Abstract of "Probabilistic Scene Grammars: A General-Purpose Framework For Scene Understanding" by Jeroen Chua, Ph.D., Brown University, May 2018.

We propose a general-purpose probabilistic framework for scene understanding tasks. We show that several classical scene understanding tasks can be modeled and addressed under a common representation, approximate inference scheme, and learning algorithm. We refer to this approach as the Probabilistic Scene Grammar (PSG) framework. The PSG framework models scenes using probabilistic grammars which capture relationships between objects in terms of compositional rules that provide important contextual cues for inference with ambiguous data. We show how to represent the distribution defined by a probabilistic grammar using a factor graph. We also show how to estimate the parameters of a grammar using an approximate version of Expectation-Maximization, and describe an approximate inference scheme using Loopy Belief Propagation with an efficient message-passing scheme. Inference with Loopy Belief Propagation naturally combines bottom-up and top-down contextual information and leads to a robust algorithm for aggregating evidence. To demonstrate the generality of the approach, we evaluate the PSG framework on the scene understanding tasks of contour detection, face localization, and binary image segmentation. The results of the PSG framework are competitive with algorithms specialized for these scene understanding tasks.

Probabilistic Scene Grammars: A General-Purpose Framework For Scene Understanding

by
Jeroen Chua
B. A. Sc., University of Toronto, 2010
M. A. Sc.,University of Toronto, 2012

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2018

This dissertation by Jeroen Chua is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____

Professor Pedro Felzenszwalb, Director

Recommended to the Graduate Council

Date _____ _____

Professor Erik Sudderth, Reader

Date _____ _____

Professor Stuart Geman, Reader

Approved by the Graduate Council

Date _____ _____

Andrew G. Campbell
Dean of the Graduate School

Acknowledgements

First and foremost, I'd like to thank my advisor Pedro Felzenszwalb for guiding me through graduate school. Throughout my PhD, Pedro has provided insightful comments, crucial connections to past work, and pointed out where more work was needed to understand the models and approaches we considered. Without his research vision and utmost patience with my at times floundering progress, this thesis would not be possible. Thank you, Pedro, and I hope your influence of attention to detail stays with me for the entirety of my career!

I would also like to thank Stuart Geman and Erik Sudderth. I first spoke with Professor Geman at a Snowbird Learning workshop, where he asked me a question about how a model I had presented handled explaining-away. It was an insightful comment, and I had hoped to be able to collaborate with him at Brown University. I was fortunate for this wish to be granted, and our early meetings with Pedro and Jackson Loper were invaluable in establishing the research direction I would take during my PhD. I have also been fortunate enough to interact with Erik Sudderth and his research group. Erik has provided insightful comments and suggestions during my PhD studies and welcomed me to join his group's research meetings. For that I am grateful as it provided an opportunity to learn about the cool things that his group was doing, but also gave me the opportunity to chat with his group.

Not only have the faculty here at Brown University been a source of support and inspiration, but the graduate students and post-docs have been as well. I'd particularly like to thank John Oberlin, Sobhan Parizi, and Pierre-Yves Laffont for helpful research discussions and overall just being really awesome people! I'd also like to thank Chris Blake, for extremely useful tips on writing, being an academic, and for simply being a source of encouragement and inspiration. Be it chats in the robotics lab, chats over a barbell at the gym, or chats over a fiercely contested board game, discussions were always lively and I am thankful to have had the opportunity to collaborate with you all!

I also thank my family for their support and understanding. In particular, they have tolerated my desire to be a student for seemingly as long as possible. Throughout my life, their love and support has enabled me to pursue what truly interests me and has made me the person I am today.

Last and certainly not least, I'd like to thank Sunny, my loving girlfriend, for PhD-levels of support, encouragement, and patience. Besides my advisor, I'm pretty sure Sunny has heard me talk the most about my work and the nitty gritty details of what that has entailed. This thesis would not be possible without her support, love, and encouragement.

Contents

Chapter 1

Introduction

In this thesis we present a general-purpose probabilistic framework for scene understanding tasks. We show that several classical scene understanding problems can be modeled and addressed under a common representation, approximate inference scheme, and learning algorithm. We refer to this general-purpose framework as the Probabilistic Scene Grammar (PSG) framework.

Currently in the field of computer vision, different scene understanding tasks lend themselves to different representations and algorithms. Table 1.1 gives a few examples of scene understanding tasks and approaches to address them. Table 1.1 is not an exhaustive list of tasks, but illustrates the current state of affairs in the computer vision field whereby scene understanding tasks are often tackled by problem-specific approaches.

The study and realization of a general scene understanding framework has several benefits, outlined below.

- Fundamental improvements to such a general framework yields benefits and potential improvements on many scene understanding tasks simultaneously. In contrast, if an algorithm is specifically designed for a single task, then improving that algorithm only realizes improvements on that one task.

- The space of consumer applications is rapidly expanding and novel scene understanding tasks are being proposed. One such an example is image-to-caption described in [61]. In this task, the goal is to generate a string of text that describes an image. Although it is possible to design new representations and algorithms to address each novel scene understanding task as they arise, such a strategy is laborious. An alternative to this problem-specific approach is to have a framework that is expressive enough to handle scene understanding tasks in general, concrete enough to be implemented, and fast enough to be practical. The hope is that if novel scene

Scene understanding task	Approach taken
Contour detection	Deep Learning [50, 66] Canny-edge Detection [7] Global Probability-of-Boundary [1] Field-of-patterns [16]
Image segmentation	Cut-based approaches [51, 5, 6] Level-sets [60] Random Forests [49] Global Probability-of-Boundary [1] Markov-random Fields, Conditional-random Fields [29, 31, 2] Mumford and Shah [40]
Image recognition	Convolutional Neural Networks [36, 32] Bag-of-words/Spatial Pyramid models [35, 23]
2D and 3D object localization	Dalal and Triggs pedestrian detector [9] DPM [22] Pictorial Structures [13] Convolutional Neural Network [46, 21] Clouds of Oriented Gradients [47]

Table 1.1: Common scene understanding tasks and some approaches to address them. Note the myriad of distinct approaches across and within tasks.

understanding tasks can be expressed in a form compatible with this general framework, then suitable solutions can be found with minimal research and engineering work.

- Related scene understanding tasks may constrain or provide useful information for other scene understanding tasks. For example, solving the image segmentation problem may help with object recognition since image segments may correspond to objects and the shape of the segments can be useful information in recognizing objects. In the work of [57], the problems of motion estimation and image segmentation inform one another since rigid objects tend to have similar motion, and entities with similar motion across a long time-scale may belong to the same object. By iteratively refining the solution to one task by conditioning on the solution of a related task, one may achieve a better overall result than by handling each task in isolation. A general-purpose, unified framework for scene understanding tasks would allow one to naturally model different tasks simultaneously and combine their results in a principled fashion.

- It is a scientifically interesting question to ask whether these myriad of scene understanding tasks, which have historically been addressed with different representations and algorithms in different formalisms, can be understood in a general-purpose probabilistic framework. In particular, one may ask questions such as “How can we represent different scene understanding

tasks under a common schema?", "What is a practical, effective problem-agnostic inference scheme?", and "How does one perform problem-agnostic learning and parameter estimation?". Studying such questions may deepen our understanding of the visual world and the nature of scene understanding tasks facing the field of computer vision. In this thesis, we take some steps in answering such questions.

1.1 Design considerations

To design a general-purpose probabilistic framework for scene understanding tasks, two issues must be carefully considered: the modeling of contextual information and efficiency of inference. Consider the image recognition task in Figure 1.1. Even to humans, the image patches shown are ambiguous and it can be difficult to determine what the objects are. The full images from which the image patches were taken are shown in Figure1.2. After seeing the entire image, recognizing the depicted objects and object parts is straightforward.

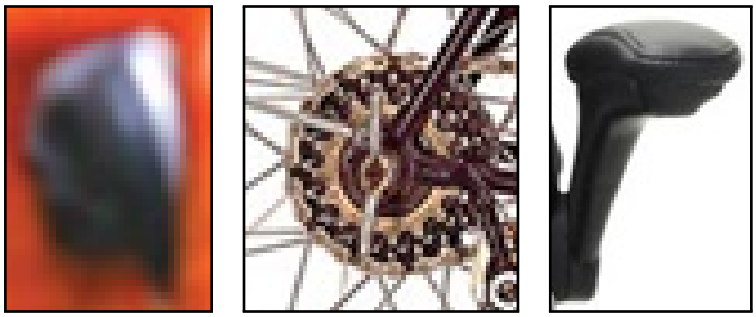


Figure 1.1: Each image patch depicts a part of an object. Name the object and part.

There is little agreement in the computer vision literature about what constitutes "context", though it is typically taken to denote "any and all information that may influence the way a scene is perceived" [55]. In the tasks of image recognition and object localization, one notion of context is that objects have part/whole relationships and certain objects often co-occur. In the examples in Figure 1.1, knowing the object from which those object parts come from aid in recognizing those parts. In the task of contour detection, one may use the idea that contours tend to be long, contiguous curves. In image segmentation, one may use the idea that objects tend to be compact in space, and so image segments should be compact. If one is to build a general-purpose framework for scene understanding, it is crucial for that framework to be able to express a notion of context suitable for a range of scene understanding tasks. In the PSG framework, we model the broad notion of context in terms of compositional and geometric relationships between objects.

Figure 1.2: Original images the image patches from Figure 1.1 were taken from. The image patches are denoted in blue boxes. The objects/parts are: bird/beak, bicycle/cogset, chair/armrest.

For practical reasons, we seek to develop efficient inference schemes for tackling scene understanding tasks. Suppose we have an autonomous driving car that needs to detect other cars to avoid collisions; a system in practice has milliseconds to detect the other cars and plan a collision-avoiding route. Consider another example where one has a 3D brain scan of a hospital patient, and one must determine if the patient has a life-threatening brain hemorrhage and if so, output a 3D segmentation that localizes the site of the hemorrhage. Here too, time is of the essence. Because some scene understanding tasks may be time-sensitive, we are concerned with developing a general-purpose framework that is not only flexible enough to be applicable to a diverse set of scene understanding tasks, but also admits efficient inference. Unfortunately, exact inference in a general probabilistic model is intractable. In this work, we seek to develop efficient approximate inference schemes.

1.2 Thesis contributions

In this thesis we address four key aspects of defining and assessing a general-purpose probabilistic framework for scene understanding: 1) the **representation** of scene understanding tasks under a common schema, 2) efficient, problem-agnostic **approximate inference**, 3) the **learning** of model parameters under varying levels of supervision, 4) the **experimental evaluation** of the framework. A final contribution is the concertization of the framework in a single, general implementation. We refer to the framework developed in this thesis as the Probabilistic Scene Grammar (PSG) framework.

1.2.1 Representation

To represent general scene understanding tasks, we use probabilistic grammars which have been successful in object modeling for object recognition (see [28, 3, 59, 15, 68, 22, 17, 13, 17, 70]). Probabilistic grammars are defined in terms of a set of symbols, a set of rules that represent relationships between symbols, and a set of rule probabilities that encode how often those relationships occur. The set of symbols represents entities we wish to reason about. For example, the symbols of the grammar might be a face and its parts if we wish to detect faces in scenes, or it might be a set of short curves that compose into long curves if we wish to detect contours. Compositional relationships such as “a face has two eyes, a nose and a mouth, and sometimes a beard”, and geometric relationships such as “the mouth is located somewhere below the centre of the face”, are encoded as rules and rule probabilities in the grammar. Importantly, probabilistic grammars express a notion of context through compositional and geometric relationships. Such relationships provide contextual cues for inference with ambiguous data. For example, the presence of some parts of a face in a scene provides contextual cues for the presence of a face and its other parts.

To better understand a probabilistic model, it is often helpful to examine samples from the model when possible. To give a sense of the kinds of models that can be represented in the PSG framework, we show samples drawn from example models in Figure 1.3. Chapter 3 describes the exact models used to generate the samples.

1.2.2 Approximate inference

To perform approximate inference, rather than operate directly on probabilistic grammars, we first define a novel transformation from a probabilistic grammar to a graphical model represented by a factor graph. This transformation induces a probability distribution over interpretations of a scene. Unfortunately, exact inference with a general probabilistic model is is NP-hard (see [8]), so if we are to have a flexible representation applicable to many understanding tasks, it is necessary to either restrict the form of the probabilistic model to make inference tractable or to employ approximate inference techniques. In this thesis, we choose the latter. Fortunately, there has been much work on approximate inference schemes in factor graphs; Loopy Belief Propagation (LBP) [33] is one such approach and has been shown in practice to give good results on a variety of tasks [42, 33, 14]. LBP performs approximate inference by passing "messages" between the nodes of a factor graph until some convergence criterion is met. The messages can then be used to compute marginal probabilities and answer questions such as "what is the probability there is a face at location (x, y) in the scene?". The PSG framework makes use of special cases of LBP whereby messages can be computed efficiently. One of our contributions is the derivation of efficient analytical methods for computing messages for the factor graphs under consideration.

1.2.3 Learning

As with many scene understanding approaches, the PSG framework has model parameters that are ideally learned from data. A rule probability encoding how often a face has a beard is one such model parameter. In general, to learn model parameters, we employ an approximate Expectation-Maximization (EM) algorithm. Here, the exact posterior quantities computed in the Expectation-step are replaced by an approximation to the posterior computed by LBP, and the Maximization-step is standard. The general idea of replacing the exact posteriors computed in the Expectation-step by approximate posteriors computed by LBP was studied in [26]. While [26] was primarily concerned with convergence guarantees, for the models in this thesis, the primary issues are speed and performance of the learned models; convergence failure was not a major issue. Further, [26] specifies a Maximization-step that can be intractable to perform for some probabilistic models. In this

thesis, we show that for the family of models considered, the Maximization-step can be computed efficiently.

Since the Expectation-step is only approximate, this approximate EM algorithm is not guaranteed to have a non-decreasing data-likelihood; however, we have found that this approximate EM algorithm leads to empirically good results.

1.2.4 Experimental evaluation

We evaluate the PSG framework on three scene understanding tasks: contour detection, face localization, and image segmentation. We show that the PSG framework is competitive with algorithms specifically designed for these tasks, despite the generality of the framework.

For the tasks of contour detection and image segmentation, we have a noisy real-valued image $\mathbf{D}$ and we seek to recover a binary-valued map $\mathbf{B}$ that is the same size as $\mathbf{D}$. We assume that $\mathbf{D}$ is obtained by sampling each pixel $\mathbf{D}(i,j)$ independently from a Normal distribution whose mean depends on the value of $\mathbf{B}(i,j)$ with some known standard deviation σ. Formally,

$$\mathbf{D}(i,j) \sim \mathcal{N}(\mu_{\mathbf{B}(i,j)}, \sigma).$$

The goal of inference is to recover $\mathbf{B}$ from $\mathbf{D}$. For contour detection, we evaluate on the Berkeley Segmentation Dataset (BSD500) described in [1] using a standard train/test split. For image segmentation, we evaluate on a subset of the Swedish Leaf Dataset described in [53].

For the task of face localization, we have have images with one or more faces. The task is given an image, localize the face(s) and the parts of each face. We evaluate on a subset of Labelled Faces in the Wild (LFW) dataset introduced in [27], and our own dataset of family portraits collected from the Internet. We manually annotate each image with bounding box information of all faces and their parts.

Figure 1.4 shows examples of the inputs, desired outputs, and actual outputs from the PSG framework on these scene understanding tasks.

1.2.5 General implementation

In this thesis, experiments involving the PSG framework were performed using a single, general implementation of the PSG framework. The ideas and formalisms outlined in this thesis not only provide a conceptual framework in which one can reason abstractly about different scene understanding tasks, but also allows one to realize a concrete, unified framework to handle diverse scene understanding tasks in practice.

To handle different tasks in this general implementation, one simply expresses a model in a high-level "language", and designs an appropriate data model. The implementation automatically constructs a data structure representing the probabilistic model (or an approximation of it), and performs parameter estimation (learning) and inference with the model. The contribution here is of an engineering nature: the knowledge that it is possible to take the conceptual framework outlined in this thesis and concretize them in a truly general implementation.

This approach to a general implementation for generic tasks is similar to the approach of the Probabilistic Programming Language (PPL) community [48, 34, 58] whereby a user specifies an appropriate data model and how to sample from a prior probability distribution, and a potentially suitable inference algorithm is automatically constructed by the PPL framework. We outline the connections to this community in 1.3.

1.3 Related work

The desire for a general-purpose computational framework for scene understanding tasks is shared by the PPL community. In particular, the Picture and Edward frameworks described in [34] and [58], respectively, share the high-level goal of having a general-purpose representation and inference engine for scene understanding. However, these works differ from the PSG framework in both the goal and method of inference. Picture and Edward seek to find high-probability scene representations encoded as probabilistic program traces via Markov-Chain Monte-Carlo (MCMC) sampling methods and variational inference schemes. The PSG framework finds marginal distributions over aspects of the scene using LBP. The incorporation of the data model differs substantially as well. For example, in the Picture framework, the data model is combined with a prior over scenes using a computer-graphics renderer. In contrast, the PSG framework incorporates data terms using unary potentials in a factor graph defined in terms of extracted features. Although the incorporation of data terms is simpler in the PSG framework than in many PPL frameworks, the abilities to handle explaining-away and generate photo-realistic images are sacrificed by the PSG framework. Lastly, PPL approaches such as Picture and those proposed by Ritchie (see [48]) take the view of performing inference as analysis-by-synthesis, or as a Bayesian inverse-graphics problem. In contrast, the PSG framework frames the problem of inference in a purely analytical approach whereby generating realistic images is not a goal of the framework.

The PSG framework describes scene understanding tasks in a compositional framework. The idea of performing scene understanding in a compositional framework has been a long-standing goal in computer vision. The notion of perceptual organization using grouping and compositional rules

goes back at least to the Gestalt theory of perception described by [45] and the "neocognitron" model of [19].

The idea of representing scene understanding tasks in a compositional framework has also been investigated in modern approaches. A major relevant work in this vein is the work of [28] whereby a hierarchical "compositional machine" describes relationships between entities in a Bayesian network. The representation scheme used by the PSG framework is inspired by [28], however, the PSG framework is subtly different as it uses a factor graph to represent the distribution over scenes. For inference, [28] uses a greedy search heuristic to find plausible interpretations of the scene. In contrast, the PSG framework uses LBP for approximate inference. Also, we study the performance of the PSG framework on a more diverse set of tasks; while [28] studies the task of reading vehicle license plates, we consider the tasks of contour detection, face localization, and image segmentation.

Deformable Part Models (DPM) [12] and Pictorial Structures (PS) [13] are compositional frameworks that the PSGs framework takes much inspiration from. DPM and PS represent objects as a collection of parts and connections between parts and can be understood as a special kind of probabilistic grammar. The form of PS and DPM models allows for efficient exact inference via dynamic programming. Further, PS assumes there is one of each object in the image, while the PSG framework makes no such assumption. The PSG framework considers more general object models and make fewer assumptions about scenes. The trade-off, however, is the inference scheme the PSG framework employs is only approximate and in practice, is slower than the exact inference schemes used by DPM and PS. Nevertheless, the scope of tasks representable in the PSG framework is larger and, as we show in Chapter 9, is capable of outperforming PS on a face localization task.

There has been much work in the area of inference for probabilistic compositional models similar to the PSG framework. The problem of exact inference with general probabilistic models is NP-hard (see [8]). Indeed, efficient inference has been the bane of many probabilistic compositional models. To deal with inference in such models, a variety of ad-hoc or slow sampling schemes have been proposed in the literature (see [34, 58, 28, 59, 68]). For example, the works of [34], [58], [59] and [68] use MCMC techniques for inference, and [28] uses a coarse-to-fine greedy approach to search for potential objects. Ad-hoc heuristics are brittle and are often only applicable to a narrow range of situations. Approaches that rely on MCMC sampling schemes can also be brittle if the MCMC scheme relies on the design of effective proposal distributions. In this work, we use LBP as it is robust in practice and requires relatively little problem-specific engineering. To our knowledge, this thesis is the first to employ LBP for inference with a probabilistic grammar.

The problem of inference for general probabilistic models is a main area of research in the field of machine learning. As such, there are potential alternatives to the LBP approximate inference scheme used in the PSG framework. Variational Inference [63] is a well-studied approximate inference

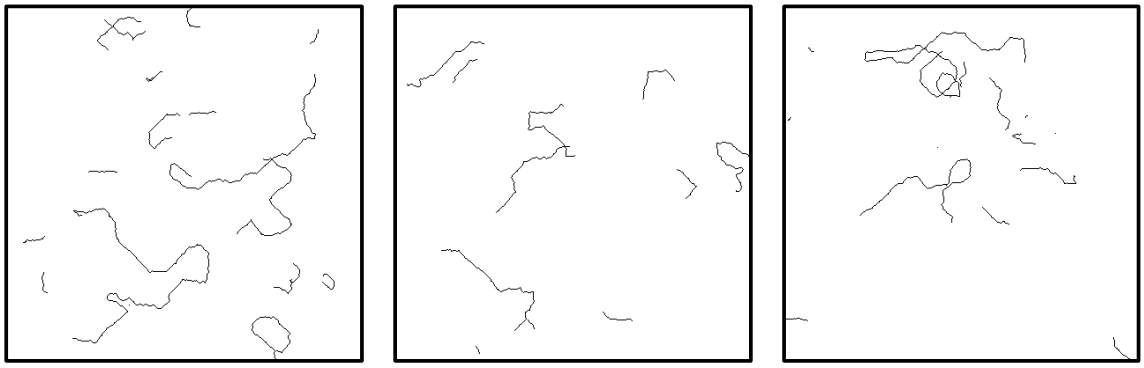
scheme that is fast in practice. However, Variational Inference tends to produce inferior results to LBP in certain situations [65]. MCMC techniques such as Gibbs sampling [20], Metropolis-Hastings [25], Hybrid Monte Carlo [44], and Slice Sampling [43] can also be used to perform inference in the kinds of probabilistic models considered by the PSG framework. Indeed, as stated earlier in this chapter, past approaches have used MCMC techniques. However, MCMC techniques in practice can suffer from being slow to converge to the target posterior distribution and may require careful tuning and design, which makes them not ideal for handling general scene understanding tasks where time is of the essence. Other message-passing schemes for inference in loopy graphs exist, such as Tree-reweighted Belief Propagation [30], Generalized Belief Propagation [67], Convergent Belief Propagation [38], and Non-parameteric Belief Propagation [56]. It is possible to employ any of these message-passing schemes as the inference engine in the PSG framework. Tree-reweighted Belief Propagation in particular can be useful if LBP has convergence issues, and Generalized Belief Propagation can be useful when one wishes to trade inference speed for increased inference accuracy. In this work, we use LBP as the inference engine since it is a relatively simple message-passing scheme, and we can exploit the particular form of the probabilistic models expressible in the PSG framework to perform efficient message computation.

1.4 Thesis organization

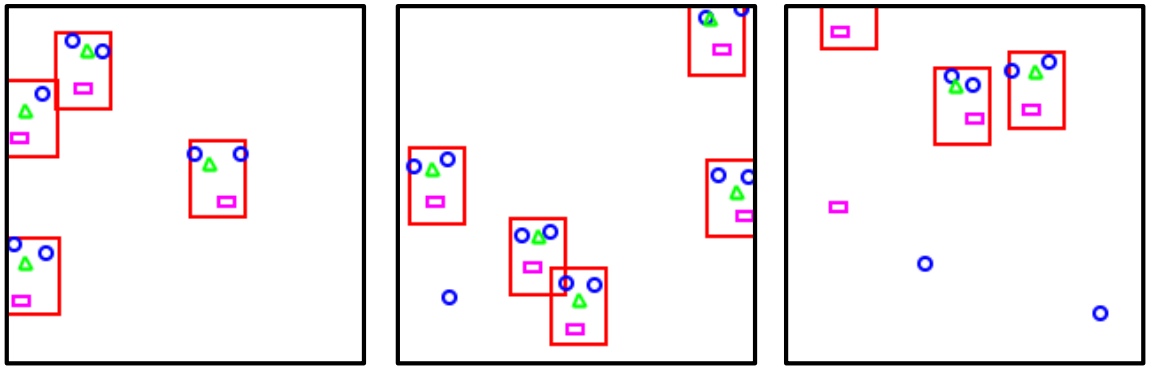
The outline below specifies the organization of this thesis.

- Chapter 2: a formal description of the representation (a probabilistic grammar) used by the PSG framework.
- Chapter 3: some example grammars that can be specified in the PSG framework.
- Chapter 4: description of the transformation of a PSG model into a factor graph.
- Chapter 5: description of the approximate inference scheme used in the PSG framework. In particular, Chapter 5 contains the derivations of the LBP message-passing equations and characterizes the time-complexity of computing messages in the PSG factor graph.
- Chapter 6: examples of running LBP on the example grammars specified in Chapter 3.
- Chapter 7 elucidates the connections between the PSG framework and the Pictorial Structures model of [13].
- Chapter 8: description of the approximate EM learning algorithm used in the PSG framework.

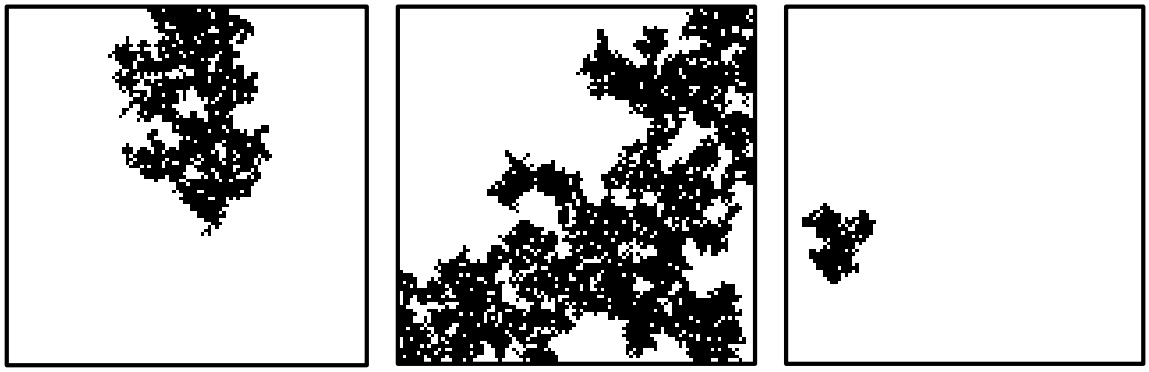
- Chapter 9: experimental evaluation of the PSG framework on the tasks of contour detection, face localization, and binary image segmentation.
- Chapter 10: description of PSG model transformations that allow for even faster approximate inference.
- Chapter 11: summary of research contributions and suggestions for future research directions that build off the PSG framework.



(a) Samples of contour maps generated by a model of contours. Note that the contours are of varying lengths and shapes and have variable curvature.

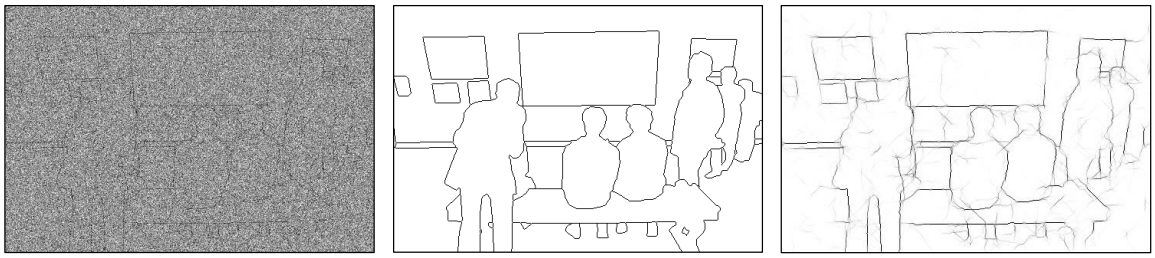


(b) Samples of faces generated by a model of faces. Note the geometric variability in the locations of the parts of the face and the variable number of faces in each scene. This face model allows parts of the face to appear on their own.

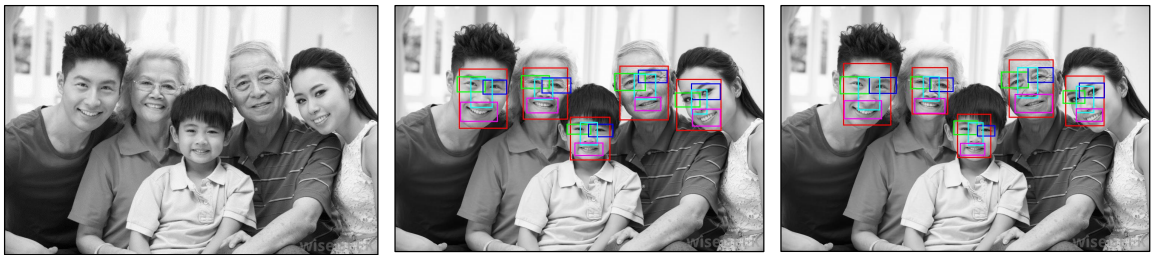


(c) Samples of binary image segmentation maps generated by an image segmentation model. Foreground is shown in black, background is shown in white. The model used here constrains the foreground to be a single connected component but allows for "holes" in the foreground.

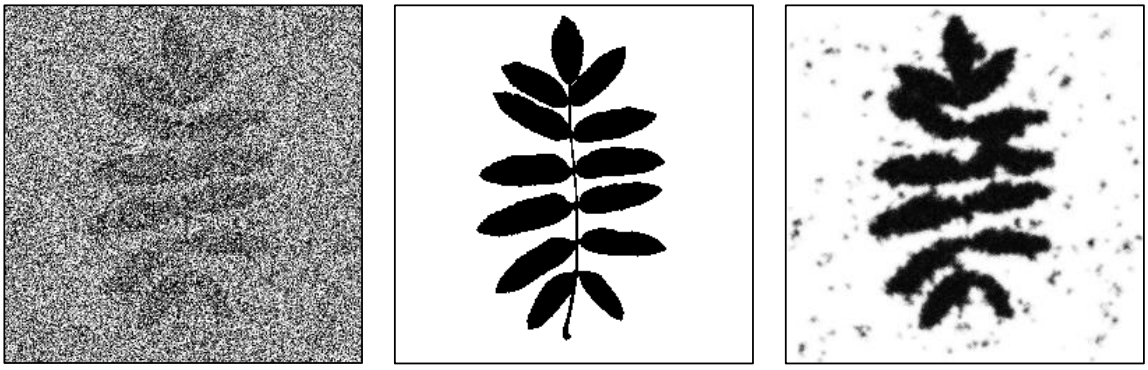
Figure 1.3: Samples from models used for contour detection, face localization, and binary image segmentation. All models are expressed in the PSG framework.



(a) The task of contour detection. **Left**: a noisy input image $\mathbf{D}$. **Middle**: the desired output: a binary contour map, $\mathbf{B}$. **Right**: Visualization of the approximate marginal probabilities $\hat{p}(\mathbf{B} \mid \mathbf{D})$ computed by the PSG framework. Darker pixels indicate a higher approximate marginal probability.



(b) The task of face localization. Here, we wish to localize faces, left eyes, right eyes, noses, and mouths. **Left**: an input image $\mathbf{D}$. **Middle**: the desired output: a localization of the face and each of its parts in terms of bounding boxes. **Right**: The top K detections for faces and their parts, where K is the ground truth number of faces in the image.



(c) The task of binary image segmentation. **Left**: a noisy input image $\mathbf{D}$. **Middle**: the desired output: a binary segmentation map, $\mathbf{B}$. **Right**: Visualization of the approximate marginal probabilities $\hat{p}(\mathbf{B} \mid \mathbf{D})$ computed by the PSG framework. Darker pixels indicate a higher approximate marginal probability.

Figure 1.4: Examples of the scene understanding tasks we use to evaluate the PSG framework.

Chapter 2

Probabilistic Scene Grammars

We take a Bayesian point of view where the goal of a computer vision algorithm is to estimate a description of a scene from a set of observations. A key component of this approach is a prior model over scenes, $p(S)$, that captures the statistical regularities of scenes in the world.

A probabilistic scene grammar (PSG) defines a set of possible scenes and a probability distribution over them. Scenes are defined using a library of building blocks, or bricks. Each brick is a pair of a type and a pose. The type is a symbol from a finite alphabet and the pose is an element from a finite pose space. For example, one brick in a scene might be the pair $(\mathsf{FACE}, (30, 40))$ representing a face at location $(30, 40)$ in the image. We capture structural and geometric relationships between bricks using a library of production rules.

To define a distribution over scenes $p(S)$ we consider a process for generating random scenes using a set of production rules. The process starts from an initial set of bricks that are spontaneously generated. Each of the initial bricks is probabilistically expanded to generate new bricks. This process continues until all bricks in the scene have been expanded. The result is a set of bricks organized in a hierarchical fashion. The formal definition of this process is given below. In the next chapter we describe some example grammars and illustrate the random scenes they generate.

In a probabilistic scene grammar the initial generation of bricks in a scene is governed by self-rooting probabilities. The possible expansions of a brick into other bricks is determined by a set of production rules, rule selection probabilities and conditional pose distributions.

Definition 1 *A probabilistic scene grammar (PSG) is defined by a 6-tuple* $\mathcal{G} = (\Sigma, \Omega, \mathcal{R}, q, \epsilon, \gamma)$.

1. Σ *is a finite set (the symbols).*
2. $\Omega = \{ \Omega_A \mid A \in \Sigma \}$ *where* Ω_A *is a finite set (the pose spaces).*
3. $\mathcal{R}$ *is a finite set of production rules of the form* $A_0 \to A_1, \ldots, A_n$ *where* $n \geq 0$ *and* $A_i \in \Sigma$.

Let r be a rule in $\mathcal{R}$. We use n_r to denote the number of symbols in the right-hand-side (RHS) of r. We use $A_{(r,0)}$ to denote the left-hand-side (LHS) of rule r and $A_{(r,i)}$, $1 \leq i \leq n_r$, to denote the i-th symbol in the RHS of r. We denote the set of rules with symbol A in the LHS by $\mathcal{R}_A$.

4. *$q = \{ q_A \mid A \in \Sigma \}$ where q_A is a distribution over $\mathcal{R}_A$ (the rule selection probabilities).*

5. *$\epsilon = \{ \epsilon_A \mid A \in \Sigma \}$ is a set of probabilities (the self-rooting probabilities).*

6. *$\gamma = \{ \gamma_{(\omega,r,i)} \mid r \in \mathcal{R}, 1 \leq i \leq n_r, \omega \in \Omega_{A_{(r,0)}} \}$ is a set of conditional pose distributions. Each conditional pose distribution $\gamma_{(\omega,r,i)}$ has an associated set of parameters $\theta_{(\omega,r,i)}$ indexed by $\Omega_{A_{(r,i)}}$. We have $\gamma_{(\omega,r,i)} : \{0,1\}^{\Omega_{A_{(r,i)}}} \to \mathbb{R}_{\geq 0}$ with*

$$\sum_{W} \gamma_{(\omega,r,i)}(W \mid \theta_{(\omega,r,i)}) = 1 \quad \forall \omega \in \Omega_{A_{(r,0)}}$$

where the summation is over all possible values of $W \in \{0,1\}^{\Omega_{A_{(r,i)}}}$. We use

$$\theta = \{\theta_{(\omega,r,i)} \mid r \in \mathcal{R}, 1 \leq i \leq n_r, \omega \in \Omega_{A_{(r,i)}}\}$$

to denote the set of parameters that govern the conditional pose distributions γ.

Intuitively, the conditional pose distributions γ model geometric and cardinality relationships between bricks. For example, consider a FACE at location $(30, 40)$ in the image. A conditional pose distribution could model that a FACE has exactly one NOSE, and model the distribution over the location of the NOSE of the FACE. As another example, consider an EYE at location $(30, 40)$ in the image. A conditional pose distribution could model how many EYELASHES an EYE has, and the distribution over locations of the EYELASHES of the EYE.

In this thesis, we consider two kinds of conditional pose distributions: the $\mathrm{Categorical}$ distribution, and the $\mathrm{IndBern}$ (short for $\mathrm{Independent\text{-}Bernoullis}$) distribution, defined below. Below, let W be a set of binary random variables indexed by Υ. Define the set $I(W) = \{k \mid W_k = 1, k \in \Upsilon\}$.

Definition 2 *Let Υ be an index set for W and a set of parameters $\theta = \{\theta_k \mid k \in \Upsilon, 0 \leq \theta_k \leq 1, \sum_{k \in \Upsilon} \theta_k = 1\}$. We define*

$$\mathrm{Categorical}(W \mid \theta) = \begin{cases} \prod_{k \in \Upsilon} \theta_k^{W_k}, & \sum_{k \in \Upsilon} W_k = 1 \\ 0, & \textit{otherwise.} \end{cases} \tag{2.1}$$

Definition 3 *Let Υ be an index set for W and a set of parameters $\theta = \{\theta_k \mid k \in \Upsilon, 0 \leq \theta_k \leq 1\}$. We define*

$$\mathrm{IndBern}(W \mid \theta) = \prod_{k \in \Upsilon} \theta_k^{W_k} (1 - \theta_k^{W_k})^{1 - W_k}. \tag{2.2}$$

Note that the $\mathrm{IndBern}$ distribution is defined in terms of independent but not identically-distributed Bernoulli distributions. Also, in a $\mathrm{Categorical}$ distribution, a single binary random variable from the set W has value 1, and in an $\mathrm{IndBern}$ distribution, the binary random variables are independent. Consider a rule $r \in \mathcal{R}$ and the i-th symbol in the RHS of r. A $\mathrm{Categorical}$ distribution is useful to model a situation in which a brick of type $A_{(r,0)}$ generates exactly one brick of type $A_{(r,i)}$ (*e.g.*, a FACE has one NOSE). An $\mathrm{IndBern}$ distribution is useful to model a situation in which there is a set of bricks of type $A_{(r,i)}$ that a brick of type $A_{(r,0)}$ can generate, and elements from the set are selected independently (*e.g.*, an EYE may have any number of EYELASHES above the EYE).

Note that unlike a context-free grammar model used in natural language processing, a scene grammar has no start symbol and instead we have self-rooting probabilities. We also make no distinction between terminal and non-terminal symbols, and allow for rules with empty right-hand-side. A scene generated by a scene grammar is defined in terms of a finite set of available bricks.

Definition 4 *The bricks defined by a grammar $\mathcal{G}$ are pairs of symbols and poses,*

$$\mathcal{B} = \{\, (A, \omega) \mid A \in \Sigma, \omega \in \Omega_A \,\}.$$

Definition 5 *A scene S is defined by:*

1. *A set $O \subseteq \mathcal{B}$ of bricks that are present in the scene.*
2. *For each brick $(A_0, \omega) \in O$ we have a rule $r = A_0 \to A_1, \ldots, A_n \in \mathcal{R}_{A_0}$ and $\forall\, 1 \leq i \leq n_r$, we have a value $W_i \in \{0,1\}^{\Omega_{A_i}}$ such that $\forall z \in I(W_i)$, $(A_i, z) \in O$. We say that a brick (A_0, ω) expands to, or is a parent of, the set of bricks $\{(A_i, z) \mid 1 \leq i \leq n_r, z \in I(W_i)\}$.*

Let $\mathcal{S}$ be the set of scenes defined by a scene grammar $\mathcal{G}$. The set $\mathcal{S}$ is the "Language" generated by $\mathcal{G}$. To generate a scene we consider a random algorithm that grows a scene starting from an initial set of random bricks.

The scene generation process starts from an initial set of bricks that are included in the scene independently at random. We then repeatedly expand bricks in the scene that have not been expanded before. The expansion of a brick generates new bricks that are added to the scene and expanded further. This random algorithm defines a distribution, $p(S)$, that can capture regularities in natural

scenes. For example, the process can capture which objects tend to co-occur in a scene and the typical relative positions between different objects.

To formally define the scene generation process we use a set O to keep track of bricks in the scene and a set Q to keep track of bricks that are in the scene but have not been expanded yet. Initial bricks are included in O independently according to self-rooting probabilities. All of these bricks are queued for expansion in Q. If an expansion generates a brick that is not already in O we add the brick to O and queue it for expansion in Q.

Definition 6 *A probabilistic scene grammar $\mathcal{G}$ defines a random algorithm for generating scenes:*

1. *Initially $O = \emptyset$ and $Q = \emptyset$.*
2. *For each brick $(A, \omega) \in \mathcal{B}$ we add (A, ω) to O and Q with probability ϵ_A.*
3. *While $Q \neq \emptyset$ we remove a brick (A, ω) from Q and expand it.*
4. *Expanding (A, ω) involves*
   (a) *sampling a rule $r = A_0 \to A_1, \ldots, A_n \in \mathcal{R}_A$ according to q_A,*
   (b) *for $1 \leq i \leq n_r$, sampling a set W_i of binary values according to $\gamma_{(\omega,r,i)}(W_i \mid \theta_{(\omega,r,i)})$, and for each $z \in I(W_i)$, if $(A_i, z) \notin O$ adding it to both O and Q.*

The scene S is defined by O and the choices made when expanding each brick in O.
The output of this algorithm defines a distribution $p(S)$ over scenes in $\mathcal{S}$.

We note that the scene generation algorithm terminates after a finite number of expansions bounded by the total number of bricks in $\mathcal{B}$. As discussed above the queue Q keeps track of bricks that are in the scene but have not been expanded yet. When Q is empty every brick in O has been expanded exactly once. Therefore when the process terminates we have a scene $S \in \mathcal{S}$.

We also note that the order in which the bricks from Q are selected for expansion does not affect the probability of generating a particular scene. Therefore the arbitrary choice of expansion order does not change the distribution over scenes defined by the algorithm.

Remark 7 *Scene grammars are related to context-free grammars used in language modeling. We note however that they generate different types of structures.*

Recall that a context-free grammar generates rooted derivation trees, where the vertices are labeled with symbols from a finite alphabet. In a derivation tree there is a single vertex (the root) with no parents and every other vertex has a unique parent.

A scene generated by a scene grammar defines a directed graph G over the bricks that are present in the scene. The edges of the scene graph capture the parent relationship over the bricks in a scene. We note that G resembles a derivation tree, but it has more general connectivity structure. In particular we can have multiple vertices with no parents (roots) in G, and the graph can have multiple disjoint components. We can also have vertices with multiple parents in G. Therefore multiple roots can lead to the same vertex and there can be multiple paths from one vertex to another. The scene graph will also have directed cycles when a brick (A, ω) in the scene leads to a sequence of expansions that eventually generate (A, ω) again.

Finally we note that every scene graph is a subgraph of the complete directed graph over $\mathcal{B}$, and the number of possible scene graphs is finite (although it can be very large). This is in contrast to the fact that a context-free grammar can generate trees of unbounded size.

Chapter 3

Example grammars

In this chapter we give some examples of PSGs and illustrate the random scenes they generate.

Recall that a PSG $\mathcal{G}$ is defined by a 6-tuple $(\Sigma, \Omega, \mathcal{R}, q, \epsilon, \gamma)$. In the examples below we combine the description of $\mathcal{R}$, q, and γ to simplify the notation.

Let $r = A_0 \to A_1, \ldots, A_n$ be a rule in $\mathcal{R}$. To specify the rule r, the rule selection probability q_r, and the conditional pose distributions associated with r, we write,

$$q_r, \quad (A_0, \omega_0) \quad \to \quad (A_1, \gamma_{(\omega_0,r,1)}(\cdot|\theta_{(\omega_0,r,1)})), \ldots, (A_n, \gamma_{(\omega_0,r,n)}(\cdot|\theta_{(\omega_0,r,n)})). \tag{3.1}$$

In the examples in this chapter, the pose spaces are grids of integer points $[N_1] \times \cdots \times [N_D]$ where $[N] = \{0, \ldots, N-1\}$. Denote such a pose space by Υ. Below, we use $\mathrm{Rect}(a, b)$ to indicate the set of grid points in the hyperrectangle with diagonal (a, b).

We define two special kinds of $\mathrm{Categorical}$ distributions; the $\mathrm{UniformRect}$ distribution and a distribution concentrated at a single point in the grid. We also define a special kind of $\mathrm{IndBern}$ distribution: a $\mathrm{UniformBern}$ (short for $\mathrm{Uniform\text{-}Independent\text{-}Bernoullis}$) distribution.

Definition 8 *Let W be a set of binary random variables indexed by Υ. Let a and b be two elements of Υ. We define the* $\mathrm{UniformRect}$ *distribution as*

$$\mathrm{UniformRect}(W; a, b) = \begin{cases} \frac{1}{|\mathrm{Rect}(a,b)|}, & \sum_{k \in \Upsilon} W_k = 1, I(W) \subseteq \mathrm{Rect}(a, b) \\ 0, & \textit{otherwise} \end{cases} \tag{3.2}$$

where $|\mathrm{Rect}(a, b)|$ denotes the size of the set $\mathrm{Rect}(a, b)$.

We denote a distribution concentrated at a single point by

$$\delta(W; a) = \mathrm{UniformRect}(W; a, a).$$

Definition 9 *Let W be a set of binary random variables indexed by Υ and let $T \subseteq \Upsilon$ be a set. We define the* UniformBern *distribution as*

$$\text{UniformBern}(W; T, \theta) = \prod_{k \in \Upsilon} \theta_k^{W_k} \times (1 - \theta_k)^{1 - W_k} \tag{3.3}$$

$$\theta_k = \begin{cases} \theta, & k \in T \\ 0, & \textit{otherwise}. \end{cases} \tag{3.4}$$

For brevity, for the rest of this thesis we drop the argument W from the distributions above, and will denote them as $\text{UniformRect}(a, b)$, $\delta(a)$, and $\text{UniformBern}(T, \theta)$.

3.1 Scenes with curves

Grammar 1 generates scenes with discrete curves. Figure 3.1 shows some images generated by this model. The grammar generates scenes with a random number of curves and where each curve has a random length and shape, giving preference to curves with low-curvature. The approach is related to the Elastica model in [41] where the tangent function of a random curve is defined by a random walk. In Chapter 6 we show how this model can be used for contour completion and in Chapter 9 we show how the model can be used to detect curves in noisy images.

A curve is represented by a sequence of oriented elements. Curves are extended one element at a time, moving from one pixel in the image to a neighboring pixel in a direction close to the current orientation. At each step a curve can also end or change orientation with small probability. As a curve is generated the process leaves a trace of ink in the image.

The grammar has two symbols, $\Sigma = \{\mathsf{CURVE}, \mathsf{INK}\}$. The CURVE bricks represent oriented elements that are connected sequentially to form curves. The pose of a CURVE brick specifies a pixel location and one of 8 possible orientations. The INK bricks represent the pixels that are covered by a curve and capture what we see in an image. The pose of an INK brick specifies only a pixel location and has no orientation information.

Grammar 1 *A grammar for 2D images with curves. The function T_θ denotes a rotation in the plane by an angle θ and* Round *maps a point in the plane to the nearest grid point.*

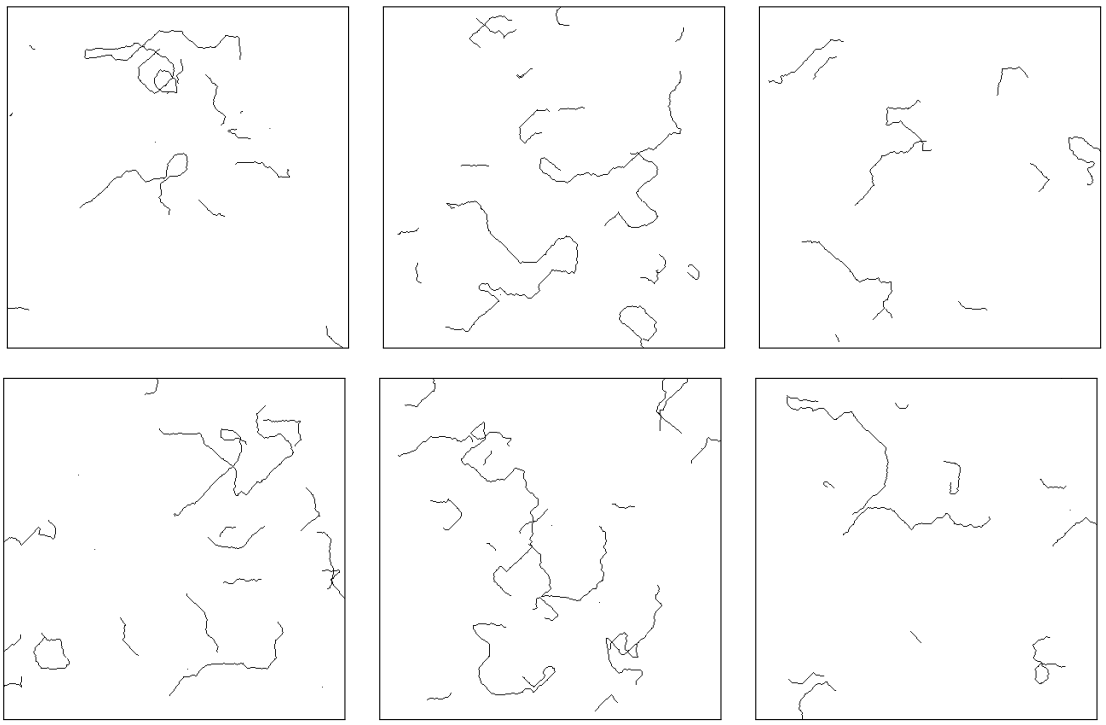


Figure 3.1: Random images generated using Grammar 1. The black pixels represent the INK bricks that are present in a random scene. The grammar generates discrete curves of varying lengths and shapes, giving a preference to curves with low curvature.

$\Sigma = \{\mathsf{CURVE}, \mathsf{INK}\}$.

$\Omega_{\mathsf{CURVE}} = [N] \times [M] \times [8]$.

$\Omega_{\mathsf{INK}} = [N] \times [M]$.

Rules:

$$
\begin{array}{lrl}
0.65, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow (\mathsf{INK}, \delta((x, y))), (\mathsf{CURVE}, \delta(((x, y) + \mathrm{Round}(T_\theta(1, 0)), \theta))) \\
0.10, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow (\mathsf{INK}, \delta((x, y))), (\mathsf{CURVE}, \delta(((x, y) + \mathrm{Round}(T_\theta(1, -1)), \theta))) \\
0.10, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow (\mathsf{INK}, \delta((x, y))), (\mathsf{CURVE}, \delta(((x, y) + \mathrm{Round}(T_\theta(1, +1)), \theta))) \\
0.05, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow (\mathsf{CURVE}, \delta((x, y, \theta + 1))) \\
0.05, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow (\mathsf{CURVE}, \delta((x, y, \theta - 1))) \\
0.05, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow (\mathsf{INK}, \delta((x, y))), \\
1.00, & (\mathsf{INK}, (x, y)) & \rightarrow \emptyset
\end{array}
$$

$\epsilon_{\mathsf{CURVE}} = \epsilon_{\mathsf{INK}} = 10^{-4}$.

The first three rules that can be used to expand a CURVE brick capture the possible extensions of a curve along a direction that is close to the current orientation. When we extend a curve at pixel (x, y) with orientation θ, we move to one of 3 neighbors of (x, y) that are approximately in the direction θ. Figure 3.2 illustrates the possible extensions for a horizontal element.

The last three rules that can be used to expand a CURVE brick capture changes in orientation and the ending of a curve. The probability of changing the current orientation is small, so curves tend to take multiple steps along a single discrete orientation before turning. As we generate a curve we also generate INK bricks tracing the path of the curve.

3.2 Scenes with faces

Grammar 2 generates scenes with faces and parts of faces. The model captures the notion that each scene has a variable number of objects, and that faces have certain parts at appropriate locations. We also allow for parts of faces to appear on their own, capturing the notion that a scene is made up of a set of faces and other components that look like parts of faces. Figure 3.3 shows some examples of random scenes generated by this grammar.

In this grammar the pose of a brick specifies a 2D location for an object of fixed size and orientation. The parameters N and M denote the number of pixels in each dimension of a 2D image.

The compositional model for a face is captured by the rule FACE $\rightarrow$ EYE, EYE, NOSE, MOUTH. When expanding a FACE brick, the possible locations for the parts are defined relative to the face location. In the grammar considered here, the location of each part is selected uniformly at random from a rectangular region defined relative to the location of the face. Figure 3.4 shows the possible locations for the parts when a face is at the origin. This part-based representation for a face captures

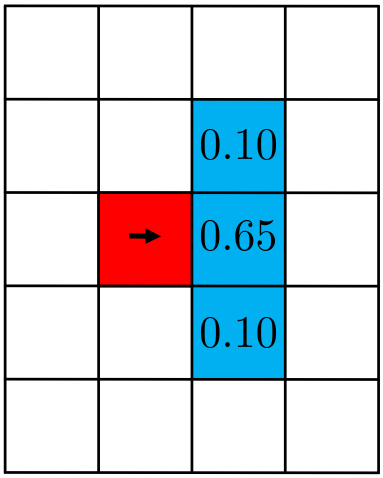


Figure 3.2: A depiction of the possible extensions of a curve by one pixel. In this case the horizontal CURVE brick indicated by the red pixel expands to a CURVE brick of the same orientation in one of the blue pixels with the indicated probabilities. The remaining probability mass is reserved for the choice to end the curve or change its orientation.

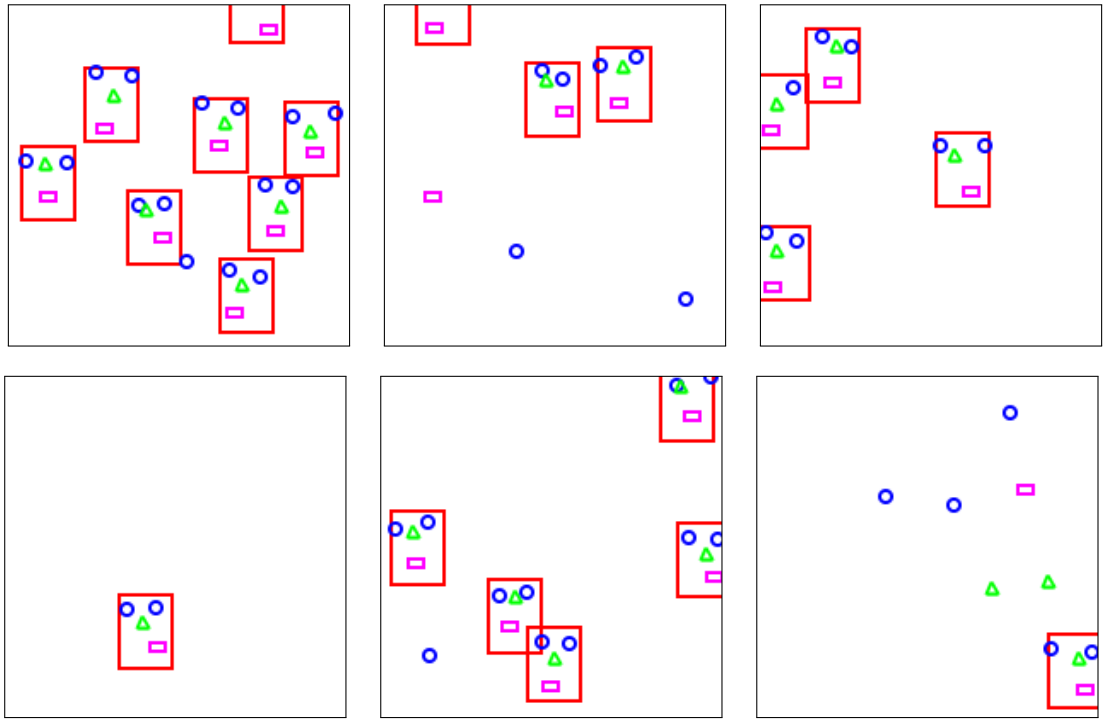


Figure 3.3: Random scenes with faces and parts of faces generated using Grammar 2. Faces are represented by red rectangles, eyes by blue circles, noses by green triangles, and mouths by magenta rectangles. Scenes have multiple objects and parts of faces can appear both in the context of a face and on their own. The location of a part, such as the nose, can vary within a range of possible locations relative to the face.

pairwise relationships between locations of different parts and is similar to a Pictorial Structures model ([18, 13]).

Grammar 2 *A grammar for 2D scenes with faces and parts of faces:*

$\Sigma = \{\mathsf{FACE}, \mathsf{EYE}, \mathsf{NOSE}, \mathsf{MOUTH}\}$.

$\forall A \in \Sigma,\ \Omega_A = [N] \times [M]$.

Rules:

$$
\begin{array}{llll}
1.0, & (\mathsf{FACE}, \omega) & \rightarrow & (\mathsf{EYE}, \mathrm{UniformRect}(\omega + a_1, \omega + b_1)), \\
 & & & (\mathsf{EYE}, \mathrm{UniformRect}(\omega + a_2, \omega + b_2)), \\
 & & & (\mathsf{NOSE}, \mathrm{UniformRect}(\omega + a_3, \omega + b_3)), \\
 & & & (\mathsf{MOUTH}, \mathrm{UniformRect}(\omega + a_4, \omega + b_4)) \\
1.0, & (\mathsf{EYE}, \omega) & \rightarrow & \emptyset \\
1.0, & (\mathsf{NOSE}, \omega) & \rightarrow & \emptyset \\
1.0, & (\mathsf{MOUTH}, \omega) & \rightarrow & \emptyset
\end{array}
$$

$\epsilon_{\mathsf{FACE}} = 10^{-4}$,

$\epsilon_{\mathsf{EYE}} = \epsilon_{\mathsf{NOSE}} = \epsilon_{\mathsf{MOUTH}} = 10^{-5}$.

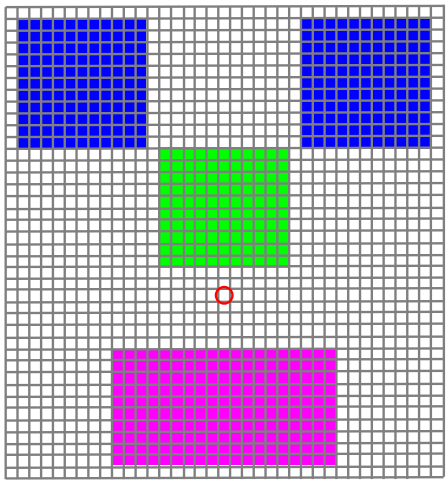


Figure 3.4: A depiction of the possible locations of the face parts when the FACE is located at the pixel indicated by the red circle. The blue, green, and magenta pixels indicate the possible locations for the EYE, NOSE, and MOUTH symbols, respectively.

We note that Grammar 2 can be extended to represent objects of different sizes and orientations by augmenting the pose spaces with scale and orientation information. In Chapter 9, we show how a similar model can be used for face detection. The grammar defined above can also be extended to

define scenes with multiple objects from different categories where each object is defined in terms of a set common parts.

3.3 Scenes with binary segmentation masks

Grammar 3 generates a binary segmentation mask. The foreground generated by this grammar is a single, non-empty connected component of pixels where connections are considered in an 8-neighbourhood around a grid point. Figure 3.5 shows some images generated by this grammar. As shown in the figure, the foreground generated may have "holes" in it.

Grammar 3 *A grammar for 2D foreground/background image segmentation for an $N \times M$ scene:*

$\Sigma = \{\mathsf{SEED}, \mathsf{FG}\}$.
$\Omega_{\mathsf{SEED}} = [1]$.
$\Omega_{\mathsf{FG}} = [N] \times [M]$.
Rules:

$$1.0, \quad (\mathsf{SEED}, \omega) \quad \rightarrow \quad (\mathsf{FG}, \mathrm{UniformRect}((1,1),(N,M)))$$

$$1.0, \quad (\mathsf{FG}, \omega) \quad \rightarrow \quad (\mathsf{FG}, \mathrm{UniformBern}(\mathrm{Rect}(\omega - (1,1), \omega + (1,1)) \setminus \omega, 0.25))$$

$\epsilon_{\mathsf{SEED}} = 1$,
$\epsilon_{\mathsf{FG}} = 0$.

Note that $\mathrm{Rect}(\omega - (1,1), \omega + (1,1)) \setminus \omega$ is the set of points in the rectangle with diagonal $(\omega - (1,1), \omega + (1,1))$ excluding the centre of the rectangle. That is, the 8 neighbours of pixel ω.

Grammar 3 can be thought of as assigning a label (foreground or background) to each grid point. The approach is related to an Ising model on a grid, but here we consider the 8-neighbourhood around a grid point rather than the 4-neighbourhood. As in the Ising model, a point and its neighbours are encouraged to have the same label. Unlike the Ising model, however, the assignment of labels to grid points can be formulated in a generative process that is guaranteed to produce a single, non-empty, connected foreground component. Further, the set of labelings that can be produced by the grammar is exactly the set of labelings such that there is a single, non-empty connected foreground component.

The grammar has two symbols, $\Sigma = \{\mathsf{SEED}, \mathsf{FG}\}$. Intuitively, the SEED symbol selects a location in the image from which to start growing the foreground. This guarantees that there is at least one grid point labelled foreground in the image. Each grid point labelled foreground selects a subset of its 8 neighbours to be foreground as well; each of its neighbours is considered independently and selected with probability 0.25. Figure 3.6 illustrates for a given FG brick, the set of other FG bricks it can generate and with what probability it does so. Since $\epsilon_{\mathsf{FG}} = 0$, and the generative process of expanding a brick (FG, ω) considers selecting other FG bricks in an 8-neighbourhood around ω,

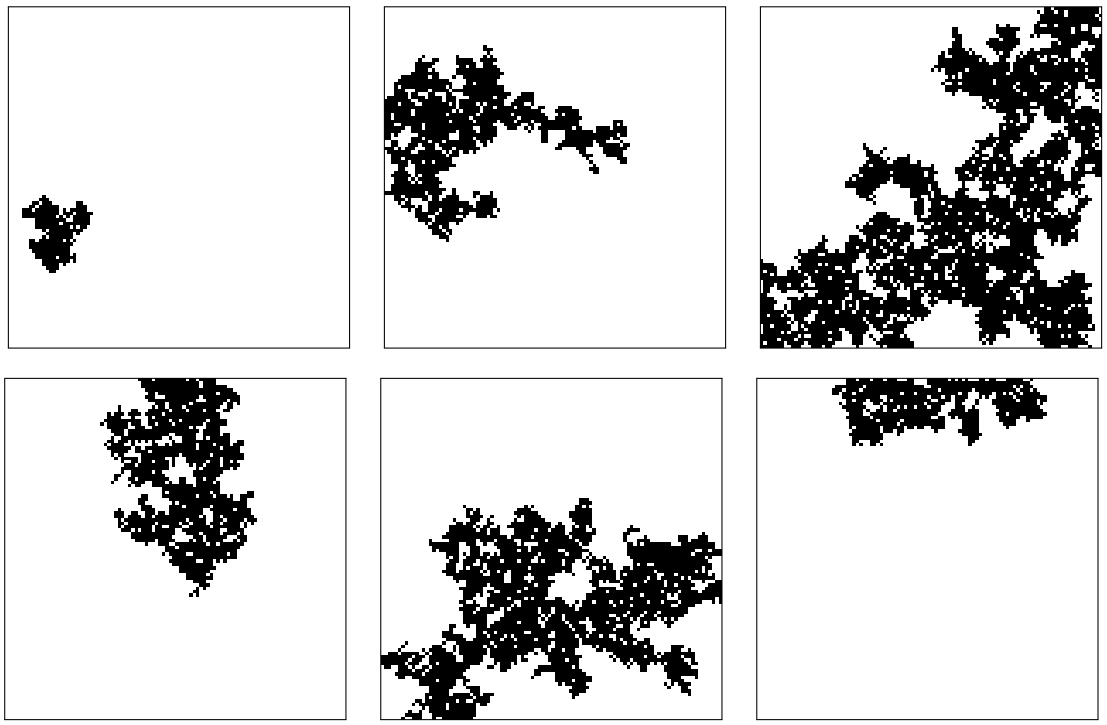


Figure 3.5: Random scenes generated using Grammar 3. The black pixels represent the FG bricks that are present in a random scene. The model generates a single, non-empty, connected (in the 8-neighbourhood sense) foreground segment.

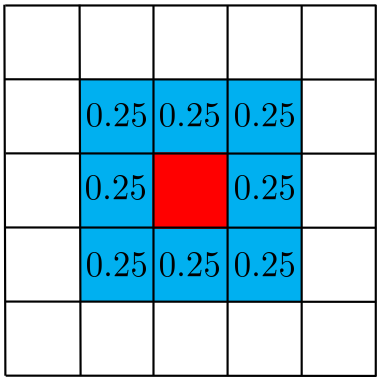


Figure 3.6: A depiction of the possible generations of a FG brick. Indicated in red is the location of the FG brick being expanded. The possible FG bricks that may be generated by this brick are indicated in blue, and the probability of generation is shown. The potentially generated bricks are considered independently.

the generative process is guaranteed to create a single connected component. Note that the expected number of bricks a brick (FG, ω) expands to is 2, and so one may be concerned that the generative process will never terminate. Recall that in the generative process described in Definition 6, a brick can only be expanded once, so the generative process will terminate with probability 1. We will expand at most $N \times M$ FG bricks, and 1 SEED brick.

Chapter 4

Factor Graph Representation

A scene grammar defines a probability distribution, $p(S)$, over scenes. Here we describe a factorization of $p(S)$ and a representation of this distribution by a factor graph with a finite number of binary random variables. In practice the factor graph representation can be used as a data structure for inference. In particular we can use this representation for computing posterior marginals with Loopy Belief Propagation (Chapter 5). The factor graph formulation can also be used for learning model parameters with an approximate EM algorithm (Chapter 8).

We start by considering a representation of scenes using a finite set of binary random variables.

Definition 10 *For a brick $(A, \omega) \in \mathcal{B}$, a rule $r \in \mathcal{R}_A$, and $1 \leq i \leq n_r$. Define*

$$\Gamma_{(\omega,r,i)} = \{\omega' \mid \theta_{(\omega,r,i,\omega')} > 0\}.$$

Note that $\Gamma_{(\omega,r,i)} \subseteq \Omega_{A_{(r,i)}}$ since $\Omega_{A_{(r,i)}}$ indexes $\theta_{(\omega,r,i)}$. The set $\{(A_{(r,i)}, z) \mid z \in \Gamma_{(\omega,r,i)}, 1 \leq i \leq n_r\}$ is the set of bricks that brick $(A_{(r,0)}, \omega)$ can generate when rule r is chosen.

Definition 11 *A scene S generated by a grammar $\mathcal{G}$ defines a collection of binary random variables associated with each brick $(A, \omega) \in \mathcal{B}$,*

$$\mathbf{X}(A, \omega) \in \{0, 1\}, \tag{4.1}$$

$$\mathbf{R}(A, \omega) = \{\, \mathbf{R}(A, \omega, r) \in \{0, 1\} \mid r \in \mathcal{R}_A \,\}, \tag{4.2}$$

$$\mathbf{C}(A, \omega) = \{\, \mathbf{C}(A, \omega, r, i, \omega') \in \{0, 1\} \mid r \in \mathcal{R}_A, 1 \leq i \leq n_r, \omega' \in \Gamma_{(\omega,r,i)} \,\} \tag{4.3}$$

where

$\mathbf{X}(A, \omega) = 1$ *if* (A, ω) *is in the scene,*

$\mathbf{R}(A, \omega, r) = 1$ *if rule* r *is used to expand* (A, ω)*,*

$\mathbf{C}(A, \omega, r, i, \omega') = 1$ *if* (A, ω) *is expanded with rule* r*, and brick* $(A_{(r,i)}, \omega')$ *is one of the bricks generated when considering the* i*-th symbol in the RHS of the rule.*

We note that a scene S is uniquely defined by the value of the random variables $\{\mathbf{X}, \mathbf{R}, \mathbf{C}\}$.

Let $\mathcal{G}$ be a scene grammar. We say $\mathcal{G}$ is *acyclic* if there is no sequence of expansions that generates a brick starting from itself. To make this notion precise let H be a directed graph over the bricks, with an edge from (A, ω) to (B, z) if we can generate (B, z) from (A, ω) in one expansion. The grammar $\mathcal{G}$ is acyclic if H is acyclic. For example, the grammar for scenes with faces in Section 3.2 is acyclic. On the other hand, the grammar for scenes with curves in Section 3.1 is cyclic, because a sequence of expansions starting from a CURVE brick can generate the initial brick again.

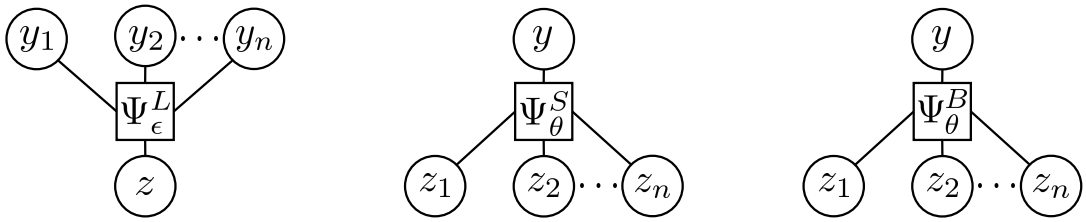
A *topological ordering* of $\mathcal{B}$ is a linear ordering of $\mathcal{B}$ such that (A, ω) appears before (B, z) whenever (A, ω) can generate (B, z) after one or more expansions. We note that when $\mathcal{G}$ is acyclic there is always a topological ordering of $\mathcal{B}$ and such ordering can be computed by topological sorting the vertices of H.

4.1 Factorization

Let $p(\mathbf{X}, \mathbf{R}, \mathbf{C})$ denote the distribution defined by the scene generation algorithm. For an acyclic grammar the distribution $p(\mathbf{X}, \mathbf{R}, \mathbf{C})$ can be factored into a product of local potential functions. The factorization gives a simple closed form expression for $p(\mathbf{X}, \mathbf{R}, \mathbf{C})$ and leads to a factor graph representation that can be used for inference with a scene grammar. The factorization described here is analogous to the expression of the joint distribution in a Bayesian network. We note that the factorization is only exact for acyclic grammars but it can also be used in practice as an approximation for inference with cyclic grammars.

There are three types of factors in the factorization of $p(\mathbf{X}, \mathbf{R}, \mathbf{C})$. Below, for a set of binary values W, let $c(W)$ be the number of ones in W. The three types of factors are illustrated in Figure 4.1 and defined below.

Definition 12 *A Leaky-OR potential* $\Psi^L_\epsilon(Y, z)$ *is a function of a set of binary inputs* $Y = \{y_1, \cdots, y_n\}$ *and a binary output* z*. It represents the conditional probability of each possible output in a probabilistic OR gate. If* $c(Y) > 0$ *we have* $z = 1$ *with probability 1. If* $c(Y) = 0$ *we have* $z = 1$ *with probability* ϵ*.*



(a) Leaky-OR factor. (b) Selection factor. (c) Berns factor.

Figure 4.1: The three types of factors in the factorization of $p(\mathbf{X}, \mathbf{R}, \mathbf{C})$.

$$\Psi^L_\epsilon(Y, z) = \begin{cases} 1 & z = 1,\ c(Y) > 0, \\ 0 & z = 0,\ c(Y) > 0, \\ \epsilon & z = 1,\ c(Y) = 0, \\ 1 - \epsilon & z = 0,\ c(Y) = 0. \end{cases}$$

Definition 13 *A Selection potential $\Psi^S_\theta(y, Z)$ is a function of a binary input y and a set of binary outputs $Z = \{z_1, \cdots, z_n\}$. This factor models the selection of a random output. If $y = 0$, then the output Z such that $c(Z) = 0$ is selected with probability 1. If $y = 1$, then exactly one of the z_i has value 1. The choice of which z_i to set to 1 (select) is governed by the probabilities defined by θ.*

$$\Psi^S_\theta(y, Z) = \begin{cases} 1 & y = 0,\ c(Z) = 0, \\ 0 & y = 0,\ c(Z) > 0, \\ \text{Categorical}(Z \mid \theta) & y = 1. \end{cases}$$

Definition 14 *A Berns potential $\Psi^B_\theta(y, Z)$ is a function of a binary input y and a set of binary outputs $Z = \{z_1, \cdots, z_n\}$. This factor models the selection of multiple outputs conditional on y. If $y = 0$, then the output Z such that $c(Z) = 0$ is selected with probability 1. If $y = 1$, then $z_i = 1$ with probability θ_{z_i}.*

$$\Psi^B_\theta(y, Z) = \begin{cases} 1 & y = 0,\ c(Z) = 0, \\ 0 & y = 0,\ c(Z) > 0, \\ \text{IndBern}(Z \mid \theta) & y = 1. \end{cases}$$

Our main observation is that $p(\mathbf{X}, \mathbf{R}, \mathbf{C})$ can be expressed in closed form in terms of a product of potentials of the types defined above.

To formulate the factorization we consider the following collections of random variables,

$$\begin{aligned}
\mathbf{C}(A,\omega,r,i) &= \{\, \mathbf{C}(A,\omega,r,i,\omega') \mid \omega' \in \Gamma_{(\omega,r,i)} \,\}, \\
\mathrm{par}(\mathbf{X}(A,\omega)) &= \{\, \mathbf{C}(B,\omega',r,i,\omega) \mid B \in \Sigma, \omega' \in \Omega_B, r \in \mathcal{R}_B, 1 \leq i \leq n_r, A_{(r,i)} = A, \\
&\qquad \omega \in \Gamma_{(\omega',r,i)} \,\}.
\end{aligned}$$

The set $\mathbf{C}(A,\omega,r,i)$ includes all the poses that can be associated with the i-th child of brick (A,ω) if rule r is used to expand (A,ω). The set $\mathrm{par}(\mathbf{X}(A,\omega))$ includes all the random variables that can indicate a parent of $\mathbf{X}(A,\omega)$ in the scene.

Proposition 15 *The distribution $p(\mathbf{X},\mathbf{R},\mathbf{C})$ defined by an acyclic grammar $\mathcal{G}$ can be expressed as,*

$$p(\mathbf{X},\mathbf{R},\mathbf{C}) = \prod_{(A,\omega)\in\mathcal{B}} \left(p(\mathbf{X}(A,\omega) \mid \mathrm{par}(\mathbf{X}(A,\omega))) p(\mathbf{R}(A,\omega) \mid \mathbf{X}(A,\omega)) \prod_{\substack{r\in\mathcal{R}_A, \\ 1\leq i\leq n(r)}} p(\mathbf{C}(A,\omega,r,i) \mid \mathbf{R}(A,\omega,r)) \right). \tag{4.4}$$

Moreover if $\mathcal{G}$ is acyclic we have,

$$p(\mathbf{X}(A,\omega) = z \mid \mathrm{par}(\mathbf{X}(A,\omega)) = Y) = \Psi^L_{\epsilon_A}(Y,z) \tag{4.5}$$

$$p(\mathbf{R}(A,\omega) = Z \mid \mathbf{X}(A,\omega) = y) = \Psi^S_{q_A}(y,Z) \tag{4.6}$$

$$p(\mathbf{C}(A,\omega,r,i) = Z \mid \mathbf{R}(A,\omega,r) = y) = \Psi^S_{\theta_{(\omega,r,i)}}(y,Z) \text{ or } \Psi^B_{\theta_{(\omega,r,i)}}(y,Z) \tag{4.7}$$

Proof Let $\mathbf{V}_i = \{\mathbf{X}_i, \mathbf{R}_i, \mathbf{C}_i\}$ denote the random variables associated with the i-th brick in a topological ordering of $\mathcal{B}$. We can write

$$p(\mathbf{X},\mathbf{R},\mathbf{C}) = \prod_i p(\mathbf{V}_i \mid \mathbf{V}_{j<i}) \tag{4.8}$$

$$= \prod_i p(\mathbf{X}_i \mid \mathbf{V}_{j<i}) p(\mathbf{R}_i \mid \mathbf{X}_i, \mathbf{V}_{j<i}) p(\mathbf{C}_i \mid \mathbf{R}_i, \mathbf{X}_i, \mathbf{V}_{j<i}) \tag{4.9}$$

Based on the definition of the scene generation algorithm, and using the topological ordering constraint we can see that

$$p(\mathbf{X}_i \mid \mathbf{V}_{j<i}) = p(\mathbf{X}_i \mid \mathrm{par}(\mathbf{X}_i)), \tag{4.10}$$

$$p(\mathbf{R}_i \mid \mathbf{X}_i, \mathbf{V}_{j<i}) = p(\mathbf{R}_i \mid \mathbf{X}_i), \tag{4.11}$$

$$p(\mathbf{C}_i \mid \mathbf{R}_i, \mathbf{X}_i, \mathbf{V}_{j<i}) = p(\mathbf{C}_i \mid \mathbf{R}_i). \tag{4.12}$$

This leads to the factorization pf $p(\mathbf{X}, \mathbf{R}, \mathbf{C})$ above. The expression of each of the factors using the Leaky-OR, Selection, and Berns potentials also follows directly from the definition of the scene generation algorithm and the topological ordering constraint. ■

4.2 Graphical Model

A factor graph $\mathcal{F}$ ([33]) is a bipartite undirected graph that represents a factored probability distribution. The factor graph has a set of variable nodes V and a set of factor nodes F. Let x denote an outcome for the random variables in V. For $U \subseteq V$ we use x_U to denote the values of the random variables in U. Associated with each factor node $f \in F$ is a non-negative potential function Ψ_f. Let $N(f)$ denote the neighbors of $f \in F$. The potential Ψ_f is a function of $x_{N(f)}$. The factor graph $\mathcal{F}$ defines a joint distribution

$$Q(x) = \frac{1}{Z} \prod_{f \in F} \Psi_f(x_{N(f)}). \tag{4.13}$$

The factorization in Eqn. 4.4 suggests the following construction which leads to an exact representation for acyclic grammars and an approximation (or an alternative model) for cyclic grammars.

Definition 16 *Let $\mathcal{G}$ be a scene grammar. We define the factor graph $\mathcal{F} = (V \cup F, E)$ as follows:*

1. *The variable nodes V correspond to the random variables associated with each brick.*
2. *For each brick $(A, \omega) \in \mathcal{B}$ we have a factor node $f^1_{(A,\omega)}$ with potential function $\Psi^L_{\epsilon_A}$ connected to a set of input variables $\mathrm{par}(\mathbf{X}(A, \omega))$ and an output variable $\mathbf{X}(A, \omega)$.*
3. *For each brick $(A, \omega) \in \mathcal{B}$ we have a factor node $f^2_{(A,\omega)}$ with potential function $\Psi^S_{q_A}$ connected to an input variable $\mathbf{X}(A, \omega)$ and a set of output variables $\mathbf{R}(A, \omega)$.*
4. *For each brick $(A, \omega) \in \mathcal{B}$, rule $r \in R_A$, and $1 \leq i \leq n(r)$ we have a factor node $f^3_{(A,\omega,r,i)}$ with potential function $\Psi^S_{\theta_{(\omega,r,i)}}$ or $\Psi^B_{\theta_{(\omega,r,i)}}$ connected to an input variable $\mathbf{R}(A, \omega, r)$ and a set of output variables $\mathbf{C}(A, \omega, r, i)$. If $f^3_{(A,\omega,r,i)}$ is to represent a* Categorical *conditional pose distribution, then $f^3_{(A,\omega,r,i)}$ has potential function $\Psi^S_{\theta_{(\omega,r,i)}}$. Otherwise, it has potential function $\Psi^B_{\theta_{(\omega,r,i)}}$.*

In practice, we attach unary potentials to the random variables $\mathbf{X}$ that can be used as an "external field" in Q. For example, we can attach a unary potential to the random variable $X(\mathsf{FACE}, (3, 4))$ to encode the image evidence for a face being present at location $(3, 4)$ in the scene.

Figure 4.2 illustrates the variables and factors in $\mathcal{F}$ that are associated with a single brick in the general case. To provide a concrete example of the part of the factor graph $\mathcal{F}$ corresponding to a single brick, consider the PSG given in Grammar 4. Figure 4.3 illustrates the variables and factors in $\mathcal{F}$ associated with the brick $(B, 3)$ for this particular PSG.

Grammar 4 *A simple acyclic grammar:*

$\Sigma = \{\mathsf{B}, \mathsf{C}, \mathsf{D}\}$.

$\Omega_A = \{1, 2, 3, 4, 5\}, \forall A \in \Sigma$.

Rules:

$$
\begin{array}{llll}
0.5, & (\mathsf{B}, \omega) & \rightarrow & (\mathsf{C}, \mathrm{UniformRect}(\omega - 1, \omega + 1)) \\
0.5, & (\mathsf{B}, \omega) & \rightarrow & \emptyset \\
1.0, & (\mathsf{C}, \omega) & \rightarrow & (\mathsf{D}, \mathrm{UniformRect}(\omega - 1, \omega + 1)) \\
1.0, & (\mathsf{D}, \omega) & \rightarrow & \emptyset
\end{array}
$$

$\epsilon_A = 10^{-4}$

$\epsilon_B = \epsilon_C = 0$

To illustrate the connectivity between blocks of variables in $\mathcal{F}$, consider again the PSG given in Grammar 4. Figure 4.4 shows an illustration of the connectivity between blocks of variables in $\mathcal{F}$ for this grammar. In this figure the set of random variables associated with a brick are grouped together. Note that although the grammar is acyclic, the factor graph has undirected cycles.

Let Q denote the distribution defined by $\mathcal{F}$. When $\mathcal{G}$ is an acyclic grammar, Proposition 15 implies that $Q(\mathbf{X}, \mathbf{R}, \mathbf{C}) = p(\mathbf{X}, \mathbf{R}, \mathbf{C})$. On the other hand, for cyclic grammars the two distributions are not the same. In this case the factor graph $\mathcal{F}$ can be used as an approximation, or as an alternative, to the model defined by the grammar $\mathcal{G}$.

We note that even in the case of an acyclic grammar the factor graph $\mathcal{F}$ will often have undirected cycles. For example, we see cycles in $\mathcal{F}$ whenever there are two different sequences of expansions that can be used to generate one brick from another. We also see cycles when two different bricks can both generate two other bricks. For example, the grammar for scenes with faces described in Section 3.2 is acyclic but the corresponding factor graph has cycles. Figure 4.4 illustrates a concrete example where an acyclic grammar gives rise to a factor graph with undirected cycles.

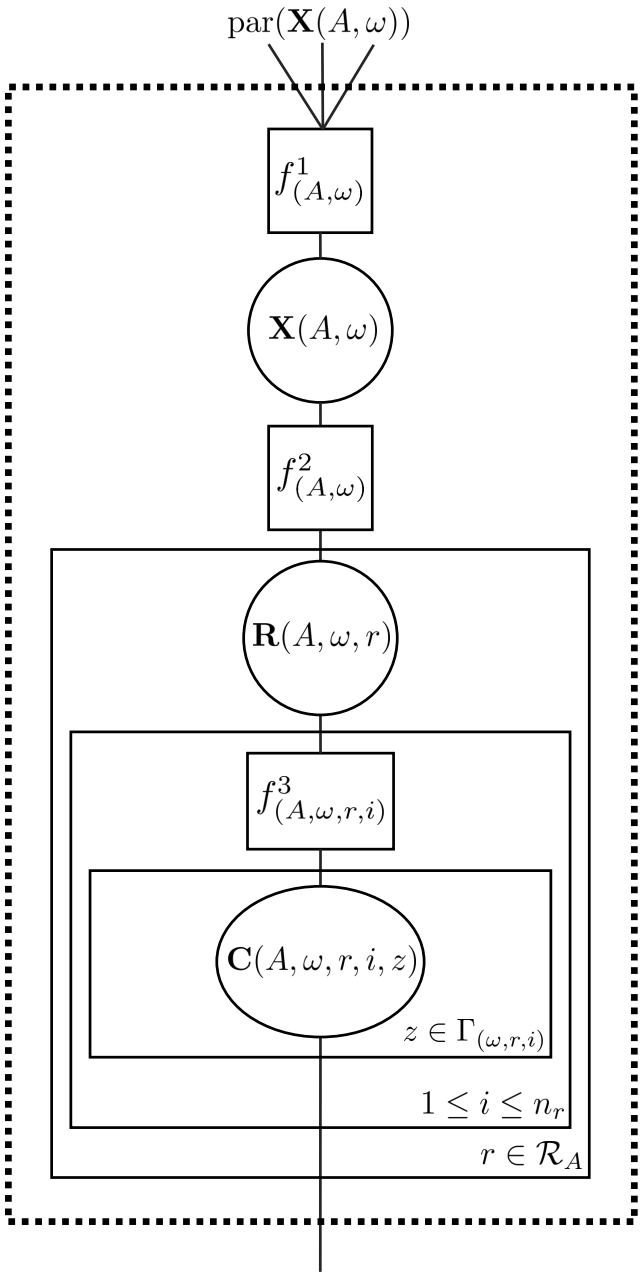


Figure 4.2: The part of the factor graph $\mathcal{F}$ corresponding to a single brick (A, ω) in plate notation.

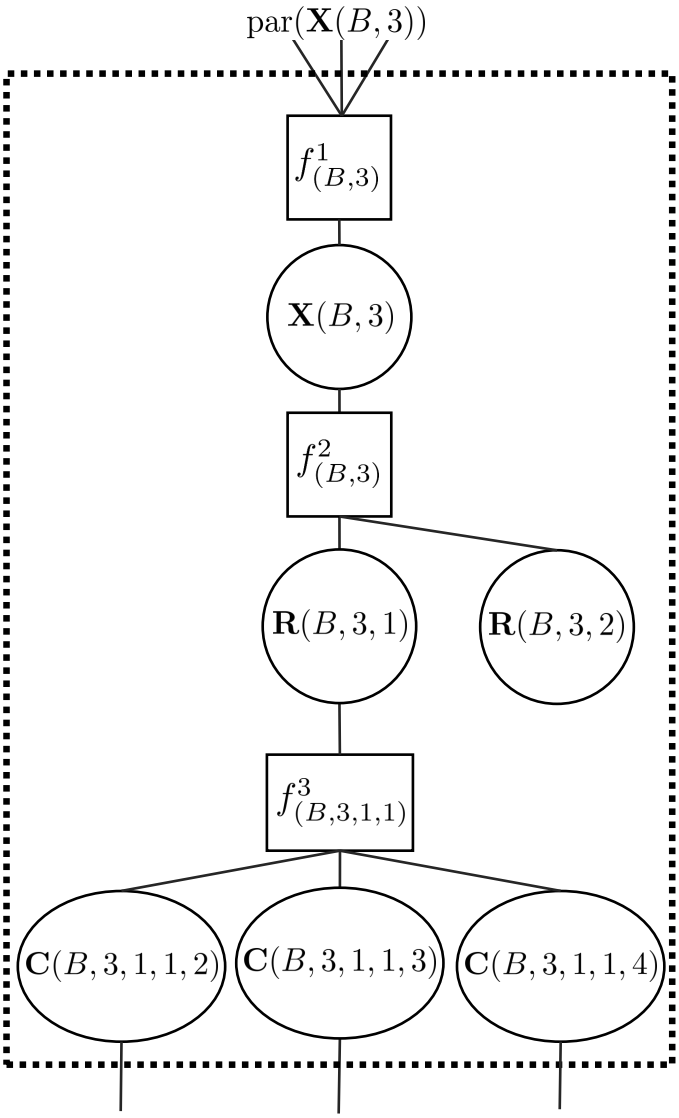


Figure 4.3: The part of the factor graph $\mathcal{F}$ corresponding to the particular brick $(B,3)$ in the factor graph representation of Grammar 4.

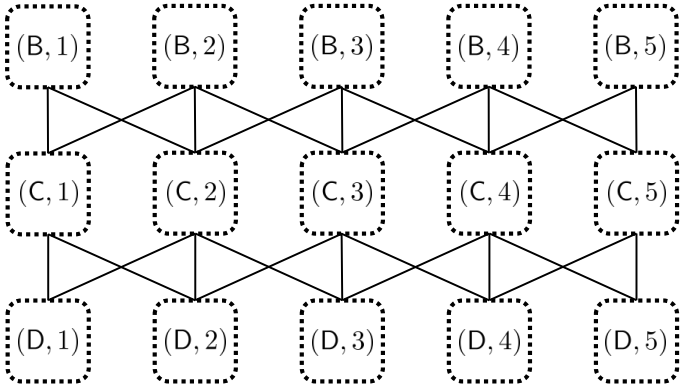


Figure 4.4: An illustration of the connectivity in the factor graph $\mathcal{F}$ constructed from Grammar 4. Each node in this graph represents the part of $\mathcal{F}$ associated with a brick. Note that even though the grammar is acyclic, the factor graph has cycles.

Chapter 5

Inference Using Loopy Belief Propagation

In the PSG framework, the goal of inference is to compute conditional marginal probabilities for all random variables, $\{\mathbf{X}, \mathbf{R}, \mathbf{C}\}$. Since we formulate the PSG framework in terms of a factor graph with undirected loops, we use the well-studied message passing scheme of Loopy Belief Propagation (LBP) (see [42, 33]) to perform approximate inference. The results of LBP can be used to approximate the marginal probabilities of interest.

In this chapter, we derive an efficient message-passing scheme for LBP in the PSG framework. Since our goal is to produce marginal probabilities, we use the sum-product variant of LBP. A similar scheme can be applied in the max-product setting if max-marginals are desired.

5.1 Overview of LBP

LBP on a factor graph is an iterative message-passing scheme that operates by sending non-negative "messages" between neighbouring factor nodes and variables nodes. Messages from a variable node to a factor node indicate that variable node's preferences for its possible states. Messages from a factor node to a variable node indicate that factor node's preference for each of the variable node's possible states. We denote a message from a variable node v to a factor node f concerning state x_v by $\mu_{v \to f}(x_v)$. Messages from a factor node to a variable node are denoted similarly.

LBP proceeds by first initializing all messages followed by repeatedly updating messages between nodes until some convergence criterion is met (*e.g.*, the messages have reached a fixed point). The computation of messages differs depending on whether the message is from a variable node to a factor node or vice versa.

Below, we will normalize messages between the nodes of the factor graph so that they sum to 1. Although this is not strictly necessary, this normalization will simplify the derivation of the message passing equations for the PSG factor graph. Also, in practice, one typically normalizes messages to avoid numerical underflow.

Definition 17 *Let v denote a variable node, $N(v)$ its neighbouring factor nodes, and let f be any element of $N(v)$. The message from a variable node to a factor node is defined to be*

$$\mu_{v\to f}(x_v) \propto \prod_{g\in N(v)\setminus f} \mu_{g\to v}(x_v). \tag{5.1}$$

where the constant of proportionality is chosen so that $\sum_{x_v} \mu_{v\to f}(x_v) = 1$.

Definition 18 *Let f be a factor node, $N(f)$ its neighbouring variable nodes, and Ψ_f the factor node's associated potential function. Let v be any element of $N(f)$. The message from a factor node to a variable node is defined to be*

$$\mu_{f\to v}(x_v) \propto \sum_{x_{N(f)\setminus v}} \Psi_f(x_{N(f)}) \prod_{u\in N(f)\setminus v} \mu_{u\to f}(x_u) \tag{5.2}$$

where the summation is over all possible configurations of $x_{N(f)\setminus v}$ and the constant of proportionality is chosen so that $\sum_{x_v} \mu_{f\to v}(x_v) = 1$.

Definition 19 *After convergence, the marginal probability of a variable node v is approximated as*

$$\hat{p}(x_v) \propto \prod_{f\in N(v)} \mu_{f\to v}(x_v) \tag{5.3}$$

where the constant of proportionality is chosen so that $\sum_{x_v} \hat{p}(x_v) = 1$.

In the case where the factor graph contains no loops, $\hat{p}(x_v)$ matches the true marginal, $Q(x_v)$. In the general case of the factor graph containing loops, $\hat{p}(x_v)$ is only an approximation to $Q(x_v)$. In the context of the PSG framework, using LBP for inference generally produces an approximation to the true marginals since a factor graph constructed in the PSG framework may contain loops.

5.2 Efficient message computation for LBP in the PSG framework

The key computations in LBP are the computations of messages. As such, it is crucial that message computation is efficient. Consider the case in which all variable nodes are binary; generally, the time-complexity for passing messages from factor nodes to binary variable nodes is exponential in the degree of the factor node. The main result of this section is:

Theorem 20 *Consider a factor graph where the variable nodes are binary and all factor potentials are one of: Leaky-OR, Selection, or Berns. The time-complexity to compute messages from all nodes to all of their neighbouring nodes is linear in the number of edges in the factor graph.*

Note that the factor graph construction described in Chapter 4 defines a factor graph consisting of Leaky-OR, Selection, and Berns factor potentials. Hence, Theorem 20 applies to the factor graphs constructed in the PSG framework.

To prove Theorem 20, we will show that for all nodes in such a factor graph, computing the messages from a node to all of its neighbours is linear in the degree of the node. Theorem 20 then follows trivially. In the process of proving Theorem 20 we will derive efficient message-passing schemes for the Leaky-OR, Selection, and Berns factor nodes.

In the proofs below, we make use of the following observation:

Observation 1 *Let* $m = (m_1, \cdots, m_n)$*, and let* $d_i = \prod_{j \neq i} m_j$ *for* $1 \leq i \leq n$*. If* $m_j \neq 0$*,* $1 \leq j \leq n$*, then all the* d_i*'s can be computed jointly in* $\mathcal{O}(n)$ *time.*

Note that d_i *can be written as*

$$d_i = \frac{\prod_j m_j}{m_i} \tag{5.4}$$

since $m_j \neq 0$*,* $1 \leq j \leq n$*. We can first compute* $\prod_j m_j$ *in* $\mathcal{O}(n)$ *time, then compute the* d_i*,'s jointly by applying Eqn. 5.4 in* $\mathcal{O}(n)$ *time*[1].

Theorem 21 *All messages from a binary variable node to all of its neighbouring factor nodes can be computed in time linear in the degree of the variable node.*

Proof Recall the general message passing equation for a variable node to a factor node defined in Definition 17. Consider a variable node v. For each possible value of x_v, apply Observation 1 to compute the messages $\mu_{v \to f}(x_v)\ \forall f \in N(v)$ simultaneously in $\mathcal{O}(|N(v)|)$ time. Since the degree of v is $|N(v)|$, and if v is a binary variable node, then all of the messages from v to all $f \in N(v)$ can be computed simultaneously in time linear in the degree of v.

■

In the remainder of this section we show that the messages from the factor nodes to all of their neighbouring variable nodes in the PSG factor graph can be computed in time linear in the degree

[1] We can use a similar trick to compute the d_i's jointly in $\mathcal{O}(n)$ time even if $\exists i$ such that $m_i = 0$. If exactly only one of the entries is zero, say d_j, then $d_i = 0, \forall j \neq i$, and d_j can be computed directly in $\mathcal{O}(n)$ time from the definition of d_j. If more than one of the entries is zero, then $d_i = 0$, $1 \leq i \leq n$.

of the factor node. For an arbitrary potential, computing the messages from the factor to all of its neighbouring nodes takes time exponential in the degree of the factor. However, if the potential can be expressed in a parametric form with some structure, then it may be possible to perform message computation more efficiently.

5.2.1 Message passing for Leaky-OR factors

Recall the definition of the Leaky-OR potential in Definition 12 and the general message passing equation for factors nodes to variable nodes given in Eqn. 5.2. We will show that exploiting the structure of the Leaky-OR potential allows one to compute all messages from a Leaky-OR factor node to its neighbouring variable nodes in time linear in the degree of the factor node.

Below, let Ψ_ϵ^L be a leaky-OR potential and let f be the corresponding Leaky-OR factor node. As illustrated in Figure 4.1(a), the factor node has neighbouring variable nodes $N(f) = Y \cup \{z\}$. We assume that $\mu_{u\to f}(x_u) > 0$ and $\sum_{x_u} \mu_{u\to f}(x_u) = 1, \forall u \in N(f)$ and $x_u \in \{0, 1\}$.

Theorem 22 *All messages from a Leaky-OR factor node to all of its neighbouring variable nodes can be computed in time linear in the degree of the factor node.*

To prove Theorem 22, we require two lemmas.

Lemma 23 *The messages $\mu_{f\to z}(x_z)$ can be expressed as*

$$\mu_{f\to z}(0) = (1-\epsilon)\prod_{y\in Y}\mu_{y\to f}(0) \tag{5.5}$$

$$\mu_{f\to z}(1) = 1 - \mu_{f\to z}(0). \tag{5.6}$$

Lemma 24 *The messages $\mu_{f\to y}(x_y)\ \forall y \in Y$ can be expressed as*

$$\mu_{f\to y}(0) \propto \mu_{z\to f}(1) + \Big(\prod_{u\in Y\setminus y}\mu_{u\to f}(0)\Big)\big((1-\epsilon)(\mu_{z\to f}(0) - \mu_{z\to f}(1))\big) \tag{5.7}$$

$$\mu_{f\to y}(1) \propto \mu_{z\to f}(1) \tag{5.8}$$

where the constants of proportionality are chosen so that $\mu_{f\to y}(0) + \mu_{f\to y}(1) = 1$.

Proof of Lemma 23

Substituting the form of the Leaky-OR potential defined in Definition 12 into the general message passing equation given in Eqn. 5.2 and noting that the quantity of interest is $\mu_{f\to z}(x_z)$ yields:

$$\mu_{f\to z}(x_z) \propto \sum_{x_Y} \Psi_\epsilon^L(x_Y, x_z) \prod_{y\in Y} \mu_{y\to f}(x_y) \tag{5.9}$$

where the summation is over all possible configurations of x_Y. Consider the case $x_z = 0$:

$$\begin{aligned}
\mu_{f\to z}(0) &\propto \sum_{x_Y} \Psi_\epsilon^L(x_Y, 0) \prod_{y\in Y} \mu_{y\to f}(x_y) && (5.10)\\
&= \sum_{x_Y : c(x_Y)=0} (1-\epsilon) \prod_{y\in Y} \mu_{y\to f}(x_y) && (5.11)\\
&= (1-\epsilon) \prod_{y\in Y} \mu_{y\to f}(0). && (5.12)
\end{aligned}$$

Note that $\Psi_\epsilon^L(x_Y, 1) = 1 - \Psi_\epsilon^L(x_Y, 0)$ and so following the derivation above, we have

$$\mu_{f\to z}(1) \propto 1 - (1-\epsilon) \prod_{y\in Y} \mu_{y\to f}(0). \tag{5.13}$$

The constants of proportionality in Eqns. 5.10 and 5.13 can be set to 1 to ensure $\sum_{x_z} \mu_{f\to z}(x_z) = 1$.

■

Proof of Lemma 24

Substituting the form of the Leaky-OR potential defined in Definition 12 into the general message passing equation given in Eqn. 5.2 and noting that the quantity of interest is $\mu_{f\to y}(x_y)$, $y \in Y$ yields:

$$\mu_{f\to y}(x_y) \propto \sum_{x_{Y\setminus y}} \sum_{x_z} \Psi_\epsilon^L(x_Y, x_z) \mu_{z\to f}(x_z) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u) \tag{5.14}$$

where $\sum_{x_{Y\setminus y}}$ is a summation over all configurations of $x_{Y\setminus y}$. Consider the case $x_y = 0$:

$$\mu_{f\to y}(0) \propto \sum_{x_{Y\setminus y}:c(x_{Y\setminus y})=0} \sum_{x_z} \Psi_\epsilon^L(x_Y,x_z)\mu_{z\to f}(x_z) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u) \tag{5.15}$$

$$+ \sum_{x_{Y\setminus y}:c(x_{Y\setminus y})>0} \sum_{x_z} \Psi_\epsilon^L(x_Y,x_z)\mu_{z\to f}(x_z) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u)$$

$$= \epsilon\mu_{z\to f}(1) \prod_{u\in Y\setminus y} \mu_{u\to f}(0) + (1-\epsilon)\mu_{z\to f}(0) \prod_{u\in Y\setminus y} \mu_{u\to f}(0) \tag{5.16}$$

$$+ \sum_{x_{Y\setminus y}:c(x_{Y\setminus y})>0} \mu_{z\to f}(1) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u)$$

$$= \Big(\prod_{u\in Y\setminus y} \mu_{u\to f}(0)\Big)\big((1-\epsilon)(\mu_{z\to f}(0)-\mu_{z\to f}(1))+\mu_{z\to f}(1)\big) \tag{5.17}$$

$$+ \sum_{x_{Y\setminus y}} \mu_{z\to f}(1) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u) - \sum_{x_{Y\setminus y}:c(x_{Y\setminus y})=0} \mu_{z\to f}(1) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u)$$

$$= \Big(\prod_{u\in Y\setminus y} \mu_{u\to f}(0)\Big)\big((1-\epsilon)(\mu_{z\to f}(0)-\mu_{z\to f}(1))+\mu_{z\to f}(1)\big) \tag{5.18}$$

$$+\mu_{z\to f}(1) - \mu_{z\to f}(1) \prod_{u\in Y\setminus y} \mu_{u\to f}(0)$$

$$= \mu_{z\to f}(1) + \Big(\prod_{u\in Y\setminus y} \mu_{u\to f}(0)\Big)\big((1-\epsilon)(\mu_{z\to f}(0)-\mu_{z\to f}(1))\big). \tag{5.19}$$

Consider the case $x_y = 1$:

$$\mu_{f\to y}(1) \propto \sum_{x_{Y\setminus y}} \Psi_\epsilon^L(x_Y,1)\mu_{z\to f}(1) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u) \tag{5.20}$$

$$= \sum_{x_{Y\setminus y}} \mu_{z\to f}(1) \prod_{u\in Y\setminus y} \mu_{u\to f}(x_u) \tag{5.21}$$

$$= \mu_{z\to f}(1). \tag{5.22}$$

■

Proof of Theorem 22

From Lemma 23, it is clear that messages $\mu_{f\to z}(x_z)$, $x_z = \{0,1\}$ can be computed in time $\mathcal{O}(|Y|)$.

From Lemma 24, the computation of $\mu_{f\to y}(1)\,\forall y \in Y$ is trivial. The quantities $\mu_{f\to y}(0)\,\forall y \in Y$ can be computed jointly in $\mathcal{O}(|Y|)$ time by applying Observation 1. Therefore, all messages from a Leaky-OR factor node to its neighbouring variables nodes can be computed in $\mathcal{O}(|Y|)$ time. Noting the degree of the Leaky-OR factor node is $|Y|+1$ completes the proof.

■

5.2.2 Message passing for Selection factors

Recall the definitions of the Selection potential in Definition 13, and the general message passing equation for factors nodes to variable nodes given in Eqn. 5.2. We will show that exploiting the structure of the Selection potential allows one to compute all messages from a Selection factor node to its neighbouring variable nodes in time linear in the degree of the factor node.

Below, let Ψ_θ^S be a Selection potential and let f be the corresponding Selection factor node. As illustrated in Figure 4.1(a), the factor has neighbouring nodes $N(f) = Z \cup \{y\}$. We assume that $\mu_{u\to f}(x_u) > 0$ and $\sum_{x_u} \mu_{u\to f}(x_u) = 1, \forall u \in N(f)$ and $x_u \in \{0, 1\}$.

Theorem 25 *All messages from a Selection factor node to all of its neighbouring variable nodes can be computed in time linear in the degree of the factor node.*

To prove Theorem 25, we require two lemmas.

Lemma 26 *The message passing equations $\mu_{f\to y}(x_y)$ can be expressed as*

$$\mu_{f\to y}(0) \propto \prod_{z\in Z} \mu_{z\to f}(0) \tag{5.23}$$

$$\mu_{f\to y}(1) \propto \Big(\prod_{z\in Z} \mu_{z\to f}(0)\Big) \sum_{z\in Z} \theta_z \frac{\mu_{z\to f}(1)}{\mu_{z\to f}(0)} \tag{5.24}$$

where the constants of proportionality are chosen so that $\mu_{f\to y}(0) + \mu_{f\to y}(1) = 1$.

Lemma 27 *The message passing equations $\mu_{f\to z}(x_z)\ \forall z \in Z$ can be expressed as*

$$\mu_{f\to z}(0) \propto \Big(\prod_{u\in Z\setminus z} \mu_{u\to f}(0)\Big)(\mu_{y\to f}(0) + \mu_{y\to f}(1) \sum_{v\in Z\setminus z} \theta_v \frac{\mu_{v\to f}(1)}{\mu_{v\to f}(0)}) \tag{5.25}$$

$$\mu_{f\to z}(1) \propto \theta_z(\mu_{y\to f}(1) \prod_{u\in Z\setminus z} \mu_{u\to f}(0)) \tag{5.26}$$

where the constants of proportionality are chosen so that $\mu_{f\to z}(0) + \mu_{f\to z}(1) = 1$.

Proof of Lemma 26

Substituting the form of the Selection potential defined in Definition 13 into the general message passing equation given in Eqn. 5.2 and noting that the quantity of interest is $\mu_{f\to y}(x_y)$ yields:

$$\mu_{f\to y}(x_y) \propto \sum_{x_Z} \Psi_\theta^S(x_y, x_Z) \prod_{z\in Z} \mu_{z\to f}(x_z) \tag{5.27}$$

where the summation is over all possible configurations of x_Z.

Consider the case of $x_y = 0$:

$$\begin{aligned} \mu_{f\to y}(0) &\propto \sum_{x_Z} \Psi_\theta^S(0, x_Z) \prod_{z\in Z} \mu_{z\to f}(x_u) && (5.28)\\ &= \prod_{z\in Z} \mu_{z\to f}(0). && (5.29) \end{aligned}$$

Consider the case of $x_y = 1$:

$$\begin{aligned} \mu_{f\to y}(1) &\propto \sum_{x_Z : c(x_Z)=1} \Psi_\theta^S(1, x_Z) \prod_{z\in Z} \mu_{z\to f}(x_u) && (5.30)\\ &= \sum_{z\in Z} \theta_z \big(\mu_{z\to f}(1) \prod_{u\in Z\setminus z} \mu_{u\to f}(0)\big) && (5.31)\\ &= \big(\prod_{z\in Z} \mu_{z\to f}(0)\big) \sum_{z\in Z} \theta_z \frac{\mu_{z\to f}(1)}{\mu_{z\to f}(0)} && (5.32) \end{aligned}$$

■

Proof of Lemma 27 Substituting the form of the Selection potential defined in Definition 13 into the general message passing equation given in Eqn. 5.2 and noting that the quantity of interest is $\mu_{f\to z}(x_z)$, $z \in Z$ yields:

$$\mu_{f\to z}(x_z) \propto \sum_{x_{Z\setminus z}} \sum_{x_y} \Psi_\theta^S(x_y, x_Z) \mu_{y\to f}(x_y) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u) \tag{5.33}$$

where $\sum_{x_{Z\setminus z}}$ is a summation over all configurations of $x_{Z\setminus z}$. Consider the case $x_z = 0$:

$$\mu_{f\to z}(0) \propto \sum_{x_{Z\setminus z}:c(x_{Z\setminus z})=0} \sum_{x_y} \Psi_\theta^S(x_y, x_Z)\mu_{y\to f}(x_y) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u) \tag{5.34}$$

$$+ \sum_{x_{Z\setminus z}:c(x_{Z\setminus z})=1} \sum_{x_y} \Psi_\theta^S(x_y, x_Z)\mu_{y\to f}(x_y) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u)$$

$$= \mu_{y\to f}(0) \prod_{u\in Z\setminus z} \mu_{u\to f}(0) \tag{5.35}$$

$$+ \sum_{x_{Z\setminus z}:c(x_{Z\setminus z})=1} \Psi_\theta^S(1, x_Z)\mu_{y\to f}(1) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u)$$

$$= \mu_{y\to f}(0) \prod_{u\in Z\setminus z} \mu_{u\to f}(0) \tag{5.36}$$

$$+\mu_{y\to f}(1) \sum_{v\in Z\setminus z} \theta_v(\mu_{v\to f}(1) \prod_{u\in Z\setminus\{z,v\}} \mu_{u\to f}(0))$$

$$= \mu_{y\to f}(0) \prod_{u\in Z\setminus z} \mu_{u\to f}(0) \tag{5.37}$$

$$+\mu_{y\to f}(1)\Big(\prod_{u\in Z\setminus z} \mu_{u\to f}(0)\Big) \sum_{v\in Z\setminus z} \theta_v \frac{\mu_{v\to f}(1)}{\mu_{v\to f}(0)}$$

$$= \Big(\prod_{u\in Z\setminus z} \mu_{u\to f}(0)\Big)(\mu_{y\to f}(0) + \mu_{y\to f}(1) \sum_{v\in Z\setminus z} \theta_v \frac{\mu_{v\to f}(1)}{\mu_{v\to f}(0)}) \tag{5.38}$$

Consider the case $x_z = 1$:

$$\mu_{f\to z}(1) \propto \sum_{x_{Z\setminus z}:c(x_{Z\setminus z})=0} \sum_{x_y} \Psi_\theta^S(x_y, x_Z)\mu_{y\to f}(x_y) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u) \tag{5.39}$$

$$= \theta_z(\mu_{y\to f}(1) \prod_{u\in Z\setminus z} \mu_{u\to f}(0)). \tag{5.40}$$

■

Proof of Theorem 25

From Lemma 26, the messages $\mu_{f\to y}(x_y)$, $x_y = \{0, 1\}$ can be computed in $\mathcal{O}(|Z|)$ time since it only requires computing a sum and product over simple quantities for each element of Z.

From Lemma 27, the quantities $\mu_{f\to z}(0) \forall z \in Z$ can be computed jointly in $\mathcal{O}(|Z|)$ time by applying Observation 1. The quantities $\mu_{f\to z}(1) \forall z \in Z$ can also be computed jointly in $\mathcal{O}(|Z|)$ time by applying Observation 1 to compute $\big(\prod_{u\in Z\setminus z} \mu_{u\to f}(0)\big)$, and applying Observation 1 in the log domain to compute $\sum_{v\in Z\setminus z} \theta_v \frac{\mu_{v\to f}(1)}{\mu_{v\to f}(0)}$. Therefore, all messages from a Selection factor node to

its neighbouring variables nodes can be computed in $\mathcal{O}(|Z|)$ time. Noting the degree of the Selection factor node is $|Z| + 1$ completes the proof.

■

5.2.3 Message passing for Berns factors

Recall the definitions of the Berns potential in Definition 14, and the general message passing equation for factors nodes to variable nodes given in Eqn. 5.2. We will show that exploiting the structure of the Berns potential allows one to compute all messages from a Berns factor node to its neighbouring variable nodes in time linear in the degree of the factor node. The Berns factor can be expressed as a product of pairwise factors connecting the input binary variable and one of the output binary variables. Since the Berns factor can be expressed as a product of factors, it is intuitive that message computation can be performed in time linear in the degree of the Berns potential. For completeness, we will prove this result and derive the message passing equation for a Berns factor to its neighbouring variable nodes.

Below, let Ψ_θ^B be a Berns potential and let f be the corresponding Berns factor node. As illustrated in Figure 4.1(c), the factor has neighbouring nodes $N(f) = Z \cup \{y\}$. We assume that $\mu_{u\to f}(x_u) > 0$ and $\sum_{x_u} \mu_{u\to f}(x_u) = 1, \forall u \in N(f)$ and $x_u \in \{0, 1\}$.

Theorem 28 *All messages from a Berns factor node to all of its neighbouring variable nodes can be computed in time linear in the degree of the factor node.*

To prove Theorem 28, we require two lemmas.

Lemma 29 *The message passing equations $\mu_{f\to y}(x_y)$ can be expressed as*

$$\mu_{f\to y}(0) \propto \prod_{z\in Z} \mu_{z\to f}(0) \tag{5.41}$$

$$\mu_{f\to y}(1) \propto \prod_{z\in Z} ((1-\theta_z)\mu_{z\to f}(0) + \theta_z \mu_{z\to f}(1)) \tag{5.42}$$

where the constants of proportionality are chosen so that $\mu_{f\to y}(0) + \mu_{f\to y}(1) = 1$.

Lemma 30 *The message passing equations $\mu_{f\to z}(x_z)\ \forall z \in Z$ can be expressed as*

$$\mu_{f\to z}(0) \propto \mu_{y\to f}(0)\prod_{u\in Z\setminus z}\mu_{u\to f}(0) \tag{5.43}$$
$$+(1-\theta_z)\mu_{y\to f}(1)\prod_{u\in Z\setminus z}((1-\theta_u)\mu_{u\to f}(0)+\theta_u\mu_{u\to f}(1))$$
$$\mu_{f\to z}(1) \propto \theta_z\mu_{y\to f}(1)\prod_{u\in Z\setminus z}((1-\theta_u)\mu_{u\to f}(0)+\theta_u\mu_{u\to f}(1)) \tag{5.44}$$

where the constants of proportionality are chosen so that $\mu_{f\to z}(0)+\mu_{f\to z}(1)=1$.

Proof of Lemma 29

Substituting the form of the Berns potential defined in Definition 14 into the general message passing equation given in Eqn. 5.2 and noting that the quantity of interest is $\mu_{f\to y}(x_y)$ yields:

$$\mu_{f\to y}(x_y) \propto \sum_{x_Z}\Psi_\theta^B(x_y,x_Z)\prod_{u\in Z}\mu_{u\to f}(x_u) \tag{5.45}$$

where the summation is over all possible configurations of x_Z.

Consider the case of $x_y=0$:

$$\mu_{f\to y}(0) \propto \sum_{x_Z}\Psi_\theta^B(0,x_Z)\prod_{u\in Z}\mu_{u\to f}(x_u) \tag{5.46}$$
$$= \prod_{z\in Z}\mu_{z\to f}(0). \tag{5.47}$$

Consider the case of $x_y=1$:

$$\mu_{f\to y}(1) \propto \sum_{x_Z}\Psi_\theta^B(1,x_Z)\prod_{z\in Z}\mu_{z\to f}(x_z) \tag{5.48}$$
$$= \sum_{x_Z}\prod_{z\in Z}\left(\theta_z^{x_z}(1-\theta_z)^{1-x_z}\mu_{z\to f}(x_z)\right) \tag{5.49}$$
$$= \prod_{z\in Z}\left((1-\theta_z)\mu_{z\to f}(0)+\theta_z\mu_{z\to f}(1)\right). \tag{5.50}$$

■

Proof of Lemma 30 Substituting the form of the Berns potential defined in Definition 14 into the general message passing equation given in Eqn. 5.2 and noting that the quantity of interest is $\mu_{f\to z}(x_z)$, $z\in Z$ yields:

$$\mu_{f\to z}(x_z) \propto \sum_{x_{Z\setminus z}} \sum_{x_y} \Psi^B_\theta(x_y, x_Z)\mu_{y\to f}(x_y) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u) \tag{5.51}$$

where $\sum_{x_{Z\setminus z}}$ is a summation over all configurations of $x_{Z\setminus z}$. Consider the case $x_z = 0$:

$$\begin{aligned}\mu_{f\to z}(0) &\propto \sum_{x_{Z\setminus z}} \left(\Psi^B_\theta(0, x_Z)\mu_{y\to f}(0) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u)\right) &(5.52)\\ &\quad + \sum_{x_{Z\setminus z}} \left(\Psi^B_\theta(1, x_Z)\mu_{y\to f}(1) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u)\right)\\ &= \mu_{y\to f}(0) \prod_{u\in Z\setminus z} \mu_{u\to f}(0) &(5.53)\\ &\quad + (1-\theta_z)\mu_{y\to f}(1) \prod_{u\in Z\setminus z} ((1-\theta_u)\mu_{u\to f}(0) + \theta_u\mu_{u\to f}(1)).\end{aligned}$$

Consider the case $x_z = 1$:

$$\begin{aligned}\mu_{f\to z}(1) &\propto \sum_{x_{Z\setminus z}} \left(\Psi^B_\theta(0, x_Z)\mu_{y\to f}(0) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u)\right) &(5.54)\\ &\quad + \sum_{x_{Z\setminus z}} \left(\Psi^B_\theta(1, x_Z)\mu_{y\to f}(1) \prod_{u\in Z\setminus z} \mu_{u\to f}(x_u)\right)\\ &= \theta_z\mu_{y\to f}(1) \prod_{u\in Z\setminus z} ((1-\theta_u)\mu_{u\to f}(0) + \theta_u\mu_{u\to f}(1)). &(5.55)\end{aligned}$$

■

Proof of Theorem 28

From Lemma 29, the messages $\mu_{f\to y}(x_y)$, $x_y \in \{0,1\}$ can be computed in time $\mathcal{O}(|Z|)$ since it only requires computing a product over easily computable quantities for each element of Z.

From Lemma 30, the quantities $\mu_{f\to z}(x_z)\forall z \in Z$,$x_z \in \{0,1\}$ can be computed jointly in $\mathcal{O}(|Z|)$ time by applying Observation 1. Therefore, all messages from a Berns factor node to its neighbouring variables nodes can be computed in $\mathcal{O}(|Z|)$ time. Noting the degree of the Berns factor node is $|Z|+1$ completes the proof.

■

5.2.4 Proof of Theorem 20

Proof The result follows directly from Theorems 21, 22, 25, and 28. ■

5.3 Markov Chain Monte Carlo as an alternative to LBP

In this thesis, we are mainly concerned with computing marginal probabilities such as the probability of the presence/absence of bricks in a given scene. As pointed out in Chapter 1, Markov Chain Monte Carlo (MCMC) techniques have been successfully used for inference in other grammar-based frameworks. A natural question is if MCMC is a feasible alternative to LBP for inference in the PSG framework. For the factor graphs we consider, simple MCMC schemes such as Gibbs sampling ([20]) and Metropolis-Hastings ([25]) are impractical.

The main issue with applying Gibbs sampling or Metropolis-Hastings sampling here is that the factor graph under consideration may contain on the order of millions to billions of tightly coupled random variables. Because of the tight coupling, it is extremely difficult to design good move proposals for Metropolis-Hastings, and Gibbs sampling may be unable to mix since sampling a random variable from its conditional distribution is likely to result in no change in state. The crux of the issue is that due to the tight coupling of random variables, it is difficult to design a transition kernel that defines an irreducible Markov Chain yet also has a reasonable probability that proposed moves will be accepted. The sheer size of the factor graph exacerbates the problem of the MCMC chain mixing.

Consider using Metropolis-Hastings MCMC as the inference engine for the toy grammar listed in Grammar 4 and consider the following situation. Let $\mathcal{A} = \{(B,3),(C,3),(D,3)\}$ be a set of bricks. Considering the setting $\mathbf{X}(a) = 1$, $a \in \mathcal{A}$ and $\mathbf{X}(a') = 0\ \forall a' \notin \mathcal{A}$. In this setting, the random variable $\mathbf{X}(C,3)$ can have value 1 only if the brick $(B,3)$ generated it. Similarly, the random variable $\mathbf{X}(D,3)$ can only have value 1 if the brick $(C,3)$ generated it. Now, consider proposing a new value for the random variable $\mathbf{X}(C,3)$. Its state cannot be set to 0 since the brick $(B,3)$ will have no generated bricks, which is impossible under the model. Also, the brick $(D,3)$ will have no parent, which is also impossible under the model. Similarly, it not possible for Metropolis-Hastings to propose a different state for $\mathbf{X}(B,3)$ or $\mathbf{X}(D,3)$; Metropolis-Hastings is "stuck".

As we have demonstrated, for inference in the factor graph representing the model described in Grammar 4, serial MCMC techniques are too slow to be practical and will suffer from getting "stuck" in a particular state. To alleviate these difficulties, one may try block-move schemes such as Block Gibbs Sampling and Block Metropolis-Hastings whereby a large joint move involving the

random variables of multiple bricks is considered. However, the tight coupling of random variables will still pose a problem since for the factors graphs we consider, most joint assignments to random variables will be extremely unlikely or invalid. One could imagine designing effective model-specific block sampling schemes. But, as we are interested in a general, problem-agnostic framework for scene understanding, we wish to avoid inference schemes that are sensitive to the problem under consideration.

It may be possible to use more sophisticated MCMC methods, such as Slice Sampling ([43]) which was effective in the Picture framework ([34]), Hybrid Monte Carlo([44]), or a combination of MCMC methods. Such an approach may be viable and is an interesting direction for future research.

Chapter 6

Example grammars: Inference with LBP

In this chapter we examine the results obtained by running LBP on the factor graph representation of the example grammars in Chapter 3. In the examples in this chapter, we condition on the presence/absence of a set of bricks in the scene and run LBP to convergence. We then compute approximate marginals for each unconditioned brick using Eqn. 5.3.

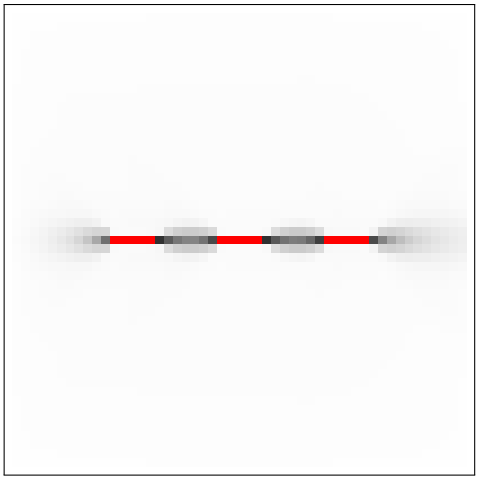
6.1 LBP computations with a curve grammar

In this section we demonstrate the ability of the PSG framework to perform contour completion using Grammar 1. Figure 6.1 shows two examples of contour completions. In these experiments we condition on the presence of some INK bricks (shown in red) and compute the approximate marginal probabilities of the remaining INK bricks being present in the scene using LBP. As shown in the figure, the PSG framework is capable of completing gaps in contours. Note that in both contour completions, there is uncertainty in the precise completions.

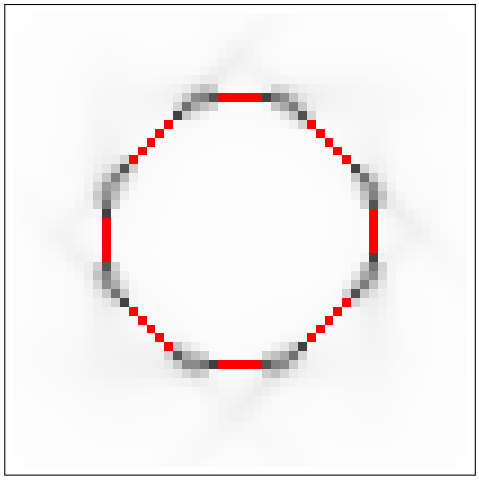
In the example in Figure 6.1(a), there is uncertainty as to whether the contour should be straight or if there are slight deviations along the contour in the vertical direction. Also, the PSG framework places non-trivial probability mass on the event that the contour continues on either side.

In the example in Figure 6.1(b), the model captures uncertainty in the precise completion of the contour(s). Also, the PSG framework places little probability mass on the event that observed contours intersect and extend past the point of intersection. Instead, the PSG framework estimates that it is more likely that each observed contour "turns" into a neighbouring contour via a change of orientation.

As these two examples demonstrate, the PSG framework uses some notion of context to perform contour completion.



(a) Contour gap completions. Note that the PSG framework expresses variability in the precise gap completions. Also, there is some uncertainty as to whether the contour extends to the left and to the right.



(b) Completion of several contour gaps. Here, the PSG framework is capable of filling in gaps between observed contours around plausible intersection points. As with the example in Figure 6.1(a), there is variability over the precise completions. See text for discussion.

Figure 6.1: A visualization of two contour completion examples. Each pixel represents an INK brick present at that location. The INK bricks conditioned to be present in the scene are denoted by a red pixel; all other bricks in the scene are unconditioned. The gray-scale values show the resulting approximate marginal probabilities computed by LBP. Darker pixels indicate a higher approximate marginal probability for the corresponding INK brick to be present in the scene.

6.2 LBP computations with a face grammar

In this section we demonstrate the ability of the PSG framework to perform face and face part localization using Grammar 2. Note that the grammar forms a two level hierarchy in that FACE bricks generate EYE, NOSE, and MOUTH bricks. So, one can talk about top-down contextual information in the sense of a FACE brick providing context for EYE, NOSE, and MOUTH bricks, and bottom-up contextual information in the sense of EYE, NOSE, and MOUTH bricks providing context for a FACE brick. Figure 6.2 shows the results of inference after conditioning on the presence of three different sets of bricks in the scene.

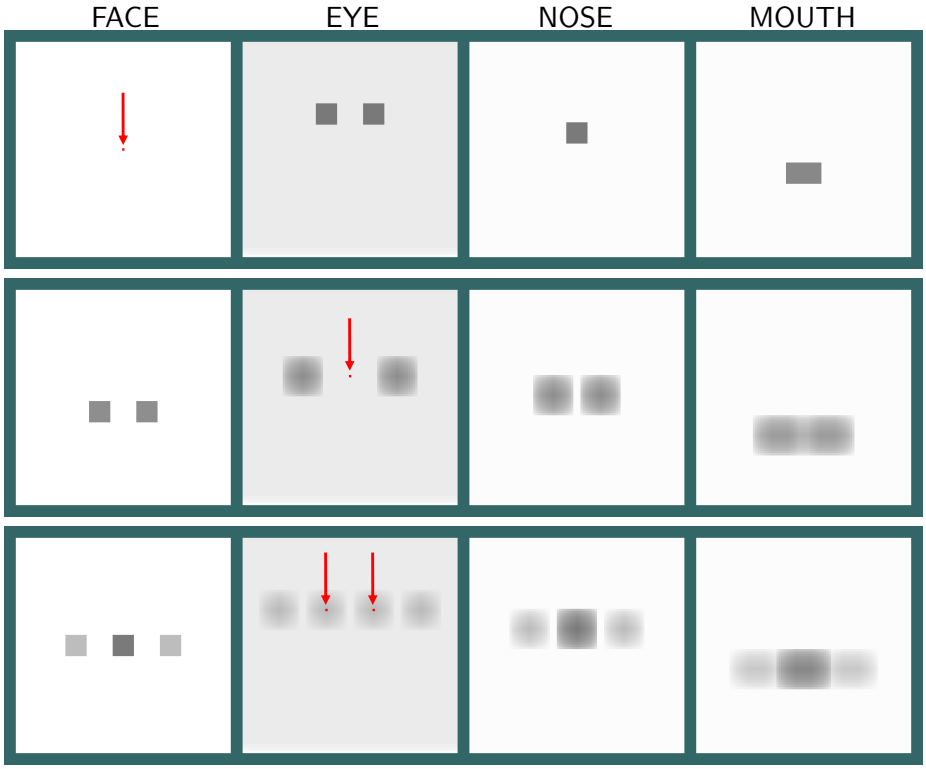


Figure 6.2: A visualization of face and face part localization in various contexts. Each row is a different example where a different set of bricks was conditioned to be present in the scene. Each column represents a symbol of the grammar. Each pixel represents a brick at that pixel location. The bricks conditioned to be present in the image are denoted by a red pixel with a red arrow pointing to them. All other bricks in the scene are unconditioned. The gray-scale values are a visualization of the resulting approximate marginal probabilities of a brick being present in the scene as computed by LBP. Darker pixels indicate a higher approximate marginal probability to be present. For visualization purposes, a non-linear monotonic transformation was applied to the approximate marginal probabilities to enhance contrast. See text for an analysis of the results of inference.

Row 1 of Figure 6.2 shows the results of LBP when the FACE brick at the centre of the scene is conditioned to be present. The PSG framework performs top-down reasoning and posits possible locations for the face parts. Since this grammar model allows for variability in the location of the face parts, there is a region of plausible locations for each part.

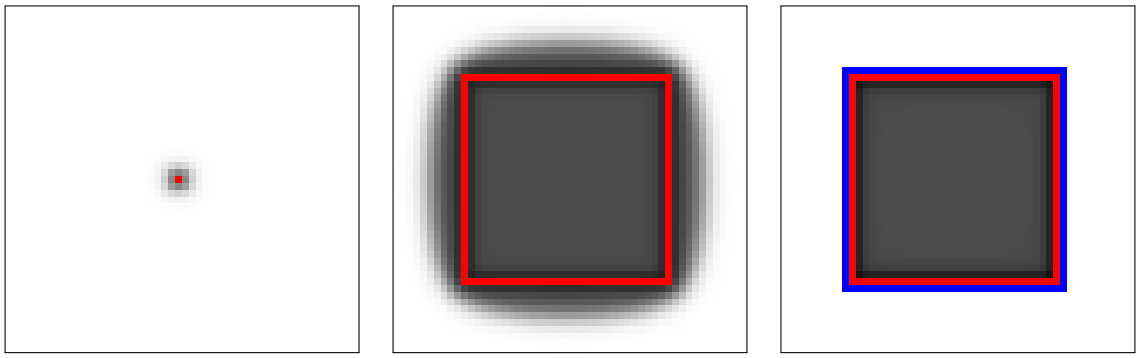
Row 2 of Figure 6.2 shows the results of LBP when the EYE brick at the centre of the scene is conditioned to be present. Here, the PSG framework performs bottom-up and top-down reasoning. First, since an EYE seldom appears on its own, the PSG framework infers that it is likely that there is a FACE present in the scene. This is an example of bottom-up reasoning. Due to the variability in the relative poses between a FACE and a constituent EYE, the framework is uncertain of the precise location of the FACE. Moreover, the distribution over possible FACE poses is bimodal since an EYE can either be the left eye or the right eye, so the framework reasons about both possibilities. For each FACE brick that may be present in the scene, the framework reasons about all of its constituent parts, hence the distribution over NOSE and MOUTH bricks is bimodal as well. This is an example of top-down reasoning. Note that to make the deduction that there is likely to be a NOSE or MOUTH in the scene based solely on observing an EYE requires a chain of reasoning that goes from bottom-up to infer the presence of a FACE, and then to top-down to infer the presence of the other FACE parts.

Row 3 of Figure 6.2 shows the results of LBP when two EYE bricks that can both be generated by a single FACE brick are conditioned to be present in the scene. Note that the approximate marginal distribution of the FACE bricks present in the scene is trimodal. The two smaller modes correspond to the possibility that each EYE brick is generated by a different FACE brick, and so there may be two FACE bricks present in the scene. The centre mode corresponds to the possibility that both EYE bricks are generated by the same FACE brick. The face grammar used here indicates that the presence of any particular FACE brick in a scene is rare through a low self-rooting probability for FACE bricks. Because of this, the PSG framework places higher probability on the event that there is a single FACE brick present in the scene with both conditioned-on EYE bricks as constituent parts and a lower probability on the event that there are two FACE bricks present in the scene with each EYE brick being generated by a different FACE brick.

The three examples in Figure 6.2 showcase the ability of the PSG framework to simultaneously perform top-down and bottom-up reasoning. Both kinds of reasoning are crucial to capture a notion of context. Note that there is no explicit notion of top-down nor bottom-up here, but such a concept naturally emerges due to the hierarchical nature of Grammar 2. We argue that the ability to reason contextually in a top-down and bottom-up fashion is crucial if one wishes to capture the notion of context and leverage it in scene understanding tasks. Here, we have demonstrated the ability of the PSG framework to capture a notion of context.

6.3 LBP computations with a binary segmentation grammar

In this section, we analyze the capability of the PSG framework to reason about binary image segments using Grammar 3. Figure 6.3 shows the results of inference on three examples scenes.



(a) (b) (c)

Figure 6.3: A visualization of three binary image segmentation examples. Each pixel represents an FG brick at that location. FG bricks conditioned to be present in the scene are denoted by a red pixel. FG bricks conditioned to be absent in the scene are denoted by a blue pixel. All other bricks in the scene are unconditioned. The gray-scale values show the resulting approximate marginal probabilities computed by LBP. Darker pixels indicate a higher approximate marginal probability to be present in the scene. See text for discussion. Best viewed in colour.

In Figure 6.3(a), we condition on the presence of the FG brick in the centre of the scene. Its presence influences other FG bricks near it to be present in the scene as well, with its influence decreasing with distance. The most probable interpretation of the scene under this grammar is that the SEED brick chose the FG brick indicated by the red pixel to be present, and the chosen FG brick potentially generates other FG bricks. However, another possible scene interpretation is that the SEED brick chose some other FG brick to be present, and through the generative process, the FG brick indicated by the red pixel was generated. The latter event has many ways to occur since the SEED brick could have chosen any other FG brick to start the generative process. The PSG framework reasons about both of these scene interpretations.

In Figure 6.3(b), we condition on a set of FG bricks being present in the scene. The FG bricks that are conditioned to be present form the boundary of a square in the scene. Here, there are many possible scene interpretations since the SEED brick could have chosen any of the FG bricks conditioned to be present to start the generative process. The results of inference suggests that there is a high probability that the FG bricks inside the square are present. Outside the square, the approximate marginal probabilities computed by LBP decay rapidly with distance from the square's boundary. Note that the approximate marginal probability for the FG brick at the centre of the square

is higher than most of the FG bricks outside of the square. In fact, the set of FG bricks outside the square and more than 2 pixels from the square's boundary have lower approximate marginal probability to be present in the scene than the centre FG brick, despite the centre FG brick being 16 pixels away from the square's boundary. This suggests that the PSG framework is capable of "filling in" shapes.

In Figure 6.3(c), we condition on a set of FG bricks being present in the scene, and condition on another set of FG bricks being absent in the scene. Both sets of FG form squares in the scene. Note that this example is similar to the example shown in Figure 6.3(b), except now we also condition on the absence of a set of bricks. Recall that Grammar 3 generates scenes with a single, non-empty connected foreground component. So, under the distribution over scenes induced by Grammar, 3, it is impossible for FG bricks outside the blue square to be present in the scene since some FG bricks inside the blue square must be present. The approximate marginals estimated by LBP reflect this fact. However, LBP is not always able to reason about such global constraints correctly.

Figure 6.4 shows another run of LBP using the same set of conditioned on bricks as the example in Figure 6.3(c). Here, LBP produces very different results. Recall from Chapter 5 that LBP requires an initialization of messages. In the example shown in Figure 6.4, the messages of LBP were initialized to favour the presence of all unconditioned FG in the scene. In contrast, in the example shown in Figure 6.3(c), the messages of LBP were initialized to discourage the presence of any unconditioned FG bricks in the scene. In Figure 6.4, the approximate marginals computed by LBP are inconsistent with the distribution over scenes induced by Grammar 3 since under the distribution over scenes induced by Grammar 3, it is impossible for any FG bricks outside of the blue square to be present.

There are three causes that contribute to the mismatch between the approximate marginals produced by LBP and the distribution over scenes induced by Grammar 3 in the example shown in Figure 6.4. The first issue is numerical. In our implementation of LBP, messages are constrained to be non-zero to avoid numerical issues. If one uses Eqn. 5.3 to compute approximate marginal probabilities, then it is impossible to have a marginal probability be exactly 0 since that requires at least one of the LBP messages to be exactly 0. Hence, in our implementation of LBP, we cannot capture the notion that it is impossible for some set of bricks to be present in the scene.

The second issue is that the distribution over scenes represented by the factor graph constructed from Grammar 3 using Definition 16 is different than the distribution over scenes represented by the original grammar. The factor graph construction assumes an acyclic grammar, but Grammar 3 is cyclic, and hence the distributions over scenes are not the same. Since we perform inference using the constructed factor graph, we are performing inference with a distribution over scenes that is related to but different from the one induced by Grammar 3.

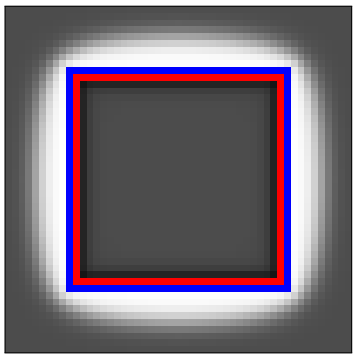


Figure 6.4: A visualization of a binary image segmentation task where the approximate marginals computed by LBP are inconsistent with the underlying grammar model. The gray-scale values show the resulting approximate marginal probabilities computed by LBP. Darker pixels indicate a higher approximate marginal probability to be present in the scene. The set of FG bricks conditioned on is the same as in Figure 6.3(c), but here the messages of LBP have been initialized to favour the presence of all FG bricks in the scene. Note that as in Figure 6.3(c), according to the generative process, it is impossible for any FG brick outside of the blue square to be present. However, in this case LBP incorrectly reasons about this constraint.

The third issue is that LBP is a heuristic for performing approximate inference in loopy factor graphs. Although in practice LBP seems to produce reasonable approximations to marginal quantities for many tasks (see [42, 33, 14]), there is no guarantee it will work well for arbitrary tasks and factor graphs. Empirically, LBP has difficulty producing accurate approximations when the underlying factor graph contains many loops, as is the case here.

Chapter 7

Connections to Pictorial Structures

In this chapter, we elucidate the connections between the PSG framework and Pictorial Structures described in [13]. In particular, we show that the prior over scenes that a Pictorial Structure (PS) model defines can be expressed as a PSG as described in Chapter 2, but the reverse is not true. Thus, the PSG representation can be viewed as a generalization of the PS model representation. Also, the graphical model representation of a PS model differs significantly from the factor graph representation used in the PSG framework. The difference in graphical model representation has consequences on the accuracy and speed of inference. Namely, the PS graphical model allows for efficient and exact maximum a posteriori (MAP) estimation via dynamic programming and generalized distance transforms. Recall from Chapter 5 that in contrast, the PSG framework employs the approximate inference scheme of LBP which, in practice, is slower than the exact inference schemed used in the PS framework.

7.1 Pictorial Structures: Overview

We first describe the PS model, as presented in [13]. A PS model represents objects as a collection of parts and connections between parts. A PS model can be represented as an undirected graph $G = (V, E)$, where $V = \{v_1, \ldots v_n\}$ represents a set of objects/parts, and E is a set of pairs $\{v_i, v_j\}$. A configuration of parts is given by $L = \{l_1, \ldots l_n\}$ where l_i specifies the pose of part v_i in the scene. Poses may correspond to pixel coordinates, for example. Importantly, a PS model implicitly assumes that there is one instance of each object in the scene since l_i represents a single pose for part v_i. To model the geometric relationship between parts, a PS model has pairwise terms $d_{ij}(l_i, l_j)$ that measures the degree of disagreement between the placement of parts v_i and v_j at locations l_i and

l_j, respectively. Finally, given an image Y, the term $m_i(l_i, Y)$ is a cost for placing the object v_i at location l_i based on the image evidence.

The energy of a configuration L given an image Y is defined to be

$$F(L, Y) = \sum_{\{v_i, v_j\} \in E} d_{ij}(l_i, l_j) + \sum_{i=1}^{n} m_i(l_i, Y). \tag{7.1}$$

The energy in Eqn. 7.1 defines a probability

$$p(L, Y) = \frac{1}{Z} \prod_{\{v_i, v_j\} \in E} \left(e^{-d_{ij}(l_i, l_j)}\right) \left(\prod_{i=1}^{n} e^{-m_i(l_i, Y)}\right). \tag{7.2}$$

Eqn. 7.2 can be viewed as a conditional probability of L given Y defined by a product of a prior over scenes, $p(L)$, and an image likelihood $p(Y \mid L)$, where

$$p(L) \propto \prod_{\{v_i, v_j\} \in E} e^{-d_{ij}(l_i, l_j)} \tag{7.3}$$

$$p(Y \mid L) \propto \prod_{i=1}^{n} e^{-m_i(l_i, Y)}. \tag{7.4}$$

In [13], inference is performed using maximum a posteriori (MAP) by minimizing Eqn. 7.1. For efficient inference, $G = (V, E)$ is constrained to be a tree and the d_{ij} must be expressible as a Mahalanobis distance between locations in a transformed space. These constraints allow for exact MAP estimation using dynamic programming and generalized distance transforms. [13] also discusses the computation of marginal distributions. Here, we will focus on the MAP setting but the connections outlined in this chapter are equally applicable to the setting where marginals are computed.

7.2 Expressing a Pictorial Structure model as a PSG

In this section, we show how to represent the prior over scenes defined by a PS model, $p(L)$, in the language of a PSG as in Definition 1.

We describe the construction of a PSG from an arbitrary PS model below. Since a PS model is restricted to be tree-structured, without loss of generality, we take the root of the tree to be v_1 and we assume that the v_i are ordered with a breadth-first search starting at v_1.

Definition 31 *Construction process for transforming a PS model into a PSG:*

1. *Represent each object/part $v_i \in V$ in the PS model as a symbol in the PSG. For simplicity, we will use v_i $1 \leq i \leq n$ as symbols in the PSG.*
2. *Let L_i be the set of possible values for l_i, $1 \leq i \leq n$. Define the pose space of v_i to be L_i.*
3. *For each v_i, create one production rule with v_i in the LHS and include v_j in the RHS if $\{i, j\} \in E$ and $j > i$. Assign this rule probability 1. Since each symbol v_i appears only once in the LHS of the set of production rules, without loss of generality, assume rule r has v_r in the LHS.*
4. *Set all conditional pose distributions to be $\mathrm{Categorical}$ distributions. For all pairs $\{v_i, v_j\}$ where $\{v_i, v_j\} \in E$ and $j > i$, set the parameter $\theta_{(l_i,i,k,l_j)} \propto e^{-d_{ij}(l_i,l_j)}$ for $l_i \in L_i$ and $l_j \in L_j$. The constant of proportionality is chosen so that the set of parameters $\theta_{(l_i,i,k)}$ sums to 1.*
5. *Set $\epsilon_{v_i} = 0$, $1 \leq i \leq n$.*
6. *Introduce a symbol v_{SEED}. This symbol will be used to ensure that there is exactly one instance of v_1 in the scene.*
7. *Set the pose space of v_{SEED} to be $[1]$.*
8. *Set $\epsilon_{v_{\mathsf{SEED}}} = 1$.*
9. *Create a production rule with v_{SEED} in the LHS and only v_1 in the RHS. Without loss of generality, assume this rule is the $(n+1)$th rule.*
10. *Set $\gamma_{(n+1,1)}$ to be a $\mathrm{Uniform}$ distribution over the elements of L_1.*
11. *Set all rule probabilities to 1.*

Following the construction above yields a PSG that describes a prior over scenes that matches an arbitrary PS model's prior over scenes.

Although the prior over scenes that a PS model defines can be represented as a PSG, the reverse is not true. Below are several aspects of a PSG model that cannot be expressed in a PS model.

- A PSG can have multiple instances of each object in the scene.
- The conditional pose distributions in a PSG can be an $\mathrm{IndBern}$ distribution.
- A PSG can have multiple possible compositions for a given object, with each composition having a different probability of occurring.

- The grammar of a PSG need not be tree-structured.

Since a PS model's prior over scenes, $p(L)$, can be represented as a PSG but the reverse is not true, we say that the PSG representation described in Chapter 2 is a generalization of the PS model representation.

7.2.1 Example construction

In this subsection, we give a concrete example of using the construction described in Definition 31 to represent a PS model as a PSG .

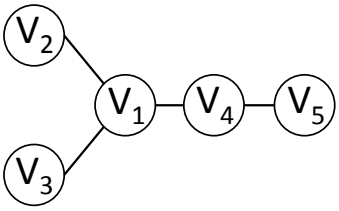


Figure 7.1: Undirected graphical model representation for a PS model where $V = \{v_1, \ldots, v_5\}$.

Figure 7.1 represents the undirected graphical model for a PS model where $V = \{v_1, \ldots, v_5\}$ and the state space of v_i is L_i. Following the construction process described in Definition 31, the corresponding PSG is given in Grammar 5.

Grammar 5 *A PSG representation of the PS model in Figure 7.1:*

$\Sigma = \{v_1, \ldots, v_5, v_{\mathsf{SEED}}\}$.
$\Omega_{v_i} = L_i$, $1 \leq i \leq 5$.
$\Omega_{v_{\mathsf{SEED}}} = [1]$.
Rules:

$$
\begin{aligned}
&1.0, \quad (v_1, l_1) \quad \rightarrow \quad (v_2, \text{Categorical}(\cdot \mid \theta_{(l_1,1,1)})),\\
&\qquad\qquad\qquad\quad (v_3, \text{Categorical}(\cdot \mid \theta_{(l_1,1,2)})),\\
&\qquad\qquad\qquad\quad (v_4, \text{Categorical}(\cdot \mid \theta_{(l_1,1,3)})),\\
&1.0, \quad (v_2, l_2) \quad \rightarrow \emptyset\\
&1.0, \quad (v_3, l_3) \quad \rightarrow \emptyset\\
&1.0, \quad (v_4, l_4) \quad \rightarrow \quad (v_5, \text{Categorical}(\cdot \mid \theta_{(l_4,4,1)})),\\
&1.0, \quad (v_5, l_5) \quad \rightarrow \emptyset\\
&1.0, \quad (v_{\mathsf{SEED}}, 1) \quad \rightarrow \quad (v_1, \text{Uniform}(L_1))
\end{aligned}
$$

$\epsilon_{v_i} = 0$, $1 \leq i \leq 5$,
$\epsilon_{v_{\mathsf{SEED}}} = 1$

Grammar 5 induces a distribution over scenes where each object/part $v_i \in \{v_1, \ldots, v_5\}$ appears exactly once in the scene, and the probability of placing v_j at l_j given that object v_i is placed at l_i is proportional to $e^{-d_{ij}(l_i,l_j)}$ for $\{v_i, v_j\} \in E$ and $j > i$. This prior over scenes induced by the PSG in Grammar 5 matches the prior over scenes induced by the PS model depicted in Figure 7.1.

7.3 Pictorial Structures vs. PSG : graphical models and inference

Inference in the PS framework differs substantially from inference in the PSG framework. The differences in inference can be attributed to the underlying graphical model representations. For a PS model, the graphical model representation is a tree-structured Markov random field (MRF). Examining the Eqn. 7.1, one takes the l_i, $1 \leq i \leq n$, to be the random variables of the MRF, and the terms $e^{-d_{ij}(l_i,l_j)}$ and $e^{-m_i(l_i,Y)}$ correspond to the pairwise and unary potentials, respectively. This formulation is significantly different than the PSG factor graph. The key differences are that 1) the PS graphical model is tree-structured while the PSG graphical model is not tree-structured, and 2) the PS graphical model typically has a few random variables with large state spaces while the PSG graphical model typically has a large number of binary random variables. These differences have implications for both the accuracy and speed of inference. For PS, MAP estimation with its graphical model is exact and fast since one can use dynamic programming and generalized distance transforms. On the other hand, if one were to perform MAP estimation with the PSG factor graph, [1] inference would only be approximate and slower than inference in the PS framework since the PSG framework uses LBP in a loopy graph.

As shown in Section 7.2, a PS model can be expressed in the PSG framework, but the reverse is not true; the set of models expressible in the PS framework is a proper subset of the models expressible in the PSG framework. However, the cost for the additional modeling power of the PSG framework is that inference is only approximate and slower than in the PS framework.

[1] Although we use LBP to compute marginals as stated in Chapter 5, we could use max-product LBP to perform inference in a MAP setting.

Chapter 8

Learning Model Parameters

Recall from Definition 1 that a PSG is defined by a 6-tuple $\mathcal{G} = (\Sigma, \Omega, \mathcal{R}, q, \gamma, \epsilon)$. The chief learning problem we consider in this work is estimating model parameters q, γ, and ϵ.

We first formalize the parameter estimation problem, briefly describe the Expectation-Maximization (EM) algorithm, and finally describe a modification to the EM algorithm so that it can be applied in the PSG framework. The modification is to replace exact posterior quantities required by the Maximization-step of the EM algorithm with approximate posterior quantities by computed LBP. The general idea of replacing the exact posteriors by approximate posteriors computed by LBP is related to the work of [26]. The learning algorithm described in this chapter can be thought of as an approximation variant of the EM algorithm.

8.1 Maximum likelihood estimation

Consider a set of n datapoints $\mathbf{D} = \{\mathbf{D}_1, \cdots, \mathbf{D}_n\}$ that are independent and identically distributed. Let Φ be the set of model parameters. The data likelihood function is

$$p(\mathbf{D} \mid \Phi) \quad = \quad \prod_{i=1}^{n} p(\mathbf{D}_i \mid \Phi). \tag{8.1}$$

In the maximum likelihood setting, the goal is to find a setting for Φ that maximizes the data likelihood, or equivalently, the data log-likelihood. Formally, we seek to solve

$$\Phi^* \quad = \quad \arg\max_{\Phi} \log p(\mathbf{D} \mid \Phi) \tag{8.2}$$

$$\quad = \quad \arg\max_{\Phi} \sum_{i=1}^{n} \log p(\mathbf{D}_i \mid \Phi). \tag{8.3}$$

Suppose that the probabilistic model under consideration has hidden variables $\mathbf{Z} = \{\mathbf{Z}_1, \cdots, \mathbf{Z}_n\}$ where $\mathbf{Z}_i$ is the set of hidden variables associated with datapoint $\mathbf{D}_i$. The hidden variables can represent missing observations, or variables that cannot be directly observed. The joint distribution $p(\mathbf{D}, \mathbf{Z} \mid \Phi)$ is commonly called the complete-data likelihood. With hidden variables, Eqn. 8.3 can be written as

$$\Phi^* = \arg\max_{\Phi} \sum_{i=1}^{n} \log \big(\sum_{\mathbf{Z}_i} p(\mathbf{D}_i, \mathbf{Z}_i \mid \Phi) \big). \tag{8.4}$$

Unfortunately, solving Eqn. 8.4 exactly in the general case is intractable. Fortunately, the EM algorithm is specifically designed to address the maximum-likelihood estimation problem given in Eqn. 8.3. We describe the EM algorithm in the next section.

8.2 EM algorithm

Recall that the EM algorithm first described in [10] can be applied to maximum-likelihood estimation problems with hidden variables. When used to solve 8.4, the EM algorithm produces a locally optimal solution for Φ.

Generally, the EM algorithm is an iterative algorithm that alternates between computing the posterior distribution over hidden variables given a setting of the model parameters, and computing a setting of the model parameters given a posterior distribution over hidden variables. The two alternating steps are called the Expectation-step (E-step) and the Maximization-step (M-step), respectively.

Definition 32 *Let* $\mathbf{Z}$ *denote the hidden variables of a probabilistic model, let* $\mathbf{D}$ *denote the observed data, and let* Φ *denote the model parameters. The E-step of the EM algorithm computes the posterior distribution* $p(\mathbf{Z} \mid \mathbf{D}, \Phi)$.

The M-step of EM algorithm makes use of the expectation of the complete-data likelihood under the posterior distribution over hidden variables. This expectation is commonly called the $\mathcal{Q}$-distribution and is defined below.

Definition 33 *The* $\mathcal{Q}$*-distribution is defined as*

$$\mathcal{Q}(\Phi', \Phi) = \mathbb{E}_{p(\mathbf{Z}|\mathbf{D},\Phi)}[\log p(\mathbf{D}, \mathbf{Z} \mid \Phi')]. \tag{8.5}$$

Definition 34 *Let $\Phi^{(t)}$ be the set of model parameters at EM iteration t. The M-step of the EM algorithm involves solving the optimization problem*

$$\Phi^{(t+1)} = \arg\max_{\Phi'} \mathcal{Q}(\Phi', \Phi^{(t)}). \tag{8.6}$$

In the EM algorithm, the model parameters are first initialized to some starting value. Then, the algorithm alternates between performing the E-step and M-step described in Definitions 32 and 34, respectively. The algorithm is guaranteed to converge (see [10] for proof) and the resulting solution for the model parameters Φ are taken as an approximate solution to maximum-likelihood estimation problem. Although the EM algorithm is not guaranteed to find the global optimal solution for Φ, it is guaranteed to find a locally optimal solution.

8.3 Applying EM to the PSG framework

In the PSG framework, we seek to fit the model parameters $\Phi = \{q, \epsilon, \gamma\}$ (fitting the conditional pose distributions γ entails fitting the set of parameters θ which govern those distributions). Here, we assume that the PSG is acyclic and work with its factor graph representation, as described in Section 4.2. In our setting, each datapoint $\mathbf{D}_i$ corresponds to an image. Note that we differentiate between an image and a scene; a scene is a description of the image that contains higher level information, such as what objects are present in the image and what are the compositional relationships between them.

Denote by $\mathbf{X}_i(A, \omega)$ the random variable $\mathbf{X}(A, \omega)$ associated with scene i. Define $\mathbf{R}_i(A, \omega)$ and $\mathbf{C}_i(A, \omega)$ analogously. Define the following sets of random variables:

$$\begin{aligned}
\mathbf{X}_i &= \{\mathbf{X}_i(A, \omega) \mid A \in \Sigma, \omega \in \Omega_A\} \\
\mathbf{R}_i &= \{\mathbf{R}_i(A, \omega) \mid A \in \Sigma, \omega \in \Omega_A\} \\
\mathbf{C}_i &= \{\mathbf{C}_i(A, \omega) \mid A \in \Sigma, \omega \in \Omega_A\} \\
\mathbf{Z}_i &= \{\mathbf{X}_i, \mathbf{R}_i, \mathbf{C}_i\}.
\end{aligned}$$

We have implicitly assumed that all scenes have the same set of bricks, which may not be true in practice. For example, scenes may be different sizes and so the pose spaces of the symbols may differ. The results in this chapter can be modified to accommodate scenes of varying sizes. For simplicity, we assume below that all scenes are the same size.

Generally, the PSG framework contains hidden variables. For example, although a FACE brick may be present in a scene, there may be no direct image evidence as to which rule was chosen

to expand that FACE brick. In general, the PSG framework treats the set of random variables $\mathbf{Z} = \{\mathbf{Z}_1, \cdots \mathbf{Z}_n\}$ as hidden variables. Estimating the parameters Φ can be formulated in the maximum likelihood setting with hidden variables, as in Eqn. 8.4.

Below, we first outline the M-step updates assuming that the posterior distribution computed in the E-step is available. We then outline a modification to the EM algorithm's E-step whereby computation of the exact posterior distribution is replaced by computation of an approximation to the posterior.

8.3.1 M-step

In this subsection, given a set of model parameters $\Phi^{(t)}$, we show how to solve for the updated model parameters $\Phi^{(t+1)}$, as in Eqn. 8.6.

We assume that the posterior quantity $p(\mathbf{Z} \mid \mathbf{D}, \Phi^{(t)})$ has been computed in the E-step (see next subsection). Since we assume the $\mathbf{D}_i$'s are independent and identically distributed, the $\mathcal{Q}$-distribution can be expressed as

$$\mathcal{Q}(\Phi', \Phi^{(t)}) = \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i \mid \mathbf{D}_i, \Phi^{(t)})}[\log p(\mathbf{Z}_i \mid \Phi') + \log p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')] \tag{8.7}$$

where $p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')$ is the likelihood of the i-th datapoint and $p(\mathbf{Z}_i \mid \Phi')$ is the prior distribution over scenes defined by the PSG factor graph. We assume that the PSG model parameters do not appear in the likelihood $p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')$.

Proposition 35 *Let* $\text{par}(\mathbf{X}_i(A, \omega)) = 0$ *denote the setting in which* $\mathbf{C} = 0 \; \forall \mathbf{C} \in \text{par}(\mathbf{X}_i(A, \omega))$. *The M-step update for the self-rooting parameters* ϵ_A, $A \in \Sigma$, *is*

$$\epsilon_A = \frac{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} p(\mathbf{X}_i(A, \omega) = 1, \text{par}(\mathbf{X}_i(A, \omega) = 0) \mid \mathbf{D}_i, \Phi^{(t)})}{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{x \in \{0,1\}} p(\mathbf{X}_i(A, \omega) = x, \text{par}(\mathbf{X}_i(A, \omega) = 0) \mid \mathbf{D}_i, \Phi^{(t)})}. \tag{8.8}$$

Proposition 36 *Let* $q_{(A,r)}$ *denote the probability of choosing rule* $r \in \mathcal{R}_A$. *The M-step update for the rule selection probability* $q_{(A,r)}$, $A \in \Sigma$, $r \in \mathcal{R}_A$ *is*

$$q_{(A,r)} = \frac{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} p(\mathbf{X}_i(A, \omega) = 1, \mathbf{R}_i(A, \omega) = \mathbb{I}(E_r) \mid \mathbf{D}_i, \Phi^{(t)})}{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{r' \in \mathcal{R}_A} p(\mathbf{X}_i(A, \omega) = 1, \mathbf{R}_i(A, \omega) = \mathbb{I}(E_{r'}) \mid \mathbf{D}_i, \Phi^{(t)})}. \tag{8.9}$$

*where E denotes a set of binary random variables indexed by $\mathcal{R}_A$, and $\mathbb{I}(E_r)$ denotes the setting $E_r = 1$, $E_{r'} = 0\ \forall r' \neq r$ (*i.e., *$\mathbb{I}(E_r)$ indicates that rule r was selected).*

To prove the propositions above, we require the following lemma:

Lemma 37 *In the the PSG framework, the $\mathcal{Q}$-distribution can be expressed in the form*

$$\begin{aligned}\mathcal{Q}(\Phi', \Phi^{(t)}) \;=\; & \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}[\log p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')] \\ & + \sum_{i=1}^{n} \sum_{(A,\omega)\in\mathcal{B}} \mathbb{E}_{p(\mathbf{X}_i(A,\omega),\mathrm{par}(\mathbf{X}_i(A,\omega))|\mathbf{D}_i,\Phi^{(t)})}\Big[\log \Psi^{L}_{\epsilon_A}(x_{\mathrm{par}(\mathbf{X}_i(A,\omega))}, x_{\mathbf{X}_i(A,\omega)})\Big] \\ & + \sum_{i=1}^{n} \sum_{(A,\omega)\in\mathcal{B}} \mathbb{E}_{p(\mathbf{X}_i(A,\omega),\mathbf{R}_i(A,\omega)|\mathbf{D}_i,\Phi^{(t)})}\Big[\log \Psi^{S}_{q_A}(x_{\mathbf{X}_i(A,\omega)}, x_{\mathbf{R}_i(A,\omega)})\Big] \\ & + \sum_{i=1}^{n} \sum_{(A,\omega)\in\mathcal{B}} \mathbb{E}_{p(\mathbf{R}_i(A,\omega),\mathbf{C}_i(A,\omega)|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{\substack{r\in\mathcal{R}_A \\ 1\le j\le n_r}} \log p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r), \Phi')\Big].\end{aligned} \tag{8.10}$$

Proof of Lemma 37

Recall from Eqn. 4.4 that we express the prior distribution over scenes in a factorized form. Substituting Eqn. 4.4 into Eqn. 8.7 yields

$$\begin{aligned}\mathcal{Q}(\Phi', \Phi^{(t)}) \;=\; & \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}[\log p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')] \\ & + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{(A,\omega)\in\mathcal{B}} \log p(\mathbf{X}_i(A,\omega) \mid \mathrm{par}(\mathbf{X}_i(A,\omega)), \Phi')\Big] \\ & + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{(A,\omega)\in\mathcal{B}} \log p(\mathbf{R}_i(A,\omega) \mid \mathbf{X}_i(A,\omega), \Phi')\Big] \\ & + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{\substack{(A,\omega)\in\mathcal{B} \\ r\in\mathcal{R}_A \\ 1\le j\le n_r}} \log p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r), \Phi')\Big].\end{aligned} \tag{8.11}$$

Simplifying,

$$
\begin{aligned}
\mathcal{Q}(\Phi', \Phi^{(t)}) \;=\; & \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}[\log p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')] \\
& + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{X}_i,\mathbf{C}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{(A,\omega)\in\mathcal{B}} \log p(\mathbf{X}_i(A,\omega) \mid \mathrm{par}(\mathbf{X}_i(A,\omega)), \Phi')\Big] \\
& + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{X}_i,\mathbf{R}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{(A,\omega)\in\mathcal{B}} \log p(\mathbf{R}_i(A,\omega) \mid \mathbf{X}_i(A,\omega), \Phi')\Big] \\
& + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{R}_i,\mathbf{C}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{\substack{(A,\omega)\in\mathcal{B} \\ r\in\mathcal{R}_A \\ 1\le j\le n_r}} \log p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r), \Phi')\Big].
\end{aligned} \tag{8.12}
$$

The conditional terms $p(\mathbf{X}_i(A,\omega) \mid \mathrm{par}(\mathbf{X}_i(A,\omega)), \Phi')$ and $p(\mathbf{R}_i(A,\omega) \mid \mathbf{X}_i(A,\omega), \Phi')$ can be expressed in terms of a Leaky-OR potential (Definition 12) and a Selection potential (Defintion 13), respectively, using a subset of the model parameters. Substituting the form of the potential functions yields

$$
\begin{aligned}
\mathcal{Q}(\Phi', \Phi^{(t)}) \;=\; & \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}[\log p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')] \\
& + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{X}_i,\mathbf{C}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{(A,\omega)\in\mathcal{B}} \log \Psi^{L}_{\epsilon_A}(x_{\mathrm{par}(\mathbf{X}_i(A,\omega))}, x_{\mathbf{X}_i(A,\omega)})\Big] \\
& + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{X}_i,\mathbf{R}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{(A,\omega)\in\mathcal{B}} \log \Psi^{S}_{q_A}(x_{\mathbf{X}_i(A,\omega)}, x_{\mathbf{R}_i(A,\omega)})\Big] \\
& + \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{R}_i,\mathbf{C}_i|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{\substack{(A,\omega)\in\mathcal{B} \\ r\in\mathcal{R}_A \\ 1\le j\le n_r}} \log p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r), \Phi')\Big].
\end{aligned} \tag{8.13}
$$

Using the linearity of expectations,

$$\begin{aligned}\mathcal{Q}(\Phi', \Phi^{(t)}) \;=\; & \sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{Z}_i|\mathbf{D}_i,\Phi^{(t)})}[\log p(\mathbf{D}_i \mid \mathbf{Z}_i, \Phi')] \\ & + \sum_{i=1}^{n} \sum_{(A,\omega)\in\mathcal{B}} \mathbb{E}_{p(\mathbf{X}_i(A,\omega),\mathrm{par}(\mathbf{X}_i(A,\omega))|\mathbf{D}_i,\Phi^{(t)})}\Big[\log \Psi^{L}_{\epsilon_A}(x_{\mathrm{par}(\mathbf{X}_i(A,\omega))}, x_{\mathbf{X}_i(A,\omega)})\Big] \\ & + \sum_{i=1}^{n} \sum_{(A,\omega)\in\mathcal{B}} \mathbb{E}_{p(\mathbf{X}_i(A,\omega),\mathbf{R}_i(A,\omega)|\mathbf{D}_i,\Phi^{(t)})}\Big[\log \Psi^{S}_{q_A}(x_{\mathbf{X}_i(A,\omega)}, x_{\mathbf{R}_i(A,\omega)})\Big] \\ & + \sum_{i=1}^{n} \sum_{(A,\omega)\in\mathcal{B}} \mathbb{E}_{p(\mathbf{R}_i(A,\omega),\mathbf{C}_i(A,\omega)|\mathbf{D}_i,\Phi^{(t)})}\Big[\sum_{\substack{r\in\mathcal{R}_A \\ 1\le j\le n_r}} \log p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r), \Phi')\Big].\end{aligned} \tag{8.14}$$

■

Proof of Proposition 35

The M-step involves optimizing the $\mathcal{Q}$-distribution with respect to the model parameters. Consider setting the self-rooting parameters ϵ_A, $A \in \Sigma$, to optimize the factorized $\mathcal{Q}$-distribution given in Eqn. 8.10. From the definition of the Leaky-OR potential in Definition 12, the self-rooting parameter is used only when all the input values are zero. Hence, in fitting ϵ_A, we only need to consider the case $\mathrm{par}(\mathbf{X}_i(A,\omega)) = 0$, $1 \le i \le n$, $\omega \in \Omega_A$.

Substituting the form of the Leaky-OR potential for the case $\mathrm{par}(\mathbf{X}_i(A,\omega)) = 0$ into Eqn. 8.10 and taking the partial derivative with respect to ϵ_A,

$$\begin{aligned}&\frac{\partial \mathcal{Q}(\Phi', \Phi^{(t)})}{\partial \epsilon_A} = \\ &\sum_{i=1}^{n} \sum_{\omega\in\Omega_A} \sum_{x\in\{0,1\}} p(\mathbf{X}_i(A,\omega) = x, \mathrm{par}(\mathbf{X}_i(A,\omega) = 0) \mid \mathbf{D}_i, \Phi^{(t)}) \Big(\frac{x}{\epsilon_A} - \frac{1-x}{1-\epsilon_A}\Big)\end{aligned} \tag{8.15}$$

where we have written the form of the expectation under the posterior explicitly. Setting the derivative to zero and solving for ϵ_A yields

$$\epsilon_A \;=\; \frac{\sum_{i=1}^{n} \sum_{\omega\in\Omega_A} p(\mathbf{X}_i(A,\omega) = 1, \mathrm{par}(\mathbf{X}_i(A,\omega) = 0) \mid \mathbf{D}_i, \Phi^{(t)})}{\sum_{i=1}^{n} \sum_{\omega\in\Omega_A} \sum_{x\in\{0,1\}} p(\mathbf{X}_i(A,\omega) = x, \mathrm{par}(\mathbf{X}_i(A,\omega) = 0) \mid \mathbf{D}_i, \Phi^{(t)})}. \tag{8.16}$$

■

Proof of Proposition 36

Consider setting the parameter q_A to optimize the factorized $\mathcal{Q}$-distribution given in Eqn. 8.10. From the definition of the Selection potential in Definition 13, the selection probabilities are used only when the binary input value is 1. Hence, in fitting q_A, we need only consider the case where $\mathbf{X}_i(A,\omega) = 1, 1 \leq i \leq n, \omega \in \Omega_A$.

Recall that q_A represents rule selection probabilities, so we have the constraint $\sum_{r \in \mathcal{R}_A} q_{(A,r)} = 1$. Hence, optimizing the $\mathcal{Q}$-distribution with respect to q_A is a constrained optimization problem. Recall that the method of Lagrange multipliers is a method that allows one to find local maxima/minima of a function subject to equality constraints. Using the method of Lagrange multipliers, we seek to maximize the Lagrange function $\mathcal{L}(\Phi', \Phi^{(t)})$:

$$\mathcal{L}(\Phi', \Phi^{(t)}) = \mathcal{Q}(\Phi', \Phi^{(t)}) - \lambda(\sum_{r \in \mathcal{R}_A} q_{(A,r)} - 1). \tag{8.17}$$

Taking the partial derivative of Eqn. 8.17 with respect to $q_{(A,r)}$,

$$\frac{\partial \mathcal{L}(\Phi', \Phi^{(t)})}{\partial q_{(A,r)}} = \frac{\partial \mathcal{Q}(\Phi', \Phi^{(t)})}{\partial q_{(A,r)}} - \lambda. \tag{8.18}$$

Substituting the form of the Selection potential for the case $\mathbf{X}_i(A,\omega) = 1$ into Eqn. 8.10, and taking the partial derivative with respect to $q_{(A,r)}$,

$$\frac{\partial \mathcal{Q}(\Phi', \Phi^{(t)})}{\partial q_{(A,r)}} = \sum_{i=1}^{n} \sum_{\omega \in \Omega_A} p(\mathbf{X}_i(A,\omega) = 1, \mathbf{R}_i(A,\omega) = \mathbb{I}(E_r) \mid \mathbf{D}_i, \Phi^{(t)}) \Big(\frac{1}{q_{(A,r)}} \Big). \tag{8.19}$$

Now, plug-in Eqn. 8.19 into Eqn. 8.18, set $\frac{\partial \mathcal{L}(\Phi', \Phi^{(t)})}{\partial q_{(A,r)}} = 0$, and solve for $q_{(A,r)}$. This yields the solution

$$q_{(A,r)} = \frac{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} p(\mathbf{X}_i(A,\omega) = 1, \mathbf{R}_i(A,\omega) = \mathbb{I}(E_r) \mid \mathbf{D}_i, \Phi^{(t)})}{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{r' \in \mathcal{R}_A} p(\mathbf{X}_i(A,\omega) = 1, \mathbf{R}_i(A,\omega) = \mathbb{I}(E_{r'}) \mid \mathbf{D}_i, \Phi^{(t)})}. \tag{8.20}$$

■

We now detail how to fit the conditional pose distributions γ, which can either represent a $\mathrm{Categorical}$ distribution or an $\mathrm{IndBern}$ distribution. In both cases, distributions γ are governed by parameters θ.

Below, we assume that the operation of subtraction is defined on the pose spaces $\Omega_A \in \Omega$. For example, a pose $\omega \in \Omega_A$ for a brick could represent a vector. We use the notation

$$K_{(r,j)} = \{\Delta \mid \Delta = \omega - z, \omega \in \Omega_{A_{(r,0)}}, z \in \Omega_{A_{(r,j)}}\}.$$

$K_{(r,j)}$ represents the set of possible differences between the pose of a brick of type $A_{(r,0)}$ and the pose of a brick of type $A_{(r,j)}$ for a rule r and the j-th symbol in the RHS of the rule.

In practice, it may be helpful to tie together parameters of the model to represent invariances in the model. One such invariance often useful in computer vision is *shift-invariance*. For example, consider selecting the location for the mouth of a face; the distribution over the location of the mouth may be most naturally expressed relative to the centre of the face. Below, we derive updates for θ when the conditional pose distributions they parameterize are shift-invariant.

We can reparameterize the parameters θ to represent shift invariance. Consider the parameters $\theta_{(\omega,r,j)}$, $r \in \mathcal{R}$, $1 \leq j \leq n_r$, $\omega \in \Omega_{A_{(r,0)}}$. We tie together the set of parameters of $\{\theta_{(\omega,r,j)} \mid \omega \in \Omega_{A_{(r,0)}}\}$ so that $\theta_{(\omega,r,j,z)} = \theta_{(\omega',r,j,z')}$ whenever $(\omega - z) = (\omega' - z')$. Now, associate with each $\Delta \in K_{(r,j)}$ a parameter $\hat{\theta}_{(\Delta,r,j)}$. The parameters $\theta_{(\omega,r,j)}$ can be written in terms of $\hat{\theta}_{(\Delta,r,j)}$. Concretely, $\theta_{(\omega,r,j,z)} = \hat{\theta}_{(\Delta,r,j)}$ where $\Delta = \omega - z$. Note that a setting for the parameters $\hat{\theta}$ implies a setting for the parameters θ.

Proposition 38 *Suppose the set of conditional pose distributions $\{\gamma_{(\omega,r,j)} \mid \omega \in \Omega_{A_{(r,0)}}\}$, $r \in \mathcal{R}$, $1 \leq j \leq n_r$, is a set of shift-invariant* Categorical *distributions. This implies that the conditional pose distributions $p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r), \mathbf{D}_i, \Phi^{(t)})$, $\omega \in \Omega_{A_{(r,0)}}$, $r \in \mathcal{R}$, $1 \leq j \leq n_r$ are represented by Selection potentials. The M-step update for the parameter $\hat{\theta}_{(\Delta,r,j)}$ is given by*

$$\hat{\theta}_{(\Delta,r,j)} = \frac{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} p(\mathbf{C}_i(A,\omega,r,j) = \mathbb{I}(E_z), \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})}{\sum_{\Delta' \in K} \sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta'}} p(\mathbf{C}_i(A,\omega,r,j) = \mathbb{I}(E_z), \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})} \tag{8.21}$$

*where E denotes a set of binary random variables indexed by $\Omega_{A_{(r,j)}}$ and $\mathbb{I}(E_z)$ denotes the setting $E_z = 1$, $E_{z'} = 0\ \forall z' \neq z$ (*i.e.*, $\mathbb{I}(E_z)$ indicates that pose z was selected).*

Proposition 39 *Suppose the set of conditional pose distributions $\{\gamma_{(\omega,r,j)} \mid \omega \in \Omega_{A_{(r,0)}}\}$, $r \in \mathcal{R}$, $1 \leq j \leq n_r$, is a set of shift-invariant* IndBern *distributions. This implies that the conditional pose distributions $p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r), \mathbf{D}_i, \Phi^{(t)})$, $\omega \in \Omega_{A_{(r,0)}}$, $r \in \mathcal{R}$, $1 \leq j \leq n_r$, are represented by a Berns potential. The M-step update for the parameter $\hat{\theta}_{(\Delta,r,j)}$ is given by*

$$\hat{\theta}_{(\Delta,r,j)} = \frac{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} p(\mathbf{C}_i(A,\omega,r,j,z) = 1, \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})}{n \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} 1}. \tag{8.22}$$

Proof of Proposition 38

From the definition of the Selection potential in Definition 13, the parameters $\theta_{(\omega,r,j)}$ (and so $\hat{\theta}_{(\Delta,r,j)}$), are used only when the binary input value is 1. Hence, we need only consider cases when $\mathbf{R}_i(A,\omega,r) = 1, 1 \leq i \leq n$.

Here, the set of parameters $\{\hat{\theta}_{(\Delta,r,j)} \mid \Delta \in K_{(r,j)}\}$ represents the selection probabilities of a Categorical distribution, so we have the constraint $\sum_{\Delta \in K} \hat{\theta}_{(\Delta,r,j)} = 1$. Hence, optimizing the $\mathcal{Q}$-distribution with respect to $\hat{\theta}_{(\Delta,r,j)}$ can be expressed as a constrained optimization problem.

Similar to updating the parameters q, we employ the method of Lagrange multipliers to maximize the Lagrange function

$$\mathcal{L}(\Phi', \Phi^{(t)}) = \mathcal{Q}(\Phi', \Phi^{(t)}) - \lambda(\sum_{\Delta \in K_{(r,j)}} \hat{\theta}_{(\Delta,r,j)} - 1). \tag{8.23}$$

Taking the partial derivative of Eqn. 8.23 with respect to $\hat{\theta}_{(\Delta,r,j)}$,

$$\frac{\partial \mathcal{L}(\Phi', \Phi^{(t)})}{\partial \hat{\theta}_{(\Delta,r,j)}} = \frac{\partial \mathcal{Q}(\Phi', \Phi^{(t)})}{\partial \hat{\theta}_{(\Delta,r,j)}} - \lambda. \tag{8.24}$$

Substituting the form of the Selection potential for the case $\mathbf{R}_i(A,\omega,r) = 1$ into Eqn. 8.10, and taking the partial derivative with respect to $\hat{\theta}_{(\Delta,r,j)}$,

$$\frac{\partial \mathcal{Q}(\Phi', \Phi^{(t)})}{\partial \hat{\theta}_{(\Delta,r,j)}} = \sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} \frac{p(\mathbf{C}_i(A,\omega,r,j) = \mathbb{I}(E_z), \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})}{\hat{\theta}_{(\Delta,r,j)}} \tag{8.25}$$

where we have written the form of the expectation under the posterior explicitly. Now, plug-in Eqn. 8.25 into Eqn. 8.24, set $\frac{\partial \mathcal{L}(\Phi', \Phi^{(t)})}{\partial \hat{\theta}_{(\Delta,r,j)}} = 0$, and solve for $\hat{\theta}_{(\Delta,r,j)}$. This yields the solution

$$\hat{\theta}_{(\Delta,r,j)} = \frac{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} p(\mathbf{C}_i(A,\omega,r,j) = \mathbb{I}(E_z), \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})}{\sum_{\Delta' \in K} \sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta'}} p(\mathbf{C}_i(A,\omega,r,j) = \mathbb{I}(E_z), \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})}. \tag{8.26}$$

■

Proof of Proposition 39

From the definition of the IndBern potential in Definition 14, the parameters $\theta_{(\omega,r,j)}$ (and so $\hat{\theta}_{(\Delta,r,j)}$), are only used when the binary input value is 1. Hence, we need only consider cases when $\mathbf{R}_i(A,\omega,r) = 1, 1 \leq i \leq n$.

Unlike in Proposition 38, the parameters here do not have a constraint. Hence, we can directly optimize the $\mathcal{Q}$-distribution with respect to $\hat{\theta}_{(\Delta,r,j)}$.

Substituting the form of the Berns potential for the case $\mathbf{R}_i(A,\omega,r) = 1$ into Eqn. 8.10 and taking the partial derivative with respect to $\hat{\theta}_{(\Delta,r,j)}$,

$$\frac{\partial \mathcal{Q}(\Phi', \Phi^{(t)})}{\partial \hat{\theta}_{(\Delta,r,j)}} = \tag{8.27}$$
$$\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} \sum_{c' \in \{0,1\}} p(\mathbf{C}_i(A,\omega,r,j,z) = c', \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)}) \Big(\frac{c'}{\hat{\theta}_{(\Delta,r,j)}} \Big)$$
$$- \sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} \sum_{c' \in \{0,1\}} p(\mathbf{C}_i(A,\omega,r,j,z) = c', \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)}) \Big(\frac{1-c'}{1-\hat{\theta}_{(\Delta,r,j)}} \Big)$$

where we have written the form of the expectation under the posterior explicitly. Setting the derivative to zero and solving for $\hat{\theta}_{(\Delta,r,j)}$ yields

$$\hat{\theta}_{(\Delta,r,j)} = \frac{\sum_{i=1}^{n} \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} p(\mathbf{C}_i(A,\omega,r,j,z) = 1, \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})}{n \sum_{\omega \in \Omega_A} \sum_{\substack{z \in \Gamma_{(\omega,r,j)}: \\ \omega - z = \Delta}} 1}. \tag{8.28}$$

■

8.3.2 Approximate E-step

In this subsection, we describe an approximation scheme to compute the posterior quantities necessary for the M-step updates of the model parameters Φ.

Recall that in addition to computing approximate marginals over a single random variable, the results of LBP can be used to compute approximate joint marginals over a set of random variables.

Definition 40 *Let f be a factor node with neighbours $U = N(f)$ and potential Ψ_f. Let x_U denote an outcome for the variables in U. LBP approximates the joint marginal disttribution $p(x_U)$ by*

$$\hat{p}(x_U) \propto \Psi_f(x_U) \prod_{u \in U} \mu_{u \to f}(x_u) \tag{8.29}$$

where the distribution is normalized to sum to one over all possible settings of x_U.

When the factor graph contains no loops $\hat{p}(x_U)$ matches the true joint marginal. However, the factor graphs considered in the PSG framework generally contain loops, and so $\hat{p}(x_U)$ is generally only an approximation to the true joint marginal. Nevertheless, in practice, $\hat{p}(x_U)$ serves as a useful approximation.

The main result of this subsection is stated below.

Proposition 41 *Consider the posterior quantities necessary for the M-step updates of Φ. All such posterior quantities can be approximated using the messages of LBP and Eqn. 8.29.*

Proof of Proposition 41

The posterior quantities we seek to approximate are

- $p(\mathbf{X}_i(A,\omega), \text{par}(\mathbf{X}_i(A,\omega) = 0) \mid \mathbf{D}_i, \Phi^{(t)})$ for $(A,\omega) \in \mathcal{B}, 1 \leq i \leq n$,
- $p(\mathbf{X}_i(A,\omega) = 1, \mathbf{R}_i(A,\omega) = \mathbb{I}(E_r) \mid \mathbf{D}_i, \Phi^{(t)})$ for $(A,\omega) \in \mathcal{B}, r \in \mathcal{R}_A, 1 \leq i \leq n$,
- $p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})$ for $(A,\omega) \in \mathcal{B}, r \in \mathcal{R}_A, j \in \Omega_{A_{(r,j)}}, 1 \leq i \leq n$.

Let U be a set of random variables that participates in a posterior quantity we seek to approximate. To prove the proposition, it is sufficient to show that there exists a factor f with $U \subseteq N(f)$ for each posterior quantity of interest. Recall the factor graph representation from Figure 4.2.

For the posterior quantities of the form $p(\mathbf{X}_i(A,\omega), \text{par}(\mathbf{X}_i(A,\omega) = 0) \mid \mathbf{D}_i, \Phi^{(t)})$, the factors $f^1_{(A,\omega)}$ contain the random variables $\mathbf{X}_i(A,\omega)$ and $\text{par}(\mathbf{X}_i(A,\omega))$ as neighbours.

For the posterior quantities of the form $p(\mathbf{X}_i(A,\omega) = 1, \mathbf{R}_i(A,\omega) = e_r \mid \mathbf{D}_i, \Phi^{(t)})$, the factors $f^2_{(A,\omega)}$ contain the random variables $p(\mathbf{X}_i(A,\omega))$ and $\mathbf{R}_i(A,\omega)$ as neighbours.

For the posterior quantities of the form $p(\mathbf{C}_i(A,\omega,r,j) \mid \mathbf{R}_i(A,\omega,r) = 1, \mathbf{D}_i, \Phi^{(t)})$, the factors $f^3_{(A,\omega,r,i)}$ contain the random variables $\mathbf{C}_i(A,\omega,r,j)$ and $\mathbf{R}_i(A,\omega,r)$ as neighbours.

■

To perform an approximate E-step in the PSG framework, we run LBP to convergence and use Eqn. 8.29 to approximate each posterior quantity of interest.

8.4 Effectiveness of approximate EM learning

Unlike the standard EM algorithm, the approximate EM algorithm we describe in this chapter is not guaranteed to increase a lower bound on the marginal likelihood of the observed data. Also, note that the approximate EM algorithm has two sources of approximation. First, recall that the E-step in the EM algorithm described in Section 8.2 requires computation of exact posterior quantities. Here, we use an E-step that computes approximate posterior quantities by running LBP. Second, recall that the factor graph construction described in Definition 16 leads to an exact representation of the distribution over scenes induced by a PSG only when the grammar is acyclic. When the grammar is cyclic, the factor graph construction leads to an approximate representation of the PSG , and so any learning algorithm defined using the factor graph construction is fitting a potentially different (but related) model. Despite the different sources of approximation, in practice, the approximate EM algorithm outlined here is effective in learning model parameters.

To demonstrate the effectiveness of the approximate EM algorithm described in this chapter, we show the performance of several PSG models on two scene understanding tasks as a function of approximate EM iteration. In Figure 8.1, we show performance in terms of area under a precision-recall curve (AUC) for several PSG models on the tasks of contour detection and image segmentation (see Chapter 9 for a full description of the models and tasks). Note that higher AUC indicates superior performance. As shown in the figure, performance tends to improve with subsequent approximate EM

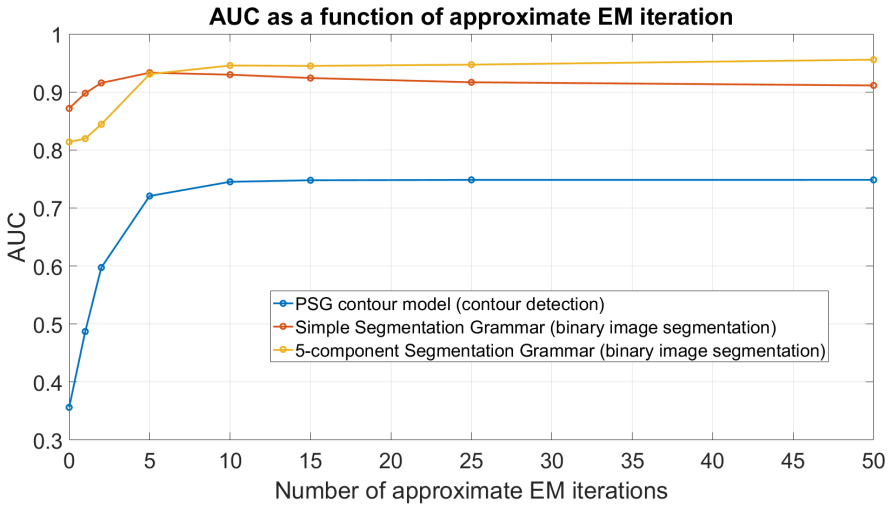


Figure 8.1: Area under the precision-recall curve (AUC) as a function of approximate EM iteration. Higher AUC indicates better performance. Each line in the plot corresponds to a PSG model for a particular scene understanding task indicated in brackets. Overall, the approximate EM algorithm described in this chapter seems to improve performance. See Chapter 9 for a full description of the models and tasks.

iterations. Although the performance decreases slightly for one of the models/tasks, the approximate EM algorithm generally seems to improve performance.

Chapter 9

Experiments

To demonstrate the generality of the PSG framework, we show experimental results on three different scene understanding tasks: contour detection, face localization, and binary image segmentation. As discussed in Chapter 1, previous approaches for these tasks have typically employed fairly distinct methods. Here, we demonstrate that the PSG framework can address all three problems. In particular, we describe PSGs for each scene understanding task in the language of Definition 1 and demonstrate that LBP can be used as the inference engine for these tasks. We use partially-supervised learning[1] to fit model parameters for contour detection and binary image segmentation, and supervised learning to fit model parameters for face localization. We report the speed of inference as performed on a laptop with an Intel® i7 2.5GHz CPU and 16 GB of RAM. Our framework is implemented in Matlab/C using a single thread.

All experiments were performed using a common and general implementation of the PSG framework. To handle the different tasks in this general implementation, one simply expresses an appropriate PSG in a high-level "language" like the one used in Chapter3 and designs an appropriate data model. The implementation automatically constructs the factor graph, and performs parameter estimation (learning) and inference.

9.1 Contour detection

To study contour detection, we use the Berkeley Segmentation Dataset (BSD500) described in [1] following the experimental setup described in [16]. The dataset contains natural images and object boundaries manually marked by human annotators. For our experiments, we used the standard split of the dataset with 200 training images and 200 test images. For each image we use the

[1] So-called because the supervision labels specify only the presence/absence of a subset of bricks.

boundaries marked by a single human annotator to define ground-truth binary contour maps. From a binary contour map $\mathbf{B}$ we generate a noisy real-valued image $\mathbf{D}$ by sampling each pixel $\mathbf{D}(i,j)$ independently from a Normal distribution whose mean depends on the value of $\mathbf{B}(i,j)$. Formally,

$$\mathbf{D}(i,j) \sim \mathcal{N}(\mu_{\mathbf{B}(i,j)}, \sigma). \tag{9.1}$$

For the experiments, we used $\mu_0 = 150$, and $\mu_1 = 100$, $\sigma = 40$.

9.1.1 The PSG contour model

The contour model we use in the experiments below is similar to the model described in Grammar 1, but with the model parameters $\{\epsilon, q\}$ learned in a partially-supervised approach. For learning, we treat the ground-truth contour maps $\mathbf{B}$ as observations for the grammar symbol INK, and use the approximate EM algorithm described in Chapter 8 to fit parameters. Note that we do not have fully observed data, since 1) we do not have observations for the states of the CURVE bricks, and 2) we do not have observations for the rule choices made by the bricks present in the scene. Recall that for approximate EM learning, we use LBP to compute approximations to the posterior quantities of interest during the E-step. To speed-up convergence of LBP, we use warm-starts between EM iterations. *I.e.*, the LBP messages for the E-step of EM iteration $t+1$ are initialized to the converged LBP messages from the E-step of EM iteration t.

Grammar 6 shows the learned contour model. We will refer to this contour model as the "PSG contour model".

Grammar 6 *PSG contour model: a grammar for contour detection learned in a partially-supervised setting. The function T_θ denotes a rotation in the plane by an angle θ and* Round *maps a point in the plane to the nearest grid point.*

$\Sigma = \{\mathsf{CURVE}, \mathsf{INK}\}$.

$\Omega_{\mathsf{CURVE}} = [N] \times [M] \times [8]$.

$\Omega_{\mathsf{INK}} = [N] \times [M]$.

Rules:

$$
\begin{array}{llll}
0.647, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow & (\mathsf{INK}, \delta((x, y))), (\mathsf{CURVE}, \delta(((x, y) + \mathrm{Round}(T_\theta(1, 0)), \theta))) \\
0.147, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow & (\mathsf{INK}, \delta((x, y))), (\mathsf{CURVE}, \delta(((x, y) + \mathrm{Round}(T_\theta(1, -1)), \theta))) \\
0.152, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow & (\mathsf{INK}, \delta((x, y))), (\mathsf{CURVE}, \delta(((x, y) + \mathrm{Round}(T_\theta(1, +1)), \theta))) \\
0.019, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow & (\mathsf{CURVE}, \delta((x, y, \theta - 1))) \\
0.019, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow & (\mathsf{CURVE}, \delta((x, y, \theta + 1))) \\
0.012, & (\mathsf{CURVE}, (x, y, \theta)) & \rightarrow & (\mathsf{INK}, \delta((x, y))) \\
1.00, & (\mathsf{INK}, (x, y)) & \rightarrow & \emptyset
\end{array}
$$

$\epsilon_{\mathsf{CURVE}} = 4.28 \times 10^{-5}$, $\epsilon_{\mathsf{INK}} = 1.87 \times 10^{-12}$.

Recall that for inference, we convert a PSG to a factor graph and run LBP. To incorporate the data model given in Eqn. 9.1 into the factor graph representation, we attach unary potentials to the set of variables nodes $\{\mathbf{X}(\mathsf{INK}, (i, j)) \mid (i, j) \in \Omega_{\mathsf{INK}}\}$. For a variable node $\mathbf{X}(\mathsf{INK}, (i, j))$ we attach a unary potential $f^0_{(\mathsf{INK},(i,j))}(\mathbf{X}(\mathsf{INK}, (i, j)) = x, \mathbf{D}) = \mathcal{N}(\mathbf{D}(i, j); \mu_x, \sigma)$. In this case, the resulting factor graph represents the conditional distribution $p(S \mid \mathbf{D})$.

9.1.2 Qualitative contour detection results

In Figure 9.1 we show contour detection results on examples from the BSDS500 test set. We show the approximate marginal probability that each pixel is part of a curve as computed by LBP, $\hat{p}(\mathbf{X}(\mathsf{INK}, (i, j)) = 1 \mid \mathbf{D})$. Running LBP to convergence on a 481×321 test image took on average 1.5 hours.

As shown in Figure 9.1, despite the PSG contour model's simplicity, it is able to perform reasonably well in detecting contours in noisy images. Note that the model sometimes has trouble localizing curved contours. Since the model is similar to a first-order Markov model, it is unable to faithfully model and capture high-order contour statistics, such as curvature, thus hampering its ability to detect contours that do not have low curvature. This suggests that modelling curvature is important for contour detection. As such, we believe more realistic models of contours will be able to capture richer curvature information and outperform the simple contour model described in 6. Nevertheless, the qualitative contour detection results demonstrate the feasibility of the PSG framework's approach on this task. In the next section, we provide a qualitative analysis of the performance of the PSG contour model.

Figures 9.2 shows more contour detection results on images from the BSDS500 at a larger resolution so the reader can examine more details.

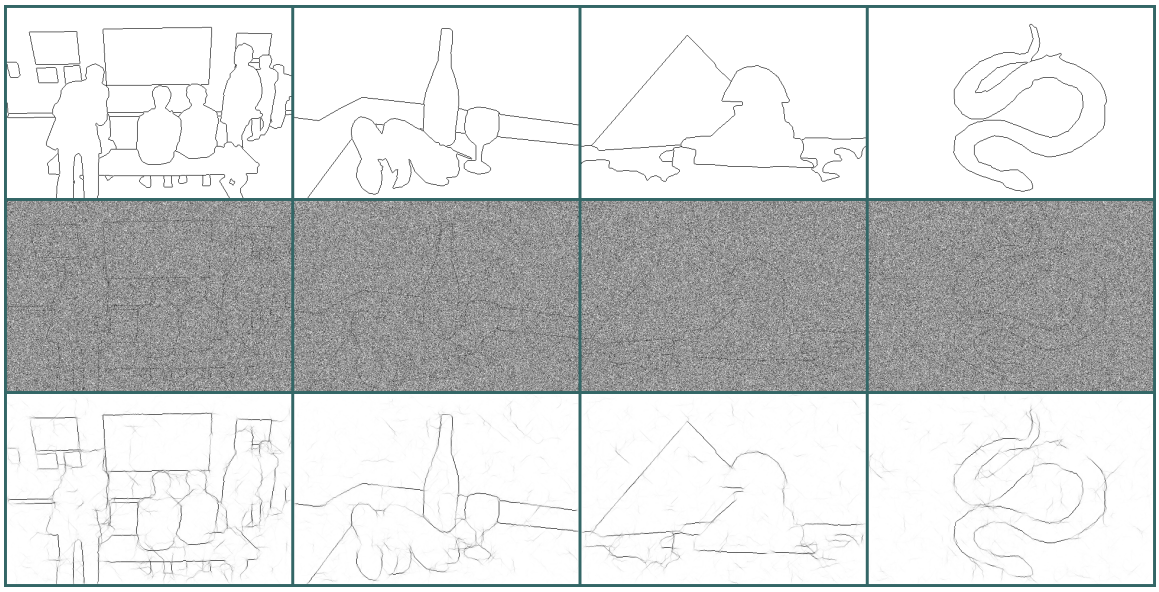


Figure 9.1: Contour detection results on four examples from the BSDS500 test set. **Top row**: Ground-truth contour maps $\mathbf{B}$. **Middle row**: Noisy observations, $\mathbf{D}$. **Bottom row**: Visualization of the approximate marginal probabilities of INK bricks present in the scene computed by LBP. Each pixel represents an INK brick at that location. The gray-scale values show the approximate marginal probabilities $\hat{p}(\mathbf{X}(\mathsf{INK}, (i,j)) = 1 \mid \mathbf{D})$ at each pixel (i,j) computed by LBP. Darker pixels indicate a higher marginal probability. Despite the PSG contour model's simplicity, the model is able to perform reasonably in detecting contours in noisy images. However, the model has trouble localizing curved contours; this is particularly evident in the left and rightmost examples.

9.1.3 Quantitative contour detection results

In this subsection we perform a quantitative comparison between the PSG contour model and baseline models. Our chief comparison is to the Field-of-Patterns (FOP) models of [16], wherein their proposed model is specifically designed to recover binary images. To demonstrate the importance of context for contour detection, we also compare against a PSG where all of the compositional rules are of the form $A \to \emptyset$; *i.e.* all bricks are independent. We will refer to this PSG as the "No-Context PSG " since such a PSG relies solely on the data model for contour detection. For the PSG contour model and the No-Context PSG, we compute the area under the precision-recall curve (AUC) by threshholding $\hat{p}(\mathbf{X}(\mathsf{INK}, (i,j)) = 1 \mid \mathbf{D})$, $(i,j) \in \Omega_{\mathsf{INK}}$, over a range of values. The authors of [16] have provided us with their experimental data, which we use to make comparisons.

Table 9.1 compares the AUC of the PSG contour model to baselines. Figure 9.3 compares the precision-recall curves of the PSG contour model and baseline methods. Comparing the AUC achieved by the PSG contour model to the No-Context PSG it is clear that the use of contextual

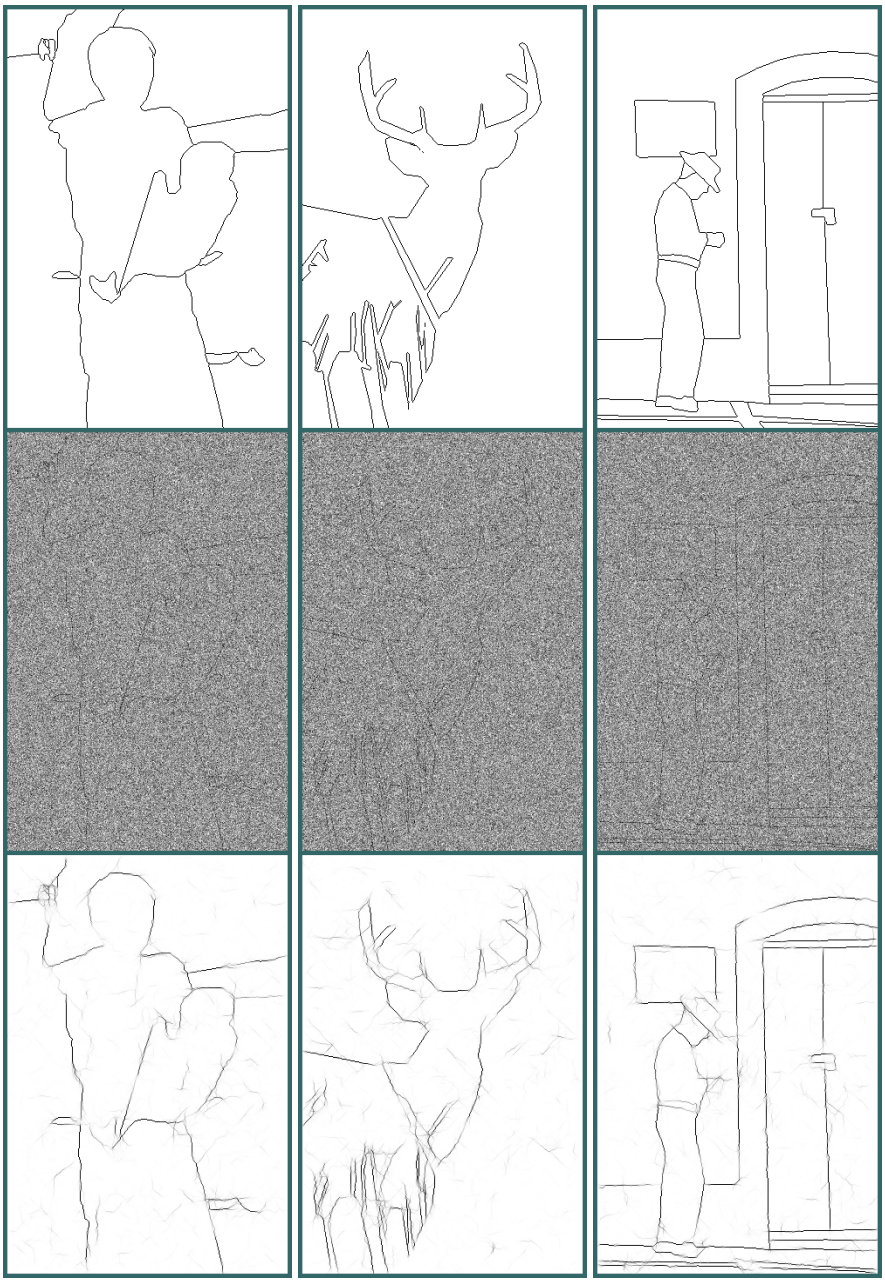


Figure 9.2: Contour detection results for three examples from the BSDS500 test set. **Top row**: Ground-truth contour maps $\mathbf{B}$. **Middle row**: Noisy observations, $\mathbf{D}$. **Bottom row**: Visualization of the approximate marginal probabilities of INK bricks present in the scene computed by LBP. Each pixel represents an INK brick at that location. The gray-scale values show the approximate marginal probabilities $\hat{p}(\mathbf{X}(\mathsf{INK}, (i,j)) = 1 \mid \mathbf{D})$ at each pixel (i,j) computed by LBP. Darker pixels indicate a higher marginal probability.

Model	AUC
No-Context PSG	0.12
PSG contour model	0.75
1-level FOP, [16]	0.73
4-level FOP, [16]	0.78

Table 9.1: AUC for the No-Context PSG, PSG contour model, and the 1-level and 4-level field-of-patterns (FOP) models from [16]. Note that the PSG contour model is competitive with the 1-level FOP model of [16] which is specifically designed to recover binary images from noisy images. While the 4-level FOP model outperforms the PSG contour model, the PSG contour model is still competitive despite the generality of the PSG framework. Note that the No-Context PSG performs poorly, demonstrating that the use of contextual information is crucial for contour detection.

information is of critical importance for high-quality contour detection. Note that the PSG contour model achieves competitive results compared to the FOP models described in [16], despite the PSG framework's general-purpose nature.

We believe it is possible to define more realistic models of contours in the PSG framework to improve performance. For example, one could make use of higher-order contour statistics such as curvature information, and operate in a multi-scale fashion similar to the 4-level FOP model; both of these concepts can in principle be described in the language of a PSG. However, the goal of this thesis is to demonstrate the generality of the PSG framework and we leave the design and structure learning of more sophisticated contour models for future research.

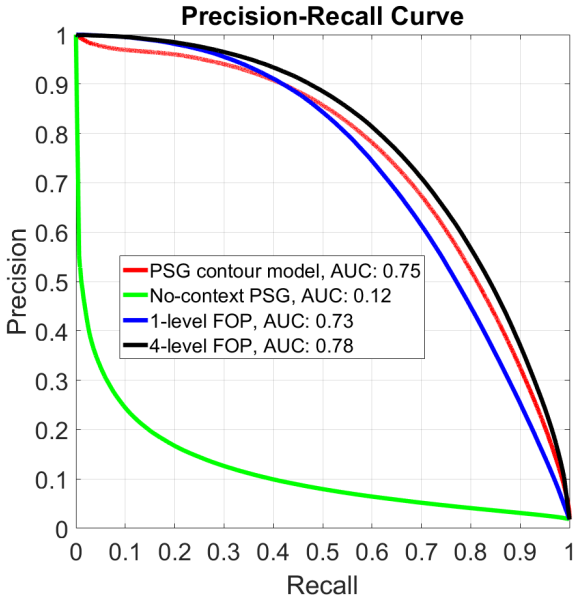


Figure 9.3: Precision-recall curves for the PSG contour model and baseline models. The FOP results were obtained from the authors of [16]. The AUC for each method is shown in the legend. The PSG contour model is competitive with the 1-level FOP model from [16], but is outperformed by the 4-level FOP model. The overall poor performance of the No-Context PSG demonstrates the importance of using a notion of context for contour detection.

9.2 Face Localization

The PSG framework can be applied to the problem of object localization. Here, we demonstrate the application of PSGs to the problem of face localization. The goal is to localize one or more faces in a set of images, as well as the faces' left eye, right eye, nose, and mouth. We study face localization on two datasets; one dataset has a single face per image, and the other has multiple faces per image. We describe the datasets below.

9.2.1 Dataset: Labelled Faces in the Wild

To study face localization when there is exactly one face in the image, we use the Labelled Faces in the Wild (LFW) dataset introduced in [27]. The dataset contains faces in unconstrained environments. We randomly select 200 images for training, and 100 images for testing. Although the dataset comes annotated with the identity of the person in the image, it does not come with part annotations. We manually annotate all training and test images with bounding box information for the face, left eye, right eye, nose, and mouth. Examples of bounding box annotations are shown Figure 9.4.

9.2.2 Dataset: Family Portraits

The LFW dataset images are constrained to have only one face per image and is not suitable for evaluating localization performance when there are multiple faces in an image. To study multiple face detection, we collect a dataset of 40 images of family and class portraits taken from the Internet. We used the search string "family portraits", "class portraits" and "school portraits" on Google™ in November 2016. We manually annotated each image with bounding box information for the face, left eye, right eye, nose, and mouth. Examples of bounding box annotations are shown in the Figure 9.5. On average, there are 5.9 faces per image. We refer to this dataset as "Portraits".

9.2.3 Face Detection Grammar

The PSG we use for face detection experiments is similar to the grammar described in Grammar 2, but with several differences:

- The EYE symbol in the grammar is replaced by LEFT-EYE and RIGHT-EYE symbols. Thus, the grammar distinguishes between left eyes and right eyes.
- Scale information is included in the pose space. This enables the grammar to express relationships such as "a small face has a small mouth that is only a few pixels below the centre of the face" and "a large face has a large mouth that can be many pixels below the centre of

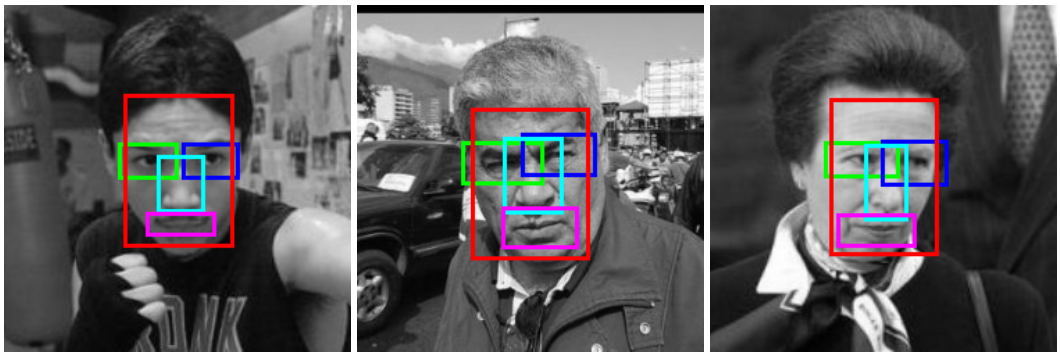


Figure 9.4: Examples of manually annotated images from the LFW dataset. Images are annotated with bounding boxes for the face (red), left eye (green), right eye (blue), nose (cyan), and mouth (magenta). Note that we distinguish between left and right eyes. All LFW images are 250×250 pixels.

the face". The pose space is defined so that objects detected at smaller scales can be localized with higher precision than objects detected at larger scales.

- The grammar does not use $\mathrm{Uniform}$ conditional pose distributions to express the geometric relationship between a face and its constituent parts. Instead, the conditional pose distributions are $\mathrm{Categorical}$ distributions whose parameters are learned in a supervised learning approach described later in Section 9.2.5

- The grammar contains symbols that represent the concept of "look-alikes". "Look-alike" symbols provide a mechanism for the PSG to handle false positives that arise due to weaknesses in the given data model. For example, a MOUTH "look-alike" brick represents an entity that merely looks like a mouth under the data model, but may not truly be a mouth. Given an image patch that looks like a mouth, there are two possibilities for the image patch: 1) the image patch is truly a mouth with other face parts nearby, or 2) the image patch only looks like a mouth with no other face parts nearby. The "look-alike" symbols explicitly model these possibilities in the grammar. We include corresponding look-alike symbols for the FACE LEFT-EYE, RIGHT-EYE, NOSE, and MOUTH symbols. Although not an integral part of the model, we have found that in practice, "look-alike" symbols improve performance by reducing false detections. The problem of false detections caused by weaknesses in the data model, especially those based on gradient information, is discussed in [62]. We will denote "look-alike" symbols with the prefix $\mathsf{T}-$.

We will refer to our PSG for face localization as the "PSG Face Grammar" model. The specification of the PSG Face Grammar is given in Grammar 7. We use L to denote the number of

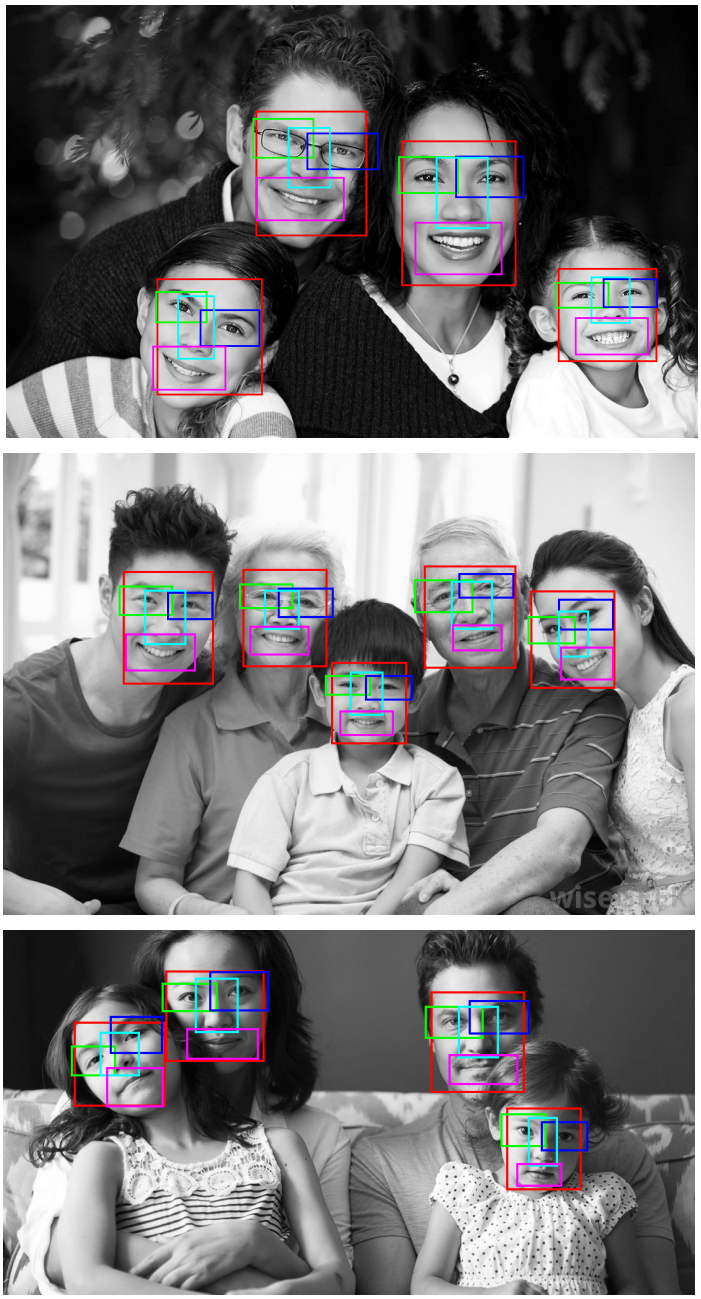


Figure 9.5: Examples of manually annotated images from the Portraits dataset. Images for both datasets are annotated with bounding boxes for the face (red), left eye (green), right eye (blue), nose (cyan), and mouth (magenta). Note that we distinguish between left and right eyes. The sizes of the images in the Portraits dataset is variable.

scales considered for all symbols in the grammar, and $[N_s] \times [M_s]$ denotes a grid of points at a scale $1 \leq s \leq L$.

Grammar 7 *The PSG Face Grammar:*

$\Sigma = \{\mathsf{FACE}, \mathsf{LEFT\text{-}EYE}, \mathsf{RIGHT\text{-}EYE}, \mathsf{NOSE}, \mathsf{MOUTH},$
$\mathsf{T\text{-}FACE}, \mathsf{T\text{-}LEFT\text{-}EYE}, \mathsf{T\text{-}RIGHT\text{-}EYE}, \mathsf{T\text{-}NOSE}, \mathsf{T\text{-}MOUTH}\}$
$\forall A \in \Sigma,\ \Omega_A = \bigcup_{s=1}^{L} \{[N_s] \times [M_s]\}.$
Rules:

$1.0,\quad (\mathsf{FACE}, \omega) \rightarrow (\mathsf{T\text{-}FACE}, \mathrm{Categorical}(\cdot \mid \theta_{(\omega,1,1)})),$
$(\mathsf{LEFT\text{-}EYE}, \mathrm{Categorical}(\cdot \mid \theta_{(\omega,1,2)})),$
$(\mathsf{RIGHT\text{-}EYE}, \mathrm{Categorical}(\cdot \mid \theta_{(\omega,1,3)})),$
$(\mathsf{NOSE}, \mathrm{Categorical}(\cdot \mid \theta_{(\omega,1,4)}),$
$(\mathsf{MOUTH}, \mathrm{Categorical}(\cdot \mid \theta_{(\omega,1,5)}))$

$1.0,\quad (\mathsf{LEFT\text{-}EYE}, \omega) \rightarrow (\mathsf{T\text{-}LEFT\text{-}EYE}, \delta(\omega))$

$1.0,\quad (\mathsf{RIGHT\text{-}EYE}, \omega) \rightarrow (\mathsf{T\text{-}RIGHT\text{-}EYE}, \delta(\omega))$

$1.0,\quad (\mathsf{NOSE}, \omega) \rightarrow (\mathsf{T\text{-}NOSE}, \delta(\omega))$

$1.0,\quad (\mathsf{MOUTH}, \omega) \rightarrow (\mathsf{T\text{-}MOUTH}, \delta(\omega))$

$1.0,\quad (\mathsf{T\text{-}LEFT\text{-}EYE}, \omega) \rightarrow \emptyset$

$1.0,\quad (\mathsf{T\text{-}RIGHT\text{-}EYE}, \omega) \rightarrow \emptyset$

$1.0,\quad (\mathsf{T\text{-}NOSE}, \omega) \rightarrow \emptyset$

$1.0,\quad (\mathsf{T\text{-}MOUTH}, \omega) \rightarrow \emptyset$

$\epsilon_{\mathsf{FACE}} = 10^{-4}$
$\epsilon_{\mathsf{LEFT\text{-}EYE}} = \epsilon_{\mathsf{RIGHT\text{-}EYE}} = \epsilon_{\mathsf{NOSE}} = \epsilon_{\mathsf{MOUTH}} = 10^{-12}$
$\epsilon_{\mathsf{T\text{-}FACE}} = \epsilon_{\mathsf{T\text{-}LEFT\text{-}EYE}} = \epsilon_{\mathsf{T\text{-}RIGHT\text{-}EYE}} = \epsilon_{\mathsf{T\text{-}NOSE}} = \epsilon_{\mathsf{T\text{-}MOUTH}} = 10^{-4}.$

9.2.4 Face data model

We incorporate image evidence for a brick (A, ω) to be present/absent in a scene by attaching a unary potential to the variable node $\mathbf{X}(A, \omega)$, $A \in \{$T-FACE,T-LEFT-EYE,T-RIGHT-EYE,T-NOSE, T-MOUTH$\}$, $\omega \in \Omega_A$ in the factor graph. In other words, only the "look-alike" symbols have an associated data model. Here, we describe the form of the factor $f^0_{(A,\omega)}$ we use for face and face part localization.

Given an image $\mathbf{Y}$, we denote a unary potential attached to $\mathbf{X}(A, \omega)$ by $f^0_{(A,\omega)}(\mathbf{X}(A, \omega), \mathbf{Y})$. We define the unary potentials for a symbol A using a histogram-of-oriented-gradients (HOG) filter (see

[9] for a description of HOG features). Let $H_{(A,\omega)}(\mathbf{Y})$ be the response of a HOG filter associated with symbol A and pose ω in an image $\mathbf{Y}$. We define

$$f^0_{(A,\omega)}(\mathbf{X}(A,\omega),\mathbf{Y}) = p_A(H_{(A,\omega)}(\mathbf{Y}) \mid \mathbf{X}(A,\omega)) \tag{9.2}$$

where p_A is a conditional distribution over discretized HOG detection scores for symbol A. The procedure to fit these distributions is described below.

Note that each brick associated with a unary potential is also associated with a HOG detection score, and so a HOG filter. Thus, each brick has a bounding box associated with it corresponding to the spatial extent of the HOG filter.

To build the data model, we first train HOG filters using publicly-available code from [12][2]. We train separate filters for each "look-alike" symbol using annotated bounding boxes to define positive examples. The negative examples are taken from the PASCAL VOC 2012 dataset described in [11], with images containing the class "People" removed. We use 10 scales per octave for each object/part and do **not** use hard negative mining. Figure 9.6 shows a visualization of the HOG filters learned using 200 images from the LFW as positive examples. For all face detection experiments, the positive examples are taken from the LFW dataset.

For a symbol A, to construct $p_A(\cdot \mid \mathbf{X}(A,\omega) = 1)$ we first obtain a set of detection scores by finding in each image the highest HOG detection score whose associated spatial extent has an intersection-over-union measure of at least 0.7 with the ground truth bounding box. We then clamp the detection scores to be in the range $[-2, 2]$, construct a 20-bin frequency histogram of detection scores, normalize the histogram to sum to 1, and finally smooth the distribution by a Gaussian kernel to obtain $p_A(\cdot \mid \mathbf{X}(A,\omega) = 1)$. To construct $p_A(\cdot \mid \mathbf{X}(A,\omega) = 0)$, we use a similar approach, but we use all the detection scores in each image as the set of HOG detection scores. Figure 9.7 shows the learned distributions $p_A(\cdot \mid \mathbf{X}(A,\omega) = 1)$ and $p_A(\cdot \mid \mathbf{X}(A,\omega) = 0)$.

9.2.5 Fitting model parameters

For all face detection experiments, we fit the conditional pose distributions of the PSG Face Grammar using the LFW dataset. To fit the conditional pose distributions, we use ground truth bounding information to provide supervision. For each face in the training set, we have its bounding box and the bounding box for its constituent parts. We convert each ground truth bounding box for the face, left eye, right eye, nose, and mouth into a pose for the corresponding symbol in the grammar. To do this, first recall that bricks has an associated bounding box. We select the pose associated with the

[2] https://cs.brown.edu/~pff/latent-release4/

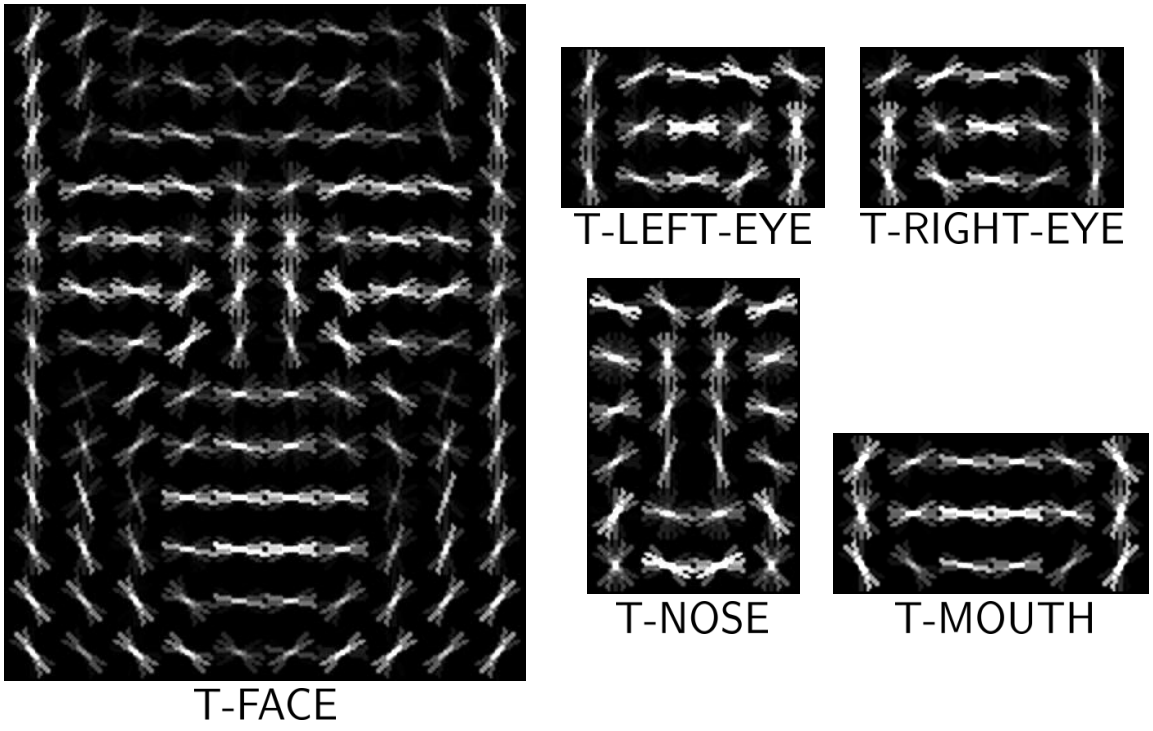


Figure 9.6: Visualization of the HOG filters learned using 200 examples from the LFW as positive examples. Note that the HOG filters for the T-LEFT-EYE and T-RIGHT-EYE symbols are subtly different, indicating there is a visual difference between the two parts. Also note that the T-MOUTH filter shares some similarities to both the T-LEFT-EYE and T-RIGHT-EYE filters, indicating that HOG filters may not be an ideal feature representation to distinguish between mouths and eyes.

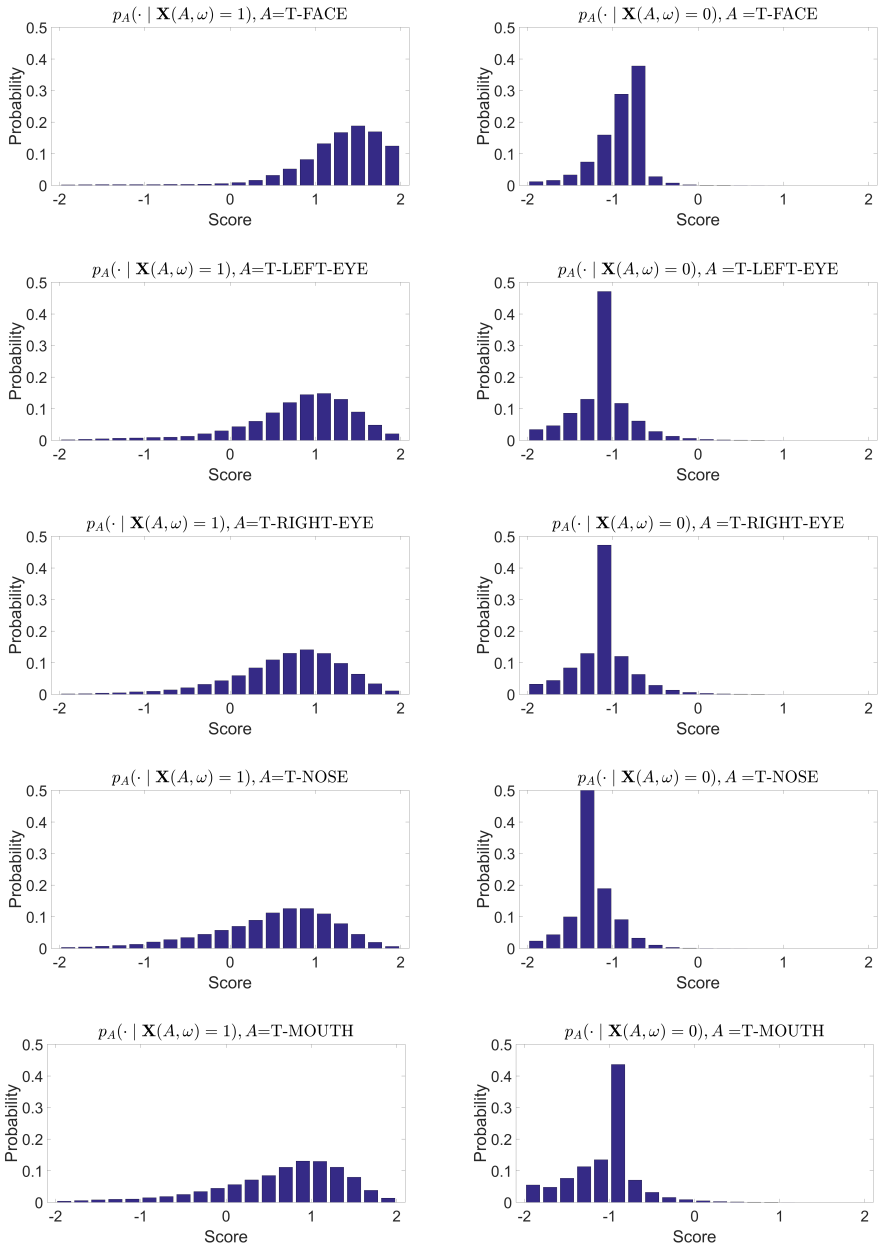


Figure 9.7: Distributions over HOG detections scores representing the data model.

highest detection score with an intersection-over-union measure of at least 0.7 as the ground truth bounding box. This process converts each annotated bounding box into a pose in the pose space of the corresponding symbol. Using this information, we can fit the conditional pose distributions using maximum likelihood estimation.

Note that since each symbol occurs only once in the left-hand-side of in the set of rules, there is no need learn the parameters q.

We keep the self-rooting probabilities fixed to those given in the PSG Face Grammar model (Grammar 7). Note that the parts of the face, $\{$LEFT-EYE, RIGHT-EYE, NOSE, MOUTH $\}$, have low self-rooting probability (10^{-12}), indicating that the model places low probability on the event that these symbols appear on their own. In contrast, the corresponding "look-alike" symbols have a much higher self-rooting probability (10^{-4}). As a result, an image region that looks like a face part but appears on its own is more likely to be explained as a self-rooting "look-alike" symbol rather than as a true face part.

9.2.6 Face localization results on single-face images: LFW

The data model and conditional pose distributions were fit using 200 annotated training examples from the LFW dataset. We use a separate 100 examples for testing.

The output of LBP with the PSG Face Grammar gives for each brick (A, ω) in the scene, an approximate marginal probability that the brick is present: $\hat{p}(\mathbf{X}(A, \omega) = 1 \mid \mathbf{Y})$. Since there is only a single face in each image in the LFW dataset, to perform face localization in an image, $\forall A \in \{$FACE, RIGHT-EYE, LEFT-EYE, NOSE, MOUTH$\}$ we output

$$\omega^* = \arg\max_{\omega \in \Omega_A} \hat{p}(\mathbf{X}(A, \omega) = 1 \mid \mathbf{Y}) \tag{9.3}$$

as the predicted pose for symbol A in the scene.

As baseline models, we use our own implementation of Pictorial Structures and a model that uses only the individual HOG filter scores to perform localization of each part independently. We refer to the latter approach as the "HOG Filters" approach.

To perform inference with Pictorial Structures, we use the MRF representation of a Pictorial Structures model described in Chapter 7. The symbols of the Pictorial Structures model are FACE, LEFT-EYE, RIGHT-EYE, NOSE, and MOUTH. The pose spaces for the symbols are the same as in the PSG Face Grammar. Since the MRF is acyclic, the marginal probabilities can be computed exactly using dynamic programming and we use Eqn. 9.3 to output a predicted pose for each symbol.

To perform inference using only HOG filter scores, the predicted pose for each symbol $A \in \{$FACE, LEFT-EYE, RIGHT-EYE, NOSE, MOUTH$\}$ given an image $\mathbf{Y}$ is

$$\omega^* = \arg\max_{\omega \in \Omega_A} \frac{p_A(H_{(A,\omega)}(\mathbf{Y}) \mid \mathbf{X}(A,\omega) = 1)}{p_A(H_{(A,\omega)}(\mathbf{Y}) \mid \mathbf{X}(A,\omega) = 0)}. \tag{9.4}$$

Inference for the PSG Face Grammar model using LBP on a 250×250 test image took 120 seconds, and inference for both Pictorial Structures and the HOG Filters baseline models took around 5 seconds. We show qualitative localization results in Figure 9.8.

As shown in Figure 9.8, the HOG Filters model performs poorly and often confuses mouths with left eyes and right eyes. This occurs in three of the four examples shown and can be attributed to the similarity of the learned HOG filters for mouths and eyes, as shown in Figure 9.6. In contrast, the PSG Face Grammar does not confuse mouths and eyes since it uses geometric information encoded in the conditional pose distributions to localize the parts of the face. The use of geometric structure and "look-alike" symbols help the PSG Face Grammar to compensate for the ambiguous data model and robustly perform face localization. Pictorial Structures performs similarly to the PSG Face Grammar on this dataset. This is to be expected since one major difference between Pictorial Structures and the PSG Face Grammar is that Pictorial Structures assumes there is only one object of each type per image, which is an accurate assumption for the LFW dataset. Comparing the results of the HOG Filters to the results of the PSG Face Grammar and Pictorial Structures models demonstrates contextual information is crucial for accurate face and face-part localization.

Table 9.2 provides a quantitative evaluation of the PSG Face Grammar model and the baseline models in terms of mean distance away from centre of the ground truth bounding box.

Model	FACE	LEFT-EYE	RIGHT-EYE	NOSE	MOUTH	**Average**
HOG Filters	3.7	4.7	8.2	3.3	13.6	6.7
Pictorial Structures	3.3	2.6	3.1	2.4	3.4	3.0
PSG Face Grammar	3.5	2.6	3.3	2.4	3.5	3.1

Table 9.2: Mean distance of top detections to the centre of the ground truth bounding box, in pixels, on the LFW dataset. A key difference between Pictorial Structures and the PSG Face Grammar is that Pictorial Structures assumes one object per image, while the PSG Face Grammar makes no such assumption. However, in the LFW dataset, there is indeed only one face per image, and so the two models perform similarly on this dataset.

Table 9.3 provides an evaluation in terms of area under the precision-recall curves. For this evaluation, we perform non-maximum suppression for the symbols FACE, LEFT-EYE, RIGHT-EYE, NOSE, MOUTH separately. We first sort the detection probabilities for each symbol, then perform suppression so that no two detections of the same symbol overlap. We consider a detection to be a true positive if it is the highest scoring detection with an intersection-over-union ratio of at least 0.5 with the ground truth bounding box. We consider a detection a false positive if it is not the

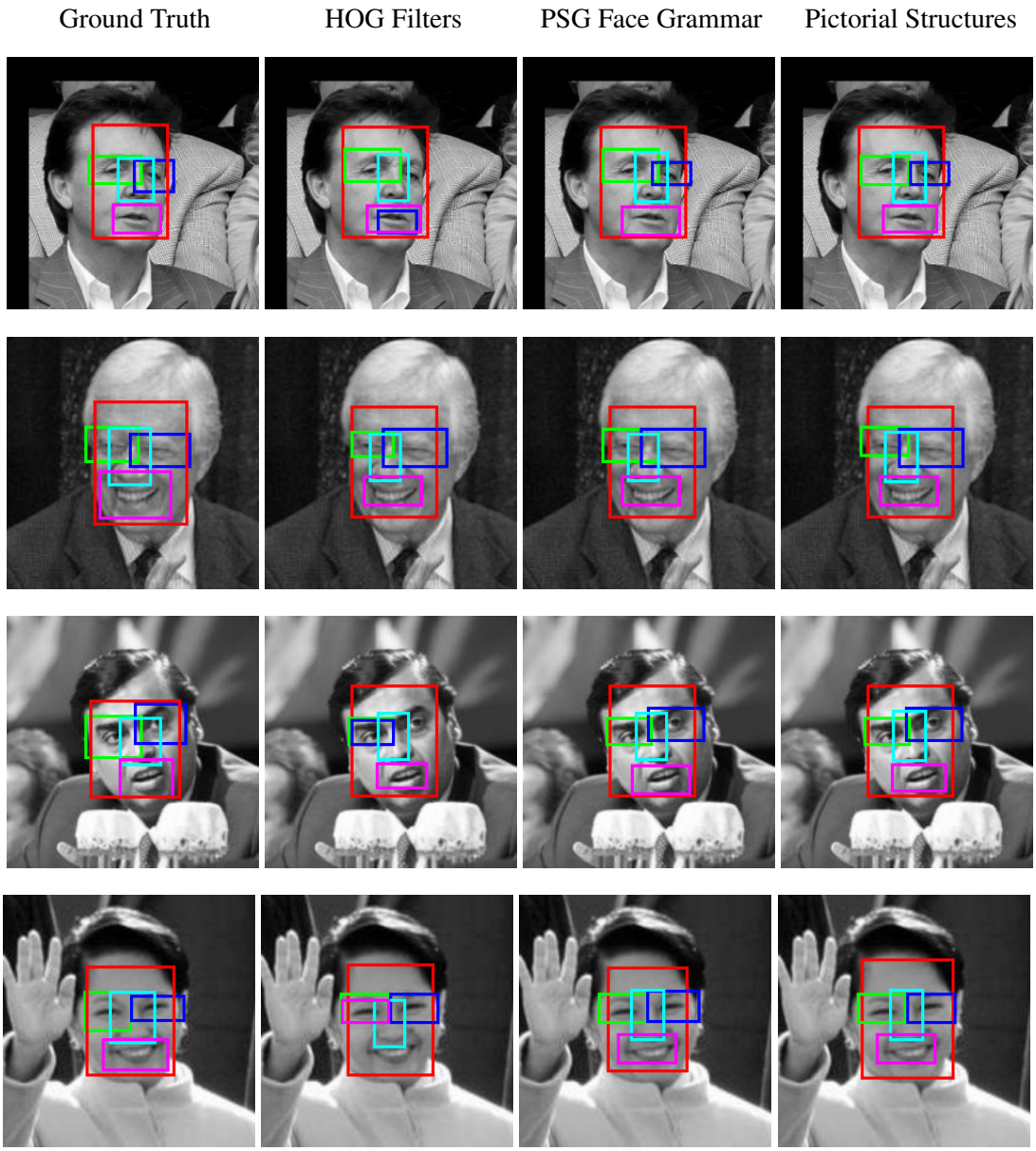


Figure 9.8: Localization results on four examples from the LFW dataset. **Left**: annotated ground-truth bounding boxes. **Middle-Left**: results of the HOG Filters model. **Middle-Right**: results of the PSG Face Grammar model. **Right**: results of Pictorial Structures. The parts are FACE (red), LEFT-EYE (green), RIGHT-EYE (blue), NOSE (cyan), and MOUTH (magenta). For each symbol, we show the bounding box corresponding to the pose with the highest computed marginal probability. Note that both the PSG Face Grammar model and Pictorial Structures perform well while the HOG Filter model performs poorly in some cases, suggesting that using geometric information is crucial for accurate localization.

highest scoring detection with an intersection-over-union ratio of at least 0.5 with the ground truth bounding box, thus penalizing multiple detections of the same object. Once again, the HOG Filters model performs poorly while the PSG Face Grammar model and Pictorial Structures perform well, demonstrating the importance of geometric information.

Model	FACE	LEFT-EYE	RIGHT-EYE	NOSE	MOUTH	**Average**
HOG Filters	1.00	0.76	0.65	0.96	0.60	0.80
Pictorial Structures	1.00	0.97	0.93	0.98	0.90	0.96
PSG Face Grammar	1.00	0.98	0.92	0.98	0.92	0.96

Table 9.3: Area under the precision-recall curve on the LFW dataset. Note that the HOG Filters model performs significantly worse than the PSG Face Grammar model and Pictorial Structures, demonstrating the importance of contextual information for accurate object localization. The PSG Face Grammar model and Pictorial Structures perform similarly, as is expected since there is only one face per image in this dataset.

9.2.7 Face localization results on multiple-face images: Portraits

A key difference between the general PSG framework and the Pictorial Structures model of [13] is that the PSG framework makes no assumptions concerning the number of symbols of each type in an image, while Pictorial Structures assumes there is exactly one symbol of each type in an image. As such, while the performance of both approaches may be similar when localizing faces in scenes with a single face, performance may be quite different in scenes with a variable number of faces.

To study the ability of the PSG Face Grammar and baseline methods to detect multiple faces in an image, we perform face localization on the Portraits dataset described in Section 9.2.2. We use the same PSG Face Grammar, model parameters, and data model as in the LFW face detection experiments.

Figures 9.4 and 9.5 shows a qualitative localization comparison between the PSG Face Grammar and the baseline methods on the Portraits dataset. We show the top K detection for each symbol after performing non-maximum suppression, where K is the ground truth number of faces in the image. Non-maximum is performed in the same fashion as described in Section 9.2.6.

Table 9.6 compares the area under the precision-recall curves for the PSG Face Grammar and baseline methods. For this evaluation, we use the same non-maxima suppression approach and criterion for true/false positives as for the LFW dataset. In particular, we perform non-maximum suppression for the symbols FACE, LEFT-EYE, RIGHT-EYE, NOSE, MOUTH separately. We first sort the detection probabilities for each symbol, then perform suppression so that no two detections of the same symbol overlap. We consider a detection to be a true positive if it is the highest scoring

Table 9.4: Top K localization results on two examples from the Portraits dataset after non-maximum suppresion. K is set to the ground-truth number of faces in the image for visualization purposes. **Top row**: annotated ground-truth bounding boxes. **Middle-top row**: results of the HOG Filters model. **Middle-bottom row**: results of Pictorial Structures. **Bottom row**: results of the PSG Face Grammar model. The parts are FACE (red), LEFT-EYE (green), RIGHT-EYE (blue), NOSE (cyan), and MOUTH (magenta). Note that in the both examples, Pictorial Structures makes a mistake in localizing the mouth of one of the subjects. However, the PSG Face Grammar model does not make this mistake. This is because of the PSG Face Grammar model's use of "look-alike" symbols, which is a concept that cannot be captured in Pictorial Structures. The HOG Filters model performs poorly, demonstrating the importance of using contextual information in object localization.

Table 9.5: Top K localization results on two examples from the Portraits dataset after non-maximum suppression. **Top row**: annotated ground-truth bounding boxes. K is set to the ground-truth number of faces in the image for visualization purposes. **Middle-top row**: results of the HOG Filters model. **Middle-bottom row**: results of Pictorial Structures. **Bottom row**: results of the PSG Face Grammar model. The parts are FACE (red), LEFT-EYE (green), RIGHT-EYE (blue), NOSE (cyan), and MOUTH (magenta). The example on the right shows a failure mode for all models. The lightning creates a challenging environment due to shadows, the left-most subject's head is significantly rotated, and the pattern on the couch resembles a face. A richer PSG model that includes in-plane rotation as part of the pose space may be able to address the failure modes in the example on the right.

detection with an intersection-over-union ratio of at least 0.5 with the ground truth bounding box. We consider a detection a false positive if it is not the highest scoring detection with an intersection-over-union ratio of at least 0.5 with the ground truth bounding box, thus penalizing multiple detections of the same object.

Model	FACE	LEFT-EYE	RIGHT-EYE	NOSE	MOUTH	**Average**
HOG Filters	0.95	0.50	0.48	0.90	0.32	0.63
Pictorial Structures	0.97	0.78	0.69	0.96	0.73	0.82
PSG Face Grammar	0.97	0.81	0.81	0.96	0.80	0.87

Table 9.6: Area under the Precision-Recall curves for the Portraits dataset. Note that the HOG Filters model performs much worse than the PSG Face Grammar, as was the case in the LFW dataset results. Here, however, the PSG Face Grammar outperforms the Pictorial Structures model. A key difference between the two models is that the Pictorial Structures model assumes that there is one face per image, while the PSG Face Grammar does not. Since the Portraits dataset contains a variable number of faces per image, the one-face assumption made by Pictorial Structures is violated, thus leading to degraded performance.

Unlike the results on the LFW dataset, on the Portraits dataset, the PSG Face Grammar model significantly outperforms the Pictorial Structures model. The key difference between the two models is that the Pictorial Structures model assumes that there is one face per image, while the PSG Face Grammar does not make that assumption. Since the Portraits dataset contains a variable number of faces per image, the one-face assumption made by Pictorial Structures is violated. This causes Pictorial Structures to become unable to select a consistent detection threshold to report positive detections since such a threshold is dependent on the number of faces in the scene. To demonstrate this point, consider the case where there are K identical faces in a scene. Since Pictorial Structures assumes that there is only one face in the scene, each of the K faces receives $\frac{1}{K}$ of the probability mass. If K can vary between images, as is the case here, it is not possible to set a consistently tight detection threshold. The PSG Face Grammar model does not suffer from this consistent threshold issue since it does not make any assumptions concerning the number of faces in the scene.

9.2.8 Face localization without a Face data model

We argue that contextual information plays a key role in object localization. To study the role of contextual information in the task of face localization, we repeat the experiments on the LFW and Portraits dataset using the PSG Face Grammar, but without a data model for the T-FACE symbol. In other words, there are no unary potentials attached to the random variables $\mathbf{X}(\text{T-FACE}, \omega), \omega \in \Omega_{\text{T-FACE}}$. In this setting, the notion of a face is solely defined in terms of its relationship to its parts;

the idea of a face is a purely abstract concept. In this section, we explore the ability of this "Faceless Grammar" to perform face localization despite not having a data model for faces.

We compare the area under the precision-recall curves on the Portraits dataset in Table 9.7. Although the Faceless Grammar model does worse on this measure, the face localization still performs reasonably well, achieving an area under the precision-recall curve of 0.93.This demonstrates that it is possible to perform face localization without an explicit data model for faces.

Model	FACE	LEFT-EYE	RIGHT-EYE	NOSE	MOUTH	**Average**
PSG Face Grammar	0.97	0.81	0.81	0.96	0.80	0.87
Faceless Grammar	0.93	0.78	0.80	0.95	0.76	0.84

Table 9.7: Area under the precision-recall curves on the Portraits dataset. Note that the Faceless Grammar performs worse than the PSG Face Grammar. However, the Faceless Grammar performs reasonably well considering it is attempting to localize an object for which it has no image evidence.

9.3 Binary image segmentation

To study binary image segmentation, we use the Swedish Leaf Dataset described in [53]. We use only the Rowan leaves class in our experiments because of their varied and complex shapes. The Rowan leaf class contains 75 examples. Following the experimental setup described in [16], we use 50 examples for training and the rest for testing. Each example contains exactly one Rowan leaf and is encoded as a binary map $\mathbf{B}$.

From a binary map $\mathbf{B}$ we generate a noisy real-valued image $\mathbf{D}$ by sampling each pixel $\mathbf{D}(i,j)$ independently from a Normal distribution whose mean depends on the value of $\mathbf{B}(i,j)$. Formally,

$$\mathbf{D}(i,j) \sim \mathcal{N}(\mu_{\mathbf{B}(i,j)}, \sigma). \tag{9.5}$$

For the experiments, we used $\mu_0 = 150$, $\mu_1 = 100$, $\sigma = 100$.

Note that the data model in 9.5 is the same as for contour detection, but in these experiments we use a higher value of σ.

Figure 9.9 shows examples of binary maps and the noisy real-valued images.

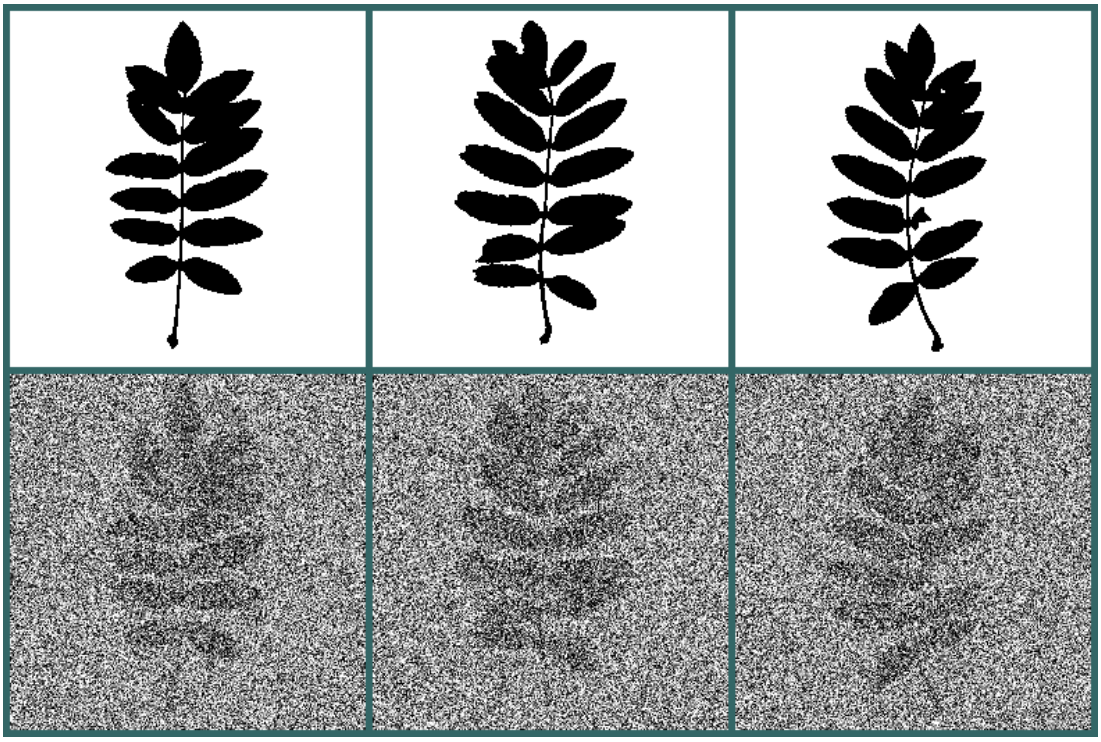


Figure 9.9: Examples of the data used for the binary image segmentation experiments. Top row: examples of $\mathbf{B}$. Foreground pixels are shown in black. Bottom row: corresponding examples of $\mathbf{D}$.

9.3.1 The PSG binary image segmentation models

In the experiments below, we study two PSGs for binary image segmentation. Both PSG grammars are cyclic, and we constrain the parameters θ so that $\theta_{(\omega,r,i,z)} = \theta_{(\omega',r,i,z')}$ whenever $\omega - z = \omega' - z'$. *I.e.*, the conditional pose distributions are constrained to be shift-invariant. We also constrain the PSGs so that no brick may generate itself in one production rule.

The first PSG we study for binary image segmentation is similar to the one described in Grammar 3, but with different model parameters. The model parameters are learned by using the ground-truth contour maps $\mathbf{B}$ as observations for the symbol FG then using the approximate EM algorithm described in Chapter 8. Note that this is a partially-supervised setting since the supervision labels specify only the presence/absence of a subset of bricks.

The learned grammar is shown in Grammar 8. Note that this grammar is cyclic, and so the factor graph construction described in Definition 16 leads to a different (but related) distribution over scenes. We will refer to this grammar as the "Simple Segmentation Grammar".

For readability, we denote $\theta_{(\omega,1,1)}$ by θ_{SEED} since $|\Omega_{\mathsf{SEED}}| = 1$, and $\theta_{(\omega,2,1)}$ by $\theta_{(\omega,\mathsf{FG})}$ $\forall \omega \in \Omega_{\mathsf{FG}}$.

Grammar 8 *The "Simple Segmentation Grammar" for 2-D binary image segmentation in an $N \times M$ scene with model parameters learned in a partially unsupervised setting:*

$\Sigma = \{\mathsf{SEED}, \mathsf{FG}\}$.
$\Omega_{\mathsf{SEED}} = [1]$.
$\Omega_{\mathsf{FG}} = [N] \times [M]$.
Rules:

$$1.0, \quad (\mathsf{SEED}, 1) \quad \rightarrow \quad (\mathsf{FG}, \text{Categorical}(\cdot \mid \theta_{\mathsf{SEED}}))$$
$$1.0, \quad (\mathsf{FG}, \omega) \quad \rightarrow \quad (\mathsf{FG}, \text{IndBern}(\cdot \mid \theta_{(\omega,\mathsf{FG})}))$$

$\epsilon_{\mathsf{SEED}} = 1$,
$\epsilon_{\mathsf{FG}} = 0$.

Recall that for inference, we convert a PSG to a factor graph and run LBP. To incorporate the data model given in 9.5 into the factor graph representation, we attach unary potentials to the set of variables nodes $\{\mathbf{X}(\mathsf{FG}, (i,j)) \mid (i,j) \in \Omega_{\mathsf{FG}}\}$. In particular, for variable node $\mathbf{X}(\mathsf{FG}, (i,j))$ we attach a unary potential $f^0_{(\mathsf{FG},(i,j))}(\mathbf{X}(\mathsf{FG}, (i,j)) = x, \mathbf{D}) = \mathcal{N}(\mathbf{D}(i,j); \mu_x, \sigma)$.

One weakness of the Simple Segmentation Grammar is that it is incapable of modeling structured variations in local shape. Compare the shapes of the foreground in 3×3 regions around a pixel on stem of the leaf, and 3×3 regions around a pixel on a lobe (components jutting off the stem). Locally, the shape of the foreground is very different around these two areas. Around a pixel located on the stem of the leaf, the foreground tends to extend above and below the pixel. Around a pixel located

in the middle of a lobe, the foreground tends to extend in all directions. To model these structured variations in the local shape of the foreground, we use a PSG that has the capacity to model different local foreground shapes.

Grammar 9 describes a binary image segmentation model with a capacity to model different local foreground shapes [3]. Note that this grammar is cyclic, and so the factor graph construction described in Definition 16 leads to a different (but related) distribution over scenes Each symbol S_j, $1 \leq j \leq 5$ models a different local shape. Each brick (S_j, ω), $\omega \in \Omega_{\mathsf{S}_j}$ can generate a brick (FG, ω) and other S_j bricks in an 8-neighbourhood around it, or it can generate a brick (S_k, ω), $k \neq j$ to model a change of local shape. The single SEED brick chooses an S_1 brick to start the generative process. The model parameters are learned in the same fashion as for Grammar 8.

Note that apriori, the set of symbols symbols $\{\mathsf{S}_j \mid 2 \leq j \leq 5\}$ have no semantic meaning and are exchangeable in the model. To break symmetries in the model, we randomly initialize the model parameters relating to the set of symbols symbols $\{\mathsf{S}_j \mid 2 \leq j \leq 5\}$. We will refer to this model as the "5-component Segmentation Grammar".

For both binary segmentation models described above, to incorporate the data model given in Eqn. 9.5 into the factor graph representation, we attach unary potentials to the set of variables nodes $\{\mathbf{X}(\mathsf{FG}, (i,j)) \mid (i,j) \in \Omega_{\mathsf{FG}}\}$. For a variable node $\mathbf{X}(\mathsf{FG}, (i,j))$ we attach a unary potential $f^0_{(\mathsf{FG},(i,j))}(\mathbf{X}(\mathsf{FG}, (i,j)) = x, \mathbf{D}) = \mathcal{N}(\mathbf{D}(i,j); \mu_x, \sigma)$. In this case, the resulting factor graph represents the conditional distribution $p(S \mid \mathbf{D})$.

[3] For readability, we denote $\theta_{(\omega,1,1)}$ by θ_{SEED} since $|\Omega_{\mathsf{SEED}}| = 1$. We also denote $\theta_{(\omega,2,1)}$ by $\theta_{(\omega,\mathsf{S}_1)}$, $\theta_{(\omega,7,1)}$ by $\theta_{(\omega,\mathsf{S}_2)}$, $\theta_{(\omega,12,1)}$ by $\theta_{(\omega,\mathsf{S}_3)}$, $\theta_{(\omega,17,1)}$ by $\theta_{(\omega,\mathsf{S}_4)}$, and $\theta_{(\omega,22,1)}$ by $\theta_{(\omega,\mathsf{S}_5)}$.

Grammar 9 *The "5-component Segmentation Grammar" for 2-D binary image segmentation in an $N \times M$ scene with model parameters learned in a partially unsupervised setting:*

$\Sigma = \{\mathsf{SEED}, \mathsf{FG}, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4, \mathsf{S}_5\}$.

$\Omega_{\mathsf{SEED}} = [1]$.

$\Omega_{\mathsf{FG}} = [N] \times [M]$.

$\Omega_{\mathsf{S}_j} = [N] \times [M]$, $1 \leq j \leq 5$.

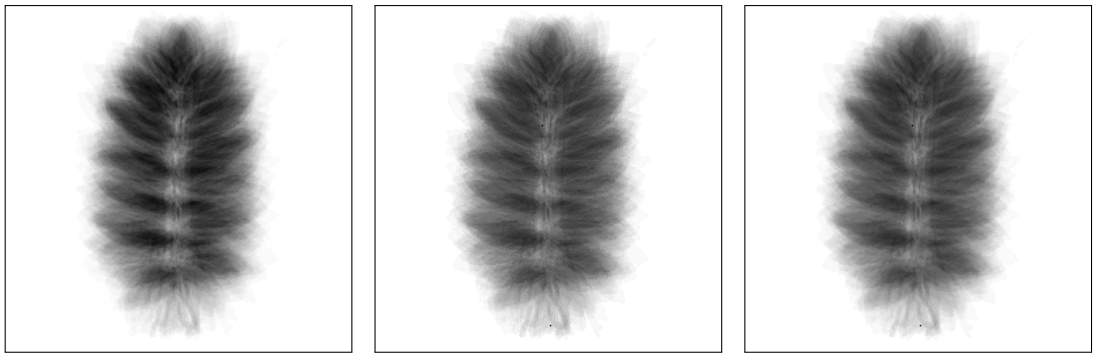
Rules:

$$
\begin{array}{lrcl}
1.000, & (\mathsf{SEED}, 1) & \to & (\mathsf{S}_1, \mathrm{Categorical}(\cdot \mid \theta_{\mathsf{SEED}})) \\
0.841, & (\mathsf{S}_1, \omega) & \to & (\mathsf{S}_1, \mathrm{IndBern}(\cdot \mid \theta_{(\omega,\mathsf{S}_1)})), (\mathsf{FG}, \delta(\omega)) \\
0.042, & (\mathsf{S}_1, \omega) & \to & (\mathsf{S}_2, \delta(\omega)) \\
0.040, & (\mathsf{S}_1, \omega) & \to & (\mathsf{S}_3, \delta(\omega)) \\
0.037, & (\mathsf{S}_1, \omega) & \to & (\mathsf{S}_4, \delta(\omega)) \\
0.040, & (\mathsf{S}_1, \omega) & \to & (\mathsf{S}_5, \delta(\omega)) \\
0.824, & (\mathsf{S}_2, \omega) & \to & (\mathsf{S}_2, \mathrm{IndBern}(\cdot \mid \theta_{(\omega,\mathsf{S}_2)})), (\mathsf{FG}, \delta(\omega)) \\
0.043, & (\mathsf{S}_2, \omega) & \to & (\mathsf{S}_1, \delta(\omega)) \\
0.045, & (\mathsf{S}_2, \omega) & \to & (\mathsf{S}_3, \delta(\omega)) \\
0.042, & (\mathsf{S}_2, \omega) & \to & (\mathsf{S}_4, \delta(\omega)) \\
0.045, & (\mathsf{S}_2, \omega) & \to & (\mathsf{S}_5, \delta(\omega)) \\
0.842, & (\mathsf{S}_3, \omega) & \to & (\mathsf{S}_3, \mathrm{IndBern}(\cdot \mid \theta_{(\omega,\mathsf{S}_3)})), (\mathsf{FG}, \delta(\omega)) \\
0.038, & (\mathsf{S}_3, \omega) & \to & (\mathsf{S}_1, \delta(\omega)) \\
0.043, & (\mathsf{S}_3, \omega) & \to & (\mathsf{S}_2, \delta(\omega)) \\
0.037, & (\mathsf{S}_3, \omega) & \to & (\mathsf{S}_4, \delta(\omega)) \\
0.040, & (\mathsf{S}_3, \omega) & \to & (\mathsf{S}_5, \delta(\omega)) \\
0.858, & (\mathsf{S}_4, \omega) & \to & (\mathsf{S}_4, \mathrm{IndBern}(\cdot \mid \theta_{(\omega,\mathsf{S}_4)})), (\mathsf{FG}, \delta(\omega)) \\
0.033, & (\mathsf{S}_4, \omega) & \to & (\mathsf{S}_1, \delta(\omega)) \\
0.038, & (\mathsf{S}_4, \omega) & \to & (\mathsf{S}_2, \delta(\omega)) \\
0.036, & (\mathsf{S}_4, \omega) & \to & (\mathsf{S}_3, \delta(\omega)) \\
0.035, & (\mathsf{S}_4, \omega) & \to & (\mathsf{S}_5, \delta(\omega)) \\
0.844, & (\mathsf{S}_5, \omega) & \to & (\mathsf{S}_5, \mathrm{IndBern}(\cdot \mid \theta_{(\omega,\mathsf{S}_5)})), (\mathsf{FG}, \delta(\omega)) \\
0.037, & (\mathsf{S}_5, \omega) & \to & (\mathsf{S}_1, \delta(\omega)) \\
0.042, & (\mathsf{S}_5, \omega) & \to & (\mathsf{S}_2, \delta(\omega)) \\
0.040, & (\mathsf{S}_5, \omega) & \to & (\mathsf{S}_3, \delta(\omega)) \\
0.037, & (\mathsf{S}_5, \omega) & \to & (\mathsf{S}_4, \delta(\omega))
\end{array}
$$

$\epsilon_{\mathsf{SEED}} = 1$, $\epsilon_{\mathsf{FG}} = 0$, $\epsilon_{\mathsf{S}_j} = 0$, $1 \leq j \leq 5$.

To gain insight into the PSGs learned, we study the model parameters θ learned by both models. First, consider the parameters θ_{SEED}. For both models, this parameter encodes the distribution over

the FG brick that will start the generative process of creating the foreground of the leaf. The models must consider two issues in setting the parameters θ_{SEED}. Let $\bar{\mathbf{B}}$ represent the mean of the binary maps across training examples. Intuitively, one might expect θ_{SEED} to resemble $\bar{\mathbf{B}}$ since $\bar{\mathbf{B}}(i,j)$ is the probability that the brick $(\mathsf{FG},(i,j))$ will be present in a scene in the training set. On the other hand, the model might prefer to start the generative process at a more central FG brick in the scene. Consider the probability that the generative process will cause brick $(\mathsf{FG},(i,j))$ to be present in the scene when the generative process starts at brick $(\mathsf{FG},(i_0,j_0))$. This probability decreases as the distance between (i,j) and (i_0,j_0) increases, and so the model may favour more central locations to the start the generative process. Figure 9.10 shows a visualization of $\bar{\mathbf{B}}$ and the parameters θ_{SEED} learned for the Simple Segmentation Grammar and the 5-component Segmentation Grammar. Note that $\log(\theta_{\mathsf{SEED}})$ for both models resembles $\bar{\mathbf{B}}$.



(a) Visualization of $\bar{\mathbf{B}}$. Darker pixels correspond to a higher probability of that pixel being labeled foreground in the training images.

(b) Visualization of $\log(\theta_{\mathsf{SEED}})$ learned for the Simple Segmentation Grammar.

(c) Visualization of $\log(\theta_{\mathsf{SEED}})$ learned for the 5-component Segmentation Grammar.

Figure 9.10: Visualization of $\bar{\mathbf{B}}$ and the parameters θ_{SEED} learned for the Simple Segmentation Grammar and the 5-component Segmentation Grammar. For panels (b) and (c), darker pixels correspond to a higher value of θ_{SEED} for that location. For visualization, the parameters θ_{SEED} are shown in the log domain and linearly scaled to be between 0 and 1.

In Figure 9.11, we show a visualization for the parameters $\theta_{(\omega,\mathsf{FG})}$ learned for the Simple Segmentation Grammar and $\theta_{(\omega,\mathsf{S}_j)}$, $1 \leq j \leq 5$, learned for the 5-component Segmentation Grammar. As shown in the figure, the parameter $\theta_{(\omega,\mathsf{FG})}$ learned for the Simple Segmentation Grammar tends to favour expanding a FG brick in all directions, while the parameters $\theta_{(\omega,\mathsf{S}_j)}$, $1 \leq j \leq 5$, encode variations in the shape of the local foreground.

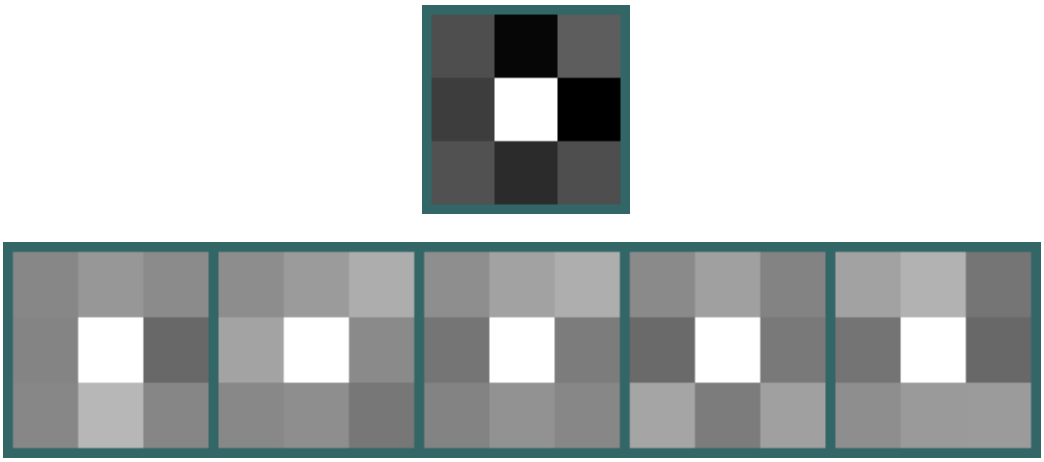


Figure 9.11: Visualization of the learned parameters $\theta_{(\omega,\mathsf{FG})}$ and $\theta_{(\omega,\mathsf{S}_j)}$, $1 \leq j \leq 5$, for the Simple Segmentation Grammar and 5-component Segmentation Grammar, respectively. Darker pixels indicate a higher value of θ. Recall that we constrain the parameters $\theta_{(\omega,\mathsf{FG})}$ and $\theta_{(\omega,\mathsf{S}_j)}$, $1 \leq j \leq 5$, to be shift-invariant and so that a brick may not generate itself in one production. Given a brick with pose ω, the visualizations show the probability that the brick will generate each of its 8-neighbours where the centre pixel corresponds to the brick with pose ω. The visualizations have been scaled consistently for comparison. **Top row**: Visualization of the parameters $\theta_{(\omega,\mathsf{FG})}$ learned for the Simple Segmentation Grammar. **Bottom row**: Left to right: a visualization of the parameters $\theta_{(\omega,\mathsf{S}_j)}$ for $j = [1, \cdots, 5]$ learned for the 5-component Segmentation Grammar.

9.3.2 Qualitative binary image segmentation results

In Figure 9.12 we show binary image segmentation results on examples from the Swedish Leaf Dataset described in [53]. We show the approximate marginal probability that each FG brick is present in the scene, $\hat{p}(\mathbf{X}(\mathsf{FG}, (i, j)) = 1 \mid \mathbf{D})$, as computed by LBP. On 256×256 test images, running LBP to convergence took less than 260 and 1900 seconds for the Simple Segmentation Grammar and the 5-component Segmentation Grammar, respectively.

As shown in Figure 9.12, the Simple Segmentation Grammar creates "blob-like" foreground segmentations and does a poor job of differentiating the lobes of the leaves. In contrast, the 5-component Segmentation Grammar can more faithfully capture the shape of the lobes of the leaves, although it does so crudely. The ability to more finely capture the shape of the lobes can be attributed to the explicit modeling of local variations in the shape of the foreground. However, the 5-component Segmentation Grammar is more susceptible to picking up speckled noise, as shown in the figure.

Note that although both grammar models constrain scenes to have a single non-empty connected foreground component, the results of inference indicate that this constraint is being violated. As discussed in Section 6.3, the factor graph construction described in Chapter 4 does not match the distribution over scenes induced by a cyclic grammar. Since both the Simple Segmentation Grammar

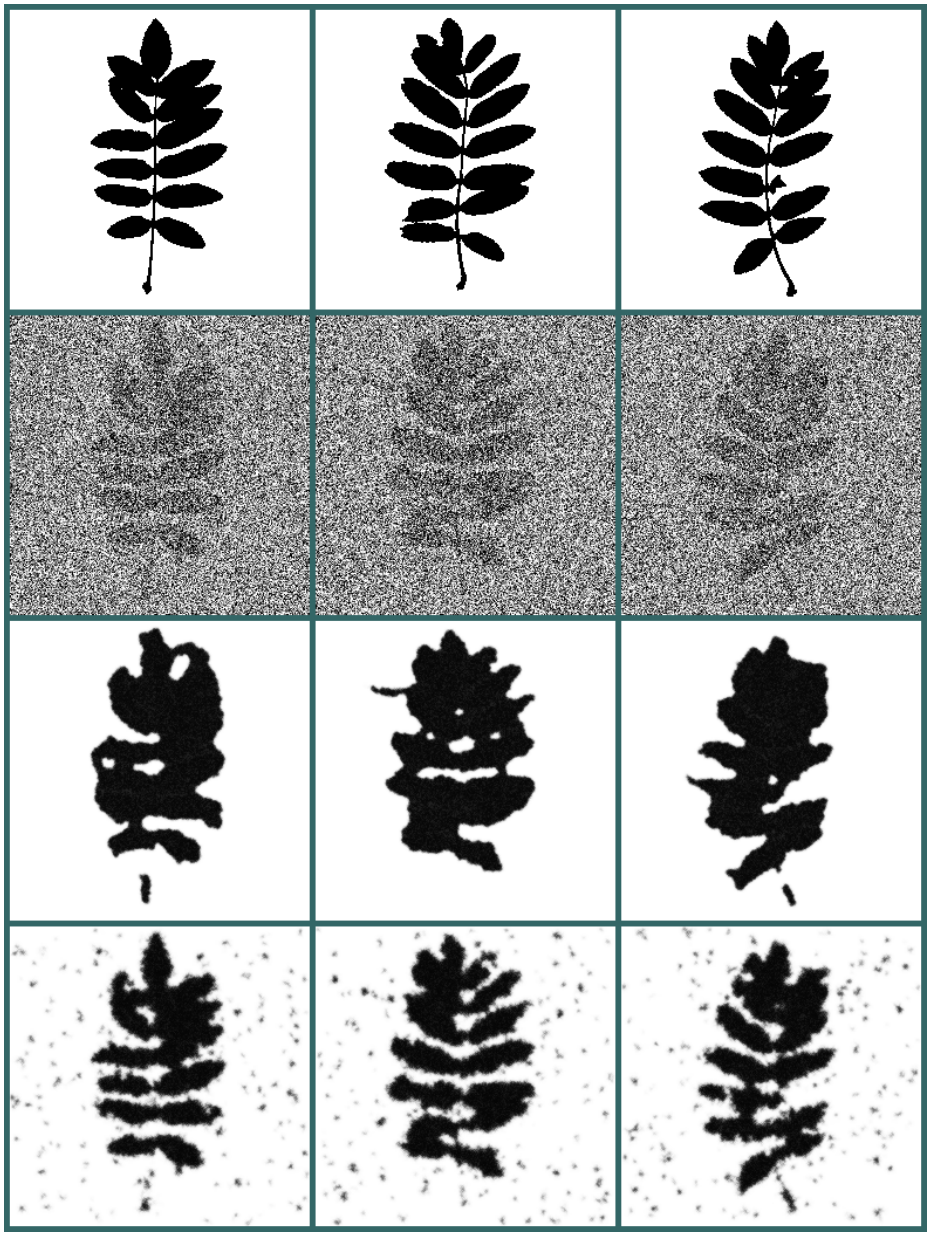


Figure 9.12: Binary image segmentation results on three examples from the Swedish Leaf test set. In the bottom two rows, each pixel represents an FG brick at that location. The gray-scale values show the approximate marginal probabilities $\hat{p}(\mathbf{X}(\mathsf{FG}, (i, j)) = 1 \mid \mathbf{D})$ computed by LBP. Darker pixels indicate a higher approximate marginal probability. **First row**: Ground-truth contour maps $\mathbf{B}$. **Second row**: Noisy observations, $\mathbf{D}$. **Third row**: Results of the Simple Segmentation Grammar. **Fourth row**: Results of the 5-component Segmentation Grammar.

and the 5-component Segmentation Grammar are cyclic, the factor graphs on which LBP are run do not faithfully represent the PSGs they are derived from. Moreover, since LBP is an approximate inference scheme, there is no guarantee that LBP will produce marginals consistent with the single non-empty connected foreground constraint even if the factor graphs were faithful representations of the PSGs they are derived from. These two issues result in the approximate marginals produced by LBP being inconsistent with the constraint that the foreground be a single non-empty connected component. Nevertheless, as Figure 9.12 shows, the PSG framework still produces reasonable binary image segmentations despite these flaws.

9.3.3 Quantitative binary image segmentation results

In this subsection, we perform a quantitative comparison between the Simple Segmentation Grammar, 5-component Segmentation Grammar, and the baseline methods. Our chief comparison is to the work FOP models of [16]. To demonstrate the importance of context for binary image segmentation detection, we also compare against a PSG where $\Sigma = \{\mathsf{FG}\}$, the FG bricks are allowed to self-root, and the PSG's only rule is $\mathsf{FG} \rightarrow \emptyset$; *i.e.*, all bricks are independent. We will refer to this model as the "No-Context PSG".

As in the task of contour detection, for the PSG models, we compare performance using the area under the precision-recall curve (AUC) by thresholding $\hat{p}(\mathbf{X}(\mathsf{FG}, (i,j)) = 1 \mid \mathbf{D})$, $(i,j) \in \Omega_{\mathsf{FG}}$, over a range of values. For the work of [16], the authors have shared their experimental results and we compute AUC in a similar fashion as for the PSG models.

Table 9.8 compares the AUC of the Simple Segmentation Grammar, 5-component Segmentation Grammar, and No-context PSG as well as the 1-level and 4-level FOP models of [16]. Figure 9.13 compares the precision-recall curves of these methods. Note that the No-context PSG performs the worst out of all methods tested, demonstrating that some notion of context is crucial for producing high-quality binary image segmentations on this dataset. Also, the 5-component Segmentation Grammar significantly outperforms the Simple Segmentation Grammar, indicating the importance of modeling the variation in local foreground shape. Although the PSG segmentation models are outperformed by both FOP models, both the Simple Segmenetation Grammar and the 5-component Segmentation Grammar give reasonable results despite the general-purpose nature of the PSG framework. Also, the gap in performance between the best performing PSG model and the 1-level FOP model is relatively small.

We believe it is possible to define more effective models for binary image segmentation. As illustrated in Section 6.3, the use of cyclic grammars can sometimes be problematic in the PSG framework. A different model for binary image segmentation could perhaps be specified as an acyclic

PSG, which may prove to be more effective than cyclic PSGs. For example, one could design a hierarchical acyclic PSG that can express long-range dependencies between FG bricks. The design of more sophisticated image segmentation models in the PSG framework is a future research goal.

Model	AUC
No-context PSG	0.310
Simple Segmentation Grammar	0.911
5-component Segmentation Grammar	0.956
1-level FOP, [16]	0.967
4-level FOP, [16]	0.976

Table 9.8: Comparison of AUC for several different models. See text for discussion.

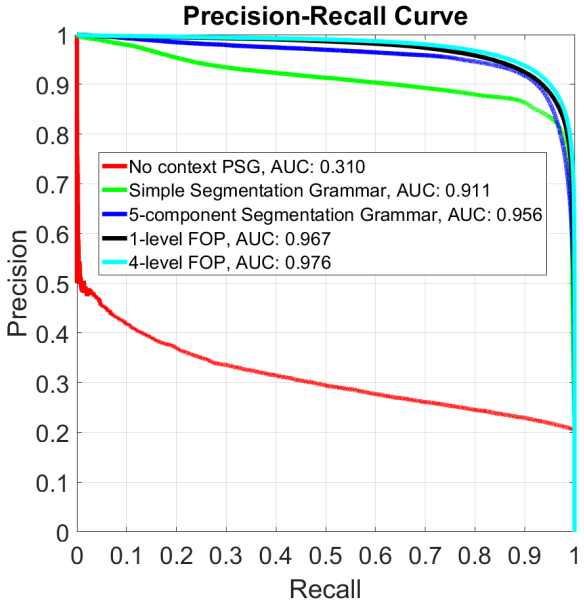


Figure 9.13: Precision-recall curves for several PSGs and the 1-level and 4-level FOP models. The AUC for each model is shown in the legend. Note the poor performance of the No-context PSG, demonstrating the importance of contextual information in binary image segmentation. Also note that the 5-component Segmentation Grammar significantly outperforms the Simple Segmentation Grammar, illustrating the importance of modeling the variation in the local shape of the foreground. Lastly, the best performing PSG model, the 5-component Segmentation Grammar, is competitive with the 1-level FOP model.

Chapter 10

Grammar Transformations

As discussed in Chapter 1, we seek efficient approximate inference algorithms as a scene understanding framework in practice may be deployed in time-sensitive scenarios. Recall that the run time of LBP in the factor graph representation of a PSG is linear in the number of edges of the factor graph. Unfortunately, the PSG factor graph may contain upwards of millions of edges for a moderately sized model and so LBP inference may be slow. For example, in the contour detection experiments described in Section 9.1, the factor graph had roughly 50 million edges and running LBP to convergence on a modern machine took approximately 1.5 hours. If one wishes to apply the PSG framework to more complex models than the ones expressed in this thesis and on larger scenes, then it is clear that the practical issues of run time (and memory) must be dealt with. In this chapter, we discuss strategies for reducing the number of edges in the factor graph representation of a PSG .

The main idea presented in this chapter is that a $\mathrm{Categorical}$ conditional pose distribution in the PSG can be represented by a combination of distributions. In the PSG framework, as we will see shortly, the "cost" of representing a distribution is the support size of the distribution. So, we seek strategies and approximations of distributions that reduce their support size.

We consider two special cases. First, we approximate a general N-D $\mathrm{Categorical}$ conditional pose distribution by a product of N one-dimensional $\mathrm{Categorical}$ conditional pose distributions. For example, suppose we have a distribution over two variables, $p(X, Y)$. We seek to represent this distribution by a factorized distribution $p(X)p(Y)$. Here, the total support size of $p(X)$ and $p(Y)$ can be significantly less than the support size of $p(X, Y)$.

Second, consider a $\mathrm{Uniform}$ distribution over K elements. We approximate this distribution as a combination of $\mathrm{Uniform}$ distributions, each one over a set of fewer elements. For example, consider a $\mathrm{Uniform}$ distribution over the set $\{0, 1, \ldots, 99\}$. One could sample from this distribution by the process of first drawing X uniformly from the set $\{0, 10, \ldots, 90\}$, Y uniformly from the set

$\{0, \ldots, 9\}$, then declaring $Z = X + Y$ as the sample drawn. The distribution over Z is uniform on the set $\{0, \ldots, 99\}$, but here, Z is represented as a combination of two $\mathrm{Uniform}$ distributions over 10 elements. We make these ideas more concrete in the rest of this chapter and show how they can be applied in the PSG framework.

10.1 Counting factor graph edges

In order to reduce the number of edges in the factor graph representation of a PSG, we first analyze how the number of edges depends on the parameters of the PSG . To simplify the analysis below, we assume that $|\Gamma_{(\omega,r,i)}| = |\Gamma_{(\omega',r,i)}| \; \forall \omega, \omega' \in \Omega_{A_{(r,i)}}$ (*i.e.*, $|\Gamma_{(\omega,r,i)}|$ is a constant with respect to ω). We will use the notation $|\Gamma_{(\omega,r,i)}| = S_{(r,i)} \; \forall \omega \in \Omega_{A_{(r,i)}}$.

Recall the factor graph representation in Figure 4.2 for a single brick (A, ω), $A \in \Sigma$, $\omega \in \Omega_A$. We can read off the number of edges connected to (degree of) each factor for a single brick. Table 10.1 summarizes the results as a function of the parameters of the PSG .

Factor node type	**Degree of factor**
$f^1_{(A,\omega)}$	$1 + \lvert \mathrm{par}(\mathbf{X}(A,\omega)) \rvert$
$f^2_{(A,\omega)}$	$1 + \lvert \mathcal{R}_A \rvert$
$f^3_{(A,\omega,\cdot,\cdot)}$	$\lvert \mathcal{R}_A \rvert + \sum_{r \in \mathcal{R}_A} \sum_{i=1}^{n_r} S_{(r,i)}$

Table 10.1: Number of edges connected to each type of factor for a single brick (A, ω).

The total number of edges associated with each type of factor can be computed by summing over all bricks in the factor graph. Table 10.2 summarizes the results.

Factor node type	**Number of edges connected to factors of this type**
$f^1_{(\cdot,\cdot)}$	$\sum_{A \in \Sigma} \lvert \Omega_A \rvert + \sum_{A \in \Sigma} \sum_{\omega \in \Omega_A} \lvert \mathrm{par}(\mathbf{X}(A,\omega)) \rvert$ $= \sum_{A \in \Sigma} \lvert \Omega_A \rvert + \sum_{A \in \Sigma} \lvert \Omega_A \rvert \sum_{r \in \mathcal{R}_A} \sum_{i=1}^{n_r} S_{(r,i)}$
$f^2_{(\cdot,\cdot)}$	$\sum_{A \in \Sigma} \lvert \Omega_A \rvert + \sum_{A \in \Sigma} \lvert \Omega_A \rvert \lvert \mathcal{R}_A \rvert$
$f^3_{(\cdot,\cdot)}$	$\sum_{A \in \Sigma} \lvert \Omega_A \rvert \lvert \mathcal{R}_A \rvert + \sum_{A \in \Sigma} \lvert \Omega_A \rvert \sum_{r \in \mathcal{R}_A} \sum_{i=1}^{n_r} S_{(r,i)}$

Table 10.2: Number of edges connected to each type of factor over all bricks in the PSG factor graph.

From Table 10.2, we can express the total number of edges in the factor graph as

$$\text{number of edges in factor graph} = 2 \sum_{A \in \Sigma} |\Omega_A| (1 + |\mathcal{R}_A| + \sum_{r \in \mathcal{R}_A} \sum_{i=1}^{n_r} S_{(r,i)}). \tag{10.1}$$

10.2 Reducing the number of factor graph edges

We first define the Uniform distribution.

Definition 42 *Let W be a set of binary random variables indexed by a set Υ. Also, let $T \subseteq \Upsilon$ be a set. Recall that we define the set $I(W) = \{k \mid W_k = 1, k \in \Upsilon\}$. We define the* Uniform *distribution as*

$$\text{Uniform}(W; T) = \begin{cases} \frac{1}{|T|} & , \sum_{k \in \Upsilon} W_k = 1, I(W) \subseteq T \\ 0 & , \textit{otherwise} \end{cases} \tag{10.2}$$

For brevity, for the rest of this chapter we drop the argument W from the Uniform distribution, and will denote it as $\text{Uniform}(T)$.

Examining Eqn. 10.1, to reduce the number of edges in the factor graph representation of a PSG, we can reduce the size of the pose spaces, the number of productions rules, and the size of the support of the conditional pose distributions.

For some PSG models, the size of the pose spaces can be reduced without greatly affecting modeling power. For example, suppose the pose space for a symbol of a grammar was all pixel locations in a scene. Rather than consider all pixel locations, one could consider a coarse grid of pixel locations, *e.g.*, every other pixel. If the pose space of a symbol includes orientation, as in Grammar 1, one could reduce the number of orientations considered.

Production rules often model compositions between objects. For example, a compositional rule may model that a FACE is comprised of a LEFT-EYE, RIGHT-EYE, NOSE, and a MOUTH, as in Grammar 2. So, it may not be possible to reduce the number of production rules without drastically changing the model.

Conditional pose distributions represent geometric relationships between objects. For example, such a distribution may encode the fact that the NOSE of a FACE is located somewhere in the middle of the FACE within a region of uncertainty. As can be seen from Eqn. 10.1, the number of edges in the factor graph grows linearly with the total size of the support of the conditional pose distributions. In the next two sections, we focus on strategies for representing and approximating conditional pose distributions as combinations of distributions. We consider two special cases. In Section 10.3, we consider approximating a general N-D Categorical distribution by a product of N one-dimensional distributions. In Section 10.4, we consider representing a Uniform distribution as a combination of Categorical distributions. These techniques will allow us to reduce the number of edges in the factor graph via reducing the term $\sum_{r \in \mathcal{R}_A} \sum_{i=1}^{n_r} S_{(r,i)}$ in Eqn. 10.1.

10.3 Approximating an N-D distribution by a product of N 1-D distributions

First, recall that we use the notation $[M]$ to indicate the set of points $\{0, \ldots, M-1\}$. Let $X = \{x_1, \ldots, x_N\}$ with $x_i \in [M_i]$. Let $p(X)$ be an N-D distribution. Our goal is to approximate $p(X)$ by a product of N one-dimensional distributions $\prod_{i=1}^N p_i(x_i)$. Note that the representation of $p(X)$ by $\prod_{i=1}^N p_i(x_i)$ is exact only when the x_i are independent. Also, while $\|p(X)\|_0 = \prod_{i=1}^N (M_i - 1)$, $\sum_{i=1}^N \|p_i(x_i)\|_0 = \sum_{i=1}^N (M_i - 1)$, and so approximating $p(X)$ by $\prod_{i=1}^N p_i(x_i)$ can lead to a significantly smaller total support size.

To measure the quality of approximation of $p(X)$ by $\prod_{i=1}^N p_i(x_i)$, we use the Kullback-Leibler (KL) divergence, defined below.

Definition 43 *The Kullback-Leibler (KL) divergence between discrete probability distributions r and s is defined to be*

$$D_{KL}(r||s) = \sum_X r(X) \log(\frac{r(X)}{s(X)}) \tag{10.3}$$

where the summation is the over the union of the supports of r and s.

Proposition 44 *Let $X = \{x_1, \ldots, x_N\}$ with $x_i \in [M_i]$. Let $p(X)$ be an N-D distribution that we seek to approximate by $\prod_{i=1}^N p_i(x_i)$. If the quality of approximation is measured as $D_{KL}(p(X)||\prod_{i=1}^N p_i(x_i))$ and we seek to solve the optimization problem*

$$p_1^*(x_1), \ldots, p_N^*(x_N) = \underset{p_1(x_1),\ldots,p_N(x_N)}{\arg\min} D_{KL}(p(X)||\prod_{i=1}^N p_i(x_i)) \tag{10.4}$$

then

$$p_i^*(x_i) = p(x_i), \quad 1 \leq i \leq N \tag{10.5}$$

Proof of Proposition 44

$$D_{KL}(p(X)||\prod_{i=1}^{N} p_i(x_i)) = \sum_X p(X)\log\Big(\frac{p(X)}{\prod_{i=1}^{N} p_i(x_i)}\Big) \tag{10.6}$$

$$= \sum_X p(X)\log p(X) - \sum_X \sum_{i=1}^{N} p(X)\log(p_i(x_i)) \tag{10.7}$$

$$= \sum_X p(X)\log p(X) - \sum_{i=1}^{N} \sum_{x_i} p(x_i)\log(p_i(x_i)) \tag{10.8}$$

Since the $p_i(x_i)$ are probability distributions, we have the constraint that they must sum to 1. Hence, solving Eqn. 10.4 is a constrained optimization problem. We use the method of Lagrange multipliers to enforce the constraint that $\sum_{x_i} p_i(x_i) = 1$. We formulate the Lagrange function $\mathcal{L}(p(X), \prod_{i=1}^{N} p_i(x_i))$:

$$\mathcal{L}(p(X), \prod_{i=1}^{N} p_i(x_i)) = D_{KL}(p(X)||\prod_{i=1}^{N} p_i(x_i)) - \lambda_i(\sum_{x_i} p_i(x_i) - 1). \tag{10.9}$$

Now, taking the partial derivative of the Lagrangian with respect to $p_i(x_i)$,

$$\frac{\mathcal{L}(p(X), \prod_{i=1}^{N} p_i(x_i))}{\partial p_i(x_i)} = \frac{\partial D_{KL}(p(X)||\prod_{i=1}^{N} p_i(x_i))}{\partial p_i(x_i)} - \lambda_i \tag{10.10}$$

$$= \frac{p(x_i)}{p_i(x_i)} - \lambda_i. \tag{10.11}$$

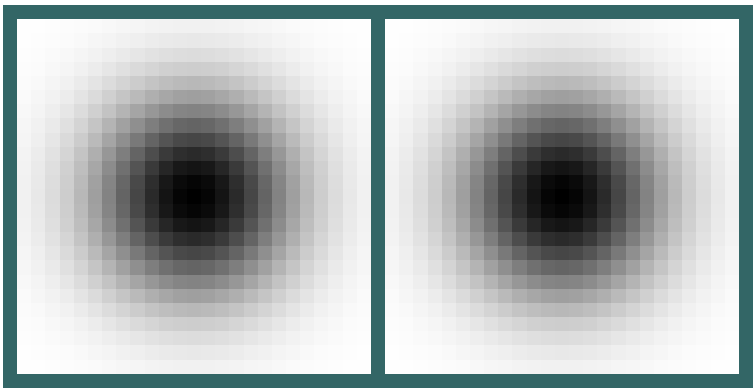
Setting $\frac{\mathcal{L}(p(X), \prod_{i=1}^{N} p_i(x_i))}{\partial p_i(x_i)} = 0$ and solving yields

$$p_i^*(x_i) = p(x_i), \quad 1 \leq i \leq N. \tag{10.12}$$

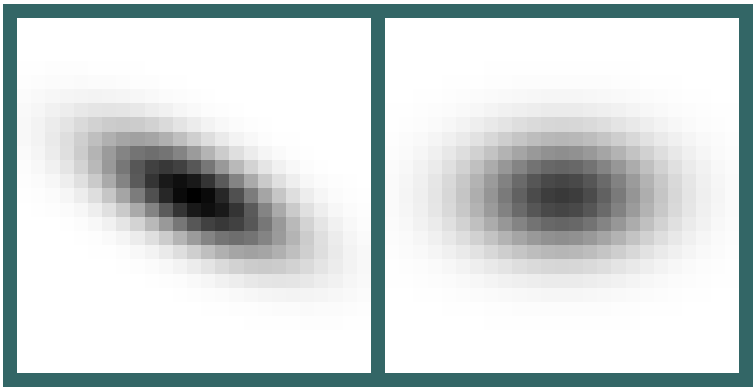
■

Next, we give some examples of using Proposition 44 to approximate a 2-D distribution as a product of two 1-D distributions. Figure 10.1 shows examples of this approximation.

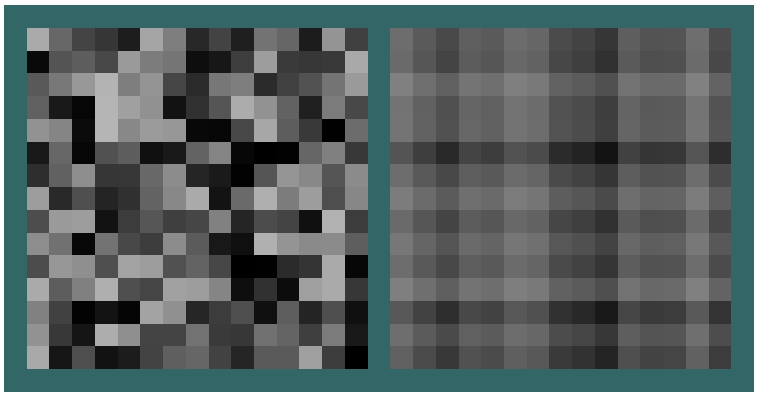
Note that if the distribution $p(x_1, x_2)$ can be factorized, then $D_{KL}(p(x_1, x_2)||p_1^*(x_1)p_2^*(x_2)) = 0$ and $p(x_1, x_2) = p_1^*(x_1)p_2^*(x_2)$. Figure 10.1(a) shows an example where $p(x_1, x_2)$ can be factorized, and indeed, the solution given by Eqn. 10.5 is the factorization. Figures 10.1(b) and 10.1(c) show examples where $p(x_1, x_2)$ cannot be factorized. As can be seen, the quality of the approximation can sometimes be poor, especially if there is no structure to the distribution being approximated. A more thorough analysis of the approximation produced by Eqn. 10.5 can be found in [4].



(a) Visualization of factorizing a separable 2-D Gaussian. Since the Gaussian is separable, it can be represented exactly as the product of two 1-D Gaussians.



(b) Visualization of factorizing a non-separable 2-D Gaussian. Since this Gaussian is not separable, it cannot be represented exactly as the product of two 1-D Gaussians.



(c) Visualization of factorizing a randomly-generated 2-D probability distribution. Note that the distribution is not separable, so its representation as the product of two 1-D distributions is not exact.

Figure 10.1: Visualization of three examples of using Proposition 44 to approximate a 2-D probability distribution by a product two 1-D distributions. Left figures: visualization of a 2-D distribution $p(x_1, x_2)$. Right figures: visualization of $p_1^*(x_1)p_2^*(x_2)$. Darker pixels correspond to higher probabilities.

10.3.1 Alternative approximations

To measure the disagreement between two distributions, one could use a function different than the particular KL divergence used in Proposition 44. One alternative is to reverse the direction of the KL divergence and instead use $D_{KL}(\prod_{i=1}^{N} p_i(x_i)||p(X))$. This direction of the KL divergence is the same as the one used in variational inference (see [63] and [4]), while the direction of the KL divergence in Proposition 44 is the same as in Expectation-Propagation (see [39]).

Another alternative function to measure the disagreement between two distributions is the Frobenius Norm. In the special case of a 2-D distribution, the problem of finding an optimal decomposition of a 2-D distribution in terms of two 1-D distributions is related to finding the best rank-1 approximation of a matrix in the Frobenius Norm sense, which can be solved using the Singular Value Decomposition. This problem is also related to generating separable filters (see [52]).

10.4 Decomposing a 1D $\mathrm{Uniform}$ distribution

Consider a $\mathrm{Uniform}$ distribution over the set $[K]$. In this section, we consider representing this distribution as a combination of $\mathrm{Categorical}$ distributions. Our motivation for doing so is that a $\mathrm{Uniform}$ distribution over the set $[K]$ has support size of K, but a representation of this distribution as a combination of $\mathrm{Categorical}$ distributions may have total support less than K. As shown in Eqn. 10.1 the number of edges in the factor graph representation of a PSG is proportional to the total support of the conditional pose distributions. By representing a $\mathrm{Uniform}$ distribution as a combination of $\mathrm{Categorical}$ distributions, the PSG factor graph may have fewer edges. As an example, suppose $p(z)$ is a $\mathrm{Uniform}$ distribution over the set $[100]$. Then, let $p_1(z_1)$ be a $\mathrm{Uniform}$ distribution over the set $[10]$ and let $p_2(z_2)$ be a $\mathrm{Uniform}$ distribution over the set $\{0, 10, \ldots, 90\}$. The distribution over $z_0 = z_1 + z_2$ is uniform over the set $[100]$ and the total support is 20.

Definition 45 *Consider a set of* $\mathrm{Categorical}$ *distributions* $P = \{p_i(z) \mid 1 \leq i \leq N\}$. *Let* Λ_i *denote the support of* $p_i(z)$ *and let* $\Lambda = \{\Lambda_i \mid 1 \leq i \leq N\}$. *Define*

$$|\Lambda| = \sum_{i=1}^{N} |\Lambda_i|. \tag{10.13}$$

Intuitively, the higher $|\Lambda|$ is, the more factor graph edges must be used to represent P in the PSG factor graph. In this section, given a $\mathrm{Uniform}$ distribution, we seek a representation of it as a set of $\mathrm{Categorical}$ distributions P with an associated set of supports Λ such that $|\Lambda|$ is minimized over all possible choices of P. We first show how to compute the minimum value of $|\Lambda|$, and then show how one can find such a P that achieves this minimum value for $|\Lambda|$.

Theorem 46 *Let $P = \{p_i(z) \mid 1 \leq i \leq N\}$ be a set of* Categorical *distributions specified so that the sum $z_0 = \sum_{i=1}^{N} z_i$, $z_i \sim p_i(z)$ is distributed uniformly on the set $[K]$. Let $\Lambda = \{\Lambda_i \mid 1 \leq i \leq N\}$ where Λ_i is the support of $p_i(z)$. The minimum possible value of $|\Lambda|$ is the sum of the prime factors (with repetition) of K.*

Consider again the problem of representing a Uniform distribution over the set $[100]$. Here, $K = 100$, so Theorem 46 states that the minimum possible value of $|\Lambda|$ is 14 in this situation since $100 = 2 \times 2 \times 5 \times 5$.

Before proving 46, we require some intermediate results.

Theorem 47 *Let $P = \{p_i(z) \mid 1 \leq i \leq N\}$ be a set of* Categorical *distributions and define $z_0 = \sum_{i=1}^{N} z_i$, $z_i \sim p_i(z)$. Then, $p(z_0)$ can be expressed as*

$$p(z_0) = p_1(z_0) * p_2(z_0) * \ldots * p_N(z_0)$$

where $*$ is the convolution operator.

Theorem 47 is a well-known result from the statistics literature.

Proposition 48 *Let $p_1(z)$ and $p_2(z)$ be two* Categorical *distributions such that $p(z) = p_1(z) * p_2(z)$ is a* Uniform *distribution on the set $[K]$. Let Λ_i be the support set of $p_i(z)$. Then, $p_1(z) = \frac{1}{|\Lambda_1|}$ $\forall z \in \Lambda_1$, and $p_2(z) = \frac{1}{|\Lambda_2|}$ $\forall z \in \Lambda_2$. In other words, both $p_1(z)$ and $p_2(z)$ are* Uniform *distributions over the sets Λ_1 and Λ_2, respectively.*

The proof of Proposition 48 can be found in [69].

Proposition 49 *Let $P = \{p_i(z) \mid 1 \leq i \leq N\}$ be a set of* Categorical *distributions such that $p(z) = p_1(z) * p_2(z) * \ldots * p_N(z)$ is a* Uniform *distribution on the set $[K]$. Let Λ_i be the support of $p_i(z)$. Then, there exists a setting of P such that $p_i(z) = \frac{1}{|\Lambda_i|}$ $\forall z \in \Lambda_i$, $1 \leq i \leq N$ (*i.e.*, the $p_i(z)$ are* Uniform *distributions).*

Proof of Proposition 49

The result follows by mathematical induction and using Proposition 48 as the base case.

■

Proposition 50 *Let $P = \{p_i(z) \mid 1 \leq i \leq N\}$ be a set of* Categorical *distributions such that $p(z) = p_1(z) * p_2(z) * \ldots * p_N(z)$ is a* Uniform *distribution on the set $[K]$. Let P be specified such that $p_i(z) = \frac{1}{|\Lambda_i|}$ $\forall z \in \Lambda_i$, $1 \leq i \leq N$. Consider the quantity $z_0 = \sum_{i=1}^{N} z_i$, $z_i \sim p_i(z)$. For all possible values of z_0, there is a unique setting of the z_i that sums to z_0.*

Proof of Proposition 50

Consider the smallest possible value of z_0. Denote this value by $z^\dagger$. There is only one setting of the z_i that sums to $z^\dagger$; in particular, this setting is to choose the smallest possible value for each of the z_i. From Proposition 49, $p(z^\dagger) = \prod_{i=1}^N \frac{1}{|\Lambda_i|}$.

Suppose there was value of z_0 for which there were multiple settings of the z_i that sum to it. Denote this value by z^*. By Proposition 50, each setting of the z_i that sums to z^* is drawn with probability $\prod_{i=1}^N \frac{1}{|\Lambda_i|}$. Therefore, if there are m settings of the z_i that sum to z^*, then $p(z^*) = m \prod_{i=1}^N \frac{1}{|\Lambda_i|}$. However, since P is specified so that $p(z)$ is a $\mathrm{Uniform}$ distribution, then we must have $p(z^\dagger) = p(z^*)$. This implies that $m = 1$, which implies that z^* has a unique setting of the z_i that sums to it. We have arrived at a contradiction. ■

Proposition 51 *Suppose we express a number K as a product of natural numbers, $K = \prod_{i=1}^N a_i$, $a_i \in \mathbb{N}$. Suppose we wish to minimize the quantity $a_0 = \sum_{i=1}^N a_i$ over all possible choices for N and all settings of the a_i. Setting the a_i to be the prime factors (with repetition) of K achieves the minimum value of a.*

Proof of Proposition 51

Suppose there exists a number K and a setting for the a_i for which the proposition does not hold. Then, one of the a_i must not be prime. Without loss of generality, suppose a_1 is not prime. If a_1 is not prime, then it must have at least two factors neither of which are 1. Denote these factors by c and d. Suppose we replace a_1 by cd. Let $a_0^* = c + d + \sum_{i=2}^N a_i$, which represents the effect of replacing a_1 by cd on a_0. Now,

$$\begin{aligned} a_0 - a_0^* &= \sum_{i=1}^N a_i - (c + d + \sum_{i=2}^N a_i) && (10.14)\\ &= a_1 - c - d && (10.15)\\ &= cd - c - d && (10.16)\\ &= (c-1)(d-1) - 1 && (10.17) \end{aligned}$$

and so $a_0 - a_0^* \geq 0$ whenever $c \geq 2$ and $d \geq 2$, which implies $a_0 \geq a_0^*$ whenever $c \geq 2$ and $d \geq 2$. This is a contradiction and the proposition must hold. ■

We are now ready to prove Theorem 46.

Proof of Theorem 46

From Theorem 47, $p(z_0)$ can be represented as $p(z_0) = p_1(z_0) * p_2(z_0) * \ldots * p_N(z_0)$. Let Λ_0 denote the set of possible values of z_0. From Proposition 50, each possible value of z_0 has a unique setting of the z_i that sums to it. This implies that $|\Lambda_0| = \prod_{i=1}^{N} |\Lambda_i|$. Since $p(z_0)$ is distributed uniformly on the set $[K]$, $|\Lambda_0| = K$. Therefore, we seek a setting for P such that $K = \prod_{i=1}^{N} |\Lambda_i|$ and $|\Lambda| = \sum_{i=1}^{N} |\Lambda_i|$ is minimized. From Proposition 51, the minimum is achieve when P is set such that the $|\Lambda_i|$ are the prime factors (with repetition) of K, and so the minimal value of $|\Lambda|$ is the sum of the prime factors (with repetition) of K. ■

We finish this subsection by giving a construction of P that realizes the minimum value of $|\Lambda|$.

Proposition 52 *Suppose we seek a set of* Categorical *distributions* $P = \{p_i(z) \mid 1 \leq i \leq N\}$ *such that* $p_1(z) * p_2(z) * \ldots * p_N(z)$ *is a* Uniform *distribution on the set* $[K]$. *Denote the support of* $p_i(z)$ *by* Λ_i, *and let* $\Lambda = \{\Lambda_i \mid 1 \leq i \leq N\}$. *Let* $l_i = \sum_{j=1}^{i-1} |\Lambda_j|$. *Consider the following specification of* P.

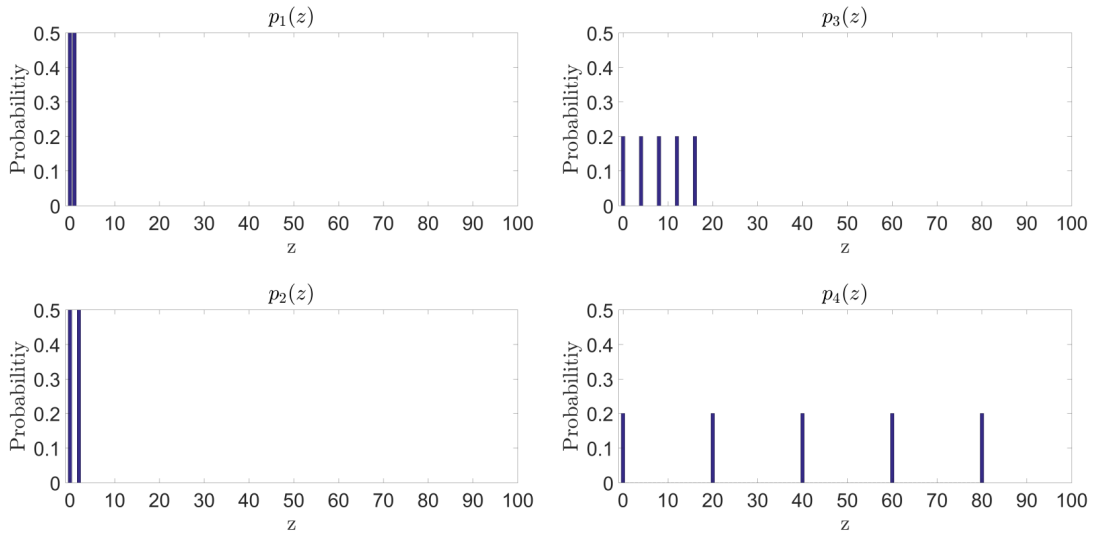
Set N *to be the number of prime factors (with repetition) of* K, *set the* $|\Lambda_i|$ *to be the prime factors (with repetition) of* K, *and set* $p_1(z) = \text{Uniform}(\{0, 1, \ldots, |\Lambda_1| - 1\})$ *and* $p_i(z) = \text{Uniform}(\{0, l_i, 2l_i, \ldots, (|\Lambda_i| - 1)l_i\})$ *for* $2 \leq i \leq N$. *Then,* $p(z) = p_1(z) * p_2(z) * \ldots * p_N(z)$ *is a* Uniform *distribution on the set* $[K]$ *and out of all settings of* P *that represents a* Uniform *distribution on the set* $[K]$, *the above construction achieves the minimum value for* $|\Lambda|$.

Proof of Proposition 52

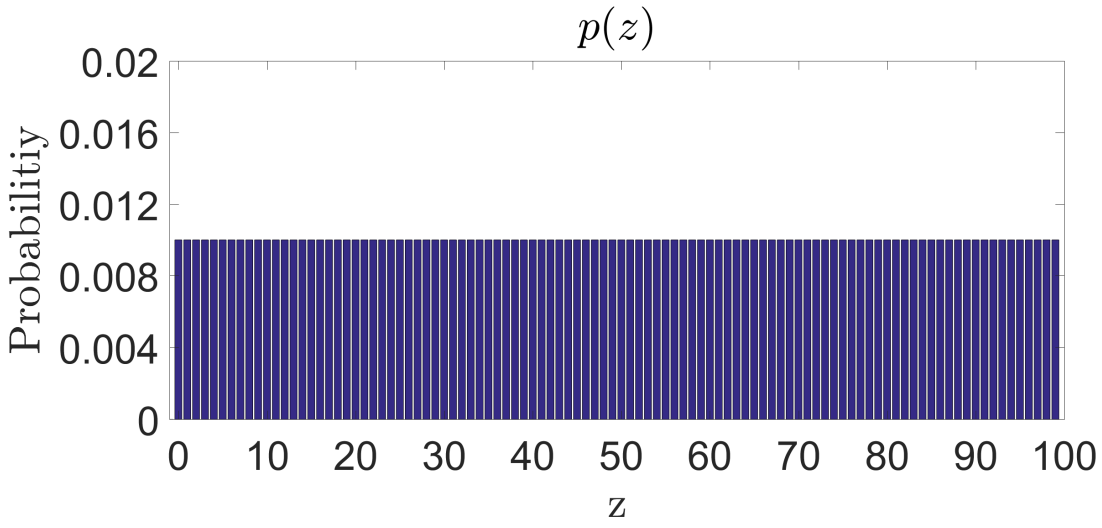
First, $p(z)$ represents a Uniform distribution over the set $[K]$ since each value of z has a unique representation as a sum of the possible values of the z_i, $z_i \sim p_i(z)$ Finally, since the $|\Lambda_i|$ are set to be the prime factors of K (with repetition), by Theorem 46, the construction achieves the minimum value for $|\Lambda|$. ■

Theorem 46 can be trivially extended to Uniform distributions on the set $\{N, N+1, \ldots, N+K-1\}$. To construct a set of distributions P that represents such a Uniform distribution, use the construction described in Proposition 52 except set $p_1(z) = \text{Uniform}(\{N, N+1, \ldots, N+|\Lambda_1|-1\})$

We give some examples of using the construction described in Proposition 52 to represent a 1D Uniform distribution. Figures 10.2 and 10.3 show examples of this construction. As can be seen in the figures, the constructions are exact and according to Theorem 46, achieve the minimal value of $|\Lambda|$.

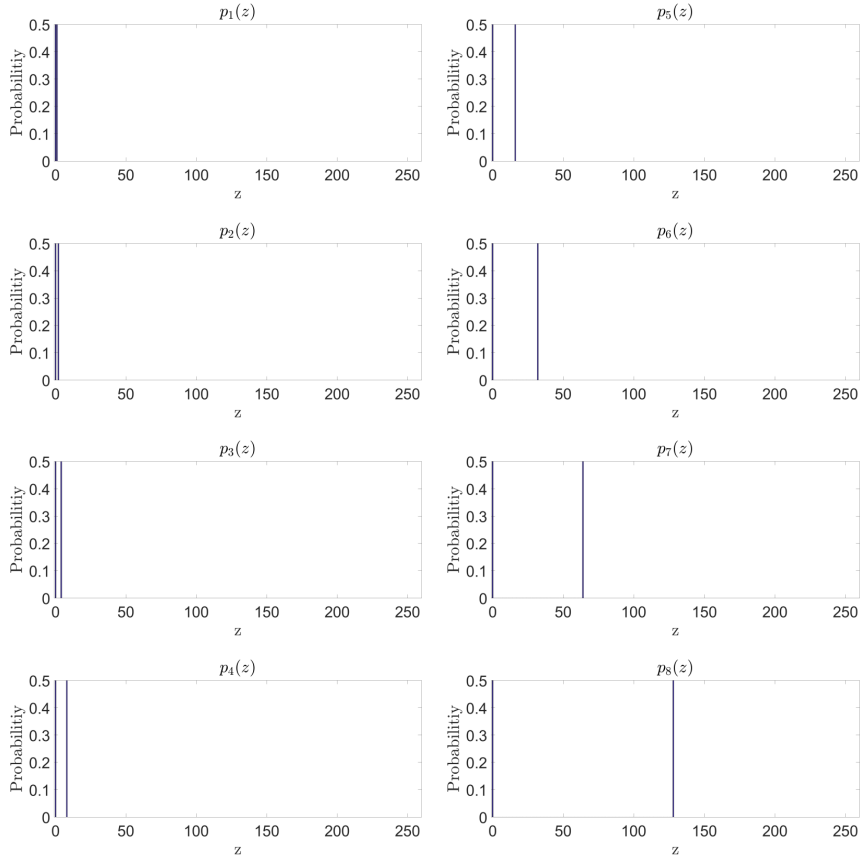


(a) $p_i(z), 1 \leq i \leq 4$ for representing a $\mathrm{Uniform}$ distribution on the set $[100]$ constructed using the process described in Proposition 52. Note that the support sizes are 2,2,5,5.

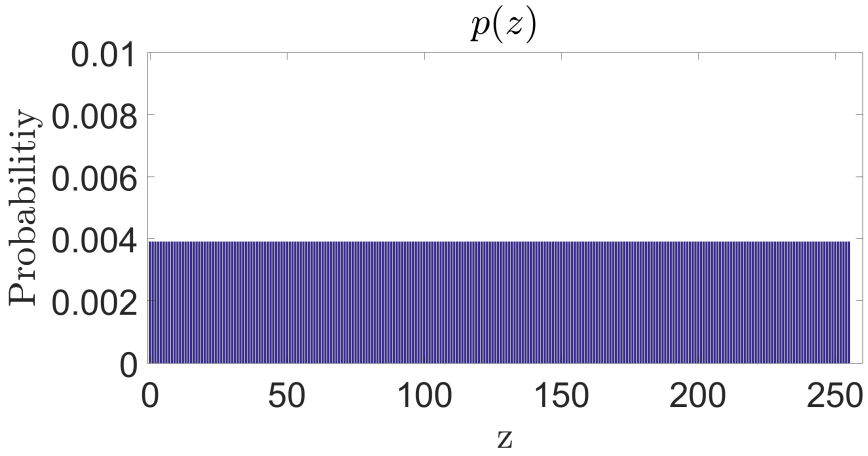


(b) Probability distribution $p(z) = p_1(z) * p_2(z) * p_3(z) * p_4(z)$. Note that $p(z)$ is uniform over the set $[100]$.

Figure 10.2: Example of using the construction described in Proposition 52 to represent a $\mathrm{Uniform}$ distribution over the set $[100]$.



(a) $p_i(z)$, $1 \leq i \leq 8$ for representing a Uniform distribution on the set $[256]$ constructed using the process described in Proposition 52. Note that the support size of each distribution is 2.



(b) Probability distribution $p(z) = p_1(z) * p_2(z) * \ldots * p_8(z)$. Note that $p(z)$ is exactly uniform over the set $[256]$.

Figure 10.3: Example of using the construction described in Proposition 52 to represent a Uniform distribution over the set $[256]$.

10.5 Applications to PSG design

In the PSG framework, a PSG $\mathcal{G}$ is associated with a factor graph $\mathcal{F}$ whose construction is described in Chapter 4. Suppose one wishes to specify a PSG $\mathcal{G}'$ such that its associated factor graph $\mathcal{F}'$ has fewer edges than $\mathcal{F}$, but $\mathcal{G}'$ induces a similar distribution over scenes as $\mathcal{G}$. In this section, given $\mathcal{G}$, we outline a construction of such a grammar $\mathcal{G}'$ using the techniques developed in Sections 10.3 and 10.4. In particular, we will apply the approximation scheme described in Proposition 44 and the construction described in Proposition 52 to reduce the total size of the supports of the conditional pose distributions.

Section 10.5.2 describes how the distributions over scenes induced by $\mathcal{G}$ and $\mathcal{G}'$ are related.

10.5.1 Constructing $\mathcal{G}'$

Informally, we construct a PSG $\mathcal{G}'$ from a $\mathcal{G}$ by adding symbols and modifying and adding compositional rules to $\mathcal{G}$.

Let $\mathcal{G} = (\Sigma, \Omega, \mathcal{R}, q, \epsilon, \gamma)$ represent a PSG. Let the conditional pose distribution $\gamma_{(\omega,r,i)}, \omega \in \Omega_{A_{(r,i)}}$, $r \in \mathcal{R}$, $1 \leq i \leq n_r$ be a Categorical distribution. Suppose we wish to represent $\gamma_{(\omega,r,i)}$ as a combination of distributions, as in Propositions 44 and 52 . If $\gamma_{(\omega,r,i)}$ is an N-D Categorical distribution, apply the approximation described in Proposition 44 to approximate it as N 1-D Categorical distributions. If $\gamma_{(\omega,r,i)}$ is a 1-D Uniform distribution, use the construction described in Proposition 52 to represent it as a set of Categorical distributions. In both cases, $\gamma_{(\omega,r,i)}$ is replaced by a set of distributions $P_{(\omega,r,i)} = \{p_{(\omega,r,i,j)} \mid 1 \leq j \leq k_{(\omega,r,i)}\}$ where $k_{(\omega,r,i)}$ is the number of distributions $\gamma_{(\omega,r,i)}$ is represented by. For simplicity, we assume that $k_{(\omega,r,i)}$ is a constant with respect to ω. For brevity, we will denote $k_{(\omega,r,i)}$ by $k_{(r,i)}$ $\forall \omega \in \Omega_{A_{(r,i)}}$.

Below, let $r = A_0 \to A_1, \ldots, A_n$ be a rule. To denote the rule r, the rule selection probability, q_r, and the conditional pose distributions $\gamma_{(\omega,r,i)}$ for $1 \leq i \leq n_r$ and a given pose $\omega \in \Omega_{A_0}$, we write,

$$q_r, \quad (A_0, \omega) \quad \to \quad (A_1, \gamma_{(\omega,r,1)}), \ldots, (A_n, \gamma_{(\omega,r,n)}). \tag{10.18}$$

Definition 53 *Given a PSG $\mathcal{G} = (\Sigma, \Omega, \mathcal{R}, q, \epsilon, \gamma)$, a rule $r \in \mathcal{R}$ and the i-th symbol in the RHS of r, consider replacing the conditional pose distribution $\gamma_{(\omega,r,i)}$, $\omega \in \Omega_{A_{(r,i)}}$, with a set of distributions $P_{(\omega,r,i)} = \{p_{(\omega,r,i,j)} \mid 1 \leq j \leq k_{(r,i)}\}$ $\forall \omega \in \Omega_{A_{(r,i)}}$. Let $\mathcal{G}' = (\Sigma', \Omega', \mathcal{R}', q', \epsilon', \gamma')$ be a PSG that realizes such a replacement. Below is a process of constructing $\mathcal{G}'$ from $\mathcal{G}$. The process is to be followed in the order given.*

1. *Create a set of fresh symbols $B = \{B_j \mid 1 \leq j \leq k_{(r,i)} - 1\}$.*

2. *Set* $\Omega_b = \Omega_{A_{(r,i)}}$, $\forall b \in B$.

3. *Set* $\epsilon_b = 0$, $\forall b \in B$.

4. *Assign* $\Sigma' = \Sigma \cup B$.

5. *Assign* $\Omega' = \Omega \bigcup \Omega_{B_1} \bigcup \ldots \bigcup \Omega_{B_{k_{(r,i)}-1}}$.

6. *Assign* $\epsilon' = \epsilon \bigcup \epsilon_{B_1} \bigcup \ldots \bigcup \epsilon_{B_{k_{(r,i)}-1}}$.

7. *Assign* $\mathcal{R}' = \mathcal{R}$, $q' = q$, $\gamma' = \gamma$.

8. *Rule* $r \in \mathcal{R}'$ *has the form*

$$q_r, \quad (A_0, \omega) \quad \rightarrow \quad (A_1, \gamma_{(\omega,r,1)}), \ldots, (A_i, \gamma_{(\omega,r,i)}), \ldots, (A_n, \gamma_{(\omega,r,n)}).$$

   Replace it by the rule

$$q_r, \quad (A_0, \omega) \quad \rightarrow \quad (A_1, \gamma_{(\omega,r,1)}), \ldots, (B_1, p_{(\omega,r,i,1)}), \ldots, (A_n, \gamma_{(\omega,r,n)}).$$

9. *Add rules of the form*

$$1.0, \quad (B_{j-1}, \omega) \quad \rightarrow \quad (B_j, p_{(\omega,r,i,j)}),$$

   $2 \leq j \leq k_{(r,i)} - 1$, *to* $\mathcal{G}'$.

10. *Add a rule of the form*

$$1.0, \quad (B_{k_{(r,i)}-1}, \omega) \quad \rightarrow \quad (A_{(r,i)}, p_{(\omega,r,i,k_{(r,i)})})$$

    to $\mathcal{G}'$.

The transformation process described in Definition 53 can be repeated for as many pairs (r, i), $r \in \mathcal{R}$, $1 \leq i \leq n_r$ as one desires. As we will see in the next subsection where we give examples of using the process above, this transformation process can reduce the number of factor graph edges used to represent the PSG.

10.5.2 Examples: transformation of grammars

Consider the PSG below that models faces and noses.

Grammar 10 *A grammar that models faces and noses in an* $N \times M$ *scene:*

$\Sigma = \{\mathsf{FACE}, \mathsf{NOSE}\}$.
$\forall A \in \Sigma,\ \Omega_A = [N] \times [M]$.
Rules:
$1.0, \quad (\mathsf{FACE}, \omega) \quad \rightarrow \quad (\mathsf{NOSE}, \text{UniformRect}(\omega - (12, 12), \omega + (12, 12)))$
$1.0, \quad (\mathsf{NOSE}, \omega) \quad \rightarrow \quad \emptyset$
$\epsilon_{\mathsf{FACE}} = \epsilon_{\mathsf{NOSE}} = 10^{-4}$,

We apply Proposition 53 to transform Grammar 10 into Grammar 11 below. In particular, we apply the approximation described in Proposition 44 to rule 1 and the first RHS symbol to factorize its associated 2-D distribution conditional pose distribution.

Grammar 11 *A transformation of Grammar 10 for an $N \times M$ scene using Proposition 44:*

$\Sigma = \{\mathsf{FACE}, \mathsf{NOSE}, \mathsf{NOSE\text{-}Y}\}$.
$\forall A \in \Sigma,\ \Omega_A = [N] \times [M]$.
Rules:
$1.0, \quad (\mathsf{FACE}, \omega) \quad \rightarrow \quad (\mathsf{NOSE\text{-}Y}, \text{Uniform}(\{\omega + (0, -12), \ldots, \omega + (0, 12)\}))$
$1.0, \quad (\mathsf{NOSE\text{-}Y}, \omega) \quad \rightarrow \quad (\mathsf{NOSE}, \text{Uniform}(\{\omega + (-12, 0), \ldots, \omega + (12, 0)\}))$
$1.0, \quad (\mathsf{NOSE}, \omega) \quad \rightarrow \quad \emptyset$
$\epsilon_{\mathsf{FACE}} = \epsilon_{\mathsf{NOSE}} = 10^{-4}$,
$\epsilon_{\mathsf{NOSE\text{-}Y}} = 0$.

Consider expanding a FACE brick using Grammar 11. Rules 1 and 2 model a FACE brick generating a NOSE brick. In particular, rules 1 and 2 model the sequential process of a FACE brick first choosing a Y coordinate for the NOSE brick, then choosing an X coordinate for the NOSE brick.

Finally, we apply the process described in Definition 53 to transform Grammar 11 into Grammar 12 below. In particular, we use the construction described in Proposition 52 to the first RHS symbol of rules 1 and 2.

Grammar 12 *A transformation of Grammar 11 for an $N \times M$ scene using Proposition 52:*

$\Sigma = \{\mathsf{FACE}, \mathsf{NOSE}, \mathsf{NOSE\text{-}Y}, \mathsf{NOSE\text{-}Y1}, \mathsf{NOSE\text{-}Y2}\}$.

$\forall A \in \Sigma,\ \Omega_A = [N] \times [M]$.

Rules:

$$
\begin{aligned}
&1.0, \quad (\mathsf{FACE}, \omega) && \rightarrow \quad (\mathsf{NOSE\text{-}Y1}, \text{Uniform}(\{\omega + (0,-12), \ldots, \omega + (0,-8)\})) \\
&1.0, \quad (\mathsf{NOSE\text{-}Y1}, \omega) && \rightarrow \quad (\mathsf{NOSE\text{-}Y}, \text{Uniform}(\{\omega + (0,0), \omega + (0,5), \ldots, \omega + (0,20)\})) \\
&1.0, \quad (\mathsf{NOSE\text{-}Y}, \omega) && \rightarrow \quad (\mathsf{NOSE\text{-}Y2}, \text{Uniform}(\{\omega + (-12,0), \ldots, \omega + (-8,0)\})) \\
&1.0, \quad (\mathsf{NOSE\text{-}Y2}, \omega) && \rightarrow \quad (\mathsf{NOSE}, \text{Uniform}(\{\omega + (0,0), \omega + (5,0), \ldots, \omega + (20,0)\})) \\
&1.0, \quad (\mathsf{NOSE}, \omega) && \rightarrow \quad \emptyset
\end{aligned}
$$

$\epsilon_{\mathsf{FACE}} = \epsilon_{\mathsf{NOSE}} = 10^{-4}$,

$\epsilon_{\mathsf{NOSE\text{-}Y}} = \epsilon_{\mathsf{NOSE\text{-}Y1}} = \epsilon_{\mathsf{NOSE\text{-}Y2}} = 0$.

Consider expanding a FACE brick using Grammar 12. Rules 1-4 model a FACE brick generating a NOSE brick. In particular, rules 1 and 2 model the process of choosing a Y coordinate for the NOSE brick in terms of a two-stage process, and rules 3 and 4 model the process of choosing an X coordinate for the NOSE brick in terms of a two-stage process.

Table 10.3 summarizes the effect of recursively applying the construction process outlined in Section 10.5.1 to Grammar 10 on the number of factor graph edges. Note that Grammar 11 has more than an order of magnitude fewer edges in its associated factor graph than Grammar 10, and Grammar 12 has even fewer edges than Grammar 11. Recall from Chapter 5 that the run time of one iteration of LBP is linear in the number of edges in the factor graph. Thus, recursively applying the techniques described in Sections 10.3 and 10.3 on Grammar 12 confers an approximately 21x speed-up.

Grammar	**Number of edges in the factor graph**
Grammar 10	$1258NM$
Grammar 11	$112NM$
Grammar 12	$60NM$

Table 10.3: Number of edges in the factor graph of the Grammar models described in this section. Recall that the grammar consider an $N \times M$ scene.

Next, we study the distribution over scenes induced by Grammars 10, 11, and 12. In particular, we show that the distribution over scenes is not the same and show a scenario in which they are not the same.

Figure 10.4 shows the approximate marginals computed by running LBP on the factor graph representations of Grammars 10, 11, and 12 for several examples. Note that the approximate marginals are similar across all examples for the three grammars. This suggests that in practice, any of Grammars 10, 11, or 12 can be used in place of any of the other grammars. However, the

computed approximate marginals are not the same for all grammars. Let $\hat{p}_A^0$, $\hat{p}_A^1$, and $\hat{p}_A^2$ denote the largest approximate marginals computed by LBP for a symbol $A \in \{\mathsf{FACE}, \mathsf{NOSE}\}$ for Grammars 10, 11, and 12, respectively. Table 10.4 lists the ratios $\frac{\hat{p}_A^0}{\hat{p}_A^1}$ and $\frac{\hat{p}_A^0}{\hat{p}_A^1}$ $A \in \{\mathsf{FACE}, \mathsf{NOSE}\}$.

Ratio	Example: Figure 10.4(a)	Example: Figure 10.4(b)	Example: Figure 10.4(c)	Example: Figure 10.4(d)
$\hat{p}^0_{\mathsf{FACE}}/\hat{p}^1_{\mathsf{FACE}}$	1.0000	1.0000	1.0000	1.0000
$\hat{p}^0_{\mathsf{NOSE}}/\hat{p}^1_{\mathsf{NOSE}}$	1.0001	1.0185	1.0000	1.0095
$\hat{p}^0_{\mathsf{FACE}}/\hat{p}^2_{\mathsf{FACE}}$	1.0000	1.0000	1.0023	1.0000
$\hat{p}^0_{\mathsf{NOSE}}/\hat{p}^2_{\mathsf{NOSE}}$	1.0000	1.0187	1.0001	1.0097

Table 10.4: $\hat{p}_A^0$, $\hat{p}_A^1$, and $\hat{p}_A^2$ denote the largest approximate marginals computed by LBP for a symbol $A \in \{\mathsf{FACE}, \mathsf{NOSE}\}$ for Grammars 10, 11, and 12, respectively. Some ratios of these approximate marginals are shown in the table. As shown, the largest approximate marginals computed by LBP are not identical across the grammars, suggesting that each of the grammars models induces a different distribution over scenes.

The ratios in Table 10.4 show that the approximate marginals produced by LBP are not identical, suggesting that Grammars 10, 11, and 12 each induce a slightly different distribution over scenes. In general, a PSG $\mathcal{G}$ and a transformation of it $\mathcal{G}'$ constructed using the process described in Section 10.5.1 do not induce the same distribution over scenes. We demonstrate this fact below for Grammars 10, 11, and 12.

Consider two bricks (FACE, ω) and $(\mathsf{FACE}, \omega + (0, 1))$. Suppose these bricks are present in a scene, and consider the probability of the event that a sequence of expansions for both bricks leads to the generation of a single NOSE brick (*i.e.*, their expansions "collide" and only one NOSE brick is generated instead of two). For Grammar 10, the probability of this event is $\frac{600}{625} \times \frac{1}{625} = \frac{600}{625^2}$. For Grammars 11 and 12, two FACE bricks will generate the same NOSE brick if and only if they generate the same NOSE-Y brick. So, the probability of the event is $\frac{24}{25} \times \frac{1}{25} = \frac{24}{25^2}$ for Grammar 11, and $\frac{4}{5} \times \frac{1}{5} = \frac{4}{25}$ for Grammar 12. Thus, the probability of the event that the sequence of expansions for both bricks leads to the generation of a single NOSE brick is different for Grammars 10, 11, and 12.

In general, the probability that a sequence of expansions for two bricks b_1 and b_2 leads to the generation of a common brick (*i.e.*, their expansions "collide") is a function of 1) the number of bricks b_1 can generate, 2) the number of bricks b_2 can generate, and 3) the number of common bricks b_1 and b_2 can generate. As such, any PSG transformation that changes these three quantities may change the distributions over scenes induced by the PSG.

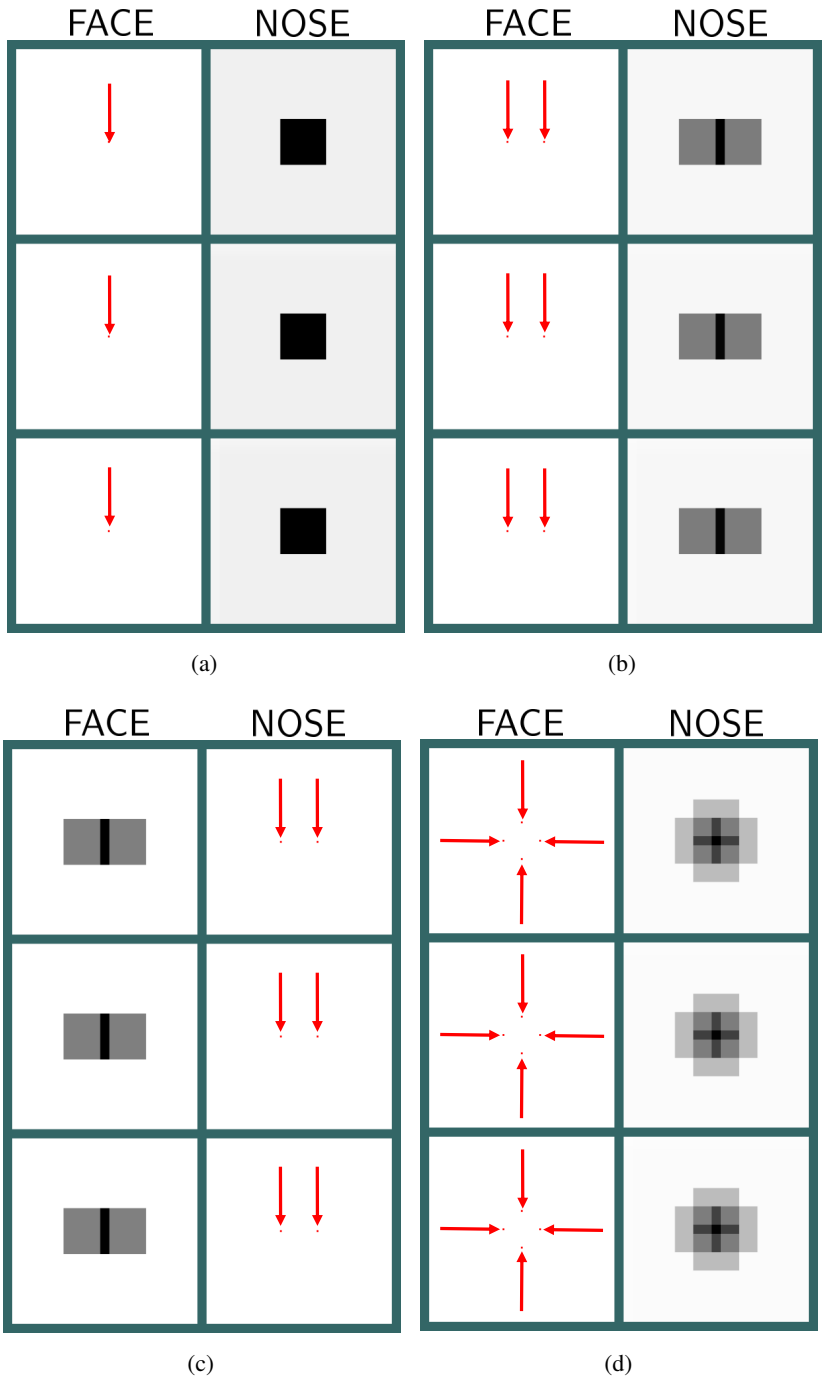


Figure 10.4: A visualization of the approximate marginals computed by LBP when conditioning on sets of bricks. Each subfigure represents a different example, and the top, middle, and bottom rows of each subfigure show the results of running LBP on the factor graph representation of Grammars 10, 11, and 12, respectively. We show only the approximate marginals for the FACE and NOSE bricks. Each pixel represents a brick at that pixel location. The bricks conditioned to be present in the image are denoted by a red pixel and a red arrow pointing to them. Darker pixels indicate a higher approximate marginal probability of being present. Note that for all examples, the approximate marginals produced by LBP are similar across the different grammars.

10.6 Notes on 1-D Uniform distributions with a prime support size

Given Theorem 46, one may be concerned that it is not possible to represent a 1-D Uniform distribution $p(z)$ with a prime support size in a manner that will result in a reduction in the number of edges of the PSG factor graph. For example, consider a Uniform distribution over $[101]$; Theorem 46 indicates that expressing this distribution as a convolution of a set of 1-D Categorical results in no computational savings since 101 is a prime number.

To deal with this scenario, one can partition the support of $p(z)$ into L contiguous partitions, express a Categorical distribution over these partitions with each partition being chosen proportional to size of the partition, and apply the decomposition in Proposition 52 to represent a Uniform distribution over the elements of each of the L partitions. The process to sample from $p(z)$ with support of size K given a partitioning of the support proceeds as follows: first, choose a partition with probability $\frac{|\Lambda_i|}{K}$ where $|\Lambda_i|$ is the size of partition i. Then, sample an element from the selected partition uniformly using the representation described in Proposition 52. The total support of this representation for $p(z)$ is $L + \sum_{i=1}^{L} |\Lambda_i|$ where $|\Lambda_i|$ is the size of partition i.

As an example, let $p(z)$ be a Uniform distribution over the set $\{0, \ldots, 100\}$. One can partition the support into two sets, $\Lambda_1 = \{0, \ldots, 50\}$ and $\Lambda_2 = \{51, \ldots, 100\}$. A Uniform distribution over the elements of each of these partitions can be represented using the decomposition in Proposition 52 . To sample from $p(z)$, first choose either L_1 or L_2 with probability $\frac{51}{101}$ and $\frac{50}{101}$, respectively. Then, sample an element from the selected partition uniformly using the representation described in Proposition 52 . The total support of representing a Uniform distribution over the set $\{0, \ldots, 100\}$ in this fashion is $2 + 20 + 12 = 34$. This technique can be applied even if the support of $p(z)$ is not prime. The solution to finding an optimal partitioning of the support of $p(z)$ to minimize the total support of the representation scheme given here is currently unknown and is a direction for future research.

10.7 Notes on general 1-D Categorical distributions

The techniques to decompose a 1-D distribution outlined in Section 10.4 only apply to 1-D Uniform distributions. In the case of a general 1-D Categorical distribution, one cannot use the aforementioned techniques. The general problem of decomposing a general 1-D Categorical distribution into a set of 1-D Categorical distributions with smaller total support is related to the problem of blind deconvolution (see [37]) with a prior that encourages sparsity in the composing distributions. This general problem is more difficult than the special case considered in this thesis where the 1-D distribution is Uniform. The solution to this more general problem is a goal of future research.

In this work, the task of decomposing a 1-D $\mathrm{Categorical}$ distribution into a set of 1-D $\mathrm{Categorical}$ distributions with smaller total support is motivated by a desire to define a PSG that gives rise to a factor graph with a smaller number edges. If one wishes to use a particular 1-D $\mathrm{Categorical}$ distribution to define a PSG, presumably that distribution was estimated from training data. Instead of decomposing a given $\mathrm{Categorical}$ distribution directly, one can first define a family of 1-D $\mathrm{Categorical}$ distributions parameterized by the parameters of N 1-D $\mathrm{Categorical}$ distributions with fixed support. Then, one can express this decomposition directly in a PSG as was done in Section 10.5.2. Finally, one can fit the parameters of the N 1-D $\mathrm{Categorical}$ distributions with the learning algorithm defined in Chapter 8 using the training data. This process results in a PSG with a smaller number of factor graph edges and an approximation to the target 1-D $\mathrm{Categorical}$ distribution in terms of a convolution of N 1-D $\mathrm{Categorical}$ distributions. Providing a detailed analysis of the goodness of the resulting approximation is a goal of future research.

Chapter 11

Contributions and Suggestions for Future Research

To conclude this thesis, we summarize the approach of the PSG framework, outline the research contributions, and suggest directions of future research that builds on the PSG framework.

11.1 Summary of approach and research contributions

In this thesis, we have introduced the Probabilistic Scene Grammar (PSG) framework: a general-purpose probabilistic framework for scene understanding tasks. For a given scene understanding task, we summarize the approach of the PSG framework below:

1. Represent a model for the given scene understanding task in the language of a grammar; Chapter 2 defines the grammar specification, and Chapter 3 gives examples of grammars for some scene understanding tasks.

2. Convert the grammar representation to a factor graph, as described in Chapter 4.

3. Directly estimate model parameters if fully-observed data is available. Otherwise, use the approximate EM learning algorithm described in Chapter 8 to estimate model parameters.

4. Use Loopy Belief Propagation (LBP) as the inference engine, as described in Chapter 5.

Importantly, the approach of the PSG framework is the same no matter the scene understanding task at hand. In theory, any scene understanding for which a suitable model can be expressed in the grammar language defined in Chapter 2 may be addressed in the PSG framework. However, practical limitations of the PSG of the framework may prevent its application to arbitrary scene

understanding tasks with arbitrary grammar models. Chapter 10 takes some steps towards addressing practical issues that must be resolved to enable the use of larger PSG models on more complex scene understanding tasks.

Chapter 9 evaluates the PSG framework on the scene understanding tasks of contour detection, face localization, and binary image segmentation. The PSG framework is competitive with specialized algorithms for these scene understanding tasks.

The main contributions of this thesis can be summarized as addressing four key aspects of defining and assessing a general-purpose probabilistic framework for scene understanding: 1) the **representation** of scene understanding tasks under a common schema (Chapters 2, 3 and 4), 2) efficient, problem-agnostic **approximate inference** (Chapters 5, 6, and 10), 3) the **learning** of model parameters under varying levels of supervision (Chapter 8), and 4) the **experimental evaluation** of the framework (Chapter 9). The concertization of this framework in a general implementation is a final, engineering-oriented contribution.

11.2 Directions for future research

In this section, we discuss promising directions for future research that build on the PSG framework. The proposed directions deal with issues concerning 1) the use of richer data models (*e.g.*, data models defined by deep learning models), 2) the application of the PSG framework to more scene understanding tasks, and 3) the practical limitations of the PSG framework in its current form.

11.2.1 Integration of deep learning models

In recent years, the approach of deep learning has demonstrated impressive results on scene understanding tasks (see [50, 66, 36, 32, 46, 21] for a few examples). In a sense, deep learning is a general-purpose scene understanding framework as well in the sense that deep learning seeks to learn a mapping between inputs (*e.g.*, images) and outputs (*e.g.*, class labels, segmentations, object localizations, etc.). It is crucial to note that deep learning does not have to be viewed as a competitor to probabilistic models; both can be used together in a coherent system. The emerging subfield of Bayesian Deep Learning (see [64] for a brief survey) seeks to combine probabilistic approaches with deep learning techniques.

In the context of the PSG framework, one could imagine using the output of a convolution neural network (CNN) such as the one described in [32] as a data model. We believe such an approach could combine the ability of deep learning to produce excellent low-level representations with the high-level reasoning ability of the PSG framework. Such a combination could allow one to tackle

scene understanding tasks that are currently difficult for both approaches. Consider, for example, the problem of detecting conversations in scenes when one has many examples of faces but few examples of conversations. The notion of a conversation is naturally modeled in a compositional relationship: a conversation can be thought of as a composition of faces that are facing each other and in close spatial proximity. Such compositional models can be naturally expressed in the PSG framework. Deep learning is capable of building a high-quality representation of objects when one has many examples of that object. In this example, deep learning could be used to build an excellent face-detector, but perhaps cannot be used to build a conversation-detector. One could use deep learning to detect faces, and the PSG framework to detect conversations using the face detections as an input.

11.2.2 Applications to more scene understanding tasks

In this thesis, we evaluate the PSG framework on the scene understanding tasks of face localization, contour detection, and binary image segmentation. As the PSG framework is general-purpose, there is a myriad of other scene understanding tasks that can be addressed in this framework. For example, the PSG framework can be used for motion tracking. The concept of motion can be naturally expressed in the PSG framework. Consider the problem of tracking a face through time. Recall that the face models we describe in Chapter 3 describe the location of faces and face parts in terms of spatial location. One could extend the model to include both spatial and temporal information. For example, a FACE brick at location (x, y) at time t could generate a FACE brick at location (x, y) at time $t + 1$.

The PSG framework could also be applied to larger and more complex scene understanding tasks. Consider the problem of localizing tumours from magnetic resonance imaging (MRI) brain scans. MRI brain scans are volumetric 3-D scans of a patient's brain. These scans can be fairly large; a typical scan may contain $200 \times 200 \times 144$ measurements. Given a brain scan, one seeks to output a 3-D segmentation of the brain that localizes tumours in the brain, if any. Here, the image data is the MRI scan which has two orders of magnitude more measurements than any of the scene understanding tasks we address in this thesis, and so speed/memory issues may arise. Also, the shape of a tumour can be quite complex, necessitating a more sophisticated notion of shape than the one used in Chapter 9 for leaf segmentation. Nevertheless, we believe tackling such large, complex scene understanding tasks is in the realm of possibility.

11.2.3 Structure learning

Recall from Chapter 8 that in this thesis, we propose a method to estimate model parameters of a PSG. However, we have not proposed a method to learn the structure of the grammar itself. For example, the PSG we use for face localization described in Chapter 9 has a notion that a FACE is comprised of a LEFT-EYE, RIGHT-EYE, NOSE, and a MOUTH. What if we did not know apriori that faces had this compositional structure? What if we did not know the parts of a face? In the context of the PSG framework, addressing such questions requires learning the compositional rules of the model and learning an appropriate set of symbols. The study of learning structure in compositional models has been examined in [24] and [54]. However, it may not be practical to directly apply the techniques of [24] and [54] to the PSG framework. The problem of efficiently learning the structure of a PSG model is difficult and is a key question one must solve before applying the PSG framework to scene understanding tasks where one cannot rely on expert knowledge to design the structure of the grammar.

Bibliography

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[2] Julian Besag. On the statistical analysis of dirty pictures. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY B*, 48(3):48–259, 1986.

[3] Elie Bienenstock, Stuart Geman, and Daniel Potter. Compositionality, MDL priors, and object recognition. In *Advances in Neural Information Processing Systems*, pages 838–844, 1997.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient n-d image segmentation. *Int. J. Comput. Vision*, 70(2):109–131, November 2006.

[6] Yuri Y. Boykov and Marie-Pierre Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *8th IEEE International Conference on Computer Vision*, volume 1, pages 105–112. IEEE Comput. Soc, 2001.

[7] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[8] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2):393 – 405, 1990.

[9] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[12] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

[13] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[14] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *Int. J. Comput. Vision*, 70(1):41–54, October 2006.

[15] Pedro F. Felzenszwalb and David McAllester. Object detection grammars. *Univerity of Chicago Computer Science Technical Report 2010-02*, 2010.

[16] Pedro F. Felzenszwalb and John G. Oberlin. Multiscale fields of patterns. In *Advances in Neural Information Processing Systems*, pages 82–90, 2014.

[17] Sanja Fidler, Marko Boben, and Aleš Leonardis. Learning a hierarchical compositional shape vocabulary for multi-class object representation. In *ArXiv:1408.5516*, 2014.

[18] Martin A. Fischler and Robert A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, (1):67–92, 1973.

[19] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[20] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, November 1984.

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[22] Ross B. Girshick, Pedro F. Felzenszwalb, and David Mcallester. Object detection with grammar models. In *Advances in Neural Information Processing Systems*, pages 442–450, 2011.

[23] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the Tenth IEEE International Conference on*

Computer Vision - Volume 2, ICCV '05, pages 1458–1465, Washington, DC, USA, 2005. IEEE Computer Society.

[24] Matthew T. Harrison. Discovering compositional structures. Technical report, 2005.

[25] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[26] Tom Heskes, Onno Zoeter, and Wim Wiegerinck. Approximate expectation maximization. In S. Thrun, L. K. Saul, and P. B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 353–360. MIT Press, 2004.

[27] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[28] Ya Jin and Stuart Geman. Context and hierarchy in a probabilistic image model. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2145–2152, 2006.

[29] Zoltan Kato and Ting-Chuen Pong. A markov random field image segmentation model for color textured images. *Image and Vision Computing*, 24(10):1103–1114, 2006.

[30] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, October 2006.

[31] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 109–117. Curran Associates, Inc., 2011.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[33] Frank R Kschischang, Brendan J Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[34] Tejas D. Kulkarni, Pushmeet Kohli, Joshua B. Tenenbaum, and Vikash K. Mansinghka. Picture: A probabilistic programming language for scene perception. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4390–4399, 2015.

[35] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[36] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[37] Anat Levin, Yair Weiss, Fredo Durand, and William T. Freeman. Understanding blind deconvolution algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(12):2354–2367, December 2011.

[38] Talya Meltzer, Amir Globerson, and Yair Weiss. Convergent message passing algorithms: A unifying view. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 393–401, Arlington, Virginia, United States, 2009. AUAI Press.

[39] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[40] David Mumford. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. Pure Applied Mathematics*, pages 577–685, 1989.

[41] David Mumford. Elastica and computer vision. In *Algebraic geometry and its applications*, pages 491–506. Springer, 1994.

[42] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

[43] Radford Neal. Slice sampling. *Annals of Statistics*, 31:705–767, 2000.

[44] Radford M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

[45] Stephen E Palmer. *Vision science: Photons to phenomenology*, volume 1. MIT press, 1999.

[46] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.

[47] Zhile Ren and Erik B. Sudderth. Three-dimensional object detection and layout prediction using clouds of oriented gradients. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1525–1533, 2016.

[48] Daniel Ritchie. *Probabilistic Programming for Procedural Modeling and Design*. PhD thesis, Stanford University, 2016.

[49] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Object class segmentation using random forests. In *Proc. British Machine Vision Conference (BMVC)*, January 2008.

[50] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3982–3991, 2015.

[51] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.

[52] Douglas Shy and Pietro Perona. Xy separable pyramid steerable scalable kernels. In *CVPR*, pages 237–244, 1994.

[53] Oskar Söderkvist. Computer vision classification of leaves from swedish trees. Master's thesis, 2001.

[54] Andreas Stolcke. Bayesian learning of probabilistic language models. Technical report, 1994.

[55] Thomas M. Strat. Employing contextual information in computer vision. In *In Proceedings of ARPA Image Understanding Workshop*, pages 217–229, 1993.

[56] Erik B. Sudderth, Alexander T. Ihler, Michael Isard, William T. Freeman, and Alan S. Willsky. Nonparametric belief propagation. *Commun. ACM*, 53(10):95–103, 2010.

[57] Deqing Sun, Jonas Wulff, Erik B. Sudderth, Hanspeter Pfister, and Michael J. Black. A fully-connected layered model of foreground and background flow. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2451–2458, 2013.

[58] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

[59] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of computer vision*, 63(2):113–140, 2005.

[60] Luminita A Vese and Tony F Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International journal of computer vision*, 50(3):271–293, 2002.

[61] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*, 2015.

[62] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz, and Antonio Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 1–8, Washington, DC, USA, 2013. IEEE Computer Society.

[63] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, January 2008.

[64] Hao Wang and Dit-Yan Yeung. Towards bayesian deep learning: A framework and some existing methods. *IEEE Trans. on Knowl. and Data Eng.*, 28(12):3395–3408, December 2016.

[65] Yair Weiss. Comparing the mean field method and belief propagation for approximate inference in mrfs, 2001.

[66] Jimei Yang, Brian L. Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. Object contour detection with a fully convolutional encoder-decoder network. *CoRR*, abs/1603.04530, 2016.

[67] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, 2005.

[68] Yibiao Zhao and Song-Chun Zhu. Image parsing with stochastic scene grammar. In *Advances in Neural Information Processing Systems*, pages 73–81, 2011.

[69] Anatoly Zhigljavsky, Nina Golyandina, and Svyatoslav Gryaznov. Deconvolution of a discrete uniform distribution. *Statistics and Probability Letters*, 118:37 – 44, 2016.

[70] Song-Chun Zhu and David Mumford. A stochastic grammar of images. *Found. Trends. Comput. Graph. Vis.*, 2(4):259–362, January 2006.