Supervised Template Estimation for Document Image Decoding

Gary E. Kopec, *Member, IEEE,* and Mauricio Lomelin, *Member, IEEE*

July 20, 1997

Abstract

An approach to supervised training of character templates from page images and unaligned transcriptions is proposed. The template training problem is formulated as one of constrained maximum likelihood parameter estimation within the document image decoding framework. This leads to a three-phase iterative training algorithm consisting of transcription alignment, aligned template estimation (ATE) and channel estimation steps. The maximum likelihood ATE problem is shown to be NP-complete and thus an approximate solution approach is developed. An evaluation of the training procedure in a document-specific decoding task using the Univ. of Washington UW-II database of scanned technical journal articles is described.

Index Terms: document image decoding, Markov models, template estimation, character recognition, document recognition, maximum likelihood

[0] Gary E. Kopec is with the Xerox Palo Alto Research Center, Palo Alto CA 94304 USA; email: kopec@parc.xerox.com

[0] Mauricio Lomelin is with the Microsoft Corporation, Seattle WA; email: mauricil@microsoft.com

1 Introduction

Document image decoding (DID) is an approach to document recognition that is based on a communication theory view of the processes of document creation, transmission and recognition [11]. A major issue in applying DID to a particular recognition problem is developing explicit models for the character shapes found in the documents to be recognized [10]. Unlike most approaches to character recognition, DID emphasizes the use of document-specific templates as character models, rather than "omni-font" features. The focus on document-specific models is motivated by the observation that the error rate of a font-specific recognizer can be significantly less than that of a multi-font system [2]. A major advantage of using templates is that there is no need to segment the printed text into isolated glyph images prior to character classification.

The traditional approach to creating character templates is to average and threshold a set of aligned sample images for each character. The sample images are typically obtained by interactively segmenting and labeling a set of training pages. Manual segmentation and labeling is simple, reliable and applicable to a wide variety of document types. However, it can be prohibitively expensive. Moreover, the notion of creating a collection of isolated training glyphs is incongruous with segmentation-free decoding.

The success of automatic speech recognition systems based on hidden Markov models (HMMs) has motivated recent attempts to develop HMM-based methods for printed text recognition [15, 4]. One of the main advantages of the HMM approach is that character model parameters can be estimated using whole word or text line images with unaligned transcriptions. This eliminates the need to segment the training data into individual glyph images prior to training. However, existing HMM training algorithms are applicable to only a limited range of character shape and positioning models. HMM text recognizers typically are based on a simple 1-dimensional typesetting model in which a line of text can be partitioned into individual glyphs by vertical cuts. Thus, although *segmented* input is not required it is assumed that the image is *segmentable* into disjoint rectangular regions representing individual glyphs.

Complex graphical notations such as music and equations are not well described in terms of non-overlapping rectangular components. Moreover, even for simple text, bounding box overlap

can occur in the case of italics or pairwise kerning. Thus, current HMM training methods are not suitable for the full range of typesetting models found in contemporary documents.

This paper presents an approach to supervised maximum likelihood (ML) template estimation, similar in spirit to current HMM methods, that is applicable to the widely used sidebearing model of character shape and positioning [19]. The inputs to the training procedure are images of whole pages or text lines plus the corresponding unaligned transcriptions. The method does not assume that the training data can be segmented into individual glyphs by rectangular subdivision. Rather, glyph segmentation and template estimation are performed contemporaneously on a pixel-by-pixel basis.

The proposed algorithm is a three-phase iterative procedure. The first phase is *transcription alignment*, in which glyph origins are located and labeled using a current set of templates. The heart of the algorithm is the second phase, *aligned template estimation* (ATE), during which templates are estimated using the current labeled glyph positions. The third phase is *channel estimation* in which parameters of the assumed image degradation model are updated. The procedure is applicable to the class of multilevel character shape models defined in [14] and the class of Markov image sources defined in [11].

Development of the training algorithm was motivated by the paradigm of *document-specific decoding* [12]. The basic scenario in document-specific decoding is to train a set of templates using a subset of the document pages plus their transcriptions and then use the templates to decode the remaining pages. Document-specific decoding is intended to address the needs of high-volume, high-accuracy conversion efforts [20] and has been successfully applied to document capture for digital library applications [13].

The rest of this paper is organized as follows. Section 2 briefly reviews the theory of Markov image sources and multilevel channels [11, 14] and then formulates the supervised ML template estimation problem for such sources. Sections 3, 4 and 5 discuss aligned template estimation, channel estimation and transcription alignment, respectively. Section 6 illustrates use of the proposed algorithm for document-specific decoding. Finally, section 7 contains a brief summary.

2 Problem Formulation

This section briefly reviews the theory of multilevel Markov image sources and channels developed in [11, 14] and then formulates the problem of supervised maximum likelihood template and channel parameter estimation.

2.1 Multilevel Image Sources

The role of an image source model in DID is to define how a composite ideal image Q is constructed from primitive "templates" to encode a message string M. A multilevel Markov image source is a directed graph consisting of a finite set of states (nodes, vertices) $\mathcal{N}$ and a set of directed transitions (branches, edges) $\mathcal{B}$. Two distinguished members of $\mathcal{N}$ are the initial state n_I and the final state n_F. Each transition t is labeled with a 4-tuple of attributes, $(Q_t, m_t, a_t, \vec{\Delta}_t)$, where Q_t is a *template*, m_t is a *message* string, a_t is the *transition probability* and $\vec{\Delta}_t$ is the vector *displacement* of t.

A multilevel template Q_t is an array of pixels that defines a primitive image element. The pixel value (color) of Q_t at position $\vec{x}$ relative to its local origin is denoted $q_t(\vec{x})$, where $q_t(\vec{x}) \in \{0, \ldots, L-1\}$. Level 0 denotes the background color and levels $1, \ldots, L-1$ are foreground colors. In principle, each template extends over the entire image plane. However, the *support* of Q_t (the set of locations where $q_t(\vec{x})$ differs from the background color) is typically localized to a small region in the vicinity of the origin. In particular, we assume that the support of each template lies within a rectangle $\mathcal{C}$ of height H and width W. We call this rectangular region the *canvas* of the template, illustrated in fig. 1. The canvas parameters H, W, χ and ψ may depend on the template, although we will suppress this dependence. A complete set of pixel values $\{q_t(\vec{x}) \mid t \in \mathcal{B}, \vec{x} \in \mathcal{C}\}$ defines a template parameter vector Θ whose elements are indexed by $(t, \vec{x})$.

A *complete path* Π through a Markov source is a sequence of transitions $t_0, \ldots, t_{P-1}$ that connects n_I to n_F.[1] In general, a given transition will occur multiple times in a complete path; we will let N_t be the number of occurances of transition t. A complete path defines a composite message, or *transcription*, M_Π, formed by concatenating the message strings of the transitions of the path. Also associated with each path is a sequence of positions in the image plane, $\vec{x}_0, \ldots, \vec{x}_P$,

[1] All complete paths through a source do not necessarily have the same length. However, for notational simplicity the dependence of P on Π will not be indicated.

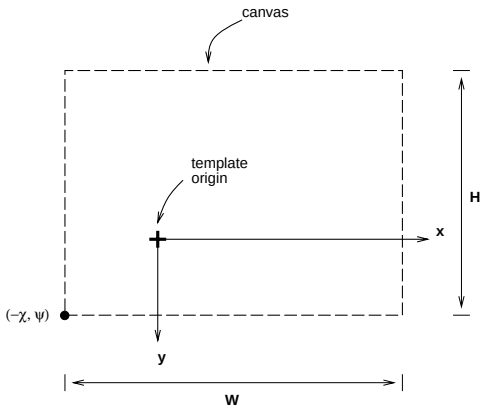


Figure 1: Canvas parameters and template coordinate system. The lower left corner of the canvas is at $(-\chi,\ \psi)$ in the template coordinate system.

recursively defined by $\vec{x}_{i+1} = \vec{x}_i + \vec{\Delta}_{t_i}$, where $\vec{x}_0$ is an initial position, normally $\vec{0}$.[2] The pairs $\{(\vec{x}_i, t_i), i = 0, \ldots, P-1\}$ are called the *labeled glyph positions* defined by Π and we will let $\vec{x}_1^{(t)}, \ldots, \vec{x}_{N_t}^{(t)}$ denote the coordinates of the glyph positions labeled with a particular transition t. By virtue of the labeled glyph positions a path defines an *alignment* between the transcription and the image.

A complete path or a set of labeled glyph positions defines a composite ideal image Q_Π by

$$Q_\Pi = \biguplus_{i=0}^{P-1} Q_{t_i}[\vec{x}_i] \tag{1}$$

where $Q_{t_i}[\vec{x}_i]$ denotes Q_{t_i} shifted so that the origin of its local coordinate system is located at $\vec{x}_i$ and $\biguplus$ represents a template composition rule that defines how the individual templates are merged to form Q_Π. Image sources are usually designed to satisfy a *template disjointness constraint* that asserts that the shifted templates $Q_{t_i}[\vec{x}_i]$ on the right hand side of (1) have disjoint supports [11]. The disjointness constraint is motivated by the observation that typefaces are normally designed so that adjacent characters in a line of printed text do not overlap. If the templates are disjoint, a natural choice for $\biguplus$ is pixelwise addition.

The template disjointness constraint can be formalized as follows. Given a set of labeled glyph positions, each template pixel $(t, \vec{x})$ defines a set of image locations $\mathcal{I}(t, \vec{x}) = \left\{\vec{x} + \vec{x}_i^{(t)}; i = 1, \ldots, N_t\right\}$

[2] We use an image coordinate system in which x increases to the right, y increases downward, and the upper left corner of the page is at $x = y = 0$.

that can be loosely described as the pixels of Q_Π or Z that are "covered" by $(t, \vec{x})$. Two template pixels $(t_1, \vec{x}_1)$ and $(t_2, \vec{x}_2)$ are said to *conflict* if $\mathcal{I}(t_1, \vec{x}_1) \cap \mathcal{I}(t_2, \vec{x}_2) \neq \emptyset$. The set of template pixels that conflict with $(t, \vec{x})$ will be denoted $\mathcal{K}(t, \vec{x})$. The template disjointness constraint prohibits pairs of distinct foreground template pixels from conflicting.

2.2 Multilevel Channel

The role of a channel model is to provide an expression for the log likelihood ratio

$$\mathcal{L}(Z \mid Q) \equiv \log \frac{\Pr\{Z \mid Q\}}{\Pr\{Z \mid Q_0\}} \tag{2}$$

where Q and Z are ideal and observed images, respectively and Q_0 is the all-background image [11]. An L-input symbol, 2-output symbol multilevel channel is characterized by a vector of level parameters $\vec{\alpha} = (\alpha_0, \ldots, \alpha_{L-1})$, where $\alpha_l = \Pr\{z(\vec{x}) = 1 \mid q(\vec{x}) = l\}$ for $l = 1, \ldots, L-1$ and $\alpha_0 = \Pr\{z(\vec{x}) = 0 \mid q(\vec{x}) = 0\}$ [14].[3]

The log likelihood ratio for the special case of a bilevel channel ($L = 2$) is given by

$$\mathcal{L}(Z \mid Q) = \gamma \|Q \wedge Z\| + \beta \|Q\| \tag{3}$$

where $\|X\|$ denotes the number of 1's in X, $\wedge$ is the bitwise **and** operator and

$$\gamma = \log \frac{\alpha_0 \alpha_1}{(1-\alpha_0)(1-\alpha_1)} \tag{4}$$

$$\beta = \log \frac{1-\alpha_1}{\alpha_0}. \tag{5}$$

The log likelihood ratio for the general case is

$$\mathcal{L}(Z \mid Q) = \sum_{l=1}^{L-1} \left[\gamma_l \|\tilde{Q}^{(l)} \wedge Z\| + \beta_l \|\tilde{Q}^{(l)}\|\right] \tag{6}$$

$$= \sum_{l=1}^{L-1} \mathcal{L}(Z \mid \tilde{Q}^{(l)}) \tag{7}$$

[3] The exceptional definition of α_0 maintains consistency with the notation for bilevel channels defined in [11].

where $\tilde{Q}^{(l)}$ is the bilevel bitmap representing the l^{th} level set of Q, i.e.,

$$\tilde{q}^{(l)}(\vec{x}) \equiv \begin{cases} 1 & \text{if } q(\vec{x}) = l \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and γ_l and β_l are given by (4) and (5) for each level.

A foreground pixel for which $\alpha_l > 1 - \alpha_0$ is called "write-black" because such a pixel is more likely than a background pixel to be observed black. For a write-black level, $\gamma_l > 0$ and $\beta_l < 0$. Conversely, $\alpha_l < 1 - \alpha_0$ for a "write-white" level and $\gamma_l < 0$ and $\beta_l > 0$.

The multilevel log likelihood ratio satisfies a basic superposition property with important algorithmic and theoretical consequences. If Q_1 and Q_2 are multilevel templates with disjoint support then

$$\mathcal{L}(Z \mid Q_1 \uplus Q_2) = \mathcal{L}(Z \mid Q_1) + \mathcal{L}(Z \mid Q_2) \tag{9}$$

where $\uplus$ denotes elementwise addition. In particular, for Q_Π defined by (1) we have

$$\mathcal{L}(Z \mid Q_\Pi) = \sum_{i=0}^{P-1} \mathcal{L}(Z \mid Q_{t_i}[\vec{x}_i]) \tag{10}$$

if the template disjointness constraint is satisfied. Decomposition (10) is the basis for linear-time decoding using dynamic programming.

2.3 Template and Channel Estimation

Supervised maximum likelihood (ML) template and channel estimation is the problem of finding $(\hat{\Theta}, \hat{\vec{\alpha}})$ such that

$$(\hat{\Theta}, \hat{\vec{\alpha}}) = \arg\max_{(\Theta, \vec{\alpha})} \Pr\{Z \mid \Theta, \vec{\alpha}, M\} \tag{11}$$

for a given image Z and transcription M, where $\hat{\Theta}$ satisfies the template disjointness constraint. By expanding in terms of paths that are consistent with the transcription, (11) can be written

$$(\hat{\Theta}, \hat{\vec{\alpha}}) = \arg\max_{(\Theta,\vec{\alpha})} \sum_{\Pi|M_\Pi=M} \Pr\{Z, \Pi \mid \Theta, \vec{\alpha}, M\} \tag{12}$$

$$= \arg\max_{(\Theta,\vec{\alpha})} \sum_{\Pi|M_\Pi=M} \Pr\{Z \mid \Pi, \Theta, \vec{\alpha}, M\} \Pr\{\Pi \mid \Theta, \vec{\alpha}, M\}. \tag{13}$$

Since $(\Theta, \vec{\alpha})$ consists only of pixel color values and channel parameters, it clear that $\Pr\{\Pi \mid \Theta, \vec{\alpha}, M\} = \Pr\{\Pi \mid M\}$. Furthermore, since M is fixed it is easily shown that $\Pr\{\Pi \mid M\}$ can be replaced by $\Pr\{\Pi\}$. Moreover, $\Pr\{Z \mid \Pi, \Theta, \vec{\alpha}, M\} = \Pr\{Z \mid Q_{\Pi,\Theta}, \vec{\alpha}\}$, where the subscripts on $Q_{\Pi,\Theta}$ reflect dependence on both the path and the template parameters, and thus (13) becomes

$$(\hat{\Theta}, \hat{\vec{\alpha}}) = \arg\max_{(\Theta,\vec{\alpha})} \sum_{\Pi|M_\Pi=M} \Pr\{Z \mid Q_{\Pi,\Theta}, \vec{\alpha}\} \Pr\{\Pi\}. \tag{14}$$

As discussed in [11], it is usual to approximate the sum in (14) by its largest term, in which case

$$(\hat{\Theta}, \hat{\vec{\alpha}}) = \arg\max_{(\Theta,\vec{\alpha})} \max_{\Pi|M_\Pi=M} \Pr\{Z \mid Q_{\Pi,\Theta}, \vec{\alpha}\} \Pr\{\Pi\}. \tag{15}$$

A straightforward approach to the joint maximization over Θ, $\vec{\alpha}$ and Π implied by (15) is to iteratively maximize over Π, Θ and $\vec{\alpha}$ in separate parameter update steps,

$$\textbf{step 1.} \quad \Pi' \leftarrow \arg\max_{\Pi|M_\Pi=M} \Pr\{Z \mid Q_{\Pi,\Theta}, \vec{\alpha}\} \Pr\{\Pi\} \tag{16}$$

$$\textbf{step 2.} \quad \Theta' \leftarrow \arg\max_{\Theta} \Pr\{Z \mid Q_{\Pi',\Theta}, \vec{\alpha}\} \tag{17}$$

$$\textbf{step 3.} \quad \vec{\alpha}' \leftarrow \arg\max_{\vec{\alpha}} \Pr\{Z \mid Q_{\Pi',\Theta'}, \vec{\alpha}\} \tag{18}$$

starting with some initial parameters or glyph positions and continuing until a stopping criterion is reached. Note that (16) and (17) remain valid if their right hand sides are divided by $\Pr\{Z \mid Q_0\}$, where Q_0 is the all-background image, and the logarithm is taken. If this is done we obtain

$$\textbf{step 1.} \quad \Pi' \leftarrow \arg\max_{\Pi|M_\Pi=M} [\mathcal{L}(Z \mid Q_{\Pi,\Theta}) + \log \Pr\{\Pi\}] \tag{19}$$

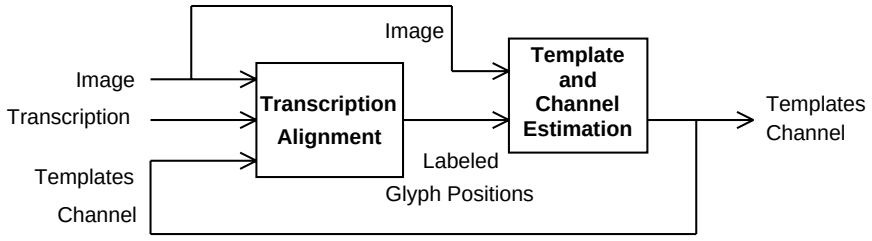


Figure 2: Iterative transcription alignment and aligned template and channel estimation.

$$\textbf{step 2.} \quad \Theta' \leftarrow \arg\max_{\Theta} \mathcal{L}(Z \mid Q_{\Pi',\Theta}) \tag{20}$$

$$\textbf{step 3.} \quad \vec{\alpha}' \leftarrow \arg\max_{\vec{\alpha}} \Pr\{Z \mid Q_{\Pi',\Theta'}, \vec{\alpha}\} \tag{21}$$

where the dependence on $\vec{\alpha}$ is implicit in (19) and (20). Step 1 of the iteration finds an alignment between the transcription and the image, using the current set of templates. Step 2 finds a set of templates given the current alignment and is called *aligned template estimation* (ATE). Finally, step 3 updates the channel parameters given a set of aligned templates and is called *channel estimation* (CE).

Fig. 2 summarizes the overall structure of a supervised template estimation algorithm based on (19), (20) and (21). For simplicity, the ATE and CE steps are shown merged into a single estimation step.

3 Aligned Template Estimation (ATE)

Traditional character template estimation is concerned with deriving a character template from a set of *isolated* images of the character. The character images are typically extracted from a page image during a segmentation operation that preceeds template estimation. By contrast, the inputs to ATE are a page image and a path in the page image (equivalently, the set of labeled glyph positions defined by a path). Although ATE is not based on character image segmentation, it is neveretheless instructive to consider the analog of the isolated character image in ATE.

If $(\vec{x}_i, t_i)$ is a labeled glyph position, the corresponding *glyph image region* is defined to be that portion of Z within the canvas of Q_{t_i}, when the canvas origin is positioned at $\vec{x}_i$. Fig. 3

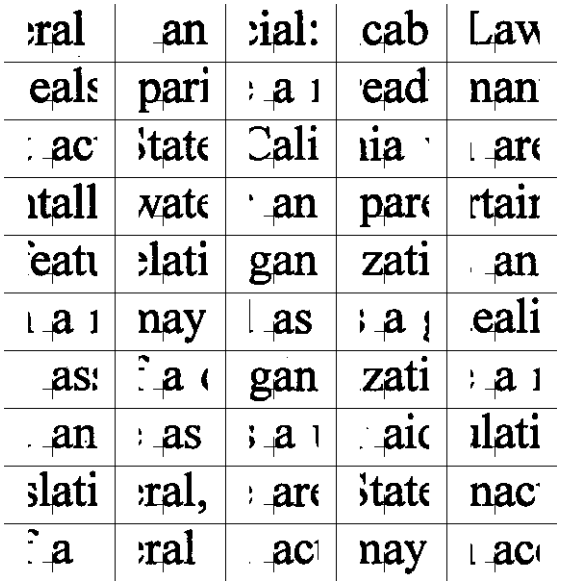


Figure 3: Glyph image regions for Times Roman “a”. The canvas origin is indicated by a cross.

shows a collection of glyph image regions for the Times-Roman character “a” drawn from a set of scanned pages [3]. As the figure illustrates, a glyph image region typically contains glyphs and/or parts of glyphs in addition to the glyph located at the canvas origin. Similarly, it is clear that there can be significant overlap between the glyph image regions at adjacent glyph positions. As part of the estimation process, an ATE algorithm will effectively decide which of the pixels in each glyph image region belong to the labeled glyph, and which belong to adjacent glyphs.

Multilevel template estimation is the problem of choosing Θ to maximize $\mathcal{L}(Z \mid Q_{\Pi,\Theta})$ while satisfying the template disjointness constraint. By expanding the right hand side of (6) in terms of individual pixels, $\mathcal{L}(Z \mid Q_{t_i,\Theta}[\vec{x}_i])$ in (10) can be written

$$\mathcal{L}(Z \mid Q_{t_i,\Theta}[\vec{x}_i]) \;=\; \sum_{\vec{x}\in\mathcal{C}} \sum_{l=1}^{L-1} \left[\gamma_l \tilde{q}_{t_i}^{(l)}(\vec{x}) z(\vec{x}+\vec{x}_i) \;+\; \beta_l \tilde{q}_{t_i}^{(l)}(\vec{x})\right]. \tag{22}$$

Substituting this into (10) gives

$$\mathcal{L}(Z \mid Q_{\Pi,\Theta}) \;=\; \sum_{i=0}^{P-1} \sum_{\vec{x}\in\mathcal{C}} \sum_{l=1}^{L-1} \left[\gamma_l \tilde{q}_{t_i}^{(l)}(\vec{x}) z(\vec{x}+\vec{x}_i) \;+\; \beta_l \tilde{q}_{t_i}^{(l)}(\vec{x})\right]. \tag{23}$$

If the outer summation over path transitions is replaced by a summation over source model transitions, (23) becomes

$$\mathcal{L}(Z \mid Q_{\Pi,\Theta}) = \sum_{t\in\mathcal{B}}\sum_{i=1}^{N_t}\sum_{\vec{x}\in\mathcal{C}}\sum_{l=1}^{L-1}\left[\gamma_l \tilde{q}_t^{(l)}(\vec{x}) z(\vec{x}+\vec{x}_i^{(t)}) \; + \; \beta_l \tilde{q}_t^{(l)}(\vec{x})\right] \tag{24}$$

$$= \sum_{t\in\mathcal{B}}\sum_{\vec{x}\in\mathcal{C}}\sum_{l=1}^{L-1}\tilde{q}_t^{(l)}(\vec{x})\left[\gamma_l \sum_{i=1}^{N_t} z(\vec{x}+\vec{x}_i^{(t)}) \; + \; \beta_l N_t\right]. \tag{25}$$

In the $L = 2$ case of bilevel templates, (25) reduces to

$$\mathcal{L}(Z \mid Q_{\Pi,\Theta}) = \sum_{t\in\mathcal{B}}\sum_{\vec{x}\in\mathcal{C}} q_t(\vec{x})\left[\gamma \sum_{i=1}^{N_t} z(\vec{x}+\vec{x}_i^{(t)}) \; + \; \beta N_t\right]. \tag{26}$$

Comparison of (25) and (26) suggests that multilevel estimation is equivalent to bilevel estimation of the level set bitmaps, with two minor differences. First, the level set bitmaps must satisfy the constraint that only one value of $\tilde{q}_t^{(l)}(\vec{x})$ may be non-zero for a given t and $\vec{x}$. However, this will happen automatically as a consequence of the template disjointness constraint. Second, in (26) the same values of γ and β are used for all templates, whereas in (25) they depend on the level. However, this is simply a bookeeping issue that introduces no real difficulty. Thus, the rest of this section focuses primarily on algorithms for bilevel estimation. Except where noted, the corresponding multilevel versions are obvious.

3.1 Independent Color Assignment

If the glyphs in Z are spaced far enough apart (or the template canvasses are small enough) that the glyph image regions do not overlap, then the template disjointness constraint is satisfied trivially. In that case, ATE reduces to separately maximizing each term of the right hand side of (26) and the bilevel template pixel color assignment rule becomes

$$q_t(\vec{x}) = \begin{cases} 1 & \text{if } S(t,\vec{x}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

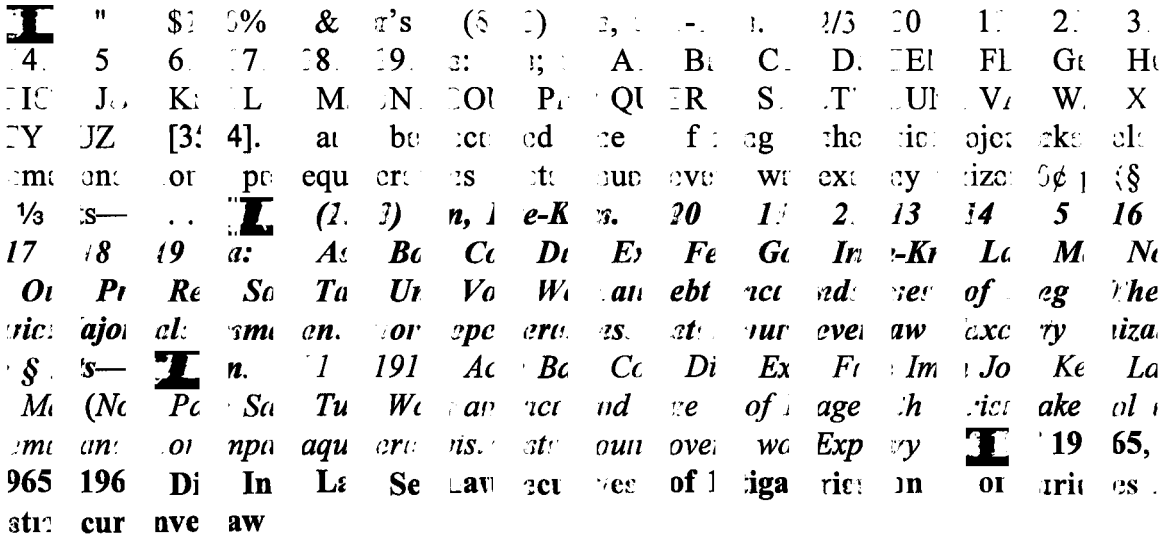


Figure 4: Templates estimated by independent application of (27) to each canvas pixel.

where

$$S(t,\vec{x}) \;\equiv\; \gamma \sum_{i=1}^{N_t} z(\vec{x}+\vec{x}_i^{(t)}) \;+\; \beta N_t. \tag{28}$$

If the foreground is write-black, the condition $S(t,\vec{x}) > 0$ can be written as

$$\frac{1}{N_t}\sum_{i=1}^{N_t} z(\vec{x}+\vec{x}_i^{(t)}) \;>\; \left|\frac{-\beta}{\gamma}\right| \tag{29}$$

which has a simple interpretation. The left hand side of (29) is the fraction of pixels at $\vec{x}$ that are black in the glyph image regions for Q_t. The color assignment rule prescribes that the template pixel at $\vec{x}$ should be black if the fraction of black pixels at $\vec{x}$ in the aligned glyph image regions exceeds a threshold. Simply, the template is computed by averaging and thresholding the aligned glyph image regions. Similar results hold for write-white foregrounds, except that the write-white analog of (29) has the sense of the inequality reversed.

Fig. 4 shows a set of templates generated by (27) in a situation where the glyph image regions overlap significantly. The templates were estimated from 20 scanned pages of text in 12pt Times family fonts drawn from a large legal document [3]. The dominant style is Roman, with

small amounts of BoldItalic, Italic and Bold. The training data contains a total of 212 characters, where characters in different fonts are considered to be distinct. The mean number of glyphs of each character varies from about 500 for Roman to 2 for Bold. Indeed, the Bold text consists solely of the string "**District Securities Investigation Law of 1965**" in which 13 characters appear just once. The fonts in fig. 4 are shown in order of decreasing training glyph count (Roman, BoldItalic, Italic, Bold). The template in the upper left corner is a write-white space template [14].

Each of the templates clearly shows the "correct" character shape, aligned at the origin of the canvas, plus "extra" foreground pixels due to neighboring glyphs in the glyph image regions. The number of extra pixels is relatively small for the most common characters of the Roman font and increases as the amount of training data decreases. Overall, the Roman templates are remarkably clean, suggesting that with a sufficiently large and diverse training set most of the extra foreground would be eliminated as a consequence of the averaging process in (29).

When the glyph image regions overlap, independent application of (27) does not produce a set of maximum likelihood templates, even if the disjointness constraint is removed. However, it is clear that $S(t, \vec{x}) > 0$ is a *necessary* condition for $(t, \vec{x})$ to be a foreground pixel in a maximum likelihood template, i.e., (27) defines a superset of the ML foreground pixels. For this reason, (27) is useful for generating initial sets of candidate foreground pixels in ATE algorithms.

3.2 Greedy Color Assignment

Maximizing (26) subject to the template disjointness constraint is an NP-complete problem, as shown in appendix A. Indeed, the problem remains NP-complete even if the disjointness constraint is removed. The implied worst-case exponential complexity would not necessarily be troublesome if ATE problems qualified as "small". However, it is not uncommon to have 100,000 or more candidate foreground pixels. For comparison, we have found that exhaustive evaluation of alternative pixel assignments is feasible for up to about 20 pixels.

The NP-completeness of ATE implies that a practical estimation algorithm cannot guarantee that (26) will be maximized. Nevertheless, for image decoding applications it does appear possible to construct effective templates without this guarantee. Fig. 5 shows a simple greedy pixel coloring algorithm that is useful by itself and also forms the basis for more sophisticated algorithms. The

procedure $(\mathcal{B}, \mathcal{C}, Z)$ ***do begin***
$q_t(\vec{x}) := 0 \;\; \forall t \in \mathcal{B}, \vec{x} \in \mathcal{C}$
$\mathcal{Q} := \{(t, \vec{x}) \mid S(t, \vec{x}) > 0\}$
while $\mathcal{Q} \neq \emptyset$ ***do begin***
$(s, \vec{w}) := \arg \max_{(t,\vec{x}) \in \mathcal{Q}} S(t, \vec{x})$
$q_s(\vec{w}) := 1$
$\mathcal{Q} := \mathcal{Q} - \mathcal{K}(s, \vec{w}) - \{(s, \vec{w})\}$
end
end

Figure 5: A simple greedy template pixel color assignment algorithm.

algorithm begins by assigning all pixels to the background and constructing a set of candidate foreground pixels $\mathcal{Q}$ using (27). The algorithm then iteratively selects the single most promising pixel $(s, \vec{w})$ from $\mathcal{Q}$ and assigns it to the foreground. In case of a tie, one of the maximum score alternatives is selected randomly. The selected pixel and all of its conflicts $\mathcal{K}(s, \vec{w})$ are removed from $\mathcal{Q}$. The iteration continues until $\mathcal{Q}$ is empty.

Fig. 6 shows a set of templates generated by the greedy algorithm from the training data of the previous example. Compared with the independent templates the greedy templates have significantly fewer extra foregound pixels. However, the greedy algorithm exhibits two failure modes that motivate refinements to the procedure.

The first problem arises because the greedy algorithm repeatedly selects the single best candidate and does not consider groups of multiple non-conflicting candidates. Several of the templates in fig. 6 have "holes" in them, most notably the BoldItalic numerals. The BoldItalic parentheses, conversely, have extra pixels that exactly match these holes. Each of the BoldItalic numerals except zero occurs adjacent to a parenthesis at least once in the training data. The holes arise because the foreground pixels of the parentheses are assigned before those of the numerals— both left and right parentheses occur more frequently than any single numeral (but less frequently than all numerals combined). To satisfy the disjointness constraint, the conflicting pixels in the numerals are then assigned to the background. A preferable (higher likelihood) assignment would close the holes in all of the numerals and remove the conflicting pixels from the parentheses.

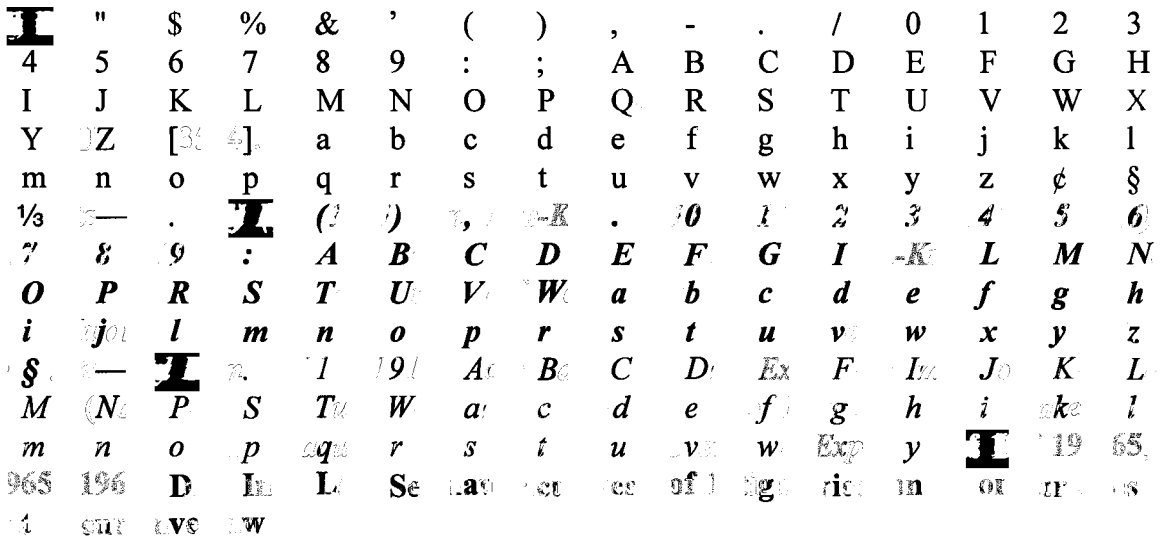


Figure 6: Templates estimated using the greedy algorithm.

The second problem arises because the greedy algorithm uses a random selection procedure to break ties. The effect of this is evident in the Italic "*E*" and "*x*" templates in fig. 6. The training data contains one sample of each of these characters, in the word "*Explanatory*". Every allowable distribution of the foreground pixels of "*Ex*" between the two templates has the same score and thus the maximum likelihood templates are inherently ambiguous. A similar situation arises with BoldItalic "-*K*" and Bold "**1965**."

The next section describes modifications to the greedy algorithm that address these two problems.

3.3 Color Assignment Refinement

A simple approach to eliminating the template "hole" problem noted above is to follow the greedy algorithm with a refinement procedure that examines perturbations to the set of identified foreground pixels. Fig. 7 shows an example of such a refinement procedure. The inputs to the algorithm include an initial assignment that is assumed to satisfy the disjointness constraint. The algorithm attempts to improve on the initial assignment while preserving disjointness. The basic strategy during each iteration is to identify a small set of pixels whose assignments can be modified, as a group,

```
procedure (B, C, Z) do begin
    Q := {(t, x⃗) | S(t, x⃗) > 0}
    while Q ≠ ∅ do begin
        (s, w⃗) := arg max_{(t,x⃗)∈Q} S(t, x⃗)
        if q_s(w⃗) = 1 do begin
            W := K(s, w⃗)
            F := {(t, x⃗) | q_t(x⃗) = 1}
            V := ⋃_{(t,x⃗)∈W} K(t, x⃗) ∩ F
            make assignments to W ∪ V
        end
        Q := Q − {(s, w⃗)}
    end
end
```

Figure 7: A pixel color assignment refinement procedure.

without considering pixels outside the set. The set is constructed by considering two levels of conflicts from a single "root" pixel $(s, \vec{w})$ that is in the foreground in the current assignment. The root pixels are selected by considering the members of a candidate set $\mathcal{Q}$ in order of decreasing score. The set $\mathcal{W}$ consists of all the pixels that conflict with the root. Since the disjointness constraint is assumed to hold, each of the pixels in $\mathcal{W}$ is 0. The set $\mathcal{V}$ contains all of the conflicts with $\mathcal{W}$ that are currently in the foreground (including the root pixel).

It is not hard to see that no set of assigments to the pixels in $\mathcal{W} \cup \mathcal{V}$ can introduce a foreground conflict with any pixel outside the set. Thus, the assignments to $\mathcal{W} \cup \mathcal{V}$ may be optimized, subject to the disjointness constraint, without regard for any other pixels. If $\mathcal{W} \cup \mathcal{V}$ contains fewer than about 20 members exhaustive optimization is feasible. For larger sets a useful approach is to compare the current assignment with just one alternative, created by applying the greedy algorithm first to $\mathcal{W}$ and then to the pixels of $\mathcal{V}$ that do not conflict with the resulting foreground pixels in $\mathcal{W}$. The refinement procedure can be applied repeatedly; we have found that after four or five iterations the change in likelihood typically becomes insignificant.

Fig. 8 shows the result of four iterations of the refinement procedure on the templates of fig. 6. The holes are filled and all of the templates, except the Bold ones and those with inherent

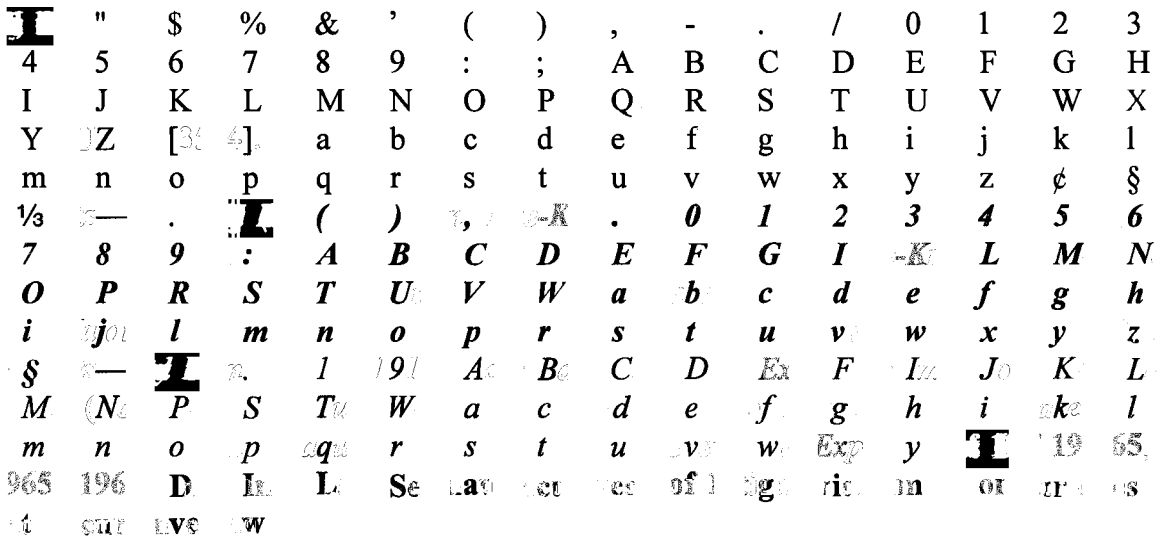


Figure 8: Templates derived from fig. 6 after four iterations of the refinement procedure.

ambiguities, are of fairly high visual quality.

Finally, fig. 9 shows the result of a simple modification to the refinement procedure intended to resolve some of the cases of template ambiguity. Rather than randomly select among equal likelihood assignments to $\mathcal{W} \cup \mathcal{V}$, an assignment was chosen with minimum total absolute horizontal distance of the foreground pixels from the canvas origin. This has the effect of prefering templates whose supports are localized near the origin. Comparing fig. 9 with fig. 8 shows that the result can be a significant improvement in visual template quality.

4 Channel Parameter Estimation

Maximizing $\Pr\{Z \mid Q_{\Pi',\Theta'}, \vec{\alpha}\}$ over $\vec{\alpha}$ in (21) is straightforward by noting that

$$\log \Pr\{Z \mid Q, \vec{\alpha}\} \;=\; \sum_{l=0}^{L-1} \left[\log \frac{\epsilon_l}{1-\epsilon_l} \, \|\tilde{Q}^{(l)} \wedge Z\| \;+\; \log(1-\epsilon_l) \, \|\tilde{Q}^{(l)}\| \right] \tag{30}$$

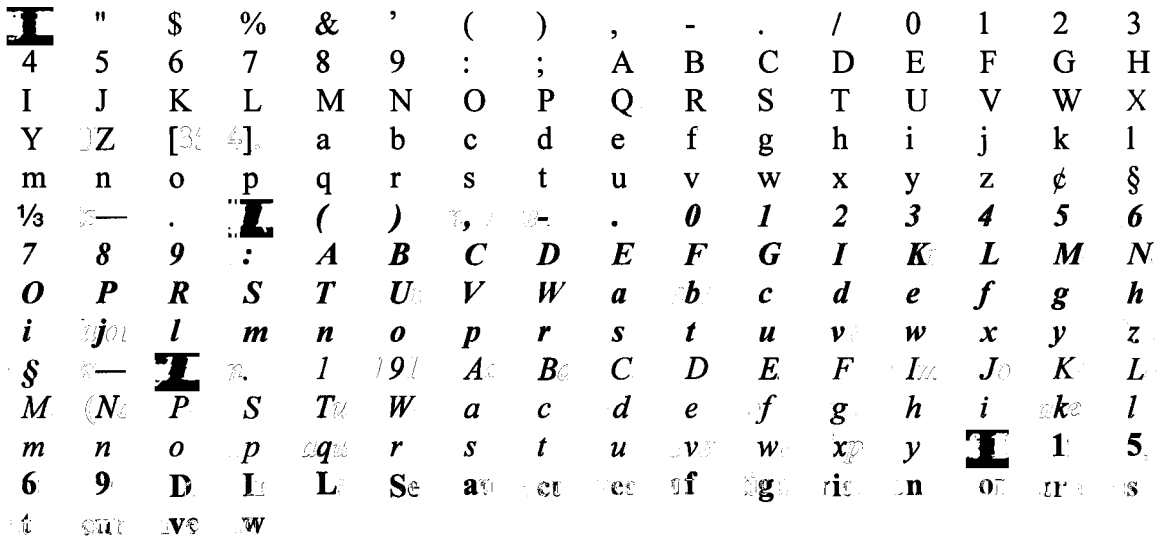


Figure 9: Templates derived from fig. 6 using the refinement procedure with a distance-based tie breaking criterion.

where $\epsilon_0 = 1 - \alpha_0$ and $\epsilon_l = \alpha_l$ for $l > 0$ and the subscripts on Q have been dropped for simplicity. Setting each partial derivative with respect to α_l to zero leads immediately to

$$\hat{\alpha}_0 = 1 - \frac{\|\tilde{Q}^{(0)}_{\Pi',\Theta'} \wedge Z\|}{\|\tilde{Q}^{(0)}_{\Pi',\Theta'}\|} \tag{31}$$

and

$$\hat{\alpha}_l = \frac{\|\tilde{Q}^{(l)}_{\Pi',\Theta'} \wedge Z\|}{\|\tilde{Q}^{(l)}_{\Pi',\Theta'}\|} \tag{32}$$

for $l > 0$.

One issue that arises in the estimation of α_0 is the extent of images Q and Z. The simplest choice is to make them coextensive with the page image. However, if the page includes halftones or other dense non-text regions the estimated value will be too low. An alternative is to consider only the text block regions. However, this makes $\hat{\alpha}_0$ depend on the exact shape and size of the block boundaries as well as on the interline spacing. A third possibility is to consider just the immediate neighborhoods of the text lines. However, this introduces the problem of how to define such

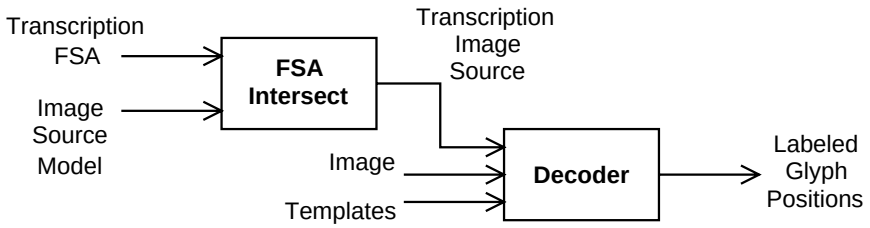


Figure 10: Transcription alignment by constrained decoding.

neighborhoods. Fortunately, experience suggests that neither the templates nor the decoding performance is very sensitive to the exact value of α_0 as long as the other parameters are optimized. For this reason, we typically set α_0 to a nominal value (e.g. .99) and exclude it from the estimation process. The above problem does not arise with the foreground channel parameters because the support of Q covers just the relevant text image pixels.

5 Transcription Alignment

Transcription alignment (19) involves finding a complete path Π that maximizes $\mathcal{L}(Z \mid Q_{\Pi,\Theta}) + \log \Pr\{\Pi\}$, subject to the constraint that $M_\Pi = M$, where M is the given transcription. This is a constrained decoding problem that is similar to the speech recognition problem of training hidden Markov models using sentence-level transcriptions [18]. By analogy with the speech case, transcription alignment can be carried out using the two-step procedure shown in fig. 10.

The first step is to combine the image source model and the transcription into a *transcription image source*. The transcription image source is defined as a source that generates the subset of the original source images consisting of just those whose message is M. If the transcription is represented as a finite-state automaton (FSA) then the transcription image source can be constructed using a straightforward extension to the standard algorithm for intersecting two finite-state automata [6]. The training image Z is then decoded using the transcription image model and the usual dynamic programming decoding algorithm [11]. The resulting complete path defines an alignment between M and Z.

5.1 Image Source Models for Alignment

A straightforward way to train templates for a decoder is to use the decoder's source model as the image source and use source language messages as transcription strings. However, full-page models and transcriptions can result in transcription image sources that are impractically large. The number of nodes in the transcription image model can be as large as the product of the number of nodes in the image source and the number of characters in the transcription. The complete model for a moderately complex page, such as the yellow page column decoder described in [11], can easily contain 1,000 nodes. For a model of this size, a 3,000 character page transcription can thus result in a transcription image model with several million nodes. Fortunately, simple source models are usually adequate for template estimation, even when the decoder model is complicated.

There are two main situations that typically call for a complex decoder. The first is when a large language model (e.g. exhaustive word lexicon) is used to constrain the sequence of recognized characters. This usually occurs because image degradation or character shape variation makes it impossible to achieve adequate recognition accuracy without tight constraints. The second situation is when detailed information about the logical structure of the recognized text is required. In that case the model must include markup (e.g. SGML tags) that cannot be directly transcribed from the image itself.

Neither of the above situations applies to transcription alignment. The purpose of alignment is to determine the spatial positions of a sequence of glyphs whose character labels are given. For this purpose, the logical structure of the text is irrelevant. In addition, if the transcription is known to satisfy all relevant constraints on character order such constraints need not be represented in the source model.

When images of individual text lines can be reliably extracted using simple segmentation techniques the alignment process can be further simplified by using one-dimensional line sources rather than full two-dimensional models [8]. We describe two simple line sources for template estimation from segmented line images and their transcriptions. These models have been found useful in training single- and multiple-font template sets, respectively.

Fig. 11(a) shows a text line source that generates images of unconstrained text strings. The branches labeled "a" ... "z" represent the templates to be trained. The transcription image mod-

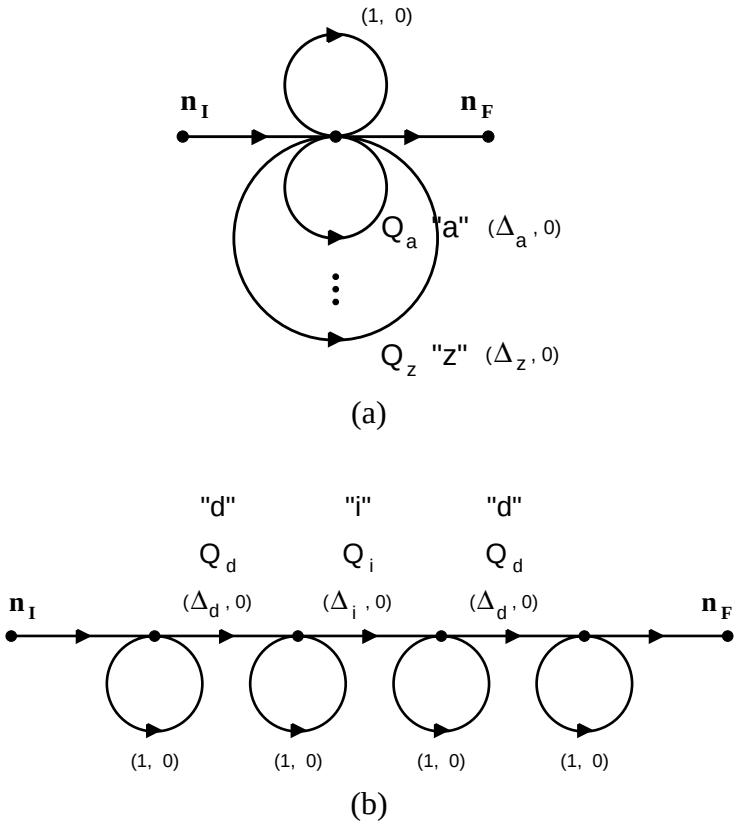


Figure 11: Simple alignment sources. (a) Text line image source. (b) Transcription image source for “did”.

els for this source have the form illustrated in fig. 11(b), which shows a source for images of the string “did”. The model consists of a chain of transitions for the characters of the string plus unit displacement self-transitions for horizontal spacing adjustment.

The model in fig. 11(a) can also be used to train multiple templates for each character (i.e. multiple fonts). If the character labels are augmented to include both a font identifier and a base character code (e.g., as in a PostScript *composite font* [1]), multi-font training reduces to the single font case. However, this approach can greatly increase the cost of preparing the training data since the font of each glyph must be explicitly noted in the transcription. Alternatively, the source can simply include multiple template branches labeled with each character. In that case, the transcription image source will have multiple parallel branches for each character of the transcription. For example, if the source in fig. 11(a) includes two “d” branches, each of the branches labeled “d” in

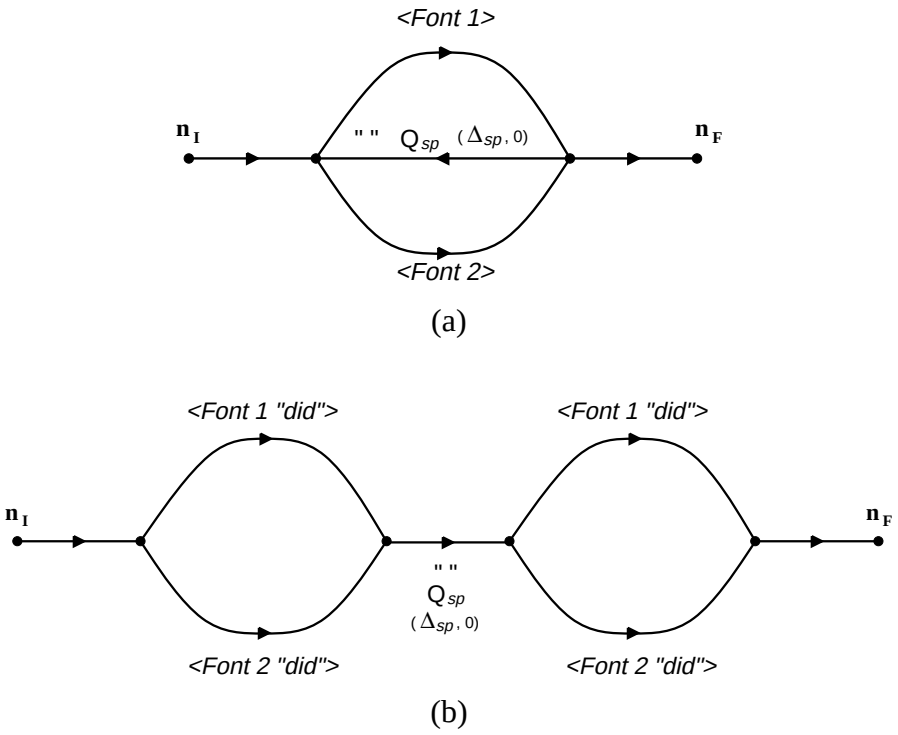


Figure 12: Alignment sources with constrained font changes. (a) Text line image source. (b) Transcription image source for “did did”.

fig. 11(b) would be replaced by a pair of branches. During decoding, the “d” branch whose template better matches the image will be included in the alignment path.

The approach just described is attractive because it avoids the need to include font information in the transcription. However, it places no constraints on when font changes can occur. In ordinary typeset text it is relatively uncommon for the font to change within a word. When estimating templates from such text, we have found it preferable to use a source that restricts font changes to word boundaries. Fig. 12(a) shows an example of a multi-font source that constrains font changes to occur only immediately after a space character. The branches labeled $<$*Font 1*$>$ and $<$*Font 2*$>$ are single-font line sources as in fig. 11(a). For simplicity, a single interword space template is shown although different templates could be used for each font. Fig. 12(b) shows a transcription image source for the two-word string “did did” generated using the constrained font change model. Each of the branches labeled $<$*Font 1 “did”*$>$ and $<$*Font 2 “did”*$>$ represents a copy of the model in fig. 11(b).

5.2 Initial Conditions

The iterative training algorithm can be started with any of the three steps (19), (20) or (21). Most often we anticipate beginning with the alignment step, which requires a set of initial templates and channel parameters. We have found that the training procedure is not very sensitive to the exact shapes or sizes of the initial templates or to the initial channel parameters. One way to obtain initial templates is to excise a sample of each required character from the training images using an interactive tool [5]. Alternatively, initial templates can be derived from an existing bitmap or outline font, as described in the following section.

6 Experimental Results

The template estimation procedure has been applied to a variety of document types including newspaper columns, 15^{th} century books, degraded microfilm images of 19^{th} century newspapers, connected text in script-like fonts and mathematical notation [12, 13, 7]. In this section we report on its use to create document-specific templates for the English journal subset of the Univ. of Washington UW-II database [17]. The quality of the templates was assessed by their ability to support high-accuracy recognition, using the commercial OCR program *TextBridge*[4] for comparison. It is acknowledged that recognition accuracy is at best an indirect measure of template estimation performance, since it depends on a number of factors besides the templates (e.g. language model, character set size, image quality). However it does provide a crude indication of overall template quality since poor templates are unlikely to support accurate recognition.

The UW-II database contains bilevel scanned images (300 ppi) of 311 pages from 30 technical journal articles. Two images of each page are provided, scanned from first and second generation photocopies, respectively. The pages are zoned and each zone is described by manually prepared records that specify attributes such as zone type, bounding box coordinates and ground truth text. A subset of the "text-body" zones was used, consisting of those containing only 92 standard ascii characters (space, A–Z, a–z, 0–9, left and right double quotes, and the 27 punctuation symbols `!"#$%&''()+,-./[]_:;<>?@{}=`) with no subscripts, superscripts, overlines, un-

[4] Xerox ScanSoft, Peabody, MA.

derlines or dot sequences. We will call these zones the *simple text-body zones*. The UW-II database includes 3400 simple text-body zones on 540 pages, comprising 1.15 million glyphs (1.34 million including space characters).

A document-specific decoding scenario was enacted separately for each of the articles. The simple text-body zones for the article were divided into disjoint training and test sets by partitioning the page images into two collections, each containing a complete set of article pages with approximately equal numbers of first and second generation copies in each set. This insured that the character set, fonts and image quality of the training data matched those of the test set. The articles in the UW-II database have names of the form "W0x" and "W1x", where the middle digit identifies the copy generation number and the final character is a digit or uppercase letter. We will let "Wx-Tr" and "Wx-Te" refer to the derived training and test articles, respectively.

Text line baselines for all zones were estimated using a least-squares procedure, similar to that described in [10], to fit fifth order polynomials to glyph image positions obtained from a modified version of *TextBridge*.

6.1 Template Training

Templates and channel parameters were estimated from the training pages using either the ground truth text or the output of *TextBridge* as the training transcription. The templates included three foreground levels— two write-black and one write-white— as suggested by a previous investigation of multilevel templates [14]. The initial channel parameters were $\alpha_0 = .99$, $\alpha_1 = .9$, $\alpha_2 = .001$ and $\alpha_3 = .6$. The initial font for each article was a composite font constructed from one to four bilevel fonts drawn from a collection of 300ppi bitmap fonts generated using PostScript font programs. The templates of the PostScript fonts were used as the initial $\tilde{Q}^{(1)}$ level sets of the composite font; the remaining initial foreground level sets were empty. Each template canvas was 10 pixels wider than the corresponding PostScript font template. All template canvases had the same height, defined by the vertical projection profile of the PostScript fonts.

The dictionary of base fonts included three Times family typefaces (Roman, Bold, Italic), two Helvetica typefaces (Bold and BoldOblique) and Courier, in integral point sizes from 7 through 11 pts. Each base font contained the 92 characters enumerated above. The base fonts for an article

were selected on the basis of visual inspection of the training pages. Times, Helvetica and Courier family fonts were included when the article contained serif, sans-serif and fixed-pitch text, respectively.

Five iterations of the alignment-ATE-ACE cycle were carried out using the constrained font change image source of fig. 12(a). The glyphs positions typically converged after 3 or 4 iterations. Following the final iteration, five additional ATE-ACE cycles were performed to insure convergence of the template and channel parameters. After training, all untrained templates (i.e. those with $N_t = 0$) were removed from the font. The mean numbers of characters and trained templates per article were 72.5 and 115.7, respectively. The total number of templates for the thirty articles was about 3500. The mean estimated channel parameters were $\alpha_1 = .97$, $\alpha_2 = 5.6 \times 10^{-5}$ and $\alpha_3 = .44$.

The time required to train an article font grew roughly linearly with the length of the article and the number of templates. The shortest article (W1-Tr) contained 8,700 glyphs (165 text lines, 52.9 glyphs/line) and required about 11 minutes to train 106 templates using the greedy algorithm (7.5 msec/line/iteration/template). The longest article (WG-Tr) comprised 57,900 glyphs (1111 lines, 52.2 glyphs/line) and required about 2 hours to train 176 templates (7.7 msec/line/iteration/template).

6.2 Decoder

The test pages were decoded using the unconstrained text line decoder of fig. 11(a) with a vertical jitter of ± 1 pixel [9]. A complete specification of the decoder requires values for the horizontal displacements of the template branches (i.e. $\Delta_a \ldots \Delta_z$). A template branch displacement defines the minimum distance from the origin of the template to the origin of the next template in a line of text and corresponds to typographic set width. The displacement for each template was estimated by forming a histogram of the distances from glyphs labeled with that template to the following glyph, using the final alignment glyph positions. The branch displacement was taken to be the tenth percentile of this distribution.

The decoding time ranged from about 6 minutes for W1-Tr (2.33 sec/line) to 110 minutes for WG-Tr (5.91 sec/line).

ATE	Errors	% Error	#1	#2
Greedy	3478	0.52	l ⇒ I (1077)	ϵ ⇒ s (191)
Refined	3404	0.51	l ⇒ I (871)	ϵ ⇒ . (208)
TextBridge	4978	0.74	ϵ ⇒ i (322)	u ⇒ t (175)

(a)

ATE	Errors	% Error	#1	#2
Greedy	1746	0.29	ϵ ⇒ s (191)	ϵ ⇒ f (174)
Refined	1798	0.30	ϵ ⇒ . (201)	ϵ ⇒ f (173)
TextBridge	4258	0.70	ϵ ⇒ i (318)	u ⇒ t (175)

(b)

Table 1: Character recognition results for Univ. of Washington English Document Database II (simple text-body zones). (a) Results for complete 30 article test set. (b) Results for 28 articles (W6-Te and WA-Te excluded).

6.3 Results Using Ground Truth Transcriptions

Table 1(a) summarizes the overall character recognition performance on the 30 test articles for templates estimated using the greedy and refined ATE procedures and the ground truth text strings. The error counts are the total numbers of character substitutions, deletions and insertions computed by a standard string alignment procedure. The columns labeled "#1" and "#2" identify the two most frequent substitution (e.g. l ⇒ I), insertion (e.g. ϵ ⇒ s) or deletion (e.g. *sp* ⇒ ϵ) errors and their frequencies.

The data suggest only a minor difference between the greedy and refined templates. The error rate using either type is about 0.5%, which corresponds to 30% fewer errors than *Textbridge*. As noted previously, the refined templates frequently have superior visual quality. While this can be important for imaging-oriented applications such as font generation and error visualization [12] it does not appear very significant for decoding.

The greedy template decoder made fewer errors than *TextBridge* on 22 of the articles; on those articles *TextBridge* made 4.84 times as many errors. Conversely, on the remaining eight articles the decoder made 2.08 times as many errors as *TextBridge*.

Two of the articles, comprising about 10% of the text, accounted for about half of the decoding errors. Inspection of these articles indicated that the poor decoder performance was not the

result of template estimation problems.

Article W6-Te was scanned from a bound journal and has a significant amount of text at the spine-side page edges that appears as unreadable black blotches. These blotches were decoded as long strings of ‘i’ and ‘l’ characters, which resulted in a large number of insertion errors in addition to complete loss of the actual text. The decoder error rate with greedy templates was 2.1%, compared with 1.5% for *TextBridge*.

Article WA-Te is printed in a sans-serif font in which ‘l’ and ‘I’ are almost indistinguishable. This one article accounted for nearly all of the ’l’ $\Rightarrow$ ’I’ substitution errors made using greedy templates (911 of 1077). The decoder error rate on WA-Te was 2.9%, compared with 0.29% for *TextBridge*. Like most commercial OCR programs, *TextBridge* uses sophisticated lexical postprocessing to resolve ambiguities such as sans-serif ’l’ and ’I’ confusions.

Table 1(b) gives statistics when the results for W6-Te and WA-Te are excluded. The overall error rate is 0.3%, i.e. 60% fewer errors than *TextBridge*.

6.4 Results After Template Editing

Investigation of the most frequent decoding errors revealed that some of them could be attributed to gross mistakes in template shape or alignment. Article W5-Tr included a region of text that was almost totally washed out, resulting in three templates (’f’, ’o’, ’s’) that consisted of just a few isolated blobs. These templates were responsible for the ’s’ and ’f’ insertions indicated in table 1. Another thirty templates in 11 articles were either severely misregistered with respect to their origins or had unreasonably large or small estimated set widths.

Table 2 shows results obtained after deleting the three mis-estimated templates from the W5-Tr font and manually correcting the gross font metric errors. About 40% of the errors on the 28 article test set were eliminated by editing less than 1% of the templates (33 of 3500). This suggests that manual template editing can be cost-effective in a labor-intensive conversion process if it leads to a reduced need for proofreading. Of course, editing is not an option in a fully automatic system.

ATE	Errors	% Error	#1	#2
Greedy	2883	0.42	l ⇒ I (1077)	$\epsilon \Rightarrow sp$ (150)
Refined	2705	0.40	l ⇒ I (871)	$\epsilon \Rightarrow$ i (210)

(a)

ATE	Errors	% Error	#1	#2
Greedy	1101	0.18	l ⇒ I (166)	$sp \Rightarrow \epsilon$ (135)
Refined	1099	0.18	$\epsilon \Rightarrow sp$ (142)	$sp \Rightarrow \epsilon$ (130)

(b)

Table 2: Character recognition results after font editing. (a) 30 articles. (b) 28 articles.

Spaces?	Errors	% Error	#1	#2
Y	5797	0.86	$\epsilon \Rightarrow sp$ (2484)	l ⇒ I (986)
N	3047	0.53	l ⇒ I (985)	i ⇒ l (209)

(a)

Spaces?	Errors	% Error	#1	#2
Y	4349	0.72	$\epsilon \Rightarrow sp$ (2444)	l ⇒ I (419)
N	1649	0.32	l ⇒ I (419)	i ⇒ l (102)

(b)

Table 3: Character recognition results after training with errorful transcriptions and font editing. (a) 30 articles. (b) 28 articles.

6.5 Results Using Errorful Transcriptions

The above results are based on training using a perfect ground-truth transcription. This raises the issue of sensitivity to the accuracy of the training text. Table 3 shows decoding results for templates trained using the errorful *TextBridge* output rather than the ground truth strings. Results are shown after font editing; error rates before editing were 20–80% higher. Not surprisingly, more editing was required than with perfect transcriptions. Twenty-six templates were deleted, 55 required origin or set width adjustments and 12 template support regions were reduced in width. Support width reduction involved clearing a vertical strip of pixels at the left or right edge of the canvas to remove extraneous foreground pixels similar to those of the Bold templates at the bottom of fig. 9.

The columns labeled "spaces?" indicate whether or not space characters are included in the scoring. When included, space errors account for about half of the total errors and cause the overall decoding accuracy to be less than that of *TextBridge*. If space is excluded, the decoder error rate on the 28 article set is about 40% less than that of *TextBridge*. The possibility of eliminating space errors by font editing or automatic means was not investigated.

7 Summary

A maximum likelihood approach to supervised character template estimation using unaligned transcriptions has been developed. The kernel of the approach is a method of aligned template estimation (ATE) that allows the bounding boxes of adjacent glyphs to overlap. While the method was developed specifically to support the creation of document-specific decoders in the DID framework, it is applicable to any situation in which character templates are to be derived from sample images.

8 Acknowledgements

The authors wish to thank Phil Chou for many stimulating discussions throughout the course of this work. The legal document discussed in section 3 was provided by the Univ. of California Environmental Digital Library Project, headed by Robert Wilensky and supported by National Science Foundation grant IRI-9411334 as part of the NSF/NASA/ARPA Digital Library Initiative. Ben Wittner of Xerox ScanSoft provided the version of *TextBridge* used in the experiments.

A NP-Completeness of Aligned Template Estimation

Aligned template estimation (ATE) is the problem of choosing Θ to maximize $\mathcal{L}(Z \mid Q_{\Pi,\Theta})$ in (26) subject to the template disjointness constraint. The theory of NP-completeness applies to decision problems, i.e. problems whose solution is either "yes" or "no". ATE can be transformed into a decision problem by the standard technique of introducing a numerical parameter B and asking whether there is a Θ for which $\mathcal{L}(Z \mid Q_{\Pi,\Theta}) \geq B$. It is clear that the template disjointness constraint

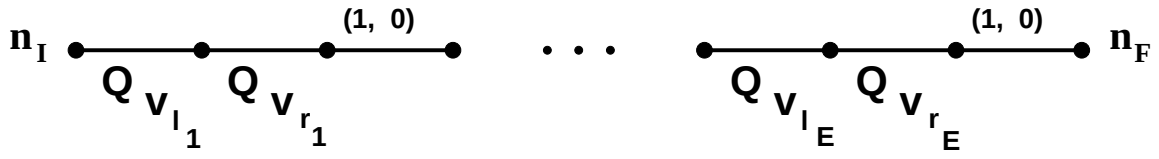


Figure 13: Image source for independent set problem.

and the condition $\mathcal{L}(Z \mid Q_{\Pi,\Theta}) \geq B$ can be verified for a candidate set of templates in polynomial time. Thus, ATE $\in$ NP. We establish that ATE is NP-hard (and thus NP-complete) by showing that any algorithm that solves ATE exactly can be used to solve the *independent set* problem, which is known to be NP-complete [16].

Let $\mathcal{G}$ be an undirected graph consisting of a finite set of vertices $\mathcal{V} = \{v_1 \ldots v_V\}$ and a finite set of edges $\mathcal{E} = \{(v_{l_i},\ v_{r_i}), i = 1 \ldots E\}$. An *independent set* of vertices is a subset $\mathcal{V}' \subset \mathcal{V}$ with the property that no pair of vertices in $\mathcal{V}'$ are joined by an edge in $\mathcal{E}$. Given a positive integer K, the independent set (IS) problem is to decide whether or not $\mathcal{G}$ contains an independent set of size at least K.

Given a graph $\mathcal{G}$ and positive integer K, we construct an ATE decision problem whose solution is equivalent to IS for $\mathcal{G}$ and K. This involves specifying a source model, a complete path Π, an image Z and a likelihood bound B. Fig. 13 shows the source model, a simple chain of $3E$ branches that has exactly one complete path Π. Each graph vertex $v \in \mathcal{V}$ corresponds to a template Q_v whose canvas consists of the single point at the template origin. We will view the template pixels as being indexed by the vertices in $\mathcal{V}$. The branches of the source model occur in groups of three, where each triple corresponds to one edge in $\mathcal{E}$. The first two of the three branches representing edge $(v_{l_i},\ v_{r_i})$ are labeled with templates $Q_{v_{l_i}}$ and $Q_{v_{r_i}}$ and have zero displacement. The third branch has unit horizontal displacement, $(1, 0)$.

It is not hard to see that $\|Q_\Pi\| = E$ and that template pixels u and v conflict if and only if $(u,\ v) \in \mathcal{E}$. Thus, each independent subset $\mathcal{V}' \subset \mathcal{V}$ corresponds to a set of template pixels that do not conflict, i.e. a candidate pixel assignment Θ that satisfies the template disjointness constraint.

If N_v denotes the number of edges attached to vertex v it is clear that $\|\mathcal{I}(v)\| = N_v$. Let each template Q_v be associated with a separate write-black level and let the channel parameters be

given by

$$\alpha_0 = \frac{e}{1+e} \tag{33}$$

$$\alpha_v = \frac{e^{\frac{1}{N_v}}}{1+e}. \tag{34}$$

Then, if Z is taken to be the ideal image Q_Π it is easy to show from (4), (5) and (28) that $S(v) = 1$ for each pixel v. As a result, $\|\mathcal{V}'\| = \mathcal{L}(Z \mid Q_{\Pi,\Theta})$, where Θ is the pixel assignment that corresponds to $\mathcal{V}'$. It should be clear that the ATE decision problem with $B = K$ is equivalent to the given IS problem.

If the template disjointness constraint is removed, a similar construction can be used to show that a solution to the (uncontrained) ATE problem implies a solution to the NP-complete *minimum cover* [16] problem.

References

[1] Adobe Systems Inc., *PostScript Language Reference Manual*, second edition, 1990, Reading: Addison-Wesley.

[2] H. Baird and G. Nagy, "A self-correcting 100-font classifier", in *Document Recognition*, L. Vincent and T. Pavlidis, editors, Proc. SPIE vol. 2181, pp. 106–115, 1994.

[3] California Dept. of Water Resources, *General Comparison of Water District Acts*, Bulletin 155-94, March, 1994.

[4] F. Chen, D. Bloomberg and L. Wilcox, "Spotting phrases in lines of imaged text", *Document Recognition II*, L. Vincent and H. Baird, editors, Proc. SPIE vol. 2422, pp. 256–269, 1995.

[5] T. Fruchterman, "DAFS: a standard for document and image understanding", *Proc. 1995 Symposium on Document Image Understanding Technology*, Bowie, MD, Oct. 24–25, 1995, pp. 94–100.

[6] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*, 1979, Reading: Addison-Wesley.

[7] J. Hull, “Recognition of Mathematics Using a Two-Dimensional Trainable Context-Free Grammar”, M. Eng. Thesis, Massachusetts Institute of Technology, Cambridge, MA, June, 1996.

[8] A. Kam and G. Kopec, “Separable source models for document image decoding”, *Document Recognition II*, L. Vincent and H. Baird, editors, Proc. SPIE vol. 2422, pp. 84–97, 1995.

[9] A. Kam and G. Kopec, “Document image decoding by heuristic search”, *IEEE. Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, Sept. 1996, pp. 945–950.

[10] G. Kopec and P. Chou, “Automatic generation of custom document image decoders”, *Proc. Second Intl. Conf. on Document Analysis and Recognition*, Tsukuba Science City, Japan, Oct. 20–22, 1993.

[11] 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.

[12] G. Kopec and M. Lomelin, “Document-specific character template estimation”, in *Document Recognition III*, L. Vincent and J. Hull, editors, Proc. SPIE vol. 2660, pp. 14–26, 1996.

[13] G. Kopec, “Document image decoding in the Berkeley digital library project”, in *Document Recognition III*, L. Vincent and J. Hull, editors, Proc. SPIE vol. 2660, pp. 2–13, 1996.

[14] G. Kopec, “Multilevel character templates for document image decoding”, in *Document Recognition IV*, L. Vincent and J. Hull, editors, Proc. SPIE vol. 3027, 1997.

[15] 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, Aug., 1994, pp. 842–848.

[16] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, 1982, Englewood Cliffs: Prentice-Hall.

[17] I. Phillips, S. Chen, J. Ha and R. Haralick, *Reference Manual for the UW English/Japanese Document Image Database II*, Version 2.01, ISL report, Dept. of Electrical Eng., Univ. of Washington, Seattle, WA, March 8, 1995.

[18] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, 1993, Englewood Cliffs: Prentice-Hall.

[19] R. Rubenstein, *Digital Typography*, 1988, Reading: Addison-Wesley.

[20] H. Stabler, “Experiences with high-volume, high-accuracy document capture”, in *Document Analysis Systems*, L. Spitz and A. Dengel, eds., 1995, Singapore: World Scientific Publishing.

Index Terms

document image decoding, Markov models, template estimation, character recognition, document recognition, maximum likelihood

Figure Captions

1. Canvas parameters and template coordinate system. The lower left corner of the canvas is at $(-\chi,\ \psi)$ in the template coordinate system.
2. Iterative transcription alignment and aligned template and channel estimation.
3. Glyph image regions for Times Roman “a”. The canvas origin is indicated by a cross.
4. Templates estimated by independent application of (27) to each canvas pixel.
5. A simple greedy template pixel color assignment algorithm.
6. Templates estimated using the greedy algorithm.
7. A pixel color assignment refinement procedure.
8. Templates derived from fig. 6 after four iterations of the refinement procedure.
9. Templates derived from fig. 6 using the refinement procedure with a distance-based tie breaking criterion.
10. Transcription alignment by constrained decoding.
11. Simple alignment sources. (a) Text line image source. (b) Transcription image source for “did”.
12. Alignment sources with constrained font changes. (a) Text line image source. (b) Transcription image source for “did did”.
13. Image source for independent set problem.

Table Captions

1. Character recognition results for Univ. of Washington English Document Database II (simple text-body zones). (a) Results for complete 30 article test set. (b) Results for 28 articles (W6-Te and WA-Te excluded).
2. Character recognition results after font editing. (a) 30 articles. (b) 28 articles.
3. Character recognition results after training with errorful transcriptions and font editing. (a) 30 articles. (b) 28 articles.

Footnotes

0. Gary E. Kopec is with the Xerox Palo Alto Research Center, Palo Alto CA 94304 USA; email: kopec@parc.xerox.com
0. Mauricio Lomelin is with the Microsoft Corporation, Seattle WA; email: mauricil@microsoft.com
1. All complete paths through a source do not necessarily have the same length. However, for notational simplicity the dependence of P on Π will not be indicated.
2. We use an image coordinate system in which x increases to the right, y increases downward, and the upper left corner of the page is at $x = y = 0$.
3. The exceptional definition of α_0 maintains consistency with the notation for bilevel channels defined in [11].
4. Xerox ScanSoft, Peabody, MA.

Author Contact Information

Gary E. Kopec: Information Sciences and Technologies Laboratory, Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304; e-mail: kopec@parc.xerox.com; phone: (415)-812-4454; fax: (415)-812-4374.

Mauricio Lomelin: Microsoft Corp., One Microsoft Way, Redmond, WA 98052-6399; email: mauricil@microsoft.com; phone: (425)-936-0945; fax: (425)-936-7329.

Author Technical Biographies

Gary E. Kopec was born in Cleveland, OH on June 11, 1952. He received B.S., M.S. and Ph.D degrees from M.I.T. in 1973, 1976 and 1980, respectively. From 1980 through the present, he worked for M.I.T., Schlumberger, Atlantic Aerospace Electronics Corp, and Xerox, where he was involved variously, in VLSI design, speech processing and recognition, knowledge-based signal processing, and document image processing and recognition. His current research interests are statistical pattern recognition, image analysis and document recognition. Currently he is a Principal Scientist with the Xerox Palo Alto Research Center in Palo Alto, CA. Dr. Kopec is a member of the ACM and the IEEE Computer and Signal Processing Societies. He was technical co-chairman for the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, held in San Francisco, CA.

Mauricio Lomelin received his BS and MS degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1995. During 1994 and 1995, as a graduate student, he worked at the Xerox Palo Alto Research Center in the Document Image Decoding Group, doing research as part of his Master's thesis on Character Template Estimate. He is currently at Microsoft Corporation working as a Program Manager in the Windows CE Product Unit.

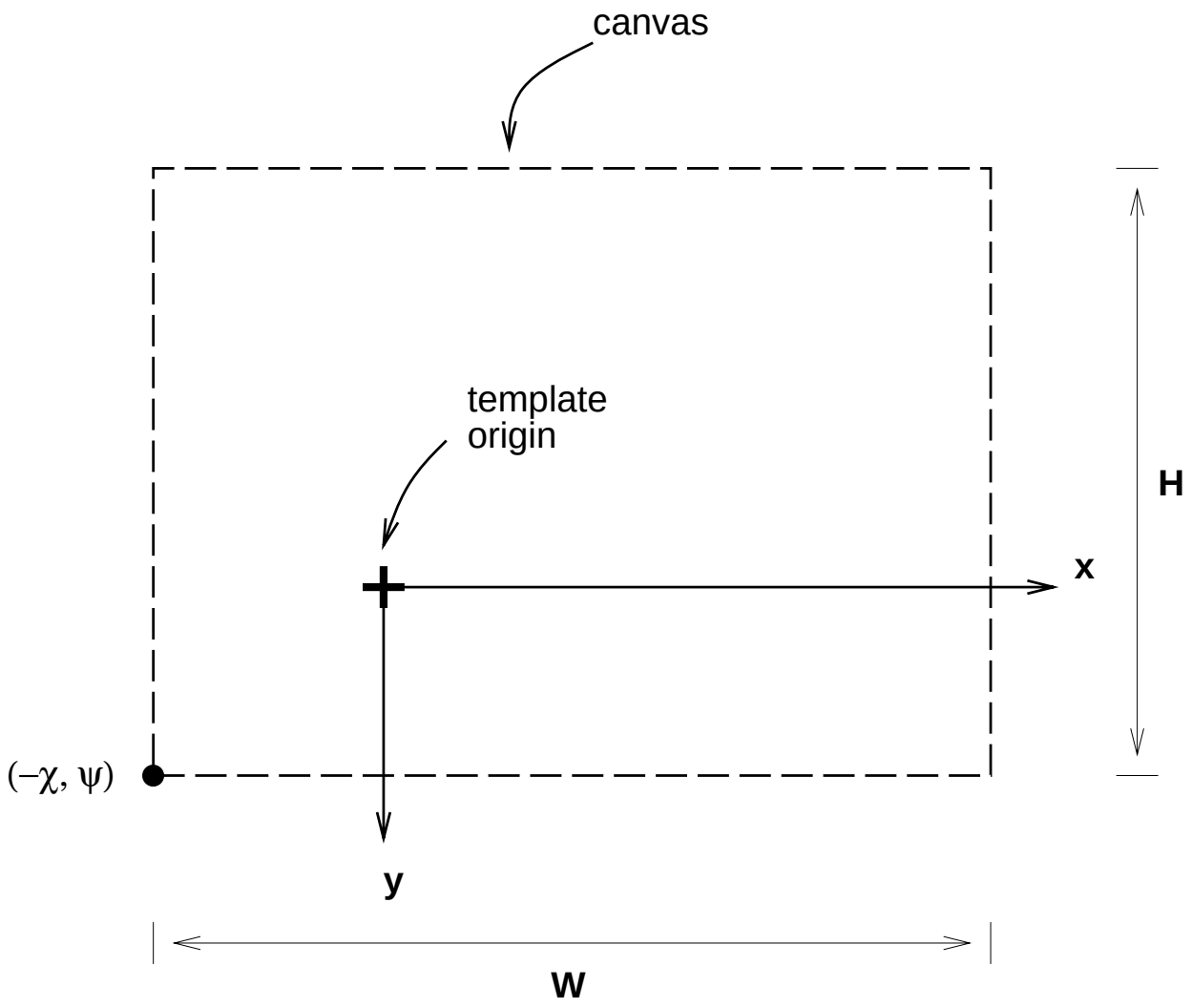


Figure 1: Canvas parameters and template coordinate system. The lower left corner of the canvas is at $(-\chi,\ \psi)$ in the template coordinate system.

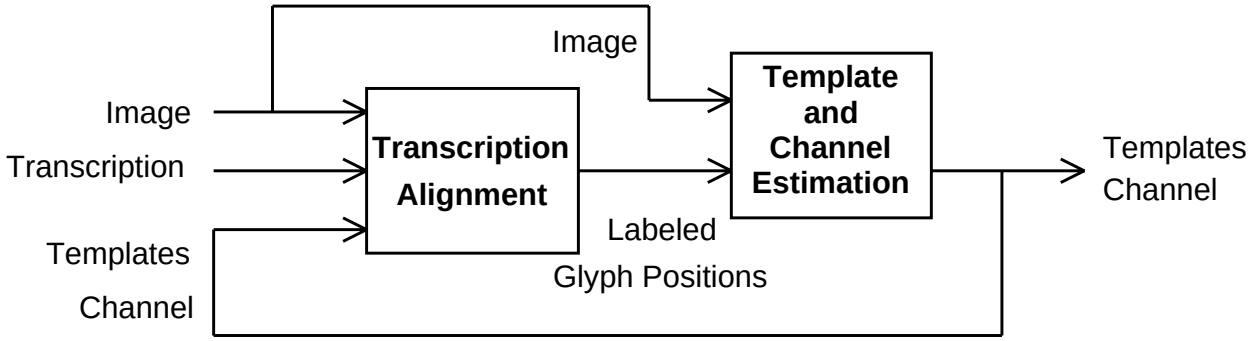


Figure 2: Iterative transcription alignment and aligned template and channel estimation.

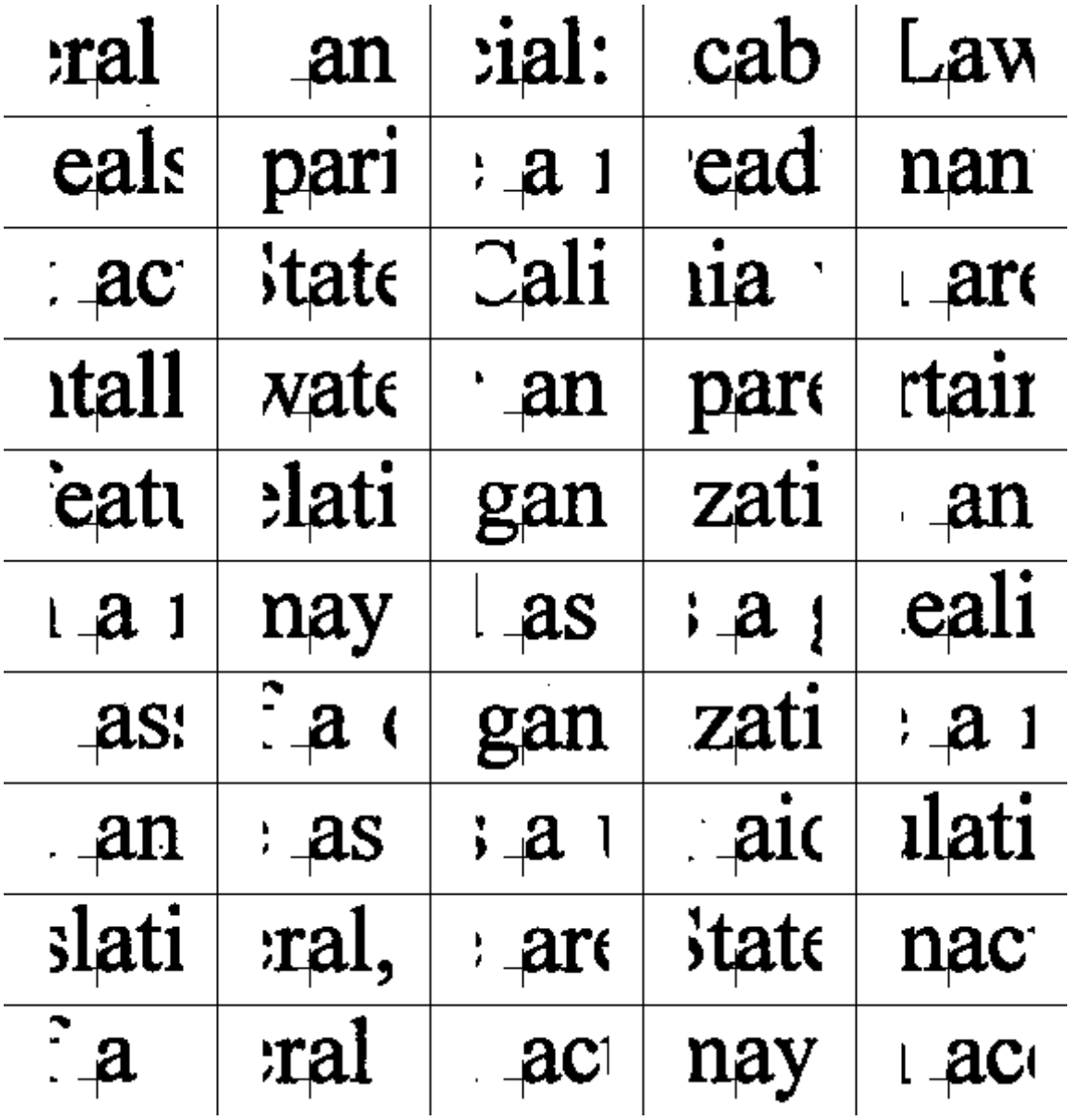


Figure 3: Glyph image regions for Times Roman “a”. The canvas origin is indicated by a cross.

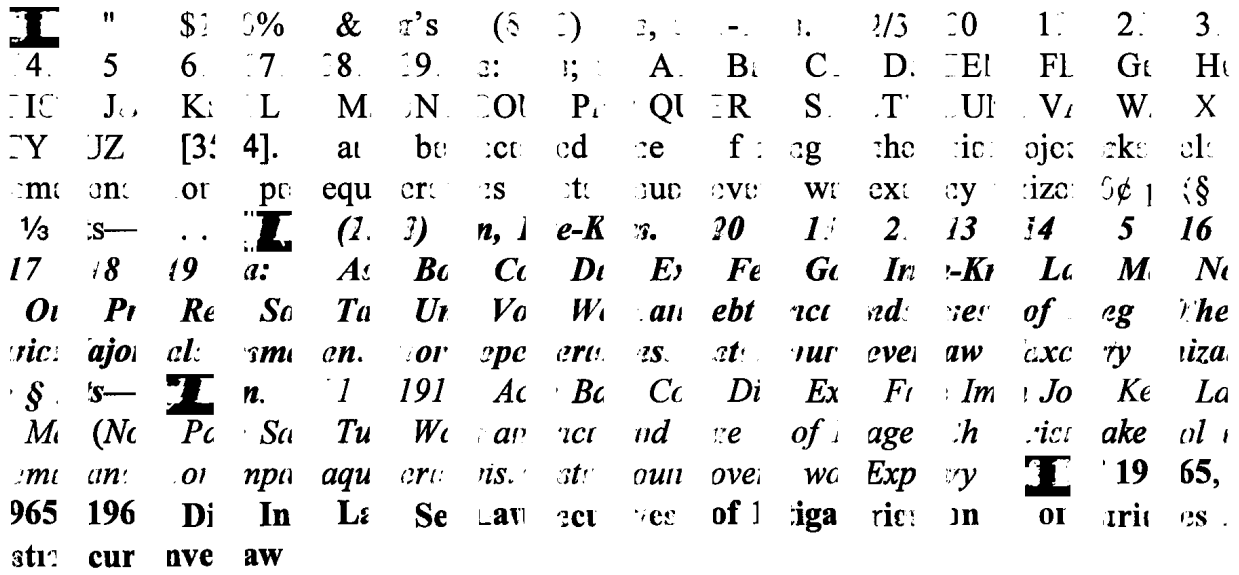


Figure 4: Templates estimated by independent application of (27) to each canvas pixel.

procedure $(\mathcal{B}, \mathcal{C}, Z)$ ***do begin***
$q_t(\vec{x}) \;:=\; 0 \;\; \forall t \in \mathcal{B}, \vec{x} \in \mathcal{C}$
$\mathcal{Q} \;:=\; \{(t, \vec{x}) \mid S(t, \vec{x}) > 0\}$
while $\mathcal{Q} \neq \emptyset$ ***do begin***
$(s, \vec{w}) \;:=\; \arg \max_{(t,\vec{x}) \in \mathcal{Q}} S(t, \vec{x})$
$q_s(\vec{w}) \;:=\; 1$
$\mathcal{Q} \;:=\; \mathcal{Q} - \mathcal{K}(s, \vec{w}) - \{(s, \vec{w})\}$
end
end

Figure 5: A simple greedy template pixel color assignment algorithm.

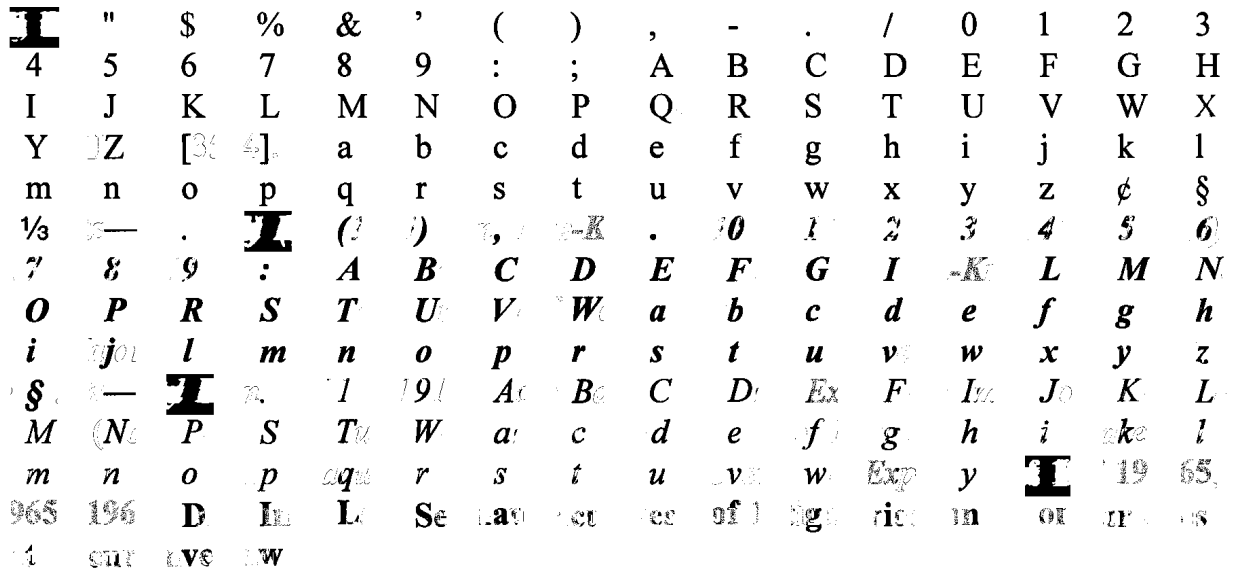


Figure 6: Templates estimated using the greedy algorithm.

procedure $(\mathcal{B}, \mathcal{C}, Z)$ ***do begin***
    $\mathcal{Q} := \{(t, \vec{x}) \mid S(t, \vec{x}) > 0\}$
    while $\mathcal{Q} \neq \emptyset$ ***do begin***
        $(s, \vec{w}) := \arg \max_{(t,\vec{x}) \in \mathcal{Q}} S(t, \vec{x})$
        if $q_s(\vec{w}) = 1$ ***do begin***
            $\mathcal{W} := \mathcal{K}(s, \vec{w})$
            $\mathcal{F} := \{(t, \vec{x}) \mid q_t(\vec{x}) = 1\}$
            $\mathcal{V} := \bigcup_{(t,\vec{x}) \in \mathcal{W}} \mathcal{K}(t, \vec{x}) \cap \mathcal{F}$
            make assignments to $\mathcal{W} \cup \mathcal{V}$
        end
        $\mathcal{Q} := \mathcal{Q} - \{(s, \vec{w})\}$
    end
end

Figure 7: A pixel color assignment refinement procedure.

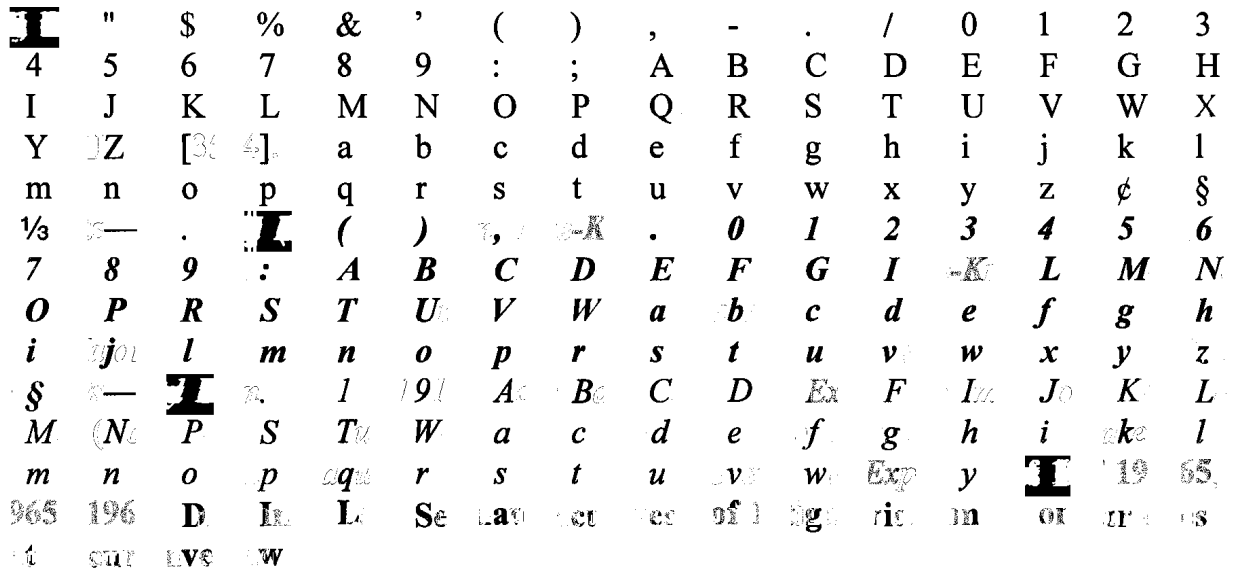


Figure 8: Templates derived from fig. 6 after four iterations of the refinement procedure.

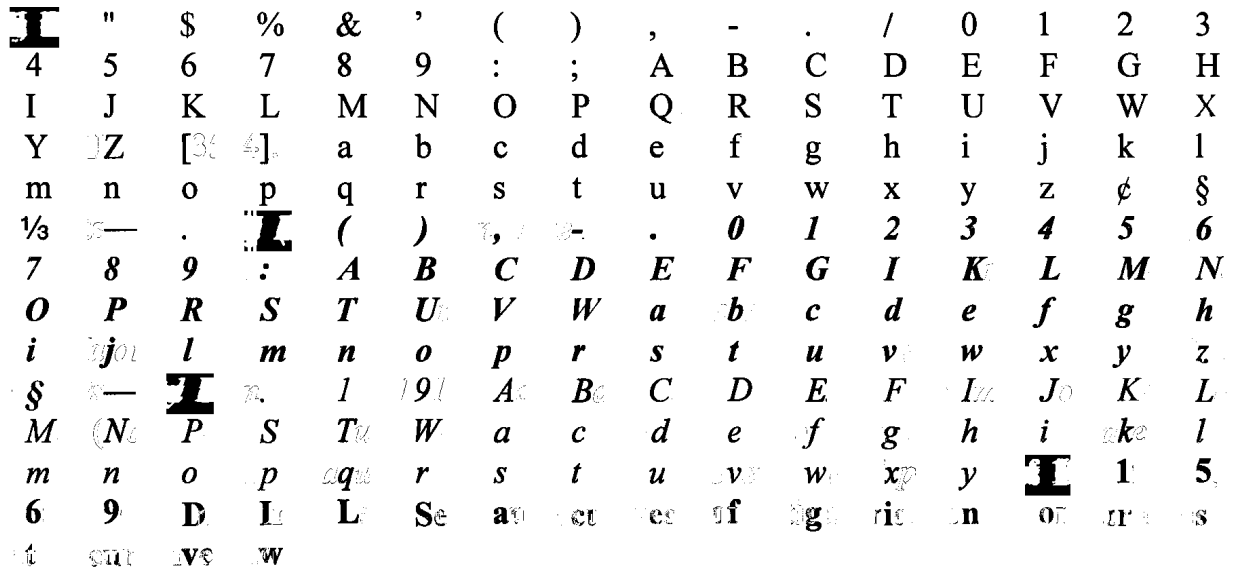


Figure 9: Templates derived from fig. 6 using the refinement procedure with a distance-based tie breaking criterion.

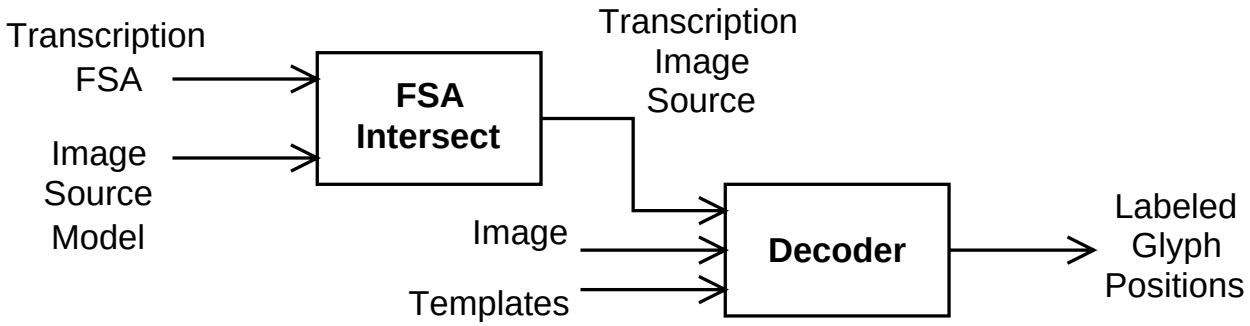
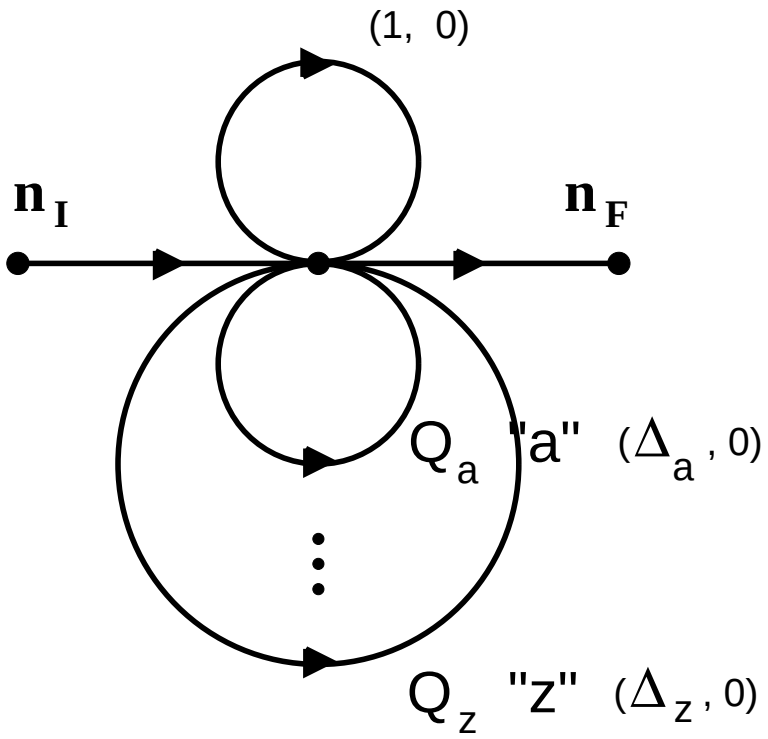
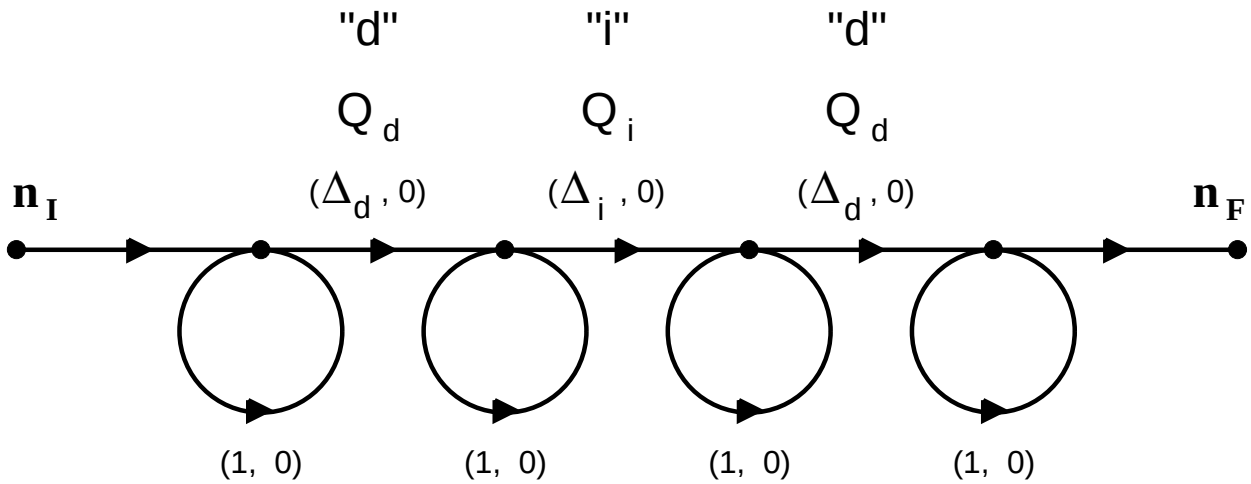


Figure 10: Transcription alignment by constrained decoding.

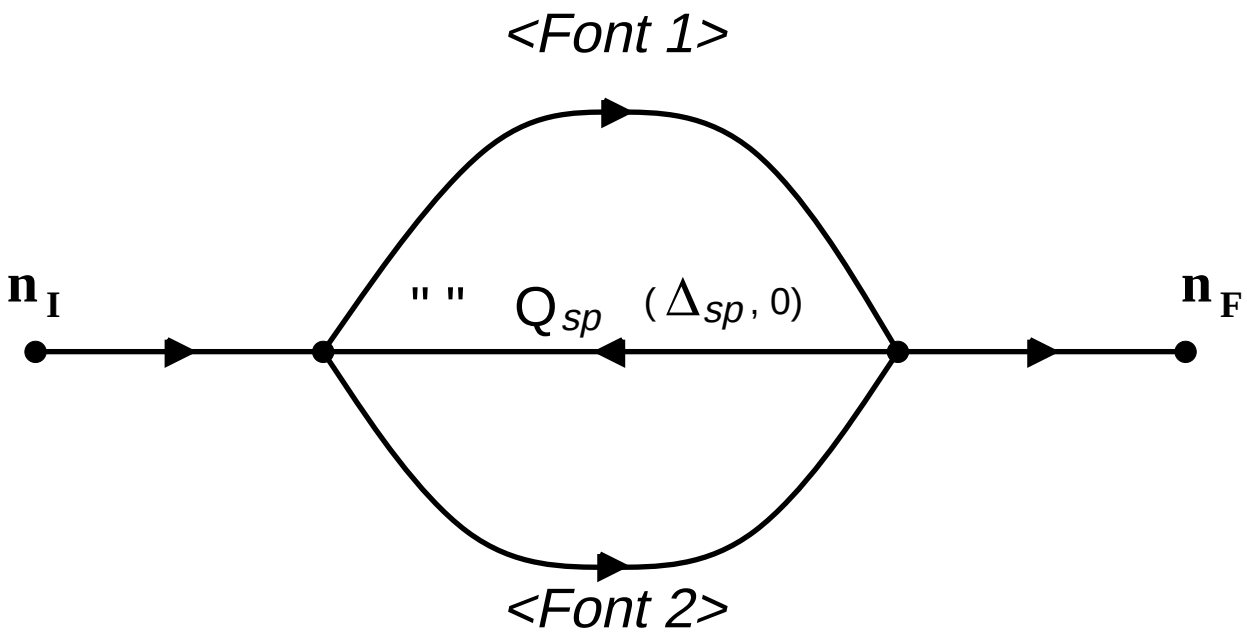


(a)

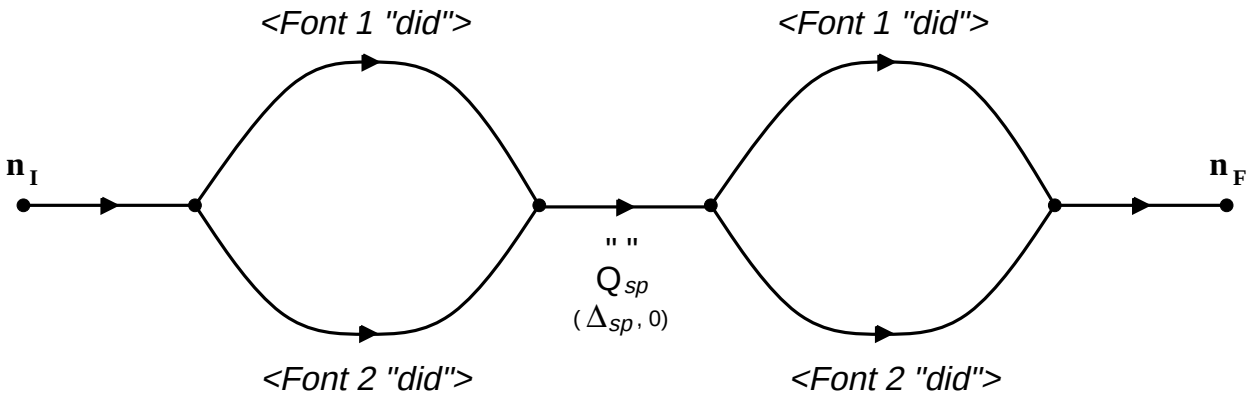


(b)

Figure 10: Simple alignment sources. (a) Text line image source. (b) Transcription image source for “did”.



(a)



(b)

Figure 11: Alignment sources with constrained font changes. (a) Text line image source. (b) Transcription image source for “did did”.

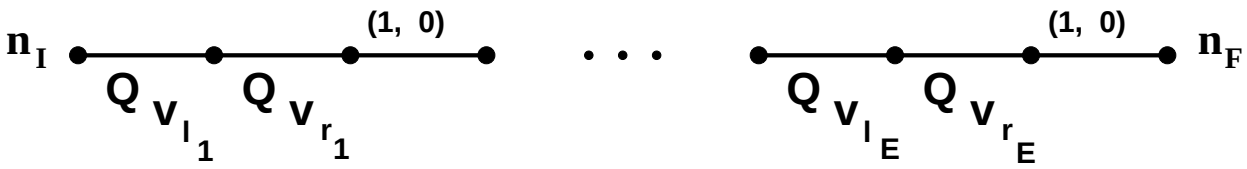


Figure 12: Image source for independent set problem.

ATE	Errors	% Error	#1	#2
Greedy	3478	0.52	l $\Rightarrow$ I (1077)	$\epsilon \Rightarrow$ s (191)
Refined	3404	0.51	l $\Rightarrow$ I (871)	$\epsilon \Rightarrow$. (208)
TextBridge	4978	0.74	$\epsilon \Rightarrow$ i (322)	u $\Rightarrow$ t (175)

(a)

ATE	Errors	% Error	#1	#2
Greedy	1746	0.29	$\epsilon \Rightarrow$ s (191)	$\epsilon \Rightarrow$ f (174)
Refined	1798	0.30	$\epsilon \Rightarrow$. (201)	$\epsilon \Rightarrow$ f (173)
TextBridge	4258	0.70	$\epsilon \Rightarrow$ i (318)	u $\Rightarrow$ t (175)

(b)

Table 1: Character recognition results for Univ. of Washington English Document Database II (simple text-body zones). (a) Results for complete 30 article test set. (b) Results for 28 articles (W6-Te and WA-Te excluded).

ATE	Errors	% Error	#1	#2
Greedy	2883	0.42	l $\Rightarrow$ I (1077)	$\epsilon \Rightarrow sp$ (150)
Refined	2705	0.40	l $\Rightarrow$ I (871)	$\epsilon \Rightarrow$ i (210)

(a)

ATE	Errors	% Error	#1	#2
Greedy	1101	0.18	l $\Rightarrow$ I (166)	$sp \Rightarrow \epsilon$ (135)
Refined	1099	0.18	$\epsilon \Rightarrow sp$ (142)	$sp \Rightarrow \epsilon$ (130)

(b)

Table 2: Character recognition results after font editing. (a) 30 articles. (b) 28 articles.

Spaces?	Errors	% Error	#1	#2
Y	5797	0.86	$\epsilon \Rightarrow sp$ (2484)	l $\Rightarrow$ I (986)
N	3047	0.53	l $\Rightarrow$ I (985)	i $\Rightarrow$ l (209)

(a)

Spaces?	Errors	% Error	#1	#2
Y	4349	0.72	$\epsilon \Rightarrow sp$ (2444)	l $\Rightarrow$ I (419)
N	1649	0.32	l $\Rightarrow$ I (419)	i $\Rightarrow$ l (102)

(b)

Table 3: Character recognition results after training with errorful transcriptions and font editing. (a) 30 articles. (b) 28 articles.