A Stochastic Attribute Grammar Model of Document Production and its use in Document Image Decoding

Philip A. Chou and Gary E. Kopec

Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304.

ABSTRACT

Document Image Decoding (DID) refers to the process of document recognition within a communication theory framework. In this framework, a logical document structure is a *message* communicated by *encoding* the structure as an ideal image, *transmitting* the ideal image through a noisy channel, and *decoding* the degraded image into a logical structure as close to the original message as possible, on average. Thus document image decoding is document image recognition where the recognizer performs optimal reconstruction by explicitly modeling the source of logical structures, the encoding procedure, and the channel noise. In previous work, we modeled the source and encoder using probabilistic finite-state automata and transducers. In this paper, we generalize the source and encoder models using context-free attribute grammars. We employ these models in a document image decoder that uses a dynamic programming algorithm to minimize the probability of error between original and reconstructed structures. The dynamic programming algorithm is a generalization of the Cocke-Younger-Kasami parsing algorithm.

Keywords: document recognition, document formatting, stochastic grammars, two-dimensional parsing, model-based recognition, Bayesian analysis, image defect modeling

1 INTRODUCTION

In a previous paper[1] we outlined a communication theory framework for document recognition. In the communication theory view, logical document structure is communicated by encoding the structure as an ideal image and transmitting the image through a noisy medium (paper). The document recognizer is essentially a decoder that inverts the effects of the encoder and noisy channel. Ambiguities are resolved by knowledge of the prior probabilities of different logical structures. Since in this framework document recognition is the inverse of document production, the recognizer incorporates within it an explicit model of document production. In this paper, we show how a document recognizer can be based on a stochastic attribute grammar model of document production.

To be more precise, in the communication theory view, depicted in Figure 1, a person, or *source*, randomly produces a message, or logical document structure X; an imager, or *encoder*, deterministically encodes X into an ideal bitmapped image Y; a *channel*, such as a printer, a scanner, and everything in between, randomly

degrades Y into an observed bitmapped image Z; and a recognizer, or a *decoder* deterministically maps Z into a reconstructed logical document structure $\hat{X}$.

The optimal decoder in this framework is the deterministic function g that minimizes the expected error between the source message X and the reconstructed message $\hat{X}$. That is, the optimal g minimizes $E\ell(X, g(Z))$, where $\ell(X, \hat{X})$ is a loss function appropriate for the problem. For example, a loss function appropriate for the problem of document recognition may be an editing distance between X and $\hat{X}$. A simple special case is the 0-1 loss function

$$\ell(X, \hat{X}) = \begin{cases} 0 & \text{if } X = \hat{X} \\ 1 & \text{otherwise} \end{cases} .$$

For this loss function, it is well known that the optimal decoder g is the *maximum a posteriori* (MAP) rule

$$\hat{X} = g(Z) = \arg\max_X P(X|Z), \tag{1}$$

where $P(X|Z)$ is the posterior probability of X given Z. That is, the optimal decoder chooses the most likely underlying document structure X based on the observed noisy scanned image Z. But by Bayes' rule, $P(X|Z) = P(X, Z)/P(Z) = \sum_Y P(X, Y, Z)/P(Z)$, and since X, Y, Z is a Markov chain, $P(X, Y, Z) = P(X)P(Y|X)P(Z|Y)$. Furthermore, since X is encoded deterministically as $Y = f(X)$ for some function f, $P(Y|X)$ equals 1 if $Y = f(X)$ and equals 0 otherwise. Hence the MAP rule (1) becomes

$$\hat{X} = g(Z) = \arg\max_X P(X)P(Z|f(X)). \tag{2}$$

To implement the optimal decoder g in a real-world document recognition system, it is fundamentally important to

1. realistically model the source $P(X)$, the encoder $f(X)$, and the channel $P(Z|Y)$; and
2. efficiently perform the arg max over all X.

In this paper, we identify X as a parse tree, $P(X)$ as a stochastic attribute grammar, and $f(X)$ as a syntax-directed translation. These are particularly good choices, for two reasons. First, attribute grammars are fairly realistic models of document production; they have actually been used to format (and edit) equations,[2,3] graphical objects,[4] and, to some extent, more general documents.[5–8] Second, attribute grammars invite the use of compiler technology to efficiently perform the arg max. Indeed, the decoder can be regarded as a parser for the observed image Z. By choosing $P(Z|Y)$ to be an independent binary noise, the arg max can be specified recursively. This leads immediately to a dynamic program for the optimal decoder g. This dynamic program is a generalization of the Cocke-Younger-Kasami parsing algorithm.

We now discuss the components of Figure 1 in more detail.

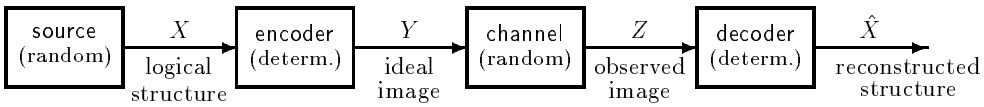

Figure 1: Communication Theory View of Document Recognition

2 THE SOURCE

The source is a stochastic context-free or regular grammar $G = (\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$, where $\mathcal{N} = \{A, B, C, \ldots\}$ is a set of nonterminal symbols, $\mathcal{T} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots\}$ is a set of terminal symbols, $\mathcal{P} = \{A \to \sigma, B \to \tau, \ldots\}$ is a set of production rules, and $S \in \mathcal{N}$ is the start symbol. Associated with each production rule $A \to \sigma$ is a probability $P(A \to \sigma)$ such that $\sum_\sigma P(A \to \sigma) = 1$ for each left hand side $A \in \mathcal{N}$. This naturally induces a probability distribution on parse trees X by $P(X) = \prod P(r_i)$, where r_i is the rule used at the ith internal node of the parse tree. It is easy to verify that $\sum_X P(X) = 1$, unless G is a degenerate grammar that allows derivations of infinite length. A parse tree can be generated at random by placing the start symbol at the root and selectively applying production rules according their probabilities until only terminal symbols remain at the leaves. The resulting tree can be regarded as the logical document structure produced by the source.

3 THE ENCODER

The encoder is a deterministic mapping f from parse trees X to bitmapped images Y, called a syntax directed translation. A syntax directed translation associates with each symbol in the grammar a collection of attributes, each of which may be either *synthesized* or *inherited*, and it associates with each production rule a collection of functions, one function for each synthesized attribute on the left side of the rule and one for each inherited attribute on the right side of the rule. These functions take as arguments the other attributes appearing in the rule, that is, the inherited attributes on the left and the synthesized attributes on the right. Such a grammar is called an *attribute grammar*. In any parse tree in an attribute grammar, each instance of a symbol maintains its own attribute values. Thus, given a parse tree X, and given the values of the inherited attributes at the root and the values of the synthesized attributes at the leaves, the values of all the remaining attributes in the tree can be computed by using the functions associated with the production rules at the internal nodes in an appropriate sequence.

For example, in one version of a document encoder model*, we associate with each symbol synthesized *box metric* attributes, inherited *coordinate transformation* attributes, and synthesized *bitmap* attributes, which we now describe.

The box metric attributes of a symbol extend the notion of font metrics for a single character to metrics for a spatially localized collection (or box) of characters that are represented by that symbol. In PostScript, the font metrics for a character consist of three two-dimensional vectors. The first two vectors, (x_l, y_b) and (x_r, y_t), represent the left bottom and right top corners of the bounding box of the character in its local (character) coordinate system, as shown in Figure 2. The third vector, (x_e, y_e), represents the endpoint of the character, which is typically the location of the origin of the next character in the coordinate system of the current character, as shown in Figure 3. Bounding boxes may overlap. The line through the origin along (x_e, y_e) is the *baseline* of a character. The local coordinate system of the character is oriented with x increasing to the right and y increasing upward, and is scaled so that unity equals the pointsize of the font, i.e., the nominal spacing between baselines in close-set type. The box metric attributes for a symbol when the symbol is a single character (i.e., when it is a terminal symbol) are precisely the font metrics for the character. The box metrics for a symbol when the symbol represents a collection of characters, such as a paragraph or paragraph fragment (i.e., when it is a nonterminal symbol) consist also of vectors (x_l, y_b) and (x_r, y_t) for the bounding box of the characters in a local coordinate system, and usually also the vector (x_e, y_e), representing a distinguished point where the origin of the "next" symbol (i.e., collection of characters) should be positioned. Sometimes additional vectors can be associated with a symbol, indicating other geometrically distinguished points, such as the locations of various character baselines. The scale of the local coordinate system for collection of characters is typically set so that unity equals the predominant point size in the collection. The box metric attributes of a symbol can be

*In this version logical and layout structures are essentially equivalent.

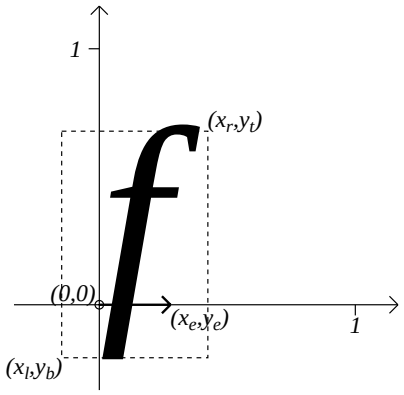

Figure 2: Font metric information.

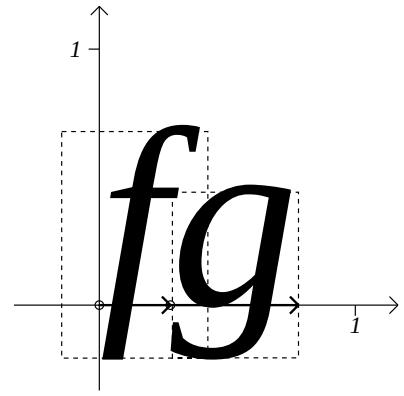

Figure 3: Character concatenation.

represented by an array

$$F = \begin{bmatrix} (x_l, y_b) \\ (x_r, y_t) \\ (x_e, y_e) \\ \vdots \end{bmatrix}.$$

The box metrics for all the symbols in a parse tree can be computed bottom up, given the font metrics at the leaves. For example, at any internal node of the tree using a rule $A \to B_1 \cdots B_M$ to lay out the B_is along some baseline, the box metrics of A can be computed from the box metrics of the B_is using the following functions associated with the rule:

$$\begin{aligned} A.x_e &= \sum_i B_i.x_e \\ A.y_e &= \sum_i B_i.y_e \end{aligned}$$

$$
\begin{aligned}
A.x_l &= \min_i \left\{ B_i.x_l + \sum_{j<i} B_j.x_e \right\} \\
A.y_b &= \min_i \left\{ B_i.y_b + \sum_{j<i} B_j.y_e \right\} \\
A.x_r &= \max_i \left\{ B_i.x_r + \sum_{j<i} B_j.x_e \right\} \\
A.y_t &= \max_i \left\{ B_i.y_t + \sum_{j<i} B_j.y_e \right\}
\end{aligned}
$$

That is, the endpoint vector for A is the sum of the endpoint vectors for the B_is, and the bounding box for A is the overall bounding box for the B_is, when their endpoint vectors are laid head to tail.

The coordinate transformation attributes of a symbol represent the coordinate transformation (which is typically affine)

$$
G : (x, y) \mapsto (x, y) \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + (x_0, y_0),
$$

that maps each vector in the symbols's local coordinate system into the *global* coordinate system (also called the *root*, or *page* coordinate system), which is defined arbitrarily to be located at the left bottom corner of the first page, with x increasing to the right and y increasing upward, and scaled so that unity equals 1 point. Any such affine transformation can be represented by six parameters. It may suffice in practice to represent G by two, three, or four parameters, depending on whether G can be restricted to a simple translation, a uniform scaling followed by a translation, a uniform scaling followed by a rotation and a translation, or a non-uniform scaling (independent in x and y) followed by a translation. More generally, of course, G could be some non-linear transformation and could require a large number of parameters for its specification. A possible additional piece of global coordinate transformation information is the page number. Page numbers can be interpreted as integer translations along a third axis, whose origin is located on page 0. The coordinate transformation attributes for all the symbols in a parse tree can be computed top down, given the coordinate transformation at the root. For example, at the internal nodes of the tree using a rule $A \to B_1 \cdots B_M$ to lay out the B_is along some baseline, the coordinate transformation of the B_is can be computed from the coordinate transformations of A using the following functions associated with the rule:

$$
\begin{aligned}
B_i.a_{mn} &= A.a_{mn} \\
B_i.x_0 &= A.x_t + \sum_{j<i} B_j.x_e \\
B_i.y_0 &= A.x_t + \sum_{j<i} B_j.y_e
\end{aligned}
$$

That is, the scaling matrices for the B_is equal the scaling matrix for A, and the translations for the B_is equal the translation for the A plus the endpoints of all the previous B_js.

The bitmap attribute for a symbol is a bitmap of an ideal, page-sized image (at least conceptually). The bitmap attributes for all the symbols in a parse tree can be computed bottom up, given the coordinate transformations of every leaf. In principle this can be done by rendering or imaging each leaf into its bitmap at the appropriate coordinate transformation, and then ORing the bitmaps up through the tree. The bitmap at the root then becomes the ideal image Y. In practice, this computation can be done simply by rendering the leaves one at a time onto a single page-sized image.

This defines the deterministic encoding f of a parse tree X into an ideal binary image Y by means of a syntax-directed translation.

4 THE CHANNEL

The channel is an independent binary asymmetric channel $P(Z|Y)$ in which every white bit in the ideal image flips to black with probability ϵ_0 and every black bit flips to white with probability ϵ_1. Thus

$$P(Z|Y) = \prod_{i:y_i=0} \epsilon_0^{z_i}(1-\epsilon_0)^{(1-z_i)} \cdot \prod_{i:y_i=1} \epsilon_1^{(1-z_i)}(1-\epsilon_1)^{z_i}.$$

In particular, if $\emptyset$ is the all-white background image $Y = 0$, then the normalized probability

$$\tilde{P}(Z|Y) \equiv \frac{P(Z|Y)}{P(Z|\emptyset)} = \prod_{i:y_i=1} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)\left(\frac{(1-\epsilon_0)(1-\epsilon_1)}{\epsilon_0\epsilon_1}\right)^{z_i}$$

is a product over only those pixels that are black in the ideal image. This normalization leads to the following decomposition property. If the ideal image Y is composed of disjoint black regions Y_i, then $\tilde{P}(Z|Y) = \prod_i \tilde{P}(Z|Y_i)$.

5 THE DECODER

The optimal decoder is given by (2), as we have seen. We can normalize $P(Z|f(X))$ in (2) by $P(Z|\emptyset)$, which does not depend on X, to obtain

$$\hat{X} = g(Z) = \arg\max_X P(X)\tilde{P}(Z|f(X)). \tag{3}$$

Then by the decomposition property, $\tilde{P}(Z|f(X)) = \prod_i \tilde{P}(Z|f(X_i))$, where X_i is the ith subtree of the root of X, and $f(X_i)$ is its corresponding ideal subimage. (The $f(X_i)$s are assumed disjoint.) Furthermore, since $P(X)$ is the product of the rule probabilities at the internal nodes of X, it too can be recursively decomposed as $P(X) = P(A \to B_1 \cdots B_M)\prod_i P(X_i)$, where $A \to B_1 \cdots B_M$ is the rule at the root of X. The product $\tilde{P}(Z,X) = P(X)\tilde{P}(Z|f(X))$ thus inherits these decompositions:

$$\tilde{P}(Z,X) = \left[P(A \to B_1 \cdots B_M)\prod_{i=1}^{M} P(X_i)\right] \cdot \left[\prod_{i=1}^{M} \tilde{P}(Z|f(X_i))\right] = P(A \to B_1 \cdots B_M)\prod_{i=1}^{M} \tilde{P}(Z,X_i).$$

The maximum in (3), taken over all parse trees X having at its root the start symbol A with particular attributes α (written $A.\alpha$ or A^α), can therefore be expressed recursively as

$$\begin{aligned}
\tilde{P}_{\max}(Z|A^\alpha) &\equiv \max_{X:\mathrm{root}(X)=A^\alpha} \tilde{P}(Z,X) \\
&= \max_{\substack{B_1^{\beta_1}\cdots B_M^{\beta_M} \\ \text{s.t. } A^\alpha \to B_1^{\beta_1}\cdots B_M^{\beta_M}}} \quad \max_{\substack{X_1,\ldots,X_M \\ \text{s.t. } \mathrm{root}(X_i) = B_i^{\beta_i}}} \tilde{P}(Z,X) \\
&= \max_{\substack{B_1^{\beta_1}\cdots B_M^{\beta_M} \\ \text{s.t. } A^\alpha \to B_1^{\beta_1}\cdots B_M^{\beta_M}}} P(A \to B_1 \cdots B_M)\prod_{i=1}^{M} \tilde{P}_{\max}(Z|B_i^{\beta_i}).
\end{aligned}$$

$\tilde{P}_{\max}$ can be found using a dynamic programming algorithm with complexity $O(\sum_{A^\alpha \to B_1^{\beta_1}\cdots B_M^{\beta_M}} M)$. The maximizing parse tree itself can be found in the same time by keeping track of the maximizing production rules. In the complexity figure, the sum is over all possible combinations of production rules and attribute value combinations. This makes sense only if the attributes are discrete and finite. In practice this is the case. The box metric attributes (x_l, x_b), (x_r, x_t), and (x_e, y_e) suggested in Section 3 are typically restricted to integer pixel values, as are

the coordinate translation attributes (x_0, y_0). The coordinate scaling attributes $a_{11} = a_{22}$ are typically restricted to integer pointsizes, with shear and skew attributes $a_{21} = a_{12} = 0$. To compute the complexity in this case, recall that for each production rule, the number of attribute value combinations is at most the number of possible values of inherited attributes on the left side of the rule and synthesized attributes on the right side, since the other attributes are functions of these. Thus for a context-free grammar in Chomsky normal form (for which M is at most 2), the complexity is at most the number of coordinate transformations for the left side symbol, times the square of the number of box metric combinations for a right side symbol, times the number of production rules. The number of coordinate transformations is N^2 possible displacements times K possible scales, where N is the linear dimension of the image in pixels and K is the number of allowed pointsizes. The number of box metric combinations is $(N^2)^3 = N^6$. Thus the overall complexity is $O(N^2K(N^6)^2) = O(N^{14}K)$, i.e., the number of image pixels to the seventh power. This complexity is actually feasible with chart-style evaluation and sufficient pruning. Alternatively, the complexity can be made feasible by reducing the collection of attributes. If bounding boxes are irrelevant (i.e., only endpoints matter in layout), then the complexity reduces to $O(N^2K(N^2)^2) = O(N^6K)$, cubic in the number of image pixels. This is essentially a reduction to standard string parsing. (If only the width and height of the bounding boxes are relevant, and the endpoints do not matter, then the complexity is again $O(N^6K)$. This is essentially a reduction to the algorithm used in our work on equation image recognition.[9]) If the grammar is regular, then at least one symbol on the right side of every rule is a terminal symbol, and its box metrics (its synthesized attributes) are constant. Furthermore, in a left linear grammar the coordinate translation (inherited attributes) of the left side is constant. Hence the complexity reduces to $O(KN^2)$, linear in the number of image pixels. This is essentially a reduction to the Viterbi algorithm, as presented in our previous paper.[1]

6 SUMMARY

In the communication theory view of document recognition, document recognition is the inverse of document production. Optimal recognition is carried out with reference to an explicit production model. In this paper, we show how stochastic attribute grammars can be used as the model. Not only are attribute grammars realistic production models, but they lead to tractable algorithms for optimal decoding. These algorithms are defined in terms of recurrence relations over the attribute values. The computational complexity of the algorithms ranges from linear to the seventh power of the number of image pixels (or more), depending on the complexity of the grammar and its attributes.

7 REFERENCES

[1] G. E. Kopec and P. A. Chou. Document image decoding using Markov source models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(6):602–617, June 1994.

[2] B. W. Kernighan and L. L. Cherry. A system for typesetting mathematics. Technical report, Bell Laboratories, Murray Hill, NJ, 1975.

[3] P. Franchi-Zannettacci. Attribute specifications for graphical interface generation. In *Proc. XI World Computer IFIP Congress*, San Francisco, CA, August 1989.

[4] L. A. Barford and B. T. Vander Zanden. Attribute grammars in constraint-based graphics systems. *Software Practice and Experience*, 19(4):309–328, April 1989.

[5] T. Reps and T. Teitelbaum. *The Synthesiser Generator Reference Manual.* Department of Computer Science, Cornell University, 1985.

[6] V. Donzeau-Gouge, G. Huet, G. Kahn, and B. Lange. Programming environments based on structured editors: the MENTOR experience. In D. Barstow, E. Sandewall, and H. Shrobe, editors, *Interactive Programming Environments*, pages 128–140. McGraw Hill, New York, NY, 1984.

[7] P. Franchi-Zannettacci and D. S. Arnon. Context-sensitive semantics as a basis for processing structured documents. In *Workshop on Object-Oriented Document Manipulation*, pages 135–146, Rennes, France, May 1989. BIGRE.

[8] D.S. Arnon, P. Franchi-Zannettacci, and P.A. Chou. Document layout by sequential attribute grammars. *Journal of Mathematical and Computer Modeling*, October 1994. Submitted.

[9] P. A. Chou. Recognition of equations using a two-dimensional stochastic context-free grammar. In *Visual Communications and Image Processing*, pages 852–863, Philadelphia, PA, November 1989. SPIE.