Compositional Pattern Recognition

by

Daniel Frederic Potter

Sc.B., Brown University, 1988
Sc.M., Brown University, 1990

Thesis

Submitted in partial fulfillment of the requirements for
the Degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

PROVIDENCE, RHODE ISLAND

May 1999

Abstract of "Compositional Pattern Recognition," by Daniel Frederic Potter, Ph.D., Brown University, May 1999

This thesis introduces a syntactic and probabilistic approach to pattern recognition based on the use Compositional Grammars and Compositional Distributions. Such grammars are related in spirit to the constraint-based grammar formalisms now popular in linguistics.

Analytic definitions and some basic properties of several classes of compositional grammars and distributions are established. These grammars and distributions are used to describe (that is, define a prior on) the objects to be recognized. A bayesian MAP or equivalently MDL formulation for scene recognition/interpretation is defined.

A chapter on recognition algorithms discusses some simple brute-force techniques for approximating solutions of this MAP/MDL problem. Another chapter presents an algorithm amenable to sampling certain compositional distributions.

Experiments with recognition and synthesis of online handprint characters and words provide an example of the approach. A compositional grammar and distribution is first used to define a prior on objects up to scale, position, and orientation; thus, a compositional grammar and distribution is used to define a measure on object orbits under the action of the semidirect product group $SE(2) \propto R_+$. Use of an application-specific conditional distribution on the remaining position, scale and orientation parameters extends the distribution to the actual objects to be recognized and sampled.

This dissertation by Daniel Frederic Potter is accepted in its present form by the
Division of Applied Mathematics as satisfying the
dissertation requirement for the degree of
Doctor of Philosophy

Date..... .....
Stuart Geman

Recommended to the Graduate Council

Date..... .....
Basilis Gidas

Date..... .....
Donald McClure

Approved by the Graduate Council

Date..... .....

The Vita of Daniel Frederic Potter

Daniel Frederic Potter was born on September 26, 1964, in Appleton, Wisconsin, United States of America. He attended Antioch College, Ohio in 1981 and 1982, and then worked for several years as a software engineer designing interactive museum exhibits and point of sales devices. He entered Brown University in 1986 where he received both a Sc.B. in Mathematics and Computer Science (with honors), in 1988, and a Sc.M. in Applied Mathematics, in 1990. In the summer of 1990 he worked in the Speech Recognition Group at Bell Communications Research. From 1991 to 1998 he worked as a independent consultant and on various projects at the Woods Hole Oceanographic Institution where he holds the title of Engineer. In 1992 he was admitted to the Ph.D. program in the Division of Applied Mathematics at Brown University. While in the Division of Applied Mathematics, and a full-time student (1992-1996), he has been the recipient of a Dean's Fellowship as well as support from the Office of Naval Research, the Army Research Office, and the National Science Foundation.

He was elected a member of the Sigma Xi scientific society in 1992. He is also a member of the Society for Industrial and Applied Mathematics (SIAM). Recent Papers:

Composition Systems, with S. Geman and Z. Chi. Technical Report, Brown University, 1998.

Compositionality – MDL Priors and Object Recognition, with E. Bienenstock and S. Geman. In M. Mozer, M.I. Jordan, T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 838. The MIT Press, 1997.

Acknowledgments

This work would have not been possible without the sound and patient guidance of my thesis advisor, Professor Stuart Geman. I would also like to express my appreciation and gratitude to Professors Ulf Grenander and David Mumford. They both played instrumental roles in my understanding of the foundations of pattern theory.

The Division of Applied Mathematics is blessed with a wonderful support staff including Laura Leddy, Jean Radican, Trudee Trudell, and Roselyn Winterbottom. These individuals have each helped me above and beyond the call of duty numerous times.

My family and friends also deserve a hearty thank you. I have worked on this Ph.D a long time, and you have all stood by me patiently.

Contents

List of Figures

Chapter 1

Introduction

Three areas are of fundamental interest in Bayesian approaches to pattern recognition. First, the statistical properties of the objects (or, equivalently, patterns) of interest must be captured by use of a prior. Second, a likelihood model for the way objects express themselves in the data must be developed. Third, effective algorithms for either simulating or performing MAP-like computations on the resulting *a posteriori* distribution must be developed.

The main contributions of this thesis fall into the first and last of these categories. A syntactic and probabilistic approach to specifying priors is developed which allows both the structure and statistics of patterns to be modeled in a hierarchical manner. Patterns which may be described in this way will be said to have a "compositional" structure. The primary benefits of this approach, from a modeling perspective, are that priors on large and complex sets of patterns with compositional structure can be defined using a relatively small number of parameters and that these parameters have an obvious and logical interpretation.

These priors possess some interesting computational properties. In some cases they allow for exact probability calculations. Also, a simple Monte Carlo (but non-MCMC) algorithm for sampling these prior is sometimes applicable.

For recognition, a number of algorithms are presented for doing MAP approximation on an a posteriori distribution defined using these priors. They are essentially "image-parsing" algorithms.

In order to illustrate the ideas presented, an online handprint recognition system (OHRS) is developed. The prior used models high level objects such as words as well as strings of letters, individual letters and letter components in a hierarchical manner which is essentially scale-, rotation- and translation-invariant. Initial input to the recognition system consists of digitizing stylus stroke/coordinate data organized in time; see Figure 1.1. This data is preprocessed to extract a set of line-segment primitives which act as actual input to the image-parsing algorithms; see Figure 1.2. (Chapter 5 includes a full description of the actual experimental setup and preprocessing method used.) While performance of this recognizer has not been validated on a large handprint data set, it seems quite promising.

My earlier joint work on these topics is documented in [2] with E.Bienenstock and S.Geman, and especially [11] with S.Geman and Z.Chi. In [11] a rigorous analytic framework

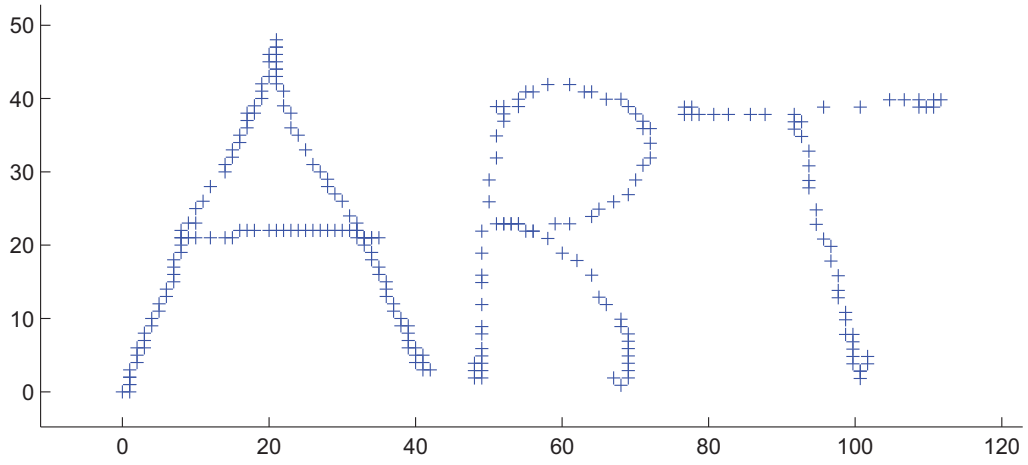

Figure 1.1: *Example of the type of online handprint data that can be recognized.*

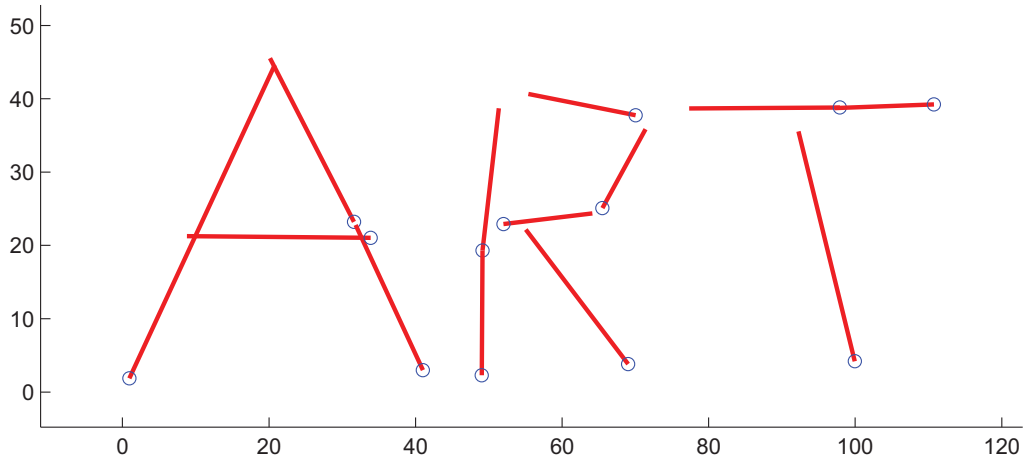

Figure 1.2: *Preprocessed version of online handprint data in Figure 1.1.*

for compositional distributions on objects and their extension to scenes of objects is laid out. The content of this thesis thus revisits and expands upon many of the ideas presented in [2] and [11].

The next section of this chapter gives a brief introduction to syntactic pattern recognition, then the remaining sections of this chapter give an overview of the thesis.

1.1 Syntactic Pattern Recognition

In general, syntactic approaches to pattern recognition model objects (or equivalently patterns) in a reductionist fashion; complex objects are defined in terms of their constituents, the constituents in turn are defined in terms of their subconstituents, and so on and so forth. At the bottom of this process lies a set of terminals (i.e., atoms) for which no further expansion is possible. This leads to hierarchical descriptions of the objects of interest.

The following two examples are illustrative: First, in a geometric setting, an automobile is composed of a body and four wheels. The body is composed of windows and doors, each door is composed of handles and door locks, etc. Each wheel is composed of a tire and a hub-cap... etc. At the bottom, the terminals could represent pixels, or perhaps polygonal faces. Second, in a linguistic setting, a complete sentence, for example,

> The bird flies swiftly.

may be modeled as being composed of a noun phrase and a verb phrase, where the noun phrase in turn is composed of a determiner ("The") and a noun ("bird") and the verb phrase in turn is composed of a verb ("flies") and an adverb ("swiftly").

Each object is defined/represented by a **labeled tree**. These trees have nodes labeled with the names of each of their constituents and perhaps other facts. The topology of these trees record the hierarchical nature of each object's definition. Figure 1.3 contains a labeled tree for the above sentence example.

Typically, the set of "allowed" or "possible" objects is determined by a set of rules called a **grammar**. There are in general many possible ways for the rules of a grammar to be expressed. One popular method is in terms of string rewriting rules called productions.

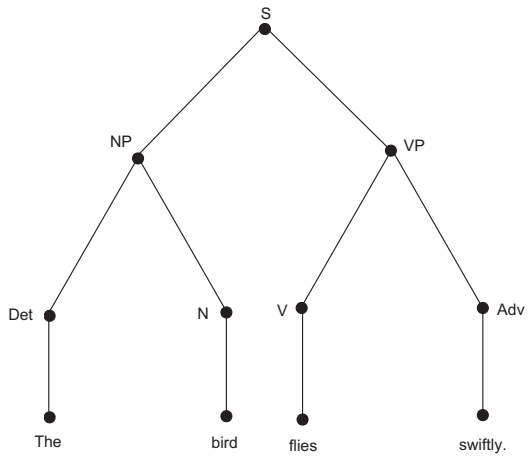

Figure 1.3: *A sentence object.*

Grammars which use these types of rules were studied and categorized by Chomsky in the 1950s, [6].

Many earlier users of syntactic approaches to pattern recognition, including K. Fu (see his [8] for a lucid overview) and A. Shaw [18], used these types of grammars as their primary pattern modeling tools.

More recently, constraint-based grammar formalisms have been examined and studied, especially in linguistics. Constraint-based grammars are defined by logical formulae which control whether several objects may come together to form another object. See S. Shieber [19] for an introduction. *The grammar formalisms developed in this thesis are in part constraint-based.*

When the leaves (i.e., terminal node labels) of objects are pixels, the leaves of any particular object define an image of that object. When the leaves of objects are words, the leaves of any particular object define a string of words. Given a particular grammar and a collection of terminals, a natural question to ask is whether or not there is an object defined by the grammar which could have generated this collection of terminals. More generally, one might want to know which objects could have produced the collection of terminals. The process of determining which objects could have generated a particular collection of terminals is known as **parsing**.

Parsing is the basis of pattern recognition for syntactic methods. Typically, a set of raw data is preprocessed into a collection of terminals; these terminals are then parsed. Output of the recognizer is a list of objects which could have generated the collection of terminals.

In practice there may be many different sets of objects which can generate the same collection of terminals. Several factors may contribute to this situation. Multiple objects may describe the same collection of terminals. The raw data may be corrupted by noise. And/or the collection of terminals to be processed may be derived from a scene consisting of several objects.

In many geometric applications, such as the one that will be developed shortly, a further complicating factor is a lack of an a priori meaningful concatenation order in the data to be processed. This tends to dramatically increase the number of possible interpretations of the data while at the same time excluding the use of the standard chart-/table-based parsing methods. An example of the situation is as follows: Imagine a grammar which defines lines in terms of collinear sets of points. Given any particular set A of N collinear points, there are $2^N - N - 1$ possible lines which generate at least a portion of A.

When multiple sets of objects can explain (i.e., generate) the same collection of terminals, there must be some way of comparing, ranking, or otherwise making useful these different interpretations of the data. One natural suggestion is to define a probability distribution on objects.[1] Classically, probabilistic context-free grammars (PCFGs) have been proposed for this task; here a context-free grammar is used to define the set of possible objects and production probabilities are used to define a probability distribution on these objects; see for example [3]. *A similar scheme will be employed in this thesis; first, a "compositional" grammar will be used to define a set of objects and then probabilities associated with the rules of the grammars will be used to assign a "compositional" probability distribution to the overall set of objects.*

[1] In fact, there may be so many interpretations of the terminals that an exhaustive enumeration of them all is not computationally tractable. In this case a probability measure on objects may be a useful tool for developing parsing algorithms which return only the more likely interpretations of the terminals.

1.2 Compositional Grammars

The grammars used in this thesis make use of *composition rules* and *relation functions* to determine when and how objects may come together as the subconstituents of a new object. The action of these composition rules enforces certain constraints on each object's subconstituents. Relation functions provide additional information on how subconstituents of an object are related.

If the automobile object above were defined with a set of compositional rules, the rules would include the facts that an automobile is composed of body and wheels, doors are composed of door handles and door locks, and tires are composed of tires and hubcaps. Furthermore, they, or the relation functions, ought to include the geometric dependencies of the situation. A car has wheels that are affixed to the car body in the appropriate locations, wheels have tires with hubcaps in their center, and car doors have door handles and door locks which enjoy a particular sort of geometric relationship.

A set of compositional rules for the sentence example must record similar facts – that a sentence can be composed of a noun phrase and a verb phrase and that a noun phrase can be composed of a determiner and a noun, etc. Additional constraints may also be desirable. For example, one might want to enforce agreement on number between sentence constituents. In this case,

The birds flies swiftly.

would not be an allowed sentence.

The use of composition rules and relation functions seem to be a quite natural way to define objects, particularly since they allow one easily to model dependencies between object constituents, such as their relative positions and/or content.

In order to facilitate subsequent discussion of objects and trees, the following notation and definitions will be adopted. Θ will represent the set of labeled, ordered trees. A labeled tree $\omega = l(\alpha, \beta)$ has a root node with label l; the left daughter of the root node is the labeled tree α and the right daughter of the root node is labeled tree β. Let ω represent

the sentence object in Figure 1.3,

$$\omega = S(NP(Det(The), N(bird)), VP(V(flies), Adv(swiftly.)))$$

The left daughter of ω is

$$NP(Det(The), N(bird))$$

The right daughter of ω is

$$VP(V(flies), Adv(swiftly.))$$

.

In general, the Greek letters, α, β, and ω will be used to represent labeled trees. Starred Greek letters α^*, β^*, and ν^* will represent ordered sets of labeled trees (or, equivalently, strings of labeled trees).

A variety of functions will be defined on labeled trees. Three of the most frequently used will be the root label function $L(\omega)$, the yield function $Y(\omega)$, and the leaf set function $\kappa(\omega)$. These functions are defined as follows: For $\omega = l(\alpha^*)$, $L(\omega) = l$; for a single node tree $\omega = t$, $L(\omega) = t$. The yield of a tree $Y(\omega)$ is the value of its terminal node labels taken in left to right order. For $\omega = 1(1(a, a), b)$, $Y(\omega) = (a, a, b)$. The leaf set of a tree $\kappa(\omega)$ is the set of its leaf node labels; for $\omega = 1(1(a, a), b)$, $\kappa(\omega) = \{a, b\}$.

Formally, each **composition rule** will have a unique label l associated with it – thus we will often refer to a composition rule by use of its label. Operation of each composition rule is defined by a binding function B_l and a set of allowed binding function values S_l. Objects (i.e., trees) $\alpha^* = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ are allowed to come together to form a new object $l(\alpha^*)$ if $B_l(\alpha^*) \in S_l$. Any function from labeled trees to some arbitrary range space can act as a binding function. Thus B_l is a mapping from finite strings of trees $\Theta^* \equiv \cup_{n=1}^{\infty} \Theta^n$ to some arbitrary range space; call it R_l.

$$B_l : \Theta^* \rightarrow R_l$$

This formulation of "composition rule" is most general in that it allows one to enforce *any* constraint on the constituents of an object.

An example of a geometric composition rule is as follows:[2]

Example 1 *L-junction.*

Assume trees with root label 1 *represent lines. Define the binding function* B_2 *as follows: For lines* α *and* β*, define* $B_2(\alpha, \beta)$ *as the relative angle, length, and position of the endpoints of* β *with respect to* α*,*

$$B_2(\alpha, \beta) = (\theta_\beta - \theta_\alpha, \frac{r_\beta}{r_\alpha}, R_{-\theta_\alpha} \frac{x_\beta - x_\alpha}{r_\alpha}), \quad L(\alpha) = L(\beta) = 1$$

where $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$*. For other possible arguments of* B_2*,* $\alpha^* \in \Theta^* - \{\omega \in \Theta : L(\omega) = 1\}^2$*, define* $B_2(\alpha^*) = 0$*. Let the set of allowed binding function values be* $S_2 = [3\pi/8, 5\pi/8] \times [.9, 1.1] \times \{(x, y) : \sqrt{x^2 + y^2} < .1\}$ *Lines* α *and* β *may come together to form* $2(\alpha, \beta)$ *only if they form an approximately 90 degree angle, are of approximately equal lengths, and have endpoints which lie approximately next to one another.*

Figure 1.4 depicts an example composition (c), of line α *(a), and line* β *(b). Here each line is itself composed of several line segments.*

The binding function in the above example enforces a dependency between constituents of $2(\alpha, \beta)$, since for any line α there is a very restricted class of lines β with which α may compose to form $2(\alpha, \beta)$.

A **compositional grammar** $C = (T, N, \{B_l, S_l\}_{l \in N})$ is defined by a set of terminals T and a set of composition rules $\{B_l, S_l\}_{l \in N}$ indexed by N. The **set of objects** Ω defined by such a grammar is the closure of the composition rules applied to the terminals. Elements of Ω are trees with leaf labels in T and interior labels in N. A simple (albeit abstract) compositional grammar is defined in the following example.

Example 2 *No Overlap.*

Let $T = \{a, b, c\}$*,* $N = \{1\}$*. Define the binding function* B_1 *associated with composition*

[2]The particular form of binding function B_l used in this example defines the action of this composition rule in a scale-, translation-, and rotation-invariant manner, in the sense that if T is an element of the scale, translation and rotation group $SE(2) \times R_+$, then $B_l(T\alpha, T\beta) = B_l(\alpha, \beta)$. This type of relative encoding of the geometric parameters of β with respect to α will appear frequently.

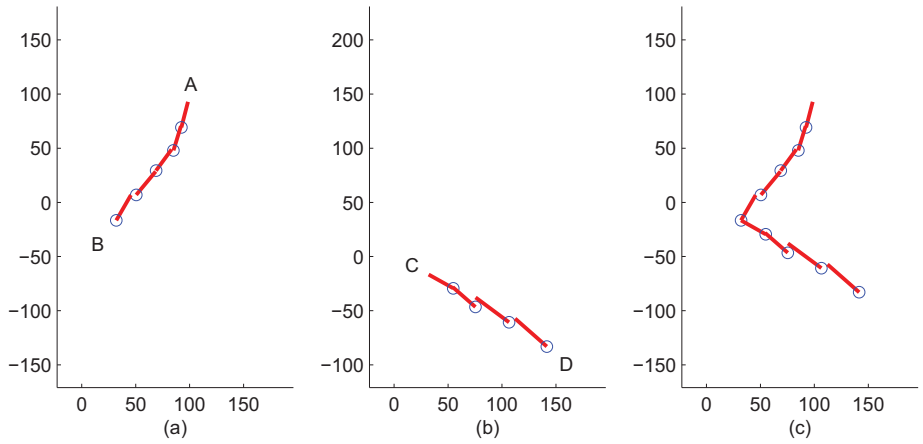

Figure 1.4: *L-junction Formation. (a) Line α with endpoints AB, $\theta_\alpha = <AB$, $r_\alpha = \|AB\|$, $x_\alpha = B$. (b) Line β with endpoints CD, $\theta_\beta = <CD, r_\alpha = \|CD\|, x_\beta = C$. (c) L-junction $2(\alpha, \beta)$.*

rule 1 as

$$B_1(\alpha, \beta) = \begin{cases} 1, & \kappa(\alpha) \cap \kappa(\beta) = \phi \\ 0, & else \end{cases}$$

with a set of allowed binding function values $S_1 = \{1\}$. (Here, and in the future, definitions of composition rules given explicitly only on some subset A of Θ^* will be assumed to extend to all of Θ^* by a mechanism which forbids composition of any elements in $\Theta^* - A$.) When $\kappa(\alpha) \cap \kappa(\beta) = \phi$ the leaves of tree α are all distinct from those of tree β. There are 21 trees in the space of objects Ω defined by this composition system:

$$\Omega = \left\{ \begin{array}{cccccc} a & b & c & & & \\ 1(a,b) & 1(a,c) & 1(b,a) & 1(b,c) & 1(c,a) & 1(c,b) \\ 1(1(a,b),c) & 1(1(a,c),b) & 1(1(b,a),c) & 1(1(b,c),a) & 1(1(c,a),b) & 1(1(c,b),a) \\ 1(c,1(a,b)) & 1(b,1(a,c)) & 1(c,1(b,a)) & 1(a,1(b,c)) & 1(b,1(c,a)) & 1(a,1(c,b)) \end{array} \right\}$$

Of course the objects in this example are of a very abstract nature. More practically, in

a geometric or imaging setting, one might want the set of terminals T to represent points, pixels, edges, or perhaps surface patches in R^2 or R^3. These "basic" building blocks along with a set of composition rules and relation functions can then be used to define a very large and complex set of objects Ω. In a linguistic setting T could be a list of words (a lexicon – possibly with associated word attributes) or some even finer component of speech such as phonemes or morphemes.

In the next section, when a probability distribution on objects is introduced, and later in the thesis, various computational difficulties will arise when objects are defined solely in terms of composition rules. In an effort to side-step some of these issues, the use of relation functions is introduced. These functions are used to model certain types of object constituent dependencies in an essentially independent (or context-free) manner. The general idea is to use labeled trees to represent sets of objects, as well as individual objects. For example, instead of ω representing a particular geometric object, ω can now represent that object up to scale, translation and rotation. The set of objects represented by a particular composition $\omega = l(\alpha, \beta)$ is determined by the sets of objects represented by its daughters α and β and a relation function associated with label l. As a notational convenience, the primed Greek letters will denote trees which are utilized to denote sets of objects in this way.

Formally, the set of objects represented by a particular object ω' will defined in a bottom-up manner through use of an **interpretation function** I defined as follows: Let $I(\alpha')$ and $I(\beta')$ be sets of objects represented by trees α' and β'. For $l'(\alpha', \beta')$ with a vector-valued root label $l' = (l, v)$ the set of objects represented by $l'(\alpha', \beta')$ is

$$I(l'(\alpha', \beta')) = \{l(\alpha, \beta) | (\alpha, \beta) \in I(\alpha') \times I(\beta'), f_l(\alpha, \beta) = v\} \tag{1.1}$$

where f_l is a **relation function** defined on $I(\alpha') \times I(\beta')$.

The following example shows how the composition rule in Example (1) can be re-encoded using this approach.

Example 3 *L-junction. Let $I(\alpha')$ and $I(\beta')$ each be orbits[3] of lines under the scale, trans-*

[3]When a tree ω' is used to represent an object up to the action of a group, it represents the **orbit** of

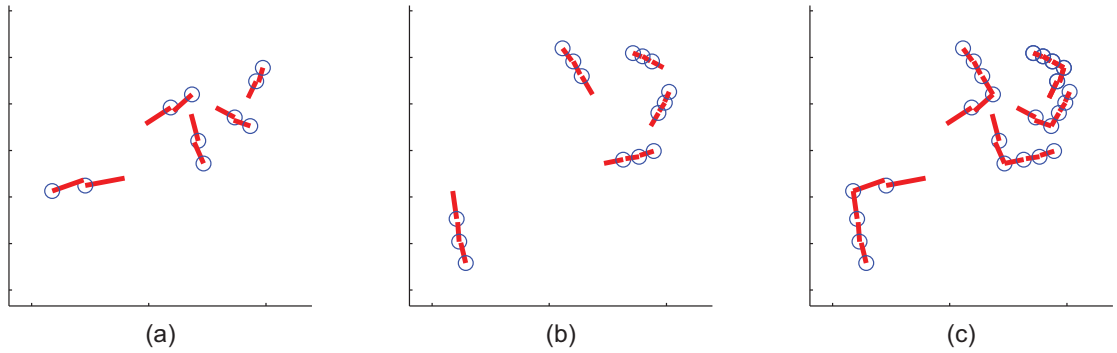

Figure 1.5: *(a) Some elements of $I(\alpha')$ (b) Some elements of $I(\beta')$ (c) Some elements of $I(l'(\alpha', \beta'))$.*

lation and rotation group. Define the relation function $f_l(\alpha, \beta)$ as the relative angle, length and end-point position of β with respect to α,

$$f_l(\alpha, \beta) = (\theta_\beta - \theta_\alpha, \frac{r_\beta}{r_\alpha}, R_{-\theta_\alpha} \frac{x_\beta - x_\alpha}{r_\alpha})$$

For $l' = (l, v)$ and $v = (\pi/2, 1, (0, 0))$ the set of objects $I(l'(\alpha', \beta'))$ is the set of all 90 degree L-junctions with equal length sides that can be formed between elements of $I(\alpha')$ and $I(\beta')$; see Figure 1.5. Varying the components of v allows definition of a flexible class of L-junction.

Here there are no dependencies between the constituents of $l'(\alpha', \beta')$ since any two trees α', β' which represent line orbits can be composed to represent an orbit of L-junctions $l'(\alpha', \beta')$.

An **interpreted compositional grammar** C' is defined by a collection of relation functions and allowed relation function values $\{f_l, V_l\}$ and a set of terminal element interpretations $\{I(t)\}_{t \in T}$ in addition to the usual components of a compositional grammar:

$$C' \equiv (T, N, \{B_l, S_l, f_l, V_l\}_{l \in N}, \{I(t)\}_{t \in T})$$

an object. The current example uses a relation function to combine two orbits $I(\alpha')$ and $I(\beta')$ into a third orbit $I(l'(\alpha', \beta'))$.

Operation of composition rules in an interpreted compositional grammar (ICG) is similar to that in a non-interpreted compositional grammar, except that a string of trees α'^* may now be composed to form any tree in $\{l'(\alpha'^*)|l' \in l \times V_l\}$ if $B_l(\alpha'^*) \in S_l$. The symbol Ω' will denote the set of trees formed by the closure of the composition rules applied to the terminals. The interpretation of each composition $l'(\alpha'^*) \in \Omega'$ will be defined recursively as

$$I(l'(\alpha'^*)) \equiv \{l(\alpha^*)|\alpha^* \in I(\alpha'_1) \times \cdots \times I(\alpha'_n), f_l(\alpha^*) = v\}$$

for $l' = (l, v)$ and $\alpha'^* = (\alpha'_1, \cdots, \alpha'_n)$. This is a direct generalization of (1.1).

Example 4 *ICG for Lines and L-junctions.*
Let T={0}, N={1,2}. Terminal objects with label 0 *will represent line-segment primitives.* $I(0) = R^2 \times R^2$. *Define the sets of allowed relation function values*

$$V_1 = [-\frac{\pi}{8}, \frac{\pi}{8}] \times [.8, 1.2] \times [.9, 1.1] \times [-.1, .1]$$

and

$$V_2 = V_1 + (\frac{\pi}{2}, 0, 0, 0)$$

Objects with root labels in $1 \times V_1$ *will represent lines. These are built up recursively from other line representations and line segment primitives.*

$$B_1(\alpha', \beta') = \begin{cases} 1 & L(\alpha'), L(\beta') \in \{0\} \cup (1 \times V_1) \\ 0 & else \end{cases}$$

with $S_1 = \{1\}$. *Elements of* $I((1, v)(\alpha', \beta'))$ *are determined via*

$$f_1(\alpha, \beta) = (\theta_{\beta_-} - \theta_{\alpha_+}, \frac{r_{\beta_-}}{r_{\alpha_+}}, R_{-\theta_{\alpha_+}} \frac{x_{\beta_-} - x_{\alpha_+}}{r_{\alpha_+}})$$

where α_+ *is the rightmost terminal of* α, $\alpha_+ = t_n$ *for* $(t_1, \cdots, t_n) = Y(\alpha)$, *and* β_- *is the leftmost terminal of* β, $\beta_- = t_1$, *for* $(t_1, \cdots, t_n) = Y(\beta)$. *see Figure 1.6.*[4]

[4]Many other line relation functions could be used, for example instead of encoding α and β relative to the endpoints of α_+ and β_- one could encode α and β relative to the overall endpoints of α and β, however in this case the set of lines generated would not in general have slowly varying line-segment dimensions.

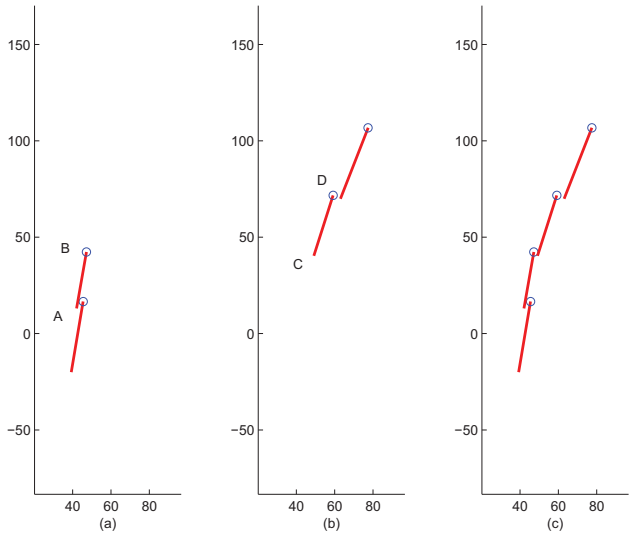

Figure 1.6: *Line formation. (a) Line α, with α_+ endpoints AB. (b) Line β, with β_- endpoints CD. (c) Line $1(\alpha, \beta)$.*

Objects with root labels in $2 \times V_2$ will represent L-junctions. These objects are composed of pairs of lines therefore, $B_2(\alpha', \beta') = B_1(\alpha', \beta')$ and $S_2 = \{1\}$. Elements of $I((2, v)(\alpha', \beta'))$ are determined by encoding the overall endpoints of β relative to the endpoints of α.

$$f_1(\alpha, \beta) = f_2(\alpha, \beta) = (\theta_\beta - \theta_\alpha, \frac{r_\beta}{r_\alpha}, R_{-\theta_\alpha} \frac{x_\beta - x_\alpha}{r_\alpha})$$

See Figure 1.4.

1.3 Compositional Probability Distributions

Compositional grammars allow large sets of flexible yet coherent patterns to be defined. Specifying a probability distribution on these patterns is of fundamental importance if they are to be used as the basis for a statistically based approach to pattern recognition. For use as a prior, these distributions should agree with the underlying "real-world" statistics of the situation.

In this thesis, probability distributions on objects are specified which allow one to explic-

itly control the statistics salient to any given composition. This is done by specifying a set of external probability measures on label and binding function values and then extending these measures to an overall distribution on objects. The marginals on labels and binding function values of this overall distribution match the external measures.

Given only marginal distribution constraints, there may be an infinity of possible distributions on objects which agree with these constraints. Therefore some additional constraints on the form of the overall distribution are required to make it unique. One approach would be to use maximum entropy [14]; here we take another, as follows:

The overall distribution on objects is specified as the solution of a system of equations. Formally, let $C = (T, N, \{B_l, S_l\}_{l \in N})$ define a compositional grammar with T discrete, N discrete, and dyadic binding functions B_l. The probability of a composition $\omega = l(\alpha, \beta) \in \Omega$ is defined in terms of the probability of the daughter trees α and β, and *external measures* Q on labels $T \cup N$, and $\{Q_l\}_{l \in N}$ on allowed binding function values $\{S_l\}_{l \in N}$:

$$P(\omega) = \begin{cases} Q(l)Q_l(B_l(\alpha,\beta))\frac{P(\alpha)P(\beta)}{P \times P(B_l(\alpha,\beta))} & \omega = l(\alpha,\beta) \\ Q(t) & \omega = t \in T \end{cases} \tag{1.2}$$

where

$$P \times P(B_l(\alpha,\beta)) \equiv \sum_{(\eta,\xi) \ni B_l(\eta,\xi) = B_l(\alpha,\beta)} P(\eta)P(\xi) \tag{1.3}$$

Any non-negative P which satisfies (1.2) defines a **compositional probability distribution** on objects Ω. It is a simple exercise to check that such a P defines a probability measure and that the marginals $P(l, v) \equiv P(\omega : \omega = l(\alpha, \beta), B_l(\alpha, \beta) = v)$ match $Q(l)Q_l(v)$.

From a coding perspective, the definition of P has a clear interpretation. A composition ω may be transmitted (using Shannon Codes) first by sending root label l and binding function value $B_l(\alpha, \beta)$; this uses $-\log Q(l)Q_l(B_l(\alpha, \beta))$ bits.[5] The constituents of ω may now be transmitted in the context of $B_l(\alpha, \beta)$; this takes $-\log \frac{P(\alpha)P(\beta)}{P \times P(B_l(\alpha,\beta))}$ bits.

In the following example, a compositional distribution is defined based on the no overlap composition system defined in Example 2.

[5]For a source Ω with distribution P a Shannon Code for Ω has codewords $c(\omega)$, $\omega \in \Omega$, with lengths $-\log P(\omega)$.

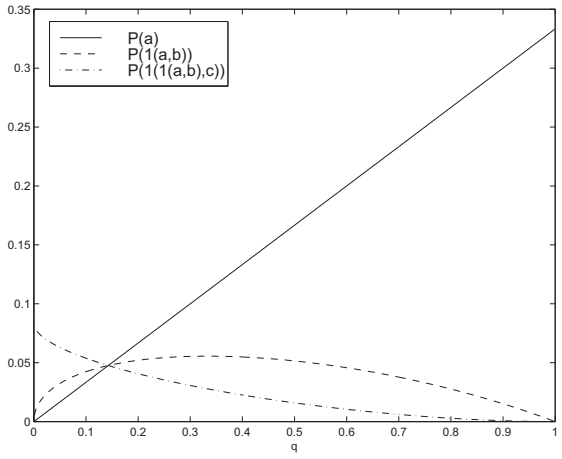

Figure 1.7: *Compositional no overlap distribution.*

Example 5 *No Overlap Distribution. Define Q for Example 2 via the following: Fix $q \in [0,1]$, let $Q(a) = Q(b) = Q(c) = q/3$ and $Q(1) = p = 1 - q$. From Equation (1.2), for $x = Q(1)/P \times P(B_1)$, the probability of any two-leaf tree in Ω is $(q/3)^2 x$, the probability of any three-leaf tree in Ω is $(q/3)^3 x^2$. Summing over these elements gives*

$$12(q/3)^3 x^2 + 6(q/3)^2 x = p$$

This equation has a unique positive solution, therefore the compositional distribution on Ω is well defined.

Figure (1.7) plots the probability of one-, two- and three-leaf trees for various values of q. Notice that these distributions are NOT in general maximum entropy distributions under the constraint $P(\omega : \omega \notin T) = p$ and $P(a) = P(b) = P(c) = q/3$.

Unfortunately, two issues enter to complicate the picture. First, the system of equations in (1.2) may not be sufficient to guarantee P is well defined: in some cases there may not be any P which satisfies them; in others there may be more than one. Second, even when P is well-defined, the values of the denominators $P \times P(B_l)$ in (1.2) may be very difficult to determine due to the generality of B_l. These values are necessary in order to compute actual object probabilities.

At present not a great deal is known about conditions for (1.2), or its generalizations which ensure P is well-defined. Section 2.2 contains further discussion on this topic, including examples of systems with non-unique and non-existent compositional distributions.

In some special cases the values of $P \times P(B_l)$ may be computed exactly. In others it may be possible to approximate them (see Geman [10] for some recent work in this regard), or it may be possible to estimate them using the Monte Carlo algorithm for sampling proposed in Chapter 3.

Compositional measures on interpreted compositional grammars are defined similarly except that here Q defines a measure on tree node labels

$$T \cup \{(l, v) : l \in N, v \in V_l\}$$

Any non-negative solution P' of

$$P'(\omega') = \begin{cases} Q(l,v)Q_l(B_l(\alpha',\beta'))\frac{P'(\alpha')P'(\beta')}{P'\times P'(B_l(\alpha',\beta'))} & \omega' = l'(\alpha',\beta'),\ l' = (l,v) \\ Q(t) & \omega' = t \in T \end{cases} \tag{1.4}$$

where

$$P' \times P'(B_l(\alpha',\beta')) \equiv \sum_{(\eta',\xi') \ni B_l(\eta',\xi') = B_l(\alpha',\beta')} P'(\eta')P'(\xi') \tag{1.5}$$

defines a compositional measure on the space of objects Ω'.

In practice this measure P' is extended from Ω' to a measure P'' on individual objects in $\Omega'' = \cup_{\omega' \in \Omega'} I(\omega')$ by defining a placement function g with range X and a conditional probability measure K on placement function values, such that $g : I(\omega') \to X$ is 1-1, and $K(g(I(\omega'))|\omega') = 1$. For $\omega \in I(\omega')$,

$$P''(\omega) \equiv P'(\omega')K(g(\omega)|\omega') \tag{1.6}$$

The coding interpretation here is clear. An object ω can be transmitted by sending a description of the object's orbit, using $\log P'(\omega')$ bits, and additional placement information, using $\log K(g(\omega)|\omega')$ bits.

Below a compositional distribution is defined on the previous ICG example and extended

to a measure on Ω''. Here the spaces of objects Ω' and Ω'' are non-discrete, therefore the definitions for P' and P'' in (1.4) and (1.6) do not directly apply. Rather, these definitions must be extended to this setting. (This is done formally in Section 2.1.)

Example 6 *ICG Lines and L-junctions continued.*
Using the ICG from Example 4, define an external probability Q on labels $\{0\} \cup 1 \times V_1 \cup 2 \times V_2$ as follows: For $q, p_1, p_2 \in [0,1]$ such that $q + p_1 + p_2 = 1$, let $Q(0) = q$ and for $E \subseteq (1 \times V_1) \cup (2 \times V_2)$

$$Q(E) = p_1 \int_{(1,v) \in E} G_1(v)dv + p_2 \int_{(2,v) \in E} G_2(v)dv$$

where G_1 and G_2 are appropriately normalized Gaussian densities evaluated on the components of v, e.g.

$$G_1(v) = \frac{1}{Z_1} exp(\theta^2/\sigma_1 + (r-1)^2/\sigma_2 + (x-1)^2/\sigma_3 + y^2/\sigma_4), \quad v = (\theta, r, x, y)$$

$$G_2(v) = \frac{1}{Z_2} exp((\theta - \frac{\pi}{2})^2/\sigma_1 + (r-1)^2/\sigma_2 + (x-1)^2/\sigma_3 + y^2/\sigma_4), \quad v = (\theta, r, x, y)$$

Since $S_1 = \{1\}$, $Q_l(x) = 1$ for $x = 1$, and 0 otherwise. $Q_2(x) = Q_1(x)$.

Extend this distribution by defining $g(\omega)$ as the endpoints of ω and defining K uniform in the region $[0,d]^4$.

The examples in Figures 1.4, 1.5, and 1.6 were sampled from a distribution similar to this one.

Work related to compositional and interpreted compositional grammars seems to be based either in the area of constraint-based grammars, as previously referenced, and also in the area of attribute grammars [16]. The current approach represents a fusion between the two in the sense that constraint-based grammars typically utilize binding predicates whereas here we allow the use of more general binding functions. And attribute grammars, while utilizing production-based context-free grammars, allow the evaluation of complex attribute functions defined on the underlying tree structures produced by the context-free grammar.

1.4 Object Recognition and Scene Interpretation

The recognition paradigm developed in this thesis is illustrated through the design of an Online Handprint Recognition System (OHRS). The general strategy is to use an interpreted compositional grammar and distribution (akin to Example 4 and Example 6) to model the objects of interest; recognition consists of parsing collections of terminals into sets of objects which could have generated these terminals.

The OHRS grammar uses line segments as terminals and defines a hierarchy of compositions for representation of lines, arcs, letter fragments, whole letters, strings of letters, and whole words. Here, the initial input to the system is a set of digitized stylus coordinate data indexed by time, as in Figure 1.1. This data is transformed into a collection of line segment primitives (terminals) by a simple preprocessing step, as in Figure 1.2.

It is assumed that the collection of terminals to be processed may have been derived from a scene consisting of an unknown but finite number of objects. Thus, the general recognition problem is one of *scene interpretation*. For example, one does not know *a priori* how many words, or isolated characters may appear as input to the OHRS.

Analytically, given a grammar which defines a set of objects Ω and a collection of terminals D, any finite subset s of Ω for which $\cup_{\omega \in s} \kappa(\omega) = D$ and $\forall \alpha, \beta \in s, \kappa(\alpha) \cap \kappa(\beta) = \phi$ defines a **scene** of non-overlapping objects which could have generated the collection of terminals. The set of all such scenes which could have generated D is defined as

$$S|D = \{s| \cup_{\omega \in s} \kappa(\omega) = D, |s| < \infty, s \subseteq \Omega, \kappa(\alpha) \cap \kappa(\beta) = \phi, \alpha \neq \beta\}$$

In general for any given D there may be many scenes in $S|D$ which explain (could have generated) D.

For example, there are many, many possible scenes of objects which could have generated even the relatively simple set of primitives in Figure 1.2. One scene is simply a collection of line segment primitives. This seems a very unlikely explanation of the data since the "happenstance" probability of laying out 13 iid line segments so that they spell the word "ART" is quite small. Other scenes include various combinations of line and arc compositions and line segment primitives. Still others include various letter compositions.

The most reasonable explanation of the data is in fact a composition for the word "ART". Bienenstock [2] contains an interesting discussion of Laplace's definition of probability [17] and this viewpoint.

Any probability distribution P on discrete objects can be extended to scenes in a straightforward manner. For a fixed collection of terminals D the probability measure P_D on $S|D$ is defined as

$$P_D(s) = \frac{1}{Z_D} \Pi_{\omega \in s} P(\omega)$$

where $Z_D = \sum_{s \in S|D} \Pi_{\omega \in s} P(\omega)$ (here Z_D is fixed over all possible scene interpretations for a particular collection of terminals, therefore this value is never needed explicitly). This probability measure is used for determining which scenes in $S|D$ are more likely than others to have generated D. It is an *a posteriori* distribution on possible interpretations of the data.[6] Geman [11] contains more information on scene distributions and scene densities.

A most likely interpretation of a collection of terminals is given by the maximum *a posteriori* (MAP) estimate

$$s^* = \arg \max_{s \in S|D} P_D(s) \tag{1.7}$$

For all but the simplest cases, determination of this quantity by exhaustive search is not possible. Therefore a greedy approximation to (1.7) is developed.

From a coding perspective interpretation, a solution of (1.7) represents the determination of a shortest possible set of code words for transmitting a particular collection of terminals D. Thus, in the current setting, recognition and compression can be viewed as synonymous activities.

1.5 Preview of the Thesis

Chapter 1 provides an introduction to the thesis contents.

The first section of Chapter 2 lays out the analytic framework for defining interpreted

[6] $P_D(s) \propto P_S(s) P_L(D|s)$ where P_S is a prior on scenes $P_S(s) \propto \Pi_{\omega \in s} P(\omega)$ and P_L is a degenerate likelihood model

$$P_L(D|s) = \begin{cases} 1 & \cup_{\omega \in s} \kappa(\omega) = D \\ 0 & else \end{cases}$$

compositional distributions P' and their extension to objects P'' in a more general setting which includes both polyadic composition rules and continuous spaces of objects. The next section is devoted to the issue of the existence and uniqueness for compositional systems. It includes examples of non-unique and non-existent compositional distributions. The last section of Chapter 2 examines computation of "encoding wins", or equivalently log likelihood ratios, between encodings for pairs of objects (α, β) under the assumption that they are a part of a composition $l(\alpha, \beta)$ versus an independence hypothesis. These types of calculations are used in the MAP approximation algorithms developed in Chapter 4.

Chapter 3 describes a simple Monte Carlo approach for sampling certain types of compositional distributions and interpreted compositional distributions.

Chapter 4 describes a collection of "image-parsing" algorithms for pattern recognition/scene interpretation. Here, an initial hopelessly brute-force approach is presented and then modified in a variety of ways in order to make it more computationally tractable.

Chapter 5 is devoted to experimental results. First the general mathematical setup used in the OHRS application is described. Then a variety of sampling results are presented. The chapter concludes with a section on the application's performance on a small hand-print data set.

A summary of some of the thesis results and suggestions for future directions of research are contained in Chapter 6.

Chapter 2

Analytic Framework

The material in this chapter is divided into three sections. The first section, Compositional Measures, generalizes the probability distributions introduced in Chapter 1 by extending them to both polyadic and non-discrete domains.

The second section, Existence and Uniqueness, demonstrates two cases in which the compositional measure on a set of objects is not well defined. In one the measure is not uniquely defined; in the other, it simply does not exist. An obvious condition which guarantees existence and uniqueness of the measure is given.

The third section, Encodings Wins and Likelihood Ratios, shows how likelihood ratios may be used to compute the relative encoding win between a composition and its constituents. (In Chapter 4, this quantity will be used in the development of a greedy approximation of the MAP scene interpretation problem.)

2.1 Compositional Measures

Chapter 1 introduced the notion of compositional probability distributions. The definitions for P and for P' assumed the use of dyadic binding functions and discrete node labels. Here both of these restrictions are relaxed.

For polyadic and varadic binding functions, and discrete N and T, any non-negative solution P to the following set of equations defines a compositional probability distribution:

$$P(\omega) = \begin{cases} Q(l)Q_l(B_l(\alpha^*))\frac{P^*(\alpha^*)}{P^*(B_l(\alpha^*))} & \omega = l(\alpha^*) \\ Q(t) & \omega = t \in T \end{cases} \tag{2.1}$$

where P^* is the star measure on Ω^*.[1] This definition is the same as the earlier one for P when the compositions in Ω are all binary.

Assuming $\{V_l\}_{l \in N}$ are also discrete, interpreted compositional probability distributions

[1] Given a measure P on Ω, define the **star measure** P^* on Ω^* through the sum of the n-fold product measures P^n on product spaces Ω^n, $P^*(A) \equiv \sum_{n=1}^{\infty} P^n(A \cap \Omega^n)$.

are defined as non-negative solutions of

$$P'(\omega) = \begin{cases} Q(l,v)Q_l(B_l(\alpha^*))\frac{P'^*(\alpha'^*)}{P^*(B_l(\alpha'^*))} & \omega = (l,v)(\alpha'^*) \\ Q(t) & \omega = t \in T \end{cases} \tag{2.2}$$

where P'^* is a star measure on Ω'^*.

Use of a discrete set of node labels ensures that the space of objects defined by a compositional grammar is also discrete.[2] More generally, use of non-discrete labels will be useful for modeling continuous phenomena (such as geometric relations); in this case, the space of objects will be non-discrete. The definitions for P and P' extend naturally to this setting.

Formally, following [11], assume N is discrete and T is possibly non-discrete. For P a measure on Ω, and P^* the star measure on Ω^*, use the measure P_l^* defined by

$$P_l^*(S) \equiv P^*(\alpha^* : B_l(\alpha^*) \in S)$$

to define a measure μ^* on Ω^*

$$\mu^*(A) \equiv \int_{\alpha^* \in A} \frac{dQ_l}{dP_l^*}(B_l(\alpha^*))dP^*(\alpha^*)$$

where $\frac{dQ_l}{dP_l^*}$ is the Radon-Nikodym derivative of Q_l with respect to P_l^*. If Q_l is absolutely continuous with respect to P_l^*, for all $l \in N$, then

$$\nu(E) = Q(E \cap T) + \int_{l(\alpha^*) \in E} Q(l)\frac{dQ_l}{dP_l^*}(B_l(\alpha^*))dP^*(\alpha^*)$$

defines a measure on Ω. When P equals ν, P is a compositional measure.

The new setup developed here for interpreted compositional measures is analogous. Q defines a measure on $T \cup (\cup_{l \in N} l \times V_l)$. N is assumed discrete, T and $\{V_l\}_{l \in N}$ are possibly

[2]Since by definition trees in Ω are finite, there are a countable number of tree topologies in Ω, and to each of these there are a countable number of possible labelings.

non-discrete. The overall measure on Ω' is given by

$$\nu'(E) = Q(E \cap T) + \int_{(l,v)(\alpha^*) \in E} dQ(l,v) \frac{dQ_l}{dP'^*_l}(B_l(\alpha^*)) dP'^*(\alpha^*)$$

When P' equals ν', P' is a interpreted compositional measure.

These definitions presuppose existence of appropriate $\sigma-$algebras on Ω and Ω'. In [11] a $\sigma-$algebra on Ω for discrete nonterminal node labels N and possibly non-discrete terminals T is developed. Here this result is extended to define a $\sigma-$algebra on Ω' for possibly non-discrete relation function values $\{V_l\}_{l \in N}$. As in [11] we will first define a $\sigma-$algebra on labeled trees of the appropriate type. Then we will show Ω' is an element of this σ-algebra when Ω' is defined by appropriately measurable binding functions $\{B_l\}_{l \in N}$.

Proposition 1 *(Sigma Algebra on Θ) Let Θ be the set of labeled finite trees with leaves in the set T and with interior labels in the set D. If $\mathcal{F}$ is constructed as follows, using $\sigma-$algebras σ_T and σ_D on T and D, then $\mathcal{F}$ is a σ-algebra on Θ.*

Construction: Define $\tilde{\Theta}$ to be the set of ordered finite trees with unlabeled nodes. Let n_s be the number of leaf nodes and m_s be the number of non-leaf nodes in $s \in \tilde{\Theta}$. Define a mapping M_s of labels $T^{n_s} \times D^{m_s}$ to the nodes of s as follows:

1. Index the terminal nodes of s from left to right, $\{1, 2, \cdots, n_s\}$.
2. Index the nonterminal nodes in depth-first left to right order, $\{1, 2, \cdots, m_s\}$.
3. Assign labels $(t_1, \cdots, t_{n_s}, l_1, \cdots, l_{m_s}) \in T^{n_s} \times D^{m_s}$ using the indexing indicated in 1 and 2: t_k is the label of terminal node k for $k \in \{1, \cdots, n_s\}$, and similarly, l_j is the label of nonterminal node j for $j \in \{1, \cdots, m_s\}$.

Let Θ_s be the set of trees in Θ with topology s; M_s is a 1-1 and onto mapping from $T^{n_s} \times D^{m_s}$ to trees in Θ_s.

Using σ-algebras σ_T and σ_D on T and D define a σ-algebra on Θ_s via $\sigma_s = \{M_s(A) : A \in \sigma_T^{n_s} \times \sigma_D^{m_s}\}$. Define an overall σ-algebra $\mathcal{F}$ on Θ

$$\mathcal{F} = \{\cup_{s \in \tilde{\Theta}} A_s : A_s \in \sigma_s\}$$

by unions of elements from these σ_s's. //

This σ-algebra $\mathcal{F}$ on Θ can be extended to one on Θ^* in a straightforward manner. Let $\mathcal{F}^* = \cup_{n=1}^{\infty}\{A^n : A^n \in \mathcal{F}^n\}$; $\mathcal{F}^*$ defines a σ-algebra on Θ^*. The following lemma shows $\mathcal{F}$ is in some sense complete:

Lemma 1 *(Completeness) If $A \in \sigma_D$ and $Y \in \mathcal{F}^*$ then $\{l(\alpha^*) : l \in A, \alpha^* \in Y\} \in \mathcal{F}$.*

Proof: The set $\{l(\alpha^*)|l \in A, \alpha^* \in Y\}$ can be written as a countable union of sets, each of which is contained in $\mathcal{F}$. Using the definition of $\mathcal{F}^*$, write $Y = \cup_k Y^k$, where $Y^k = Y \cap \Theta^k \in \mathcal{F}^k$. Using the definition $\mathcal{F}$, each of these Y^k can be factored in terms of tree topology, $Y^k = \cup_{\vec{s}\in\tilde{\Theta}^k} Y^k_{\vec{s}}$, where $Y^k_{\vec{s}} = Y^k \cap (\Theta_{s_1} \times \cdots \times \Theta_{s_k}) \in \sigma_{s_1} \times \cdots \times \sigma_{s_k}$. Definition of the $\sigma'_s s$ and mapping function M_s, and the fact that $A \in \sigma_D$, guarantee the sets $\{l(\alpha^*) : l \in A, \alpha^* \in Y^k_{\vec{s}}\}$ are elements of $\mathcal{F}$. The desired set can be written as a countable union of these sets,

$$\{l(\alpha^*) : l \in A, \alpha^* \in Y\} = \cup_{k=1}^{n} \cup_{\vec{s}\in\tilde{\Theta}^k} \{l(\alpha^*) : l \in A, \alpha^* \in Y^k_{\vec{s}}\}$$

therefore, it is also an element of $\mathcal{F}$. //

For general T and discrete N, with $D = N$, the space of objects Ω defined by a compositional grammar $C = (T, N, \{B_l, S_l\}_{l\in N})$ is an element of $\mathcal{F}$ when the binding functions in C are appropriately measurable. Specifically, it is required that for each $l \in N$, B_l is $\mathcal{B}_l/\mathcal{F}^*$ measurable, where $\mathcal{B}_l$ is a σ-algebra on the range of B_l which contains the set of allowed binding function values, $S_l \in \mathcal{B}_l$. See [11].

Similar conditions ensure measurability of Ω' in case of interpreted compositional grammars.

Proposition 2 *(Measurability of Ω') For N discrete, T general, and $\{V_l\}_{l\in N}$ general, with $D = \cup_{l\in N} l \times V_l$ and $\sigma_D \ni l \times V_l \in \sigma_D \ \forall l \in N$, the space of objects Ω' defined by an interpreted compositional grammar $C' = (T, N, \{B_l, S_l, f_l, V_l\}_{l\in N})$ is an element of $\mathcal{F}$ when for each $l \in N$, B_l is $\mathcal{B}_l/\mathcal{F}^*$ measurable, where $\mathcal{B}_l$ is a σ-algebra on the range of B_l which contains the set of allowed binding function values, $S_l \in \mathcal{B}_l$.*

Proof: Let $\Omega = \Omega'$. It is enough to show that $\Omega_s \equiv \Omega \cap \Theta_s$ is measurable for each $s \in \tilde{\Theta}$. Define the daughter function $d_j(\omega)$ as the daughters of node j of ω ($d_j : \Theta \to \Theta^*$). From the definition of Ω, $\Omega_s = M_s \left\{ \cap_{j=1}^{m_s} X_{s,j} \right\}$ where

$$X_{s,j} = \{\vec{l} : B_{\phi(l_j)}(d_j(M_s(\vec{l}))) \in S_{\phi(l_j)}, \vec{l} = (t_1, \cdots, t_{n_s}, l_1, \cdots, l_{m_s}) \in T^{n_s} \times D^{m_s}\}$$

and $\phi(l') = l$ when $l' = (l, v), l \in N, v \in V_l$. The sets $X_{s,j}$ are elements of $\sigma_T^{n_s} \times \sigma_N^{m_s}$. To see this, observe the following: For each nonterminal node j of s define s_j to be the associated subtree topology. Measurability of $B_l, l \times V_l \in \sigma_D$, and the Completeness Lemma ensures $Y_l = \{(l, v)(\alpha^*) : (l, v) \in D, a^* \in B_l^{-1}(S_l)\} \in \mathcal{F}$, N discrete ensures $Y = \cup_{l \in N} Y_l \in \mathcal{F}$, therefore $Z = M_{s_j}^{-1}(\Theta_{s_j} \cap Y)$ must be an element of $\sigma_T^{n_{s_j}} \times \sigma_N^{m_{s_j}}$. The set $X_{s,j}$ is a component-wise permutation of $Z \times T^{(n_s - n_{s_j})} \times D^{(m_s - m_{s_j})}$ therefore $X_{s,j}$ is an element of $\sigma_T^{n_s} \times \sigma_D^{m_s}$.

Ω_s is M_s of a finite intersection of $X_{s,j}$ sets, therefore Ω_s is in $\mathcal{F}$. //

A measure P'' will now be constructed on $\Omega'' = \cup_{\omega' \in \Omega'} I(\omega')$ in a more general setting than the one used in Chapter 1. The definition of P'' will make use of an interpreted compositional measure P', a placement function

$$g : \Omega'' \to X$$

and conditional probabilities $K(\cdot|\omega')$ on placement function values.

Proposition 3 *(Extension of P' to $\Omega' \times X$) Let $\mathcal{F}_X$ be a $\sigma-$algebra on X. If $\forall B \in \mathcal{F}_X$, $K(B|\cdot)$ is $\mathcal{F}$-measurable and $K(\cdot|\omega')$ is a σ-finite measure on $(X, \mathcal{F}_X)$, then defined on rectangles $A \in \mathcal{F}$, $B \in \mathcal{F}_X$ extends to a unique measure*

Proposition 4 *(Sigma Algebra for Ω'', Existence of P'') Assume g is 1-1 on each $I(\omega')$. Let $S = \cup_{\omega' \in \Omega'} \omega' \times g(I(\omega'))$. Define $y : S \to \Omega''$ by $y(\omega', x') \equiv \omega \ni \omega \in I(\omega'), g(\omega) = x'$. If $S \in \mathcal{F} \times \mathcal{F}_X$, then*

$$\mathcal{F}'' = \{y(S \cap M) : M \in \mathcal{F} \times \mathcal{F}_X\}$$

is a σ-algebra on Ω''. Furthermore, if $K(g(I(\omega'))|\omega') = 1\ \forall\omega' \in \Omega'$, then

$$P''(E) = \hat{P}(y^{-1}(E))$$

is a probability measure on $(\Omega'', \mathcal{F}'')$.

Proof: $\{S \cap M : M \in \mathcal{F} \times \mathcal{F}_X\}$ is a σ-algebra. y is a 1-1 mapping from S to Ω'' since the $I(\omega')$ partition Ω'', and by assumption x is 1-1 on each $I(\omega')$. Therefore the direct image of y preserves set operations and $\mathcal{F}''$ is a σ-algebra for Ω''. In general, $P''(E) = \hat{P}(y^{-1}(E))$ is a measure on $\mathcal{F}''$, since y is a $\mathcal{F}''/\mathcal{F} \times F_X$ measurable function. When $K(g(I(\omega'))|\omega') = 1\ \forall\omega' \in \Omega'$

$$\begin{aligned} P''(\Omega'') &= \hat{P}(\cup_{\omega' \in \Omega'} \omega' \times g(I(\omega'))) \\ &= \int_{\Omega'} K(g(I(\omega'))|\omega') dP(\omega') \\ &= \int_{\Omega'} dP'(\omega') \\ &= 1 \end{aligned}$$

Thus P'' is a probability measure on $(\Omega'', \mathcal{F}'')$. //

One simple way to ensure $S \in \mathcal{F} \times \mathcal{F}_X$ is to select x such that $g(I(\omega')) = X$ for all $\omega \in \Omega'$. In this case S is simply the rectangle $\Omega' \times X$.

2.2 Existence and Uniqueness

The constraints on P given by (2.1) may not be sufficient to guarantee that P is well defined. This is due to the fact that (2.1) defines a system of equations for which P is a solution. In some cases it is possible for more than one probability distribution P to satisfy (2.1). In others there may be no solution to (2.1). Below are examples of each of these situations.

Example 7 *Nonunique Compositional Distribution.*
Let $T = \{c\}$, $N = \{1, 2\}$. Define B_1 and B_2, with $S_1 = S_2 = \{1\}$, such that

$$\Omega = \{c, 1(2(c, c)), 1(c), 2(c, c), 2(1(c), 1(c))\}$$

Let $Q(c) = q_c, Q(1) = q_1, Q(2) = q_2$. The probabilities assigned to each element in Ω by (2.1) are as follows:

$$\begin{aligned} P(c) &= q_c \\ P(1(c)) &= \frac{q_1}{q_c + P(2(c,c))} q_c \\ P(1(2(c,c))) &= \frac{q_1}{q_c + P(2(c,c))} P(2(c,c)) \\ P(2(c,c)) &= \frac{q_2}{q_c^2 + P(1(c))^2} q_c^2 \\ P(2(1(c),1(c))) &= \frac{q_2}{q_c^2 + P(1(c))^2} P(1(c))^2 \end{aligned}$$

Solving for $P(1(c))$ yields

$$P(1(c))^3 - q_1 P(1(c))^2 + (q_c^2 + q_2 q_c) P(1(c)) = q_1 q_c^2$$

Thus legitimate values for $P(1(c))$ correspond to the non-negative real roots of

$$z^3 - q_1 z^2 + (q_c^2 + q_2 q_c) z - q_1 q_c^2 \tag{2.3}$$

For some values of q_c, q_1, q_2 this polynomial has only a single non-negative real root. In this case P is unique. For other values of q_c,q_1,q_2, this polynomial has three positive real roots. In this case P is not unique. For example, when $q_c = .03, q_1 = .3395, q_2 = .6305$, the roots of (2.3) are approximately .0266, .0425 and .2704. The table below enumerates the various ensuing distributions on Ω.

$P(c)$	$P(1(c))$	$P(1(2(c,c)))$	$P(2(c,c))$	$P(2(1(c),1(c)))$
.03	.0266	.3129	.3534	.2771
.03	.0425	.2970	.2094	.4211
.03	.2704	.0691	.0077	.6288

Example 8 *Nonexistent Compositional Distribution.*

Let $T = \{c\}, N = \{1\}$,

$$B_1(\alpha, \beta) = 1$$

with $S_1 = \{1\}$. Ω is the set of all possible binary trees with leaf labels equal to c and interior labels equal to 1 union the singleton tree $\{c\}$. Define the external measure on labels by $Q(c) = q$, $Q(1) = 1 - q$. $Q_1(x)$ must necessarily be 1 for $x = 1$, and 0 otherwise.

Let E_n be the set of trees in Ω of depth less than or equal to n.

$$E_0 = \{c\}$$

$$E_{n+1} = \{c\} \cup \{1(\alpha, \beta) : (\alpha, \beta) \in E_n \times E_n\}$$

Let $S_n = P(E_n)$, by definition

$$S_{n+1} = q + (1 - q)S_n^2$$

If P is well defined it must be the case that $P(\Omega) = P(\lim E_n) = \lim S_n = 1$, since by definition $\Omega = \lim E_n$ and all compositional distributions have the property that $P(\Omega) = 1$. Examination of the S_n sequence reveals that its limit displays q-dependent critical behavior: for $q \geq 1/2$ $\lim S_n = 1$, for $q < 1/2$ $\lim S_n = q/(1 - q)$. (This can be shown formally by bounding the difference $(S_{n+1} - S_n)$ between zero and $(\min(q/(1-q), 1) - S_n)$ and observing $(S_{n+1} - S_n)$ is zero only for $S_n \in \{q/(1 - q), 1\}$.)

At present not a great deal is known about general conditions which ensure P is well defined. Under certain conditions a compositional grammar and distribution can be shown to be equivalent to a corresponding probabilistic context-free grammar, see Chapter 3 and also Geman [11]. In these cases the theory of branching processes [13], [12] may be employed to determine whether or not P is well-defined. [3]

Zhiyi Chi's Thesis [5] contains an interesting existence result for "finite" compositional systems. It states that if $\forall l \in N$ and $\forall b \in S_l$

$$\#\{\omega : L(\omega) = l, B_l(\alpha^*) = b\} < \infty$$

[3] In brief: Let P_{cfg} be the corresponding PCFG measure, if $P_{cfg}(\Omega) = 1$ then the PCFG is said to be "consistent" (or "tight"). In this case the compositional distribution P is well defined. If $P_{cfg}(\Omega) < 1$ then the PCFG is said to be "inconsistent" and the corresponding compositional P does not exist.

then at least one compositional distribution on Ω exists for any Q and Q_l, where Q_l is absolutely continuous with respect to P_l^*.

In general, problems regarding existence and uniqueness of P arise in composition systems which either directly or indirectly make recursive use of object labels and binding function values. Specifically, for each composition $l(\alpha^*) \in \Omega$ define the **type** of $l(\alpha^*)$ by

$$t_{l(\alpha^*)} \equiv (l, B_l(\alpha^*))$$

If a strict partial order $>$ on types exists in the sense that $\omega = l(\alpha_1, \cdots, \alpha_n) \in \Omega \Rightarrow t_\omega > t_{\alpha_1}, \cdots, t_\omega > t_{\alpha_n}$, then the system of equations in (2.1) defines P explicitly in a bottom-up manner. If such an ordering does not exist then the system makes recursive use of its labels and binding function values.

One way of rigging a composition system to avoid recursive use of label and binding function values is to require that each binding function include a "size" attribute:

$$s(\alpha^*) = \sum_{i=1}^{n} \#\text{nodes}(\alpha_i), \quad \alpha^* = (\alpha_1, \cdots, \alpha_n)$$

In this case such an ordering on types exists. [4]

2.3 Encoding Wins and Likelihood Ratios

From the coding perspective developed by Shannon (see, for example, [7]), the optimal number of bits to use in encoding discrete quantities X with a known probability distribution P is $-\log P(X)$. Such encodings achieve the minimum possible average code length when used for transmission or compression. The **encoding win** of a composition $l(\alpha, \beta)$ is the number of bits saved (or alternatively, compression achieved) by encoding α and β via $l(\alpha, \beta)$ versus two distinct codewords. It is given by

$$dL = \log P(l(\alpha, \beta)) - \log P(\alpha) - \log P(\beta)$$

[4] Of course even in this case, for P to exist, one must ensure $Q_l << P_l^*$ by not assigning mass to Q_l outside $B_l(\Omega^*)$. Therefore, even here, design of P may not be entirely trivial.

This quantity represents the win (or possible loss) of an encoding.

Several additional interpretations for dL are possible including:

- In analogy with physics and chemistry it can viewed as a form of "binding energy". $dL > 0$ implies that objects α and β can bind, via mechanism l, to form a new entity, $l(\alpha, \beta)$, with a lower energy state, $E_{new} = E_{old} - dL$.
- From a statistical perspective, it is the value of a log likelihood ratio test for presence of independent α and β versus an aggregate object which contains both α and β; $dL > 0$ corresponds to the aggregate object $l(\alpha, \beta)$ being more likely.

The total number of bits saved by a composition ω is defined as

$$W(\omega) = \log P(\omega) - \sum_{t \in \kappa(\omega)} \log P(t)$$

This quantity is the number of bits saved (or lost) in going from an iid encoding of the terminals $\kappa(\omega)$ with length $-\sum_{t \in \kappa(\omega)} \log Q(t)$, to an encoding ω, with length $-\log P(\omega)$. If the terminals of α do not overlap the terminals of β (i.e., $\kappa(\alpha) \cap \kappa(\beta) = \phi$), then calculation of $W(l(\alpha, \beta))$ can be defined recursively by

$$W(\omega) = \begin{cases} dL + W(\alpha) + W(\beta) & \text{when } w = l(\alpha, \beta) \\ 0 & \text{when } w \in T \end{cases}$$

Here dL is the local encoding win and $W(\alpha)$ and $W(\beta)$ are the total number of bits saved by by α and β.

The encoding win for compositional distributions has a particularly pleasing form. For discrete Ω, and a dyadic composition rule B_l, the encoding win for $l(\alpha, \beta) \in \Omega$ is

$$\begin{aligned} dL &= \log P(l(\alpha, \beta)) - logP(\alpha) - logP(\beta) && (2.4) \\ &= \log Q(l) Q_l(B_l(\alpha, \beta)) - \log P \times P(B_l(\alpha, \beta)) && (2.5) \end{aligned}$$

Bits are won or lost solely based on the efficiency of encoding the value of $B_l(\alpha, \beta)$ in the context of a composition with label l versus an independence hypothesis on α and β.

For interpreted compositional distributions, the notion of encoding win is not directly applicable since compositions do not simply offer an alternative means for encoding their daughter constituents. Rather, each composition $(l, v)(\alpha'^*) \in \Omega'$ adds information about its constituents through its relation function value v. However, for the applications developed in this thesis P' on Ω' is extended to a measure P'' on objects Ω''. Here, for Ω'' discrete, B_l a dyadic composition rule, and $l(\alpha, \beta) \in I(l'(\alpha', \beta'))$,

$$\begin{aligned} dL &= \log Q(l)Q_l(B_l(\alpha', \beta')) - \log P' \times P'(B_l(\alpha', \beta'))+ \\ &\quad \log Q(f_l(\alpha, \beta)|l)K(g(l(\alpha, \beta))|l'(\alpha', \beta')) - \log K(g(\alpha)|\alpha')K(g(\beta)|\beta') \end{aligned}$$

The first line of this expression is the win (or loss) of encoding the value of $B_l(\alpha', \beta')$ in the context of label component l versus independent α' and β'; the second line is the win (or loss) of encoding the position of α and β relative to one another and an overall placement for $l(\alpha, \beta)$ versus independent placement of α and β.

For non-discrete Ω and Ω'' encoding wins are defined in terms of log likelihood ratios. These may be interpreted as the relative code lengths between encodings for continuous variables. Their values reduce to the above cases when Ω or Ω'' are discrete.

Fix $l \in N$ and define a measure μ on $\Omega_l^* = \{\alpha^* : l(\alpha^*) \in \Omega\}$ by

$$\mu(E) = P(l(\alpha^*) : \alpha^* \in E)$$

μ represents the probability of events in Ω_l under the hypothesis that they occur as part of a composition $l(\alpha^*)$. P^* is the probability of events in Ω_l^* under the hypothesis that they occur independently. The encoding win for a composition $l(\alpha^*)$ is defined by

$$dL = \log \frac{d\mu}{dP^*}(\alpha^*) \tag{2.6}$$

$$= \log Q(l)\frac{dQ_l}{dP_l^*}(B_l(\alpha^*)) \tag{2.7}$$

since by definition $\mu(E) = \int_E Q(l)\frac{dQ_l}{dP_l^*}(B_l(\alpha^*))dP(\alpha^*)$.

For P'', encoding wins are defined analogously. Let $\Omega''_l = \{\alpha^* : l(\alpha^*) \in \Omega''\}$, and define

$$\mu''(E) = P''(l(\alpha^*) : \alpha^* \in E)$$

The encoding win for $l(\alpha^*)$ is

$$dL = \frac{d\mu''}{dP''^*}(\alpha^*)$$

(Here, technically speaking, μ'' may not be absolutely continuous with respect to P''^*. When this is the case, let $\lambda + \rho$ be the Lebesgue decomposition of μ'' with respect to P''^*, $\mu'' = \lambda + \rho, \lambda \perp P''_l, \rho << P''_l$, and define the encoding wins in terms of ρ via $dL = \log \frac{d\rho}{dP''^*}(\alpha^*)$.)

When P' and $K(\cdot|l'(\alpha', \beta'))$ are absolutely continuous with respect to Lebesgue measure, the encoding win for $l(\alpha, \beta) \in I(l'(\alpha', \beta')) \subseteq \Omega''$ reduces to

$$dL = \log \frac{Q(l, f_l(\alpha, \beta))K(g(l(\alpha, \beta))|l(\alpha, \beta))\frac{dQ_l}{dP''^*_l}(B_l(\alpha', \beta'))}{K(g(\alpha)|\alpha')K(g(\beta)|\beta')} J_{\alpha', \beta'}(g(\alpha), g(\beta))$$

where $J_{\alpha', \beta'}$ is the absolute value of the determinant of the Jacobian of the change of variables

$$T_{\alpha', \beta'}(x_\alpha, x_\beta) = (f_l(y(\alpha', x_\alpha), y(\beta', x_\beta), g(l(y(\alpha', x_\alpha), y(\beta', x_\beta)))))$$

with

$$y(\omega', x) = \omega \ni \omega \in I(\omega'), g(\omega) = x$$

evaluated at $x_\alpha = g(\alpha)$ and $x_\beta = g(\beta)$. For $(x_\alpha, x_\beta) = (x^1, \cdots, x^n)$

$$J_{\alpha', \beta'}(x_\alpha, x_\beta) \equiv |det((\frac{dT^i_{\alpha', \beta'}}{dx^i})|_{x^1, \cdots, x^n})|$$

Example 9 *Encoding Wins for Lines and L-junctions.*
(Example 4 and Example 6 Continued.) For each orbit $I(\omega')$ define the **unit object** ω^0 *as the element $\omega \in I(\omega')$ for which $g(\omega) = (0, 0, 1, 0)$. Unit objects can be used to express $T_{\alpha', \beta'}$ in a form convenient for evaluation of $|J|$ since (with an appropriate abuse of notation)*

$$y(\omega', x_\omega) = \|x_\omega\| R_{\angle x_\omega} \omega^0 + (x^1_\omega, x^2_\omega)$$

Evaluation of the right hand side of this quantity is as follows: For $x_\omega = (y_1, y_2, y_3, y_4)$, $\angle x_\omega = \arctan(y_4 - y_2, y_3 - y_1)$, $\|x_\omega\| = \sqrt{(y_4 - y_2)^2 + (y_3 - y_1)^2}$, *the scale and rotation* $\|x_\omega\| R_{\angle x_\omega} \in GL(2)$ *is applied to the endpoints of each terminal in* $Y(\omega^0)$, *then the constant* $(x^1_\omega, x^2_\omega) = (y_1, y_2) \in R^2$ *is added to each of these transformed endpoints.*

For line compositions, the components of $f_1(y(\alpha', x_\alpha), y(\beta', x_\beta))$ *expressed in terms of* α^0,β^0 *and* x_α, x_β *are*

$$(\angle x_\beta + \theta_{\beta^0_-}) - (\angle x_\alpha + \theta_{\alpha^0_+}) \tag{2.8}$$

$$\frac{\|x_\beta\| r_{\beta^0_-}}{\|x_\alpha\| r_{\alpha^0_+}} \tag{2.9}$$

$$R_{-(\angle x_\alpha + \theta_{\alpha^0_+})} \frac{\|x_\beta\| R_{\angle x_\beta} x_{\beta^0_-} + (x^1_\beta, x^2_\beta) - (\|x_\alpha\| R_{\angle x_A} x_{\alpha^0_+} + (x^1_\alpha, x^2_\alpha))}{\|x_\alpha\| r_{\alpha^0_-}} \tag{2.10}$$

The placement function for lines $g(1(y(\alpha', x_\alpha), y(\beta', x_\beta)))$ *is*

$$(x^1_\alpha, x^2_\alpha, x^3_\beta, x^4_\beta) \tag{2.11}$$

Thus $T_{\alpha',\beta'}$ *for lines is defined by equations (2.8) through (2.11); therefore*

$$J_{\alpha',\beta'}(x_\alpha, x_\beta) = \frac{r_{\beta^0_-} \|(x^3_\beta, x^4_\beta) - (x^1_\alpha, x^2_\alpha)\|^2}{\|x_\alpha\|^5 \|x_\beta\| r^3_{\alpha^0_+}}$$

The encoding win is

$$dL = \log \frac{Q(1)}{Q(0)^2 + 2Q(0)Q(1) + Q(1)^2} d^4 G_1(f_l(\alpha, \beta)) J_{\alpha',\beta'}(x_\alpha, x_\beta)$$

For L-junctions $T_{\alpha,\beta}$ *is defined by* f_2

$$(\angle x_\beta - \angle x_\beta, \frac{\|x_\beta\|}{\|x_\alpha\|}, R_{-\angle x_\alpha} \frac{(x^1_\beta, x^2_\beta) - (x^1_\alpha, x^2_\alpha)}{\|x_\alpha\|})$$

and the placement function x

$$(x^1_\alpha, x^2_\alpha, x^3_\beta, x^4_\beta)$$

Here

$$J_{\alpha',\beta'}(x_\alpha, x_\beta) = \frac{\|(x^3_\beta, x^4_\beta) - (x^1_\alpha, x^2_\alpha)\|^2}{\|x_\alpha\|^5 \|x_\beta\|}$$

and the encoding win is

$$dL = \log \frac{Q(2)}{Q(0)^2 + 2Q(0)Q(1) + Q(1)^2} d^4 G_2(f_2(\alpha, \beta)) J_{\alpha',\beta'}(x_\alpha, x_\beta)$$

Chapter 3

Sampling

3.1 The RSR Algorithm

In some instances the hierarchical nature of compositional distributions allows the use of a simple recursive Monte Carlo algorithm for sampling.

Assuming $C = (T, N, \{B_l, S_l\}_{l \in N})$ is such that T, N are discrete, and the B_l's are dyadic binding predicates (i.e., $\forall l \in N$, $S_l = \{1\}$), the basic idea is as follows: First a node label l is sampled from the external distribution Q. If the label is for a leaf node it is immediately returned. Otherwise, potential constituents (α, β) for a tree with label l as its root label are then sampled until a pair is found for which $B_l(\alpha, \beta) = 1$. The composition $l(\alpha, \beta)$ is then returned. The Recursive Sample Reject (RSR) procedure below contains a pseudo-code listing for this algorithm.[1] A call to this procedure with conditioning set $A \subseteq \Omega$ returns a sample from $P(\cdot|A)$.

Algorithm 1 *Recursive Sample Reject (RSR)*

`function` $\omega = sample\ P(\cdot|A)$

`repeat`

  $l = sample\ Q(\cdot|\{L(\omega') : \omega' \in A\})$

  `if` $(l \in T)$

    $\omega = l$;

  `else`

    `repeat`

      $\alpha = sample\ P(\cdot|\{\alpha' : \exists \beta' \in \Omega, B_l(\alpha', \beta') = 1\})$;

      $\beta = sample\ P(\cdot|\{\beta' : \exists \alpha' \in \Omega, B_l(\alpha', \beta') = 1\})$;

    `until`$(B_l(\alpha, \beta) = 1)$

    $\omega = l(\alpha, \beta)$;

  `endif`

`until`$(\omega \in A)$

[1] Modification of this procedure and the other results in this chapter to include binding functions (in addtion to binding predicates) is straightforward, as long as S_l is discrete and for each $b \in supp(Q_l)$ there exists n such that $B_l^{-1}(b) \subseteq \Theta^n$.

(The notation used in this code fragment should be obvious; $\omega = \cdots$ are assignments of appropriate representations for the terminals and compositions generated. *sample* X indicates a function call appropriate for sampling distribution X.)

Unfortunately, in some cases the RSR algorithm does not terminate with probability 1. Instead it gets lost in an infinite recursion. For now however, let us assume this algorithm terminates. In this case the following proposition establishes correctness:

Proposition 5 *When the RSR algorithm (1) terminates w.p.1, it samples* P.

Proof: Let E represent the distribution on outputs of (1) and let

$$F(A) = \{L(\omega) : \omega \in A\}$$

$$G(l) = \{\alpha' : \exists \beta' \in \Omega, B_l(\alpha', \beta') = 1\}$$

$$H(l) = \{\beta' : \exists \alpha' \in \Omega, B_l(\alpha', \beta') = 1\}$$

The distribution on α, β pairs generated by the inner sample reject loop is given by

$$\frac{E(\alpha|G(l))E(\beta|H(l))}{\sum_{(\alpha',\beta')\ni B_l(\alpha',\beta')=1} E(\alpha'|G(l))E(\beta'|H(l))} = \frac{E(\alpha)E(\beta)}{\sum_{(\alpha',\beta')\in B_l} E(\alpha')E(\beta')} \qquad (3.1)$$

since $\{(\alpha', \beta') : B_l(\alpha', \beta') = 1\} \subseteq G(l) \times H(l)$. Let $\Pi(\alpha, \beta|B_l)$ represent the right hand side of (3.1). The overall distribution on outputs of (1) is

$$\begin{aligned} E(\omega) &= \begin{cases} \frac{Q(l|F(A))}{\sum_{l'\in A, l'\in T} Q(l'|F(A)) + \sum_{l'(\alpha',\beta')\in A} Q(l'|F(A))\Pi(\alpha',\beta'|B_{l'})}, & \omega = l \in T \\ \frac{Q(l|F(A))\Pi(\alpha,\beta|B_l)}{\sum_{l'\in A, l'\in T} Q(l'|F(A)) + \sum_{l'(\alpha',\beta')\in A} Q(l'|F(A))\Pi(\alpha',\beta'|B_{l'})}, & \omega = l(\alpha,\beta) \end{cases} \\ &= \begin{cases} \frac{Q(l)}{\sum_{l'\in A, l'\in T} Q(l') + \sum_{l'(\alpha',\beta')\in C} Q(l')\Pi(\alpha',\beta'|B_{l'})}, & \omega = l \in T \\ \frac{Q(l)\Pi(\alpha,\beta|B_l)}{\sum_{l'\in A, l'\in T} Q(l') + \sum_{l'(\alpha',\beta')\in A} Q(l')\Pi(\alpha',\beta'|B_{l'})}, & \omega = l(\alpha,\beta) \end{cases} \end{aligned}$$

since $A \subseteq \{\omega : L(\omega) \in F(C)\}$. Therefore $E(\omega) = P(\omega|A)$. //

Beyond the termination constraint, two other issues limit the usefulness of this sampling algorithm. First, implementation of the RSR algorithm may be difficult since it may be

hard to determine the identity of the conditioning sets

$$G(l) = \{\alpha' : \exists \beta' \in \Omega, B_l(\alpha', \beta') = 1\} \tag{3.2}$$

$$H(l) = \{\beta' : \exists \alpha' \in \Omega, B_l(\alpha', \beta') = 1\} \tag{3.3}$$

used within the inner sample reject loop. In practice

$$G'(l) = \{\alpha'' : \exists \alpha', \beta' \in \Omega, B_l(\alpha', \beta') = 1, L(\alpha'') = L(\alpha')\}$$

$$H'(l) = \{\beta'' : \exists \alpha', \beta' \in \Omega, B_l(\alpha', \beta') = 1, L(\beta'') = L(\beta')\}$$

are used as substitutes for these sets. (In this case the proof of proposition (3.1) still holds since $G'(l) \times H'(l)$ contains $G(l) \times H(l)$.) Second, the algorithm may require an enormous amount of computer time since the inner sample reject loop exits with probability

$$\frac{P \times P(\{(\alpha, \beta) : B_l(\alpha, \beta) = 1\})}{P \times P(G(l) \times H(l))}$$

An exponential increase in computation time can occur as the number of nonterminal nodes in a composition grows. (Exchange of $G'(l)$ and $H'(l)$ for $G(l)$ and $H(l)$ may exacerbate the situation due to a decrease in exit probabilities.)

3.2 Termination

Operation of this sampling algorithm can modeled via a Markov chain. The states of this chain correspond to an initial start state s, $|\Omega|$ absorbing states representing algorithm termination, and a large set of non-absorbing states representing the possible program states of the RSR algorithm after *sample* $Q(\cdot|F(A))$ is called.

For example, define a compositional grammar C using $T = \{c\}$, $N = \{1, 2\}$ and binding predicates B_1 and B_2 such that

$$\Omega = \{c, 1(c, c), 2(c, 1(c, c)), 2(1(c, c), c)\}$$

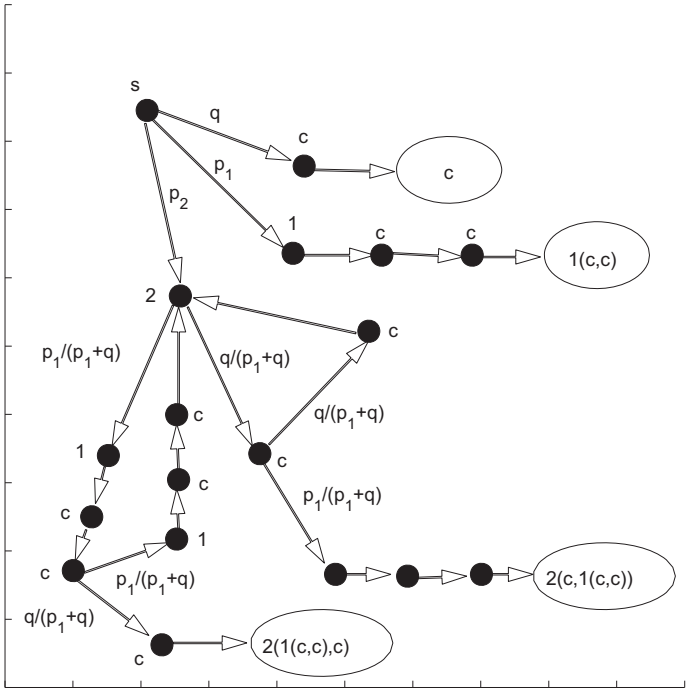

Figure 3.1: *Operation of the RSR algorithm may be modeled using a Markov chain.*

and define P by assigning $Q(c) = q, Q(1) = p_1, Q(2) = p_2$. Figure 3.1 contains a Markov chain model for operation of the RSR algorithm for sampling P. Computation begins in state s; from here a root label type is sampled, the possibilities are c with probability q, 1 with probability p_1, and 2 with probability p_2. If c is selected, the algorithm terminates and returns c. If 1 is selected, $sampleP(\cdot|\{c\})$ is called twice and the algorithm terminates and returns $1(c, c)$. If 2 is selected, $sampleP(\cdot|\{1(c, c), c\})$ is called twice. The algorithm terminates and returns if these calls return either c and $1(c, c)$ or $1(c, c)$ and c; otherwise it "reloops" and attempts to satisfy B_2 again using these two calls.

Termination of the RSR algorithm is guaranteed if each branch emanating from the starting node s is guaranteed to terminate with probability one. (Or equivalently, for all $l \in N$, *sample* $P(\cdot|\{l\})$ terminates with probability one.) For the above example, this is obviously the case since the branches c and 1 from s terminate deterministically and in branch 2 from s the number of non-absorbing states is finite, and, due to the definition of

P, there is positive probability path from each non-absorbing state to an absorbing state.[2]

A variety of ways for designing C and Q so that the RSR algorithm terminates have been explored. One obvious method is to design C so that the number of states in each branch emanating from s is finite. This precludes the explicit or implicit recursive use of labels and binding function values as described in Section 2.2.

Another method is to go ahead and allow recursive use of labels and binding function values but only as a part of a "context-free" subsystem of compositions. Compositions of this type have binding functions that depend only on the root labels and binding function values of their immediate constituents; i.e., for each composition rule l in the subsystem there exists a function B'_l such that

$$B_l(l_1(\alpha), \cdots, l_n(\alpha_n)) = B'_l(l_1, B_{l_1}(\alpha_1), \cdots, l_n, B_{l_n}(\alpha_n))$$

In this case, when the equivalent PCFG for this subsystem is consistent, the Markov chain branches associated with these compositions return with probability 1.

3.3 RSR for interpreted compositional distributions

The RSR algorithm (1) is easily modified to accommodate interpreted compositional distributions by

Algorithm 2 *ICD Recursive Sample Reject*

```
function ω = sample P'(·|A)
repeat
   l = sample Q(·|{L(ω') : ω' ∈ A})
   if (l ∈ T)
      ω = l;
   else
      repeat
```

[2]If no such path existed then Q_l would not be absolutely continuous with respect to P_l^*, and P would not be defined.

$\alpha = sample\ P'(\cdot|\{\alpha' : \exists\beta' \in \Omega, B_l(\alpha', \beta') = 1\})$;

$\beta = sample\ P'(\cdot|\{\beta' : \exists\alpha' \in \Omega, B_l(\alpha', \beta') = 1\})$;

`until(`$B_l(\alpha, \beta) = 1$`)`

$v = sampleQ(\cdot|l)$;

$\omega = (l, v)(\alpha, \beta)$;

`endif`

`until(`$\omega \in A$`)`

A variant of this algorithm was used for generating many of the examples used in this thesis.

Chapter 4

Recognition

In the present chapter, the optimization problems and algorithms developed apply equally well to compositional distributions and to the P'' extension of interpreted compositional distributions. In order to keep the presentation succinct the symbol P will serve in a dual role in that it may represent either a compositional distribution or P''. Also, in the continuous case P_D will be defined in terms of a product of compositional or P'' densities.[1]

Our approach to object recognition has been to attempt to solve, at least approximately, the MAP optimization problem presented in Section 1.4:

$$s_0 = \arg \max_{s \in S|D} P_D(s) \tag{4.1}$$

where each scene s in $S|D$ is a finite subset of Ω whose elements have non-overlapping leaves which are in the data D.

$$S|D = \{s : s \subseteq \Omega, |s| < \infty, \cup_{\omega \in s} \kappa(\omega) = D, \kappa(\alpha) \cap \kappa(\beta) = \phi, \alpha \neq \beta,\ \alpha \in s, \beta \in s\}$$

Approximation of (4.1) proceeds in two steps:

1. *Instancing.* The compositional definition of Ω and its measure P are used to create a large set L of potential scene elements. This is a bottom-up process in which composition rules are used to iteratively select new objects to add to L from the previously selected objects and data D. Each object in L is a parse of D or some fragment of D.

2. *Aggregation.* Upon completion of the instancing step, the set of potential scene elements L is processed greedily in an effort to generate a scene $\hat{s}$ (or scenes) which satisfy (4.1) at least approximately. Elements of $\hat{s}$ are selected iteratively based on their total encoding win W and the previously selected elements of $\hat{s}$.

(Using a variation on these two steps to solve 4.1 exhaustively is probably a bad idea since, in general, one typically encounters composition systems which admit an exponential (or at least very large) number of compositions which explain different portions of D, and

[1] See [11] for details on this type of construction.

this long list of compositions would then have to be processed in a manner naively equivalent to finding the solution to a set covering problem known to be NP-hard [9].)

4.1 Instancing

During the instancing process we seek to produce a set of objects L for use in construction of estimate(s) $\hat{s}$ of s_0. At one extreme, one could take $L = \Omega|D$ where $\Omega|D$ is the set of all possible elements of Ω which explain, at least partially, the data present; i.e.

$$\Omega|D \equiv \{\omega \in \Omega : \kappa(\omega) \subseteq D\}$$

Unfortunately the size of $\Omega|D$ can be enormous, perhaps infinite. In these cases carrying out an explicit construction of all of its elements on a computer would be intractable. Regardless, we begin our discussion by defining a procedure for computing $\Omega|D$.

The compositional definition of Ω allows us to define $\Omega|D$ in terms of repeated bottom-up compositions of the data present; $\Omega|D = \lim_{k\to\infty} V_k$ where

$$V_0 = \phi \tag{4.2}$$

$$V_{k+1} = D \cup \{l(\alpha,\beta) : l \in N, B_l(\alpha,\beta) \in S_l, (\alpha,\beta) \in V_k \times V_k\} \tag{4.3}$$

Direct implementation of this sequence on a computer would be very inefficient since each iteration of the sequence examines all possible dyadic pairings of the elements generated thus far. A better approach is to examine only new possible pairings:

$$\begin{aligned} V_0 &= D \\ S_0 &= \{l(\alpha,\beta) : l \in N, B_l(\alpha,\beta) \in S_l, (\alpha,\beta) \in D \times D\} \\ V_{k+1} &= V_k \cup S_k \\ S_{k+1} &= \{l(\alpha,\beta) : l \in N, B_l(\alpha,\beta) \in S_l, \\ &\qquad (\alpha,\beta) \in (S_k \times S_k) \cup (S_k \times V_k) \cup (V_k \times S_k)\} \end{aligned}$$

Here each pairing is examined only once during the generation of the entire sequence $V_1, V_2, \cdots$. Unfortunately computation of even this version of the V_k sequence may be intractable except for small k.

Depending on the nature of the compositional grammar and distribution being used, a pruned version of the V_k sequence may still yield a reasonable set L with which to construct $\hat{s}$. The basic idea is to define L using a **pruning heuristic** which eliminates composition of "unlikely" and "redundant" objects. One way of defining such a heuristic is through a mapping $h : 2^\Omega \times 2^\Omega \rightarrow 2^\Omega$ and the following sequence

$$\begin{aligned} L_0 &= D \\ S_0 &= \{l(\alpha,\beta) : l \in N, B_l(\alpha,\beta) \in S_l, (\alpha,\beta) \in D \times D\} \\ L_{k+1} &= h(S_k, L_k) \\ S_{k+1} &= \{l(\alpha,\beta) : l \in N, B_l(\alpha,\beta) \in S_l, \\ &\qquad (\alpha,\beta) \in (S_k \times S_k) \cup (S_k \times L_k) \cup (L_k \times S_k)\} \end{aligned}$$

The mapping h acts as a filter for unwanted objects, $h(S_k, L_k) \subseteq S_k \cup L_k$. In theory, one would select h such that size of L remains tractable while maintaining $s_0 \subseteq L$. In practice this is very difficult.

The pruning heuristic for the OHRS application makes use of two different types of pruning operations. The first, *Attribute Matching*, attempts to remove redundant compositions by comparing the attributes of objects in S_k with the attributes of other objects in S_k and with the attributes of the previously instanced objects in L_k. When two objects $\alpha \in S_k$ and $\beta \in S_k \cup L_k$ share similar attributes, the one with the smallest total encoding win is deleted along with any other compositions in L_k which contain it as a sub-tree. In Figure 4.1, panel (a) and panel (b) contain two characters which differ by a few terminals. An appropriately defined pruning heuristic would delete one in favor of the other. Without such similarity checks, the number of redundant compositions in L tends to grow exponentially. (However, even with such checks the number of distinct compositions in L may also grow exponentially, so this may be a stop-gap measure in terms of reducing the overall size of L.)

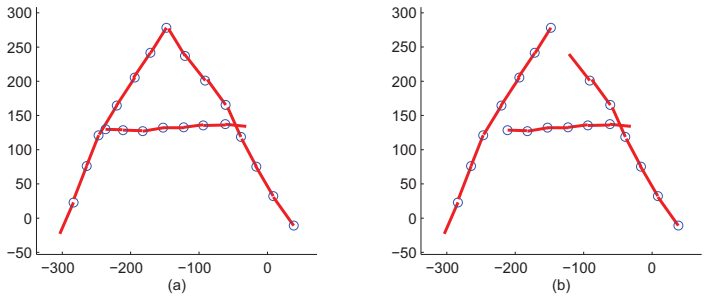

Figure 4.1: *Without the use of a pruning heuristic many similar objects such as (a) and (b) can end up in L.*

The second type of pruning operation used in the OHRS application is *Critical Region Testing*. Here highly unlikely compositions are avoided by requiring that B_l and f_l (and perhaps g) lie within associated critical regions of Q_l and $Q(\cdot|l)$ (and perhaps K). For example, in the OHRS application compositions $l(\alpha, \beta) \in S_k$ are examined to determine whether or not $f_l(\alpha, \beta)$ is within a critical region of $Q(\cdot|l)$. When $f_l(\alpha, \beta)$ is outside this region, the composition $l(\alpha, \beta)$ is deleted. This procedure seems entirely reasonable; if 99% of all objects formed via composition rule l fall within this region, and the current object falls outside of this region, there is a very good chance that $l(\alpha, \beta)$ would represent a misclassification of the data. See Figure 4.2.

Of course neither of these pruning operations is guaranteed to keep s_0 in L. One idea is to avoid actual object deletions and instead control the order of object compositions. Compositions involving objects which seem redundant or unlikely ought to be deferred for later evaluation. Using this approach L_k is $\Omega|D$ in the limit, but early on L_k would hopefully contain a sparse set of reasonable candidates for construction of s_0. Several experiments have been conducted along these lines.

This approach was partially inspired by J. Canning's approach to pattern recognition [4]. Here a priority queue was employed for determining the order in which various objects models were updated.

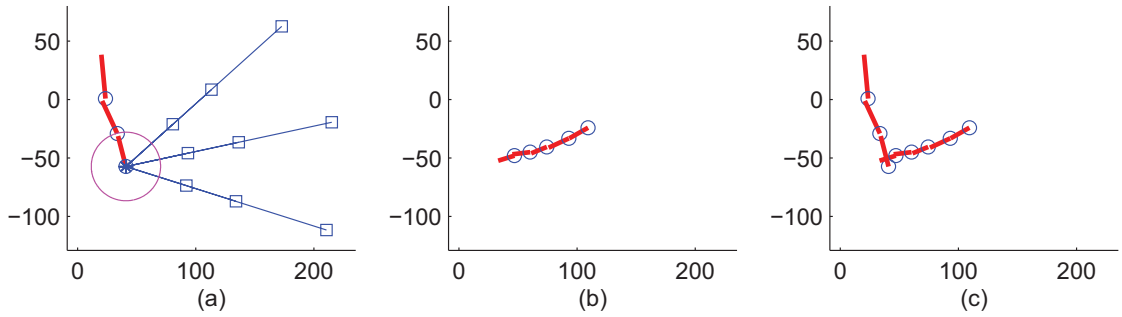

Figure 4.2: *Example critical regions for an L-junction composition. (a) Critical regions for lines β displayed with respect to α. If a line β is to compose with line α to form an L-junction its endpoint x_β must lie within the disk, its angle θ_β must fall with the arc section, and its overall length must lie between the arc section origin and the three square markers. (b) A line β whose attributes lie within these critical regions. (c) The ensuing L-junction.*

The program listing below contains pseudo-code for one version of the approach. Objects are added to L sequentially and indexed by $\{1, 2, 3, \cdots\}$. $L(i)$ will represent the ith object in L. Initially L contains the data D. During each iteration of the outer loop a **candidate selection heuristic** g is used to select a list C of "candidate" elements from L to compose. The two inner loops compose each element in C with elements in L. An "odometer" value $u(i)$ associated with each element $L(i)$ in L is used to avoid all previously attempted object pairings.

Algorithm 3 *Prioritized Calculation of* $\Omega|D$

$L = D$;
`repeat`
  `k=|L|;`
  $C = g(L, u)$
  `for` $i \in C$
    `for` $j \in \{n : n \leq k, u(n) < i, u(i) < n\}$
      $L = L \cup \{l(L(i), L(j)) : l \in N, B_l(L(i), L(j)) \in S_l\}$;
      $L = L \cup \{l(L(j), L(i)) : l \in N, B_l(L(j), L(i)) \in S_l\}$;
    `end`
    $u(i) = k$;

```
    end
until(C = φ)
```

During operation of algorithm (3), the elements in L which may not have already been composed with all the other elements in L have indices in $\{i : u(i) < |L|\}$. When $\{i : u(i) < |L|\} = \phi$ no unexplored binding possibilities remain and $L = \Omega|D$.

Here, one would like to be able to define the candidate selection heuristic g so that likely components of the scene being processed are contained in L early on. But, unfortunately, selection of such a g seems difficult. Consider for example a string of hand written letters "BBBB" acting as input to the system; there are 4 B's within this string. If the grammar being used allows composition of the upper-case latin letters, then the terminals of each of these B's may also be composed into I's,P's,L's,C's,3's, etc. Blind application of a composition rule for string concatenation would then find all possible strings (such as "BLIP" contained within this set of 4 B's. In such a case an exponential number of different string representations will be found. The grammar developed for online handprint recognition in the next chapter will suffer from this type of problem. What seems to be called for, but what has not been developed, is the use an inhibitory mechanism to prevent the construction of, say the string "PIP3", when such a string is in fact already contained in "BBBB". Perhaps some hints for design of such a mechanism can be determined by a careful examination of the structures utilized in biological vision systems.

4.2 Aggregation

The set of potential scene elements L is used to construct an estimate $\hat{s}$ for a scene s_0 which satisfies (4.1). Rather than attempting to approximate (4.1) directly, say by selecting the "most probable" element of L, and then the next "most probable" element in L, etc., a greedy approximation to a dual problem involving encoding wins is used.[2]

$$\arg\max_{s \in S|D} P_D(s) \quad \propto \quad \arg\max_{s \in S|D} \sum_{\omega \in s} \log P(\omega)$$

[2]A greedy approach based directly on object probabilities is fraught with difficulties since "smaller" objects such as terminals will generally have higher probability than "larger" objects.

$$
\begin{aligned}
&= \arg\max_{s \in S|D} \sum_{\omega \in s} W(w) + \sum_{t \in D} Q(t) \\
&\propto \arg\max_{s \in S|D} \sum_{\omega \in s} W(w)
\end{aligned}
$$

The first element ω_0 placed in $\hat{s}$ is the element in L which saves the largest total number of bits:

$$\omega_0 = \arg\max_{\omega \in L} W(\omega)$$

Each element ω_n added to $\hat{s}$ subsequent to this is the best element in L, in terms of total number of bits saved, for explaining the remaining unexplained data $D - \kappa(\omega_0) \cup \cdots \cup \kappa(\omega_{n-1})$.

$$\omega_{n+1} = \arg \max_{\omega \in L, \kappa(\omega) \cap (\cup_{i=1}^{n} \kappa(\omega_i)) = \phi} W(\omega)$$

Pseudo-code for this algorithm is as follows:

Algorithm 4 *Greedy Estimate of* s_0.

$\hat{s} = \phi$

`repeat`

$\quad \hat{s} = \hat{s} \cup (\arg\max_{\omega \in L} W(w) + \psi(\omega, \hat{s}))$;

`until(`$D = \cup_{\omega \in \hat{s}} \kappa(\omega)$`)`;

where the function ψ is an overlap penalty used to avoid selecting elements of L which explain any of the "same" data.

$$\psi(\omega, s) = \begin{cases} 0 & \kappa(\omega) \cap (\cup_{\omega' \in s} \kappa(\omega')) = \phi \\ -\infty & else \end{cases}$$

When L contains D, this algorithm is guaranteed to terminate after at most $|D|$ iterations.

In practice, L may be so impoverished that the only versions available for some high level compositions overlap slightly with other compositions, even though there may well be other objects in $\Omega|D$ equivalent to these objects in terms of non-overlapping terminal coverage and overall win. One way of compensating for this is to design ψ so that it allows

some object overlap. Elements of $\hat{s}$ which overlap must now be taken as representative of other elements in $\Omega|D$ which would not overlap. For the OHRS application a penalty ψ_ν was used

$$\psi_\nu(\omega, s) = \begin{cases} 0 & \frac{\#(\kappa(\omega) \cap (\cup_{\omega' \in s} \kappa(\omega')))}{\#(\kappa(\omega))} \leq \nu \\ -\infty & else \end{cases}$$

where the parameter ν controls the percentage overlap allowed between a candidate composition and the pre-existing compositions in $\hat{s}$.

The sequential nature of algorithm (4) means it may fail in the presence of high level scene ambiguities. For example, using a list of letter compositions, it would fail to construct a good scene estimate for Figure 4.3 if a composition representing the letter U is selected before the ones which represent the L's.

Two approaches have been explored for solving this problem. In the first, *Multiple Starts*, Algorithm (4) is run repeatedly, with various compositions held out of L initially, in search of higher overall scene wins $\sum_{\omega \in \hat{s}} W(\omega)$. The second approach is even simpler: Here, *Higher Order Compositions* are designed into the grammar to accommodate any commonly occurring ambiguities for which there is a correct interpretation. For example, if string compositions of the appropriate type for "LL" were included in L, no high level ambiguity would exist, and Algorithm (4) would succeed in finding a good estimate for s_0.

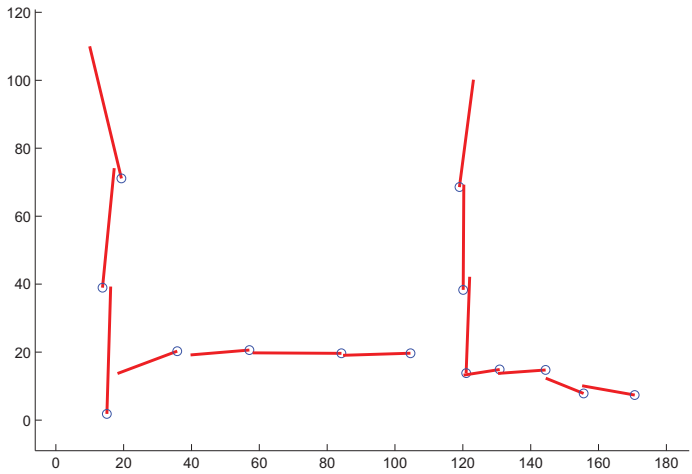

Figure 4.3: *Greedy Solution of (4.1) can get stuck in local maxima: When only letter and line compositions exist will $\hat{s}$ contain a U and a line or two L's?*

Chapter 5

Experiments

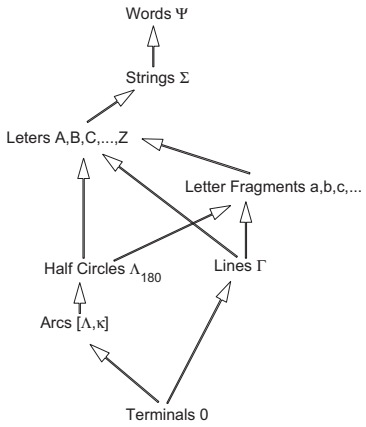

Figure 5.1: *The composition hierarchy used by the OHRS application.*

In this chapter we take up modeling and recognition of online handprint data. This application was selected to provide an informative and non-trivial example of the ideas developed earlier.

The space of possible objects which can be recognized is defined in a simple and quite natural way. Line-segment primitives are composed into longer lines and also arcs. In turn, these are incorporated into the definition of letter fragments and whole letters. Aggregations of letters are used to define strings, and finally strings are used to define words. A diagram of the overall composition hierarchy used is given in Figure 5.1. The use of an interpreted compositional distribution allows the rules of this construction to be defined in an essentially context-free manner –despite the relative complexity of geometric objects involved.

The prior used by the OHR system will be defined by first specifying an interpreted compositional grammar and distribution on object orbit representations Ω' and then extending this distribution to actual objects Ω'' by use of a generic placement function and distribution on placement function values. Example 4 and Example 6 in Chapter 1 are exemplify this approach.

Section 5.1 presents details on the definition of this prior. The compositions displayed in the figures used in this section were generated using the sampling algorithms described in Chapter 3. Recognition is taken up in Section 5.2. Here the ideas and algorithms described

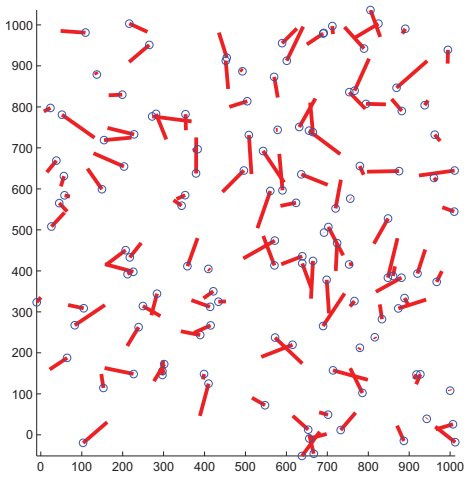

Figure 5.2: *Some elements of I(0).*

in Chapter 4 are applied to a small handprint data set–the cases considered are isolated characters, overlapping characters, strings of characters, and words.

5.1 Setup

Several remarks on the general structure of C', Ω' and Ω'' are in order. Compositions in Ω'' are built up from line segment primitives. Compositions in Ω' will be represent these objects up to scale, rotation and translation. Therefore, using the symbol 0 to represent line segment orbits, the set of terminals associated with Ω' will be

$$T = \{0\}$$

with interpretation

$$I(0) = R^2 \times R^2$$

See Figure 5.2.

Each composition in Ω' (and therefore Ω'') will either be monadic or dyadic. Interpretation functions for dyadic compositions $(l, v)(\alpha', \beta') \in \Omega'$ will make use of relation functions

of the form

$$f_l = (\theta_\beta - \theta_\alpha, \frac{r_\beta}{r_A}, R_{-\theta_\alpha}\frac{x_\beta - x_\alpha}{r_\alpha}) \tag{5.1}$$

where $(\theta_\alpha, \theta_\beta, r_\alpha, r_\beta, x_\alpha, x_\beta)$ are all l specific functions for angle, length, and position which are "covariant" with respect to $SE(2) \propto R_+$ in the sense that, for $\omega = l(\alpha, \beta)$, f_l is invariant to co-translation, co-rotation and co-scaling of α and β; i.e.

$$f_l(\alpha, \beta) = f_l(T\alpha, T\beta), \ \ \forall T \in SE(2) \propto R_+$$

The set of allowed relation function values associated with these compositions will always be

$$V = (-\pi, \pi] \times (0, \infty) \times R^2 \times R^2$$

Monadic compositions in Ω' will not make use of relation functions. The interpretation of $\omega = l(\alpha')$ will be defined as follows: If ω' has a single leaf node $\omega' = l_1(l_2(\cdots(l_k(t_0))\cdots)$, then $I(\omega') = \{l_1(l_2(\cdots(l_k(x))\cdots) : x \in I(t_0)\}$. Otherwise the interpretation of $l(\omega')$ is defined in terms of the first possible dyadic expansion of ω', e.g., if $\omega' = l_1(l_2((l_3, v)(\alpha', \beta')))$, $I(\omega') = \{l_1(l_2(l_3(\alpha, \beta))) : l_3(\alpha, \beta) \in I((l_3, v)(\alpha', \beta'))\}$.

The external distributions will be defined in terms of a discrete distribution $q(l)$ for $l \in T \cup N$, a set of densities $\{\nu_l(v)\}$ on relation function values $v \in V$, and a set of discrete probability distributions $\{Q_l\}_{l \in N}$ on binding function values.

Define N_2 as the set of dyadic composition indices in N, $N_2 \equiv \{l : (l, v)(\alpha', \beta') \in \Omega\}$. For $E \subseteq T \cup (N - N_2) \cup (N_2 \times V)$

$$Q(E) = \sum_{l \in E} q(l) + \sum_{l \in N_2} q(l) \int_{\{v:(l,v) \in E\}} \nu_l(v) dv$$

The relation function densities $\{\nu_l(v)\}$ share a common form. The components of $v = (\theta, r, x, y)$ are independent where θ is wrapped gaussian, r is log normal, and x, y are each gaussian. Label-dependent parameters $(\mu_\theta, \sigma_\theta, \mu_r, \sigma_r, \mu_x, \sigma_x, \mu_y, \sigma_y)$ determine the specific form of each of these densities.

5.2 Composition Rules and Relation Functions

A variety of attribute functions will be introduced for defining the binding functions used in the definition of Ω'.

The **c-index** D of a composition in Ω' returns the index of the composition rule used to create it. The c-index of a terminal $t \in \Omega'$ returns t itself

$$D(\omega') = l \text{ when } L(\omega') = l \times v \in N \times V, \text{ or } L(\omega') = l \in T \cup N$$

The **width** $\mid \omega \mid$ of an object is the number of its leaves

$$\mid \omega \mid = n \text{ when } Y(\omega) = (t_1, \cdots, t_n)$$

Lines. Line compositions in Ω' have c-index Γ. They are constructed using the binding function [1]

$$B_\Gamma(\nu^*) = \begin{cases} 1, & \nu^* = \alpha, D(\alpha) = 0 \\ |\alpha| + 1, & \nu^* = (\alpha, \beta), D(\alpha) = \Gamma, D(\beta) = 0 \\ -1, & else \end{cases}$$

with a set of allowed binding function values

$$S_\Gamma = \{1, 2, 3, \cdots\}$$

By including the width of each line composition in B_Γ, the marginal distribution on the number of line-segment primitives per line is controlled explicitly by Q_Γ. In practice Q_Γ was defined as a geometric distribution.

For dyadic line compositions, the relation function components $(\theta_\alpha, r_\alpha, x_\beta)$ are defined in terms of the endpoints of the right-most terminal α_+ in α; $(\theta_\beta, r_\beta, x_\beta)$ are defined by

[1] This is a polyadic binding rule – it causes some problems with the original definition for ICD which are ameliorated if one considers the simple generalization:

$$P'(\omega) = \begin{cases} Q(l) Q_l(B_l(\alpha'^*)) \frac{P'^*(\alpha'^*)}{P^*(B_l(\alpha'^*))} \nu_{l, B_l(\alpha^*)}(v) & \omega = (l, v)(\alpha'^*) \\ Q(t) & \omega = t \in T \end{cases} \tag{5.2}$$

Here, from the coding perspective, one can imagine sending l and B_l, orbits α' and β' and then the relative position f_l of α,β.

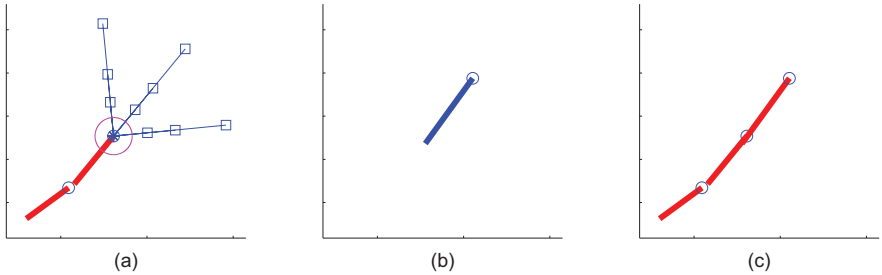

Figure 5.3: *(a)* α *(b)* β *(c)* $\Gamma(\alpha, \beta)$

the endpoints of β. Figure 5.3 contains an example line composition; panel (a) depicts an example α and a 3 sigma critical region of ν_Γ for acceptable $(\theta_\beta, r_\beta, x_\beta)$. Panel (b) shows a β whose $(\theta_\beta, r_\beta, x_\beta)$ lie within these regions. Panel (c) shows the ensuing composition.

More generally, one might want to construct lines using rules more elaborate than "line"+"line-segment" $\rightarrow$ "line", For example, a collinearity rule, "line" + "line" $\rightarrow$ "line", which composes pairs of collinear lines, might be advantageous. However, for the present application, the current approach has been satisfactory.

Arcs. A set of composition rules and relation functions are used to define arcs of varying mean curvature. Each is indexed by a discrete vector valued c-index $\{[\Lambda, \kappa] : \kappa \in -90, \cdots, 90\} \subseteq N$.

$$B_{[\Lambda,\kappa]} = (D(\alpha), |\alpha|, D(\beta))$$

$$S_{[\Lambda,\kappa]} = \{([\Lambda, \kappa], n, 0) : n > 1\}$$

$$f_{[\Lambda,\kappa]}(\alpha, \beta) = f_\Gamma(\alpha, \beta)$$

See Figure 5.4. The discrete parameter κ is used to control the mean angle change between line segments $(\theta_{\alpha_+} - \theta_\beta)$ by setting $\int_V \theta \nu_{[\Lambda,\kappa]}(\theta, r, x, y) d\theta dr dx dy = \kappa$. A geometric distribution on arc width and uniform distribution on allowed mean curvature changes was employed in the design of Q_Λ.

Half Circles. Half circles are examples of monadic composition rules

$$B_{\Lambda_{180}}(D(\alpha)) = (D(\alpha), |\alpha|)$$

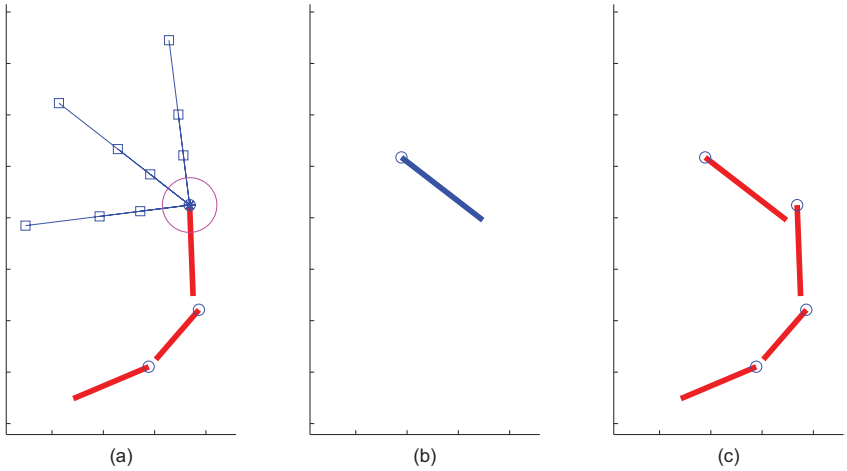

Figure 5.4: *(a)* α *(b)* β *(c)* $[\Lambda, 10](\alpha, \beta)$

$$S_{\Lambda_{180}} = \{([\Lambda, ceil(180/n)], n) : n > 2\}$$

where $ceil(x)$ rounds x up to the nearest integer, e.g., $ceil(1.7) = 2$. A geometric distribution $Q_{\Lambda_{180}}$ on half-circle width was employed.

Letters. The compositions for lines and half-circles are incorporated into the definition of the uppercase latin letters $\{A, B, C, \cdots, Z\}$ both directly and through a collection of "helper compositions". Letter and helper composition binding functions will either be dyadic and of the form

$$B_l(\alpha', \beta') = (D(\alpha'), D(\beta'))$$

or monadic and of the form

$$B_l(\alpha') = (D(\alpha'))$$

Sets of allowed binding function values S_l for some of these compositions are listed in column 2 of Table 5.3.

l	S_l	a_l, b_l, c_l, d_l
A	$\{(t, \Gamma)\}$	$5, (.5, 0), 1, (.5, 0)$
B	$\{P, a\}$	$9, (0, 0), 1, (00)$
C	$\{\Lambda_{180}\}$	
D	$\{(\Gamma, \Lambda_{180})\}$	$1, (.5, 0), 1, (.5, 0)$
$\vdots$		
O	$\{(\Lambda_{180}, \Lambda_{180})\}$	$1, (.5, 0), 1, (.5, 0)$
P	$\{(\Gamma, a)\}$	$1, (1, 0), 1, (1, 0)$
Q	$\{(O, \Gamma)\}$	$9, (.8, .2), 1, (0, 0)$
R	$\{(P, \Gamma)\}$	$9, (1, 0), 1, (0, 0))$
$\vdots$		
a	$\{b, \Gamma\}$	$9, (0, 0), 1, (0, 0)$
b	$\{(\Gamma, \Lambda_{180})\}$	$1, (1, 0), 1, (0, 0)$
t	$\{(\Gamma, \Gamma)\}$	
$\vdots$		

(5.3)

Column 3 of Table 5.3 contains a parametric encoding for the relation function associated with each of these compositions; i.e. (a_l, b_l, c_l, d_l) specify the relation function f_l associated with each dyadic composition type. Parameters a_l, c_l are used to index attribute functions g_{a_l} and g_{c_l} from Table 5.4 below. $g_{a_l}(\alpha)$ defines a "coordinate axis" with respect to α and $g_{c_l}(\beta)$ defines a "coordinate axis" with respect to β. The parameters $b_l \in R^2$ and $d_l \in R^2$ define "pivot points" x_α and x_β around which the relative position of these axes are allowed to vary. See Figure 5.5 .

For $g_{a_l}(\alpha) = AB$, $g_{c_l}(\beta) = CD$, and $x_\alpha = A + \|AB\| R_{<AB} b_l$, $x_\beta = C + \|CD\| R_{<CD} d_l$,

$$f_l(\alpha, \beta) = (<CD - <AB, \frac{\|CD\|}{\|AB\|}, R_{-<AB} \frac{x_\beta - x_\alpha}{\|AB\|})$$

This manner of expressing relation functions is quite convenient and is used throughout the computer implementation of the OHRS application. A total of 11 different attribute functions g_i were used in the design of the line, arc, latin letter, and helper compositions.

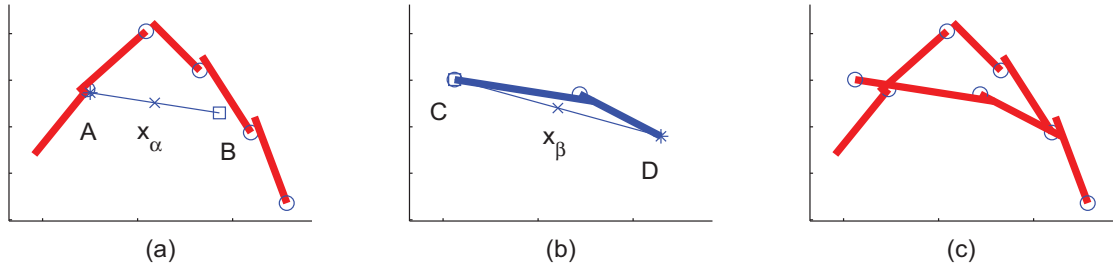

Figure 5.5: *The letter A is composed of a "tent" object and a line object. (a) Tent α with $g_{a_A}(\alpha) = AB$ and pivot point x_α. (b) Line β with $g_{c_A}(\beta) = CD$ and pivot point x_β. (c) $A(\alpha, \beta)$*

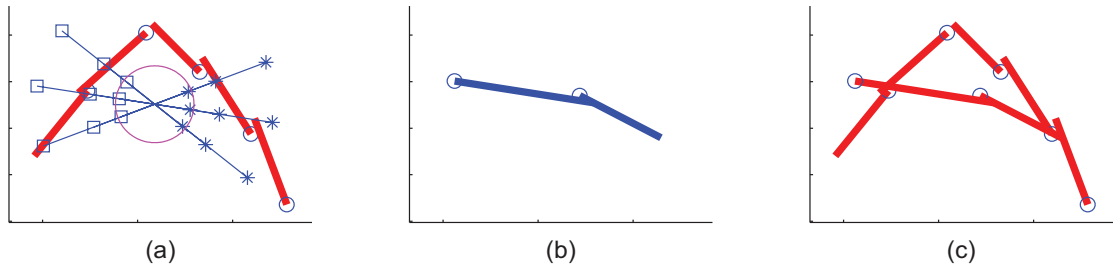

Figure 5.6: *(a) Displays critical regions for g_{c_A} and x_β with respect to α. (b) An example β. (c) Ensuing composition $A(\alpha, \beta)$.*

Thus, in the current application a fairly rich set of patterns is constructed using a limited repertoire of simple attribute functions.

Critical regions associated with the composition in Figure 5.5 are depicted in Figure 5.6.

i	$g_i(\omega)$
1	$\bar{\omega}$
2	$\bar{\omega}_-$
3	$\bar{\omega}_+$
4	$(mid(\bar{\omega}_2), mid(\bar{\omega}_2) + dif(\bar{\omega}_1))$
5	$(mid(\bar{\omega}_1), mid(\bar{\omega}_2))$
6	$((\bar{\omega_1})_2{}^1, (\bar{\omega_1})_2{}^2, ((\bar{\omega_1})_2{}^1, (\bar{\omega_1})_2{}^2)) + dif((\bar{\omega_1})_1))$
7	$(\bar{\omega_1}{}^3, \bar{\omega_1}{}^4, (\bar{\omega_1}{}^3, \bar{\omega_1}{}^4) + dif(\bar{\omega})$
8	$(mid(\bar{\omega_1}{}^3, \bar{\omega_1}{}^4, \bar{\omega_2}{}^1, \bar{\omega_2}{}^2), mid(\bar{\omega}))$
9	$\bar{\omega_2}$

(5.4)

The following notation and definitions are used in Table 5.4: Subscripts of ω' are used to select a subtree of ω relative to its root node. For $\omega' = l(\alpha', \beta')$, $\omega'_1 = \alpha'$, $\omega'_2 = \beta'$. ω'_- is the leftmost terminal of ω'. ω'_+ is the rightmost. The bar function $\bar{\omega'}$ pulls out the first coordinate pair of the leftmost terminal in ω' and the second coordinate pair of the rightmost terminal in ω'. $\bar{\omega'} \in R^2 \times R^2$. Superscripts index individual components of vectors in $R^2 \times R^2$. Precedence of these operators should be apparent: first the subscript(s) and then overbars are evaluated to yield elements of $R^2 \times R^2$. Functions mid and dif, respectively, define the midpoint and displacement of line segments. $mid((x_1, x_2), (x_3, x_4)) \equiv \frac{1}{2}(x_1 + x_3, x_2 + x_4)$. $dif((x_1, x_2), (x_3, x_4)) \equiv (x_3 - x_1, x_4 - x_2)$.

A variety of letter compositions are illustrated in Figures 5.7, 5.8, 5.9.

Strings. The c-index for strings compositions is Σ. Such compositions are built up recursively from other strings of letters and the uppercase latin letters using a **character yield** function $Y_c(\omega')$. When ω' is a letter composition ($D(\omega') \in \{A, \cdots, Z\}$) or ω' is a string composition ($D(\omega') = \Sigma$), $Y_c(\omega')$ is equal to the letters contained in ω' taken in left to right concatenation order. When ω' is not a letter or a string composition $Y_c(\omega) = 0$.

$$B_\Gamma(\nu^*) = \begin{cases} Y_c(\alpha') & \nu^* = \alpha' \\ (Y_c(\alpha'), Y_c(\beta')) & \nu^* = (\alpha', \beta') \end{cases}$$

$$S_\Sigma = \{A, \cdots, Z\} \cup \{(s, c) : s \in \{A, \cdots, Z\}^*, c \in \{A, \cdots, Z\}\}$$

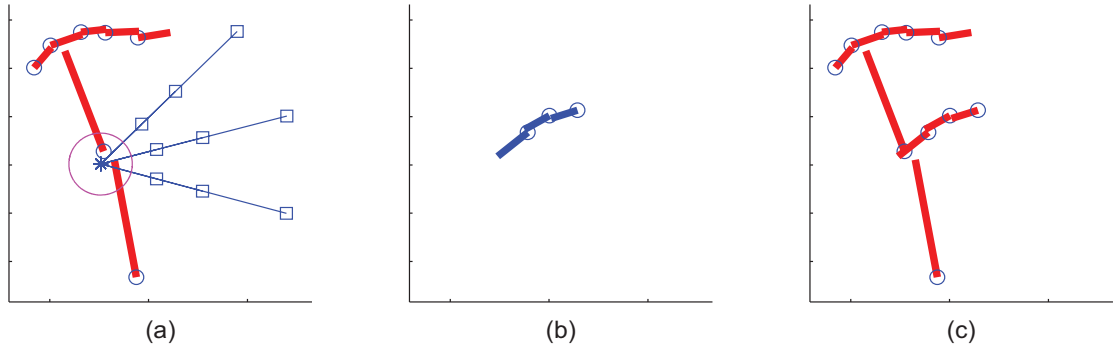

Figure 5.7: *Example letter F composition.*

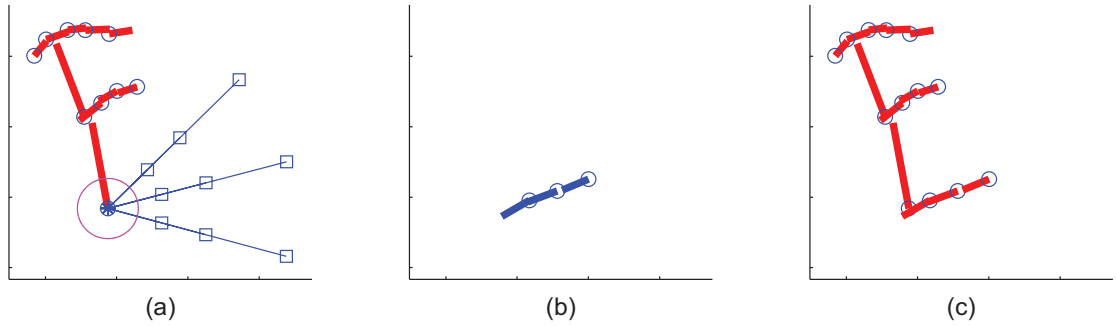

Figure 5.8: *Example letter E composition.*

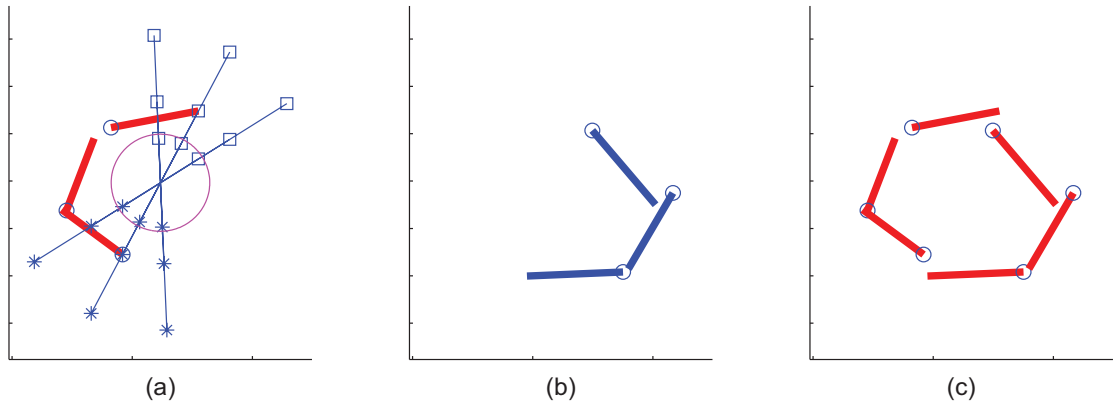

Figure 5.9: *Example letter O composition.*

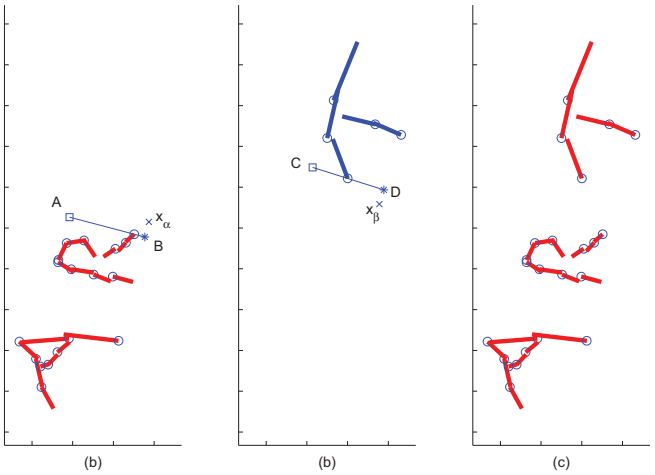

Figure 5.10: *(a) String α with $rs(\alpha) = AB$ and pivot point x_α (b) Character β with $ls(\beta) = CD$ and pivot point x_β (c) $\Sigma(\alpha, \beta)$*

The relation functions used are defined in terms of a pair of functions $rs(\alpha)$ and $ls(\beta)$ which define a right-hand side coordinate axis and a left-hand side coordinate axis for strings and characters; see Figure 5.10. These axes are encoded relative to one another in a relation function similar to the one for the dyadic letter compositions. For strings, Q_Σ is geometric on string length $|Y_c(\alpha)|$ and uniform on string contents.

Words. Words are monadic compositions of strings. Word compositions have c-index Ψ

$$B_\Psi(\alpha') = Y_c(\alpha')$$

$$S_\Psi = \{w : w \in Dictionary\}$$

where $Dictionary$ is some user supplied dictionary of words. For words Q_Ψ was taken to be uniform.

5.3 Object Probabilities and Encoding Wins

The placement function used for determining the position, scale and orientation of objects $\omega \in \Omega''$ within their orbits $I(\omega')$ will be defined generically by $g(\omega) = (\|AB\|, \measuredangle\, AB, A)$ where AB are the endpoints of the left-most terminal $\omega_{\_}$ in ω. The distribution on this

placement values was taken to be uniform

$$K(g(\omega)|\omega') = 1/Z_K$$

The general form for encoding wins for dyadic compositions is worked out in Section 2.3. Using relation functions of the form (5.1) and these generic placements functions, each Jacobian factor is of the form:

$$\frac{r_{\beta^0}}{r_\alpha^3}$$

where r_α is the l-specific scale term for α used in the construction of f_l and r_{β^0} is the l-specific scale term for β used in the construction of f_l evaluated on the unit object β^0. Therefore the local encoding win for a dyadic composition is

$$\log q(l)\frac{Q_l(B_l(\alpha',\beta'))}{P \times P(B_l(\alpha',\beta'))}\nu_l(f_l(\alpha,\beta))\frac{r_{\beta^0}}{r_\alpha^3}Z_K$$

For monadic compositions, encoding wins are given simply in terms of

$$\log q(l)\frac{Q_l(B_l(\alpha'))}{P(B_l(\alpha'))}$$

Evaluation of these wins is straightforward since the binding functions used only depend on the root label and binding function values of their constituents; i.e. in the case of dyadic compositions $P \times P(B_l) = Q \times Q(B_l)$ and in the case of monadic compositions $P(B_l) = Q(B_l)$.

5.4 Data Collection & Preprocessing

The online handprint data was collected using a Wacom ArtZ II digitizer tablet connected to an IBM-compatible PC. Each letter was entered by "writing" it on the digitizer tablet surface (with a special electronic stylus) while the PC recorded stylus tip-down and tip-up events as well as stylus tip coordinates. In this manner each letter entered was represented as a series of "strokes" consisting of stylus (x,y) locations ordered in time. For example, the letter B typically consists of two or three strokes. (In Figure 5.11, panel (a), $\times$ symbols

represent stylus locations; lines connect these points in the order in which they were digitized within each of the strokes present.)

This stroke data was then preprocessed to extract a series of line-segment primitives. (See Figure 5.11, panel (b).) The preprocessing algorithm was designed to minimize the number of line-segment primitives used to represent each stroke, while at the same time satisfying a penalty function designed to ensure that the line segment representation used remained reasonably faithful to the data.

First a fit of one line segment to a stroke is attempted. If such a fit exists then it will be used to represent that stroke; otherwise a fit of two line segments to the stroke is attempted. If such a fit exists then it will be used to represent the stroke; otherwise a fit of three line segments to the stroke is attempted, and so on and so forth.

Details of the approach are as follows:

1. The stroke data $(x_1, y_1), (x_2, y_2), \cdots$ is smoothed via low pass filtering and then represented as a smooth curve Z_s parameterized by arc-length s. ($Z_s \in R^2$)

2. Line segments are fit to this curve so that each of their endpoints lie along this curve. A pair of numbers (s, e) with $s < e$ represent the start s and end e positions of a line segment along Z.

3. Define e_0 and s_{N+1} so that Z_{e_0} is the first point along Z and $Z_{s_{N+1}}$ is the last point along Z. Line segments $(s_1, e_1), \cdots, (s_N, e_N)$ are fit to Z by solving (through dynamic programming) a discrete version of

$$\arg\min \Sigma_{i=1}^{N} \left\{ \| Z_{e_{i-1}} Z_{s_i} \|^2 + (s_i - e_i - \| Z_{s_i} Z_{e_i} \|)^2 \right\} + \| Z_{e_N} Z_{s_{N+1}} \|^2$$

   which includes a hard constraint on the maximum deviation allowed in the line-segment length versus arc length comparison term.[2]

Despite its ad hoc nature, this procedure does tend to reduce the number of line-segment primitives used to represent the data while at the same time retaining an accurate representation of strokes which contain interesting features such as the cleft in the curved stroke

[2]Specifically, solutions for which $(s_i - e_i - \| Z_{s_i} Z_{e_i s_{i+1}} \|)^2$ was greater than one-tenth $(s_i - e_i)$ were ignored.

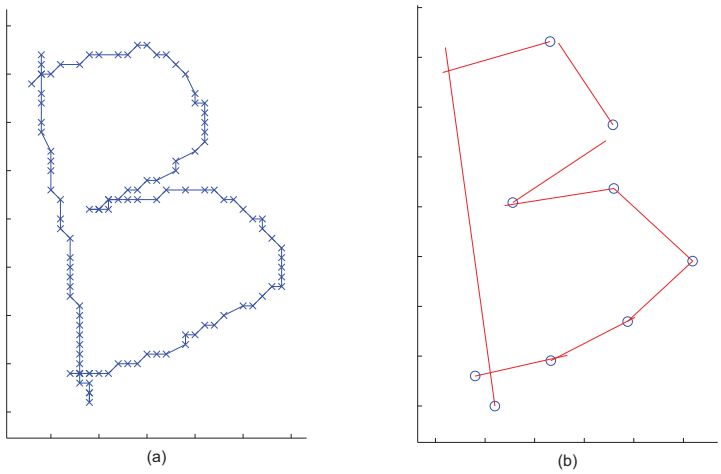

Figure 5.11: *(a) Raw data for a letter B; $\times$ symbols indicate digitized stylus locations points, lines connect these points in the order they were digitized. (b) Line segment representation derived from preprocessing the data in panel (a).*

for a B, e.g., Figure 5.11, and the small, but critical, right angle at one end of a stroke representing a G which allows us to disambiguate it from a C as in Figure 5.12.

5.5 Recognition

A version of the prioritized closure algorithm (algorithm 3) from Chapter 4 was used to test the efficacy of the prior for online handprint recognition.

The general approach was as follows: first, a model for the prior distribution on letters was developed through a hand-coding of composition rules, binding functions and distribution parameters. Then, a training data set consisting of 10 examples of each of the 26 uppercase handprint characters was collected and preprocessed. Figure 5.13 shows examples of some of the raw data.

The recognition procedure was then repeatedly run on this training data set as various parameters in the prior and recognition algorithm were varied (by hand) in an effort to increase recognition rates. All model and recognition system parameters were then fixed and a set of test data (consisting of an additional 10 examples of each of the 26 uppercase handprint characters) was collected and processed.

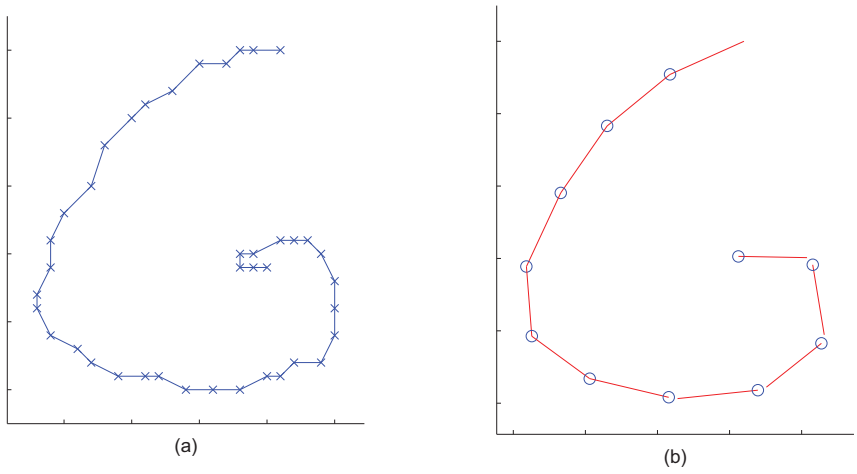

Figure 5.12: *(a) Raw data for a letter G; × symbols indicate digitized stylus locations points, lines connect these points in the order they were digitized. (b) Line segment representation derived from preprocessing the data in panel (a).*

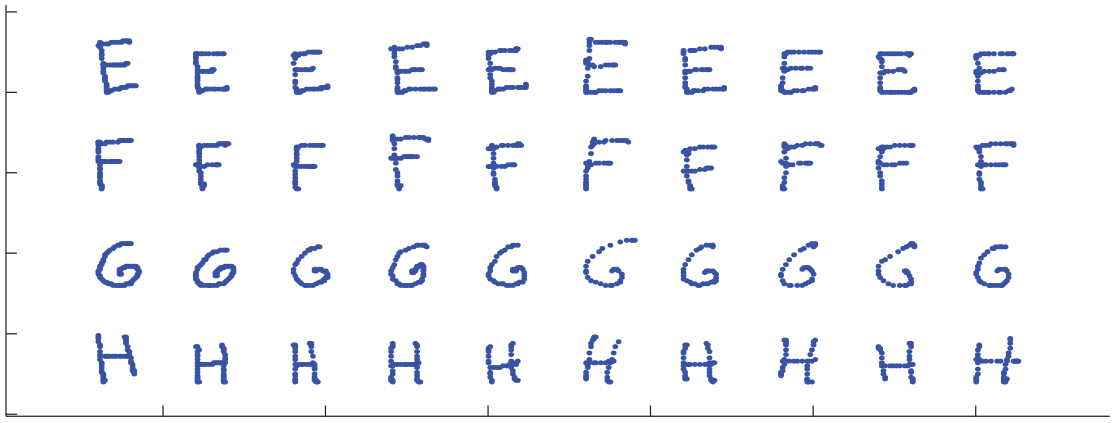

Figure 5.13: *Raw Data.*

The overall character recognition rate was 93.08% for the test data set and 92.69% for the training data set. Rates for individual characters are given in Table 5.5.

Letter	$\#Correct(Train)$	$\#Correct(Test)$
A	10	9
B	10	9
C	10	10
D	8	10
E	10	10
F	10	9
G	8	7
H	10	10
I	10	10
J	5	7
K	10	10
L	10	10
M	10	6
N	10	9
O	9	10
P	10	9
Q	9	10
R	9	10
S	10	10
T	10	10
U	5	8
V	10	10
W	10	10
X	10	9
Y	8	10
Z	10	10

(5.5)

The most commonly confused characters are U, J and V. This is expected, since in isolation such characters are in fact naturally ambiguous with one another. Addition of

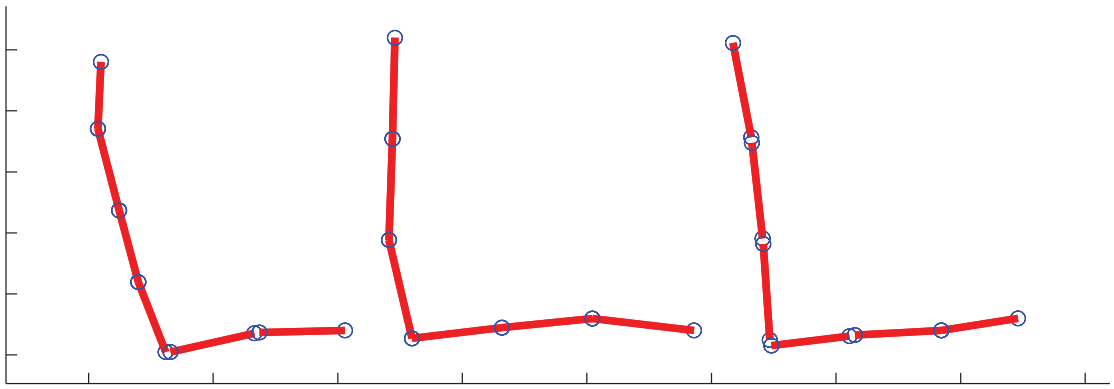

Figure 5.14: *Successfully recognized string.*

higher level compositions (such as whole word compositions) allows such ambiguities to be eliminated.

The version of the prioritized closure algorithm used included two hard constraints. First, no objects were allowed to compose with any part of themselves. Second, critical region tests were used as a pruning heuristic to avoid highly unlikely compositions.

These constraints, and the small number of terminals per character, allowed the use of an exhaustive candidate selection heuristic. The total number of compositions in the final instance list L was on the order of several hundred to several thousand possible compositions per character.

For word and string recognition the number of possible elements in L (even with these constraints) is typically too large to allow for use of an exhaustive binding heuristic. In order to process such cases, an ad hoc binding candidate selection heuristic was used whereby first all possible letter compositions are generated. Then the top 100 possible candidates for composition are selected and allowed to form strings. Then the top 90 possible candidates for composition are selected and allowed to form strings and words. Then the top 80, and so on and so forth. This method correctly interprets the strings and words such as the ones displayed in Figure 5.14, Figure 5.15 and Figure 5.16.

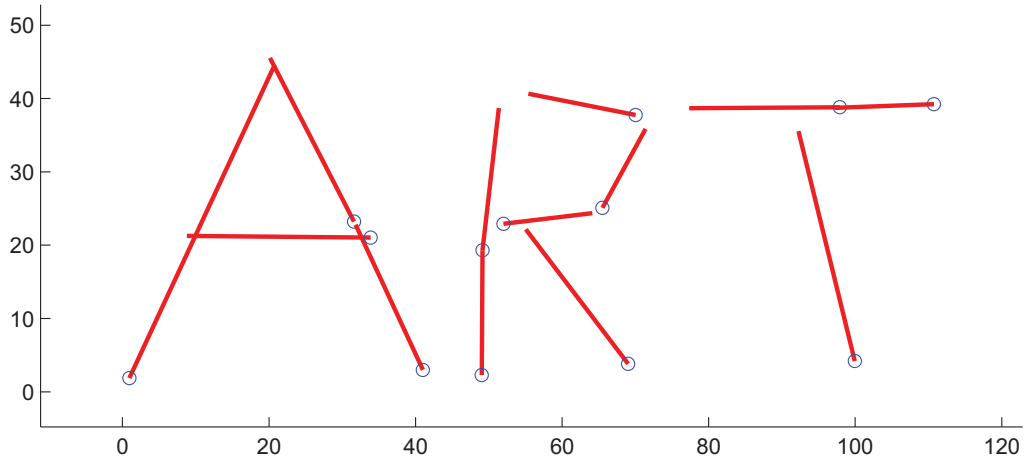

Figure 5.15: *Successfully recognized word.*

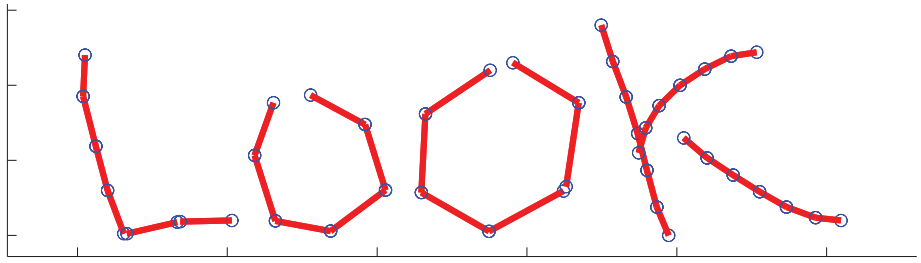

Figure 5.16: *Successfully recognized word.*

Chapter 6

Conclusion

In this thesis, I have attempted to present an analytic framework and collection of algorithms useful for pattern modeling and recognition. While the work seems promising, several issues remain, including the need for an automatic inference/parameter estimation procedure and more efficient recognition algorithms.

Some general highlights of the analytic framework developed are as follows:

1. Rich Pattern Classes. A small number of composition rules and parameters can be used to define rich set of flexible yet coherent patterns. An example of this was seen in Chapter 5 when a prior on handprint letters, strings, and words was developed.

2. Meaningful Parameters. Compositional and interpreted compositional distributions have marginals on labels and binding function values which agree with joint distribution on label and binding function values defined by the external distributions Q and Q_l. Under appropriate conditions, it should be straightforward to estimate parameters for Q and Q_l from data.

3. Scale-, Translation-, and Rotation-Invariant Models. The relation functions used in the online handprint application build in the sorts of natural invariances one would hope for, in the sense that, one can learn the definition of an object at a single scale, orientation, and position (or from examples at disparate variety of scales) and then extend this to other scales, orientations, and positions.

4. Sampling. Under certain conditions, the Recursive-Sample-Reject Algorithm developed in Chapter 3 may be used to sample compositional and interpreted compositional distributions. Many of the figures used to illustrate this thesis were generated using the output of this sampling procedure, including all of the figures in Chapter 5 used to illustrate various components of the prior used for online handprint recognition.

5. Exact Probability Calculations. Certain interesting classes of compositional and interpreted compositional distribution admit exact probability calculations, i.e., their use is not hampered by presence of unknown partition function value. The interpreted compositional distribution developed for the online handprint application is an example of one such distribution.

In Chapter 4 several algorithms for pattern recognition were proposed, and in Chapter 5 they were used to develop an online handprint recognition system. While sufficient to achieve reasonable recognition rates for isolated handprint letters and even whole words it is clear that these algorithms, in their present form, do not scale well. The total number of line-segment terminals presented as input to these algorithms was always quite small (at most 20 or 30).

One avenue for further research would be to attempt to construct some sort of dynamic programming/chart/table based parsing algorithm for scene interpretation. Unfortunately the lack of *a priori* meaningful object concatenation order and the generality of the binding functions allowed make this a daunting prospect. Another (potentially complementary) approach would be to attempt to build some notion of "inhibition" into the system, as described at the end of the section on object instancing in Chapter 4.

In both cases it seems that restricting the class of binding and relation functions used will be essential. Linguists, as the current main users of constraint-based grammars, have already recognized the need for this. In their applications, binding functions expressed as logical expressions involving the notion of *unification* [15] form the basis for a general theory on parsing [19], and for a number of computability results.

From the modeling perspective, the field is wide open. (For the online handprint recognition system all composition rules and parameters were hand-coded.) A first step in this direction would be to employ a "supervised" learning scheme in which the compositional rules employed are given, and the relation function and external distribution parameters estimated. Of course further down the line one would want to infer the composition rules as well. In this case a scheme motivated by the random decision tree generation approach used in [1] may be of some use.

Also, in terms of the online handprint prior, it would be interesting to expand the orbit representations used to allow for the use of higher dimensional group structures. For example, one might want to represent objects up to the translation and special affine group.

Bibliography

[1] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computations*, 9, 1997.

[2] Elie Bienenstock, Stuart Geman, and Daniel Potter. Compositionality – mdl priors and and object recognition. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 838. The MIT Press, 1997.

[3] T.L. Booth and R.A. Thompson. Applying probability measures to abstract languages. *IEEE transactions on Computers*, 22:442–450, 1973.

[4] John Canning. A minimum description length model for recognizing objects with variable appearances (the vapor model). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[5] Zhiyi Chi. *Probability Models for Complex Systems.* PhD thesis, Brown University, Division of Applied Mathematics, 1998.

[6] Noam Chomsky. *Syntactic structures.* Janua linguarum. Mouton, The Hague, 1957.

[7] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley, 1991.

[8] King Sun Fu. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, 1982.

[9] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[10] Stuart Geman. Invariant binding in composition systems. Technical report, Brown University, Division of Applied Mathematics, 1999, In preparation.

[11] Stuart Geman, Daniel Potter, and Zhiyi Chi. Composition systems. Technical report, Brown University, Division of Applied Mathematics, 1998.

[12] Ulf Grenander. Syntax controlled probabilites. Technical report, Brown University, Division of Applied Mathematics, 1967.

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

[14] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–30, 1957. Continued in volume 108, pages 171-190.

[15] Kevin Knight. Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1):93–124, March 1989.

[16] Donald E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):127–145, 1968.

[17] Pierre-Simon Laplace. *Essai Philosophique sur les Probabilités*. 1820. An English translation is *Philosophical Essay on Probabilities*, Dover Publications, New York, 1951.

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

[19] Stuart M. Shieber. *Constraint-based grammar formalisms*. MIT Press, Cambridge, Massachusetts, 1992.