Separable Source Models for Document Image Decoding*

Anthony C. Kam
Caliper Corp.
Boston, MA
antkam@athena.mit.edu

Gary E. Kopec
Xerox PARC
Palo Alto, CA
kopec@parc.xerox.com

1 ABSTRACT

This paper defines separable models and examines their relationship to other types of Markov source models used for document image decoding. Loosely, a separable source is one that may be factored into a product of 1-dimensional models that represent horizontal and vertical structure, respectively. More formally, a separable model is a collection of Markov subsources that is similar to a recursive transition network. Associated with some of the nodes are position constraints that restrict entry to the nodes to certain regions of the image plane. The top-level subsource is a vertical model whose nodes are all tightly constrained to specific horizontal positions. The importance of separable models is that they form the basis for fast decoding algorithms based on heuristic search. In many situations, the natural structure of a class of document images leads the user to define a model that is recursive, but not necessarily separable. We describe an algorithm for converting a recursive source into an equivalent separable form, if one exists. A key factor determining the success of source separation is the number of tightly constrained nodes. We present a procedure for propagating user-supplied position constraints, typically given for a small subset of nodes, to the remaining nodes of the model.

2 INTRODUCTION

Document image decoding (DID) is an approach to document image recognition that is based on an explicit communication theory view of the processes of document creation, transmission and recognition [4, 5, 8]. DID views a document recognition problem as consisting of four elements— a message source, an imager, a noisy channel and a decoder. The most important issue in applying DID to a particular problem involves developing stochastic models for the source and imager. Much of the work in DID has focussed on Markov (stochastic finite-state) source models with a path-based approach to imaging that is motivated by the sidebearing model for letterform shape description and positioning [8]. Models of this type have been sucessfully used for decoding simple text columns, printed music [9], telephone yellow pages, dictionary entries and baseball box scores.

This paper introduces a new class of Markov image sources called *separable sources*. Loosely, a separable source is one that may be factored into a product of 1D models that represent horizontal and vertical structure, respectively. More formally, a separable model is a collection of named Markov subsources that is similar to a recursive transition network [3]. Associated with some of the nodes of the model are *position constraints* that restrict entry to the nodes to certain regions of the image plane. The top-level subsource is a vertical model whose nodes are all *tightly constrained* to specific horizontal positions. Separable models may be viewed as a generalization of the planar (or psuedo) HMMs described in [1, 10]. The importance of separable sources is that they provide the basis for heuristic decoding algorithms that offer significant computational savings over straightforward Viterbi decoding [6, 7].

*Presented at *IS&T/SPIE 1995 Intl. Symposium on Electronic Imaging*, San Jose, CA, Feb. 5–10, 1995.

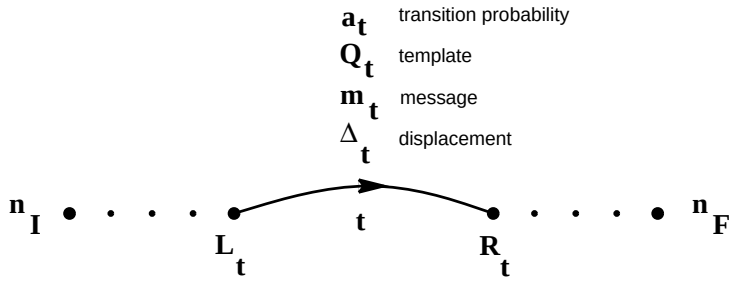

Figure 1: Simple Markov source model for image generation.

Separable sources are special forms of *recursive constrained* sources, which in turn are obtained by introducing node position constraints to the class of *simple* sources introduced in [8]. We describe a procedure for transforming certain constrained sources into separable form. An important component of the separation procedure is an algorithm for propagating user-specified position constraints, which are typically supplied for a small subset of the nodes, to the remaining nodes of the model.

The remainder of this paper is organized as follows. Section 3 defines the class of separable Markov sources and the related classes of *constrained* and *recursive* sources. Section 4 presents an algorithm for testing constrained sources for separability and performing the conversion when possible. An algorithm for constraint propagation is described.

3 SEPARABLE MARKOV SOURCES

This section develops the notion of a separable Markov source. We begin with a review of the image models introduced in [8], which we will call the *simple* Markov sources. Simple sources are then generalized to *constrained* sources by introducing node position constraints and to *recursive* sources by allowing one source to recursively "invoke" another. Separable models are then defined as a special class of recursive source.

3.1 Simple Markov Sources

A simple Markov source G, illustrated in Fig. 1, is a directed graph consisting of a finite set of states (nodes, vertices) $\mathcal{N} = \{n_1 \ldots n_N\}$ and a set of directed transitions (branches, edges) $\mathcal{B} = \{t_1 \ldots t_B\}$. Each transition t connects a pair of states, L_t and R_t, that are called, respectively, the *predecessor* (left) state and the *successor* (right) state of t. Two distinguished states are the initial state n_I and the final state n_F. With each transition is associated a 4-tuple of attributes, $(Q_t, m_t, a_t, \vec{\Delta}_t)$, where Q_t is the *template*, m_t is the *message string*, a_t is the *transition probability* and $\vec{\Delta}_t = (\Delta x_t, \Delta y_t)$ is the 2-dimensional integer vector *displacement*. The transitions of a simple Markov source are called *simple* transitions.

A *path* π in a Markov source is a sequence of transitions $t_1 \ldots t_P$ for which $R_{t_i} = L_{t_{i+1}}$ for $i = 1, \ldots, P-1$. A *complete path* is a path for which $L_{t_1} = n_I$ and $R_{t_P} = n_F$. Associated with each complete path π is a composite message, $M_\pi = m_{t_1} \cdots m_{t_P}$, formed by concatenating the message strings of the transitions of the path. A simple Markov source defines a probability distribution on complete paths by

$$\Pr\{\pi\} = \prod_{i=1}^{P} a_{t_i}. \tag{1}$$

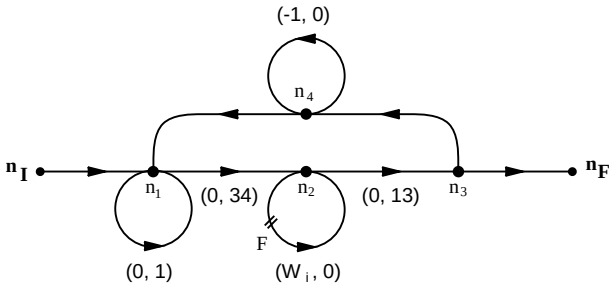

Figure 2: Simple text column source. Transition probabilities, templates and messages omitted for simplicity.

Also associated with each path π is a sequence of positions $\vec{\xi}_0 \ldots \vec{\xi}_P$ recursively defined by $\vec{\xi}_i = \vec{\xi}_{i-1} + \vec{\Delta}_{t_i}$, where $\vec{\xi}_0$ is an initial position, normally $\vec{0}$.[1] Informally, $\vec{\xi}_i$ can be viewed as the position of the cursor after the i^{th} transition in a simple "turtle graphics" imaging procedure [8]. A path defines a composite image Q by

$$Q_\pi = \bigcup_{i=1}^{P} Q_{t_i}[\vec{\xi}_{i-1}] \tag{2}$$

where $Q[\vec{x}]$ denotes Q shifted so that the origin of its local coordinate system is located at $\vec{x}$. For a path π we will define

$$\vec{\Delta}_\pi = \vec{\xi}_P - \vec{\xi}_0 = \sum_{i=1}^{P} \vec{\Delta}_{t_i} \tag{3}$$

to be the displacement of the path and will let Δx_π and Δy_π denote the x and y components of $\vec{\Delta}_\pi$, respectively. Note that P and t_i are functions of $\vec{\pi}$ and that $\vec{\xi}_i$ and $Q_{\vec{\pi}}$ depend on $\vec{\xi}_0$ as well. When we need to indicate the dependence explicitly we will write, for example, $\vec{\xi}_i(\vec{\pi}; \vec{\xi}_0)$.

Fig. 2 shows a simple source model for a text column, similar to that described in [8]. The operation of the text column source can be explained in terms of an imager automaton that moves over the image plane under control of the source model. In state n_1 the imager creates vertical whitespace; each traversal of the self-transition moves the imager down one row. At some point, the imager reaches the top of a text line and enters state n_2, which represents the creation of a horizontal text line. The displacement $(0, 34)$ of the transition into n_2 moves the imager down to the text baseline; 34 is the font height above baseline. The self-transitions at n_2 correspond to the individual characters of the font and horizontal whitespace. At the end of a text line the imager moves down by the font depth below baseline, 13, and enters n_3. At this point, the imager either exits to the final state or enters "carriage return" state n_4 to move back to the left margin in preparation for the next text line.

MAP decoding using a simple source involves finding a complete path $\hat{\vec{\pi}}$ (and thus $\hat{M}$) that maximizes the path likelihood function, defined by

$$\begin{aligned} \mathcal{L}(\vec{\pi}) &\equiv \mathcal{L}(Z \mid Q_{\vec{\pi}}) + \log \Pr\{\vec{\pi}\} \\ &= \sum_{i=1}^{P} \left[\mathcal{L}(Z \mid Q_{t_i}[\vec{\xi}_{i-1}]) + \log a_{t_i} \right] \end{aligned} \tag{4}$$

where $\mathcal{L}(Z \mid Q[\vec{x}])$ is the template match score for Q aligned at position $\vec{x}$ and depends on the channel model. MAP

[1] This paper uses an image coordinate system in which x increases to the right, y increases downward, and the upper left corner is at $x = y = 0$.

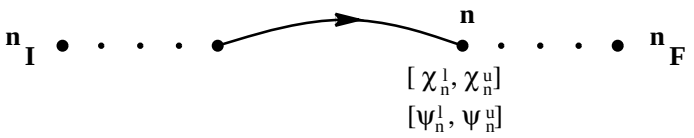

Figure 3: Constrained Markov source.

decoding can be implemented by computing the likelihood function

$$\mathcal{L}(n;\vec{x}) \quad \equiv \quad \max_{\vec{\pi} \ \text{s.t.} \ (n_I;\vec{\xi}_0) \mapsto (n;\vec{x})} \mathcal{L}(\vec{\pi}) \tag{5}$$

at each $(n,\vec{x}) \in \mathcal{N} \times \Omega$, where Ω is the integer lattice $[0,W] \times [0,H]$, where W and H are the image width and height, respectively. The notation $(n_I;\vec{\xi}_0) \mapsto (n;\vec{x})$ means that $\vec{\pi}$ is a path from node n_I at image position $\vec{\xi}_0$ to node n at $\vec{x}$. The likelihoods $\mathcal{L}(n;\vec{x})$ can be computed recursively by

$$\mathcal{L}(n;\vec{x}) \quad = \quad \max_{t|R_t=n} \Big\{ \mathcal{L}(L_t;\vec{x}-\vec{\Delta}_t) \ + \ \log a_t \ + \ \mathcal{L}(Z \mid Q_t[\vec{x}-\vec{\Delta}_t]) \Big\} \tag{6}$$

using a segmental Viterbi (dynamic programming) algorithm. The likelihood function $\mathcal{L}(n;\vec{x})$ depends implicitly on the start node n_I, the initial path position $\vec{\xi}_0$ and the source model G. When it is necessary to indicate them explicitly we will write $\mathcal{L}(n;\vec{x} \mid n_I;\vec{\xi}_0;G)$.

3.2 Constrained Markov Sources

Many aspects of document layout, such as the locations of column boundaries, page numbers and footnotes, can be naturally expressed in terms of constraints on the absolute positions of image features. Constrained Markov sources provide a framework for expressing and exploiting such constraints within the document decoding paradigm. A *constrained* Markov source is a simple source in which some or all of the nodes are labeled with x and/or y position constraints, as shown in Fig. 3. Informally, a position constraint specifies lower and upper bounds on the values of x or y when a path passes through a node. Formally, if $R_{t_i} = n$ for some transition t_i of a path in a constrained source, then $\vec{\xi}_i \in \mathcal{C}_n \equiv [\chi_n^l, \chi_n^u] \times [\psi_n^l, \psi_n^u]$. If $\chi_n^l = \chi_n^u = \chi_n$ we say that node n is *tightly constrained* in x; similarly, a node is tightly constrained in y if $\psi_n^l = \psi_n^u$. The start and final nodes of a source are typically tightly constrained to the upper left and lower right corners of the image plane, respectively, so that $\chi_{n_I} = \psi_{n_I} = 0$ and $\chi_{n_F} = W$ and $\psi_{n_I} = H$.

Table 1 shows a set of node position constraints for the text column source of Fig. 2. The columns labeled $\mathcal{C}_n$ contain the user-specified constraints; the remaining columns concern constraint propagation, discussed later. The x constraints on n_1 and n_3 force each text line (including whitespace before and after the text itself) to span the entire width of the column. For uniformity, all nodes are labeled with constraints; where no explicit constraint is supplied $[-\infty,+\infty]$ is assumed.

A constrained source defines a probability distribution on allowed complete paths by

$$\Pr\{\pi\} \ = \ \gamma \prod_{i=1}^{P} a_{t_i}, \tag{7}$$

where γ is a normalizing factor introduced so that the probabilities sum to unity. Since γ is a path-independent constant it does not enter into the decoding computation. For a constrained source, (5) is modified by restricting the maximization to

Node	$\mathcal{C}_n$	$\hat{\mathcal{C}}_n^f$	$\hat{\mathcal{C}}_n^r$	$\hat{\mathcal{C}}_n$
n_I	$[0,0]$	$[-\infty,+\infty]$	$[0,0]$	$[0,0]$
n_1	$[0,0]$	$[-\infty,W]$	$[-\infty,W]$	$[0,0]$
n_2	$[-\infty,+\infty]$	$[0,+\infty]$	$[-\infty,W]$	$[0,W]$
n_3	$[W,W]$	$[0,+\infty]$	$[0,+\infty]$	$[W,W]$
n_4	$[-\infty,+\infty]$	$[-\infty,W]$	$[0,+\infty]$	$[0,W]$
n_F	$[W,W]$	$[W,W]$	$[-\infty,+\infty]$	$[W,W]$

(a)

Node	$\mathcal{C}_n$	$\hat{\mathcal{C}}_n^f$	$\hat{\mathcal{C}}_n^r$	$\hat{\mathcal{C}}_n$
n_I	$[0,0]$	$[-\infty,+\infty]$	$[-\infty,H-47]$	$[0,0]$
n_1	$[-\infty,+\infty]$	$[0,+\infty]$	$[-\infty,H-47]$	$[0,H-47]$
n_2	$[-\infty,+\infty]$	$[34,+\infty]$	$[-\infty,H-13]$	$[34,H-13]$
n_3	$[-\infty,+\infty]$	$[47,+\infty]$	$[-\infty,H]$	$[47,H]$
n_4	$[-\infty,+\infty]$	$[47,+\infty]$	$[-\infty,H-47]$	$[47,H-47]$
n_F	$[H,H]$	$[47,+\infty]$	$[-\infty,+\infty]$	$[H,H]$

(b)

Table 1: Initial ($\mathcal{C}_n$) and propagated ($\hat{\mathcal{C}}_n$) constraints for text column source of Fig. 2. (a) Constraints on x. (b) Constraints on y.

paths that satisfy the constraints and (6) becomes

$$\mathcal{L}(n;\vec{x}) \;=\; \begin{cases} \max\limits_{t|R_t=n} \left\{ \mathcal{L}(L_t;\vec{x}-\vec{\Delta}_t) \;+\; \log a_t \;+\; \mathcal{L}(Z \mid Q_t[\vec{x}-\vec{\Delta}_t]) \right\} & \text{if } \vec{x} \in \mathcal{C}_n \\ -\infty & \text{otherwise,} \end{cases} \tag{8}$$

which represents a simple modification to the decoding algorithm. For convenience, we will often omit the constraint on $\vec{x}$ and write (8) simply as (6).

3.3 Recursive Markov Sources

The development of a large source model, such as the telephone yellow page model described in [8], is facilitated by specifying the model hierarchically. The formal basis for hierarchical description is the *recursive* Markov source, illustrated in Fig. 4. A recursive source is a collection of named subsources $G_0, G_1 \ldots G_K$, each of which is similar to a constrained Markov source except that it may include an additional type of transition. A *recursive* branch is labeled with a transition probability a_t and the name of one of the subsources, S_t. The interpretation of a recursive branch is that it represents a copy of the named subsource. One of the subsources is designated as the top-level subsource of the recursive source and is labeled G_0. The start and final nodes of a recursive source are defined to be those of G_0.

The subsources of a recursive source can be viewed as nodes in a directed dependency graph, with a branch from one subsource to another if the first subsource contains a recursive branch labeled with the name of the second. If the dependency graph of a recursive source is acyclic, the source is equivalent to a constrained Markov source derived by repeatedly replacing each recursive branch t in G_0 by a copy of S_t, until no recursive branches remain. Fig. 5 illustrates one step of the expansion. The expansion process will terminate if the dependency graph is acyclic; the resulting "flattened" source, denoted $\tilde{G}$, contains only simple transitions. If the dependency graph contains a cycle, the recursive source is not equivalent

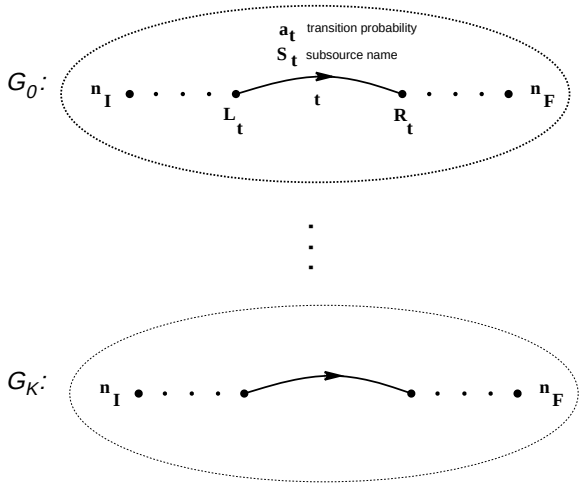

Figure 4: Recursive Markov source, showing a recursive branch in top-level subsource G_0. Subsources may also include simple branches and position constraints.

to a constrained source, but corresponds to the recursive transition network of a context-free grammar [3]. We assume in this paper that the dependency graphs are acyclic.

Fig. 6 is a simplified version of the top-level subsource of the yellow page column model.[2] The basic structure of a column is a sequence of top-level constituents separated by rows of vertical whitespace. Top-level constituents comprise ***standard***, ***bold***, and ***boxed*** individual listings, ***standard group*** and ***boxed group*** group listings, ***subject*** headings and ***junk*** lines. Fig. 7 shows a small portion of a column image that illustrates several types of constituents [11]. The image contains a reverse-video ***subject*** heading followed by one ***bold*** ("APEX COMMUNICATIONS INC") and four ***standard*** listings. The complete yellow page model is specified as a collection of 45 subsources. The corresponding flattened constrained source $\tilde{G}$ contains 1770 nodes and 6475 branches. User-supplied x constraints are associated with 128 of the nodes in $\tilde{G}$; 74 of the nodes are tightly constrained in x. Examples of these constraints are the requirements that the name field of a standard listing must begin within $.5in$ of the left edge of the column and that the number must end within $.25in$ of the right edge. The start and final nodes are tightly constrained in y; no other nodes have user-supplied y constraints.

Decoding an image with respect to a recursive source is defined to mean decoding it with respect to the equivalent constrained source $\tilde{G}$, so that, for example $\mathcal{L}(n;\vec{x} \mid G) \equiv \mathcal{L}(n;\vec{x} \mid G_0) \equiv \mathcal{L}(n;\vec{x} \mid \tilde{G})$ for $n \in G_0$. Since we are ultimately interested only in the final node likelihood $\mathcal{L}(n_F;W,H)$, it is sufficient to explicitly compute $\mathcal{L}(n;\vec{x})$ only for nodes in the top-level subsource G_0. By considering Fig. 5 and the definition of $\mathcal{L}(n;\vec{x} \mid \tilde{G})$ as a maximum path score, it is not difficult to see that (8) can be written in terms of transitions in G_0 as

$$\mathcal{L}(n;\vec{x} \mid G_0) = \max_{\substack{t \in G_0 \text{ s.t.} \\ R_t = n}} \max_{\vec{x}_1} \Big\{ \mathcal{L}(L_t;\vec{x}_1 \mid G_0) + \log a_t + \mathcal{L}(n_F(S_t);\vec{x} \mid n_I(S_t);\vec{x}_1;\tilde{S}_t) \Big\} \tag{9}$$

where for simple transitions we define

$$\mathcal{L}(n_F(S_t);\vec{x} \mid n_I(S_t);\vec{x}_1;\tilde{S}_t) \equiv \begin{cases} \mathcal{L}(Z \mid Q_t[\vec{x}_1]) & \text{if } \vec{x} - \vec{x}_1 = \vec{\Delta}_t \\ -\infty & \text{otherwise} \end{cases} \tag{10}$$

[2] The actual top-level subsource contains 10 nodes and 32 branches.

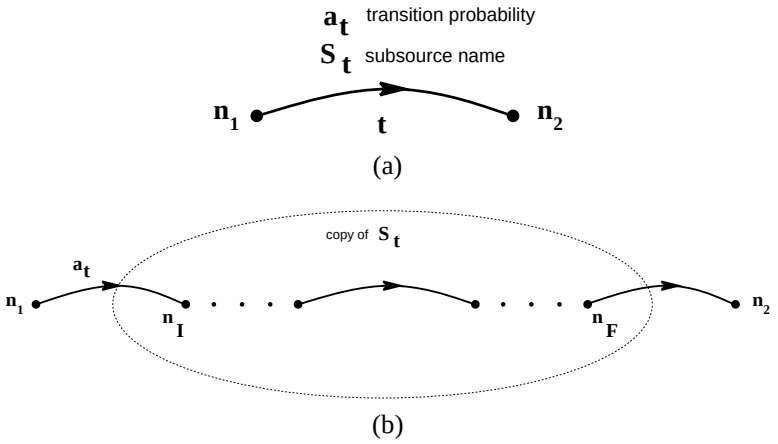

Figure 5: Expansion of a recursive transition. (a) Original transition. (b) Result of replacing t with a copy of subsource named S_t.

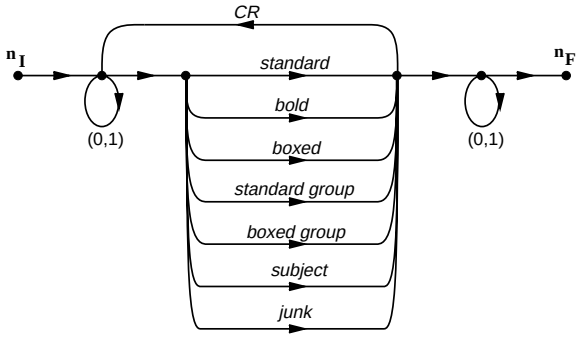

Figure 6: Top-level source model of yellow page column. Transition probabilities omitted for simplicity.

for uniformity of notation. The nested maximization over $\vec{x}_1$ is introduced because recursive branches can span image regions of variable size.

3.4 Separable Markov Sources

A separable source was previously characterized as one that can be factored into a product of 1D models that represent horizontal and vertical structure, respectively. We begin this section by developing the formal definition. The connection between formal separability and the informal notion might not be immediately apparent. Their relationship should be clear by the end of the section, however.

A subsource G_i is said to have *constant y displacement* if $\Delta y_{\vec{\pi}}$ is the same for every complete path $\vec{\pi}$ in $\tilde{G}_i$. If subsource S_t associated with a recursive transition t has constant y displacement $\Delta y_{\vec{\pi}}$ we will define the y displacement of the transition to be $\Delta y_{\vec{\pi}}$, by analogy with the displacement of a simple branch. A recursive source is said to be *separable* if each node of the top-level subsource G_0 is tightly constrained in x and if S_t has constant y displacement for every recursive branch t in

Figure 7: Small portion of a typical yellow page column.

G_0.

Fig. 8 shows a recursive text column source model that is equivalent to the constrained source defined by Fig. 2 and Table 1. This model was constructed automatically using the separation procedure described in section 4 below. Subsource G_1 models the generation of a single line of horizontal text, including the vertical displacements from the top of the line to the baseline and from the baseline to the line bottom. Subsource G_2 is the right-to-left carriage return. Each node of the top-level source G_0 is tightly constrained in x, as can be verified from Table 1(a). Furthermore, each complete path through G_1 has y displacement 47 and each path through G_2 has y displacement 0. Thus the text column source is separable. The flattened yellow page column model can be transformed, using the separation algorithm, into a separable source with 83 recursive transitions in the top-level subsource. The resulting model consists of 21 subsources, of which 19 are "printing" subsources, analogous to G_1 in Fig. 8, and two are "non-printing", similar to G_2 in Fig. 8. Examples of the printing subsources are models that generate ***standard name line*** (e.g. "Communications 2001" in Fig. 7), ***standard number line*** (e.g. "433 Airport Bl Burl . . ."), ***bold name line*** and ***standard oneliner*** (e.g. "Data-Tel SMto . . .").

If a source G is separable, the maximization over $\vec{x}_1$ in (9) reduces to the value at $\vec{x}_1 = (\chi_{L_t}, y - \Delta y_t)$ since L_t is tightly constrained in x and $\tilde{S}_t$ has constant y displacement. Moreover, since each $n \in G_0$ is tightly constrained in x, (9) further reduces to

$$\mathcal{L}(n; \chi_n, y \mid G_0) \;=\; \max_{\substack{t \in G_0 \text{ s.t.} \\ R_t = n}} \Big\{ \mathcal{L}(L_t; \chi_{L_t}, y - \Delta y_t \mid G_0) \;+\; \log a_t \;+\; \mathcal{L}(n_F(S_t); \chi_n, y \mid n_I(S_t); \chi_{L_t}, y - \Delta y_t; \tilde{S}_t) \Big\} \tag{11}$$

where the x and y coordinates are shown explicitly. If the fixed parameters in (11) are dropped as function arguments and we define

$$\mathcal{F}(t; y) \;\equiv\; \mathcal{L}(n_F(S_t); \chi_{R_t}, y \mid n_I(S_t); \chi_{L_t}, y - \Delta y_t; \tilde{S}_t) \tag{12}$$

then (11) simplifies further to

$$\mathcal{L}(n; y) \;=\; \max_{\substack{t \in G_0 \text{ s.t.} \\ R_t = n}} \{ \mathcal{L}(L_t; y - \Delta y_t) \;+\; \log a_t \;+\; \mathcal{F}(t; y) \} \,. \tag{13}$$

Recursion (13) may be interpretted as a one-dimensional segmental Viterbi recursion in y with segment scores given by $\mathcal{F}(t; y)$. If t is a simple transition, $\mathcal{F}(t; y)$ is the match score between the input image and template Q_t at image position $(\chi_{R_t}, y - \Delta y_t)$. If t is a recursive transition, $\mathcal{F}(t; y)$ is computed by finding the best path in $\tilde{S}_t$ from node $n_I(S_t)$ at image

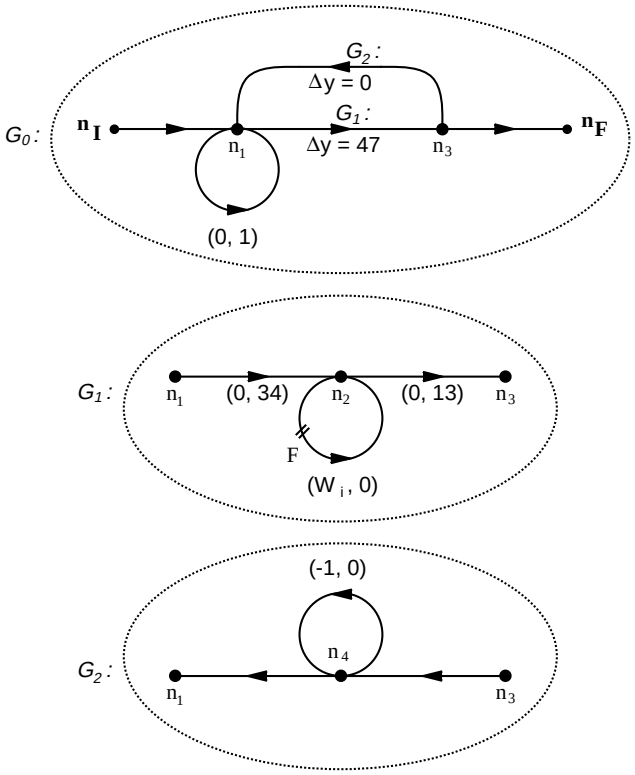

Figure 8: Separable text column source derived from source in Fig. 2.

position $(\chi_{n_I(S_t)}, y - \Delta y_t)$ to node $n_F(S_t)$ at $(\chi_{n_F(S_t)}, y)$. Since every complete path in $\tilde{S}_t$ has the same y displacement, it follows that the displacement of a path in $\tilde{S}_t$ from start node $n_I(S_t)$ to any other node n depends only on n. Thus, in computing $\mathcal{F}(t; y)$ for a given y, the nodes of $\tilde{S}_t$ can be viewed as tightly constrained in y and the computation reduces to a one-dimensional recursion in x. We will occasionally refer this horizontal recursion as "decoding row y" even though it is only the score of the final node $n_F(S_t)$ that is necessarily computed at y. In summary, a straightforward approach to decoding with a separable source is to first compute $\mathcal{F}(t; y)$ for every $t \in G_0$ and $y \in [0, H]$ and then use (13). Such a procedure is computationally equivalent to Viterbi decoding and will called the *separable* Viterbi algorithm.

The separable Viterbi algorithm, although involving the same number of arithmetic operations as the general decoding algorithm (6) can provide computational advantages in some situations. Thus, the importance of separable models extends beyond their role in heuristic decoding. One advantage is potential decoder parallelism. The cost of the separable Viterbi algorithm is dominated by the cost of the horizontal recursions that compute the values of $\mathcal{F}(t; y)$. Because these recursions can be performed independently, the separable algorithm is well-matched to parallel implementation on coarse-grained multiprocessors.

A second advantage of separable decoding concerns storage management. An important issue in the implementation of a Viterbi decoder is managing the back-pointers that are used to recover the best path. In a 1-dimensional decoder, these are typically stored in arrays, one for each node, whose lengths are the same as the length of the observation sequence. The analogous approach in two dimensions requires an array for each node that is as large as the entire image. This can be impractical for large image models. As a result, we typically use a dynamic storage allocation and recovery (garbage collection) scheme for storing partial paths during decoding. While the expense of this is not excessive, it can be significant. Moreover, the partial paths can occupy a considerable amount of storage.

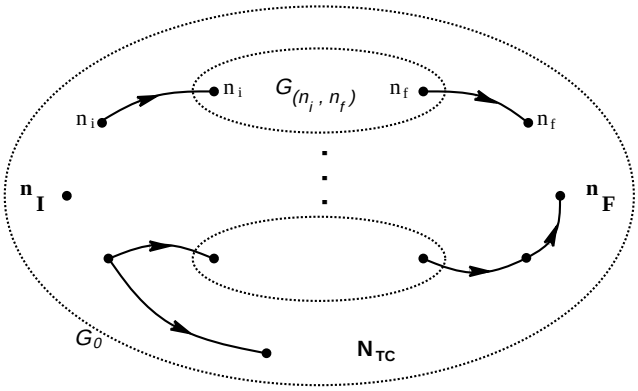

Figure 9: Conversion of constrained source to separable form.

Since the separable Viterbi algorithm consists of nested 1-dimensional recursions, it is possible to simple arrays for back pointer storage. The only complication is that the final output path consists of primitive transitions in the top-level vertical source and complete paths through some of the horizontal subsources. Thus, it is necessary to save the best paths found during all of the horizontal recursions. This can often be avoided by another optimization, however. Since the number of recursive transitions on the best path is typically much less than the total number of horizontal recursions, it can be advantageous to perform the horizontal decoding in two passes. During the first pass, all of the scores $\mathcal{F}(t; y)$ *but none of the best horizontal paths* are found. The horizontal scores are used to find the best path during the vertical recursion, as usual. After the best overall path is found, a full horizontal decoding (including best path determination) is performed for just those rows and subsources that lie on the best path. The cost of repeating some of the horizontal recursions is usually greatly outweighed by the savings that results from avoiding most of the horizontal path management overhead.

4 CONVERTING RECURSIVE SOURCES TO SEPARABLE FORM

In may situations, the natural structure of a class of images leads the user to create a model that is recursive, but not separable. For example, the ***standard*** listing is a natural candidate for a subsource in a recursive yellow page model. However, since such listings may span one or more lines a ***standard*** listing subsource will not have constant y displacement. This section describes a simple procedure for transforming recursive sources into separable form. We offer no proof that the procedure will always construct a separable form if one exists; indeed we suspect that there is no such guarantee. However, any separable form created by the algorithm is guaranteed to be equivalent to the original model.

The overall separation process occurs in three steps. The first step is to flatten the input recursive source G into the equivalent constrained source $\tilde{G}$, as described previously. The second step is to propagate the user-specified position constraints to the other nodes of the model. The third step constructs a separable model that is equivalent to $\tilde{G}$. A key factor determining the success of the third step is the number of nodes that are tightly constrained in x. Typically, position constraints are supplied by the user for a only small subset of the nodes. The objective of step two is to maximize the number of tightly constrained nodes. We first describe the construction of a separable model from a constrained source (step 3). Constraint propagation (step 2) is discussed in subsection 4.1 below.

Fig. 9 summarizes the structure of the separable source produced by the algorithm. Let $\tilde{G}$ be a constrained Markov source and let $\mathcal{N}_{TC}$ be the set of nodes that are tightly constrained in x. The nodes in $\mathcal{N}_{TC}$ become the nodes of the top-level subsource G_0. The start and final nodes of G_0 are taken to be those of $\tilde{G}$; these nodes are assumed to be in $\mathcal{N}_{TC}$. The transitions of $\tilde{G}$ that connect nodes in $\mathcal{N}_{TC}$ become simple transitions in G_0.

Source	$\tilde{G}$			G_0		$G_1 \ldots G_K$			$G_1 \ldots G_K$ (unique)		
	N	B	templates	N	B	K	$\sum N_i$	$\sum B_i$	K	$\sum N_i$	$\sum B_i$
text	6	80	70	4	5	2	6	77	2	6	77
YP	1770	6475	204	219	290	83	1121	8715	21	354	2997

Table 2: Statistics summarizing separation of text column and yellow page column models.

For each pair of tightly constrained nodes $(n_i, n_f) \in \mathcal{N}_{TC} \times \mathcal{N}_{TC}$, let $\tilde{G}(n_i; n_f)$ denote the subgraph of $\tilde{G}$ formed by removing all nodes of $\mathcal{N}_{TC}$ except n_i and n_f, all branches connected to the deleted nodes, all branches that enter n_i, all branches that exit n_f and all branches from n_i to n_f. If there is a path in $\tilde{G}(n_i; n_f)$ from n_i to n_f then G_0 includes a recursive transition from n_i to n_f. The subsource $G_{(n_i;n_f)}$ associated with that transition is the subgraph of $\tilde{G}(n_i; n_f)$ connected to both n_i and n_f. The start and final nodes of $G_{(n_i;n_f)}$ are copies of n_i and n_f. Informally, $G_{(n_i;n_f)}$ represents the paths in $\tilde{G}$ from n_i to n_f that do not enter a tightly constrained node before terminating at n_f. If each subsource $G_{(n_i;n_f)}$ has constant y displacement then G_0 plus the set of subsources forms a separable source. Otherwise, the result is a recursive source that is not separable.

The above construction generates a separable source in which only G_0 contains recursive transitions and where each subsource is invoked by a single recursive transition. The model can be simplified by partitioning the $G_{(n_i;n_f)}$ into sets of equivalent subsources and retaining only one member from each equivalence class.

The columns of Table 1 labeled $\hat{\mathcal{C}}_n$ show the propagated constraints for each node of the text column source of Fig. 2. As noted previously, Fig. 8 shows the result of applying the separation algorithm to the constrained text column source. Table 2 lists various measures of the complexities of the separable text column and yellow page column models. In this example, the text column model represents a 70 character subset of 12pt Adobe Times-Roman. As noted previously, the flattened yellow page source contains 74 nodes with tight user-supplied x constraints. Following constraint propagation, 219 nodes are tightly constrained in x. The top-level subsource G_0 of the separable yellow page model contains 83 recursive transitions. Each such transition defines a separate recursive subsource immediately after the third step of the separation procedure. By identifying equivalent subsources and removing the redundant ones, the number of subsources is reduced to 21, as indicated in the column labeled "unique". Note that the number of subsources in the final model is smaller than the number— 45— in the original recursive specification. Moreover, the final separable model contains about half the number of branches and one third as many nodes as $\tilde{G}$.

4.1 Propagation of Node Position Constraints

The primary objective of constraint propagation is to derive from the user-supplied constraints a set of implied constraints for the remaining nodes of the model. As will be seen, a secondary outcome is that some of the user-supplied constraints are tightened as a result of constraint propagation from other nodes. Thus, for uniformity, we may assume that every node has a user-specified constraint and the objective is simply to tighten them; if no constraint is explicitly provided $\mathcal{C}_n = [-\infty, +\infty]$ is assumed.

The position constraints for x and y are propagated separately. Thus, for simplicity, notation introduced previously for vector quantities will be used in this section to refer to scalar coordinates and constraints. For example, $\mathcal{C}_n$ will denote a constraint interval (rather than a rectangle), ξ will denote a scalar path position (rather than a vector), branch displacements will be scalars, etc.

Fig. 10 shows a simple example that illustrates the basic principles of constraint propagation. Suppose that nodes n_1, n, and n_2 have user-suplied constraints $\mathcal{C}_1$, $\mathcal{C}_n$ and $\mathcal{C}_2$, respectively, and consider the implications of these constraints for the

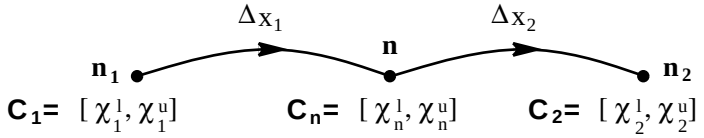

Figure 10: Simple constraint propagation example.

possible values of the path position ξ at node n. In the first place, the explicit constraint at n means that

$$\chi_n^l \quad \leq \quad \xi \quad \leq \quad \chi_n^u \tag{14}$$

must hold for every allowed path. However, because a path entering n must have just exited n_1, ξ will also satisfy the condition

$$\chi_1^l + \Delta x_1 \quad \leq \quad \xi \quad \leq \quad \chi_1^u + \Delta x_1 \tag{15}$$

as a result of the constraint at n_1. Similarly, because all paths leaving n immediately enter n_2,

$$\chi_2^l - \Delta x_2 \quad \leq \quad \xi \quad \leq \quad \chi_2^u - \Delta x_2 \tag{16}$$

will also hold. Thus, it is sufficient to allow only values of ξ in the interval $\hat{\mathcal{C}}_n = [\hat{\chi}_n^l, \hat{\chi}_n^u]$ defined by

$$\hat{\chi}_n^l \quad = \quad \max(\chi_1^l + \Delta x_1,\ \chi_2^l - \Delta x_2,\ \chi_n^l) \tag{17}$$

$$\hat{\chi}_n^u \quad = \quad \min(\chi_1^u + \Delta x_1,\ \chi_2^u - \Delta x_2,\ \chi_n^u), \tag{18}$$

or, in set notation,

$$\hat{\mathcal{C}}_n \quad = \quad \hat{\mathcal{C}}_n^f \ \cap\ \hat{\mathcal{C}}_n^r \ \cap \mathcal{C}_n \tag{19}$$

where

$$\hat{\mathcal{C}}_n^f \quad = \quad \mathcal{C}_1 \ \oplus\ \Delta x_1 \tag{20}$$

is called the *forward* propagated constraint,

$$\hat{\mathcal{C}}_n^r \quad = \quad \mathcal{C}_2 \ \ominus\ \Delta x_2 \tag{21}$$

is the *reverse* constraint, and $\oplus$ and $\ominus$ are operators that translate an interval by a specified scalar displacement. Constraint $\hat{\mathcal{C}}_n$ may be further tightened by noting that $\mathcal{C}_1$ and $\mathcal{C}_2$ in (20) and (21) may be replaced by $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$. Finally, for nodes with multiple incoming or outgoing branches (20) and (21) generalize to

$$\hat{\mathcal{C}}_n^f \quad = \quad \bigcup_{t|R_t=n} \hat{\mathcal{C}}_{L_t} \ \oplus\ \Delta x_t \tag{22}$$

and

$$\hat{\mathcal{C}}_n^r \quad = \quad \bigcup_{t|L_t=n} \hat{\mathcal{C}}_{R_t} \ \ominus\ \Delta x_t \tag{23}$$

respectively.[3]

Position constraint propagation involves solving the set of equations (19), (22) and (23) for each $\hat{\mathcal{C}}_n$, subject to the boundary condition that the solutions for the start and final nodes include specified intervals. Typical boundary conditions are $\hat{\mathcal{C}}_{n_I} = [0, 0]$ and $\hat{\mathcal{C}}_{n_F} = [W, W]$, where W is the image width.

[3] Since we are concerned with intervals, union means "range" union rather than set union. Thus, for example, $[-1, 0] \cup [1, 2] = [-1, 2]$.

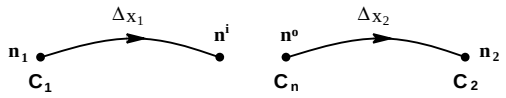

Figure 11: Result of splitting node n of Fig. 10 for forward constraint propagation.

Solving the constraint equations is complicated because the forward and reverse constraints are coupled by (19) and cycles in the source lead to recursive dependencies. One approach to reducing the difficulty is to relax the requirement that the computed constraints be as tight as possible. As long as the user-specified constraints are satisfied, the only consequence will be that the decoding trellis will be larger than necessary and/or not enough tightly constrained nodes will be identified to separate the source. Thus, we look for modifications to the constraint equations whose effect is to enlarge the solution intervals. Specifically, we will make subsitutions for $\hat{\mathcal{C}}_{L_t}$ and $\hat{\mathcal{C}}_{R_t}$ in the right sides of (22) and (23).

From (19) we have the following relations

$$\hat{\mathcal{C}}_n \subseteq \hat{\mathcal{C}}_n^f \tag{24}$$
$$\hat{\mathcal{C}}_n \subseteq \mathcal{C}_n \tag{25}$$
$$\hat{\mathcal{C}}_n \subseteq \hat{\mathcal{C}}_n^r. \tag{26}$$

We modify (22) by replacing $\hat{\mathcal{C}}_{L_t}$ with $\hat{\mathcal{C}}_{L_t}^{f\prime}$ defined by

$$\hat{\mathcal{C}}_n^{f\prime} \equiv \begin{cases} \mathcal{C}_n & \text{if } \mathcal{C}_n \text{ is a finite interval} \\ \hat{\mathcal{C}}_n^f & \text{otherwise} \end{cases} \tag{27}$$

to give

$$\hat{\mathcal{C}}_n^f = \bigcup_{t|R_t=n} \hat{\mathcal{C}}_{L_t}^{f\prime} \oplus \Delta x_t. \tag{28}$$

Similarly, (23) becomes

$$\hat{\mathcal{C}}_n^r = \bigcup_{t|L_t=n} \hat{\mathcal{C}}_{R_t}^{r\prime} \ominus \Delta x_t \tag{29}$$

where $\hat{\mathcal{C}}_n^{r\prime}$ is defined analogously. The primary effect of these substitutions is that the forward and reverse constraints become decoupled and can be propagated independently. We will discuss the propagation of forward constraints. Propagation of reverse constraints is similar and reduces to propagating forward constraints in the *transpose* of G, constructed by reversing the direction of each transition and replacing each branch displacement with its negative.

If $\mathcal{C}_n$ is finite (i.e. the user actually specified a constraint for node n) then $\hat{\mathcal{C}}_n^f$ does not occur on the right hand side of (28). As a result forward constraint propagation in G is equivalent to propagation in the modified source G' derived by spliting each finitely-constrained node n into two nodes n^i and n^o, where n^i inherits the incoming branches of n and n^o inherts the outgoing branches as well as constraint $\mathcal{C}_n$. As an example, the splitting of node n in Fig. 10 is illustrated in Fig. 11.

The advantage of splitting the constrained nodes is that forward constraint propagation in G' is equivalent to finding paths of minimum and maximum displacement in graphs that are simple derivatives of G'. The upper and lower limits of $\hat{\mathcal{C}}_n^f = [\hat{\chi}_n^{fl}, \hat{\chi}_n^{fu}]$ are found separately. The lower limits $\hat{\chi}_n^{fl}$ are computed using the graph in Fig. 12 in which the user-specified lower limits χ_n^l become displacements on the branches leaving n_S. The lower limits are propagated by finding the minimum path displacement from n_S to each node of G'. If node n was split, $\hat{\chi}_n^{fl}$ is the minimum displacement to n_i; otherwise $\hat{\chi}_n^{fl}$ is the minimum displacement to n. The upper limits $\hat{\chi}_n^{fu}$ are found similarly, by finding maximum path displacements when the displacements of the branches leaving n_S are the upper limits χ_n^u.

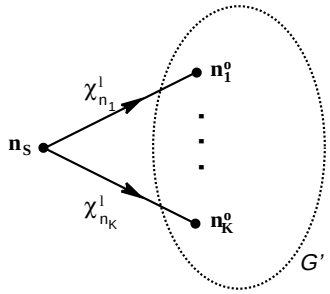

Figure 12: Graph for propagating lower limits of forward constraints. Constraints are propagated by finding paths of minimum displacement from n_S.

Finding paths of minimum or maximum displacement from a single graph node is a standard graph problem [2]. A minor issue arises in the present situation because G' may contain cycles of non-zero total displacement, with the result that one or both of the propagated constraint limits may be infinite for some nodes. Path-finding algorithhms that detect such cycles are also straightforward [6, 7].

References

[1] O. Agazzi, S. Kuo, E. Levin and R. Pieraccini, "Connected and degraded text recognition using planar hidden Markov models", *1993 IEEE International Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, MN, Apr. 27–30, 1993.

[2] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, 1974, Reading: Addison-Wesley.

[3] A. Barr and E. Feigenbaum (eds.), *The Handbook of Artificial Intelligence*, 1981, Los Altos, CA: William Kaufmann, Inc.

[4] P. Chou, "Recognition of equations using a two-dimensional stochastic context-free grammar", *SPIE Conf. on Visual Communications and Image Processing*, Philadelphia, PA, Nov. 1989.

[5] P. Chou and G. Kopec, "A stochastic attribute grammar model of document production and its use in document image decoding", these proceedings.

[6] A. Kam, *Heuristic document image decoding using separable Markov models*", M.I.T. Master of Science thesis, June, 1993.

[7] A. Kam and G. Kopec, "Heuristic image decoding using separable source models", *Proc. 1994 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, vol. V, pp. 145–148, Adelaide, South Australia, Apr. 19–22, 1994.

[8] G. Kopec and P. Chou, "Document image decoding using Markov source models", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, June, 1994, pp. 602–617.

[9] G. Kopec, P. Chou and D. Maltz, "Markov source model for printed music decoding", these proceedings.

[10] S. Kuo and O. Agazzi, "Keyword spotting in poorly printed documents using pseudo 2-d hidden Markov models", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 8, August, 1994, pp. 842–848.

[11] Pacific Bell, *Smart Yellow Pages, Palo Alto, Redwood City and Menlo Park*, 1992.