definition, $\widetilde{\mathbf{P}}([A]) \geq 0$ for any $[A] \in \mathcal{A}_\infty$. We also have $\widetilde{\mathbf{P}}(\Omega) = \widetilde{\mathbf{P}}(\Omega_0) = 1$. We now demonstrate that $\widetilde{\mathbf{P}}$ is finitely additive. As above, let $[A_1], [A_2] \in \mathcal{A}_\infty$, with $A_1 \in \mathcal{A}_{i_1}$ and $A_2 \in \mathcal{A}_{i_2}$ where $i_2 \leq i_1$.

Assume that $[A_1]$ and $[A_2]$ are disjoint. Then

$$\begin{aligned}\widetilde{\mathbf{P}}([A_1] \cup [A_2]) &= \widetilde{\mathbf{P}}([A_1 \cup A_2]) = \widetilde{\mathbf{P}}([A_1 \cup [A_2]_{i_1}]) \\ &= \mathbf{P}_{i_1}(A_1 \cup [A_2]_{i_1}) = \mathbf{P}_{i_1}(A_1) + \mathbf{P}_{i_1}([A_2]_{i_1}) \\ &= \mathbf{P}_{i_1}(A_1) + \mathbf{P}_{i_2}(A_2) = \widetilde{\mathbf{P}}([A_1]) + \widetilde{\mathbf{P}}([A_2]).\end{aligned}$$

To prove that $\widetilde{\mathbf{P}}$ is not only finitely additive but also countably additive on $\mathcal{A}_\infty$, we need to show that for any set $B \in \mathcal{A}_\infty$ which is the union of a countably infinite collection of disjoint nonempty sets $[A_1], [A_2], \ldots \in \mathcal{A}_\infty$, we have: $\widetilde{\mathbf{P}}(B) = \widetilde{\mathbf{P}}([A_1]) + \widetilde{\mathbf{P}}([A_2]) + \ldots$. Since the number of these sets is infinite but each $\mathcal{A}_i$ is a finite collection, for any one $\mathcal{A}_i$ there exists a set A_n such that $[A_n]$ is not an element of $[\mathcal{A}_i]$. Consequently, there is no i for which the infinite union $B = [A_1] \cup [A_2] \cup \ldots$ is an element of $[\mathcal{A}_i]$. Therefore, the infinite union B cannot belong to $\mathcal{A}_\infty$. The issue of countable additivity of $\widetilde{\mathbf{P}}$ on $\mathcal{A}_\infty$ is therefore moot, and so $\widetilde{\mathbf{P}}$ is a probability measure on $\mathcal{A}_\infty$. ■

Proof of the Center Recursion Formulas, Proposition 3

To prove the recursion formula (19), suppose that a tree $T \in \Omega_{\mathbf{u}}$ has a vertex α which dominates $\Box_{pq}$ where $p \neq q$. We use β and γ to denote the first and second child of α, respectively, and suppose that the production rule applied at α is the following: $j \xrightarrow{h} k, \ell$. The latter supposition is equivalent to assuming that $x_\alpha = j$, $x_\beta = k$, $x_\gamma = \ell$, and that there exists an integer d such that $\Box_{p,(d,q_2)}$ is the yield of β and $\Box_{(d+1,p_2),q}$ is the yield of γ. We then have

$$c^j_{pq}(\mathbf{u}, T, \alpha) = \mathbf{P}_{prod}(j \xrightarrow{h} k, \ell) c^k_{p,(d,q_2)}(\mathbf{u}, T, \beta) c^\ell_{(d+1,p_2),q}(\mathbf{u}, T, \gamma), \tag{34}$$

as a direct consequence of Eq. (13). If a vertical production rule $j \xrightarrow{v} k, \ell$ were applied at α, we would similarly have:

$$c^j_{pq}(\mathbf{u}, T, \alpha) = \mathbf{P}_{prod}(j \xrightarrow{v} k, \ell) c^k_{p,(q_1,d)}(\mathbf{u}, T, \beta) c^\ell_{(p_1,d+1),q}(\mathbf{u}, T, \gamma). \tag{35}$$

According to our definition of the center variable $c^j_{pq}(\mathbf{u})$ in Eq. (14), the sum of (34) and (35) over all T_α produces $c^j_{pq}(\mathbf{u})$ in the lefthand side. To calculate the righthand side, we observe that knowing T_α is equivalent to knowing the production at α, the left subtree T_β, the right subtree T_γ, and the split location d. Therefore, the sum of (34) and (35) over all T_α is the sum over all allowed split locations, all allowed k and ℓ, and all allowed subtrees T_β and T_γ:

$$c^j_{pq}(\mathbf{u}) \quad = \quad \sum_{d=p_1}^{q_1-1} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} \sum_{T_\beta} \sum_{T_\gamma} \mathbf{P}_{prod}(j \xrightarrow{h} k, \ell) c^k_{p,(d,q_2)}(\mathbf{u}, T, \beta) c^\ell_{(d+1,p_2),q}(\mathbf{u}, T, \gamma)$$

$$\begin{aligned} &+ \sum_{d=p_2}^{q_2-1} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} \sum_{T_\beta} \sum_{T_\gamma} \mathbf{P}_{prod}(j \xrightarrow{v} k,\ell) c^k_{p,(q_1,d)}(\mathbf{u},T,\beta) c^\ell_{(p_1,d+1),q}(\mathbf{u},T,\gamma) \\ &= \sum_{d=p_1}^{q_1-1} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} \mathbf{P}_{prod}(j \xrightarrow{h} k,\ell) \left[\sum_{T_\beta} c^k_{p,(d,q_2)}(\mathbf{u},T,\beta)\right] \cdot \left[\sum_{T_\gamma} c^\ell_{(d+1,p_2),q}(\mathbf{u},T,\gamma)\right] \\ &+ \sum_{d=p_2}^{q_2-1} \sum_{k\in\mathcal{N}} \sum_{\ell\in\mathcal{N}} \mathbf{P}_{prod}(j \xrightarrow{v} k,\ell) \left[\sum_{T_\beta} c^k_{p,(q_1,d)}(\mathbf{u},T,\beta)\right] \cdot \left[\sum_{T_\gamma} c^\ell_{(p_1,d+1),q}(\mathbf{u},T,\gamma)\right], \end{aligned}$$

resulting in Eq. (19).

To derive the base case formula Eq. (20), let α be the internal vertex which dominates pixel p, and suppose that the rule applied at α is $j \to u$. Then the only possible T_α is a tree consisting of two vertices, with state j at the root vertex and state u at the leaf vertex. Therefore,

$$c^j_{pp}(\mathbf{u}) = c^j_{pp}(\mathbf{u},T,\alpha) = \mathbf{P}_{prod}(j \to u),$$

which gives Eq. (20). ■

Proof of the EM Update Formulas, Proposition 6

Using our definition (5) of the probability distribution $\mathbf{P}$, we can write the Q function as follows:

$$\begin{aligned} Q(\mathcal{G},\mathcal{G}^n) &= \sum_{i=1}^{I} \sum_{T\in\Omega_{\mathbf{u}^i}} \mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n) \log \mathbf{P}_{root}(x_\rho) \\ &+ \sum_{i=1}^{I} \sum_{T\in\Omega_{\mathbf{u}^i}} \sum_{\alpha\in\mathcal{V}_{int}} \mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n) \log \mathbf{P}_{prod}(\Lambda_\alpha). \end{aligned} \tag{36}$$

In order to maximize this over $\mathbf{P}_{root}$ and $\mathbf{P}_{prod}$, we need to maximize the first term over $\mathbf{P}_{root}$ and the second term over $\mathbf{P}_{prod}$. We therefore have two constrained maximization problems with constraints given by the normalization equations (3) and (4). We solve these problems using Lagrange multipliers.

We rewrite the first term as follows:

$$\begin{aligned} \sum_{i=1}^{I} \sum_{T\in\Omega_{\mathbf{u}^i}} \mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n) \log \mathbf{P}_{root}(x_\rho) &= \sum_{i=1}^{I} \sum_{j\in\mathcal{N}} \sum_{T\in\Omega^j_{\mathbf{u}^i}} \mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n) \log \mathbf{P}_{root}(j) \\ &= \sum_{i=1}^{I} \sum_{j\in\mathcal{N}} \mathbf{P}(\Omega^j_{\mathbf{u}^i}|\mathbf{u}^i,\mathcal{G}^n) \log \mathbf{P}_{root}(j) \\ &= \sum_{i=1}^{I} \sum_{j\in\mathcal{N}} \mathbf{P}(\Omega^j|\mathbf{u}^i,\mathcal{G}^n) \log \mathbf{P}_{root}(j), \end{aligned}$$

where $\Omega^j_{\mathbf{u}^i} = \Omega_{\mathbf{u}^i} \cap \Omega^j$ is the set of all elements of $\Omega_{\mathbf{u}^i}$ with root state j. Denoting a Lagrange multiplier by t, we therefore obtain, for any $j \in \mathcal{N}$:

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{P}_{root}(j)} \left[\sum_{i=1}^{I} \sum_{j'} \mathbf{P}(\Omega^{j'} | \mathbf{u}^i, \mathcal{G}^n) \log \mathbf{P}_{root}(j') + t \left(1 - \sum_{j'} \mathbf{P}_{root}(j') \right) \right] = 0 \\
&\sum_{i=1}^{I} \frac{\mathbf{P}(\Omega^j | \mathbf{u}^i, \mathcal{G}^n)}{\mathbf{P}_{root}(j)} - t = 0 \qquad (37) \\
&\mathbf{P}_{root}(j) = \frac{\sum_{i=1}^{I} \mathbf{P}(\Omega^j | \mathbf{u}^i, \mathcal{G}^n)}{t} \qquad (38)
\end{aligned}$$

Multiplying (37) by $\mathbf{P}_{root}(j)$, summing over j, and using the normalization equation (4), we get:

$$\begin{aligned}
\sum_{i=1}^{I} \sum_{j} \mathbf{P}(\Omega^j | \mathbf{u}^i, \mathcal{G}^n) &= \sum_{j} \mathbf{P}_{root}(j) t \\
I &= t.
\end{aligned}$$

We substitute this into (38) and apply the Bayes rule in order to get an expression in terms of quantities computable with center variables:

$$\begin{aligned}
\mathbf{P}^{n+1}_{root}(j) &= \frac{1}{I} \sum_{i=1}^{I} \mathbf{P}(\Omega^j | \mathbf{u}^i, \mathcal{G}^n) = \frac{1}{I} \sum_{i=1}^{I} \frac{\mathbf{P}(\mathbf{u}^i | \Omega^j, \mathcal{G}^n) \mathbf{P}(\Omega^j | \mathcal{G}^n)}{\mathbf{P}(\mathbf{u}^i | \mathcal{G}^n)} \\
&= \frac{1}{I} \sum_{i=1}^{I} \frac{\mathbf{P}(\mathbf{u}^i | \Omega^j, \mathcal{G}^n) \mathbf{P}^n_{root}(j)}{\mathbf{P}(\mathbf{u}^i | \mathcal{G}^n)} = \frac{1}{I} \sum_{i=1}^{I} y^{i,n} c^j_{\mathbf{1},M^i}(\mathbf{u}^i) \mathbf{P}^n_{root}(j).
\end{aligned}$$

We use $\#_\Lambda(T)$ and $\#_j(T)$ to denote the number of occurrences of production Λ and state j, respectively, in a tree T. We now rewrite the second term of the Q function (36) as follows:

$$\sum_{i=1}^{I} \sum_{T \in \Omega_{\mathbf{u}^i}} \sum_{\alpha \in \mathcal{V}_{int}} \mathbf{P}(T | \mathbf{u}^i, \mathcal{G}^n) \log \mathbf{P}_{prod}(\Lambda_\alpha) = \sum_{i=1}^{I} \sum_{j \in \mathcal{N}, \Lambda \in \mathcal{P}^j} \sum_{T \in \Omega_{\mathbf{u}^i}} \mathbf{P}(T | \mathbf{u}^i, \mathcal{G}^n) \#_\Lambda(T) \log \mathbf{P}_{prod}(\Lambda).$$

For each nonterminal state j, we use Lagrange multiplier t_j:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{P}_{prod}(\Lambda)} \quad & \left[\sum_{i=1}^{I} \sum_{j' \in \mathcal{N}, \Lambda' \in \mathcal{P}^{j'}} \log \mathbf{P}_{prod}(\Lambda') \sum_{T \in \Omega_{\mathbf{u}^i}} \mathbf{P}(T | \mathbf{u}^i, \mathcal{G}^n) \#_{\Lambda'}(T) \right. \\
+ \quad & \left. \sum_{j' \in \mathcal{N}} t_{j'} \left(1 - \sum_{r' \in \mathcal{P}^{j'}} \mathbf{P}_{prod}(\Lambda') \right) \right] = 0 \\
\sum_{i=1}^{I} \frac{1}{\mathbf{P}_{prod}(\Lambda)} \quad & \sum_{T \in \Omega_{\mathbf{u}^i}} \quad \mathbf{P}(T | \mathbf{u}^i, \mathcal{G}^n) \#_\Lambda(T) - t_j = 0
\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{prod}(\Lambda) &= \frac{1}{t_j}\sum_{i=1}^{I}\sum_{T\in\Omega_{\mathbf{u}^i}}\mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n)\#_\Lambda(T)\\ t_j &= \sum_{\Lambda\in\mathcal{P}^j}\sum_{i=1}^{I}\sum_{T\in\Omega_{\mathbf{u}^i}}\mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n)\#_\Lambda(T) = \sum_{i=1}^{I}\sum_{T\in\Omega_{\mathbf{u}^i}}\mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n)\#_j(T)\end{aligned}$$

This results in the following update equation for the production probabilities:

$$\mathbf{P}^{n+1}_{prod}(\Lambda) = \frac{\displaystyle\sum_{i=1}^{I}\sum_{T\in\Omega_{\mathbf{u}^i}}\mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n)\#_\Lambda(T)}{\displaystyle\sum_{i=1}^{I}\sum_{T\in\Omega_{\mathbf{u}^i}}\mathbf{P}(T|\mathbf{u}^i,\mathcal{G}^n)\#_j(T)} = \frac{\displaystyle\sum_{i=1}^{I}\frac{1}{\mathbf{P}(\mathbf{u}^i|\mathcal{G}^n)}\sum_{T\in\Omega_{\mathbf{u}^i}}\mathbf{P}(T|\mathcal{G}^n)\#_\Lambda(T)}{\displaystyle\sum_{i=1}^{I}\frac{1}{\mathbf{P}(\mathbf{u}^i|\mathcal{G}^n)}\sum_{T\in\Omega_{\mathbf{u}^i}}\mathbf{P}(T|\mathcal{G}^n)\#_j(T)} \tag{39}$$

To calculate the inner sum of the numerator, first suppose that $T\in\Omega_{\mathbf{u}^i}$ is a tree whose vertex α dominates $\Box_{pq}$, for some p and q with $p\neq q$. Let us call the children of α $\beta = C_1(\alpha)$ and $\gamma = C_2(\alpha)$ and suppose that the production rule applied at α is $j \overset{h}{\to} k,\ell$. The latter supposition is equivalent to assuming that $x_\alpha = j$, $x_\beta = k$, $x_\gamma = \ell$, and that there exists d such that $\Box_{p,(d,q_2)}$ is the yield of β and $\Box_{(d+1,p_2),q}$ is the yield of γ. We then have, using Eqs. (13,16),

$$\mathbf{P}(T) = s^j_{pq}(\mathbf{u}^i,T,\alpha)\mathbf{P}^n_{prod}(j\overset{h}{\to}k,\ell)c^k_{p,(d,q_2)}(\mathbf{u}^i,T,\beta)c^\ell_{(d+1,p_2),q}(\mathbf{u}^i,T,\gamma).$$

The inner sum of the numerator of (39) can be obtained by summing this over all T_β, T_γ, $\overline{T}_\alpha$, d, p, and q, to yield the expression in the numerator of (27). Vertical and terminal productions are handled similarly, to result in the numerators of (28) and (29), respectively. The inner sum in the denominator of (39) is also handled similarly, to result in the denominators of (27-29). ■

References

[1] J. Baker. Trainable grammars for speech recognition. In *Speech Communications Papers for the 97th Meeting of the Acoustical Society of America*, D. Klatt and J. Wolf, Eds., pp. 557-550, 1979.

[2] M. Basseville, A. Benveniste, K. Chou, S. Golden, R. Nikoukhah, and A. Willsky. Modeling and estimation of multiresolution stochastic processes. *Trans. Inf. Theory*, 38(2):766-784, March 1992.

[3] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.*, 41(1):164-171, 1970.

[4] I.J. Bienaymé. De la loi de multiplication et de la durée des familles. *Soc. Philomat. Paris Extraits, Sér.* 5:37-39, 1845. Reprinted as an Appendix to [14].

[5] C. Bouman and M. Shapiro. Multispectral image segmentation using a multiscale model. In *Proceedings of ICASSP*, pp. 565-568, San Francisco, CA, USA, 1992.

[6] C. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Trans. Im. Proc.*, 3(2):162-177, March 1994.

[7] K.C. Chou, A.S. Willsky, A. Benveniste, and M. Basseville. Recursive and iterative estimation algorithms for multiresolution stochastic processes. In *Proc. of 29th IEEE Conf. Decision Contr.*, Honolulu, HI, USA, 1989.

[8] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[9] C.J. Everett and S. Ulam. Multiplicative Systems in Several Variables, I, II, III. Technical reports LA-683, LA-690, LA 707, Los Alamos Scientific Laboratory, 1948. Reprinted in *Analogies Between Analogies*, A.R. Bednarek and F. Ulam, Eds., University of California Press, 1990.

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

[11] T.E. Harris. *The Theory of Branching Processes.* Springer-Verlag, 1963.

[12] Stochastic language model for analyzing document physical layout. In *Document Recognition and Retrieval IX, Proceedings of SPIE vol. 4670*, P.B. Kantor, T. Kanungo, J. Zhou, Eds., SPIE, 2002.

[13] D.G. Kendall. Branching processes since 1873. *Journal London Math. Soc.*, 41:385-406, 1966.

[14] D.G. Kendall. The genealogy of genealogy: Branching processes before (and after) 1873. *Bull. London Math. Soc.*, 7:225-253, 1975.

[15] A.N. Kolmogorov and N.A. Dmitriev. Branching stochastic processes. *Doklady Akademii Nauk SSSR*, 56(1):5-8, 1947.

[16] K. Lari and S. Young. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35-56, 1990.

[17] S.E. Levinson. Continuously variable duration hidden Markov models for speech analysis. In *Proceedings of ICASSP*, pp. 1241-1244, Tokyo, Japan, 1986.

[18] M.R. Luettgen, W.C. Karl, A.S. Willsky, and R.R. Tenney. Multiscale representations of Markov random fields. *IEEE Trans. Signal Proc.*, 41(12):3377-3395, 1993.

[19] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[20] J. Neveu. *Mathematical Foundations of the Calculus of Probability.* Holden-Day, Inc., 1965.

[21] R. Otter. The multiplicative process. *The Annals of Mathematical Statistics*, 20:206-224, 1949.

[22] F. Pereira and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pp. 128-135, Newark, Delaware, 1992.

[23] D. Potter. *Compositional Pattern Recognition.* PhD Thesis, Brown University, 1999. `http://www.dam.brown.edu/people/dfp`

[24] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, January 1986.

[25] A. Rosenfeld. *Picture Languages.* Academic Press, 1979.

[26] D. Sankoff. Branching processes with terminal types: application to context-free grammars. *Journal of Applied Probability*, 8:233-240, 1971.

[27] A.C. Shaw. A formal picture description scheme as a basis for picture processing systems. *Information and Control*, 14:9-52, 1969.

[28] H.W. Watson and F. Galton. On the probability of extinction of families. *Journal of the Anthropological Institute of Great Britain and Ireland*, 4:138-144, 1875.