Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2008-011 February 25, 2008

Learning Grammatical Models for Object Recognition

Meg Aycinena, Leslie Pack Kaelbling, and Tomas Lozano-Perez

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Learning Grammatical Models for Object Recognition

Meg Aycinena, Leslie Pack Kaelbling, and Tomas Lozano-Perez
MIT CSAIL

February 23, 2008

Abstract

Many object recognition systems are limited by their inability to share common parts or structure among related object classes. This capability is desirable because it allows information about parts and relationships in one object class to be generalized to other classes for which it is relevant. With this goal in mind, we have designed a representation and recognition framework that captures structural variability and shared part structure within and among object classes. The framework uses probabilistic geometric grammars (PGGs) to represent object classes recursively in terms of their parts, thereby exploiting the hierarchical and substitutive structure inherent to many types of objects. To incorporate geometric and appearance information, we extend traditional probabilistic context-free grammars to represent distributions over the relative geometric characteristics of object parts as well as the appearance of primitive parts. We describe an efficient dynamic programming algorithm for object categorization and localization in images given a PGG model. We also develop an EM algorithm to estimate the parameters of a grammar structure from training data, and a search-based structure learning approach that finds a compact grammar to explain the image data while sharing substructure among classes. Finally, we describe a set of experiments that demonstrate empirically that the system provides a performance benefit.

1 Introduction

Many current approaches to object recognition are characterized by their representation of an object class as a collection of parts with some local appearance properties, and a model of the spatial relations among them. This representation is intuitive and attractive; object classes are often too variable to be described well using a single shape or appearance model, but they can be naturally modeled as a distribution over a set of parts and the relationships among them.

Most of these systems, however, cannot share common part models or spatial structure among related object classes. This capability would allow information about parts and relationships in one object class to be generalized to other relevant classes. For example, we might like to transfer knowledge about the relationships among the arms and back of a chair to all chair classes that have arms and backs, regardless of whether the base is composed of four legs or an axle and wheel-leg. We argue that modeling structural variability and shared part structure will allow effective parameter learning from fewer examples and better generalization of the learned models to unseen data.

Additionally, a system which models shared structure can exploit its knowledge to perform more efficient recognition. For example, we might search for regions of an image which look like four legs and a top without committing to whether we are looking for a table or chair. Furthermore, a representation which captures structural variability within object classes offers the potential to be generalized to model variability in scenes that share objects and arrangements of objects, just as objects share parts and part structures.

With these goals in mind, we present a representation and recognition framework that captures structural variability within and among object classes. We introduce *probabilistic geometric grammars* (PGGs), which represent object classes recursively in terms of their parts, thereby exploiting the hierarchical and substitutive structure inherent to many types of objects. PGGs extend probabilistic context-free grammars (PCFGs),

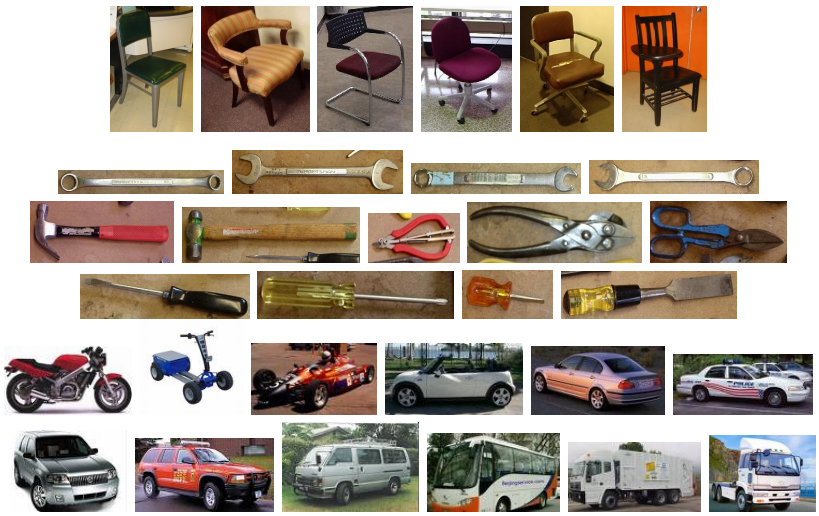


Figure 1: Many object classes exhibit structural variability.

which were developed to model natural language. To incorporate geometric and appearance information, we supplement the traditional PCFG representation with distributions over the geometric and appearance characteristics of object parts.

The PGG representation is particularly motivated by the characteristics of object classes such as furniture, tools, and vehicles, whose shape and appearance are often defined only by functional constraints and by custom. These types of classes often exhibit high variability in shape among members (see Figure 1); this variability is highly parts-based, and displays modular, hierarchical, and substitutive structure. There is structure in the type and number of parts that are present: a chair often consists of a back, seat, and four legs, or a back, seat, two arms, and four legs, but rarely a back, seat, one arm, and three legs. There are also conditional independences in the presence and shape of the parts: whether a chair has arms or not is independent of whether its base consists of four legs or an axle and wheel-legs, given the presence and location of the seat and back.

Context-free grammars capture this type of structural variability by compactly modeling hierarchical groups and substitution of subparts, and they naturally represent conditional independences between subgroups with the context-free assumption. They also allow a compact representation of the combinatorial variability in complex human-made shapes. Probabilistic grammars further allow the specification of distributions over the combination of subparts.

In this paper, we present the elements of the PGG model and describe an efficient dynamic programming recognition algorithm. We then develop an EM algorithm to address parameter learning, and a search-based structure learning approach that finds a compact grammar while sharing substructure among classes. Finally, we describe a set of experiments that demonstrate the effectiveness of the representation and algorithms.

2 Related Work

This research draws on several lines of work in object recognition and computer vision, both recent and historical: parts-based object class models, the use of hierarchical part models and part sharing among object classes, and the use of grammars.

The PGG framework is inspired by recent work on parts-based object class models, particularly those that represent an object as a collection of parts, an appearance model for each part, and a statistical model of the geometric relations among the parts. This approach has existed in the literature for decades, beginning with Fischler and Elschlager (1973), but there has been an increase of activity in this area in the last few

years. Several prominent examples include the constellation (or star) model (Fergus et al., 2003, 2005), and statistical pictorial structures or k-fans (Felzenszwalb & Huttenlocher, 2005; Crandall et al., 2005; Crandall & Huttenlocher, 2006, 2007).

In contrast to these approaches, in which each object class consists of an unstructured set of parts, the PGG model uses a fundamentally hierarchical notion of object and part. The use of part hierarchies to model object classes, especially to enable part sharing among classes, has become increasingly popular in recent years. Although they do not explicitly model a hierarchy of parts, Torralba et al. (2004, 2007) have shown that sharing part appearance models among different object classes improves both the learning rate and the recognition accuracy of the model. Ullman and Epshtein (2006) have demonstrated that using a hierarchy of fragment-based parts produces more informative and discriminative object models for classification. A line of recent work by Sudderth et al. (2005a, 2005b, 2006) has explored modeling variability in the number and structure of parts in an object, or objects in a scene. Bar Hillel and Weinshall (2006) have shown that modeling object subclasses with respect to the basic class improves recognition and learning. Finally, recent work by Felzenszwalb and Schwartz (2007), Fidler and Leonardis (2007), and Ommer and Buhmann (2007) has investigated the notion that learning hierarchical and compositional representations for classes of objects might allow learning and recognition to be more efficient.

The PGG model, while similar in spirit to these hierarchical approaches, differs fundamentally in that it allows choice ("OR") nodes in additional to the "AND" nodes that exists in simple part hierarchies; this difference is what makes it a grammar. The use of (deterministic) grammars and syntactic models was quite popular in early computer vision and pattern recognition research (Rosenfeld, 1973), but until recently, grammars had largely disappeared from modern object recognition. Most recent uses of grammars in vision have focused on modeling the segmentation of entire images, rather than object classes, (e.g., Pollak et al., 2003; Tu et al., 2005; Zhu & Mumford, 2006; Siskind et al., 2007), or on detecting mid-level visual objects, such as regions, curves, and rectangles (e.g., Han & Zhu, 2005; Tu & Zhu, 2006). The work of Zhu et al. (2006), in which they develop a grammatical model for object classification and localization, is a notable exception. Our approach contrasts with theirs in that our model exploits stronger conditional independence assumptions, allowing simple and robust algorithms for recognition and grammar learning.

Although this research and most of the above related work uses models of both the appearance of individual object parts and of the geometry among them, there has been a notable vein of recent work that focuses on modeling part appearance alone, without any geometry. These approaches, originally influenced by bag of words models in language, which do not take word order into account, have enjoyed surprising success at the image classification task; prominent examples include work by Csurka et al. (2004), Sivic et al. (2005), and Grauman and Darrell (2005). It seems clear that, for some object classes, the appearance of large numbers of visual features associated with the object class can be more characteristic and discriminative than the spatial relationships among these features.

However, these approaches seems to work best on classes with distinctive textural or pattern-based features (e.g., motorbikes, cars, and spotted cats), since these are the types of image properties that most modern feature detectors are best suited to capture and represent. For other classes, particularly those with strong structural characteristics and very little distinctive pattern or texture, geometry may be a far more powerful cue. Consider, for example, the classes that Grauman and Darrell's otherwise successful pyramid histogram match algorithm have found most difficult: ants, crabs, cranes, and anchors. In these cases, our current ability to model the appearance of visual features is limited, and models that incorporate geometry might offer potential for greater success than those that depend on appearance models alone.

Much of the work cited here uses purely generative models to represent object classes. However, there are significant reasons to consider discriminative approaches. Image data is extremely noisy and high-dimensional, and it often contains far more information than is actually necessary for the object category recognition task; as a results, generative approaches can be crippled in their attempt to explain every pixel in an image, thus wasting effort on modeling possibly unimportant aspects of the imaging process. Recent research that has successfully taken a purely discriminative approach includes work by Grauman and Darrell (2006). However, generative models are often an intuitive way to model object classes, particularly when considering the goals of structural sharing and generalization. With this in mind, our current approach to

chair:
  1.0 chair → top (ϕ_{000}) base (ϕ_{001})
top:
  0.55 top → seat (ϕ_{100}) back (ϕ_{101})
  0.45 top → seat (ϕ_{110}) back (ϕ_{111}) arm (ϕ_{112}) arm (ϕ_{113})
base:
  0.65 base → leg (ϕ_{200}) leg (ϕ_{201}) leg (ϕ_{202}) leg (ϕ_{203})
  0.35 base → axle (ϕ_{210}) wheel-leg (ϕ_{211}) wheel-leg (ϕ_{212}) wheel-leg (ϕ_{213})

seat (A_3)
back (A_4)
arm (A_5)
leg (A_6)
axle (A_7)
wheel-leg (A_8)

Figure 2: A textual description of a PGG for chairs. The symbols ϕ_{crk} represent conditional geometry models, while A_c represent appearance models.

modeling part appearance incorporates a discriminative aspect to our otherwise purely generative model. Recent examples of work that has successfully incorporated both approaches include that by Leibe and Schiele (2003), Tu et al. (2005), and Bar Hillel and Weinshall (2006).

Finally, our approach to learning PGG models carries on a long tradition of structure and grammar learning in the graphical models and natural language processing communities. Our search-based structure learning method has a similar feel to standard Bayesian network learning algorithms (Heckerman, 1999), while our specific grammar search operators resemble those in many classical approaches to grammar learning for language; e.g., those of Chen (1995), de Marcken (1996), and Nevill-Manning and Witten (1997).

3 The PGG Model

The probabilistic geometric grammar model augments traditional PCFGs with geometry and appearance models in order to represent object classes. A PGG is a set of part models, each of which is either primitive or composite (similar to terminals and nonterminals in PCFGs). A composite part model consists of a set of rules which define how the part can be broken down into subparts. A primitive part model consists of an appearance model which describes a distribution over the part's image appearance. Figure 2 shows an example of a PGG for chairs. We will describe these components in greater detail in the following sections.

For the purposes of this paper, we exploit the fact that an image I can be broken down into a finite set of components. These components could be locations, windows, or arbitrarily shaped regions. Then, for each window w and each part model c in the image, we define a binary random variable X_{cw} that denotes that window w contains an object or part of type c. We will define the inference and learning algorithms for the PGG model in terms of these variables.

3.1 Rules

Each rule r for a composite part model c defines one way that the part can be composed of subparts. A rule consists of an expansion probability $\gamma_{cr} \in [0, 1]$, and a set of rule parts. The expansion probabilities γ_{cr} must all sum to one for a fixed parent part c:

$$\forall_c \sum_r \gamma_{cr} = 1 \ .$$

Thus, the set of rules defines a distribution over the choice of ways to expand the part—an "OR" node, while rule defines an "AND" relationship over its component rule parts.

Each child rule part k in rule r for parent part model c has two components: an index d_{crk} which refers to another part model in the grammar, and a conditional geometry model ϕ_{crk}, which defines a distribution

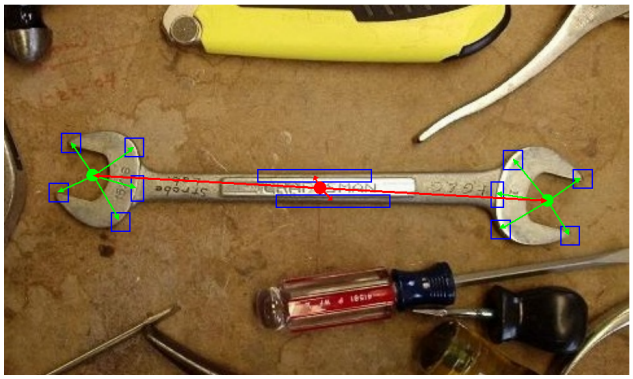


Figure 3: An example of a parse tree for a wrench. Note that only primitive parts (in blue) correspond to actual pixels, while composite parts (in red and green) essentially serve to define a reference frame for the spatial relationships of their children. The grammar upon which this parse tree is based is shown in Figure 11.

on the geometry properties of this subpart given those of its parent part (of type c). Crucially, we assume that the geometric and image properties of subparts are conditionally independent given the properties of the parent part c.

We adopt the convention of writing a rule r for part c using the form:

$$\gamma_{cr} \quad c \rightarrow \ldots, d_{crk}(\phi_{crk}), \ldots$$

Again, see Figure 2 for a concrete example of rules in a PGG model.

One way to interpret a rule is that it expresses a compositional relationship; it states that a part with class c can be composed of a set of subparts, where the kth subpart in rule r has class d_{crk}. More generally, composite part models may be viewed as hidden variables that represent geometric information upon which their child parts depend. In either of these cases, however, composite parts do not directly model pixels in the image—only primitive parts do this. This is is a crucial difference between the PGG framework and other parts-based approaches that also use tree-like structures to represent the relationship among object parts. In other approaches, all nodes of the tree, including internal ones, are primitive parts of the object. In a PGG model, primitive parts can only exist at the leaves of a parse tree, and internal nodes represent composite parts. An example of a parse tree for a wrench is shown overlaid on an image in Figure 3.

We can also think of the expansion probabilities as defining a distribution over Bayes nets. Each internal node in each Bayes net represents the geometry of a composite object part, while each leaf node represents the geometry *and* appearance of a primitive part.

3.2 Conditional Geometry Models

Each rule part k has a geometry model ϕ_{crk}, which models a conditional distribution over the geometric attributes of the kth child part, given the attributes of the parent part. The attributes over which the models are defined may be anything: location, scale, orientation, shape, etc.

In this paper, we model only the relative location of part centroids. We make the simplifying assumption that the centroid location (x_v, y_v) of a child part that occupies window v is conditionally related to the centroid (x_w, y_w) of a parent part in window w by simple translation. Then we define a Gaussian distribution over the position of v relative to w:

$$P(v|w; \phi_{crk}) = \mathcal{N}(x_v - x_w, y_v - y_w; \mu_{crk}, \Sigma_{crk}) \ .$$

To avoid overfitting, we use a diagonal covariance. Despite this simple model, multimodal distributions can

be handled naturally using multiple rules with the same symbols but different geometry models, effectively yielding Gaussian mixtures.

3.3 Appearance Models

Each primitive part model c has an appearance model A_{c} which defines the appearance or material properties of the part. The appearance model for a part allows the image evidence at a particular location or region in the image to be incorporated into the evaluation of the overall object.

The PGG formulation is modular with respect to the representation for the appearance model; the only requirement is that the model enable the calculation of the image likelihood ratio:

$$\frac{\mathsf{P}(\mathsf{I}_w|\mathsf{X}_{\mathsf{c}w})}{\mathsf{P}(\mathsf{I}_w)}$$

i.e., the ratio of the likelihood of the image pixels I_w in window w, given that the window contains a primitive part of type c, to the unconditional image likelihood $\mathsf{P}(\mathsf{I}_w)$. Using the ratio ensures that every image pixel that is not assigned to a specific primitive part is evaluated according to the unconditional model. This allows us to compare object detections occupying different numbers of pixels. Section 7.1.1 describes our current representation for this quantity in detail.

4 Efficient Recognition with PGGs

In this section, we present a top-down dynamic programming algorithm for object classification and localization in an image. It extends the pictorial structures recognition algorithm (Felzenszwalb & Huttenlocher, 2005) to hierarchical part models and choice nodes. The algorithm depends on a discretization of the image into a finite set of locations or regions, so it can recursively calculate a score for each region w and part model c while caching and reusing intermediate results.

4.1 Deriving an Algorithm

Two assumptions will enable our derivation: that primitive parts explain different parts of the image, and that subparts are conditionally independent given their parent. If the primitive parts overlap, then the derivation is approximate. There is not, however, any explicit assumption that subparts must form contiguous regions. Furthermore, we do not assume the union of subparts equals the parent part, since the background model explains pixels not assigned to a primitive part. Again, see Figure 3 for an example of a parse tree in an image, and note that the union of a set of child parts (e.g. the blue parts on the left end of the wrench) in no way equals their parent part (the green point on the left).

Recall that we defined a binary random variable $\mathsf{X}_{\mathsf{c}w}$ which denotes that window w contains an object or part of type c. Given an image with pixels or features I, we frame the recognition problem as finding the root part model c and window w that maximizes $\mathsf{P}(\mathsf{X}_{\mathsf{c}w}|\mathsf{I})$. First, we apply Bayes rule and observe that the denominator $\mathsf{P}(\mathsf{I})$ is constant for all c and w and so can be removed:

$$\begin{aligned}\operatorname*{argmax}_{\mathsf{c},w} \mathsf{P}(\mathsf{X}_{\mathsf{c}w}|\mathsf{I}) &= \operatorname*{argmax}_{\mathsf{c},w} \frac{\mathsf{P}(\mathsf{I}|\mathsf{X}_{\mathsf{c}w})\mathsf{P}(\mathsf{X}_{\mathsf{c}w})}{\mathsf{P}(\mathsf{I})} \\ &= \operatorname*{argmax}_{\mathsf{c},w} \mathsf{P}(\mathsf{I}|\mathsf{X}_{\mathsf{c}w})\mathsf{P}(\mathsf{X}_{\mathsf{c}w})\end{aligned}$$

Next we partition the image features into those inside and outside the candidate window w, and evaluate the features $\mathsf{I}_{\overline{w}}$ not in w according to an unconditional "background model" $\mathsf{P}(\mathsf{I}_{\overline{w}})$:

$$= \operatorname*{argmax}_{\mathsf{c},w} \mathsf{P}(\mathsf{I}_w|\mathsf{X}_{\mathsf{c}w})\mathsf{P}(\mathsf{I}_{\overline{w}})\mathsf{P}(\mathsf{X}_{\mathsf{c}w})$$

We make the common assumption that foreground and background pixels are conditionally independent given the partition, so that $P(I_{\overline{w}}) = P(I)/P(I_w)$. Again, the term $P(I)$ is constant with respect to c and w, so we can remove it:

$$= \underset{c,w}{\operatorname{argmax}} \frac{P(I_w|X_{cw})}{P(I_w)} P(I) P(X_{cw})$$
$$= \underset{c,w}{\operatorname{argmax}} \frac{P(I_w|X_{cw})}{P(I_w)} P(X_{cw})$$

For this paper, we assume that $P(X_{cw})$ is uniform so that all object classes and locations are equally likely *a priori*, so the term becomes constant and may be removed. (Alternatively, this term could naturally represent a context model for the where in the image we expect to see objects appear.)

$$= \underset{c,w}{\operatorname{argmax}} \frac{P(I_w|X_{cw})}{P(I_w)}$$

We can recursively decompose the image likelihood ratio according to the grammar. Let $\beta(c, w)$ be the likelihood ratio for part model c and window w:

$$\beta(c, w) = \frac{P(I_w|X_{cw})}{P(I_w)}$$

We sum over all rules r that could expand part c, and let X_{crw} denote that window w contains a part of type c that is expanded by rule r:

$$= \sum_r \frac{P(I_w, r|X_{cw})}{P(I_w)}$$
$$= \sum_r \frac{P(r|X_{cw}) P(I_w|X_{cw}, r)}{P(I_w)}$$
$$= \sum_r P(r|c) \frac{P(I_w|X_{crw})}{P(I_w)}$$

By choosing a rule r, we have hypothesized the existence of a set of child parts, so we must consider their unknown geometry; let $\mathbf{v}$ be a vector specifying the regions they occupy:

$$= \sum_r P(r|c) \sum_{\mathbf{v}} \frac{P(I_w, \mathbf{v}|X_{crw})}{P(I_w)}$$
$$= \sum_r P(r|c) \frac{1}{P(I_w)} \sum_{\mathbf{v}} P(I_w|\mathbf{v}, X_{crw}) P(\mathbf{v}|X_{crw})$$

Partition I_w into I_{ν_k}, the pixels in child region ν_k, and $I_{w-\mathbf{v}}$, the pixels in w but not in any region ν_k:

$$= \sum_r P(r|c) \frac{1}{P(I_w)} \sum_{\mathbf{v}} P(\ldots, I_{\nu_k}, \ldots, I_{w-\mathbf{v}}|\mathbf{v}, X_{crw}) P(\mathbf{v}|X_{crw})$$

Assume conditional independence of the geometry of the child parts given the parent geometry (i.e., the assignment of child parts to image regions), and of child appearance given child geometry. Let d_{crk} denote the part model referred to by the kth rule part, and ϕ_{crk} be its geometry model:

$$= \sum_r P(r|c) \frac{1}{P(I_w)} \sum_{\mathbf{v}} P(I_{w-\mathbf{v}}) \prod_k P(I_{\nu_k}|\nu_k, X_{crw}) P(\nu_k|X_{crw})$$
$$= \sum_r P(r|c) \frac{1}{P(I_w)} \sum_{\mathbf{v}} P(I_{w-\mathbf{v}}) \prod_k P(I_{\nu_k}|X_{d_{crk}, \nu_k}) P(\nu_k|w; \phi_{crk})$$

Due to our independence assumptions, we have that $P(I_{w-\mathbf{v}}) = P(I_w)/\left(\prod_k P(I_{v_k})\right)$, so we can substitute and cancel:

$$= \sum_r P(r|c) \frac{1}{P(I_w)} \sum_{\mathbf{v}} \frac{P(I_w)}{\prod_k P(I_{v_k})} \prod_k P(I_{v_k}|X_{d_{crk}, v_k}) P(v_k|w; \phi_{crk})$$

$$= \sum_r P(r|c) \sum_{\mathbf{v}} \prod_k \frac{P(I_{v_k}|X_{d_{crk}, v_k})}{P(I_{v_k})} P(v_k|w; \phi_{crk})$$

Finally, the sum over $\mathbf{v}$ is actually a set of sums over each v_k, and each term is constant from the perspective of all sums except v_k itself. So we can take the sum over regions v for each rule part k separately and then multiply the results over all k:

$$= \sum_r P(r|c) \prod_k \sum_v \frac{P(I_v|X_{d_{crk}, v})}{P(I_v)} P(v|w; \phi_{crk})$$

Our end result is a recursive expression for the likelihood ratio:

$$\boxed{\beta(c, w) = \sum_r P(r|c) \prod_k \sum_v \beta(d_{crk}, v) P(v|w; \phi_{crk})} \quad (1)$$

This expression has a satisfying structure:

- $P(r|c)$ is the rule probability γ_{cr};
- $\beta(d_{crk}, v)$ is the recursive likelihood ratio for the kth child part; and
- $P(v|w; \phi_{crk})$ is the likelihood of the geometry of v conditioned on the attributes of w.

The base case of the β function occurs when c is primitive, in which case $\beta(c, w)$ is defined directly by the appearance model for part c.

4.2 A Dynamic Programming Algorithm

Equation 1 leads to a top-down algorithm that recursively calculates a score for each region and part model, caching intermediate results. Figure 4 presents the algorithm in detail. The algorithm has complexity $O(|G||I|^2)$, where $|G|$ is the number of part models, rules, and rule parts in the grammar, and $|I|$ is the number of image regions. To see this, we note that we continue past line 1 in the subroutine GET-LIKELIHOOD-RATIO only once per pair of part model c and image region w. Then, for each time we continue past line 1, we consider each rule r in c, rule part k in r, and candidate child region v exactly once. Finally, at the top level we call GET-LIKELIHOOD-RATIO exactly once per root part model c and image region w.

Although the squared term in this analysis may seem a bit daunting at first, we can actually limit the sum over child regions v to those near its expected location given the current parent region w. In this paper, we use three standard deviations around the mean location $\boldsymbol{\mu}_{crk}$. In practice, this heuristic greatly reduces the effect of the squared term, especially in cases in which the variance of the conditional geometry distributions is low, without significantly compromising the quality of recognition.

It is crucial that we not approximate the sums over r and v in Equation 1 with maxes, although this would allow us to use the distance transform as Felzenszwalb and Huttenlocher do. Such an approximation would result in the scoring of individual parse trees, and we cannot compare the scores of two parse trees that have different numbers of parts or edges because an unequal number of terms is contributing to the likelihood function in each case. Thus, to control for the structural difference among trees, we sum them out entirely. A helpful analogy to keep in mind is that of Bayes net structure learning, in which it is necessary to integrate out the parameters of the model in order to compare two structures which may have a different number of edges.

```
INPUT:      A PGG model G with root part models (..., c, ...), and an image I.
OUTPUT:     The most likely object class c* and its location w* in I.
1.          Initialize table β with an entry for each part model c in G and each region w in I.
2.          Initialize s* = 0.
3.          For each root part model c in G:
4.              For each region w in I:
5.                  Let s = GET-LIKELIHOOD-RATIO(c, w, β).
6.                  If s > s*, let s* = s, c* = c, and w* = w.
7.          Return ⟨c*, w*⟩.

subroutine GET-LIKELIHOOD-RATIO
INPUT:      A part model c in PGG G, a region w in image I, and a partially-filled table β.
OUTPUT:     The value of the image likelihood ratio for part model c and region w.
1.          If β contains an entry for c and w, return β[c, w].
2.          Initialize s1 = 0.
3.          For each rule r in part model c,
4.              Initialize s2 = γ_cr.
5.              For each rule part k in rule r,
6.                  Initialize s3 = 0.
7.                  For each window v in I that is close to μ_crk,
8.                      Let g = P(v|w; φ_crk).
9.                      Let b = GET-LIKELIHOOD-RATIO(d_crk, v, β).
10.                     Let s3 = s3 + (g × b).
11.                 Let s2 = s2 × s3.
12.             Let s1 = s1 + s2.
13.         Store and return β[c, w] = s1.
```

Figure 4: An efficient dynamic programming recognition algorithm.

In summary, we have presented a dynamic programming algorithm for performing recognition with a PGG model. Three key properties of the model and problem formulation contribute to the efficiency of this algorithm:

- the representation of an image as a discrete set of locations or regions, which allows us to build a dynamic programming table;
- the assumption of conditional independence among child parts given a fixed parent, which allows us to avoid summing over an exponential number of assignments of rule parts to sets of candidate child regions; and,
- the natural constraints on the search over child regions provided by the conditional geometry models.

5 Parameter Learning in PGGs

We have now presented the PGG framework and described how one might perform recognition using it. However, we have not yet addressed how we might learn PGG models; that is the topic of the next two sections. In this section, we assume a fixed grammar structure and develop an EM algorithm to estimate its parameters from data, extending the standard inside-outside algorithm for PCFGs. In the next section, we will discuss a search-based approach to the structure learning problem itself.

We have a set of training images $\{I^i | i = 1 \dots N\}$ labeled with root bounding boxes u^i and root object classes ρ^i. Let I^i_w be the image pixels in region w of the ith training image. The internal tree structure and geometry of each object is not labeled. The parameters Θ we will estimate are the rule probabilities γ_{cr} and

the geometry model parameters $(\boldsymbol{\mu}_{crk}, \boldsymbol{\Sigma}_{crk})$. In this paper, we will not address learning the appearance models A_c, assuming a fixed vocabulary of primitive part detectors.

Our EM algorithm extends the standard inside-outside algorithm for parameter learning in PCFGs (see, for example, the treatment by Manning and Schütze (2002)). On each iteration, we need to estimate the parameters Θ' given the parameters Θ from the previous iteration. In the E-step, we calculate responsibilities for each grammar component in each region of each training image. Then, in the M-step, we reestimate the parameters from the data, weighted by the responsibilities.

5.1 E-step

We need to calculate the likelihood of the hidden variables X_{cw}, X_{crw}, and $X_{d_{crk}v}$; these responsibilities will be used to reestimate the parameters. First, however, we need to define the notions of inside and outside probability ratios.

5.1.1 Inside and Outside Probability Ratios

In the inside-outside algorithm for PCFGs, the *inside probability* of a substring s_{ij} and a nonterminal N is defined as $P(s_{ij}|N_{ij})$, the likelihood that the substring from position i to j was generated by the nonterminal N, summing out all possible parse trees. The analogous quantity in our context is $P(I_w|X_{cw})$, the likelihood that the pixels in window w were generated by part model c. But because the PGG framework actually models the image likelihood ratio $P(I_w|X_{cw})/P(I_w)$, we shall use the notion of the inside probability *ratio* $\beta(c, w)$ instead, which we derived recursively in Equation 1:

$$\beta(c, w) = \frac{P(I_w|X_{cw})}{P(I_w)} = \sum_r \gamma_{cr} \prod_k \sum_v \beta(d_{crk}, v) P(v|w; \phi_{crk})$$

In the inside-outside algorithm for PCFGs, the *outside probability* of a substring s_{ij} and nonterminal N is defined as $P(s_{1,i-1}, N_{ij}, s_{j+1,m})$, the total likelihood of seeing the symbols $s_{1,i-1}$ and $s_{j+1,m}$ that are on either side of the substring and the nonterminal N covering the substring s_{ij}. The analogous quantity for us is $P(I_{\overline{w}}, X_{cw})$, the likelihood of seeing the pixels $I_{\overline{w}}$ *outside* the window w and the part model c occupying the window w. But, as above, we will work with the outside probability ratio instead:

$$\alpha(c, w) = \frac{P(I_{\overline{w}}, X_{cw})}{P(I_{\overline{w}})}$$

We sum over all parent identities and windows that could contain the part of interest c in window w, and then partition $I_{\overline{w}}$ into $I_{\overline{w}'}$, the features outside the parent w', and $I_{w'-w}$, those inside w' but outside w:

$$= \sum_{c',w'} \frac{P(I_{\overline{w}'}, I_{w'-w}, X_{cw}, X_{c'w'})}{P(I_{\overline{w}'}, I_{w'-w})}$$
$$= \sum_{c',w'} \frac{P(I_{\overline{w}'}, X_{c'w'})}{P(I_{\overline{w}'})} \frac{P(I_{w'-w}, X_{cw}|X_{c'w'})}{P(I_{w'-w})}$$

Now we sum over all the possible rules r' that could expand the parent c':

$$= \sum_{c',w'} \frac{P(I_{\overline{w}'}, X_{c'w'})}{P(I_{\overline{w}'})} \sum_{r'} P(r'|c') \frac{P(I_{w'-w}, X_{cw}|X_{c'r'w'})}{P(I_{w'-w})}$$

Then we consider the rule parts k in r' that could have produced our part of interest. Only c', r', and k such that $d_{c'r'k} = c$ are considered as candidate parents:

$$= \sum_{c',w'} \frac{P(I_{\overline{w}'}, X_{c'w'})}{P(I_{\overline{w}'})} \sum_{r'} P(r'|c') \sum_{\substack{k \text{ s.t.} \\ d_{c'r'k}=c}} P(w|w'; \phi_{c'r'k}) \frac{P(I_{w'-w}|X_{cw}, X_{c'r'w'})}{P(I_{w'-w})}$$

Finally we expand the sibling parts:

$$= \sum_{c',w'} \frac{P(I_{\overline{w}'}, X_{c'w'})}{P(I_{\overline{w}'})} \sum_{r'} P(r'|c') \sum_{\substack{k \text{ s.t.} \\ d_{c'r'k}=c}} P(w|w';\phi_{c'r'k}) \prod_{k' \neq k} \sum_{v} \frac{P(I_v|X_{d_{c'r'k'},v})}{P(I_v)} P(v|w';\phi_{c'r'k'})$$

Our end result is a recursive expression for the outside probability ratio:

$$\boxed{\alpha(c,w) = \sum_{c',w'} \alpha(c',w') \sum_{r'} \gamma_{c'r'} \sum_{\substack{k \text{ s.t.} \\ d_{c'r'k}=c}} P(w|w';\phi_{c'r'k}) \prod_{k' \neq k} \sum_{v} \beta(d_{c'r'k'},v) P(v|w';\phi_{c'r'k'})} \quad (2)$$

The base case of the α function occurs when c is the labeled object class and w is the labeled bounding box, in which case $\alpha(c,w) = 1.0$.

5.1.2 Calculating the Responsibilities

We need to calculate the likelihood of the hidden variables X_{cw}, X_{crw}, and $X_{d_{crk}v}$ given the parameters Θ from the previous iteration and the observed data. The challenge here is that we have access only to the ratios defined by the functions α and β. We cannot isolate the numerator of these ratios by multiplying them by the denominators $P(I^i_w)$ or $P(I^i_{\overline{w}})$, because we do not have a direct representation for these quantities in our model. Thus we need to define the responsibilities while using these ratios as atomic units.

Let α_i and β_i be the inside and outside probability ratios applied to the ith training image. Let $X_{\rho^i u^i}$ denote that the labeled object class ρ^i occupies the labeled bounding box u^i in image I^i. Then these are the responsibilities we need to normalize the rule probabilities γ_{cr}:

$$\begin{aligned} f_i(c,w) &= P(X_{cw}|I^i, X_{\rho^i u^i}) \\ &= P(I^i) \frac{P(X_{cw}|I^i, X_{\rho^i u^i})}{P(I^i)} \\ &= \frac{P(I^i)}{P(I^i|X_{\rho^i u^i})} \frac{P(X_{cw}, I^i|X_{\rho^i u^i})}{P(I^i)} \end{aligned}$$

Partition I^i based on either the labeled bounding box u^i or the window w, and then assume conditional independence of foreground and background features given the partition:

$$\begin{aligned} &= \frac{P(I^i_{u^i}, I^i_{\overline{u}^i})}{P(I^i_{u^i}, I^i_{\overline{u}^i}|X_{\rho^i u^i})} \frac{P(I^i_w, I^i_{\overline{w}}, X_{cw}|X_{\rho^i u^i})}{P(I^i_w, I^i_{\overline{w}})} \\ &= \frac{P(I^i_{u^i})P(I^i_{\overline{u}^i})}{P(I^i_{u^i}|X_{\rho^i u^i})P(I^i_{\overline{u}^i})} \frac{P(I^i_{\overline{w}}, X_{cw})}{P(I^i_{\overline{w}})} \frac{P(I^i_w|X_{cw})}{P(I^i_w)} \\ &= \frac{\alpha_i(c,w)\beta_i(c,w)}{\beta_i(\rho^i, u^i)} \end{aligned}$$

Similarly, these are the responsibilities for the rule probabilities γ_{cr}:

$$\begin{aligned} g_i(c,r,w) &= P(X_{crw}|I^i, X_{\rho^i u^i}) \\ &= P(I^i) \frac{P(X_{crw}|I^i, X_{\rho^i u^i})}{p(I^i)} \\ &= \frac{P(I^i)}{P(I^i|X_{\rho^i u^i})} \frac{P(X_{crw}, I^i|X_{\rho^i u^i})}{P(I^i)} \end{aligned}$$

Again, partition I^i based on either the labeled bounding box u^i or the window w, and then assume conditional independence of foreground and background features given the partition:

$$= \frac{P(I^i_{u^i}, I^i_{\overline{u}^i})}{P(I^i_{u^i}, I^i_{\overline{u}^i} | X_{\rho^i u^i})} \frac{P(I^i_w, I^i_{\overline{w}}, X_{crw} | X_{\rho^i u^i})}{P(I^i_w, P^i_{\overline{w}})}$$
$$= \frac{P(I^i_{u^i}) P(I^i_{\overline{u}^i})}{P(I^i_{u^i} | X_{\rho^i u^i}) P(I^i_{\overline{u}^i})} \frac{P(I^i_{\overline{w}}, X_{cw})}{P(I^i_{\overline{w}})} P(r|c) \frac{P(I^i_w | X_{crw})}{P(I^i_w)}$$

Then, exploit conditional independence among the child parts, and simplify:

$$= \frac{P(I^i_{u^i})}{P(I^i_{u^i} | X_{\rho^i u^i})} \frac{P(I^i_{\overline{w}}, X_{cw})}{P(I^i_{\overline{w}})} P(r|c) \prod_k \sum_\nu \frac{P(I^i_\nu | X_{d_{crk}, \nu})}{P(I^i_\nu)} P(\nu | w; \phi_{crk})$$
$$= \frac{1}{\beta_i(\rho^i, u^i)} \alpha_i(c, w) \gamma_{cr} \prod_k \sum_\nu \beta_i(d_{crk}, \nu) P(\nu | w; \phi_{crk})$$

Finally, we derive the responsibilities for the geometry parameters $\boldsymbol{\mu}_{crk}, \boldsymbol{\Sigma}_{crk}$, using analogous steps:

$$\begin{aligned}
h_i(c, r, k, w, \nu) &= P(X_{crw}, X_{d_{crk}, \nu} | I^i, X_{\rho^i u^i}) \\
&= P(I^i) \frac{P(X_{crw}, X_{d_{crk}, \nu} | I^i, X_{\rho^i u^i})}{P(I^i)} \\
&= \frac{P(I^i)}{P(I^i | X_{\rho^i u^i})} \frac{P(X_{crw}, X_{d_{crk}, \nu}, I^i | X_{\rho^i u^i})}{P(I^i)} \\
&= \frac{P(I^i_{u^i}, I^i_{\overline{u}^i})}{P(I^i_{u^i}, I^i_{\overline{u}^i} | X_{\rho^i u^i})} \frac{P(I^i_w, I^i_{\overline{w}}, X_{crw} | X_{\rho^i u^i})}{P(I^i_w, P^i_{\overline{w}})} \\
&= \frac{P(I^i_{u^i}) P(I^i_{\overline{u}^i})}{P(I^i_{u^i} | X_{\rho^i u^i}) P(I^i_{\overline{u}^i})} \frac{P(I^i_{\overline{w}}, X_{cw})}{P(I^i_{\overline{w}})} P(r|c) \frac{P(I^i_w, X_{d_{crk}, \nu} | X_{crw})}{P(I^i_w)} \\
&= \frac{P(I^i_{u^i})}{P(I^i_{u^i} | X_{\rho^i u^i})} \frac{P(I^i_{\overline{w}}, X_{cw})}{P(I^i_{\overline{w}})} P(r|c) \frac{P(I^i_\nu | X_{d_{crk}, \nu})}{P(I^i_\nu)} P(\nu | w; \phi_{crk}) \prod_{k' \neq k} \sum_{\nu'} \frac{P(I^i_{\nu'} | X_{d_{crk'}, \nu'})}{P(I^i_{\nu'})} P(\nu' | w; \phi_{crk'}) \\
&= \frac{1}{\beta_i(\rho^i, u^i)} \alpha_i(c, w) \gamma_{cr} \beta_i(d_{crk}, \nu) P(\nu | w; \phi_{crk}) \prod_{k' \neq k} \sum_{\nu'} \beta_i(d_{crk'}, \nu') P(\nu' | w; \phi_{crk'})
\end{aligned}$$

5.2 M-step

Now we can reestimate the parameters Θ' for the next iteration, using the responsibilities as weights:

$$\gamma'_{cr} = \frac{\sum_i \sum_w g_i(c, r, w)}{\sum_i \sum_w f_i(c, w)}$$
$$\boldsymbol{\mu}'_{crk} = \frac{\sum_i \sum_w \sum_\nu h_i(c, r, k, w, \nu) T(\nu, w)}{\sum_i \sum_w \sum_\nu h_i(c, r, k, w, \nu)}$$
$$\boldsymbol{\Sigma}'_{crk} = \frac{\sum_i \sum_w \sum_\nu h_i(c, r, k, w, \nu) (T(\nu, w) - \boldsymbol{\mu}'_{crk})^2}{\sum_i \sum_w \sum_\nu h_i(c, r, k, w, \nu)}$$

where $T(\nu, w)$ transforms the child region ν by subtracting the centroid of the parent region w, putting it in w's coordinate frame.

Given initial parameters Θ_0, we iterate the E- and M-steps until the difference in the log likelihood scores of the data under Θ and Θ' is no greater than 0.01 of the log likelihood score under Θ.

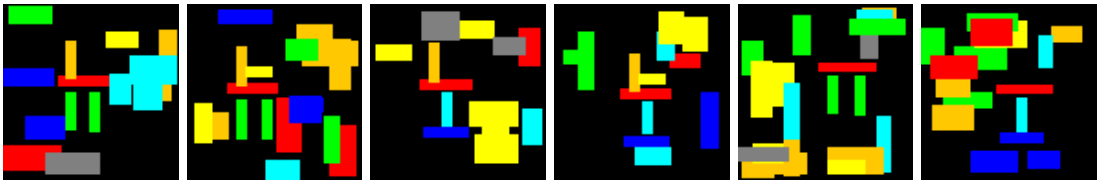


Figure 5: A very simple cartoon furniture domain. Each primitive part is associated with a single color: seat (red), back (orange), arm (yellow), leg (green), axle (cyan), foot (blue). From left to right, the ground object classes include: two-legged chair, two-legged chair with arms, footed chair, footed chair with arms, two-legged table, and footed table.

5.3 Implementation

When we are parsing a single test image, we can afford to store a dynamic programming table for β that records the quality of each part model c at each region w. However, to precompute the responsibilities during the E-step of EM, we would have to store a separate table for both the inside and outside probabilities for every image we are learning from. Furthermore, under a naïve implementation, we would need more than just a value for each c and w; we would also need to store a value for each subregion v of each region w of each image I^i according to each rule part of each rule—the functions $h_i(c, r, k, w, v)$. For large numbers of training images, this would probably require an infeasible amount of memory.

Instead, we take an approach in which we blend the E- and M-steps, computing the responsibilities on the fly as we compute running sums for the numerator and denominator of each parameter. This lets us consider the training images one at a time. We also need to take two passes over the training images at each EM iteration, since the covariance estimation depends on the already-estimated mean values.

6 Structure Learning in PGGs

Structure learning aims to find a compact grammar that explains the training data. By targeting compactness, we encourage sharing of parts and substructure among object classes. In this section, we describe a search-based optimization approach to structure learning. Throughout the section, we will use the simple cartoon furniture domain shown in Figure 5 as a running example.

6.1 Search Initialization

In this paper, we assume a pre-specified set of primitive parts, with appearance models. We also assume the labeled initial locations of a set of primitive parts making up each training object, although these are free to change during EM. The initial locations need not be terribly precise; we will improve them through successive rounds of EM. An example of training images labeled with initial primitive part guesses for our toy domain is shown in Figure 6(a).

Our search operators will focus on building up hierarchy and structure, so a natural way to initialize the structure learning algorithm is with a flat grammar. For each unique pattern of labeled primitive parts in the training data, we write down a rule with the labeled object class on the left side of the rule and the primitive parts on the right side. See an example of an initial grammar for the toy domain in Figure 6(b).

To initialize the geometry models ϕ_{crk}, we estimate the mean and variance of the primitive part positions relative to the bounding boxes' centroids, across training images with the same object class and set of primitive labels. Geometry models in rules to which only a single training image contributed are assigned a standard small prior variance.

(a) Training images labeled with initial primitive parts.

chair:
- 0.25 chair → arm (ϕ_{000}) axle (ϕ_{001}) back (ϕ_{002}) foot (ϕ_{003}) foot (ϕ_{004}) seat (ϕ_{005})
- 0.25 chair → arm (ϕ_{010}) back (ϕ_{011}) leg (ϕ_{012}) leg (ϕ_{013}) seat (ϕ_{014})
- 0.25 chair → axle (ϕ_{020}) back (ϕ_{021}) foot (ϕ_{022}) foot (ϕ_{023}) seat (ϕ_{024})
- 0.25 chair → back (ϕ_{030}) leg (ϕ_{031}) leg (ϕ_{032}) seat (ϕ_{033})

table:
- 0.5 table → axle (ϕ_{100}) foot (ϕ_{101}) foot (ϕ_{102}) seat (ϕ_{103})
- 0.5 table → leg (ϕ_{110}) leg (ϕ_{111}) seat (ϕ_{112})

arm (*yellow*)
axle (*cyan*)
back (*orange*)
foot (*blue*)
leg (*green*)
seat (*red*)

(b) The initial flat grammar.

Figure 6: An initial grammar for the toy furniture domain, with a single flat rule for each object class and unique pattern of primitive parts.

6.2 Structure Search Operators

As in other approaches to grammar learning, our search operators move the algorithm through the space of candidate grammars by proposing changes to the current grammar. First, we define the four types of operators, and then explain in greater detail how we manipulate the geometry models during the search.

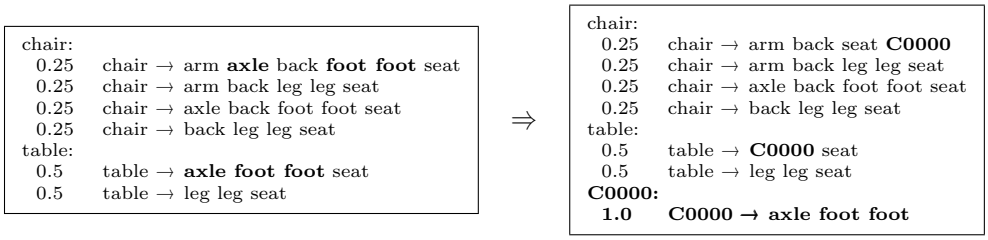
Create a new AND composite part. The role of this operator is to recognize common subsets of rule parts, and create new composite parts to stand for these patterns. A new AND part C_{and} may be proposed whenever a pattern of rule parts ξ with size no greater than n_{and} occurs on the right side of at least two rules. (In this paper, we use $n_{and} = 3$.) Symbolic search on the grammar rules is used to propose and replace instances of ξ that meet this criteria; importantly, not all instances of ξ must be affected. For example:

$$\begin{array}{l} C1 \rightarrow X1\ \mathbf{X2\ X3}\ X4 \\ C2 \rightarrow \mathbf{X2\ X3}\ X5 \\ C3 \rightarrow X2\ X3\ X6 \end{array} \quad \Rightarrow \quad \begin{array}{l} C1 \rightarrow X1\ \mathbf{C_{and}}\ X4 \\ C2 \rightarrow \mathbf{C_{and}}\ X5 \\ C3 \rightarrow X2\ X3\ X6 \\ \mathbf{C_{and} \rightarrow X2\ X3} \end{array}$$

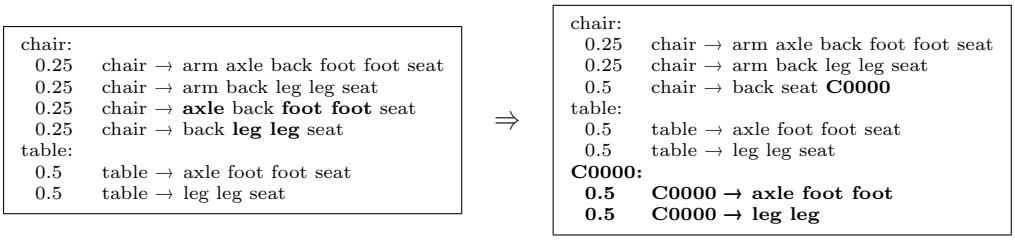
Notice that the rule for C3 was not affected by the operation, despite the fact that an instance of the pattern X2 X3 occurred within it. This may be because the geometry models of that instance did not match those of the other two instances well enough (we will describe the matching of geometry models more below). Or it might be that that pattern will be replaced by C_{and} in a later search step.

The initial geometry parameters for C_{and} are a weighted average of the transformed parameters of the instances of the pattern ξ that contributed to its creation. (We will discuss the transforming and averaging of geometry models below.) The initial mean for each replaced instance of C_{and} in the old rules is the centroid of the instance of ξ which was replaced; the initial variance parameter in each dimension is the average of the variances of the component parts that were replaced. An example of a create-AND operation in the toy furniture domain is given in Figure 7(a).

Create a new OR composite part. This operator plays the opposite role: it notices *differences* among sets of rules, and creates composite parts to more compactly express those differences. A new OR composite



(a) A create-AND operation.



(b) A create-OR operation.

Figure 7: Examples of part model creation operations in the toy furniture domain.

part C_{or} may be proposed whenever at least two rules would become identical were a pair or small subset of part models $(X1, X2, ...)$, to be renamed to C_{or} in the context of those rules. We search for sets of symbols that are in common among the rules with size no greater than n_{or}, *or* sets that are different among the rules with size no greater than n_{or}. (In this paper, we use $n_{or} = 3$.) Again, not all instances of the pattern must be affected.

$$\begin{array}{l} C1 \rightarrow X1\ X2\ \mathbf{X3} \\ C1 \rightarrow X1\ X2\ \mathbf{X4} \\ C1 \rightarrow X1\ X2\ X3 \end{array} \quad \Rightarrow \quad \begin{array}{l} C1 \rightarrow X1\ X2\ \mathbf{C_{or}} \\ C1 \rightarrow X1\ X2\ X3 \\ \mathbf{C_{or} \rightarrow X3} \\ \mathbf{C_{or} \rightarrow X4} \end{array}$$

Notice that the third rule for C1 was not affected by the operation, despite the fact that it would have been symbolically identical had X3 been renamed to C_{or} within it. Again, this could be because the geometry models of that rule did not match those of the other two rules well enough, or that replacement might occur at a later search step. Note that this operator may be used to propose new OR part models in which one rule is entirely empty, expressing compactly the notion of an optional set of parts, such as the arms on a chair.

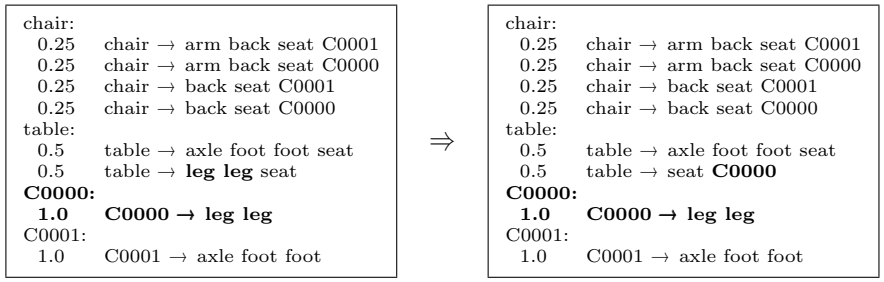
The initial geometry parameters of the merged rule are again an average of those of the contributing rules, weighted by their rule probabilities. The initial probabilities on the new rules for C_{or} are a renormalized version of the contributing rule probabilities. An example of a create-OR operation in the toy furniture domain is given in Figure 7(b).

Apply an existing AND composite part. As we said, the creation operators we just defined need not be applied immediately to all applicable rules. Thus, we have operators to apply existing composite part models rather than creating new ones.

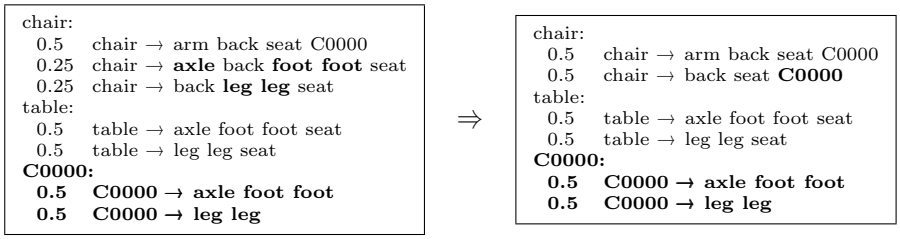
An existing AND part C_{and} with a single rule $C_{and} \rightarrow \xi$ may be applied whenever the pattern ξ occurs on the right side of at least one other rule.

$$\begin{array}{l} C1 \rightarrow \mathbf{X1\ X2}\ X3 \\ \mathbf{C_{and} \rightarrow X1\ X2} \end{array} \quad \Rightarrow \quad \begin{array}{l} C1 \rightarrow \mathbf{C_{and}}\ X3 \\ \mathbf{C_{and} \rightarrow X1\ X2} \end{array}$$

The geometry parameters of the rule $C_{and} \rightarrow \xi$ remain unchanged. As above, the initial geometry parameters for each new instance of C_{and} in the rules should be informed by the bounding box of the instance of ξ which



(a) A apply-AND operation.



(b) A apply-OR operation.

Figure 8: Examples of part model application operations in the toy furniture domain.

were replaced. This specifies the initial mean; the initial variance parameter in each dimensions should be the average of the variances of the component parts that were replaced. An example of an apply-AND operation in the toy furniture domain is given in Figure 8(a).

Apply an existing OR composite part. An existing OR composite part C_{or} with rule patterns $(\xi_1, \xi_2, ...)$ may be applied whenever at least two rules would become identical were the instances of the patterns $(\xi_1, \xi_2, ...)$ in those rules renamed to C_{or}.

$$\begin{array}{l} \text{C2} \rightarrow \text{X1 X2 } \mathbf{X3} \\ \text{C2} \rightarrow \text{X1 X2 } \mathbf{X4} \\ \text{C2} \rightarrow \text{X1 X2 X3} \\ \mathbf{C_{or} \rightarrow X3} \\ \mathbf{C_{or} \rightarrow X4} \end{array} \quad \Rightarrow \quad \begin{array}{l} \text{C2} \rightarrow \text{X1 X2 } \mathbf{C_{or}} \\ \text{C2} \rightarrow \text{X1 X2 X3} \\ \mathbf{C_{or} \rightarrow X3} \\ \mathbf{C_{or} \rightarrow X4} \end{array}$$

In this example, again notice that the third rule for C2 was not affected by the operation.

The initial geometry parameters of the merged rule are estimated in the same way as when merging rules in the "create-OR" operation, and the parameters on the rules for C_{or} remain unchanged. An example of an apply-OR operation in the toy furniture domain is given in Figure 8(b).

6.2.1 Manipulating Geometry Models

As we foreshadowed above, when rule parts are merged we transform and average their geometry models in order to initialize the parameters for the new part. We must transform the geometry models to be relative to a new local parent (we use the centroid of the selected parts), so that the models will be invariant to their positions in the context of their original rule.

Then, we need to average the geometry models for the set of rule parts, weighted by the probabilities on their original rules. Formally, we have a set of Gaussians with parameters (μ_i, σ_i^2) for the ith Gaussian, and a weight γ_i on each such that $\sum_i \gamma_i = 1$. (Because our covariance matrices are diagonal, we can do this analysis using univariate Gaussians.) We want the parameters (μ, σ^2) that would have resulted from estimating them directly from the datasets D_i that produced the contributing Gaussians. But since we do

not have access to the original datasets themselves,[1] we need expressions for the new parameters that are in terms of the parameters of the component Gaussians.

Writing an expression for the mean is straightforward. Let x_{ij} be the jth data point in the ith dataset D_i. Each μ_i is the sample mean $\mu_i = \left(\sum_j x_{ij}\right)/|D_i|$, and each weight γ_i is a simple normalized count $\gamma_i = |D_i|/\left(\sum_{i'}|D_{i'}|\right)$. We want the sample mean of the combined datasets:

$$\begin{aligned}\mu &= \frac{1}{\sum_{i'}|D_{i'}|}\sum_i\sum_j x_{ij} \\ &= \frac{1}{\sum_{i'}|D_{i'}|}\sum_i |D_i|\left(\frac{1}{|D_i|}\sum_j x_{ij}\right) \\ &= \sum_i \gamma_i \mu_i \end{aligned} \tag{3}$$

The variance is a bit trickier. Each σ_i^2 is the sample variance $\sigma_i^2 = \left(\sum_j (x_{ij}-\mu_i)^2\right)/|D_i|$. We want the sample variance of the combined datasets:

$$\begin{aligned}\sigma^2 &= \frac{1}{\sum_{i'}|D_{i'}|}\sum_i\sum_j (x_{ij}-\mu)^2 \\ &= \frac{1}{\sum_{i'}|D_{i'}|}\sum_i |D_i|\left(\frac{1}{|D_i|}\sum_j (x_{ij}-\mu)^2\right) \end{aligned} \tag{4}$$

Now examine just the inner part of this expression. Expand the square, introduce the individual means μ_i, and then rearrange, distribute, cancel, and simplify:

$$\begin{aligned}\frac{1}{|D_i|}\sum_j (x_{ij}-\mu)^2 &= \frac{1}{|D_i|}\sum_j (x_{ij}^2 - 2x_{ij}\mu + \mu^2) \\ &= \frac{1}{|D_i|}\sum_j (x_{ij}^2 - 2x_{ij}\mu + \mu^2) - 2x_{ij}\mu_i + 2x_{ij}\mu_i + \mu_i^2 - \mu_i^2 \\ &= \frac{1}{|D_i|}\sum_j (x_{ij}^2 - 2x_{ij}\mu_i + \mu_i^2) - 2x_{ij}\mu + \mu^2 + 2x_{ij}\mu_i - \mu_i^2 \\ &= \left(\frac{1}{|D_i|}\sum_j (x_{ij}-\mu_i)^2\right) - \left(2\mu\frac{1}{|D_i|}\sum_j x_{ij}\right) + \left(\frac{1}{|D_i|}|D_i|\mu^2\right) + \left(2\mu_i\frac{1}{|D_i|}\sum_j x_{ij}\right) - \left(\frac{1}{|D_i|}|D_i|\mu_i^2\right) \\ &= \sigma_i^2 - 2\mu\mu_i + \mu^2 + 2\mu_i^2 - \mu_i^2 \\ &= \sigma_i^2 + (\mu_i-\mu)^2 \end{aligned}$$

Finally, substitute this result back into Equation 4:

$$\begin{aligned}\sigma^2 &= \frac{1}{\sum_{i'}|D_{i'}|}\sum_i |D_i|\left(\sigma_i^2 + (\mu_i-\mu)^2\right) \\ &= \sum_i \gamma_i\left(\sigma_i^2 + (\mu_i-\mu)^2\right) \end{aligned} \tag{5}$$

[1] It is important that we reason directly about the geometry model parameters, rather than about the most likely object part locations in the training images. After the initialization phase, there may not be a one-to-one mapping between the geometry models and locations in the image, since the parameters will have been estimated during EM by summing, rather than maxing, over candidate locations in the images. By reasoning about the geometry model parameters directly, we avoid this problem.

This expression is satisfying in that it incorporates both the individual variances σ_i^2 and the variances among the means $(\mu_i - \mu)^2$.

We use the expressions in Equations 3 and 5 to initialize the parameters of the geometry models of new rule parts in a principled way, based on the parameters of the merging rule parts. Then, we run EM on the new grammar structure to fit the parameters to the data better. But because EM is notoriously sensitive to initialization, it is important that we are able to choose reasonable initial values for these parameters.

6.3 Structure Score and Search Control

The structure score evaluates a structure G given image data $D = \{I^i | i = 1 \dots N\}$. The score we use is a combination of the quality of the training data under the model and a penalty on the model's complexity:

$$\text{score}(G; D) = (1 - \lambda)\ell(D; G) - \lambda \frac{\log N}{2} \dim(G) \tag{6}$$

where

$$\ell(D; G) = \sum_i \log \frac{P(I^i_{u^i} | X_{\rho^i u^i}; G)}{P(I^i_{u^i}; G)}$$

is the log likelihood ratio of the training data given the G, $\dim(G)$ is the number of parameters in G, and λ trades off between the likelihood and the model complexity.[2] We use $\lambda = 0.25$ (determined empirically).

The branching factor for this search problem is quite high. However, we can apply an insight about our structure score: a grammar's score before EM has been run is a lower bound on its score after EM. We can exploit this property to find a proposal that is guaranteed to be an uphill step, rather than searching exhaustively for the maximal gradient proposal. Specifically, rather than running EM on each candidate grammar and choosing the operation with the best post-EM score, we can rank the proposals according to their pre-EM scores, run EM on the grammars in ranked order, and accept the first proposal whose post-EM score improves on the current score.

Furthermore, we can use the guiding principle of compactness to inspire several greedy search heuristics, which will bias us towards desirable structures while controlling the branching factor.

6.3.1 Encouraging Compact Structure

Because we want to encourage sharing of parts and structure, we always consider apply-AND and apply-OR operations before create-AND and create-OR operations. In practice there are often many more ways to create new parts than to apply existing ones, so this heuristic can greatly speed up the learning process, as well as encouraging a more compact structure.

To encourage compactness, we collapse unnecessary hierarchy whenever possible. After applying an operator, we inspect the new grammar for part models that have one rule and are referred to only once in the rest of the grammar. If such a part model is found, it is "collapsed": its rule is inlined into the place where the part model was referenced, and the geometry models of the rule parts are composed by adding the mean of the geometry model of the rule part that is being replaced to the means of the geometry model of each rule part that is being inlined. This inspect-and-collapse loop continues until no more can occur.

In addition to simplifying the structure of the grammar, this heuristic plays an important role in mediating some of the limits imposed by the necessarily small values of n_{and} and n_{or}, by allowing chained operations that achieve the same end effect without overwhelming the branching factor of any individual search step. For example, we might take the following set of steps if $n_{\text{and}} = 2$:

start		*after step 1*		*after step 2*		*after collapsing*
C0 → X1 X2 X3 X4 C1 → X1 X2 X3 X5	⇒	C0 → C2 X3 X4 C1 → C2 X3 X5 C2 → X1 X2	⇒	C0 → C3 X4 C1 → C3 X5 C2 → X1 X2 C3 → C2 X3	⇒	C0 → C3 X4 C1 → C3 X5 C3 → X1 X2 X3

[2] This formula resembles the BIC score, but we are not using a Bayesian approach to construct our structure score.

6.3.2 Encouraging Low Variance Geometry Distributions

Although the number of proposals at each search step is usually large, many proposals will lower the overall structure score because they will be merging parts and rules that are not geometrically compatible. We want to avoid considering operations that involve merging symbolically identical but geometrically distinct parts, so that the OR-nodes of the grammatical structure can do most of the work of expressing uncertainty and choice (i.e., mixtures) in both geometry and type.

For example, imagine a grammar for a wrench that is closed on both ends (see Figure 10(a), second from the left), where each circular end is described by two primitive curved parts (round-bot and round-top) and the handle is described with two identical horizontal bar parts (dedge):

$$\text{wrench} \rightarrow \text{round-bot round-top dedge dedge round-bot round-top}$$

There are two symbolically-identical instances of the pattern "round-bot round-top dedge", so they might be considered good candidates for a new AND part model:

$$\begin{aligned}&\text{wrench} \rightarrow \text{C0 C0}\\&\text{C0} \rightarrow \text{round-top round-bot dedge}\end{aligned}$$

However, the geometry model for dedge in the new rule for C0 will have extremely high variance, because it must deal with both the case of the left end of the wrench, where the handle is far to the right of the circle, and the case of the right end, where the handle is far to the left of the circle. We would not expect a model like this to explain the data as well as, say, this model:

$$\begin{aligned}&\text{wrench} \rightarrow \text{C0 dedge dedge C0}\\&\text{C0} \rightarrow \text{round-top round-bot}\end{aligned}$$

in which the geometry models for all parts could be well localized.

With this in mind, we bias the search process towards geometrically plausible proposals by pruning proposals that result in high variance geometry models (in this paper, any model that has a standard deviation greater than 0.2 of the corresponding dimension—width or height—of the objects in the training data). This simple heuristic greatly speeds up the ranking and search process, while at the same time removing from consideration proposals which probably should never be accepted anyway.

As we described in Section 6.2, at each step there may be multiple lexicographically identical operations. For example, there are several ways of creating an AND part for a pattern that occurs three or more times, since not all instances of the pattern must be replaced. So, again to encourage low variance distributions, we only propose the single operation from the set of identical operations whose geometry models match best, and for efficiency we only consider *pairs* of rules or rule part subsets at any given time. In the toy furniture example, we would only consider creating an AND part for the instance of the "leg leg" pattern whose geometry models match best, despite the fact that there are four such instances in the initial grammar in Figure 6(b).

In order to compare the geometry for two sets of rule parts, we can impose a canonical ordering on parts. There may still be ambiguity about how to match up rule parts if there is more than one part of the same type in each of the rules (e.g., two chair legs), but because we expect the number of such parts to be quite small, we can enumerate all the possible ways to assign the rule parts in one rule to the other. Then, for each fixed assignment, we transform the geometry models to be relative to a new local parent, and use symmetrized KL divergence:

$$\begin{aligned}D_{SKL}(N_0, N_1) &= D_{KL}(N_0 \| N_1) + D_{KL}(N_1 \| N_0)\\&= \frac{1}{2}\left(\text{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) + \text{tr}(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)\right) - N\end{aligned}$$

as a distance metric to choose the operation that would merge parts with the best matching geometry models.

Figure 9 presents the full structure learning algorithm that we have developed in this section.

INPUT: A set of labeled training data D, a set of appearance models A, and a set of initial primitive parts for each image in D.
OUTPUT: A PGG model G that explains the data in D.
1. Initialize G to a flat grammar with a primitive part for each appearance model in A and a single rule for each object class and unique pattern of initial primitive parts in D.
2. Let $\mathsf{G} = \text{RUN-EM}(\mathsf{G}, \mathsf{D})$.
3. Let $\mathsf{s} = \text{STRUCTURE-SCORE}(\mathsf{G}, \mathsf{D})$.
4. While *true*:
5. Let $\langle \mathsf{G}', \mathsf{s}' \rangle = \text{TAKE-SEARCH-STEP}(\mathsf{D}, \mathsf{G}, \mathsf{s}, \{\textit{apply-and}, \textit{apply-or}\})$.
6. If $\langle \mathsf{G}', \mathsf{s}' \rangle ==$ null:
7. Let $\langle \mathsf{G}', \mathsf{s}' \rangle = \text{TAKE-SEARCH-STEP}(\mathsf{D}, \mathsf{G}, \mathsf{s}, \{\textit{create-and}, \textit{create-or}\})$.
8. If $\langle \mathsf{G}', \mathsf{s}' \rangle ==$ null, break.
9. Let $\langle \mathsf{G}, \mathsf{s} \rangle = \langle \mathsf{G}', \mathsf{s}' \rangle$.
10. Return G.

subroutine TAKE-SEARCH-STEP
INPUT: The training data D, the current grammar G_0 and score s_0, and a set of operator types T.
OUTPUT: The next grammar G and score s.
1. Initialize the set of candidate grammars $\mathsf{Q} = \emptyset$.
2. For each operator type t, enumerate possible operations of type $\mathsf{t} \in \mathsf{T}$ on G_0, and for each proposal add candidate grammar G_q to Q. Use distance metric on geometry models to choose among symbolically-identical proposals.
3. For each $\mathsf{G}_q \in \mathsf{Q}$, collapse any unnecessary rules in G_q.
4. For each $\mathsf{G}_q \in \mathsf{Q}$, if G_q has high variance geometry models, remove G_q from Q.
5. For each $\mathsf{G}_q \in \mathsf{Q}$, let $\mathsf{s}_q = \text{STRUCTURE-SCORE}(\mathsf{G}_q, \mathsf{D})$.
6. Rank the grammars in Q using their lower-bound scores s_q.
7. For each $\mathsf{G}_q \in \mathsf{Q}$, in ranked order:
8. Let $\mathsf{G}_q = \text{RUN-EM}(\mathsf{G}_q, \mathsf{D})$.
9. Let $\mathsf{s}_q = \text{STRUCTURE-SCORE}(\mathsf{G}_q, \mathsf{D})$.
10. If $\mathsf{s}_q > \mathsf{s}_0$, return $\langle \mathsf{G}_q, \mathsf{s}_q \rangle$.
11. Return null.

Figure 9: A structure learning algorithm for PGGs. The RUN-EM subroutine is defined in detail in Section 5, and the STRUCTURE-SCORE subroutine is fully defined by Equation 6.

7 Experimental Results

In this section, we describe a set of experiments to evaluate the effectiveness of the PGG framework.

7.1 The Wrench Domain

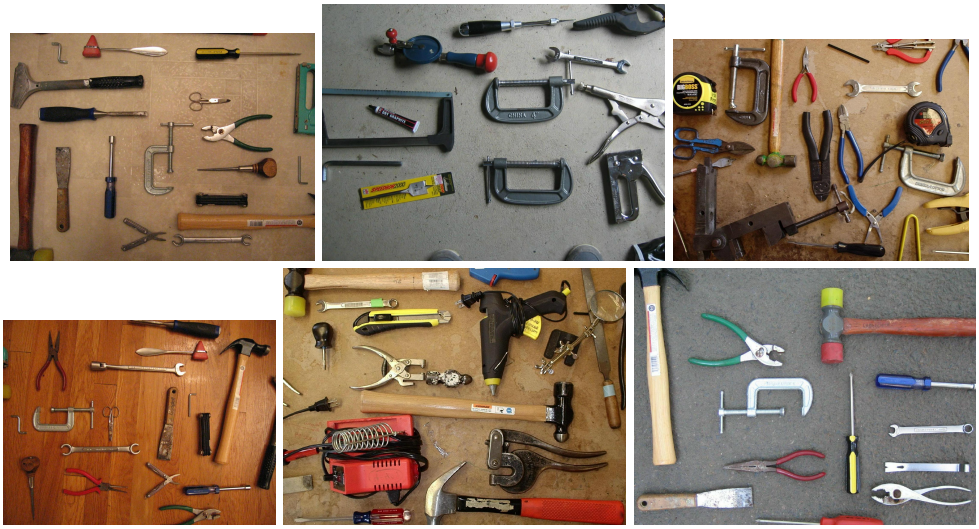
We have collected a wrench data set to use as a test bed for the framework. It consists of four ground classes of wrench: open on both ends, closed on both ends, open on the left and closed on the right, and closed on the left and open on the right (see Figure 10(a)). This domain has inherent "or" structure that makes it a natural place to start testing the PGG model.

We focus on a localization task: given a large complicated image that contains a single wrench but also many distractor items, the goal is to correctly localize the wrench. We are not yet modeling or searching over scale or orientation, so the images have been rotated and scaled so that the wrench is horizontal and of roughly uniform width, although there is some variation. Figure 10(b) shows example test images.

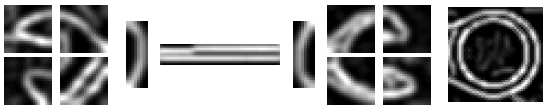
7.1.1 Edge Intensity Templates

Recall that each primitive part model c has an appearance model A_c which must define the image likelihood ratio $\mathsf{P}(\mathsf{I}_w|\mathsf{X}_{\mathsf{c}w})/\mathsf{P}(\mathsf{I}_w)$. We have a variety of choices of how to represent this quantity. For example, we

(a) The four ground wrench classes.



(b) Example test images.



(c) Hand-chosen edge intensity templates. From top to bottom and left to right: tip3, tip4, side3, side4, open-left, dedge, open-right, side2, side1, tip2, tip1, full-circle.

Figure 10: A wrench domain.

could define a generative pixel-wise edge template, following Crandall et al. (2005), or we could model region shape or curvature. Or we could define a distribution over SIFT descriptors of features extracted using one of the currently popular feature detectors; e.g., scale-invariant salient regions (Kadir & Brady, 2001) or maximally stable extremal regions (Matas et al., 2002).

For the experiments in the wrench domain described in this paper, we are controlling for the issue of learning the appearance models. Instead, we use hand-chosen edge intensity templates (shown in Figure 10(c)), and define the image likelihood ratio as follows. First, we apply Bayes rule and cancel:

$$\frac{P(I_w|X_{cw})}{P(I_w)} = \frac{P(X_{cw}|I_w)}{P(X_{cw})}$$

Next, we sum out all pixel values I'_w that could exist in window w, such that we have $P(X_{cw}) = \sum_{I'_w} P(I'_w)P(X_{cw}|I'_w)$:

$$= \frac{P(X_{cw}|I_w)}{\sum_{I'_w} P(I'_w)P(X_{cw}|I'_w)}$$

Now, we assume that the conditional likelihood $P(X_{cw}|I_w)$ is proportional to an exponentiated function f of the (unnormalized) correlation coefficient between the template T_c for class c and the pixels I_w:

$$P(X_{cw}|I_w) = \frac{1}{Z_c} f(cc(I_w, T_c))^q \ ,$$

where Z_c is a class-specific normalizing constant. Substituting again, and then canceling, we have:

$$\frac{P(I_w|X_{cw})}{P(I_w)} = \frac{f(cc(I_w, T_c))^q}{\sum_{I'_w} P(I'_w) f(cc(I'_w, T_c))^q}$$

The function f normalizes the correlation coefficient to be roughly in $[0, 1]$; in this paper we use $f(cc(I_w, T_c)) = \max(cc(I_w, T_c), 0)/1000$. The exponent q serves to strengthen strong responses and suppress weak ones (a similar approach to that of Torralba et al. (2007)). We use $q = 6$ for all templates except the full-circle wrench end; because of its larger area, we found that $q = 8$ worked better. We can estimate the denominator of this ratio by sampling over training set patches (from both the foreground and background), and computing the expected correlation response: $\frac{1}{N} \sum_{I_w} f(cc(I_w, T_c))^q$. The ratio will be greater than 1 in cases where the template response is better than "average", and less than 1 otherwise.

7.1.2 Primitive Part Labeling

As we described above, we assume that we have been given labeled initial locations of a set of primitive parts making up each training object. Rather than hand-label the parts in the training images, we instead write down a very simple grammar (and geometry models) for each of the four types of wrenches. Then, for each training image, we find the maximum-likelihood parse tree that is rooted at the labeled bounding box centroid, using the appropriate grammar. (To find the max-likelihood parse tree, simply replace the sums in Equation 1 with maxes.) The locations of the leaves of this max-likelihood parse tree are used to initialize the learning process.

7.1.3 Bounding Box Prediction

In this paper, the composite parts of the grammars we learn consist of single points, with locations but no extent. Thus, the recognition algorithm we describe in Section 4.2, if naively applied to the wrench domain, will predict the best location of the centroid of a wrench in a test image, but not necessarily its width and height. But because we also want to predict the extent of the object, we have developed a simple approach to predicting the bounding box of the object, given its object class and most likely location.

As we described in the previous section, it is simple to predict the maximum-likelihood parse tree that is rooted at a given location and root part model. The leaves of this tree are primitive parts: image patches with both location and extent. So we learn a simple mapping from these predicted primitive parts to a bounding box of the object itself.

Specifically, after the PGG model has been learned, we use the learned grammar to calculate the max-likelihood parse tree for each training image (rooted at its labeled bounding box centroid). We then compute the bounding box of each set of predicted primitive parts, and estimate the average offset of each predicted boundary (left, right, top, bottom) from the corresponding boundary of the true labeled bounding box.

At test time, we calculate the max-likelihood parse tree for the test image, rooted at the best predicted object class and location, compute the bounding box of the leaves, and apply the learned offsets to the result to produce a final prediction of the bounding box of the object. This bounding box is compared to the true labeled bounding box for accuracy, using the performance metrics described below.

7.2 Qualitative Results

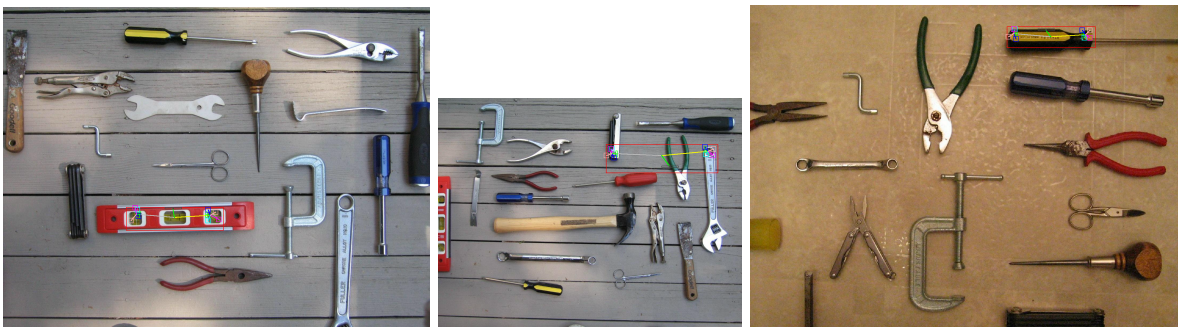
Figure 11 shows a typical learned grammar in the wrench domain. The structure makes sense: a wrench consists of a right end (C0001), a left and (C0002), and two horizontal bars; each end can be closed or open. However, the rule probabilities for the wrench ends strongly prefer the "open" choice; we have learned that, given our fixed set of appearance models, all four types of wrenches can be explained well by a model of open-open wrenches. Despite this surprising result, the model performs impressively. We expect that learning the primitive appearance models so that they span the representational space and avoid redundancy would result in even better structures and results.

Figures 12(a) and 12(b) show examples of good and bad detections produced by learned grammars.

```
wrench:
  1.0    wrench → C0001 C0002 dedge dedge
C0001:
  0.9752    C0001 → open-right side1 side2 tip1 tip2
  0.0248    C0001 → full-circle
C0002:
  0.0174    C0002 → full-circle
  0.9826    C0002 → open-left side3 side4 tip3 tip4
```

Figure 11: A typical learned grammar for wrenches. The primitive parts and their appearance models are shown in Figure 10(c).

(a) Good detections.



(b) Bad detections.

Figure 12: Examples detections produced by learned grammars.

7.3 Comparison With Simpler Models

To evaluate whether the PGG framework provides a performance benefit, we compared the full PGG model against simpler versions of the model:

- A set of grammars, one for each wrench type, where each grammar has a single flat rule.[3] No structure learning is performed, although EM is used to estimate the parameters. At test time, each grammar votes for the best location in the image, and the location with the highest likelihood score is returned as the overall prediction.

- A single grammar with a set of flat rules, one for each wrench type. Again, no structure learning is performed, but EM is used to estimate the parameters. This baseline is distinct from the first because it allows the weights on the four wrench classes to be learned rather than fixed, and furthermore for the most likely location to be chosen according to a weighted combination of the likelihood scores according to each rule, rather than a simple argmax.

[3] A flat rule has the object class on the left hand side and the set of primitive parts on the right.

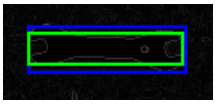


Figure 13: An example of 68% overlap. The blue (outer) box is the correct labeled bounding box, while the green (inner) box is the prediction.

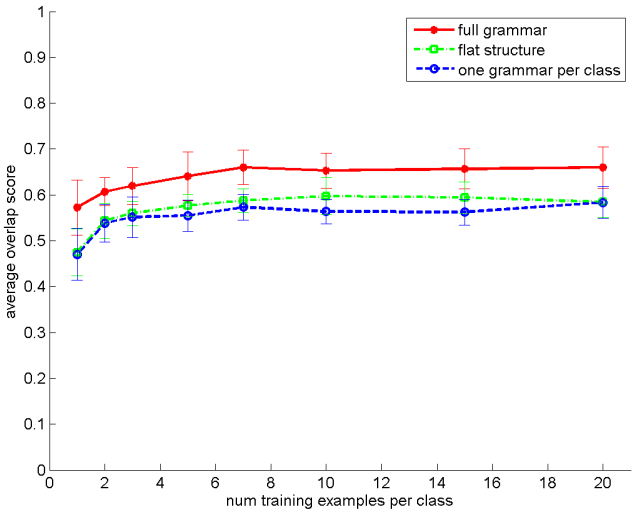


Figure 14: A comparison of the full PGG model against simpler versions of the model. Because there are four ground classes, the total training data is four times each x-axis value. Error bars represent 95% confidence intervals.

The first might be considered the simplest derivative of the PGG model. The second ensures that any benefit achieved by the full model over the first baseline is not due to tweaking the rule probabilities or geometry parameters during EM, but rather due to some aspect of the structure learning: the hierarchy introduced by building up structure, or the sharing of common substructure among different types of wrenches.

Our performance metric is the percentage overlap between the predicted bounding box w_d and the labeled bounding box w_l:

$$\frac{|\bigcap(w_d, w_l)|}{|\bigcup(w_d, w_l)|}$$

i.e., the ratio of the area of intersection of the windows to the area of their union. This will be one when the windows are identical and zero when they do not overlap at all. This metric is nice because it incorporates not just the accuracy of the predicted location, but also the predicted scale or extent.

We trained each model on training sets of increasing size, and tested the learned models on a set of 40 images (10 of each wrench type). We report the mean percentage overlap score on the test set for each training set size. We repeated this procedure 10 times for different splits of training and test data, and averaged the resulting curves together to produce Figure 14.

Reassuringly, the full PGG model enjoys a significant advantage over the two baseline models. Furthermore, the flat structure slightly outperforms the one-grammar-per-class approach, but not significantly so for most training set sizes. Therefore, we can feel confident that the structure learning process is responsible for most of the advantage of the full PGG model over the simplest model. These experiments do not allow us to determine whether the benefit is due to the hierarchy introduced during structure learning, or to the fact that this hierarchy allows compact sharing of substructure among related classes; we intend to explore this distinction further in future work.

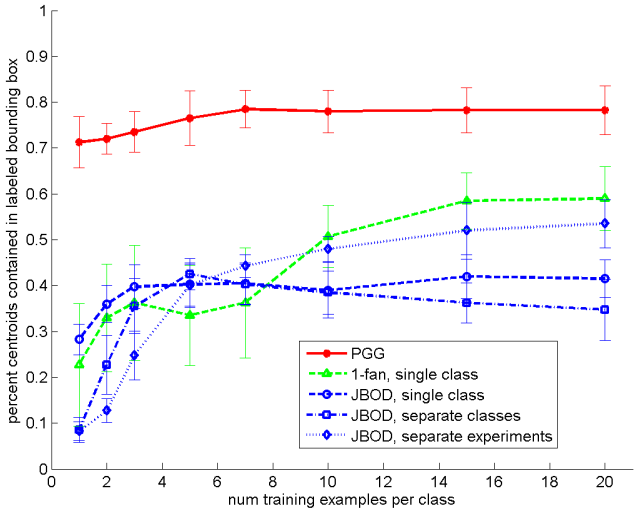


Figure 15: A comparison of the PGG model against other leading object recognition systems (Crandall & Huttenlocher, 2006; Torralba et al., 2007). Because there are four ground classes, the total training data is four times each x-axis value. Error bars represent 95% confidence intervals.

However, none of the slopes of the learning curves are particularly steep; the full model does not learn much faster than the baseline models, because they all learn quite fast. We suspect this is primarily because we are not yet learning the appearance models. In cases where appearance as well as geometry is being learned, sharing should provide much greater advantage because so many more parameters need be estimated.

Finally, to put the scores shown in Figure 14 in perspective, Figure 13 shows an example of what 68% overlap looks like in an image—it is quite a high localization score.

7.4 Comparison With Other Approaches

It is important to validate our overall approach through comparisons with other leading object recognition systems. Therefore, we also compared the full PGG model against the following systems:

- The 1-fan object detector (Crandall & Huttenlocher, 2006), treating all wrench types as one object class.
- Three variations of the joint boost object detector (JBOD) (Torralba et al., 2007): treating all wrench types as one object class; treating each wrench type as its own object class; and running a separate detection experiment for each wrench type and averaging the results.

In the first case, we used the authors' published implementation, while in the second we used our own reimplementation. We used the same values for all experimental parameters as were reported in the publications. We also controlled for object scale and orientation.

The methods to which we compare do not predict tight bounding boxes, but rather the location of primitive parts only (in the case of Crandall and Huttenlocher) or a large square centered on the object but not tightened around its silhouette (in the case of Torralba et al.). Therefore, we cannot usefully measure localization performance using the percentage overlap metric. Instead, we measure the percentage of times that the predicted object centroid falls within the labeled bounding box, across the test set. For the 1-fan detector, we take the predicted centroid to be the centroid of the predicted primitive parts, and for the joint boost detector, we take the centroid of the square as the centroid of the object. In general, the percentage of contained centroids metric is a less demanding metric than the percentage overlap, because it does not directly account for the predicted object's scale or extent.

The results are shown in Figure 15. To produce the curves, we followed a similar experimental procedure as described above in Section 7.3.

These experiments demonstrate that the localization task on the wrench data is a challenging one. Figure 15 shows that the PGG model outperforms the other systems by a significant margin on this dataset. It seems that, especially for small training set sizes, this task is indeed nontrivial; therefore we may conclude that the good performance of the PGG model is promising.

Nonetheless, it is important to point out that both of the systems against which we compare learn their appearance models, while we assume a pre-specified set of models. Although it may seem that our approach has an advantage due to the extra supervision, several recent authors have shown that choosing primitive parts automatically based on data outperforms using hand-chosen primitive parts (notably, Crandall and Huttenlocher (2006)). Thus, it seems possible that learning our appearance models would result in even better performance, and this is an important area of future work.

8 Discussion and Future Work

There are a number of possible avenues for future work in the PGG framework. First and foremost, the most pressing task is that of choosing or learning the appearance models automatically from the training data. The simplest approach to this issue might be to introduce a preprocessing phase to the learning process in which we learn a fixed set of appearance models from the data. Then, given those appearance models, the structure learning could occur as described in this paper. However, such an approach would require making a final decision about the set of primitive parts that would be included in the model prior to making any decisions about the composite parts. There is no facility in the current structure learning algorithm to add or delete primitive part models during the structure learning process, so we would be locked into the choice of primitive parts before doing any reasoning about how they fit together structurally or spatially in the model.

Instead, a better approach might be to incorporate the appearance model learning into the structure learning process. For example, we could again begin with a phase that produces an initial set of appearance models (i.e., primitive part models). But then, we could introduce structure learning operators that add, merge, or remove primitive parts from the grammar. Finally, depending on the form of the appearance models, we could extend the EM algorithm to estimate their parameters along with those of the geometry models.

Another area of future work concerns our choice of type of appearance model. Rather than edge intensity templates and a likelihood function that is based on cross correlation, we might consider building generative models over a combination of edge intensity, texture, or color features. Alternatively, we could move towards a more discriminative approach, in which images are preprocessed using bottom-up interest point, junction, or region detectors; these features could be described using, for example, histogram techniques such as SIFT descriptors (Lowe, 2004); and the appearance models could represent some distribution over the properties of the bottom-up features. The latter approach could even rely on a bottom-up detector of semi-local boundary features, for example that of Martin, Fowlkes, and Malik (2004).

With respect to the overall PGG model, we have done only preliminary work on how to model and search over scale. In the experiments described in this paper, small amounts of scale variation are modeled indirectly through the variance in the spatial models among the parts, but the primitive parts do not vary in scale at all. Ideally the scale of individual parts (both primitive and composite) in the model would be allowed to vary independently. Although this would be a natural extension of the current geometry models, it would lead to slower recognition. Instead, we might achieve a simpler form of scale invariance by modeling scale at the level of the entire object, and keeping it fixed among the parts for any specific scale. Similarly, the system described in this paper is not view-invariant, and an important area of future work might be to address this issue.

We are also interested in improving the recognition process so that we can predict more than a set of sparse primitive parts or a bounding box. For example, one could imagine using a somewhat sparse grammatical model to do the initial object detection in a novel image, identifying the most likely objects

and their best candidate locations, and then using both the most likely parse trees and the image pixels at each location as features in a segmentation procedure that predicts the set of local pixels that are contained by each object. Including reliable local boundary or junction information our appearance models would also help during a post-processing segmentation step.

Finally, there are a number of improvements that could be made to the structure learning algorithm. For example, the current structure score does not explicitly take into account the ability of a candidate grammar to discriminate among the object classes and the background. We could include a term in the score that reflects the separation among the likelihood scores assigned by each object class to the training objects of that type (including the background).

References

Bar Hillel, A., & Weinshall, D. (2006). Subordinate class recognition using relational object models. In *NIPS*.

Chen, S. F. (1995). Bayesian grammar induction for language modeling. In *Proc. Meeting of the Assoc. for Computational Linguistics (ACL)*.

Crandall, D., Felzenszwalb, P., & Huttenlocher, D. (2005). Spatial priors for part-based recognition using statistical models. In *CVPR*.

Crandall, D. J., & Huttenlocher, D. P. (2006). Weakly supervised learning of part-based spatial models for visual object recognition. In *ECCV*.

Crandall, D. J., & Huttenlocher, D. P. (2007). Composite models of objects and scenes for category recognition. In *CVPR*.

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., & Bray, C. (2004). Visual categorization with bags of keypoints. In *Proc. Workshop on Statistical Learning in Computer Vision, (ECCV)*.

de Marcken, C. G. (1996). *Unsupervised Language Acquisition*. Ph.D. thesis, Massachusetts Institute of Technology.

Felzenszwalb, P., & Schwartz, J. (2007). Hierarchical matching of deformable shapes. In *CVPR*.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *IJCV*, *61*(1).

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *CVPR*.

Fergus, R., Perona, P., & Zisserman, A. (2005). A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*.

Fidler, S., & Leonardis, A. (2007). Towards scalable representations of object categories: Learning a hierarchy of parts. In *CVPR*.

Fischler, M., & Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*, *C-22*.

Grauman, K., & Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*.

Grauman, K., & Darrell, T. (2006). Unsupervised learning of categories from sets of partially matching image features. In *CVPR*.

Han, F., & Zhu, S.-C. (2005). Bottom-up/top-down image parsing by attribute graph grammar. In *ICCV*.

Heckerman, D. (1999). A tutorial on learning with bayesian networks. In Jordan, M. (Ed.), *Learning in Graphical Models*. MIT Press.

Kadir, T., & Brady, M. (2001). Saliency, scale and image description. *IJCV*, *45*(2).

Leibe, B., & Schiele, B. (2003). Interleaved object categorization and segmentation. In *Proc. British Machine Vision Conf. (BMVC)*.

Lowe, D. G. (2004). Distinctive image features for scale-invariant keypoints. *IJCV*, *60*(2).

Manning, C. D., & Schütze, H. (2002). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Martin, D. R., Fowlkes, C. C., & Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, *26*(5).

Matas, J., Chum, O., Urban, M., & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conf. (BMVC)*.

Nevill-Manning, C. G., & Witten, I. H. (1997). Identifying hierarchical structure in sequences: A linear-time algorithm. *JAIR*, *7*.

Ommer, B., & Buhmann, J. M. (2007). Learning the compositional nature of visual objects. In *CVPR*.

Pollak, I., Siskind, J. M., Harper, M. P., & Bouman, C. A. (2003). Parameter estimation for spatial random trees using the EM algorithm. In *Proc. IEEE International Conf. on Image Processing (ICIP)*.

Rosenfeld, A. (1973). Progress in picture processing: 1969-71. *ACM Computing Surveys*, *5*(2).

Siskind, J., Sherman, Jr., J., Pollak, I., Harper, M., & Bouman, C. (2007). Spatial random trees grammars for modeling hierarchical structure in images with regions of arbitrary shape. *PAMI*, *29*(9).

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., & Freeman, W. T. (2005). Discovering objects and their location in images. In *ICCV*.

Sudderth, E. B., Torralba, A., Freeman, W. T., & Willsky, A. S. (2005a). Describing visual scenes using transformed dirichlet processes. In *NIPS*.

Sudderth, E. B., Torralba, A., Freeman, W. T., & Willsky, A. S. (2005b). Learning hierarchical models of scenes, objects, and parts. In *ICCV*.

Sudderth, E. B., Torralba, A., Freeman, W. T., & Willsky, A. S. (2006). Depth from familiar objects: A hierarchical model for 3d scenes. In *CVPR*.

Torralba, A., Murphy, K. P., & Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*.

Torralba, A., Murphy, K. P., & Freeman, W. T. (2007). Sharing visual features for multiclass and multiview object detection. *PAMI*, *29*(5).

Tu, Z., Chen, X., Yuille, A. L., & Zhu, S.-C. (2005). Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, *63*(2).

Tu, Z., & Zhu, S.-C. (2006). Parsing images into regions, curves, and curve groups. *IJCV*, *69*(2).

Ullman, S., & Epshtein, B. (2006). Visual classification by a hierarchy of extended fragments. In *Towards Category-Level Object Recognition, Lecture Notes on Computer Science*. Springer-Verlag.

Zhu, L. L., Chen, Y., & Yuille, A. (2006). Unsupervised learning of a probabilistic grammar for object detection and parsing. In *NIPS*.

Zhu, S.-C., & Mumford, D. (2006). A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, *2*(4).