Character Template Estimation from Document Images and Their Transcriptions

by

Mauricio Lomelin Stoupignan

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degrees of

BACHELOR OF SCIENCE

and

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Author .....

Department of Electrical Engineering and Computer Science

May 19, 1995

Certified by .....

Sanjoy K. Mitter

Professor of Electrical Engineering, MIT LIDS

Thesis Supervisor

Certified by .....

Gary E. Kopec

Member of the Research Staff, Xerox Palo Alto Research Center

Thesis Supervisor

Accepted by .....

Frederic R. Morgenthaler

Chairman, Departmental Committee on Graduate Students

Character Template Estimation from Document Images and Their Transcriptions

by

Mauricio Lomelin Stoupignan

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 1995, in partial fulfillment of the
requirements for the degrees of
BACHELOR OF SCIENCE
and
MASTER OF SCIENCE

Abstract

The problem of character template estimation (CTE) refers to extracting proper models for the *shapes* of characters and a set of *font metrics* to dictate alignment between adjacent characters. This thesis develops a maximum likelihood approach to CTE for estimating bitmap templates within the framework of document image decoding (DID). An iterative procedure is developed for template estimation given a text image and a transcription of the text. The transcription and image need not be aligned nor is it necessary for the individual glyphs in the image to be segmentable prior to CTE. The proposed approach is demonstrated on text in a variety of fonts, including a scriptlike font in which adjacent characters are connected.

Thesis Supervisor: Sanjoy K. Mitter
Title: Professor of Electrical Engineering, MIT LIDS

Thesis Supervisor: Gary E. Kopec
Title: Member of the Research Staff, Xerox Palo Alto Research Center

Acknowledgments

I wish to express my deepest appreciation to my advisor Gary Kopec for being a wonderful person to work with. The guidance, constant encouragement, and freedom he gave me were invaluable. I found him to be a very admirable person.

I would also like to thank Dan Bloomberg for all the help he gave me in making the theoretical aspects of this thesis a practical reality. This research was done at the Xerox Palo Alto Research Center. Thanks go to all the researchers who made my internship there very stimulating.

Special thanks go to Angie Hinrichs for venturing with me to live in San Francisco while this research was being done. It was a wonderful experience. To all my friends at MIT, I want to thank you for sharing these past five years in all the good and bad times we've had.

I dedicate this thesis to my mother Cecilia. Words cannot express my gratitude to you for everything you have done for me. You have given me the freedom to choose my own path through life, and have offered me your love and support at every step of the way.

Contents

List of Figures

Chapter 1

Introduction

Character recognition is an area of pattern recognition that has received much attention with an increase in demand for automated processes that convert printed documents to electronic documents. Two major techniques used in character recognition are feature extraction and bitmap template matching. Feature extraction strategies start with a test pattern and measure a number of features that are known, in advance, to be good descriptors for the pattern. Template matching strategies compare the test pattern with stored reference patterns.

Template based modern Optical Character Recognition (OCR) systems face the problem of decoding the degraded image of a document accurately. These systems rely on knowledge of what each character looks like, and perform recognition through pattern matching across the image.

One problem with OCR systems of this type is that they are not capable of maintaining high OCR accuracy across a heterogeneous document collection. One key observation is that image quality is likely to be relatively uniform across the pages of a single document, reflecting constancy in the physical degradation process involved. The concern is that the character template models used by an OCR system are a reflection of what an ideal character looks like, and might not be well suited to recognize document images where each observed instance of a character has been similarly degraded. One proposed solution is to adapt the character templates by training them on a set of sample images and the corresponding transcriptions so that

the new templates are able to more accurately recognize similar images.

In other situations we sometimes find document images for which we do not have models for the shapes of the characters that comprise it, but for which we would like to recongize a large number of images. One solution to this problem is to manually draw each character model. This would be a very tedious enterprise, the models would have to be drawn to represent the observed image in some optimal sense (by hand this would probably be how well the character templates look), and would probably suffer from inaccuracies introduced by the artist. Another solution is to segment a sample image, label individual glyphs using the corresponding transcription of that image, and train the character templates as before. This is tedious and sometimes error prone on images having a lot of blur, or non-segmentable text like arabic or musical scores.

This thesis addresses the two related character template estimation (CTE) problems. In the first problem, suppose one is given a possibly non-perfect transcription, a degraded document image, and an alignment of the transcription to the image consisting of a set of glyph label, glyph position pairs. The problem consists of estimating the bitmap character templates that were used to create the image. This problem differs from the classical simple estimation problem of fitting a character template to a set of observed character images in that we do not have isolated characters. Rather we know only the location of each character origin, but do not know which of the pixels in the vicinity of the origin belong to the character and which belong to neighboring characters.

The second problem we address is a generalization of the first; we are now only given a document image, a transcription of each text line in the image and the vertical position of each text line baseline. Unlike the first problem, we are not given the horizontal positions of the characters in a line; the transcription gives an ordered sequence of observed characters but not their exact locations.

A desirable characteristic of any approach to character template estimation is that it be based on strong theoretical foundations. This will give us several advantages over *ad-hoc* (hand-tuned) procedures. First, we will be able to justify our procedures

on theoretical analyses rather than just on empirical results. Second, we will be able to extend the results to a general class of images that follow an assumed model of document generation. Third, we will be able to provide an explicit measure of optimality to the problem against which we can measure our success.

This research develops an approach to CTE within the framework developed by the document image decoding group (DID) at the Xerox Palo Alto Research Center [11]. The DID approach models document generation and decoding in terms of classical communication theory, where every stage of the process is explicitly modeled using detection and estimation theory, language formalisms, and adherence to statistical and probabilistic principles.

In the DID model, an ideal document image is generated from a set of ideal character templates by "painting" these templates on an image canvass in some logical arrangement. The document is then degraded through printing, handling, and scanning, which introduces distortions such as stains, skew, blurring, thinning, stretching, pen marks, and other similar alterations. Finally, the observed document is decoded by generating the most likely original clean image making explicit reference to the imaging procedure and noise model.

The rest of this thesis is organized as follows. Chapter 2 presents the theoretical framework of document image decoding and establishes the basic definitions and notation that will be used throughout this thesis. Chapter 3 formalizes the problem of character template estimation given labeled origins on a document image and shows that it is an NP-complete problem. This chapter also provides a suite of algorithms designed to efficiently estimate a set of templates, and evaluates their performance experimentally. Chapter 4 formalizes the problem of character template estimation given baselines and transcription only, presents an iterative approach based on the results of chapter 3 and related document recognition techniques, and evaluates the approach experimentally on different types of images. Finally, chapter 5 presents a summary and discusses future directions.

Chapter 2

Document Image Decoding Models

As an important background to this research, we first review the theoretical framework of document image decoding (DID) upon which this thesis is based [12, 11]. We begin with the basic communication problem provided to us by information theory, shown in figure 2-1. The DID framework casts the problem of document image recognition in terms of classical communication theory, where every stage of the process shown below is explicitly modeled using detection and estimation theory, language formalisms, and adherence to statistical and probabilistic principles.

In this view, a *message source* randomly produces a message X; an *encoder* deterministically encodes X into an ideal bitmapped image $\mathcal{I}'$; a *channel* randomly degrades the ideal image $\mathcal{I}'$ into an observed image $\mathcal{I}$; and a *decoder* deterministically maps $\mathcal{I}$ into a reconstructed message, or logical document structure, $\hat{X}$.

We discuss the the main elements of this framework – *source, encoder, channel, decoder* – as applied to images, in the following sections.

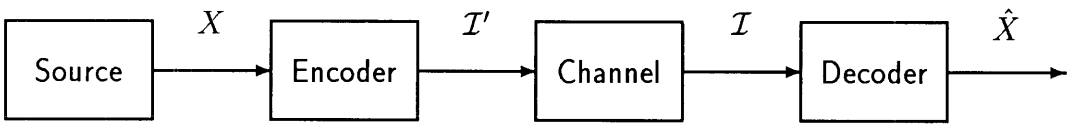


Figure 2-1: Communication System

2.1 Image Source Model

The typestyle is a visual property of the printed output; it is not part of the document's logical structure. For this reason, the source and the encoder must be regarded as separate modules; the source producing a message that someone is trying to communicate, and the encoder providing a model for putting that message into a bitmapped image that we see as a document.

In the framework described in [11], the message source and the encoder are merged into a single *image source* that simultaneously generates a message and a typographic structure. The model for the stochastic source and encoder is partly motivated by the *sidebearing model* of character shape and positioning [9]. In order to understand the stochastic source model and the encoder, we must first describe this document production model that is embedded within their descriptions.

2.1.1 Sidebearing Model

Character shape and positioning in digital typography is usually based on some variant of the *sidebearing model* [9] shown in figure 2-2. This model specifies glyph alignment parameters that specify the relative locations of the local coordinate systems of adjacent characters in a line of text.

A *character template* is a bitmap spanning the x-y plane with a distinguished local origin. An instance of a character on a page is known as a *character glyph.* The set of black pixel positions of a template is called the *support* of that template.

A set of parameters that describe the relative placement of glyphs with respect to one another are called *font metrics.* In this thesis, we will be primarly concerned with font metrics that determine horizontal letterspacing.

The *character coordinate system* is the space in which an individual character shape is defined. All metrics are interpreted in the character coordinate space. The *origin* of the character is the point (0,0) in the character coordinate system. The *setwidth* of a character Q is the vector displacement, $\vec{\Delta}_Q = (\Delta x, \Delta y)$, from the local origin of that character to the point at which the origin of the next character is nor-

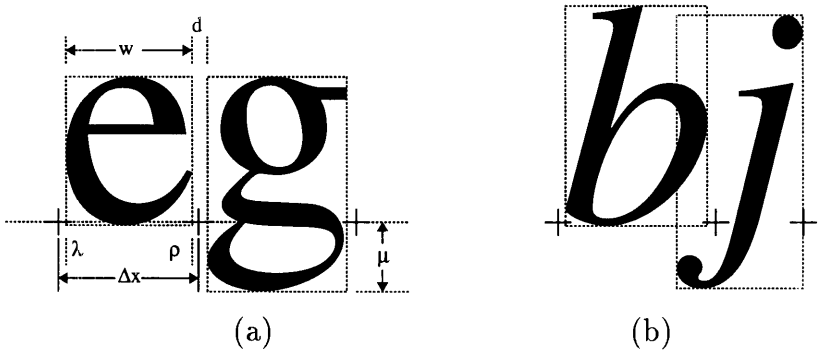


Figure 2-2: Sidebearing model for character placement. Character origins are indicated by crosses. (a) Character spacing and alignment parameters. (b) Example of negative sidebearings. The character bounding boxes overlap, but the character supports do not.

mally placed when imaging consecutive characters. Most Indo-European alphabets, including Roman, have $\Delta x > 0$ and $\Delta y = 0$; other alphabets have $\Delta x < 0$, such as Semitic, or non-zero Δy, such as Oriental.

The *bounding box* of a character is the smallest rectangle, oriented with the character coordinate axes, that will just enclose the character's shape. The *width* of a character is the corresponding width w of the bounding box. The *left sidebearing* is the horizontal displacement λ from the origin of the character to the left edge of the bounding box. Similarly, the *right sidebearing* is the horizontal displacement ρ from the right edge of the bounding box to the origin of the next character. The *depth below baseline* is the vertical distance μ from the character origin to the bottom of the character bounding box.

The horizontal component of the setwidth is related to the sidebearings and the bounding box width by the relation

$$\Delta x = \lambda + w + \rho. \tag{2.1}$$

The intercharacter spacing term d is related to sidebearings by

$$d = \rho_l + \lambda_r \tag{2.2}$$

where the subscripts l and r indicate the left and right members of a character pair,

(a) (b)

Figure 2-3: Greekings of Adobe Times-Italic. The origin and setwidth of the "j" are indicated by crosses. The gray region is the superposition of all characters from the font. (a) Right Greeking; each character is right-aligned with the origin. (b) Left Greeking; each character is left-aligned with the origin.

respectively.

Why use the sidebearing model? It provides great flexibility and can be generalized to a wide range of images. Consider a font for music notation. Musical symbols are not arranged one after the other as in text. However, by correctly specifying the font metrics for each glyph, as explained in [14], the sidebearing model is useful for describing images of music.

The design of typefaces implicitly carries with it a characteristic that will turn out to be a very important one for character template estimation, the notion of non-overlapping template supports. The non-overlapping support observation can be formalized as follows (we assume that all character images are from the same font).

Let $Y \in \mathcal{Y}$ be a character template drawn from a set of templates $\mathcal{Y}$. Let $Y[\vec{x}_0]$ denote Y shifted from an arbitrary origin (0,0) so that its origin is located at $\vec{x}_0$. Define

$$G_R = \bigcup_{Y \in \mathcal{Y}} Y[-\vec{\Delta}_Y] \tag{2.3}$$

to be the *right greeking* of $\mathcal{Y}$. Recall that the setwidth of a character Q, denoted by $\vec{\Delta}_Q = (\Delta x,\ \Delta y)$ is the vector displacement from the character origin to the point at which the origin of the next character is normally placed when imaging consecutive characters of a word. Loosely then, G_R is the superposition of font characters aligned

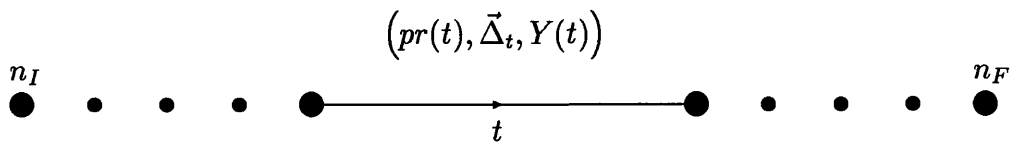


Figure 2-4: A Simple Stochastic Image Source

on the right. We can similarly define the *left greeking* of $\mathcal{Y}$ as

$$G_L = \bigcup_{Y \in \mathcal{Y}} Y[\vec{0}]. \tag{2.4}$$

The observation about non-overlapping support of characters may then be formalized by the following two conditions

$$Y \cap G_R = \emptyset, \tag{2.5}$$

$$Y[-\vec{\Delta}_Y] \cap G_L = \emptyset \tag{2.6}$$

for each $Y \in \mathcal{Y}$. This is illustrated in figure 2-3.

Although this thesis will deal mostly with documents generated using conventional letterform typography, there are extensions of this research to the analysis of images such as musical scores. Our initial approach will show that enforcing the templates to have non-overlapping supports produces visually pleasing results.

2.1.2 Stochastic Image Source

The stochastic image source is modeled by a Markov source, as shown in figure 2-4. A Markov source is a Markov chain with the additional characteristic of transitional attributes; when described as a directed graph, every transition arc t of the Markov chain is labeled with a triple,[1] $(pr(t), \vec{\Delta}_t, Y(t))$, where $pr(t)$ is the transition probability, $Y(t)$ is a template, and $\vec{\Delta}_t$ is a displacement vector. The Markov chain has a unique initial state n_I, and a final trap state n_F.

The displacement vector of a transition is analogous to the setwidth of the tem-

[1]This is a simplified version of the Markov source presented in [11]

plate associated with that transition. In our formulation, the displacement vectors associated with every transition are assumed to satisfy the requirement that the supports of two consecutively imaged templates do not overlap with each other. We will also impose the general requirement that the support of *any* imaged template does not overlap with the support of any other template on that image.

As the chain evolves according to its transition probabilities, a path π through the Markov source is described as the sequence of transitions the chain takes: $\pi = t_1 \ldots t_P$, where t_1 is a transition out of the initial state n_I. A *complete path* is a path for which t_P is a transition into the final trap state n_F.

A complete path $\pi = t_1 \ldots t_P$ through the Markov image source defines an associated document image as follows. Imagine an image automaton that begins in an initial position on the image canvass which we will take to be the upper left hand corner of that canvass.[2] With each transition t_i, the imager takes the corresponding template $Y(t_i)$, positions the origin of the template at its current position and paints the image of that template on the canvass. It then updates its location on the canvass by the vector $\vec{\Delta}_{t_i}$, and waits for the next transition until the Markov chain enters the final state n_F, and the image has been generated. It must be noted that the template $Y(t_i)$ can be a null template, such as a carriage return or small transitions to adjust intercharacter spacing, in which case the layout information is entirely carried by $\vec{\Delta}_{t_i}$.

The above imaging procedure can be interpreted as forming a union of templates across the canvass at given locations defined by the sequence of displacement vectors $\vec{\Delta}_{t_i}$; in other words, through the evolution of the Markov chain, the imager "paints" template $Y(t_i)$ on the canvass at position $\sum_{j=0}^{i-1} \vec{\Delta}_{t_j}$. We define the sequence of positions $\vec{x}_1 \ldots \vec{x}_{P+1}$ associated with each path π by

$$\vec{x}_1 = \vec{0} \tag{2.7}$$

$$\vec{x}_{i+1} = \vec{x}_i + \vec{\Delta}_{t_i}. \tag{2.8}$$

[2]This particular coordinate system uses this corner as origin, and defines positive x to the right, and positive y down.

If we define the function $f : t_i \rightarrow \{1, 2, \ldots, T\}$ that maps[3] transition t_i to a template in the set of all possible imaging templates $\mathcal{Y} = \left\{Y^{(1)}, Y^{(2)}, \ldots, Y^{(T)}\right\}$, such that $Y(t_i) = Y^{(f(t_i))}$, then the imager forms the associated document image $\mathcal{I}'$ as

$$\mathcal{I}' = \bigcup_{i=1}^{P} Y^{(f(t_i))}[\vec{x}_i]. \tag{2.9}$$

The transition probabilities $pr(t)$ of the Markov source give us a probability distribution on complete paths by

$$\Pr(\pi) = \prod_{i=1}^{P} pr(t_i). \tag{2.10}$$

Under the assumption that every path produces a unique image,[4] then $\Pr(\mathcal{I}') = \Pr(\pi)$.

We must point out that for a given transition t, the transition probability $pr(t)$ of the Markov source *cannot* be interpreted as the independent probability of the corresponding template $Y^{(f(t))}$. In a given Markov chain, two distinct transitions with different transition probabilities might have the same template associated with them. For this model, the probability distribution of clean images is a function of the possible paths through the Markov source, and *not* of the imaged templates.

2.2 Channel Model

The channel introduces distortions into a document through printing, photo-copying, handling and scanning. These distortions appear to us as random noise, pen marks, stains, skew, rotations, blurring, and other similar perturbations. A noisy channel is a random system that establishes a statistical relationship between its input and its output; it's output is not completely determined by the input. Although realistic document image defect models can be very complicated, an independent, asymmetric bit-flip noise channel shown in figure 2-5 is proposed in [11] for simplicity. This choice yields very nice analytical and experimental results.

[3] $f(\cdot)$ assigns to each transition a label corresponding to a template in $\mathcal{Y}$

[4] [11] deals with the more general case of more than one message path defining any given image

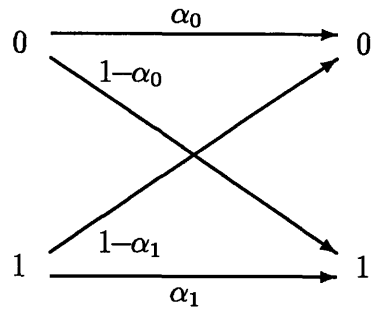


Figure 2-5: Asymmetric Bit-Flip Noise Model

In the ideal binary image $\mathcal{I}' = \{y_i \mid i \in \Omega\}$, every pixel is indexed by Ω. We will assume that images are rectangular so that Ω is the integer lattice $[0, W) \times [0, H)$, where W and H are the image width and height, respectively. The channel maps the ideal image into an observed image $\mathcal{I} = \{z_i \mid i \in \Omega\}$ by introducing distortions.

The asymmetric bit-flip channel is completely characterized by two numbers, α_1 and α_0, which represent the probability that each black (white) pixel in an ideal image y_i gets mapped to a black (white) pixel in the observed image z_i. These parameters are assumed to be constant over the image.

With this model, the probability of an observed image $\mathcal{I}$ given an ideal clean image $\mathcal{I}'$ is simply

$$\Pr(\mathcal{I}|\mathcal{I}') = \prod_{i \in \Omega} \left[\alpha_0^{(1-z_i)}(1-\alpha_0)^{z_i}\right]^{(1-y_i)} \left[\alpha_1^{z_i}(1-\alpha_1)^{(1-z_i)}\right]^{y_i}. \tag{2.11}$$

If we let $\mathcal{I}_0$ be the all-white background image, then

$$\Pr(\mathcal{I}|\mathcal{I}_0) = \prod_{i \in \Omega} \alpha_0^{(1-z_i)}(1-\alpha_0)^{z_i} \tag{2.12}$$

and we can normalize $\Pr(\mathcal{I}|\mathcal{I}')$ into a likelihood ratio form that will give us several advantages and be easier to manipulate

$$\tilde{\Pr}(\mathcal{I}|\mathcal{I}') \equiv \frac{\Pr(\mathcal{I}|\mathcal{I}')}{\Pr(\mathcal{I}|\mathcal{I}_0)} = \prod_{i \in \Omega} \left(\frac{1-\alpha_1}{\alpha_0}\right)^{y_i} \left(\frac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right)^{y_i z_i}. \tag{2.13}$$

We define the log normalized probability as

$$\begin{aligned}\mathcal{L}(\mathcal{I}|\mathcal{I}') &\equiv \log \tilde{\Pr}(\mathcal{I}|\mathcal{I}') && (2.14)\\ &= \log\left(\tfrac{1-\alpha_1}{\alpha_0}\right)\sum_{i\in\Omega} y_i + \log\left(\tfrac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right)\sum_{i\in\Omega} y_i z_i && (2.15)\\ &= \|\mathcal{I}'\|\log\left(\tfrac{1-\alpha_1}{\alpha_0}\right) + \|\mathcal{I}' \wedge \mathcal{I}\|\log\left(\tfrac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right) && (2.16)\end{aligned}$$

where $\|\mathcal{I}'\|$ gives the number of black pixels in $\mathcal{I}'$, and $\mathcal{I}' \wedge \mathcal{I}$ denotes the bitwise logical AND of images $\mathcal{I}'$ and $\mathcal{I}$. This log normalized probability has a very important decomposition property. If an image Y is generated as a union of templates whose support does not overlap as in (2.9), then it is easy to see that

$$\mathcal{L}(\mathcal{I}|\mathcal{I}') = \sum_{i=1}^{P} \mathcal{L}(\mathcal{I}|Y^{(f(t_i))}[\vec{x}_i]). \tag{2.17}$$

The quantity $\mathcal{L}(\mathcal{I}|\mathcal{I}')$ is the lognormalized probability that describes the degree of matching between an observed image $\mathcal{I}$ and a template constructed image $\mathcal{I}'$. In image decoding, $\mathcal{L}(\mathcal{I}|\mathcal{I}')$ gives a matching score for any image $\mathcal{I}'$ we consider to be the original clean one that generated the observed image $\mathcal{I}$ through the noise channel. Equation (2.17) can be interpreted as the aggregate likelihood of every independent template that makes up the ideal image across the observed image.

We define $\mathcal{L}(\cdot)$ as the likelihood function of its argument. This notation will be used extensively to refer to likelihood, and its use will be aparent from context.

2.3 Decoder Model

The decoder takes as input a noisy version of the original image and tries to invert the effects of the encoder and noisy channel. The decoder can then output a reconstructed image, a transcription of the document image, or some other structural information about the document. The decoder must, therefore, incorporate within itself knowledge of the source and channel models.

Due to the stochastic nature of the channel model, there is more than one input

image that could account for the observed output image. There is not a one-to-one correspondence between every input and every output. Instead, there exists a probability distribution between an observed noisy image and every possible clean image that could have been constructed using the source model. For example, although an all-white image could have generated an observed document through the noise channel, this is a highly improbable event. Intuitively, the image that has the highest number of matching symbols will be the one that generated the document.[5] We formalize our intuition by reverting to decision theory.

Given the finite set of templates and the document production rules embedded in the source model, there is only a finite sample space of clean images that the source model can generate on the image canvass, $\mathcal{I}'_0, \mathcal{I}'_1, \ldots, \mathcal{I}'_n$. Furthermore, given the complete description of the Markov source model, we can also associate an *a-priori* probability to each sample in the space of clean images, $\Pr(\mathcal{I}'_i)$, as shown in section 2.1.

Now, given the noise model and all possible clean images, we can associate a probability to the event that the observed noisy image was generated from a given clean image, $\Pr(\mathcal{I}|\mathcal{I}'_i)$, as shown in section 2.2.

All the decoder has to do now is to decide on the "best" clean image. Let us generalize and assume that we have a set of costs, C_{ij}, of deciding $\mathcal{I}'_i$ as the original image when the true original image is $\mathcal{I}'_j$. We must now specify a decision rule for the decoder to choose one of the n clean images based on the observed image $\mathcal{I}$, such that the expected cost of choosing the wrong image is minimized. This is exactly the *M-ary* hypothesis testing problem, where we have:

- M hypotheses $\mathcal{I}'_0, \mathcal{I}'_1, \ldots, \mathcal{I}'_M$ with a-priori probabilities $\Pr(\mathcal{I}'_i)$.
- A set of costs, C_{ij}, of deciding $\mathcal{I}'_i$ when $\mathcal{I}'_j$ is true.
- The set of distributions $\Pr(\mathcal{I}|\mathcal{I}'_i)$ of our observation given that each of the hypotheses is correct.

[5] This intuition is partly correct, as we will see, it ignores the *a-priori* probability that a given document was generated by the source model.

If the set of costs have the form $C_{ij} = 1 - \delta_{ij}$, then it is well-known [21] that the optimal decision rule is the maximum a-posteriori (MAP) rule,

$$\begin{aligned}
\hat{\mathcal{I}}' &= \operatorname*{argmax}_{\mathcal{I}'} \Pr(\mathcal{I}'|\mathcal{I}) && (2.18)\\
&= \operatorname*{argmax}_{\mathcal{I}'} \frac{\Pr(\mathcal{I}|\mathcal{I}')\Pr(\mathcal{I}')}{\Pr(\mathcal{I})} && (2.19)\\
&= \operatorname*{argmax}_{\mathcal{I}'} \Pr(\mathcal{I}|\mathcal{I}')\Pr(\mathcal{I}') && (2.20)
\end{aligned}$$

We can normalize $\Pr(\mathcal{I}|\mathcal{I}')$ above by $\Pr(\mathcal{I}|\mathcal{I}_0)$ since it does not depend on $\mathcal{I}'$, and from the monotonicity of the logarithm, we can also maximize the logarithm of the resulting function,

$$\begin{aligned}
\hat{\mathcal{I}}' &= \operatorname*{argmax}_{\mathcal{I}'} \log \frac{\Pr(\mathcal{I}|\mathcal{I}')\Pr(\mathcal{I}')}{\Pr(\mathcal{I}|\mathcal{I}_0)} && (2.21)\\
&= \operatorname*{argmax}_{\mathcal{I}'} \left[\log \tilde{\Pr}(\mathcal{I}|\mathcal{I}') + \log \Pr(\mathcal{I}')\right] && (2.22)\\
&= \operatorname*{argmax}_{\mathcal{I}'} \left[\mathcal{L}(\mathcal{I}|\mathcal{I}') + \log \Pr(\mathcal{I}')\right] && (2.23)\\
&= \operatorname*{argmax}_{\mathcal{I}'} \left\{\sum_{i=1}^{P} \left[\mathcal{L}(\mathcal{I}|Y^{(f(t_i))}[\vec{x}_i]) + \log pr(t_i)\right]\right\} && (2.24)
\end{aligned}$$

where the last equality used the results from section 2.2. Under the assumption that every clean image $\mathcal{I}'$ is produced by a unique path π through the image source, we can equivalently talk about the path through the source that optimally represents the observed image in the MAP sense.

$$\begin{aligned}
\hat{\pi} &= \operatorname*{argmax}_{\pi} \mathcal{L}(\mathcal{I}, \pi) && (2.25)\\
&= \operatorname*{argmax}_{\pi} \left[\mathcal{L}(\mathcal{I}|\mathcal{I}') + \log \Pr(\pi)\right] && (2.26)\\
&= \operatorname*{argmax}_{\pi} \left\{\sum_{i=1}^{P} \left[\mathcal{L}(\mathcal{I}|Y^{(f(t_i))}[\vec{x}_i]) + \log pr(t_i)\right]\right\}. && (2.27)
\end{aligned}$$

Both approaches can be interpreted as a maximization of the likelihood of the observed image given a clean image and the a-priori probability that this clean image came from the markov source. MAP decoding tries to find a clean template reconstructed image that most probably generated the observed image through the noise channel,

so that in the MAP sense, this template reconstructed image is the original image.

MAP decoding of an observed image $\mathcal{I}$ with respect to a given image source involves finding a complete path π through that source which maximizes $\mathcal{L}(\mathcal{I}, \pi)$ over all complete paths through the Markov source. In [11] it is shown that MAP decoding can be reduced to solving the following recurrence relation

$$\mathcal{L}(n, \vec{x}) = \max_{t|t:v\to n} \left\{ \mathcal{L}(v, \vec{x} - \vec{\Delta}_t) + \mathcal{L}(\mathcal{I}|Y^{(f(t))}[\vec{x} - \vec{\Delta}_t]) + \log pr(t) \right\} \tag{2.28}$$

where n and v are variables representing states in the Markov source and t represents a transition into state n. The maximization in (2.28) can be computed using a dynamic programming algorithm called a *Viterbi algorithm.*

The recurrence term $\mathcal{L}(n, \vec{x})$ gives the best likelihood score if a path through the source had gone through state n at a position in the image plane $\vec{x}$ during the generation of the clean image. The main recurrence arises from the fact that to go through $(n, \vec{x})$ while generating the image, the previous stage must have been $(v, \vec{x} - \vec{\Delta}_t)$, for some transition t going from v to n. The first term in the sum, $\mathcal{L}(v, \vec{x} - \vec{\Delta}_t)$ accounts for the a-posteriori probability of reaching the previous stage $(v, \vec{x} - \vec{\Delta}_t)$, the second term $\mathcal{L}(\mathcal{I}|Y^{(f(t))}[\vec{x} - \vec{\Delta}_t])$ gives a matching score for the template $Y^{(f(t))}$ at location $(\vec{x} - \vec{\Delta}_t)$ on the observed image, and the last term $\log pr(t)$ accounts for the a-priori probability of continuing this path by taking transition t. The maximization in this recursion gives the best path through the source of reaching $(n, \vec{x})$ from among all the possible previous states.

By performing this maximization for each node n and image position x, the result is a complete path through the source that maximizes the matching score between all of the imaged templates through that path, $\sum_{i=1}^{P} \mathcal{L}(\mathcal{I}|Y^{(f(t))}[\vec{x}_i]) = \mathcal{L}(\mathcal{I}|\mathcal{I}')$, and the a-priori probability of that path, $\sum_{i=1}^{P} \log pr(t_i) = \log \Pr(\pi)$.

2.4 Background Definitions

In this section we give a few basic definitions that will help establish consistent notation throughout the rest of this thesis.

$Y^{(i)}$ will denote the ith labeled character template. This can be a normal printing character, a space character, or a null template.

$\mathcal{Y} = \left\{Y^{(1)}, Y^{(2)}, \ldots, Y^{(T)}\right\}$ will denote the set of all character templates used to generate the observed image.

$\mathcal{I}$ will be the observed image.

π will be a complete path through an image source.

$\mathcal{I}' = \bigcup_{i=1}^{P} Y^{(f(t_i))}[\vec{x}_i]$ will denote the template reconstructed image given the path through a stochastic image source and the set of imaged characters $\mathcal{Y}$.

The operator $\|\mathcal{Q}\|$ gives the number of black pixels in $\mathcal{Q}$. The argument $\mathcal{Q}$ can be an image or a vector of image pixels.

The operator $|\mathcal{Q}|$ gives the total number of pixels in an image $\mathcal{Q}$, or the number of elements in a set $\mathcal{Q}$. If $\mathcal{Q}$ is a collection of sets, the operator gives the number of sets in $\mathcal{Q}$.

$\mathcal{X} \wedge \mathcal{W}$ denotes the bitwise logical AND of images $\mathcal{X}$ and $\mathcal{W}$.

$(\neg\mathcal{Q})$ denotes the inverse of an image $\mathcal{Q}$, i.e. bitwise complement.

Z_j will denote the j^{th} observed instance of a character. This image may or may not be that of an isolated character.

$\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_N\}$ denotes the set of observed instances of a given character.

$\mathcal{L}(\cdot)$ is a likelihood function. Its specific definition will be aparent from context.

The assignment operator $\mathcal{A} \leftarrow \mathcal{B}$ evaluates $\mathcal{B}$ first and assigns the result to $\mathcal{A}$.

Chapter 3

Character Template Estimation with Origins Known

Using the models and notation reviewed in the previous chapter, we pose the first problem. Given a complete path $\pi_0 = t_1, \ldots, t_P$, a mapping function $f : t_i \rightarrow \{1, \ldots, T\}$, the set of displacement vectors $\vec{\Delta}_{t_i}$, and the observed document image $\mathcal{I}$ generated by this path, the question is how to determine the set of templates $\mathcal{Y} = \left\{Y^{(1)}, Y^{(2)}, \ldots, Y^{(T)}\right\}$ that optimally represent the observed image $\mathcal{I}$. This problem is equivalent to saying that we are given the origin of every glyph on the document image, a label on each origin, and we are trying to determine the image of the templates corresponding to each label.

Using the results of section 2.3, this quantitatively translates to finding the set of templates $\mathcal{Y}$ that maximizes the likelihood of the observed image given the template reconstructed image,

$$\underset{\mathcal{I}'}{\operatorname{argmax}}\, \mathcal{L}(\mathcal{I}|\mathcal{I}') = \underset{\mathcal{I}'}{\operatorname{argmax}} \sum_{i=1}^{P} \mathcal{L}(\mathcal{I}|Y^{(f(t_i))}[\vec{x}_i]). \tag{3.1}$$

This problem would be very easy if we were given the isolated observed instances of every given character. The variable to be estimated would be the original character template, Y, and the measurements of this template would be all the isolated observed instances of that character, $\mathcal{Z}$. Estimation theory would now dictate that the best

estimate of that template is given to us by the Maximum Likelihood estimator $\hat{Y} = \max_Y \Pr\{\mathcal{Z} \mid Y\}$.

This "best" estimate of each individual template also maximizes $\mathcal{L}(\mathcal{I}|\mathcal{I}')$. Simply note that the terms in the sum on equation (3.1) can be re-grouped into classes corresponding to the likelihood of a given template given all of its observed images. Maximizing each one of these classes individually maximizes the sum of the classes, and consequently $\mathcal{L}(\mathcal{I}|\mathcal{I}')$.

Unfortunately, given a document image with character origins, there is no information about the locality of the character glyphs with respect to this origin. In addition, any local region about this origin may contain pixels belonging to the desired character **and** extraneous pixels belonging to other glyphs in the vicinity of the origin. This problem differs from the classical simple estimation problem of fitting a character template to a set of observed character images in that we do not have isolated characters. Rather we know only the location of each character origin, but do not know which of the pixels in the vicinity of the origin belong to the character and which belong to neighboring characters.

We will tackle this problem as follows. We will obtain the observed instances of every character by extracting a fixed size region of image about each of the labeled character origins on the image. We will treat this set of character images as a set of isolated characters and perform the optimization described above. The resulting set of templates will contain a lot of noise, mainly because the observed instances for each character are not isolated characters. We will then proceed to "clean" these templates by enforcing the disjoint template support requirement.

3.1 Independent Maximum Likelihood Template Estimation

In Maximum Likelihood Estimation, we assume that the variable to be estimated is unknown. Furthermore, we assume that we have a probabilistic model of the relation-

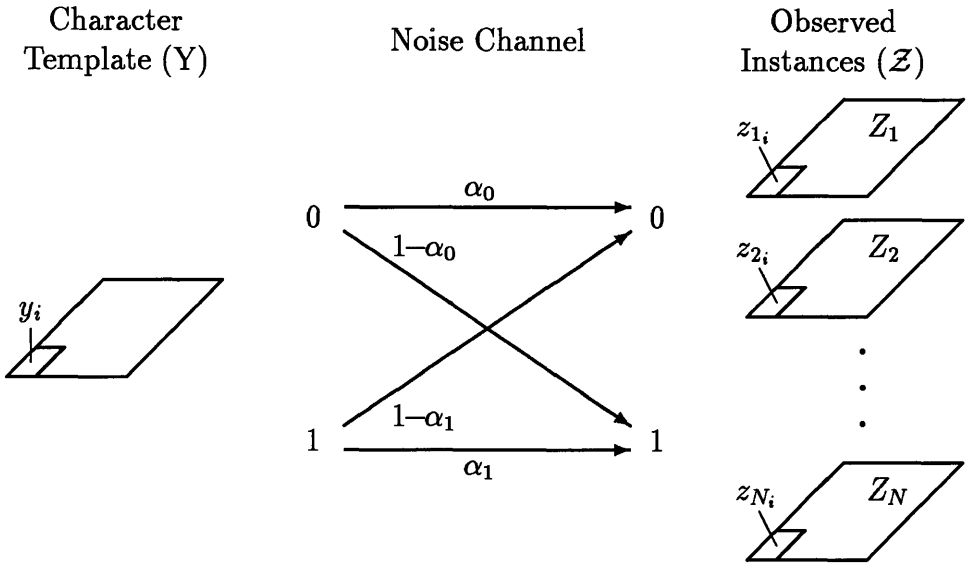


Figure 3-1: Maximum Likelihood Template Estimation

ship between the variable to be estimated, Y, and its measurements, $\mathcal{Z}$. Maximum Likelihood then tells us to take as an estimate of Y the value that makes its actually observed values of $\mathcal{Z}$ as likely as possible. That is,

$$\hat{Y} = \max_{Y} \Pr\{\mathcal{Z}|Y\}. \tag{3.2}$$

Figure 3.1 illustrates the ML template estimation scenario. The observed instances of a character, $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_N\}$, are obtained by extracting a fixed size image region about each of the labeled origins of that character. We will assume the extracted glyphs are rectangular with width W_o and height H_o so that $\mathcal{G}$ is the integer lattice $[0, W_o) \times [0, H_o)$, and in each binary image, $Z_j = \{z_{j_i} \mid i \in \mathcal{G}\}$, every pixel is indexed by $\mathcal{G}$.

Let us denote the ith indexed pixel of template Y by y_i. Through the noise channel, this pixel maps to the ith indexed pixel in every observed instance. We will define the observed data vector of y_i as $\mathcal{Z}_i = \{z_{1_i}, z_{2_i}, \ldots, z_{N_i}\}$. The probability of $\mathcal{Z}_i$

given the source pixel y_i and the noise model parameters α_0 and α_1 is

$$\Pr\{\mathcal{Z}_i \mid y_i, \alpha_0, \alpha_1\} = \prod_{j=1}^{N} \left[\alpha_0^{(1-z_{j_i})}(1-\alpha_0)^{z_{j_i}}\right]^{(1-y_i)} \left[\alpha_1^{z_{j_i}}(1-\alpha_1)^{(1-z_{j_i})}\right]^{y_i}. \quad (3.3)$$

Assuming pixels are independent of each other, we can write the probability of the observed images given their template and the channel parameters as

$$\Pr\{\mathcal{Z} \mid Y, \alpha_0, \alpha_1\} = \prod_{i \in \mathcal{G}} \Pr\{\mathcal{Z}_i \mid y_i, \alpha_0, \alpha_1\} \quad (3.4)$$

$$= \prod_{i \in \mathcal{G}} \prod_{j=1}^{N} \left[\alpha_0^{(1-z_{j_i})}(1-\alpha_0)^{z_{j_i}}\right]^{(1-y_i)} \left[\alpha_1^{z_{j_i}}(1-\alpha_1)^{(1-z_{j_i})}\right]^{y_i} \quad (3.5)$$

Substituting (3.5) into (3.2) we write the Maximum Likelihood decision rule as

$$\hat{Y} = \max_{Y,\alpha_0,\alpha_1} \prod_{i \in \mathcal{G}} \prod_{j=1}^{N} \left[\alpha_0^{(1-z_{j_i})}(1-\alpha_0)^{z_{j_i}}\right]^{(1-y_i)} \left[\alpha_1^{z_{j_i}}(1-\alpha_1)^{(1-z_{j_i})}\right]^{y_i}. \quad (3.6)$$

This maximization cannot be performed analytically, but can be done iteratively via a gradient search. Specifically, starting with arbitrary values for α_0 and α_1, we obtain a new template Y' by maximizing (3.6) for Y using fixed values of α_0 and α_1; then we update the channel parameters by maximizing (3.6) for α_0 and α_1 using Y'. This procedure is repeated until we converge to some local maximum. The following two sections derive the maximizing values for Y, α_0, and α_1 at every iteration, and derive a test to assess convergence of this procedure.

3.1.1 Optimize $\mathcal{Y}$ with α_0 and α_1 fixed

Given the observed set of images $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_N\}$ of a template Y, our objective is to decide whether to assign a value of 0 or 1 to every pixel y_i in Y. Since $y_i \in \{0, 1\}$, we have a binary hypothesis testing problem and the ML decision rule is

$$\Pr\{\mathcal{Z}_i \mid y_i = 0, \alpha_0, \alpha_1\} \underset{y_i=1}{\overset{y_i=0}{\gtrless}} \Pr\{\mathcal{Z}_i \mid y_i = 1, \alpha_0, \alpha_1\}. \quad (3.7)$$

Setting $y_i = 0$ and $y_i = 1$ in (3.3) we have

$$\prod_{j=1}^{N} \alpha_0^{(1-z_{j_i})}(1-\alpha_0)^{z_{j_i}} \underset{y_i=1}{\overset{y_i=0}{\gtrless}} \prod_{j=1}^{N} \alpha_1^{z_{j_i}}(1-\alpha_1)^{(1-z_{j_i})} \tag{3.8}$$

$$\prod_{j=1}^{N} \left(\frac{\alpha_0}{1-\alpha_1}\right)\left(\frac{(1-\alpha_0)(1-\alpha_1)}{\alpha_0\alpha_1}\right)^{z_{j_i}} \underset{y_i=1}{\overset{y_i=0}{\gtrless}} 1. \tag{3.9}$$

Taking logarithms and noting that $\sum_{j=1}^{N} z_{j_i} = \|\mathcal{Z}_i\|$, we rewrite (3.9) as

$$N\log\left(\frac{\alpha_0}{1-\alpha_1}\right) + \|\mathcal{Z}_i\|\log\left(\frac{(1-\alpha_0)(1-\alpha_1)}{\alpha_0\alpha_1}\right) \underset{y_i=1}{\overset{y_i=0}{\gtrless}} 0. \tag{3.10}$$

We must be careful of the values of each of the log terms. There are two regions of values for the α's to consider:

- $\alpha_1 > 1 - \alpha_0$.

  In this case, $\log\left(\frac{\alpha_0}{1-\alpha_1}\right) > 0$ and $\log\left(\frac{(1-\alpha_0)(1-\alpha_1)}{\alpha_0\alpha_1}\right) < 0$. We rewrite the ML decision rule as follows

$$-\|\mathcal{Z}_i\|\log\left(\frac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right) \underset{y_i=1}{\overset{y_i=0}{\gtrless}} -N\log\left(\frac{\alpha_0}{1-\alpha_1}\right) \tag{3.11}$$

$$\frac{\|\mathcal{Z}_i\|}{N} \underset{y_i=0}{\overset{y_i=1}{\gtrless}} \frac{\log\left(\frac{\alpha_0}{1-\alpha_1}\right)}{\log\left(\frac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right)}. \tag{3.12}$$

- $\alpha_1 < 1 - \alpha_0$.

  In this case, $\log\left(\frac{\alpha_0}{1-\alpha_1}\right) < 0$ and $\log\left(\frac{(1-\alpha_0)(1-\alpha_1)}{\alpha_0\alpha_1}\right) > 0$. We rewrite the ML decision rule as follows

$$\|\mathcal{Z}_i\|\log\left(\frac{(1-\alpha_0)(1-\alpha_1)}{\alpha_0\alpha_1}\right) \underset{y_i=1}{\overset{y_i=0}{\gtrless}} N\log\left(\frac{1-\alpha_1}{\alpha_0}\right) \tag{3.13}$$

$$\frac{\|\mathcal{Z}_i\|}{N} \underset{y_i=1}{\overset{y_i=0}{\gtrless}} \frac{\log\left(\frac{\alpha_0}{1-\alpha_1}\right)}{\log\left(\frac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right)}. \tag{3.14}$$

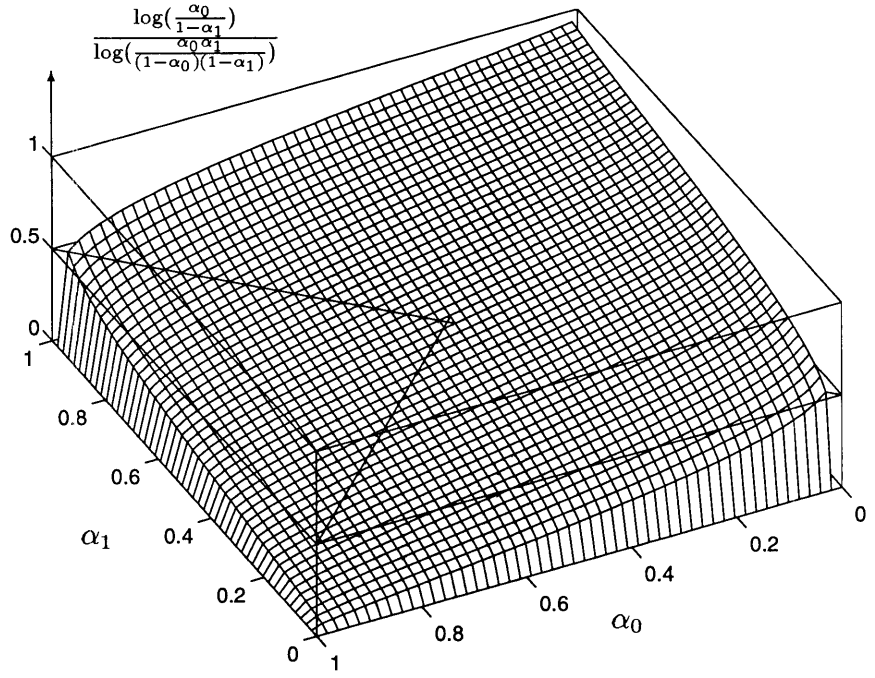


Figure 3-2: Pixel Assignment Threshold Function

In both cases, we find the same threshold on the number of black pixels that must be present at each position i in the observed images, but with different hypotheses deciding whether the template should have that pixel assigned to black or white. In the region $\alpha_1 > 1 - \alpha_0$, foreground pixels are more likely to be observed as black than background pixels, which is the normal case. In the region $\alpha_1 < 1 - \alpha_0$, background pixels are more likely to be observed as black than foreground pixels, and the channel can be interpreted as producing "reverse video". Figure 3-2 shows a plot of this threshold as a function of the channel parameters.

One of the interesting things to point out from the analytical expressions and plot of the threshold function is that there are four regions of interest:

I $\alpha_1 > 1 - \alpha_0$ and $\alpha_0 < \alpha_1$.

This is the "normal" case where the threshold sets the minimum fraction of black pixels that must be present in the observed data vector to decide that the associated template pixel is black. The threshold is strictly greater than $\frac{1}{2}$.

In this region foreground pixels are more likely to be observed as black than background pixels. Since $\alpha_1 > \frac{1}{2}$, the channel has a tendency to conserve black pixels. With no restriction on α_0, except that $\alpha_0 < \alpha_1$, there can be a large added contribution to the observation of black from background pixels if $1 - \alpha_0 > \frac{1}{2}$. This possibility is accounted for by having a large threshold.

II $\alpha_1 > 1 - \alpha_0$ and $\alpha_0 > \alpha_1$.

This is the "normal" case where the threshold sets the minimum fraction of black pixels that must be present in the observed data vector to decide that the associated template pixel is black. The threshold is strictly less than $\frac{1}{2}$.

In this region foreground pixels are more likely to be observed as black than background pixels. Since $\alpha_0 > \frac{1}{2}$, the channel has a tendency to conserve background pixels, which means that the observed black pixels more likely came from foreground pixels only, and since there is little contribution to black from background pixels, this explains the low threshold.

III $\alpha_1 < 1 - \alpha_0$ and $\alpha_0 < \alpha_1$.

This is the "reverse video" case where the threshold sets the minimum fraction of black pixels that must be present in the observed data vector to decide that the associated template pixel is white. The threshold is strictly greater than $\frac{1}{2}$.

In this region background pixels are more likely to be observed as black than foreground pixels. Since $1 - \alpha_0 > \frac{1}{2}$, the channel has a tendency to flip background pixels. With no restriction on α_1, except that $\alpha_1 > \alpha_0$, there can be a large added contribution to the observation of black pixels from foreground pixels if $\alpha_1 > \frac{1}{2}$. This possibility is accounted for by having a large threshold.

IV $\alpha_1 < 1 - \alpha_0$ and $\alpha_0 > \alpha_1$.

This is the "reverse video" case where the threshold sets the minimum fraction of black pixels that must be present in the observed data vector to decide that the associated template pixel is white. The threshold is strictly less than $\frac{1}{2}$.

In this region background pixels are more likely to be observed as black than foreground pixels. Since $\alpha_1 < \frac{1}{2}$, the channel has a tendency to flip foreground pixels, which means that the observed black pixels more likely came from background pixels only, and since there is little contribution to the black from foreground pixels, this explains the low threshold.

In the case of $\alpha_0 = \alpha_1$, the threshold becomes $\frac{1}{2}$, which is what one would expect since both hypotheses are equally likely. It is also interesting to note that the analysis carried out on regions III and IV would be the same as that of regions I and II if the hypotheses were switched and the threshold was compared to the fraction of *white* pixels in the observed data vector $\mathcal{Z}_i$ instead.

We define the number of matching pixels $\Theta_i \equiv \sum_{j=1}^{N} z_{j_i}$, and the number of mismatching pixels $\Delta_i \equiv N - \sum_{j=1}^{N} z_{j_i}$, so that Θ_i and Δ_i are the number of pixels that are ON and OFF, respectively, in the observed image vector $\mathcal{Z}_i$. We can alternatively write the ML decision rule (3.10) as follows

$$(\Delta_i + \Theta_i)\log\left(\frac{\alpha_0}{1-\alpha_1}\right) + \Theta_i\log\left(\frac{(1-\alpha_0)(1-\alpha_1)}{\alpha_0\alpha_1}\right) \underset{y_i=1}{\overset{y_i=0}{\gtrless}} 0 \tag{3.15}$$

$$(\Delta_i + \Theta_i)\log\left(\frac{\alpha_0}{1-\alpha_1}\right) - \Theta_i\log\left(\frac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right) \underset{y_i=1}{\overset{y_i=0}{\gtrless}} 0 \tag{3.16}$$

$$\Theta_i\log\left(\frac{\alpha_1}{1-\alpha_0}\right) - \Delta_i\log\left(\frac{\alpha_0}{1-\alpha_1}\right) \underset{y_i=0}{\overset{y_i=1}{\gtrless}} 0. \tag{3.17}$$

In this last form, we interpret the formula $\Theta_i\log\left(\frac{\alpha_1}{1-\alpha_0}\right) - \Delta_i\log\left(\frac{\alpha_0}{1-\alpha_1}\right)$ as the likelihood score of pixel y_i, which we will occasionally denote by $\mathcal{L}(y_i)$. Notice the presence of a pixel matching score $\Theta_i\log\left(\frac{\alpha_1}{1-\alpha_0}\right)$ and a pixel mismatching score $\Delta_i\log\left(\frac{\alpha_0}{1-\alpha_1}\right)$. Both threshold equations, (3.12) and (3.14), disguise the fact that the ML rule will only assign a pixel to template Y if its likelihood score is positive, as is explicitly shown here.

3.1.2 Optimize α_0 and α_1 with $\mathcal{Y}$ fixed

Given a fixed character template Y, and its corresponding set of observed images $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_N\}$, the objective is to find the channel parameters that maximize the likelihood of $\mathcal{Z}$, given by equation (3.5). We can equivalently maximize the log-likelihood of $\mathcal{Z}$. After some algebra we find that

$$\begin{aligned}
\log \Pr\{\mathcal{Z} \mid Y, \alpha_0, \alpha_1\} &= \log \alpha_0 \sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i)(1 - z_{j_i}) + \\
&\quad \log(1 - \alpha_0) \sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i) z_{j_i} + \\
&\quad \log \alpha_1 \sum_{i \in \mathcal{G}} \sum_{j=1}^{N} y_i z_{j_i} + \\
&\quad \log(1 - \alpha_1) \sum_{i \in \mathcal{G}} \sum_{j=1}^{N} y_i (1 - z_{j_i}).
\end{aligned} \tag{3.18}$$

We can now derive analytical expressions that can be easily computed for the maximizing values of α_0 and α_1. For α_0,

$$\frac{\partial}{\partial \alpha_0} \log \Pr\{\mathcal{Z} \mid Y, \alpha_0, \alpha_1\} = \frac{1}{\alpha_0} \sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i)(1 - z_{j_i}) - \frac{1}{1 - \alpha_0} \sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i) z_{j_i}. \tag{3.19}$$

Setting the derivative to zero we solve for the maximizing value of α_0

$$\alpha_0 = \frac{\sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i)(1 - z_{j_i})}{\sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i)(1 - z_{j_i}) + \sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i) z_{j_i}} \tag{3.20}$$

$$= \frac{\sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i)(1 - z_{j_i})}{\sum_{i \in \mathcal{G}} \sum_{j=1}^{N} (1 - y_i)} \tag{3.21}$$

$$\alpha_0 = \frac{\sum_{j=1}^{N} \|(\neg Y) \wedge (\neg Z_j)\|}{N \|(\neg Y)\|}. \tag{3.22}$$

A subtlety with the expression for α_0 in (3.22) is that it is very sensitive to the size of the template and of the observed images. Since the expression involves operations on the inverse of the original images, we can make α_0 arbitrarily close to 1 simply by using large sized template canvasses. Through the noise channel we expect some white pixels to turn black, and α_0 to have a fixed value that is realistically not 1. Background noise is observed when a white pixel in the original undistorted image is mapped to a black pixel in the observed image; it originates from "white" pixels in the original document being realized as "black" in the observed image. However, this effect is only present in the the mapping of the pixels of the original document image to pixels of the observed image. The appropriate value for α_0 should be calculated from the discrepancies between the observed image and the original image. Given the set of templates $\mathcal{Y}$ we generate a template reconstructed image $\mathcal{I}'$ to be the assumed original image, as described in section 2.1, so that

$$\Pr\{\mathcal{I} \mid \mathcal{I}', \alpha_0, \alpha_1\} = \prod_{i \in \Omega} \left[\alpha_0^{(1-z_i)}(1-\alpha_0)^{z_i}\right]^{(1-y_i)} \left[\alpha_1^{z_i}(1-\alpha_1)^{(1-z_i)}\right]^{y_i}. \tag{3.23}$$

Equation (3.23) is the likelihood function of the observed image given the reconstructed image. Similarly as before, we can find the maximizing values for α_0 and α_1 by setting to zero the partial derivatives of the logarithm of this likelihood function. With some algebra we find that

$$\begin{aligned} \log \Pr\{\mathcal{I} \mid \mathcal{I}', \alpha_0, \alpha_1\} \;=\; & \log \alpha_0 \sum_{i \in \Omega} (1-y_i)(1-z_i) + \\ & \log(1-\alpha_0) \sum_{i \in \Omega} (1-y_i) z_i + \\ & \log \alpha_1 \sum_{i \in \Omega} y_i z_i + \\ & \log(1-\alpha_1) \sum_{i \in \Omega} y_i (1-z_i). \end{aligned} \tag{3.24}$$

Once again, we can derive analytical expressions that can be easily computed for the

maximizing values of α_0 and α_1. For α_0,

$$\frac{\partial}{\partial \alpha_0} \log \Pr\{\mathcal{I} \mid \mathcal{I}', \alpha_0, \alpha_1\} = \frac{1}{\alpha_0} \sum_{i \in \Omega} (1 - y_i)(1 - z_i) - \frac{1}{1 - \alpha_0} \sum_{i \in \Omega} (1 - y_i) z_i. \quad (3.25)$$

Setting the derivative to zero and solving for α_0 we get

$$\alpha_0 = \frac{\sum_{i \in \Omega} (1 - y_i)(1 - z_i)}{\sum_{i \in \Omega} (1 - y_i)(1 - z_i) + \sum_{i \in \Omega} (1 - y_i) z_i} \quad (3.26)$$

$$= \frac{\sum_{i \in \Omega} (1 - y_i)(1 - z_i)}{\sum_{i \in \Omega} (1 - y_i)} \quad (3.27)$$

$$\alpha_0 = \frac{\|(\neg \mathcal{I}') \wedge (\neg \mathcal{I})\|}{\|(\neg \mathcal{I}')\|}. \quad (3.28)$$

The numerator $\|(\neg \mathcal{I}') \wedge (\neg \mathcal{I})\|$ gives the number of white pixels in the reconstructed image that were realized as white pixels in the observed image. Divided by the total number of white pixels present in the reconstructed image $\|(\neg \mathcal{I}')\|$, this fraction is the probability that a white pixel is mapped to a white pixel; the background channel parameter α_0. We follow the same procedure to solve for α_1,

$$\frac{\partial}{\partial \alpha_1} \log \Pr\{\mathcal{I} \mid \mathcal{I}', \alpha_0, \alpha_1\} = \frac{1}{\alpha_1} \sum_{i \in \Omega} y_i z_i - \frac{1}{1 - \alpha_1} \sum_{i \in \Omega} y_i (1 - z_i). \quad (3.29)$$

Setting the derivative to zero we solve for the maximizing value of α_1

$$\alpha_1 = \frac{\sum_{i \in \Omega} y_i z_i}{\sum_{i \in \Omega} y_i z_i + \sum_{i \in \Omega} y_i (1 - z_i)} \quad (3.30)$$

$$= \frac{\sum_{i \in \Omega} y_i z_i}{\sum_{i \in \Omega} y_i} \quad (3.31)$$

$$\alpha_1 = \frac{\|\mathcal{I}' \wedge \mathcal{I}\|}{\|\mathcal{I}'\|}. \quad (3.32)$$

In the expression for α_1, the numerator $\|\mathcal{I}' \wedge \mathcal{I}\|$ gives the number of black pixels in the assumed original image that were mapped to black pixels in the observed image. Divided by the total number of black pixels present in the reconstructed image $\|\mathcal{I}'\|$, this fraction is the probability that a black pixel is mapped as a black pixel; the foreground channel parameter α_1.

3.1.3 Assumptions

There is one subtlety that we must pay attention to. In the calculation of the channel parameters, we made the implicit assumption that the templates in the reconstructed image were disjoint. This is not generally true in this case since every estimated template will contain an estimate of its character, along with extraneous information originating from other characters in the vicinity of every instance of that character. Thus, in the template reconstructed image, there will be a lot of overlapping between the support of every template. This introduces extra noise in the reconstructed image that will be reflected in low values for α_1.

3.1.4 Convergence of $\mathcal{Y}$, α_0, α_1

The goal of template estimation is to estimate a set of templates $\mathcal{Y}$ that maximize the likelihood of the observed image given the template reconstructed image. The process we follow is an iterative one where we switch back and forth between maximizing for the bitmap templates, and then maximizing for the channel parameters, using the updated estimates of one to update the estimates of the others and vice versa.

To test for convergence we calculate the value of our objective function at every iteration and compare it to its value in the previous one. Let us denote our template reconstructed image and channel parameters at the nth iteration by $\{\mathcal{I}'_{(n)}, \alpha_0, \alpha_1\}$, and at the next iteration by $\{\mathcal{I}'_{(n+1)}, \alpha'_0, \alpha'_1\}$. With this notation our convergence test looks as follows

$$\frac{\Pr\left\{\mathcal{I} \mid \mathcal{I}'_{(n+1)}, \alpha'_0, \alpha'_1\right\}}{\Pr\left\{\mathcal{I} \mid \mathcal{I}'_{(n)}, \alpha_0, \alpha_1\right\}} < \eta \tag{3.33}$$

where η is the desired measure of convergence. We cast (3.33) into a more convenient form by taking the log:

$$\log \Pr\left\{\mathcal{I} \mid \mathcal{I}'_{(n+1)}, \alpha'_0, \alpha'_1\right\} - \log \Pr\left\{\mathcal{I} \mid \mathcal{I}'_{(n)}, \alpha_0, \alpha_1\right\} < \log \eta. \tag{3.34}$$

When the log-probability differences of our objective function between two iterations falls below the set threshold, we stop iterating. A little algebra on (3.24) shows that each of the log-probability terms above can be written as

$$\begin{aligned} \log \Pr\{\mathcal{I} \mid \mathcal{I}', \alpha_0, \alpha_1\} \;=\; & |\mathcal{I}| \log \alpha_0 - \|\mathcal{I}\| \log\left(\tfrac{\alpha_0}{1-\alpha_0}\right) - \|\mathcal{I}'\| \log\left(\tfrac{\alpha_0}{1-\alpha_1}\right) + \\ & \|\mathcal{I} \wedge \mathcal{I}'\| \log\left(\tfrac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right). \end{aligned} \tag{3.35}$$

Equation (3.35) above is the convergence measure for the template estimation and channel parameters estimation procedure. It shows the quantities that need to be computed at every step of the iteration to compare to those of the previous step in order to assess convergence.

3.1.5 Experimental Results

We will use the image in figure 3-3 to illustrate the procedure we have just described. The image was obtained by scanning the original image at a resolution of 300dpi. Figure 3-4 shows the locations of all the character origins on the page. The origins were obtained by running a commercial OCR program on the image that returned bounding box information on each character, as well as a label corresponding to the character at that location. The origin of each character was then taken to be at the lower left corner of the bounding box for that character. The recognition routines used connected components to isolate each character, and extract bounding box information from the isolated characters. In some cases, adjacent characters were connected in the observed image, such as the `ry` in `Recovery` in the second line, or `mm` in the first line after `ECONOMICS`, and were identified as a single character and labeled as the first observed character. For this reason, only the first glyph in these cases

has an origin, the other glyph was not given an individual label or an origin by the recognizer.

With the origins known, we proceed to obtain the observed instances of a character by extracting a fixed size image region about each of the origins of that character. Figure 3-5 shows all the observed instances of the character "a". Note the proper image for the "a" is in the middle of all the canvasses, and surrounded by complete and partial images of other characters. There are slight differences between each of the observed instances of the "a", particularly in the edges and in the thickness of each glyph. These are the variations that the ML template tries to capture through the noise model. In the estimation procedure we implicitly assume that the presence of other character images are extreme artifacts due to the noise model. This is not true, and in the next section we discuss how to deal with them.

Figure 3-6 shows all the ML templates after having applied the Maximum Likelihood template estimation routine to all the observed images of each character. Note the presence of the proper support in each character template. In the case of the "h", notice the marked presence of the corresponding images for the characters "t" and "e" to either side. This is because the character "h" occurred most frequently in the word `the`, and the pattern for the "t" and the "e" cannot be distinguished from the one for the "h". There are a few degenerate cases to observe, namely the templates for the "M" and for the "N". Both these character occur only once in the original image, and the best estimate for each one is simply the observed instance.

Figure 3-7 shows the Maximum Likelihood template reconstructed image. The template reconstructed image is generated by placing the origin of each character template at all origins on the image plane that are labeled with that character, and painting the corresponding template image on the canvass. The discrepancies between the template reconstructed image and the original image are very marked. This example shows the high degree of overlap that exists between all of the imaged templates, which is clearly in violation of the non-overlapping criterion that was emphasized by the document generation models, and results in a very unreadable reconstructed image. The next section develops an approach that refines these ML templates.

83
Figure 7. Recovery of different fractions in the top of the jigbed.

The relation between the time, during which the various fractions go to the top of the bed and the l/w-ratio has not been quantified. It is however clear that if the l/w-ratio increases, the separation of the Asg becomes more difficult or for a high recovery of SiC even impossible. For this reason the recovery has to be low for a high grade SiC product. With increasing jig time more SiC will move to the top, because in the last stage of the demixing process the separation will take place based on the difference in shape of the particles.

ECONOMICS

The 3-8 mm fraction of the jaw crusher product contains, according to figure 2, 20 wt percent of the total Asg/SiC mixture. With 70% SiC in the feed and 40% recovery, 1100 ton SiC can be extracted by jigging. According to the German mother company of Elektroschmelzwerk Delfzijl

Figure 3-3: Original Image

83
Figure 7. Recovery of different fractions in the top of the jigbed.

The relation between the time, during which the various fractions go to the top of the bed and the l/w-ratio has not been quantified. It is however clear that if the l/w-ratio increases, the separation of the Asg becomes more difficult or for a high recovery of SiC even impossible. For this reason the recovery has to be low for a high grade SiC product. With increasing jig time more SiC will move to the top, because in the last stage of the demixing process the separation will take place based on the difference in shape of the particles.

ECONOMICS

The 3-8 mm fraction of the jaw crusher product contains, according to figure 2, 20 wt percent of the total Asg/SiC mixture. With 70% SiC in the feed and 40% recovery, 1100 ton SiC can be extracted by jigging. According to the German mother company of Elektroschmelzwerk Delfzijl

Figure 3-4: Character Origins

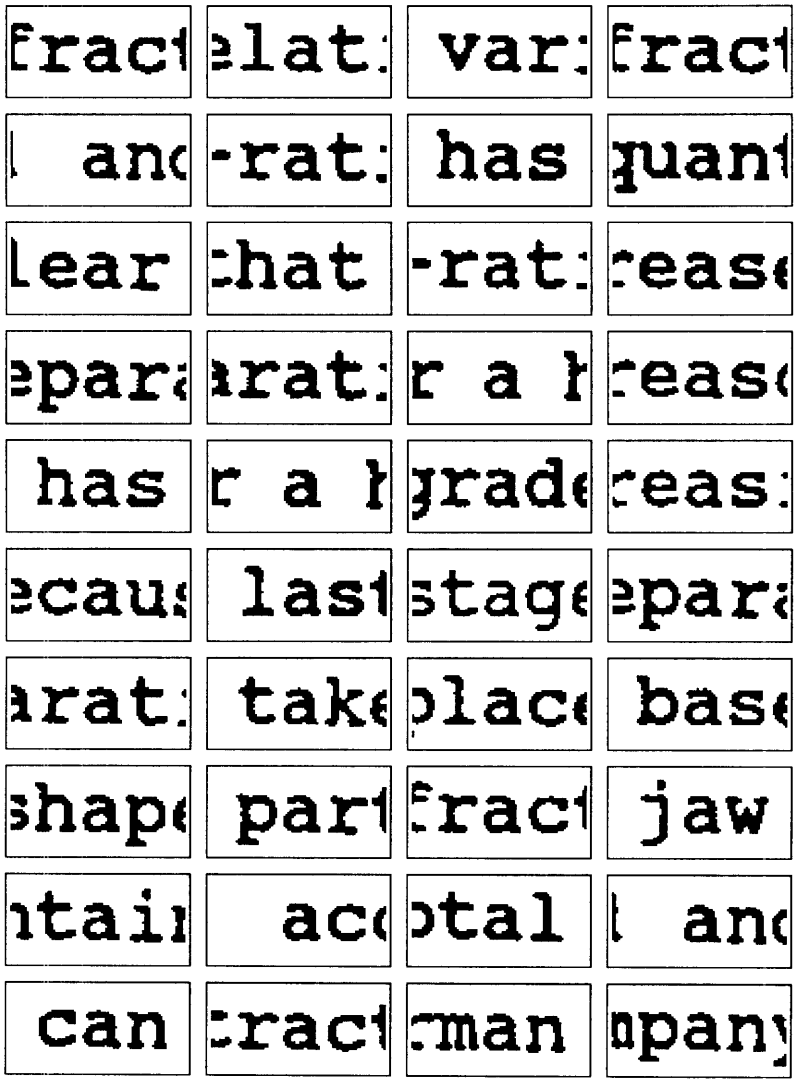


Figure 3-5: Extracted Instances of Character "a"

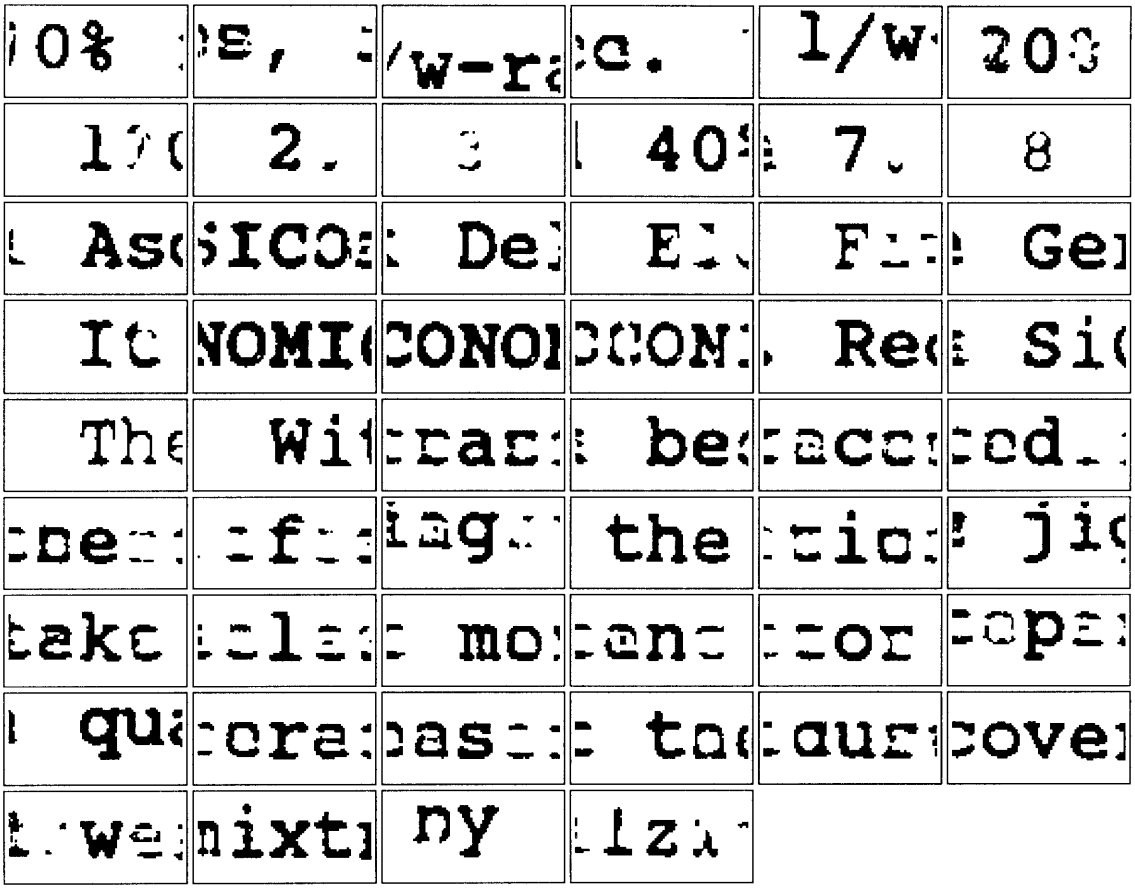


Figure 3-6: Maximum Likelihood Estimated Templates

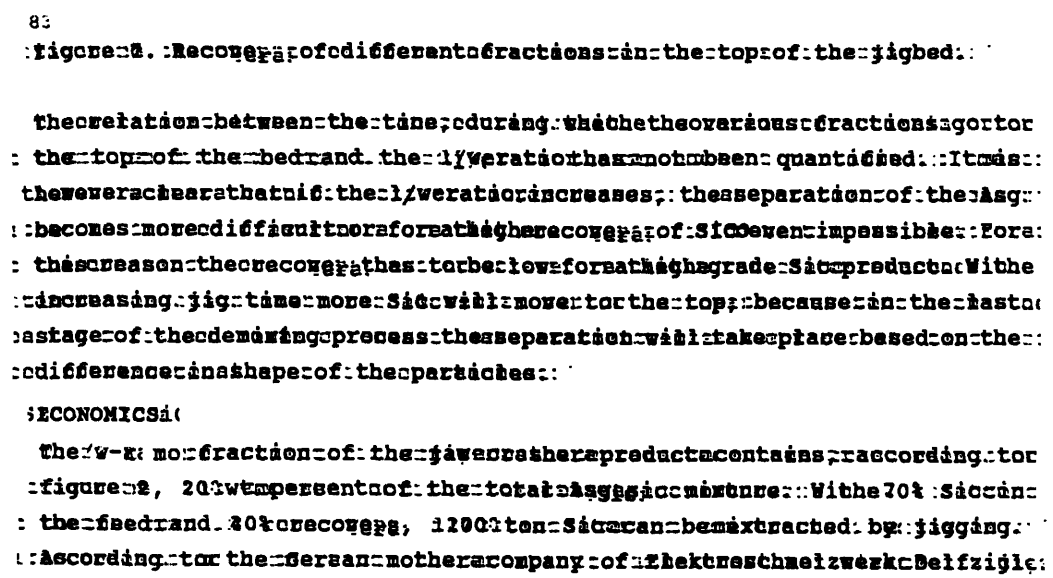


Figure 3-7: Maximum Likelihood Template Reconstructed Image

3.2 Maximum Likelihood Template Refinement

Through the independent ML template estimation routine described above, every template contains an estimate for the shape of each character. These templates also have extraneous information that does not correspond to the actual template. This is a result of the nature of the character samples. Recall that one of the problems we are addressing is the lack of isolated characters to train our templates. Given the origins, we simply extract a region of image about this origin, hoping to capture the entire character image corresponding to the label associated with that origin. Assuming we capture the entire image of a given character, we also capture the images of other characters in the vicinity of this one. All the character samples for a given label then contain this extra information that carries through the ML template estimation routine, as seen in figure 3-6. This simple procedure gives a *superset* of correct pixels for each character template. Simply notice that the presence of extra pixels in each character sample yields a high value for $1 - \alpha_1$, the foreground noise probability, in the mapping from the reconstructed image to the observed image. This results in a lower threshold for the number of black pixels that must be present in the data vector

for each template pixel, and therefore the estimation will be very tolerant of small numbers of black pixels in each data vector, which intuitively results in a superset of correct pixels. Our goal is to now remove pixels from all the templates so that only the "proper" character template image remains in each one.

The central problem is that the reconstructed image does not visually represent what we expect the assumed original image to look like. The template reconstructed image should minimize the error probability between what we are estimating to be the original template reconstructed image, $\mathcal{I}'$, and the observed sample image, $\mathcal{I}$. Given the set of ML templates, we reconstruct our assumed original image by placing at every origin location in the image canvass the template associated with every labeled origin. The template reconstructed image is very hard to understand because the templates overlap extensively; this overlapping is in violation of the non-overlapping criterion imposed by our model. Therefore, our "cleaning" routine must somehow maximize the fit between the observed image and the template reconstructed image while strictly enforcing the non-overlapping criterion.

The approach must be a principled one, where we do not make any assumption as to the shape and/or positioning of the character with respect to the origins we are given. Using ad-hoc techniques, such as looking for connected components, makes the assumption that we have some prior knowledge about the characters. This might not necessarily be true for certain types of fonts, such as cursive, where all the characters within a word are connected.

To clean the templates, we must decide which pixels in a template belong to that character, and which pixels do not. To be precise in our analysis, we define a few terms. We will refer to a template pixel w ($w \in \mathcal{G}$) as a *template indexed pixel.* We will refer to a pixel in the template reconstructed image y ($y \in \Omega$) as an *image indexed pixel.* Every template indexed pixel is mapped to a set of locations on the image plane corresponding to the relative location of this pixel on the template [1] with respect to all the origins of that template on the image plane. The term *conflicting pixels* refers to the event that more than one template indexed pixel contributes to a particular

[1] With respect to the local origin of the template.

black pixel in the reconstructed image; in other words, two template indexed pixels are said to be in conflict if they are mapped to at least one same location in the template reconstructed image plane.

Associated with the decision to maintain or erase a template pixel is a measure of the improvement or deterioration between the reconstructed image and the original image; specifically, every pixel y carries a "score" associated with it which is its likelihood score $\mathcal{L}(y)$. The specifics of how this likelihood score plays a part in the template cleaning process will be addressed later.

We now begin the analysis of template cleaning procedure. As a first case we will consider a single template pixel that maps to several locations on the template reconstructed image without any conflicts. In the event that a pixel has no conflicts, then it is obvious that it will contribute its entire likelihood score to the likelihood score of the reconstructed image. If we erase this template pixel, we must erase the pixels at all the locations on the reconstructed image where it mapped to, incuring a change in the likelihood of observed image equal to the likelihood score of that pixel.

Let us look at the case where two pixels from two different templates always map to the same locations on the reconstructed image. Turning one of these template pixels off **does not necessarily** correspond to turning that pixel off in every location of the reconstructed image where that particular template occurs. The other template will still contribute its black pixel to those locations, and turning one of these template pixels off does not change the reconstructed image. As a result of this, the likelihood score of the reconstructed image will not change. The only thing that has been gained here is reconciliation with the non-overlapping criterion. Thus, the likelihood score of each pixel does not entirely contribute to the likelihood score of the reconstructed image when there are conflicts. There is a penalty that is carried on the likelihood of each pixel when it is in conflict with other pixels.

Assume we have the more complicated case of multiple conflicts in a given image pixel, further assume that each of these pixels maps to different locations on the reconstructed image. At each of these other locations, these pixels can also be in conflict with pixels from other templates, and so on. Each of these pixels will have

a certain score associated with them. In order to decide which pixels to turn off, we must also consider all the conflict dependencies that stem out of a given pixel. When conflicts exist, we must somehow measure the aggregate measure of likelihood to the reconstructed image from all of the conflicting pixels, and then decide which of these pixels must be turned off so as to maximize the likelihood of the observed image and satisfy non-overlapping.

Clearly, we must consider all conflicts that propagate from every candidate pixel in the reconstructed image in order to decide which choice of pixel assignments maximizes the fit between the reconstructed and observed image while enforcing the non-overlapping criterion. This entails generating a "conflict closure" that contains all the conflict dependencies stemming from every pixel in that closure. We now formalize the definition of this closure.

If a particular template $Y^{(i)}$ is imaged n times in the observed image, then every ON pixel $w \in \mathcal{G}$ of that template has a mapping to n distinct locations in the template reconstructed image. We define the function $S_w = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $S_w \subseteq \Omega$, that returns the set of all locations on the template reconstructed image where a given template pixel w can be found.

A conflict closure can now be formally defined as the *transitive closure* of all pixels sets that conflict with each other, as was described above. The transitive closure of a relation can be defined as follows: If P is a "seed" pixel from which we'd like to trace its conflict closure, let

$$\begin{aligned} P_1 &= P \cup \{x | (\exists y)(y \in P \text{ and } S_x \cap S_y \neq \emptyset)\} \\ P_2 &= P_1 \cup \{x | (\exists y)(y \in P_1 \text{ and } S_x \cap S_y \neq \emptyset)\} \\ &\vdots \\ P_{n+1} &= P_n \cup \{x | (\exists y)(y \in P_n \text{ and } S_x \cap S_y \neq \emptyset)\}. \end{aligned}$$

Since the universe of possible conflicting pixels is bounded by the number of pixels in all templates, at some stage, this construction will fail to enlarge the set; $P_{n+1} = P_n$. P_n is then said to be the *transitive closure* of P. We will refer to this transitive closure

by the name *conflict cluster*, or just simply *cluster*, and denote it by the symbol $\mathcal{C}$. We define $\mathcal{S}_\mathcal{C} = \{S_{w_1}, S_{w_2}, \ldots, S_{w_n}\}$ to be the set of image locations of every cluster element. We also define the conflict dependencies between an element $w \in \mathcal{C}$ and every other element in that cluster by the set $D_w = \{x \mid x \neq w, S_x \cap S_w \neq \emptyset\}$. Notice from the definition of a cluster that $D_w \neq \emptyset \ \forall w \in \mathcal{C}$. Associated with every cluster then, is a collection of subsets of $\mathcal{C}$ that contains the conflict dependencies between every element in that cluster, $\mathcal{D}_\mathcal{C} = \bigcup_{w \in \mathcal{C}} \{D_w\}$. Thus we will specify a cluster by:

$$\mathcal{C} = \{w_1, w_2, \ldots, w_n\} \tag{3.36}$$

$$\mathcal{S}_\mathcal{C} = \{S_{w_1}, S_{w_2}, \ldots, S_{w_n}\} \tag{3.37}$$

$$\mathcal{D}_\mathcal{C} = \{D_{w_1}, D_{w_2}, \ldots, D_{w_n}\}. \tag{3.38}$$

In any given template reconstructed image there will be at least one cluster. We will identify different clusters by the index i, as in $\mathcal{C}_i$. There are several important features about conflict clusters to point out.

Every template indexed pixel in $\mathcal{Y}$ necessarily belongs to one unique conflict cluster $\mathcal{C}_i$. Specifically, if there are a total of M conflict clusters in a template reconstructed image and T templates in $\mathcal{Y}$, then

$$\sum_{i=1}^{M} |\mathcal{C}_i| = \sum_{i=1}^{T} \left\| Y^{(i)} \right\|. \tag{3.39}$$

From the definition of a cluster it follows that every conflict cluster is independent of every other conflict cluster

$$\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \tag{3.40}$$

for $i \neq j$.

Every cluster builds its own corresponding section of "patches" across the reconstructed image. Every element w of a cluster contributes a black pixel in the template reconstructed image at all the locations in S_w. From all other template indexed pixels in $\mathcal{C}_i$ there is a corresponding contribution of black pixels in the reconstructed image. More than one template indexed pixel in $\mathcal{C}_i$ may contribute to any particular image

indexed pixel. This follows from the definition, and is as a result of the fact that members of a given cluster share conflict locations on the reconstructed image. A cluster, then, clearly maps to a set of locations on the reconstructed image. This set forms a cover for all distinct locations of elements of a cluster $\mathcal{C}_i$ on the reconstructed image, $\{S_w\}$ $\forall w \in \mathcal{C}_i$. We define the set of image indexed pixels that forms a cover for the image locations of a given cluster as $\{y\}_i$. Formally stated, $\mathcal{C}_i \xrightarrow{S} \{y\}_i$, where

$$\{y\}_i = \bigcup_{w \in \mathcal{C}_i} S_w. \tag{3.41}$$

The set $\{y\}_i$ contains the image indexed pixels that form the image "patches" we referred to earlier for a cluster. Since every template indexed pixel belongs to a cluster, the locations on the image to where this pixel maps to will also belong to some set $\{y\}_i$. And since the reconstructed image is made up of all black template indexed pixels, it follows that

$$\sum_{i=1}^{M} |\{y\}_i| = \|\mathcal{I}'\|. \tag{3.42}$$

The sets $\{y\}_i$ are also independent of each other. Since clusters are independent, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for $i \neq j$, and their corresponding sets of image indexed pixels form a cover for each of them, $\mathcal{C}_i \xrightarrow{S} \{y\}_i$ and $\mathcal{C}_j \xrightarrow{S} \{y\}_j$ it follows that

$$\{y\}_i \cap \{y\}_j = \emptyset \tag{3.43}$$

for $i \neq j$.

Thus we have shown that conflict clusters and their associated sets of image indexed pixels are independent of others. We have also shown that every template indexed pixel in $\mathcal{Y}$ is referenced by some cluster, and that every pixel in the reconstructed image is referenced by an element of some cluster cover in $\mathcal{I}'$. In the analysis, then, we will only consider one conflict cluster and apply the results to all of the clusters.

Let us go back and ask ourselves the relevance of generating these clusters to the

refinement of the templates. We need to be able to determine which pixels in the ML templates need to be turned off such that their supports do not overlap in the template reconstructed image, and the resulting templates maximize $\Pr\{\mathcal{I} \mid \mathcal{I}'\}$. As the argument showed, in order to question the assignment of each black pixel in a template to see if it should be left or cleared, we must also question the assignment of all other template pixels it conflicts with, and all of their conflicts in turn. This led to the generation of conflict clusters to capture all possible conflict dependencies between template pixels. Given these clusters, we now face two tasks.

- The first task, refered to as *Cluster Likelihood*, is to find a measure of how good a particular choice of ON/OFF assignments to pixels in a cluster is.
- The second task, refered to as *Cluster Assignment*, is to design an algorithm that can find an ON/OFF assignment to a cluster that maximizes Cluster Likelihood.

3.2.1 Cluster Likelihood

For a given cluster $\mathcal{C} = \{w_1, w_2, \ldots, w_n\}$ there is an associated pixel assignment space $\Gamma = \{\gamma_{w_1}, \gamma_{w_2}, \ldots, \gamma_{w_n}\}$, where $\gamma_w \in \{\text{ON,OFF}\}$, that assigns each one of the template indexed pixels in a cluster to be either turned ON or OFF in its respective template. A particular partition of this space, Γ_i, gives a valid pixel assignment to a cluster such that there are no conflicts between the template indexed pixels w for which $\gamma_w = \text{ON}$.

For a given cluster $\mathcal{C}$ and its associated set of image indexed pixels $\{y\}$, there is also an associated set of pixels in the observed image that $\{y\}$ maps to through the noise channel. We will define this set as $\{z\}$, and denote the mapping by $\{y\} \xrightarrow{\alpha} \{z\}$. Given a cluster cover $\{y\}$, its associated mapping to the observed image $\{z\}$, the noise model, and a partition Γ_i on $\{y\}$, we define:

$\phi \equiv$ number of pixels in the cover ($|\{z\}| = |\{y\}|$).

$\varphi \equiv$ number of ON pixels in $\{z\}$ (in the observed image).

$\Sigma \equiv$ number of ON pixels in $\{y\}$ (in the template reconstructed image).

$\Theta \equiv$ number of ON pixels in $\{y\}$ that are ON in $\{z\}$.

$\Delta \equiv$ number of ON pixels in $\{y\}$ that are OFF in $\{z\}$.

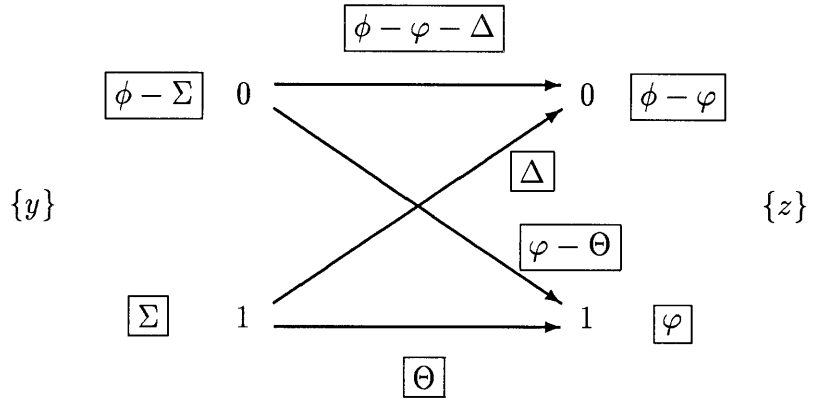


Figure 3-8: Cluster Cover Through Noise Model $\{y\} \xrightarrow{\alpha} \{z\}$

Figure 3-8 illustrates how the quantities in these variables map through the noise channel, giving us a stochastic relation between $\{y\}$ and $\{z\}$. The variables ϕ and φ only depend on the particular cluster we are dealing with. The variables Σ, Δ, and Θ, where $\Sigma = \Theta + \Delta$, depend on the particular choice of assignment Γ_i. Through the noise model, and refering to figure 3-8, the probability of a partition Γ_i, that is, of a given assignment of ON/OFF pixel sequence is

$$\Pr\{\{z\} \mid \{y\}\} = \alpha_0^{(\phi-\varphi-\Delta)}(1-\alpha_0)^{(\varphi-\Theta)}\alpha_1^{\Theta}(1-\alpha_1)^{\Delta}. \tag{3.44}$$

After some algebra the log-probability can be written as

$$\log \Pr\{\{z\} \mid \{y\}\} = \phi \log \alpha_0 + \varphi \log\left(\tfrac{1-\alpha_0}{\alpha_0}\right) - \Delta \log\left(\tfrac{\alpha_0}{1-\alpha_1}\right) + \Theta \log\left(\tfrac{\alpha_1}{1-\alpha_0}\right). \tag{3.45}$$

From partition to partition, the quantities ϕ and φ remain constant, since the cluster does not change. Thus the likelihood measure difference for partitions in a given cluster cover is given by

$$\Theta \log\left(\tfrac{\alpha_1}{1-\alpha_0}\right) - \Delta \log\left(\tfrac{\alpha_0}{1-\alpha_1}\right). \tag{3.46}$$

Notice the match and mismatching scores for good (Θ) and bad (Δ) pixel assignments. The partition we are looking for is the one that maximizes (3.46). For a symmetric

noise channel, where $\alpha_0 = \alpha_1$, note that the best partition has the largest number of matching pixels, Θ, and the smallest number of mismatching pixels, Δ, which is intuitively what one would expect.

The Cluster Assignment Problem attempts to pick a set of pixels from the cluster $\mathcal{C}$ that maximize the cluster likelihood score $\mathcal{L}(\mathcal{C}) = \Theta \log\left(\frac{\alpha_1}{1-\alpha_0}\right) - \Delta \log\left(\frac{\alpha_0}{1-\alpha_1}\right)$ from among all possible assignments. This is the measure we were looking for. What remains now is to find an algorithm that can quickly and efficiently find a valid pixel assignment that maximizes cluster likelihood. In appendix A we prove that the Cluster Assignment Problem is NP-complete.

3.2.2 Assumptions

We will assume that the channel parameters α_0 and α_1 do not change for different cluster assignments Γ_i. There are two reasons for making this assumption. The first is that if we assume that the number of elements in a cluster $\mathcal{C}$ is small compared to the quantities $|\mathcal{I}|$, $\|\mathcal{I}\|$, $\|\mathcal{I}'\|$, and $\|\mathcal{I} \wedge \mathcal{I}'\|$, then different partitions of the conflict space Γ_i will not appreciably affect the values of α_0 and α_1. Simply note that the channel parameters can be rewritten as

$$\alpha_0 = \frac{|\mathcal{I}| - \|\mathcal{I}\| - \|\mathcal{I}'\| + \|\mathcal{I} \wedge \mathcal{I}'\|}{|\mathcal{I}| - \|\mathcal{I}'\|}$$

$$\alpha_1 = \frac{\|\mathcal{I} \wedge \mathcal{I}'\|}{\|\mathcal{I}'\|}$$

and an incremental analysis shows that the channel parameters are not affected by small variations of $\|\mathcal{I}'\|$ and $\|\mathcal{I} \wedge \mathcal{I}'\|$.

The second reason is that in the implementation this would avoid having to re-compute the optimal channel parameters for every assignment Γ_i.

3.2.3 Cluster Assignment

Given a conflict cluster $\mathcal{C} = \{w_1, \ldots, w_n\}$, the Cluster Assignment Problem, or CAP for short, is to choose a subset $\mathcal{C}' \subseteq \mathcal{C}$ of pixels to turn ON such that cluster likelihood is maximized. As appendix A shows, the CAP under strict observation of the non-overlapping criterion is NP-complete. The theory of NP-completeness tells us that we can expect any algorithm designed to find an optimal pixel assignment to be exponential in cost. What if we remove the non-overlapping criterion such that the template reconstructed image still yields the most likely candidate to have produced the observed image? Appendix A shows that this variant of the CAP is also NP-complete.

To have labeled the Cluster Assignment Problem NP-complete, implies that this problem is inherently intractable. There is no known algorithm that can solve an NP-complete problem quickly. The answer can be obtained by searching all possible combinations to find out which one has the highest likelihood score. However, this scheme is operationally impractical when the number of elements to be searched through in the space of solutions is large.

By relaxing the optimality criterion in the CAP we transform this combinatorial optimization problem, where our goal is to find an optimal valid solution, into a semi-optimization problem, where we tolerate less than optimal solutions. Our main objective now is to develop an "efficient" algorithm that can find an ON/OFF assignment to a cluster that maximizes its cluster likelihood. By efficient we mean an algorithm that arrives at a good solution in a relatively fast way.

We will focus on solutions to the CAP under strict observation of the non-overlapping criterion. The goal now is to select the set of non-conflicting cluster elements $\mathcal{C}' \subseteq \mathcal{C}$, where $D_u \cap D_v = \emptyset \ \forall u, v \in \mathcal{C}'$, such that $\mathcal{L}(\mathcal{C}') = \sum_{w \in \mathcal{C}'} \mathcal{L}(w)$ is maximized. A *greedy* algorithm seems like a very attractive strategy to do this. The greedy algorithm tries to maximize the aggregate likelihood by maximizing the likelihood of individual elements in that set. The greedy algorithm used in the CAP is shown in figure 3-9.

```
    Given: C
    begin
1.    P := ∅
2.    C' := C
3.    while C' ≠ ∅ do
        begin
4.        p = argmax_{w∈C'} L(w)
5.        C' ← C' − p
6.        P ← P ∪ p
7.        K = {x | x ∈ C', S_x ∩ S_p ≠ ∅}
8.        C' ← C' − K
        end
9.    γ_p ← ON  ∀p ∈ P
10.   γ_p ← OFF ∀p ∈ C − P
    end
```

Figure 3-9: Greedy CAP Algorithm

The greedy algorithm first selects the cluster pixel p with the highest likelihood score, removes it from the cluster $\mathcal{C}'$, and adds it to the set P. Next it finds all the conflicts to p, $\mathcal{K}$, and removes them from the cluster $\mathcal{C}'$. The algorithm then selects the next highest scoring pixel from the modified cluster $\mathcal{C}'$ and repeats the procedure until $\mathcal{C}'$ is empty. On completion, the set P will contain the individually highest scoring pixels from the cluster $\mathcal{C}$. The algorithm terminates by making ON assignments to all the pixels in P and assigning all other pixels OFF.

Figure 3-10 shows the results of applying the greedy algorithm to the templates in the example of figures 3-3, 3-4, 3-5, and 3-6. Figure 3-11 shows the greedy template reconstructed image. In the reconstructed image some characters are missing, such as the "u" in `Figure` in the second line, or the "u" in `crusher` in the 12th line. This is because we did not have origins for these glyphs initially, and their images were not used in the estimation routine. There are also some origin misplacements, namely the "r" in `Recovery`. This is because the recognizer calculated the bounding box for the "r" in some instances using the adjacent image for the "y", which resulted in a much lower left hand corner for the bounding box, which we use as origin. We also point

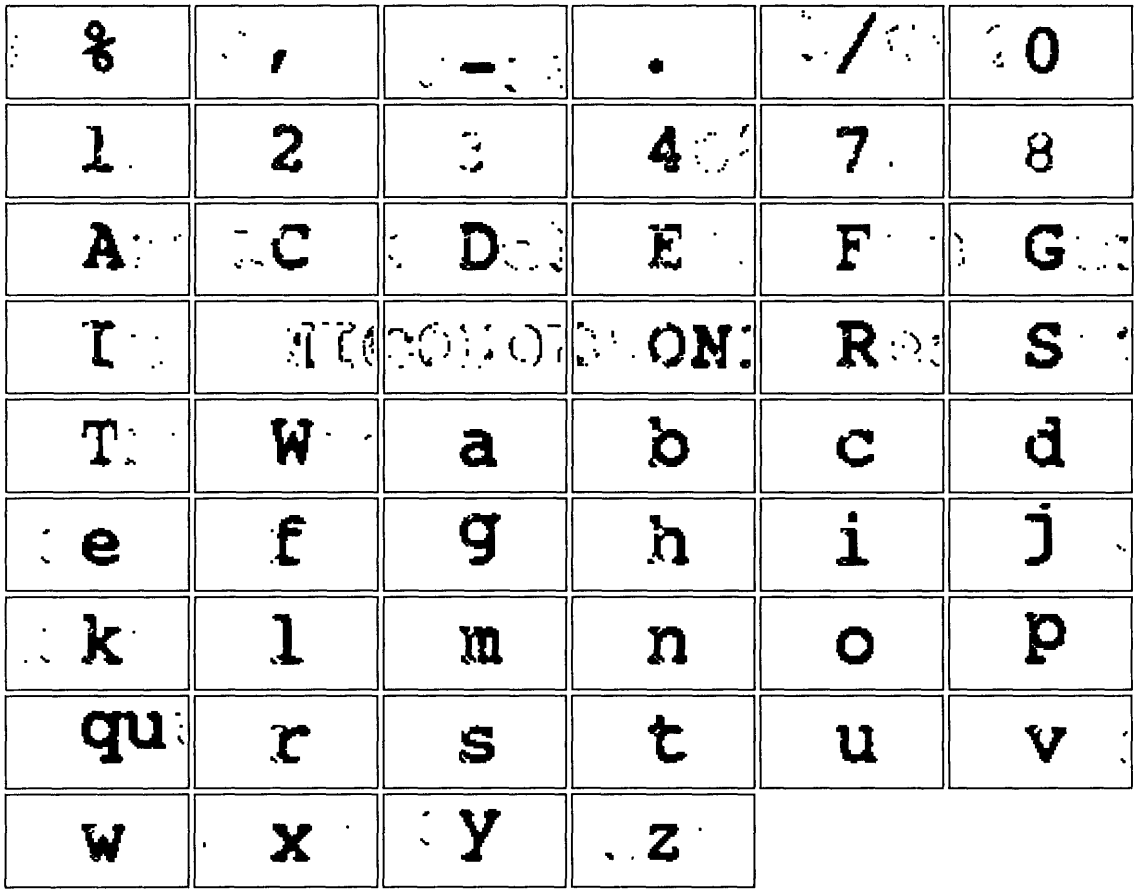


Figure 3-10: Maximum Likelihood Templates Through Greedy Algorithm

out that the transcription was not perfect, and there were some characters that were labeled incorrectly, such as the "/" in the fourth line labeled as an "l", or the "/" in the 13th line after the word **Asg** labeled as an "i". Careful inspection of the greedy templates in figure 3-10 show that these mislabelings did not affect the estimation of the templates. Tests carried out on other images further showed that the template estimation routine is not affected by a small number of mislabelings, and thus the transcription need not be perfect to obtain a good set of estimated templates.

The algorithm performs well as a first approximation to a solution. Although we have certainly not achieved the most optimal solution, the results are very aesthetically pleasing. Intuitively, the likelihood scores for every pixel in a template that

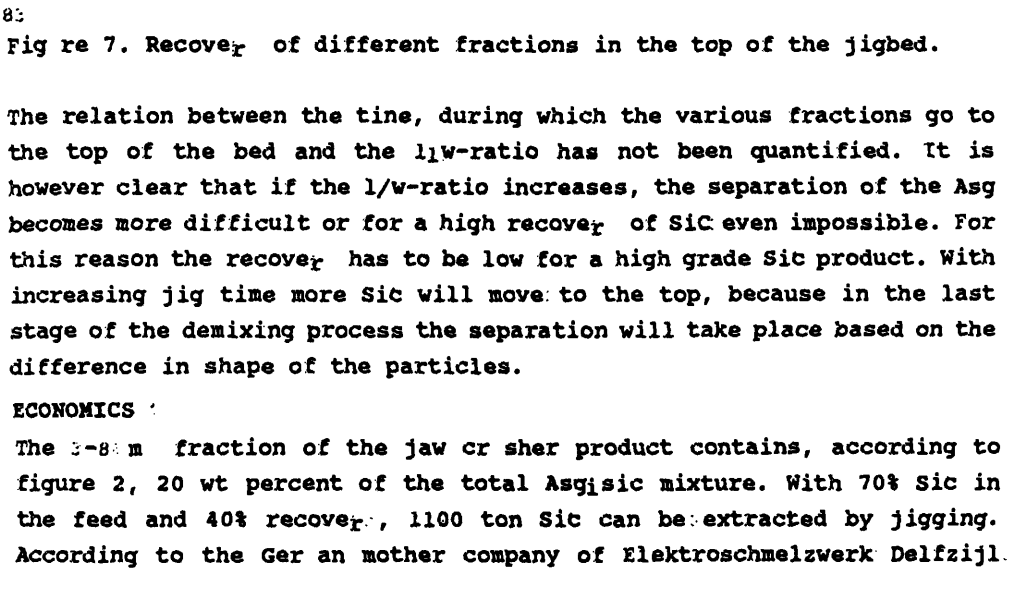


Fig re 7. Recove$_r$ of different fractions in the top of the jigbed.

The relation between the tine, during which the various fractions go to the top of the bed and the l$_1$w-ratio has not been quantified. It is however clear that if the l/w-ratio increases, the separation of the Asg becomes more difficult or for a high recove$_r$ of SiC even impossible. For this reason the recove$_r$ has to be low for a high grade Sic product. With increasing jig time more Sic will move to the top, because in the last stage of the demixing process the separation will take place based on the difference in shape of the particles.

ECONOMICS

The 3-8 m fraction of the jaw cr sher product contains, according to figure 2, 20 wt percent of the total Asg$_i$sic mixture. With 70% Sic in the feed and 40% recove$_r$, 1100 ton Sic can be extracted by jigging. According to the Ger an mother company of Elektroschmelzwerk Delfzijl.

Figure 3-11: Greedy Template Reconstructed Image

correspond to the "proper" character image should be higher than for others. We expect this because every observed image of a character contains the actual character image, along with images of other characters in the vicinity of every instance. There is always a similar pattern of pixels corresponding to the proper shape of the character across all images, and the pixels from this pattern contribute a very high likelihood score to the template pixels. In contrast, presence of other pixels in the ML template corresponds to images of other characters. The likelihood score for any of these pixels depends on the relative frequency of pixels from other characters at that location in the extracted images. Since these pixels do not necessarily form a consistent pattern from image to image, we can expect to have a larger number of mismatching pixels from these template pixels, and thus a lower likelihood score. We will refer to the template pixels that correspond to the true shape of that character as *true pixels*, and to the pixels in a template that correspond to other templates as *ghost pixels.* When placing a template in all its corresponding positions on the reconstructed image, the likelihood score of a true pixel should be higher than that of every other ghost pixel that maps at that location. From the document generation models, a true pixel will

never map to conflict with another true pixel.[2]

The argument above relies on the consistency of character shape from glyph to glyph in every observed instance of that character. We expect the likelihood score of a true pixel to always be higher than the likelihood score of any other ghost pixel it conflicts with. But as the experiment shows, that is not always the case. Notice the presence of "holes" in a lot of the characters, such as the "h", "f", "m", "r", or "f". Upon closer inspection, we find that these holes are pixels that were awarded to the template of the character "e". What is going on here? The character "e" occurred substantially more than any other character on the image. In all of its observed instances, the shape of the actual "e" repeated itself and the likelihood of these true pixels was captured by the greedy algorithm. Yet the greedy algorithm also awarded ghost pixels to the "e". By the sheer number of times that the character "e" was present on the image, a consistent pattern arising from other distinct images also appears in the ML template for the "e". This is the extra spot we find in the "e". How does this come about? The greedy algorithm assigns a pixel that is better than all of its conflicts **individually**. The algorithm does not consider the possibility that an assignment to several conflicting pixels of less individual likelihood can yield a larger aggregate likelihood. In our example, we find that the extra spot in the "e" corresponds to holes in the images of other characters. There are two approaches we can take in order to resolve this.

In the first approach, we can modify the greedy algorithm as follows. Before making an assignment to the next considered highest-scoring pixel, we should "look-ahead" and compare to the aggregate likelihood of all the conflicts to this pixel, which we will call candidates. If the likelihood score of the next highest-scoring pixel is less than the sum of the likelihoods of all the conflicts to this pixel, then we should assign all of the candidate pixels ON, and the considered pixel OFF. Otherwise, the algorithm should proceed as before. As described, this is a one level deep search through all possible assignments. A problem with this approach, however, is that there are no assignments on the cluster pixels initially, and in order to justify the

[2]This is not necessarily true in reality, where templates are designed to overlap slightly.

presence of a pixel in the candidate list we must verify in its candidacy that its possible assignment would contribute a larger likelihood score than all of its conflicts together. But considering these conflicts requires justifying their candidacy in turn, and so forth. This approach is clearly not well suited to resolve this problem.

In the second approach, we simply try to improve on the solution given to us by the greedy algorithm. From the results of our example, we should try to assign the extra spot of the "e" to all other possible locations where we can fit it, such as the "a", "b", "l", etc. We expect the change in likelihood to be positive in the re-assignment of such pixels since this would yield a much closer fit between the reconstructed image and the observed image. This procedure is carried out by local search algorithms. The idea is very simple, the algorithm starts at some initial feasible solution and uses subroutines to improve this solution by searching through a local neighborhood of solutions, and if a better solution is found, it is adopted. There is a tradeoff between searching through small and large neighborhoods. A large neighborhood provides more possibilities for a better solution with the expense of longer search time, whereas a small neighborhood might not provide a much better solution but can be searched through very quickly.

There's a fundamental difference between the two proposed approaches. The first approach branches from the top of the search tree in all directions, trying to find the solution that yields the best score. The second approach pursues the greedy algorithm to a solution at the bottom of the search tree and then does a local search around that solution. The advantages of the second approach are immediate. The greedy algorithm gives a reasonable solution to the problem; the resulting templates are aesthetically pleasing. The local search algorithm now tries to locally find a variant of the solution that yields a higher likelihood score. The local search algorithm for the greedy solution used in the CAP is shown in figure 3-12.

The local search algorithm, from now own refered to as the refinement procedure, is a very conservative and straightforward routine. For every cluster, we first select the currently ON assigned pixel that has the largest likelihood score, m. Following the previous argument, a subset of all of the pixels that conflict with m will be considered

Given: $\Gamma, \mathcal{C}$
begin
1. $\Gamma' := \Gamma$
2. $\mathcal{C}' := \mathcal{C}$
3. **while** there exists an $x \in \mathcal{C}'$ such that $\gamma'_x = \text{ON}$ **do**
   begin
4. $m = \underset{w \in \mathcal{C}' | \gamma'_w = ON}{\text{argmax}} \mathcal{L}(w)$
5. $\mathcal{C}' \leftarrow \mathcal{C}' - m$
6. $\mathcal{U} = \text{CandidateList}(m, \mathcal{C}', \Gamma')$
7. $\mathcal{V} = \text{ValidateCandidateList}(\mathcal{U})$
8. $\mathcal{M} = \text{ValidCandidateConflicts}(\mathcal{V}, \mathcal{C}', \Gamma')$
9. **if** $\sum_{w \in \mathcal{V}} \mathcal{L}(w) \geq \mathcal{L}(m) + \sum_{w \in \mathcal{M}} \mathcal{L}(w)$ **then**
10. $\gamma_w \leftarrow \text{ON} \ \ \forall w \in \mathcal{V}$
11. $\gamma_w \leftarrow \gamma'_w \leftarrow \text{OFF} \ \forall w \in \mathcal{M} \cup m$
12. $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathcal{M}$
   end
13. $\gamma_w \leftarrow \text{ON} \ \ \forall w \in \{x | (\exists y)(x \in \mathcal{C}, y \in \mathcal{Q}, x \neq y, \gamma_x = \text{OFF},$
   $S_x \cap S_y \neq \emptyset, S_x \cap S_v = \emptyset \ \forall v \in \mathcal{C} | \gamma_v = \text{ON})\}$
end

Figure 3-12: Local Search Algorithm for Greedy Solution

as candidate pixels to receive the pixel assignment. From all the conflicts with m, we generate a candidate list $\mathcal{U}$ by selecting those pixels from the conflict list that either have no other conflicts with ON pixels (except with m) or, if a particular conflict has other conflicts of its own with pixels that are ON, then we include it as a candidate if its score is larger than the sum of the scores of all of its conflicts. This is the first sign of conservativeness in the algorithm. More than one OFF conflicts of m may share conflicts with more than one ON pixel, such that considering all candidates together may result in a larger likelihood score against all of their conflicts. The concern here is that individually, these OFF candidates might not give a larger score than all of their ON conflicts. This step is carried out by the procedure CandidateList($\cdot$).

Once we have a list of candidate pixels, we must make sure that an assignment to all of these pixels still observes the non-overlapping criterion. Specifically, there may exist conflicts between candidate pixels. We go through the candidate list in

```
procedure CandidateList(m, C', Γ');
    begin
1.    U := ∅
2.    K = {x | x ∈ C', S_x ∩ S_m ≠ ∅}
3.    for each w ∈ K do
        begin
4.        R = {x | x ∈ C', γ'_x = ON, S_x ∩ S_w ≠ ∅}
5.        if L(w) ≥ Σ_{p∈R} L(p) then
6.            U ← U ∪ w
        end
7.    return U
8.  end
```

Figure 3-13: Procedure CandidateList($m, \mathcal{C}', \Gamma'$)

descending score order looking at all the conflicts of each pixel. Since some candidates were chosen by comparing their score to the sum of their conflicts' scores, they will have conflicts with ON pixels and OFF pixels. We are only concerned with conflicts between pixels in the candidate list. If a high scoring candidate pixel has a conflict with one or more lower scoring candidate pixels, then we remove the lower scoring candidate pixels from the candidate list. This is the second sign of conservativeness in the algorithm. In the event of more than one conflict, the algorithm does not consider the possibility that the aggregate score of all the conflicts is higher than the likelihood score of the first pixel. These types of arbitrary candidacy measures are instituted because verifying the alternatives increases the complexity of the search, and thus the time that it takes for the algorithm to run.[3] This step is carried out by the procedure ValidateCandidateList($\cdot$), which simply returns the list of pixels that have been assigned ON that conflict with a set of valid candidate pixels.

We illustrate the local search algorithm by considering a pixel m and all of its conflicts, $\mathcal{U}$, that qualify to be candidates under the rules mentioned above.

[3]Experimentally we find that the algorithm in its conservative form is still very computationally intensive.

```
procedure ValidateCandidateList(𝒰);
    begin
1.    𝒱 := ∅
2.    while 𝒰 ≠ ∅ do
        begin
3.        v = argmax_{w∈𝒰} ℒ(w)
4.        𝒰 ← 𝒰 − v
5.        𝒱 ← 𝒱 ∪ v
6.        ℛ = {x | x ∈ 𝒰, S_x ∩ S_v ≠ ∅}
7.        𝒰 ← 𝒰 − ℛ
        end
8.    return 𝒱
    end
```

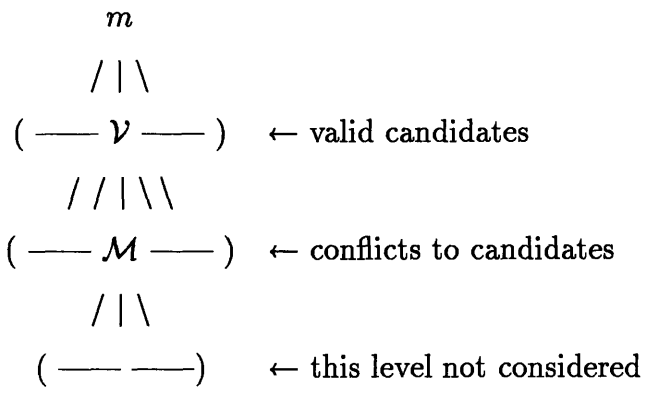
Figure 3-14: Procedure ValidateCandidateList($\mathcal{U}$)

$$
\begin{array}{c}
m \\
/\ |\ \backslash \\
(\ \text{——}\ \mathcal{U}\ \text{——}\ )
\end{array}
\quad \leftarrow \text{conflicts of } m\text{; candidates}
$$

From this candidate list we clean up the conflicts between them. The idea of the local search is that we want to replace the assignment of a pixel m with that of all the candidates that could have received it instead if their likelihood is greater. This means that the candidates are the immediate conflicts of the pixel m. But since there might be new conflicts if we were to assign ON all the pixels in the candidate list, we must remove these conflicts. This is done by validating the candidate list, $\mathcal{U} \rightarrow \mathcal{V}$. By considering the alternative of assigning ON all the pixels in the candidate list, there is a likelihood change of turning ON all of the pixels in this candidate list and having to turn OFF all of their conflicts, which include m and all other conflicts with $\mathcal{V}$, given by $\mathcal{M}$.

```
procedure ValidCandidateConflicts(V, C', Γ');
    begin
1.    M := ∅
2.    for all v ∈ V do
        begin
3.        R = {x | x ∈ C', γ'_x = ON, S_x ∩ S_v ≠ ∅}
4.        M ← M ∪ R
        end
5.    return M
    end
```

Figure 3-15: Procedure ValidCandidateConflicts($\mathcal{V}, \mathcal{C}', \Gamma'$)



Thus, we now compare the likelihood of maintaining m and $\mathcal{M}$ ON, to the likelihood of turning ON the pixels in $\mathcal{V}$ instead. The third sign of conservativeness of this algorithm is found in the depth of search. A more accurate measure would also consider the candidates that stem out of the conflicts in $\mathcal{M}$. This would entail grouping all of the conflicts of $\mathcal{M}$ that are not in $\mathcal{V}$, making sure that they are not in conflict with themselves or with elements in $\mathcal{V}$, and making sure their candidacy considers the possibility of other ON conflicts that are not in $\mathcal{M}$. Clearly, this would increase the depth and accuracy of the search, but at the expense of complexity. In our algorithm we stop at two levels, and execute the algorithm as described.

At every iteration we keep track of all the pixels $\mathcal{M}$ that we erase, since their erasure presents the possibility of assignment to a previously unassigned pixel one more level down in the search tree that we chose not to look at. The last step of the algorithm is to go through all of the conflicts of every element in $\mathcal{M}$ at are unassigned

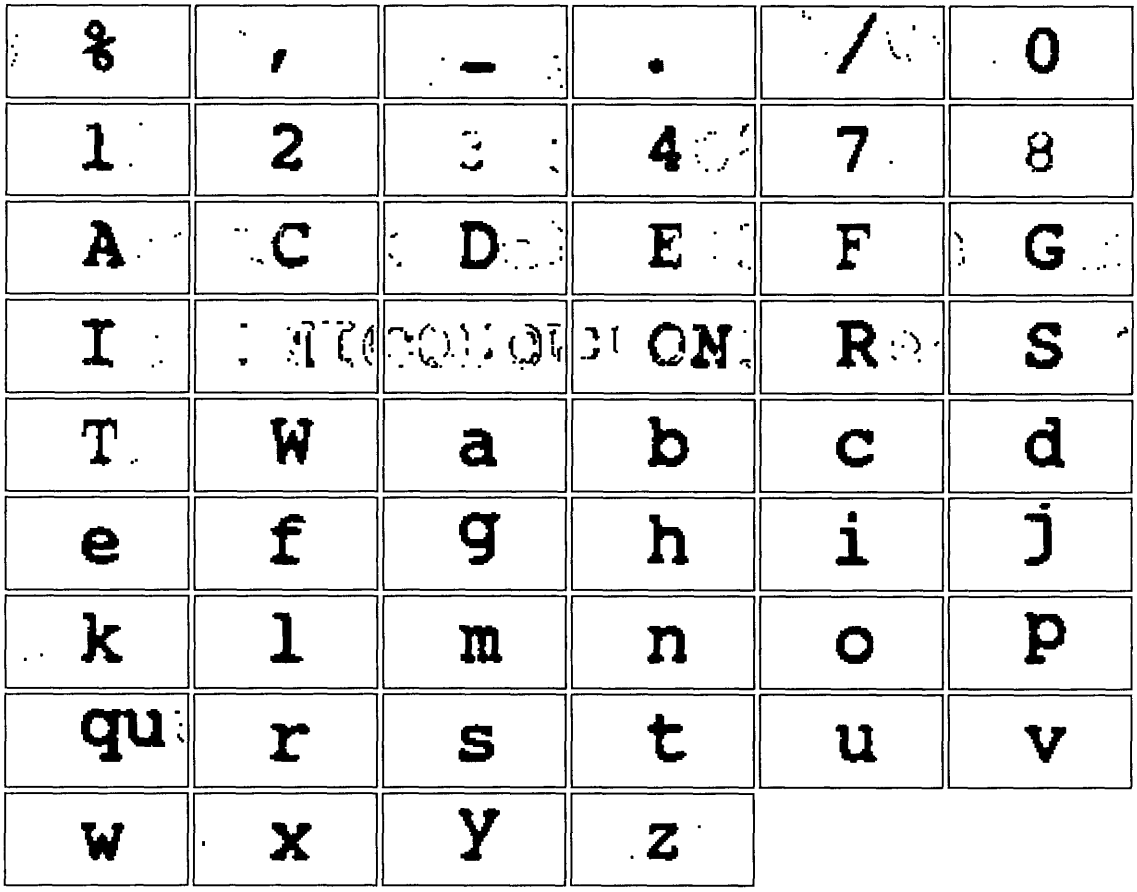


Figure 3-16: Greedy Templates After Refinement

and that do not have any conflicts with assigned pixels, and assign them.

Figure 3-16 shows the results of applying the local search algorithm to the greedy templates in our example. Figure 3-17 shows the refined template reconstructed image. The results are impressive, the "proper" template images have, for the most part, been properly extracted. Most extra material present in the templates that are not part of the character are most likely artifacts due to the small number of instances present.

Table 3.1 shows the channel parameters and likelihood scores for each of the three sets of templates: ML, greedy, and refined. The likelihood score is computed as $\mathcal{L}(\mathcal{I}|\mathcal{I}') = \|\mathcal{I}'\| \log\left(\frac{1-\alpha_1}{\alpha_0}\right) + \|\mathcal{I}' \wedge \mathcal{I}\| \log\left(\frac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right)$. Table 3.2 lists the likelihood

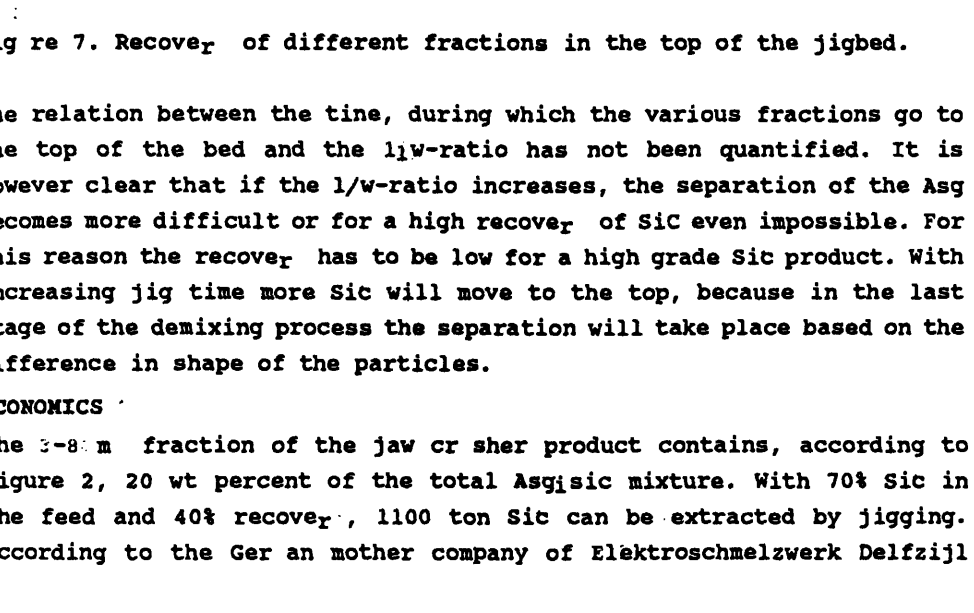


Figure 3-17: Refined Template Reconstructed Image

Table 3.1: Experimental Results using ML Templates

Templates	$\|\mathcal{I}'\|$	$\|\mathcal{I} \wedge \mathcal{I}'\|$	α_0	α_1	$\mathcal{L}(\mathcal{I}\|\mathcal{I}')$
ML	182938	126774	0.995949	0.692989	585795.8750
Greedy	126908	115798	0.990376	0.912456	500144.8750
Refined	132593	120903	0.993002	0.911835	560474.0625

score of the templates if the channel parameters were fixed $\alpha_0 = \alpha_1 = 0.90$ The results for the ML templates scored much higher than the other two. The reason is that there is a very good match between the observed image and the reconstructed image, and the low value for α_1 compensates for the large portion of extra black pixels in the reconstructed image by making the value of $\log\left(\frac{1-\alpha_1}{\alpha_0}\right)$ small. The difference between the number of pixels in the AND of the images and the number of pixels in the reconstructed image is substantial; there are a lot of extra pixels in the image that add extra noise, and show up in the low value of α_1. In the greedy templates we find the channel parameters have values that we would expect for a reasonable approximation to the original image. The number of pixels in the AND of the images and in the reconstructed image are much closer to each other, off by about 10%,

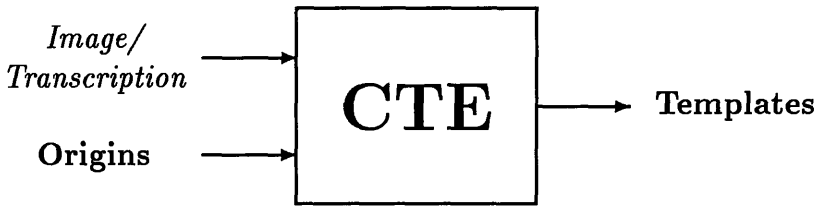
Table 3.2: Experimental Results with $\alpha_0 = \alpha_1 = 0.90$

Templates	$\|\mathcal{I}'\|$	$\|\mathcal{I} \wedge \mathcal{I}'\|$	α_0	α_1	$\mathcal{L}(\mathcal{I}\|\mathcal{I}')$
ML	182938	126774	0.90	0.90	70610 log 9
Greedy	126908	115798	0.90	0.90	104688 log 9
Refined	132593	120903	0.90	0.90	109213 log 9

meaning that the reconstructed image has achieved a much better match. In the refined template reconstructed image, the number of pixels that match the observed image went up by $120903 - 115798 = 5105$ from the greedy reconstructed image. If we compare this to the difference in the number of pixels in the refined reconstructed from the greedy reconstructed, $132593 - 126908 = 5685$ we see that 90% of the new pixels are matching pixels, while only 10% are mismatching pixels. This is reflected in the higher likelihood score for the refined template reconstructed image.

3.3 Summary of Chapter

In this chapter we have developed a procedure that estimates a set of character templates from a document image and an associated set of labeled character origins, as illustrated below



As the analysis has shown, there is no guarantee on the optimality of these templates; the output is judged on empirical results, and strong evidence is given to support the intuitive analysis on the optimality of this procedure.

Chapter 4

Character Template Estimation Given Baselines and Transcription

Having developed a procedure for estimating a set of character templates from known origin locations, we now consider the more general problem of estimating the set of character templates when only the baselines, along which the origins of each character lie, are known. More formally stated, given a stochastic image source and an observed document image generated from this source, the positions of each textline baseline, and the sequence of imaged characters along each baseline, the question is how to determine the set of templates that optimally represent the observed image.

This is a significantly more difficult problem than the first. In the first problem we are given the origins of each character on the observed image, and we are asked to estimate the shape of the templates. With every origin known, we know precisely where each character template was imaged on the document page, and the procedure used to estimate the templates is very straight forward. Every character glyph in the observed data used by the estimation procedure is origin-aligned, which means the estimated templates do not suffer from inaccuracies in the mapping of templates to known origins. Knowledge of the origins carries with it some information on the setwidth of each character as well; we can get very good setwidth estimates by following a procedure similar to the one developed in [10].

In this new problem, the tools we have developed so far are not well suited to

tackle it yet. The difficulties of this new problem is that we must now estimate two things simultaneously, the setwidth of each template, and the origin locations of each imaged character on the observed document page.

4.1 Overview

Conceptually, we would like to develop a procedure that is able to determine the origins of characters along a baseline in an image so we can then use the CTE procedure to estimate the templates.

The systematic approach we take to the problem is as follows: We first develop precise descriptions, in the context of the DID models, of the information we can extract from knowledge of the baselines and transcription, and integrate these descriptions into a page model. We then analyze the operation of the image decoder module described in 2.3, and adapt it to use this page model.

We then develop an iterative procedure where given an initial set of templates we estimate a set of origins on the page, then use the CTE procedure to update the templates, and continue this process until we converge.

4.2 Page Model Given Baselines and Transcription

In this section we examine the general concept of a baseline and the role it plays in the imaging and decoder models, and we also examine the relationship between the transcription of a line of text and the imaging procedure of that text line. This analysis develops a precise description of the information carried by the baselines and transcription.

4.2.1 Baselines

A baseline is a curve serving for location of characters on a page. More precisely, a baseline serves as a basis for the location of origins of characters as a line of text is imaged.

Generally we assume a baseline is a perfectly horizontal line along which characters are imaged. The notion of a baseline, by its definition above, is more powerful than this. In the most frequently encountered documents, baselines are horizontal. They may or may not extend all the way across the image plane. In this thesis document, for example, each baseline extends from inside one inch of the left edge of the paper, to one inch from the right edge of the paper along a horizontal line. In other documents, however, this is not true. Newspapers, for example, have multiple columns, and a baseline for any column is very short. In other written languages, such as Chinese, the "baselines" are vertical, the characters are read vertically along these baselines. This example serves to point out that baselines form a path through the image plane where consecutively placed characters are imaged.

More generally, a baseline can take on any shape. Consider a document where there is only one line of text that spirals from one corner of the page into the center of the page. Any character is still imaged so that its origin lies on the baseline, albeit in a very unorthodox manner.

This instance brings up the issue of orientation of the imaged character along the baseline. In the spiraling baseline situation, it makes more sense to image every character so that its local coordinate system is aligned with a tangent to the baseline at that point. In the class of documents this thesis considers, we will assume that characters are imaged so that their local coordinate axes are aligned with the coordinate axes of the image plane.

Another tacit issue that is not addressed in these examples is the relationship between the transitional setwidth vectors of an image source and the production of a textline image with the baselines when they are known. In the imaging procedure described in section 2.1, the position of characters on the page was given by an aggregate position indicator $\vec{x}_i = \sum_{j=0}^{i-1} \vec{\Delta}_{t_j}$. Under this representation, the sequence of transitions through the Markov source and the particular choices for $\vec{\Delta}_{t_j}$ seem to specify the "baselines" on the image.

We reconcile these two notions by the following argument. Given a baseline we can incorporate it into the imaging model as a constraint on the imaging path across

the image plane. In section 2.1 the setwidth vectors updated the position of the imager assuming a cartesian image plane coordinate system. A baseline model gives flexibility to image a document differently without modifying the setwidth vectors or path through the source. When imaging a new line of text, we simply impose the constraint that the setwidth vectors update the position of the imager **along** the baseline of that line of text.

Empirically, it has been discovered that having accurate baselines of documents helps decode text images by running the image decoder along those paths without spending too much effort trying to identify areas in a page where there is no text.

Consider a book, where every line is perfectly horizontal. We would like to scan the book and run a decoder through the resulting image. As experience has taught many of us, the reproduction of the image we see in the book to the image we transfer to the computer is not ideal. Scanning or photocopying a book for scanning introduces some severe distortions to the image that are not captured by our noise model, namely edge effects. The photocopy of the page of a book will show the baselines "curving" near the book binding.

The source model generated horizontal baselines, but the observed image shows this is far from being true. If we assume that the images of each character are not significantly distorted, then we can simply reconcile this observation by assuming the model produced the textline along a baseline that was not horizontal, but rather skewed/curved near the edge of the page.

We will identify a given baseline i by its endpoints and by an equation describing its path between these two points. The vector to the origin of the baseline will be denoted by $\vec{B}_{o_i}$, and the vector to the endpoint of the baseline by $\vec{B}_{f_i}$. The path of a baseline can be described by a pair of parametric equations $x(t) = g_i(t)$ and $y(t) = h_i(t)$ in the interval $0 \leq t \leq T_i$ such that $(x(0), y(0)) = \vec{B}_{o_i}$ and $(x(T_i), y(T_i)) = \vec{B}_{f_i}$. In this thesis, we will assume that baselines are straight lines, with possibly nonzero slope, so that we don't have to worry about the specifics of the particular functions $g_i(t)$ and $h_i(t)$ for each baseline. Figure 4-1 shows an example of a straight baseline and its corresponding endpoint vectors $\vec{B}_o$ and $\vec{B}_f$.

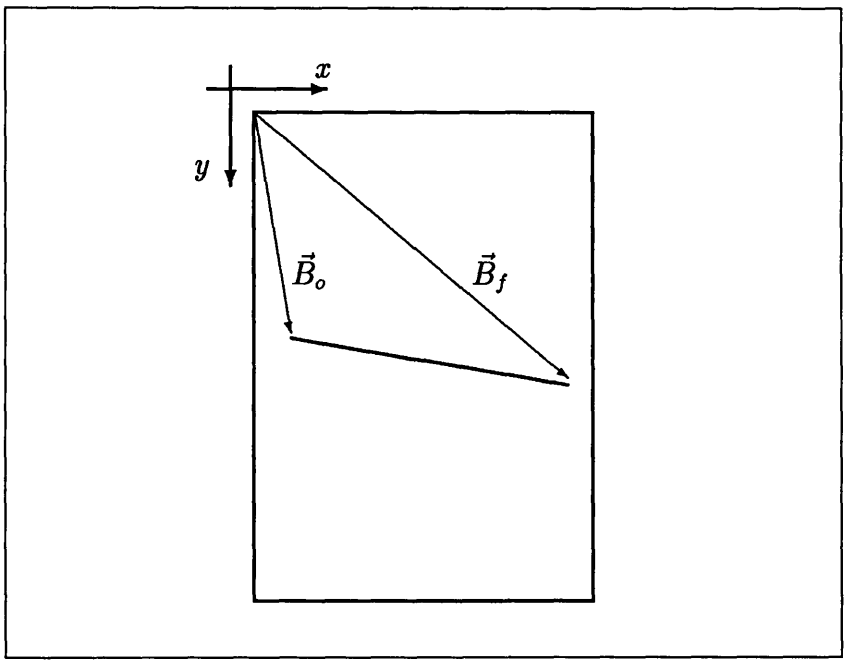


Figure 4-1: Baseline and Endpoint Vectors on Image Plane

4.2.2 Line Model

A line transcription is a sequence of character labels corresponding to the observed sequence of imaged characters along a given baseline. Clearly the transcription carries structural information about the placement of characters along that baseline. An example of a line transcription is `The cat walks through walls`. This example gives us the sequence of imaged templates along the baseline; it tells us that the templates for "T", "h", "e", " ", "c", "a", "t", " ", "w", "a", etc. are imaged from left to right along the same baseline. It must be stressed that the transcription does not tell us anything else about what we see on the image. It does not tell us, for example, that the intercharacter spacing is smaller than the "space" character; one could imagine, for example, some strange font where the intercharacter spacing is larger than the width of the space character. The transcription does not tell us anything about the size of the characters either; it could be that upper-case letters are smaller and skinnier than lower-case letters. The transcription does not tell us anything about the shape of the characters; imagine the same sentence written in

very different fonts, from Times-Roman to Mistral to ZapfChancery.

A transcription is simply an ordered sequence of labels with a one-to-one mapping to a sequence of observed image symbols that represent characters in a written language. The only restriction is that the imaged characters follow the sidebearing model for character shape and position.

Let us generalize and refer to a document transcription as a set of line transcriptions $\{L_1, L_2, \ldots, L_N\}$, where each line transcription L_i corresponds to one of N observed lines of text. We will further refer to a line transcription as a sequence of n observed symbols $L = \beta_1\beta_2 \ldots \beta_n$, where $\beta_i \in \{1, \ldots, T\}$ is a symbol corresponding to one of the possible imaging templates, including the template for the space character, as described in section 2.1.

Given a transcription L and a Markov source, we could trace a complete path π through the source that runs through all transitions that produced the imaged templates corresponding to the observed sequence $L = \beta_1\beta_2 \ldots \beta_n$. Given this path we could then find the origins of all the imaged characters, as we did in the development of the CTE procedure, and estimate the templates. Unfortunately, the path π used to generate the sequence of imaged characters is not unique. As mentioned in section 2.1, some of the transitions in the source have null templates. A path that includes these transitions will not affect the observation sequence, but will affect the layout of the observed characters on the page.[1]

Given a Markov source then, the union of all complete paths π through that source that yield the observed sequence $\beta_1\beta_2 \ldots \beta_n$ will result in the model used to generate any textline image with that observation sequence. We will refer to this model as the line model. The line model used to generate an observed sequence $\beta_1\beta_2 \ldots \beta_n$ is shown in figure 4-2.

The left-most state in the line model is the initial state n_I, and the right-most state is the final trap state n_F. It is called a left-to-right model because it has a distinct temporal structure in which transitions must always proceed to the right. This form of the model is appropriate for written text because the progressive nature of the state

[1]Remember that transitions with null templates carry layout information in their attribute $\vec{\Delta}_t$.

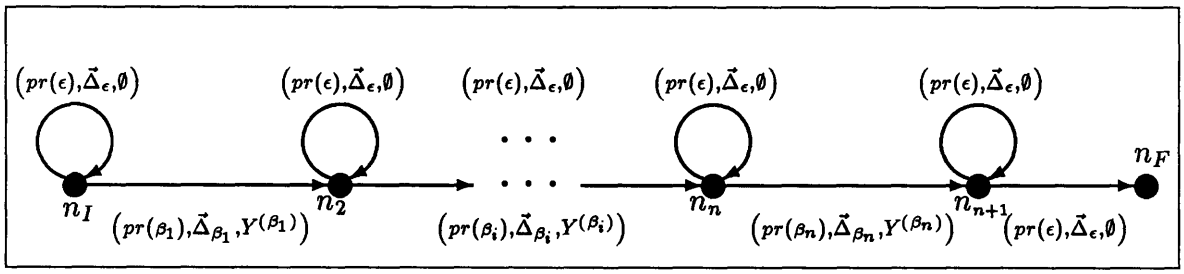


Figure 4-2: Line Model

sequence is rather unambiguous, the observed character images will follow the order in which the observation sequence is given. This model is also non-ergodic, which means that the state of the chain will always move to the right and will eventually find itself in the final state with probability one. The implication of this is that any complete path through this model will always generate an imaged character sequence corresponding to the observed sequence $\beta_1\beta_2\ldots\beta_n$, which is exactly what is given to us by the transcription.

As shown in figure 4-2, the cyclic transitions at every state have null templates, and a setwidth vector $\vec{\Delta}_\epsilon = (1,0)$ that is a single unitary pixel shift. Every other transition has an imaging template, $Y^{(\beta_i)}$ (possibly including the template for the space character), and the associated setwidth vector for that template, $\vec{\Delta}_{\beta_i}$. The line model incorporates within itself a very intuitive notion of how a line of text is imaged. Given a baseline, the imager positions itself at the beginning of that baseline. [2] As the chain evolves out of the initial state, each transition tells the imager exactly what to do. If a cyclic transition to the same state occurs, shown in the figure as circular arrows, the imager simply moves one pixel to the right along the baseline. If a transition to a different state occurs, shown in the figure as horizontal arrows, the imager takes the template corresponding to that transition, $Y^{(\beta_i)}$, positions the origin of the template at its current position along the baseline, and paints the image of that template on the page. The imager then updates its position by moving a

[2]The beginning of a baseline is defined as the leftmost point on the image plane that lies on that baseline. This assumes that $\Delta_y = 0$ and $\Delta_x > 0$ so that a line of text is generated from left to right along that baseline.

distance given by the setwidth vector of that template, $\vec{\Delta}_{\beta_i}$, along the baseline. The unitary setwidth vector of cyclic transitions, $\vec{\Delta}_{\epsilon}$, is used for fine adjustments of the intercharacter spacing that is greater than the assigned setwidths.

After the last template $Y^{(\beta_n)}$ has been imaged, the chain continues to evolve through the last cyclic transition; this is to allow the imager to finish its path through the baseline. The last transition to the final trap state n_F indicates that the imager has reached the end of the baseline, at position $\vec{B}_f$ on the image plane, and the text line is finished.

Given a complete path through the line model, we obtain the set of character origins at every location on the image where the path through the chain moves through the horizontally shown transitions labeled β_i; the origin is located at every position on the image where a new template is imaged.

4.2.3 Page Model Using Separable Sources

As described in [19], a general source model does not take advantage of the hierarchical organization in the layout or format of a typical document page. Due to this hierarchical layout, the analysis/decoding of one portion of the page can be completely separated from the analysis/decoding of another portion of the page. [19, 13] introduce *separable models* to capture this hierarchical layout information.

A separable model consists of a number of image sources, organized in a hierarchy of parent-child relations, corresponding to the hierarchy of the layout. Separable models have two features that make them very useful: locality, and positional constraints.

- **Locality:** A child source covers a smaller region of the page compared to its parent. Technically, a child source will cover known smaller areas than are allowed for the parent source.

- **Positional Constraints:** A parent source represents a coarse view of the layout of the page. Constraints on the possible positions of a parent source on the image plane are small.

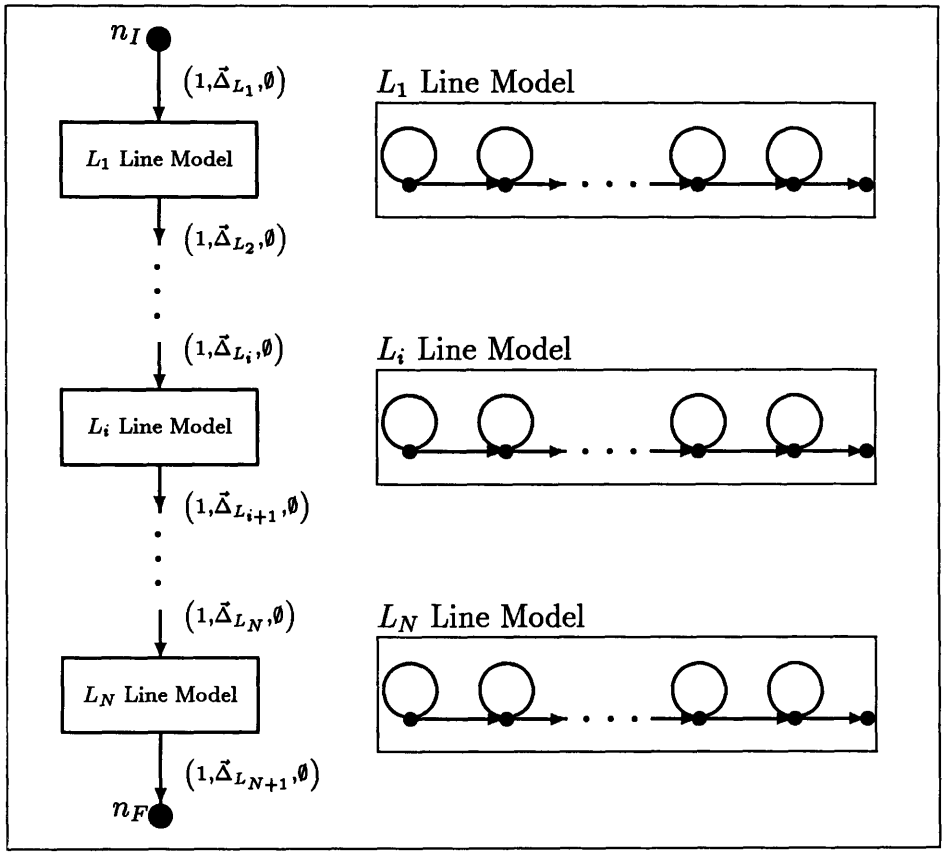


Figure 4-3: Page Model Given Baselines and Transcription

Following the ideas in [19], the separable model for an observed image given the baselines and transcription is shown in figure 4-3. In this model we assume that the imaging procedure of every line of text is independent of the imaging of every other line of text. This independence allows us to "separate" the imaging procedure of the entire page into smaller imaging units for each line.

In this model, the parent source deals with vertical motion down the page from baseline to baseline, and the child sources deal with horizontal motion along each baseline. We define the vector displacement from the endpoint of the $(i-1)$th baseline to the origin of the ith baseline as $\vec{\Delta}_{L_i} = \vec{B}_{o_i} - \vec{B}_{f_{i-1}}$ Note that if a line model starts at image location $\vec{B}_o$, corresponding to the origin of its baseline, then by definition we are guaranteed that any complete path through the line model will finish at image location $\vec{B}_f$. With this we can now associate with each transition in the parent

model a displacement vector that takes the imager to the beginning of the baseline corresponding to the next line model. For convenience we define $\vec{B}_{f_0} = \vec{0}$ and $\vec{B}_{o_{N+1}} = (W, H)$, so that the first transition in the parent model takes us to the beginning of the first baseline, and the final transition in the parent model takes us to the end corner of the image plane.

4.3 Alignment Algorithm – Modified Viterbi

One of the assumptions that allowed us to generate a page model by using separable models was the fact that the hierarchical layout of a document made one section of the page independent from other sections. The advantage of this is that the analysis/decoding of one portion of the page has no effect on the analysis/decoding of another portion.

In the page model for images whose transcription and baselines are known, the "decomposition" is based on the notion that the imaging of one line of text is independent from the imaging of other lines of text. Following the work in [19], each child source is simply defined to cover the imaging procedure of text along a unique baseline, whereas the parent source makes sure that every child is visited so that every observed line of text is imaged.

As shown in [11], a Viterbi algorithm solves the recurrence relations required to find the best path through a source that produces the MAP estimate of an observed image. As a direct result of the independence between child sources, the decoding of every textline is independent from the decoding of every other baseline, so that in order to maximize the path through the parent source we can equivalently maximize the path through each one of the line sources individually. This means that in order to decode an image where baselines and transcription are known, we can simply run the decoder along each baseline using the line model for that baseline.

The relation in (2.28) is the general recurrence relation for MAP decoding of an unknown image. In the case where we are trying to decode along a given baseline using the line model shown in figure 4-2, the recurrence relation in (2.28) now takes

on the following form

$$\mathcal{L}(n_{i+1}, \vec{x}) = \max\Big\{\mathcal{L}(n_{i+1}, \vec{x} - \vec{\Delta}_{\epsilon}) + \log pr(\epsilon), \\ \mathcal{L}(n_i, \vec{x} - \vec{\Delta}_{\beta_i}) + \mathcal{L}(\mathcal{I}|Y^{(\beta_i)}[\vec{x} - \vec{\Delta}_{\beta_i}]) + \log pr(\beta_i)\Big\} \quad (4.1)$$

where n_i and n_{i+1} are variables representing states in the line source. The form of the line model constrains transitions into state n_{i+1} to come from either state n_i or state n_{i+1} itself. This is why the maximization in the recurrence only has two terms.

The maximization on (4.1) can be easily obtained by using a Viterbi algorithm. We will modify this Viterbi algorithm to return the set of character origins that yield the MAP sequence of imaged characters along a baseline. The formal steps for this modified Viterbi, from now on referred to as *Alignment Procedure*, are shown in figure 4-4. In the Alignment algorithm there are $n + 1$ states, corresponding to the number of templates plus a transition to the final state. The algorithm runs through X points along the path of the baseline; the origin of the baseline is indicated by $x = 1$ and the endpoint is indicated by $x = X$.

A trellis structure efficiently implements the computation. The alignment procedure does a maximization over the individually most likely state of any sequence. The solution determines the most likely position of every template across the baseline following the dependencies in the observation sequence. Notice the use of the setwidth information Δ_{β_i} at every state, which enforces the sidebearing model rules for positioning. A maximization over a single unitary pixel jump is done in order to allow some flexibility in the interspace between characters that might be greater than the assigned setwidth.

The alignment traces through the states in reverse order starting from the last observation. To insure that the alignment procedure gives the most likely sequence using *all* the templates in the sequence, the first element in the sequence is initialized with a large initial score of M. This is similar to the "big M" method used in optimization techniques. This insures the path through the trelllis is "pulled-back" in the backward routine through all the elements (templates) in the sequence.

```
    begin
1.    for x = 1,...,X do
        begin
2.        L(n_1, x) = +M
3.        for j = 1,...,n do
4.          δ_j(x) = 0
        end
5.    for x = 1,...,X do
6.      for j = 1,...,n do
          begin
7.          let S_ε = L(n_{j+1}, x − 1) + log pr(ε)
8.          let S_β = L(n_j, x − Δ_{β_j}) + L̃(I|Y^{(β_j)}[x − Δ_{β_j}]) + log pr(β_j)
9.          if S_β > S_ε then
10.           δ_j(x − Δ_{β_j}) = +1
11.         L(n_{j+1}, x) = max{S_ε, S_β}
12.         Ψ_{j+1}(x) = argmax_{1≤k≤x−Δ_{β_j} | δ_j(k)=+1} L(n_j, k)
          end
13.   let x*_{n+1} = X
14.   for j = n, n − 1,...,1 do
15.     x*_j = Ψ_{j+1}(x*_{j+1})
    end
```

Figure 4-4: Alignment Algorithm – Modified Viterbi

The alignment procedure is set up to run along a given baseline. In the actual implementation, however, the baselines are not known exactly. The baselines are estimated as straight lines (possibly with non-zero slope) using some image processing techniques. Visual inspection of the example in figure 4-5 shows that individual characters frequently lie above or below the estimated baseline, for example "F" in `Figure` in the second line or "T" in `The` in the third line. We incorporate into the alignment procedure a "jitter" that checks for character locations within a small region surrounding the baseline at every x location[3]

$$\tilde{\mathcal{L}}(\mathcal{I}|Y^{(\beta_j)}[x]) = \max_{-L\leq y\leq +L} \mathcal{L}(\mathcal{I}|Y^{(\beta_j)}[(x,y)]). \tag{4.2}$$

[3]This assumes that a straight baseline is a good first order approximation of the actual baseline of a document.

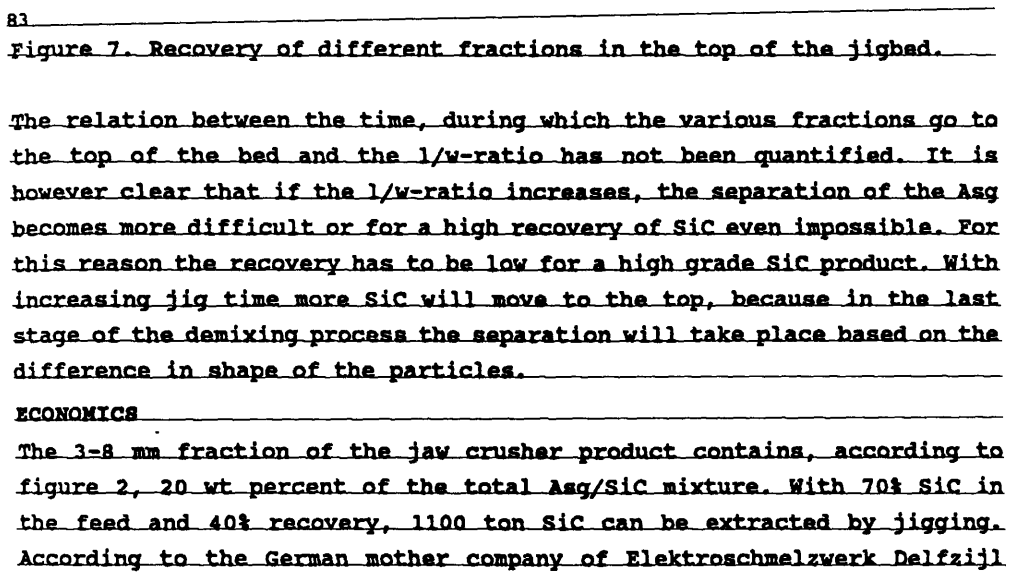
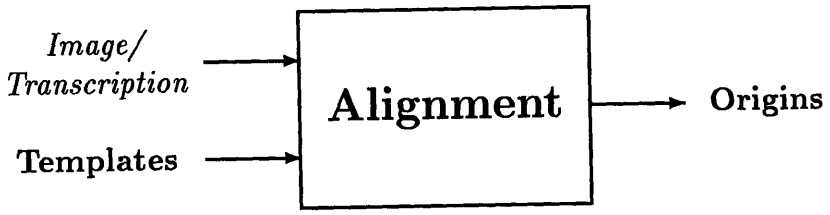


Figure 4-5: Typed Document Estimated Baselines

The "jitter" moves the template at any location along the baseline in the y-direction by some amount bounded by $\pm L$, and records the best score. The value of L is hand adjusted depending on how good the estimated baselines are with respect to the origins of the observed characters. For the baselines in figure 4-5 a value of $L = 5$ is sufficient to insure that all the origins are covered by the jitter.

Conceptually, the development of the alignment procedure is a module that, given an observed image, the baselines, text transcription, and set of templates, returns the set of origins in the observed image. The alignment procedure is illustrated below,



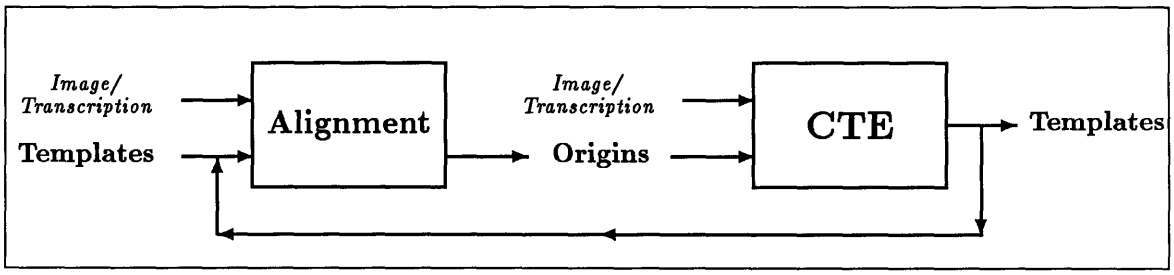


Figure 4-6: Template Estimation Procedure Using Alignment and CTE Modules

4.4 Iterative Template Estimation Procedure Using Alignment and CTE Modules

There we have it, the alignment procedure is a modified decoder that is able to determine the origin locations of an ordered set of templates along baselines in an image in the MAP sense. It appears that this procedure solves the problem we posed at the beginning of this chapter, to develop a procedure that is able to determine the origins of characters along a baseline in an image so we can then use the CTE procedure to estimate the templates. Unfortunately, the flaw of the alignment procedure with respect to our problem is that it assumes that we have initially the set of templates and their setwidths that we are trying to estimate!

Using the alignment and CTE modules already developed, we establish the iterative procedure shown in figure 4-6. As the block diagram shows, we use an initial set of templates to find an initial set of origins on the image using the alignment procedure. This set of origins is used to estimate a new set of templates using the CTE procedure. The iteration arises from using this new set of templats to re-align the image and re-estimate the set of templates.

We will first argue that the iterative scheme in figure 4-6 for estimating templates converges to some local maximum. Then we detail the iterative procedure by describing the initial set of templates used in the loop, and the estimation routine for character setwidths.

4.4.1 Convergence of $\mathcal{Y}$

Before we provide an argument for this convergence, we will make several assumptions.

1. *In the alignment procedure, the a-priori probability of a path is negligible, so that* $\mathcal{L}(\mathcal{I}, \pi) \approx \mathcal{L}(\mathcal{I}|\mathcal{I}')$.

   This assumption simply states that in MAP decoding, the a-priori probability of a path is negligible compared to the matching score of the reconstructed image with the observed image. In other words, it doesn't matter what path the source takes, so long as the clean image generated by the output closely matches the observed image.

   Simply observe that MAP decoding maximizes $\mathcal{L}(\mathcal{I}, \pi) = \mathcal{L}(\mathcal{I}|\mathcal{I}') + \log \Pr(\pi)$, and we can rewrite the likelihood score as

$$\begin{aligned} \mathcal{L}(\mathcal{I}|\mathcal{I}') &= \|\mathcal{I} \bigwedge \mathcal{I}'\| \log\left(\tfrac{\alpha_0\alpha_1}{(1-\alpha_0)(1-\alpha_1)}\right) - \|\mathcal{I}'\| \log\left(\tfrac{\alpha_0}{1-\alpha_1}\right) && (4.3) \\ &= \|\mathcal{I} \bigwedge \mathcal{I}'\| \log\left(\tfrac{\alpha_1}{1-\alpha_0}\right) + (\|\mathcal{I} \bigwedge \mathcal{I}'\| - \|\mathcal{I}'\|) \log\left(\tfrac{\alpha_0}{1-\alpha_1}\right). && (4.4) \end{aligned}$$

   When the reconstructed image matches the observed image very closely so that $\mathcal{I} \approx \mathcal{I}'$, the likelihood score above will be dominated by $\|\mathcal{I} \bigwedge \mathcal{I}'\|$. For large images, this term grows as the number of black pixels in the observed image, and will generally be much larger than $|\log \Pr(\pi)|$.

2. *Given a set of labeled origins, the CTE procedure yields the optimal set of templates that maximize* $\mathcal{L}(\mathcal{I}|\mathcal{I}')$.

   This assumption is based on experimental results. In the examples presented earlier, the greedy+refinement procedure yielded a set of templates that were aesthetically pleasing and that generated a very good reconstructed image. Based on these observations, the assumption then is that the CTE procedure maximizes $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{CTE})$.

3. *The alignment procedure maximizes* $\mathcal{L}(\mathcal{I}|\mathcal{I}')$ *while enforcing the non-overlapping criterion, or at least while minimizing the amount of overlap between the tem-*

plates in the reconstructed clean image.

The alignment procedure we presented in section 4.3 does not enforce the notion of non-overlapping templates. In its maximization, it does not record the level of overlap between templates as it calculates $\mathcal{L}(\mathcal{I}|Y^{(f(t))}[\vec{x}])$ for each distinct template. The maximization implicitly assumes the templates will be disjoint, regardless of where each one is positioned on the image. This will generally be a very difficult goal to achieve; if the templates are suboptimal then an arrangement across the page that allows overlap between them might yield a better match for the reconstructed image.

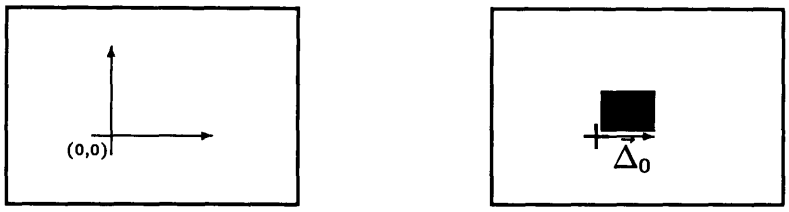
Since our goal is to estimate a set of imaged-disjoint templates, we will relax this requirement on the alignment procedure and assume that it tries to minimize the amount of overlap between templates as it maximizes $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{align})$.

Given these assumptions, let us summarize what we have:

- Given a set of templates, the alignment procedure finds the set of origins that maximize $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{align})$.
- Given a set of origins, the CTE procedure finds the set of disjoint templates that maximize $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{CTE})$.

Intuitively, the alignment procedure tries to reconstruct the best image **using the templates is has.** If the alignment observed the non-overlapping criterion, its set of origins would correspond to the set of locations where one could place the templates such that they did not overlap and maximized $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{align})$.

On the other hand, the refinement procedure derives the best set of non-overlapping templates for the origins it is given. Given the set of origins derived from the alignment procedure, the refinement procedure would then either derive the same set of templates that were used for alignment, since they were assumed to not overlap in the first place, or it would find a new set of templates that still did not overlap but with a higher likelihood score. It could not do any more worse because the original set of non-overlapping templates place a bounding score on $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{CTE})$.



(a) (b)

Figure 4-7: Template Image Canvass. (a) Local Coordinate System. (b) Initial Template. Origin is indicated by a cross, initial setwidth vector is $\vec{\Delta}_0$.

The alignment procedure returns the origins that build the best image using the templates it has. Applying the CTE procedure to these origins would then result in a set of templates that can do no worse than the templates used for alignment:

$$\mathcal{L}(\mathcal{I}|\mathcal{I}'_{CTE}) \geq \mathcal{L}(\mathcal{I}|\mathcal{I}'_{align}). \tag{4.5}$$

If the templates were optimal at the origins given through alignment, the refinement procedure could do no better. But if they are not optimal, the CTE should find a better set.

Since $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{align}) \leq \mathcal{L}(\mathcal{I}|\mathcal{I}'_{CTE}) \leq \mathcal{L}(\mathcal{I}|\mathcal{I}' = \mathcal{I})$ by the argument above, we can set up an iterative procedure where we go back and forth between estimating a set of origins and templates, as shown in figure 4-6. The convergence of this iteration occurs when $\mathcal{L}(\mathcal{I}|\mathcal{I}'_{align}) = \mathcal{L}(\mathcal{I}|\mathcal{I}'_{CTE})$.

4.4.2 Initial Templates

Figure 4-7a shows an example of the template canvass and its associated local coordinate system. Figure 4-7b shows an example of the initial templates used in the iterative procedure.

In section 2.1.1 we discussed the idealization of a template as the image of a character in a canvass spanning the x-y plane. The region of interest to us in such an idealized template is the area containing the support of the character. We chose the template canvass as a rectangular region with a local coordinate system as shown, large enough to completely fit the image of any of the character glyphs on an observed

image.

In the sidebearing model a character template is endowed with an origin, a character image positioned relative to that origin, a setwidth vector, and a few other metrics that precisely define its placement in a document along side all other characters. In this thesis, we only focus on estimating the character image and setwidth vector of each template.

An issue that arises is the position of each estimated character image relative to the origin of the template canvass. This is a problem we do not encounter when the origins are known, we simply line up the origin of the template canvass with the glyph origin on the image and extract as an observed instance of that character the image area covered by the template canvass. All of these observed instances are then guaranteed to be origin-aligned with the template canvass.

The problem now is that the template canvass has a *fixed* local origin, and the setwidth vectors are defined relative to this origin, but the character image in each template can be positioned *anywhere* on the canvass. Experiments show that in the iterative procedure the templates eventually "latch on" to the image of the character, but the origin of the canvass does not necessarily line up with the true origin of the character image. Arbitrary positions of a character image with respect to the local origin can affect the alignment procedure very drastically. Consider the case illustrated in figure 4-8a, where the image for the character "a" is positioned far to the left of the local origin of its canvass, and the image for the character "t" is positioned far to the right of the local origin. Figure 4-8b shows the image of the word at and its baseline. We expect the alignment routine to line up the templates of the "a" and "t" as shown in figure 4-8c, since this yields the best match for template reconstructed version of the word at. Unfortunately, this alignment is invalid because the origin of the "a" lies to the right of the origin of the "t", which does not follow the line model. But if the alignment in figure 4-8c can't happen, then how did the templates in figure 4-8a arise in the first place? The reason is that templates are estimated from other samples as well, where alignment is possible with these templates.

This problem is not very serious if the true origin of a character latches near the

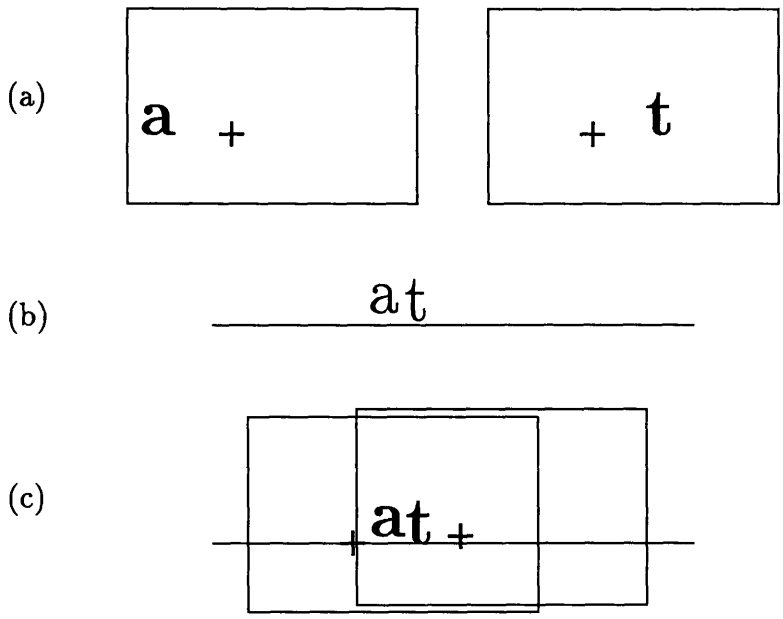


Figure 4-8: Invalid Alignment Using Templates with Arbitrary Image Positions. (a) Character images arbitrarily positioned with respect to origin. (b) Observed textline image. (c) High scoring alignment of templates on textline image.

local origin of the template canvass. The extra slack can be accounted for with small variations in the setwidth of each template. Once we have a final set of templates, there are approaches we can take to correct the misplacement of each character with respect to its origin, namely by considering the shift of the true origin of the character from the local origin of the canvass, and performing a minimization routine on the square of the difference between these measures. This approach is detailed in [10].

The reason for using as an initial template a black box at the origin of the template canvass is such that the first alignment and subsequent ones "latch on" correctly to the character glyphs in locations on the image that are close to the true origin of each glyph. In an oversimplified description of the alignment procedure, each template is positioned at the location on the baseline where there is a very high matching score between the template and the observed image. Using such an initial template, the first alignment will return a set of origins where the extracted images will contain a pattern of black pixels that repeats itself at all locations where the template was positioned, at the location where the inner "black box" maps to. This is because

the score $\mathcal{L}(\mathcal{I}|Y^{(f(t))}[\vec{x}])$ is highest where the dark block lines up with observed black images, which hopefully correspond to some portion of the proper character at their true origin.

The experimental results in figure 4-10 show how the images of each character effectively "latched" near the canvass origin. In particular, notice the variations of character placement about this origin in the template for the "G", where the true origin lies to the left of the local origin, and the template for the ",", where the true origin lies to the right of the local origin. In the case of other text images, however, we have found templates with origins that are very far from the true origin for some characters. This will be illustrated and discussed in section 4.5.

4.4.3 Setwidth Estimation

The templates start with an initial setwidth vector, $\vec{\Delta}_0$, that runs through the length of the initial dark block as shown in figure 4-7. One must be careful to choose the length of $\vec{\Delta}_0$ smaller than the smallest setwidth we can possibly encounter in the training data. We make the assumption that the initial templates and continued set of estimated templates latch on correctly to the true origin of every character at the local origin of the template canvass, so that estimating the setwidth only involves "stretching" the vector $\vec{\Delta}_0$. This assumption is partly based on experimental results, where the character images are correctly positioned on the template canvass (with small variations) with respect to its local origin. Without this assumption, the setwidth estimation routine would need to consider stretching of the setwidth vector as well as "shifting" of the origin, which is a much harder problem, and one we will partly address in section 4.5.

The alignment procedure can be thought of as an elaborate pattern matcher if we ignore the a-priori probabilities of a transition through the source: given a set of templates, the forward Viterbi runs an autocorrelation of every template on the textline image, and the backward Viterbi simply selects the "peaks" of these autocorrelations as an ordered sequence that observes the sidebearing model. The alignment procedure is very sensitive to any overestimation of the setwidth vectors. The backward

viterbi picks the highest scoring ordered sequence of template positions such that the origins of two consecutively imaged characters lie at least as far apart as the length of the setwidth vector of the character to the left. If a setwidth is overestimated, this will result in the alignment procedure to miss the right alignment and throw off all the estimated templates at the next iteration.

Experimental results show that without any modifications to the initial setwidth vector $\vec{\Delta}_0$, the character template images "congeal" to a state that is very close to their final form after a few iterations. Our template estimation procedure waits until this happens before it begins to modify the setwidths.

To avoid overestimating the setwidth, given the origin locations from the alignment procedure we use a lower bound on the setwidth estimate at each iteration. We choose the setwidth of a character c as the minimum distance from the origin of that character to the origin of every other character imaged to the right of it. If we refer to the origins on the alignment of a character c by $\mathcal{O}_{c_i}$, and to the origins on the alignment of all characters immediately to the right of c by $\mathcal{O}_{r_i}$, then the estimate of the setwidth of c at each iteration is defined by

$$\vec{\Delta}_c = \min_i [\mathcal{O}_{r_i} - \mathcal{O}_{c_i}] \tag{4.6}$$

which is a very conservative estimate for the setwidth. Note that the setwidths are strictly increasing, and that every setwidth estimate maintains the current alignment of templates on the observed image; the position of the templates with respect to one another on the page is not affected by new setwidth estimates.

4.4.4 Experimental Results

In this section we examine the results of the above procedure in three types of documents. The *Typed* document is one where all the characters have the same pitch. The *Old English* document is one where the characters have different sizes and widths, and the character images are not standard. The *Mistral* document is one where all the characters within a word are connected. The characters of the first two document

types are designed to be disjoint, while the Mistral document shows an example where the characters are designed to touch so that a word is most of the time one connected component. Each one of these documents displays different characteristics; we are interested in the behaviour of our estimation routine on each one of them.

In figures 4-10 and 4-11 of the Typed example, the templates were allowed to "congeal" for two iterations before the setwidth estimation routine (4.6) was implemented. By the fourth iteration the templates had obtained the observed form and did not change significantly in further iterations. The template reconstructed image in figure 4-11 shows a very close match with the original image in figure 3-3. The templates for the "F" and "T" are sparse because the alignment never really captured the image for them. Partly to blame is the fact that there are only two instances of each character, and they are both very far away from the estimated baselines and the jitter probably missed them. The templates for the "M", "N", and "O" are also very sparse because there are only one or two instances of each. In the reconstructed image, though, the word ECONOMICS can be perfectly well read. The templates are sparse because the estimation procedure assigned pixels arbitrarily between these three templates. As the example shows, this is one possible reassignment of pixels amongst these three templates that maximizes the fit between the reconstructed and observed images. The intuitive solution, of course, is one that assigns pixels from each separate component in the observed image to each corresponding template, as expected. Figure 4-11 looks better than figure 3-17 because the transcription was complete, i.e. all of the observed characters had labels, and the iterative routine latched on relatively close to all of the true origins, so that the reconstructed image was well aligned with the observed one.

In the Old English example, figures 4-12 through 4-17, the templates were allowed to "congeal" for six iterations before the setwidth estimation routine was implemented. By the tenth iteration the templates obtained the final form we see in figure 4-16. The template reconstruted image looks very impressive. The greedy templates at the tenth iteration illustrate very well the limits of the greedy algorithm; notice in particular the holes in most of the character templates whose corresponding

pixels were assigned to the template for the "e". The refined templates at the tenth iteration show very clearly the advantages of a local search algorithm over the solution given by greedy. The shapes of all the templates have been very well estimated, and the resulting setwidth vectors are also good approximations to the actual setwidths we would expect from the original templates. In some of the templates, such as the "C","P", "T" or "q", there are small artifacts around the proper image for each of the characters. These small pixel groups are due to the fact that there are very few instances of these characters on the image.

In the Mistral example, figures 4-18 through 4-22, the templates were allowed to "congeal" for six iterations before the setwidth estimation routine was implemented. By the eight iteration the templates obtained the final form we see. This last example serves to show that our assumption that the template canvass latches on to the origin of the characters is not always true. In this case, the origins for some of the capital letters are very far off. As a result, the estimated templates suffer from artifacts introduced by wrong alignment, namely the incomplete "o" and "r" templates as well as some white spots throughout the other templates. This example also serves to illustrate the need to fix this assumption. The following section examines an approach that resolves this.

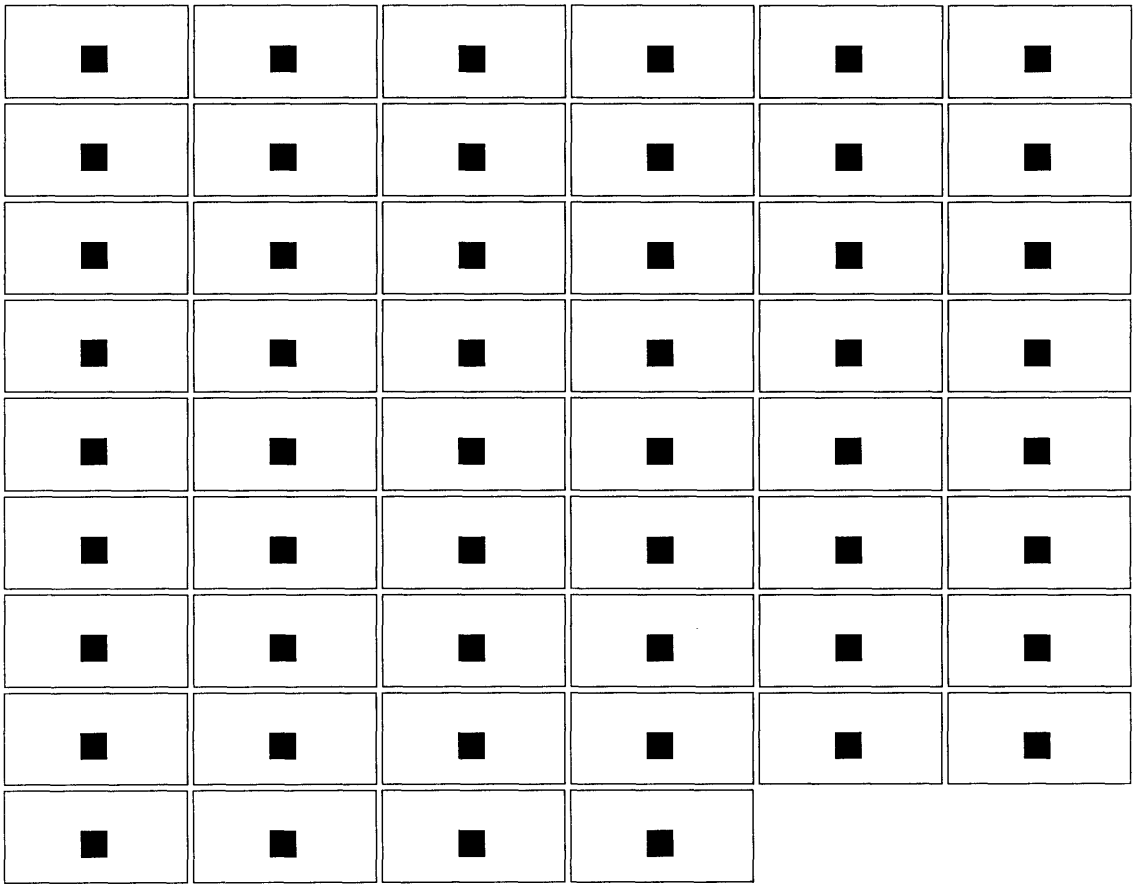


Figure 4-9: Typed Document Initial Templates

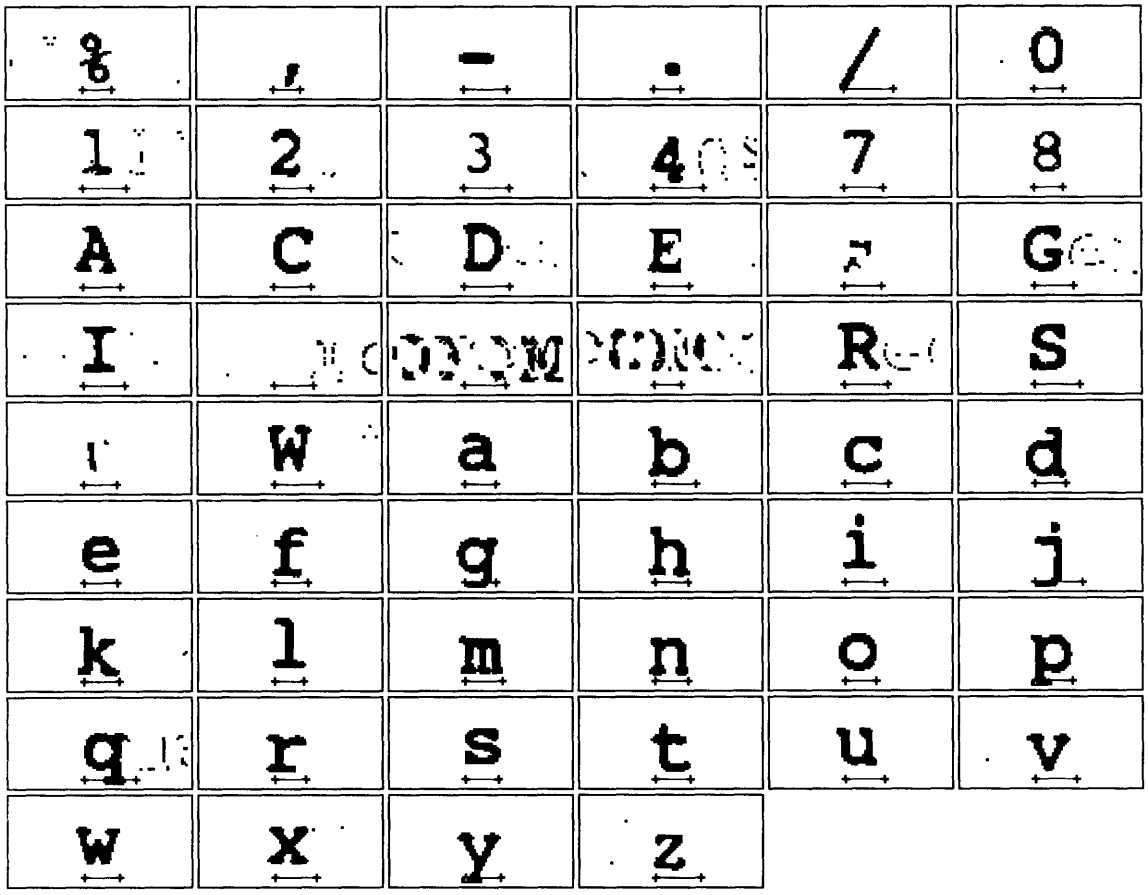


Figure 4-10: Typed Document Estimated Templates after 4 Iterations

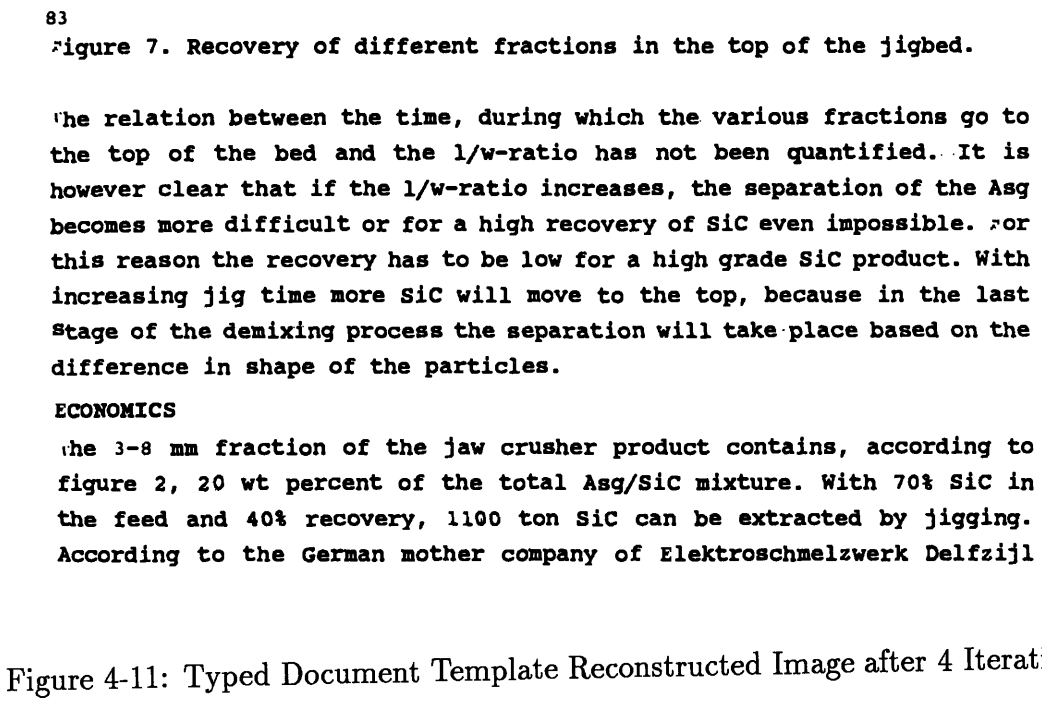


Figure 4-11: Typed Document Template Reconstructed Image after 4 Iterations

maye lyue and endure wythoute ayre.
¶ And by cause of his substancyall lyg
htnesse ayre is kyndly meuable and al-
so chaungable / and maye be coꝛrupd in
to contrary qualytees/ Therfoꝛe ofte ty-
mes he is chaungyd by vapoures of the
erthe and of the see/ ¶ foꝛ yf the vapouꝛ
stynke and is corrupte and venemouse/
the ayre is coꝛrupted and Infected to the
wyche suche pestilencyall vapour is me
delyd/ ¶ And yf smoke is resolupd and
compth of pure substaunce and clene. ¶
is of good sauour and smelle by Incoꝛ-
poracyon and medelynge wyth suche a
swete smoke the ayre receyuyth and ta-
kyth a qualitee that is frendly to kynde
¶ Also the ayre that biclyppyth vs is to
vs mooste prouffytable and necessary foꝛ
uedeof breth. and also foꝛ contynual fou
strynge and nourysshynge of the spyry-
tuell lyfe/ ¶ And yf the ayre is clere and
bryghte and clene/ thenne the humoures
and spyrytes shall be clere and bryghte/
And yf he is trowbly and myshy / humo
uꝛs shal be troꝛwblous. and spyrytes shal
be griete and thycke. and Infecte as Con
stancyne sayth and Philaretus also/
¶ And soo the ayre is ꝑ element of bodi
es and of spyrytes/ foꝛ ventynge of ayre
comynge to spyrytes is cause of amende
ment of cheyth. and of clensynge and of
purgacyon and of swagynge and lettin
ge humours that they be not brennyd /
¶ foꝛ ayre receyuyd and drawey by the
lounges to the herte. and by the herte to
all the body peuyth temperamente theꝛ-
to/ And so the ayre transposyth and cha
ungyth mooste the body / foꝛ he passyth
to the Inuer parties and to the spyrytes
And is medelyd wyth the substaunce of
cheyth whyche gyuey lyfe to the body/
¶ And soo yf that the ayre is pure clere

Figure 4-12: Old English Original Document

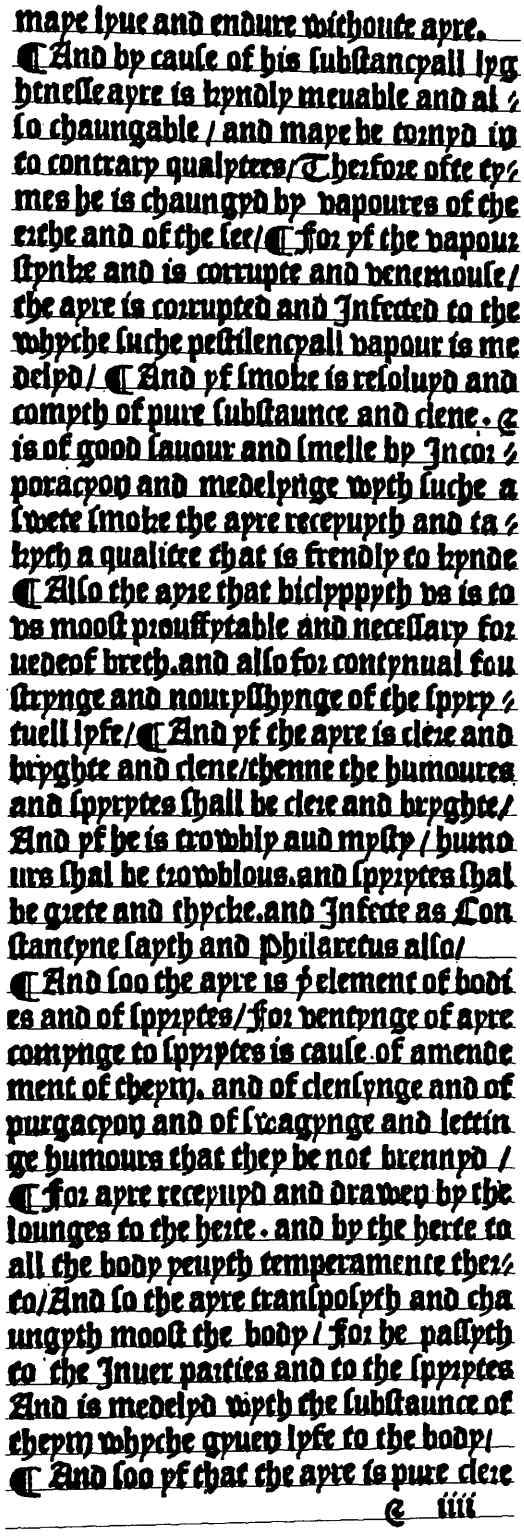

maye lyue and endure wythoute ayre.
¶ And by cause of his substancyall lyg
htnesse ayre is kyndly meuable and al ⁊
so chaungable / and maye be tornyd in
to contrary qualytees/ Therfore ofte ty⁊
mes he is chaungyd by vapoures of the
erthe and of the see/ ¶ For yf the vapour
stynke and is corrupte and venemouse/
the ayre is corrupted and Infected to the
wyche suche pestilencyall vapour is me
delyd/ ¶ And yf smoke is resolvyd and
compth of pure substaunce and clene. ⁊
is of good sauour and smelle by Incor ⁊
poracyon and medelynge wyth suche a
swete smoke the ayre receyuyth and ta ⁊
kyth a qualitee that is frendly to kynde
¶ Also the ayre that biclyppyth vs is to
vs mooste prouffytable and necessary for
uedeof breth.and also for contynual fou
strynge and nourysshynge of the spyry ⁊
tuell lyfe/ ¶ And yf the ayre is clere and
bryghte and clene/thenne the humoures
and spyrytes shall be clere and bryghte/
And yf he is trowbly and myxty / humo
urs shal be trowblous.and spyrytes shal
be grete and thycke.and Infecte as Con
stantyne sayth and Philaretus also/
¶ And soo the ayre is ꝑ element of bodi
es and of spyrytes/ For ventynge of ayre
comynge to spyrytes is cause of amende
ment of theym. and of clensynge and of
purgacyon and of swagynge and lettin
ge humours that they be not brennyd /
¶ For ayre receyuyd and drawen by the
lounges to the herte . and by the herte to
all the body peuyth temperamente ther⁊
to/ And so the ayre transposyth and cha
ungyth mooste the body / for he passyth
to the Inuer parties and to the spyrytes
And is medelyd wyth the substaunce of
theym wyche gyuen lyfe to the body/
¶ And soo yf that the ayre is pure clere

Figure 4-13: Old English Document Estimated Baselines

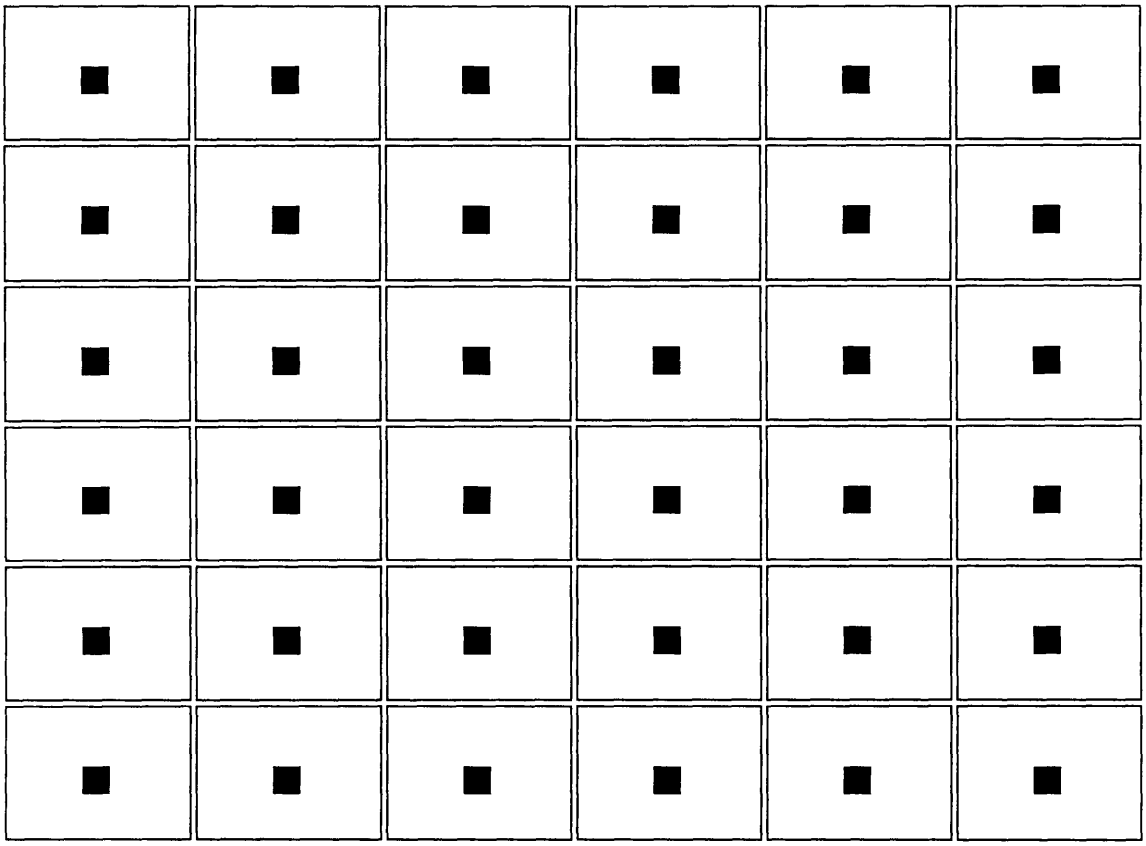


Figure 4-14: Old English Document Initial Templates

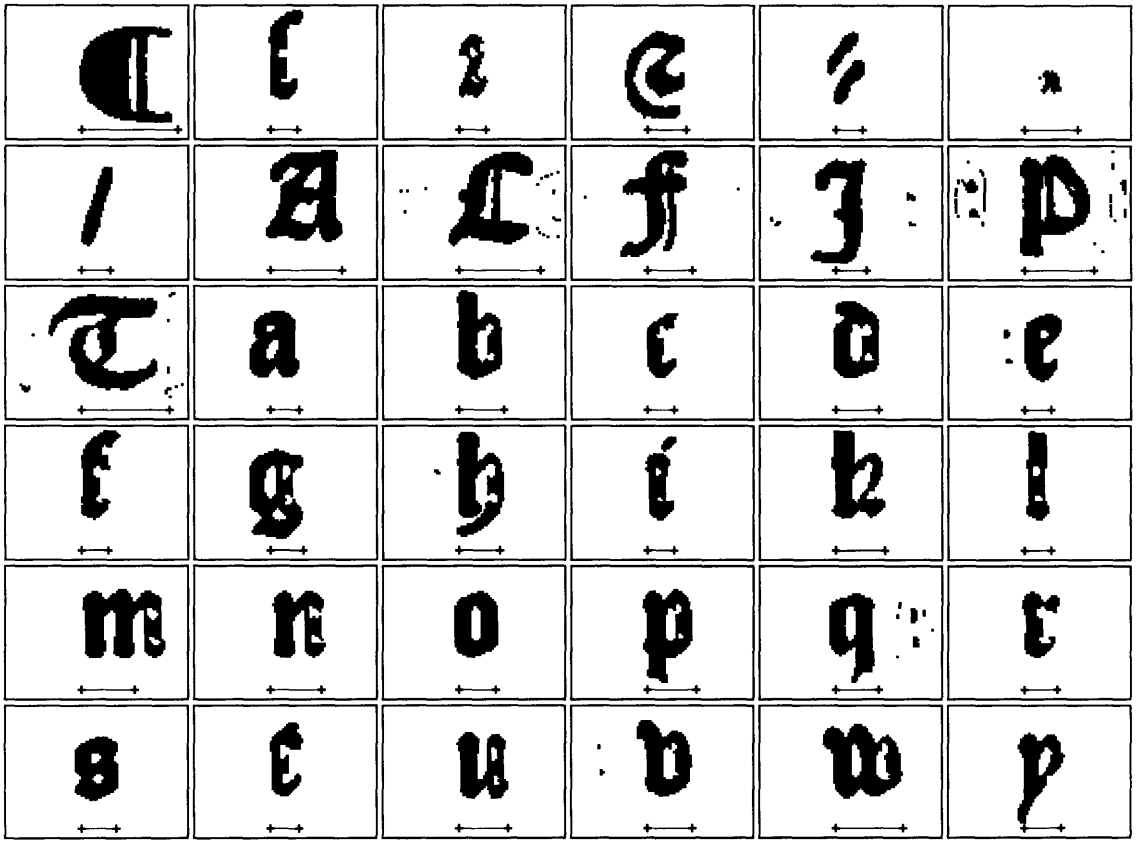


Figure 4-15: Old English Document Greedy Templates after 10 Iterations

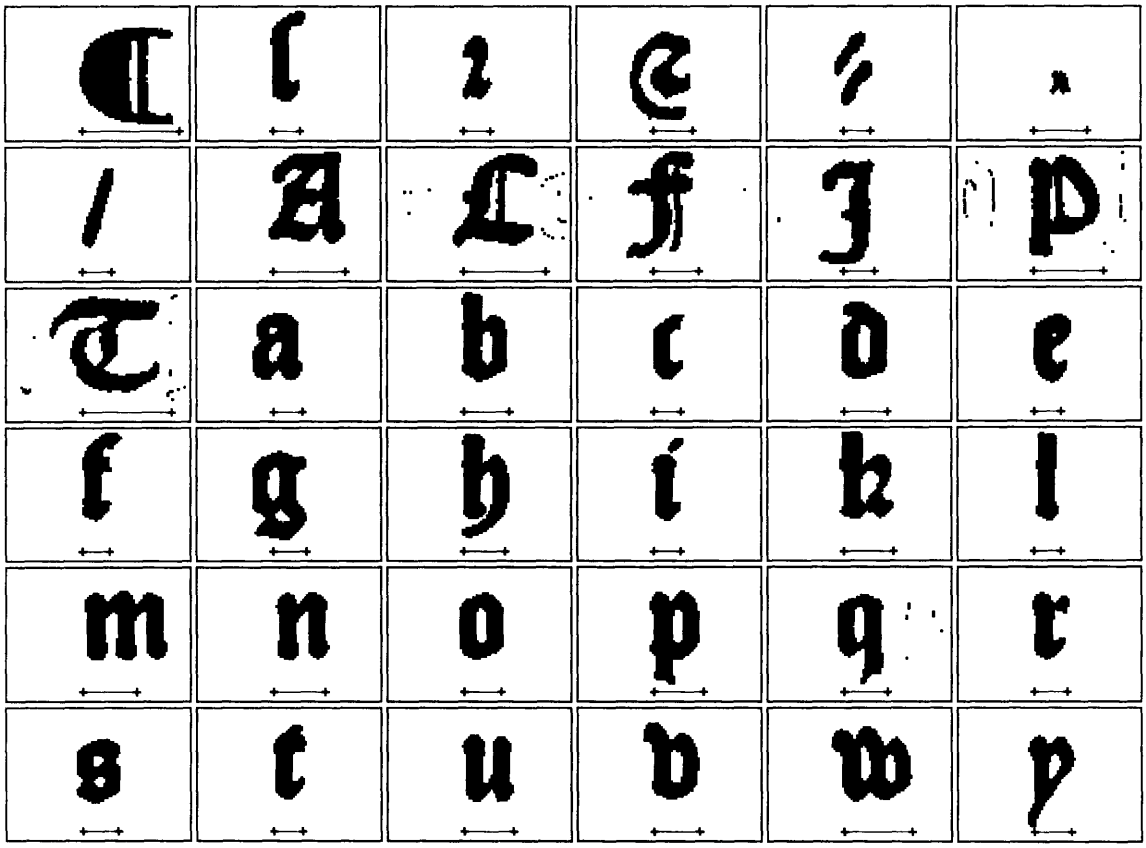
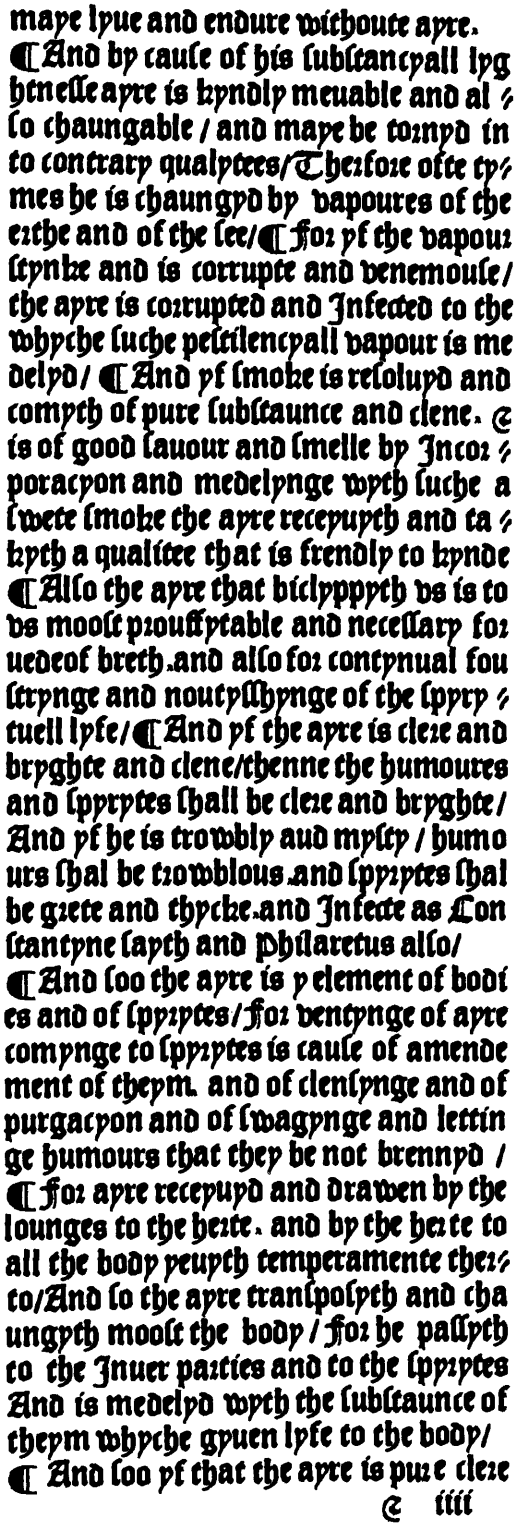


Figure 4-16: Old English Document Refined Templates after 10 Iterations


maye lyue and endure wythoute ayre.
¶ And by cause of his substancyall lyg
htnesse ayre is kyndly meuable and al ⁊
so chaungable / and maye be tornyd in
to contrary qualytees/Therfore ofte ty⁊
mes he is chaungyd by vapoures of the
erthe and of the see/¶ For yf the vapour
stynke and is corrupte and venemouse/
the ayre is corrupted and Infected to the
whyche suche pestilencyall vapour is me
delyd/ ¶ And yf smoke is resolupd and
compth of pure substaunce and clene. ℭ
is of good sauour and smelle by Incor⁊
poracyon and medelynge wyth suche a
swete smoke the ayre receyuyth and ta⁊
kyth a qualitee that is frendly to kynde
¶ Also the ayre that bisyppyth vs is to
vs moost prouffytable and necessary for
uedeof breth.and also for contynual fou
strynge and noutysshynge of the spyry ⁊
tuell lyfe/¶ And yf the ayre is clere and
bryghte and clene/thenne the humoures
and spyrytes shall be clere and bryghte/
And yf he is trowbly aud myxty / humo
urs shal be trowblous.and spyrytes shal
be grete and thycke.and Infecte as Con
stantyne sayth and Philaretus also/
¶ And soo the ayre is ꝑ element of bodi
es and of spyrytes/For ventynge of ayre
compnge to spyrytes is cause of amende
ment of theym. and of clensynge and of
purgacyon and of swagynge and lettin
ge humours that they be not brennyd /
¶ For ayre receyupd and drawen by the
lounges to the herte. and by the herte to
all the body yeuyth temperamente ther⁊
to/And so the ayre transpolyth and cha
ungyth moost the body / For he pallyth
to the Inuer parties and to the spyrytes
And is medelyd wyth the substaunce of
theym whyche gyuen lyfe to the body/
¶ And soo yf that the ayre is pure clere

ℭ iiii

Figure 4-17: Old English Template Reconstructed Image after 10 Iterations

maye lyue and endure withoute ayre. And by cause of his substancyall lyg
htnesse ayre is kyndly meuable and also chaungable / and maye be tornyd in
to contrary qualytees/Therfore ofte tymes he is chaungyd by vapoures of the
erthe and of the see/For yf the vapour stynke and is corrupte and venemouse/
the ayre is corrupted and Infected to the whyche suche pestilencyall vapour is me
delyd/ And yf smoke is resolwyd and comyth of pure substaunce and clene. &
is of good sauour and smelle by Incorporacyon and medelynge wyth suche a
swete smoke the ayre receyuyth and takyth a qualitee that is frendly to kynde
Also the ayre that bicluppyth us is to us moost prouffytable and necessary for
nedeof breth.and also for contynual fou strynge and nourysshynge of the spyry-
tuell lyfe/And yf the ayre is clere and bryghte and clene/thenne the humoures
and spyrytes shall be clere and bryghte/ And yf he is troubly and mysty / huma
urs shal be troublous.and spyrytes shal be grete and thycke.and Infecte as Con
stantyne sayth and Philaretus alsa/ And soo the ayre is y element of bodi
es and of spyrytes/For ventynge of ayre comynge to spyrytes is cause of amende
ment of theym. and of clensynge and of purgacyon and of swagynge and lettin
ge humours that they be not brennyd / For ayre receyuyd and drawen by the
lounges to the herte.and by the herte to all the body yeuyth temperamente ther-
to/And so the ayre transposyth and chaungyth moost the body / For he passyth
to the Inner parties and to the spyrytes And is medelyd wyth the substaunce of
theym whyche gyuen lyfe to the body/ And soo yf that the ayre is pure clere
& iiii

Figure 4-18: Mistral Original Document

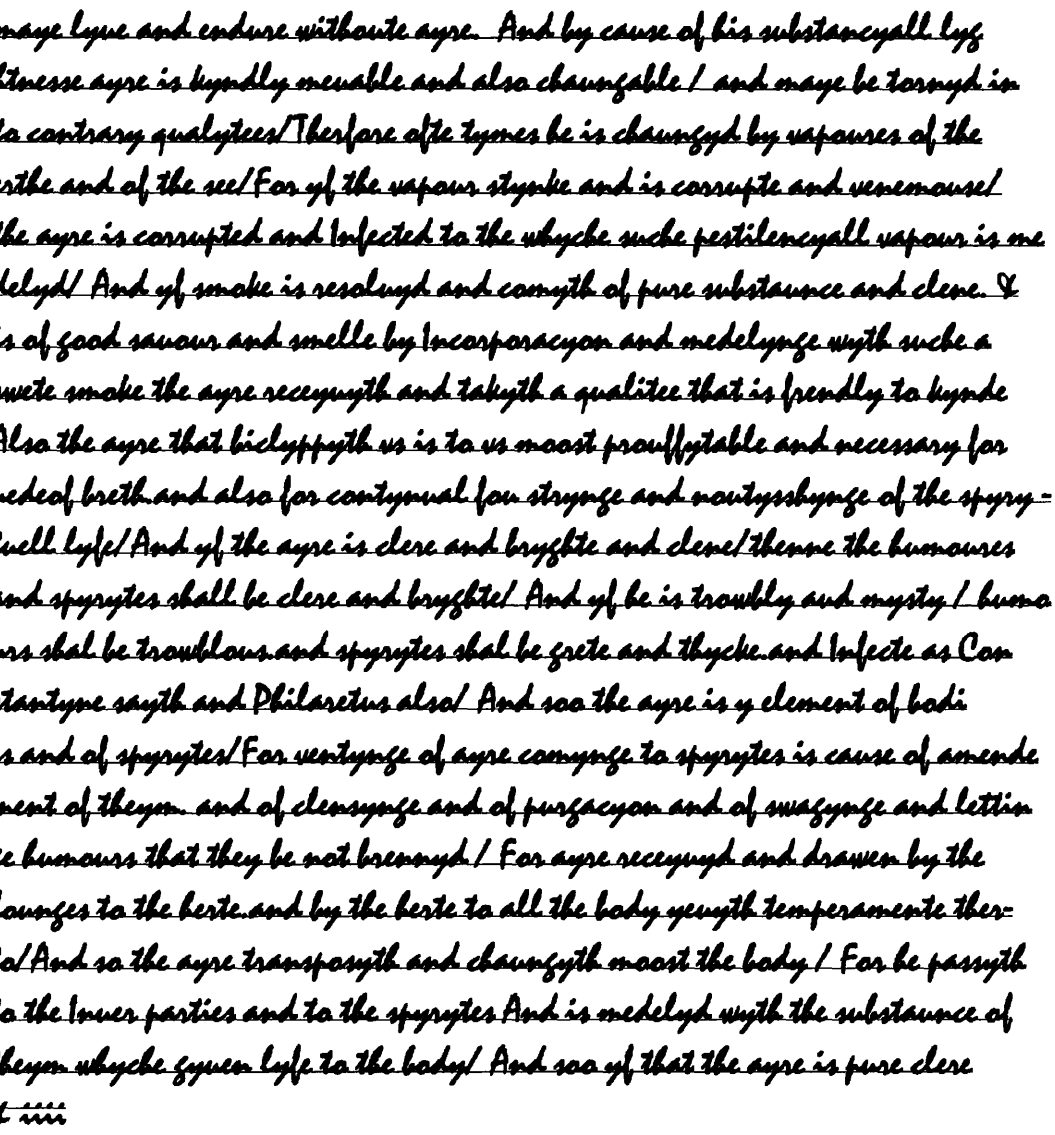


Figure 4-19: Mistral Document Estimated Baselines

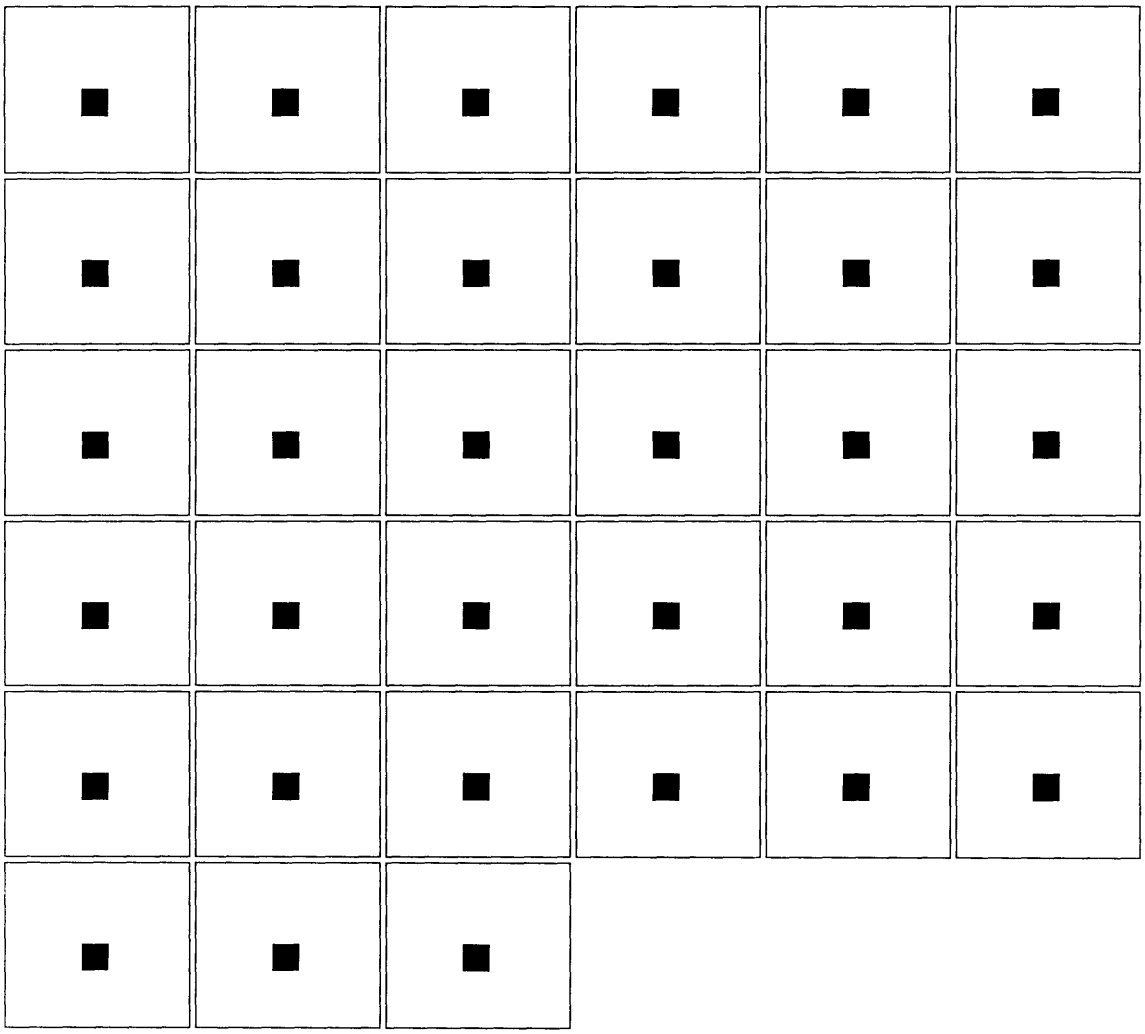


Figure 4-20: Mistral Document Initial Templates

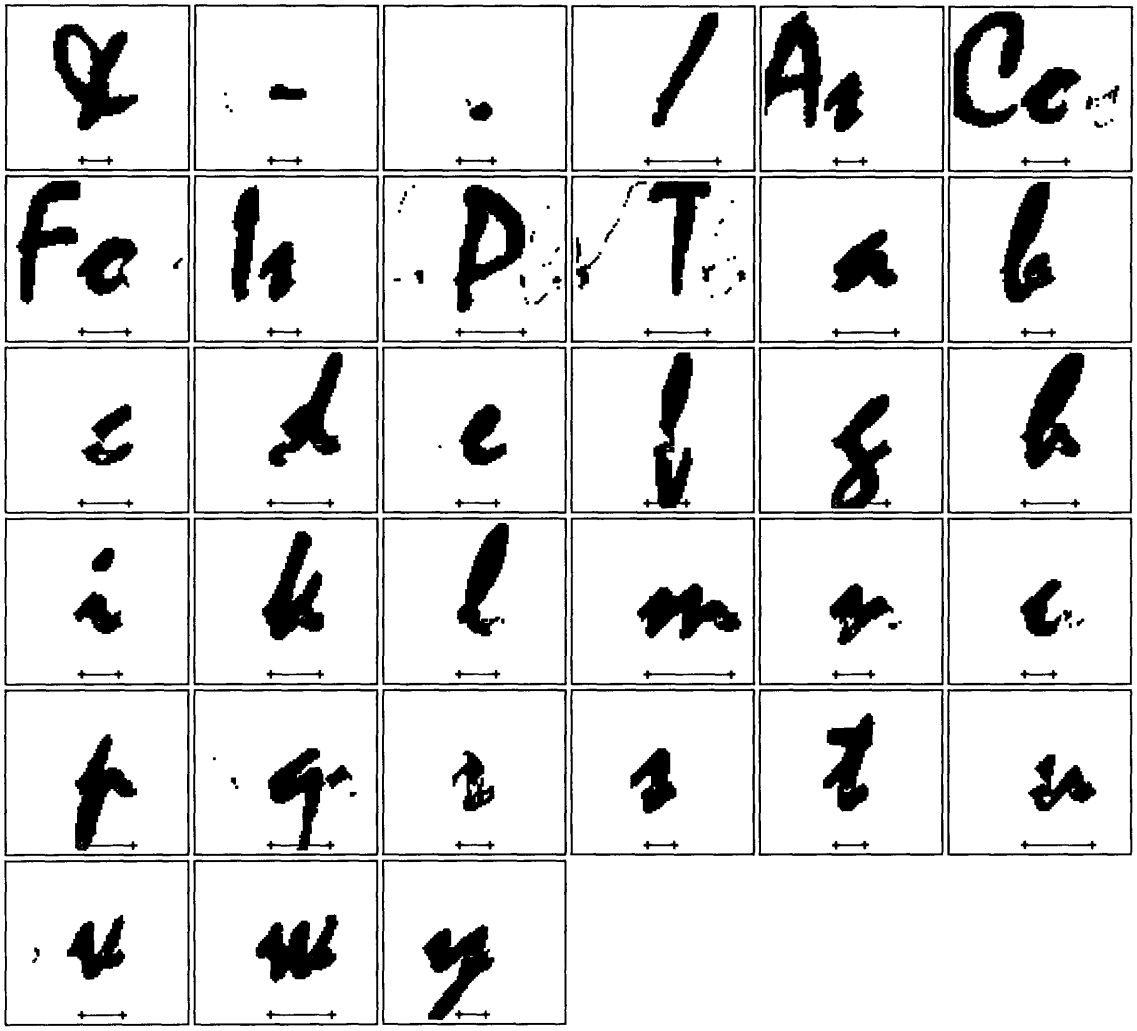


Figure 4-21: Mistral Document Estimated Templates after 8 Iterations

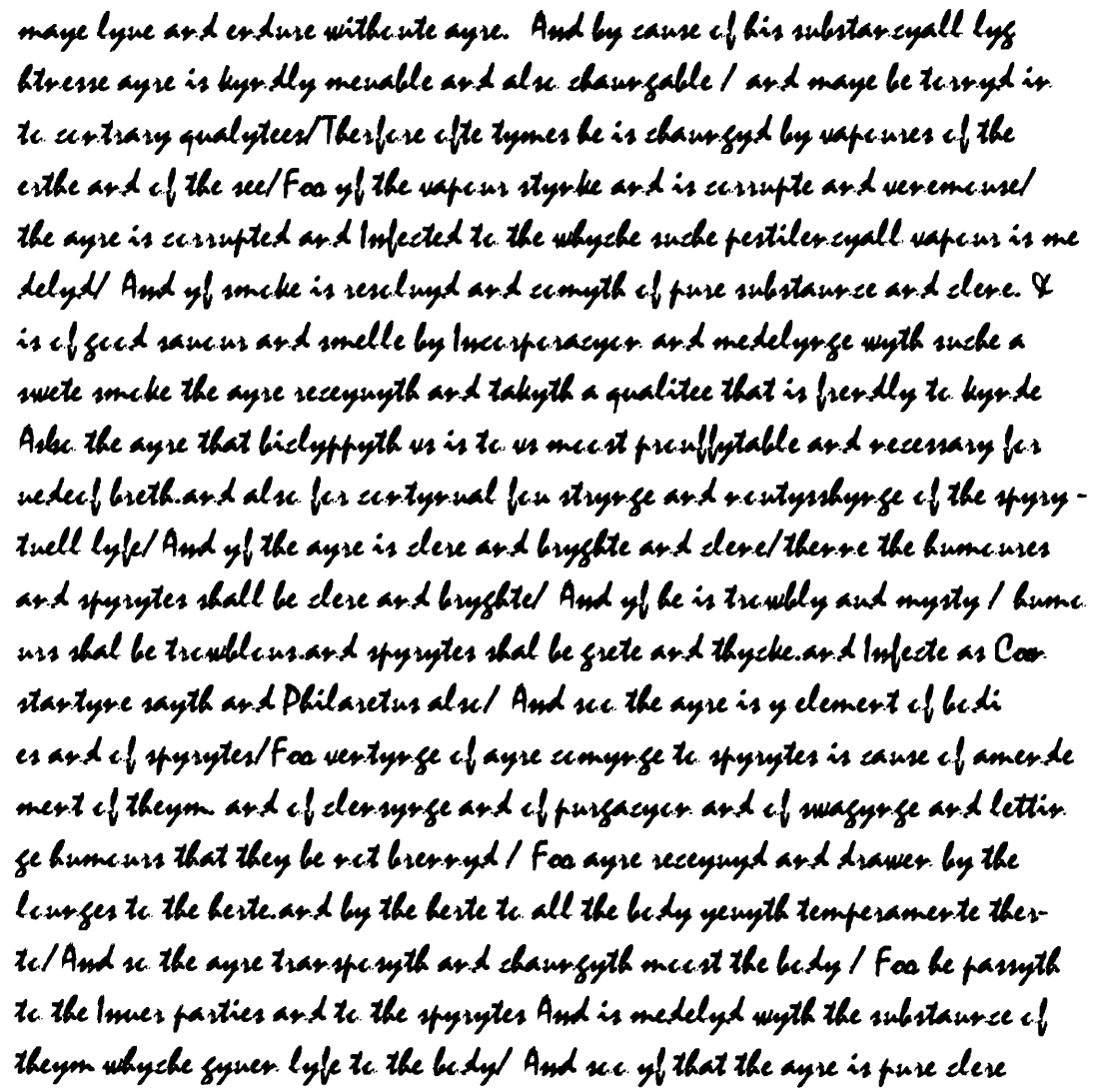


Figure 4-22: Mistral Document Template Reconstructed Image after 8 Iterations

4.5 LTR and RTL Decoding

LTR decoding refers to running the alignment procedure from *left to right* along a given baseline. RTL decoding refers to running the alignment procedure from *right to left* along a given baseline. The set of templates obtained from the LTR decoding are not the same as the templates obtained from the RTL decoding for several reasons. In the alignment procedure, if there is more than one path through the trellis that achieve the same score, the algorithm will always choose the first path that branches through an imaged template, instead of a null transition. Another inherent difference is that LTR decoding tries to latch on to the origin of each character and stretch the setwidths, whereas RTL decoding tries to latch on to the setwidths and stretch the origins.

The idea behind doing LTR and RTL decoding is to correlate the origins and setwidths between both sets of templates to obtain a new set of metrics that will yield a better alignment. In this section we examine the estimated templates from the Mistral example. The templates look good, albeit not ideal, but we cannot apply an estimation routine on the origins and setwidths (such as the one described in [10]) because the alignment on the reconstructed image does not correspond to the character placement on the original image. Refering to the template reconstructed image in figure 4-22 we call attention to the word "For" in the fourth line and a few other places as well. Careful observation of the template images used in the reconstruction show that the templates for the letters "o" and "r" occupy the space of the single letter "r" in the original image. This is due to the fact that the origin for the "F" in the template is very far from the expected origin and this results in very inaccurate alignment.

In contrast, the results for the Typed and Old English documents show that each template character image is properly aligned with its corresponding observation in the observed image. A metric estimation routine similar to the one described in [10] can now easily correct for origin shifts in the least squared sense. We now present a brief overview of this procedure.

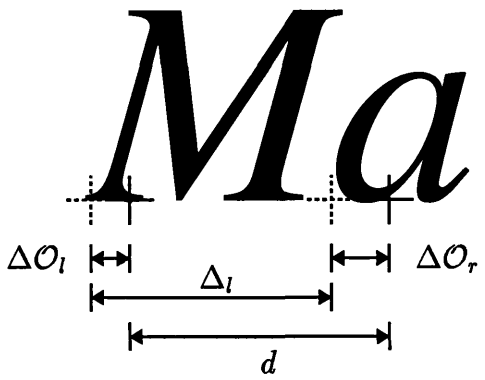


Figure 4-23: Metric Estimation on Aligned Characters. Estimated origins are indicated by solid crosses. True origins are indicated by dotted crosses.

4.5.1 Template Metric Estimation

Given a relatively accurate template alignment, as shown in figure 4-23, the objective of metric estimation is to determine $\Delta\mathcal{O}_l$ and Δ_l from a set of measurements of d. The kernel of the approach is to notice that for the ith pair of imaged templates, the origin shift and setwidth of the left glyph, $\Delta\mathcal{O}_{l_i}$ and Δ_{l_i} respectively, are related to d_i and to the origin shift of the right glyph, $\Delta\mathcal{O}_{r_i}$, by

$$\Delta_{l_i} - \Delta\mathcal{O}_{l_i} + \Delta\mathcal{O}_{r_i} + \epsilon_i = d_i \tag{4.7}$$

where ϵ_i is a random variable that accounts for small variations. The rest of the metric estimation procedure deals with finding the least squares estimates of $\Delta\mathcal{O}_k$ and Δ_k that jointly minimize the total squared estimation error $E_{\mathcal{O}} = \sum_i \epsilon_i^2$. The treatment of this procedure is outside the scope of this thesis; but a detailed description of such a procedure can be found in [10].

The metric estimation routine tacitly relies on the assumption that the character template images were correctly aligned with the glyphs on the observed image so that origin and setwidth estimation reduced to minimizing their variation in the reconstructed image. For this reason, we focus on a way of modifying the origins and setwidths of templates to obtain a good alignment that we can then apply the metric estimation routine to.

4.5.2 LTR and RTL Correlation

In any correlation between both sets of templates we must be careful to modify the origins and setwidths so that the current alignment is still feasible. There are two categories of metric correlation to consider:

- Cases where the setwidth vectors are "shrunk", meaning the origin and length of setwidth vectors are both "pulled in", and the current alignment is still feasible.
- Cases where the origin is shifted out, and we need to consider possible overlap of this shifted origin inside the setwidth vector of any character that is aligned to the left of every character instance.

The LTR and RTL templates are correlated by aligning both sets of corresponding character templates to where their matching score is greatest. Both sets of templates are taken after the iterative procedure has been allowed to "congeal." We assume that the only character that has the correct alignment is the space character, since we construct it as a "white block" that is origin- aligned inside the template canvass. In the setwidth estimation routine described earlier, if the lower bound on a setwidth is given by the space character, then we mark that setwidth as "probably correct."

The new origin and setwidth of both LTR and RTL templates is then taken to be the intersection of their setwidth vectors, shown in figure 4-24, under the following conditions.

1. If there is an overlap between the setwidths that is greater than the default setwidth, then we simply do appropriate origin shifts on each template to match origins on the ends of the intersection, and update the setwidth value on both templates to be the size of the resulting overlap.
2. If there is an overlap between the setwidths that is less than the default setwidth, then we take the midpoint of the intersection as center of the new setwidth, and set the new setwidth equal to the value of the default setwidth. [4]

[4]Notice that if there is no "overlap" between the setwidths, i.e. the setwidth vectors do not cover the same space, then the overlap will be negative and rule 2 applies.

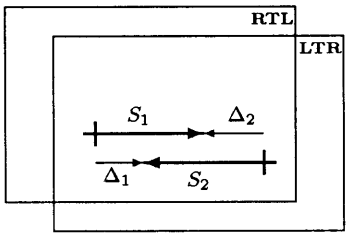


Figure 4-24: Alignment of LTR and RTL templates

The two conditions above are modified if any one or both of the templates' setwidth is marked as "probably correct."

Suppose that the setwidth of the LTR template is marked. The correlation shifts the origin of the RTL template to the setwidth of the LTR template, and resets the setwidth of the RTL template to its default value. The origin of the LTR template is then shifted so that the setwidth is reset to its default value. If the setwidth of the RTL template is marked, the symmetric analysis is followed.

Suppose that the setwidth of both the LTR and RTL templates is marked. Then we simply do appropriate origin shifts so that the origin of a template match up with the setwidth of the dual template.

Whenever an origin is shifted out, we shrink the setwidth of any character with whom the new origin location overlaps with. This is because any outwards origin shift is due to a marked setwidth in the dual template, which is "probably correct" due to the space character, and we have more confidence in this character than in an estimated one.

4.5.3 Experimental Results

Figure 4-25 and figure 4-26 show the LTR and RTL templates at the sixth iteration. The templates were allowed to "congeal" for six iterations before the setwidth estimation was applied. These are the templates before they are correlated. Figure 4-27 and figure 4-28 are the result of correlating the LTR and RTL templates shown in figure 4-25 and figure 4-26. These templates were then used again in the iterative pro-

cedure without correlating them for two iterations. Figure 4-29 shows the templates after eight iterations, with one correlation at the sixth iteration. Notice the origins for all the templates are more or less near where one would expect the true character origins to be at. These templates yield an accurate alignment on the observed image. A metric estimation routine can now be applied to these templates to more precisely determine the location of the character origins and setwidths.

Figure 4-30 shows the template reconstructed image for the templates in figure 4-29. There are slight open spots in the reconstructed image because the font was designed so that there is a slight overlap between characters as they are imaged. The template estimation routines developed in this thesis enforce the non-overlapping criterion, and this is an example where some allowance on overlapping would result in a better match between the reconstructed and observed image.

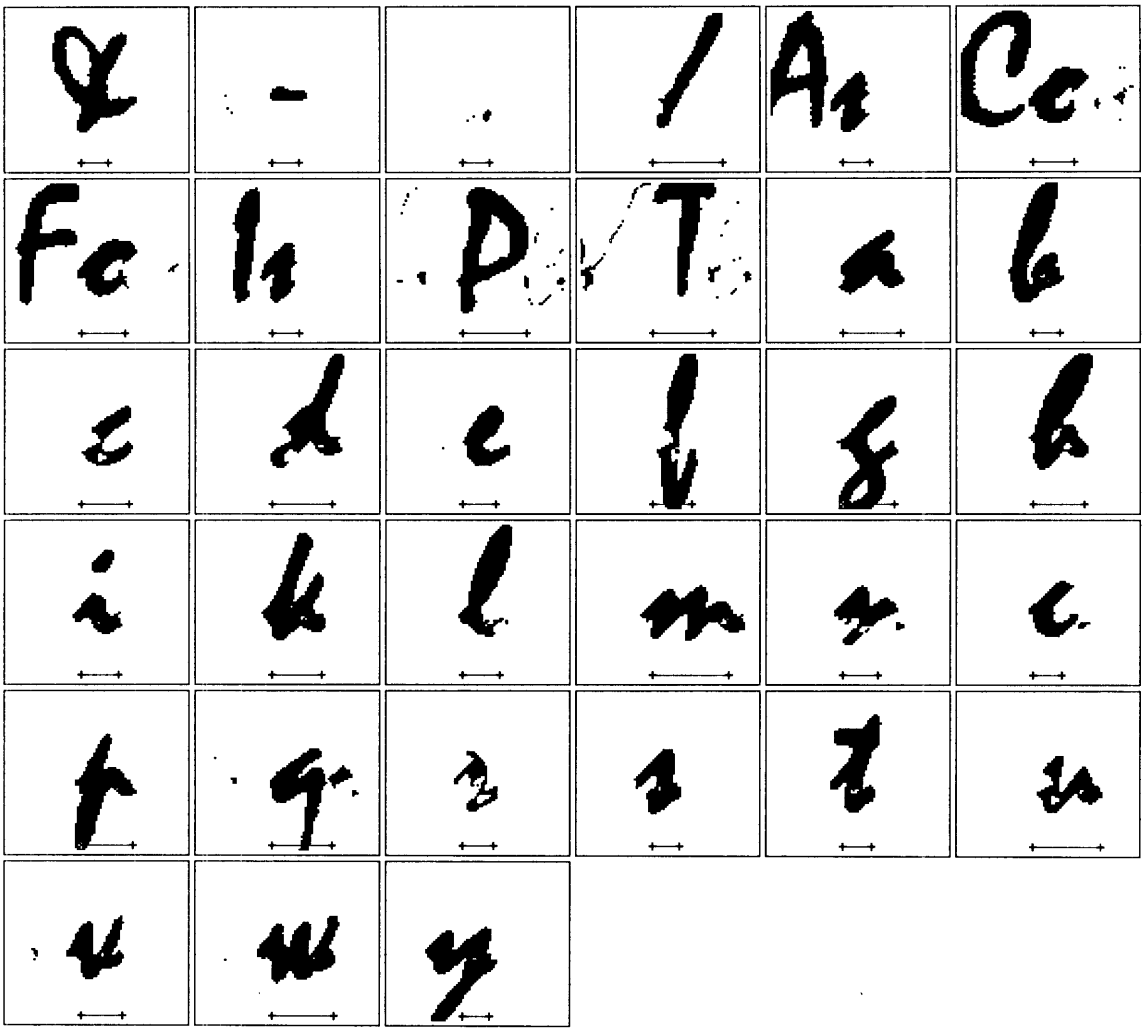


Figure 4-25: Mistral Document LTR Templates after 6 Iterations

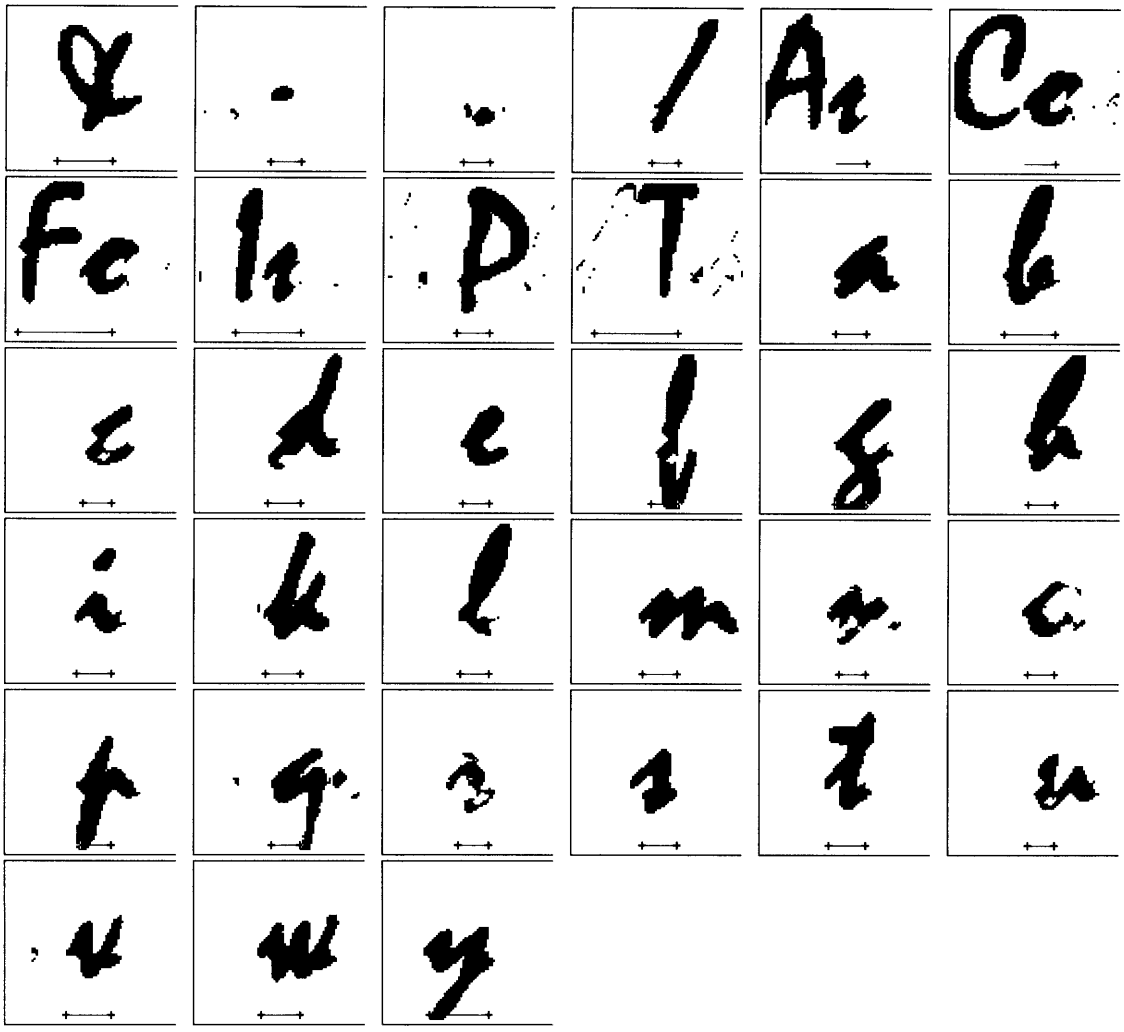


Figure 4-26: Mistral Document RTL Templates after 6 Iterations

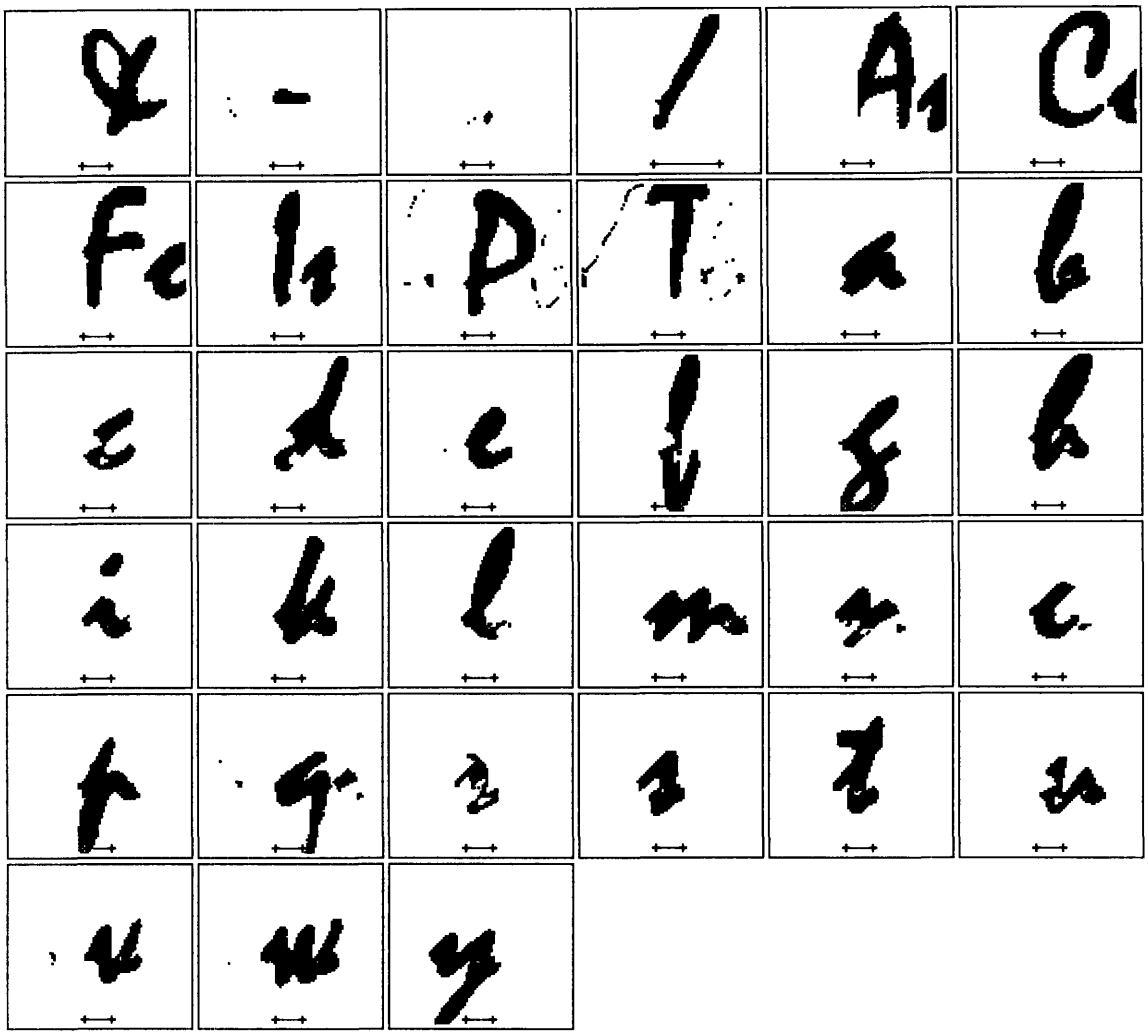


Figure 4-27: Mistral Document Correlated LTR Templates after 6 Iterations

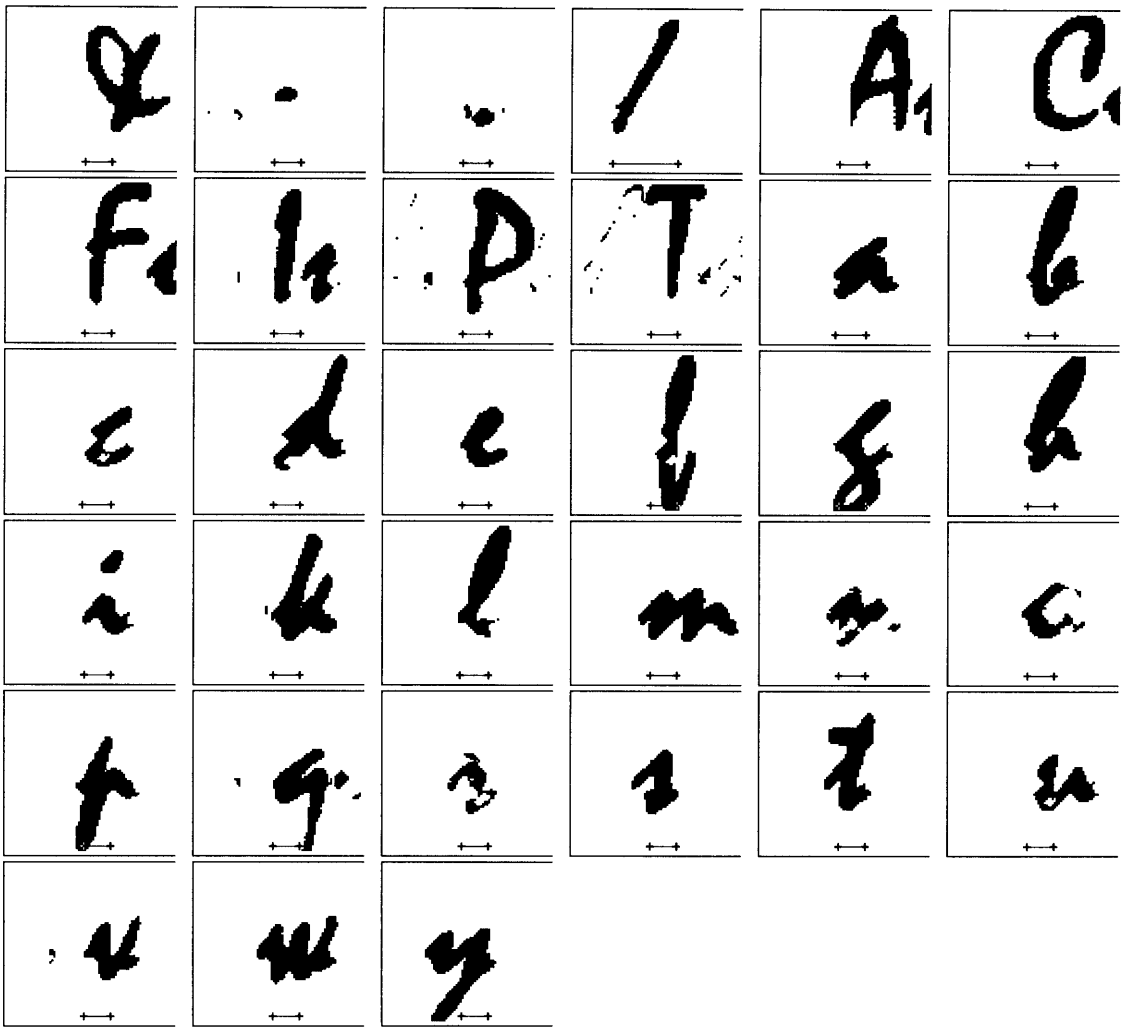


Figure 4-28: Mistral Document Correlated RTL Templates after 6 Iterations

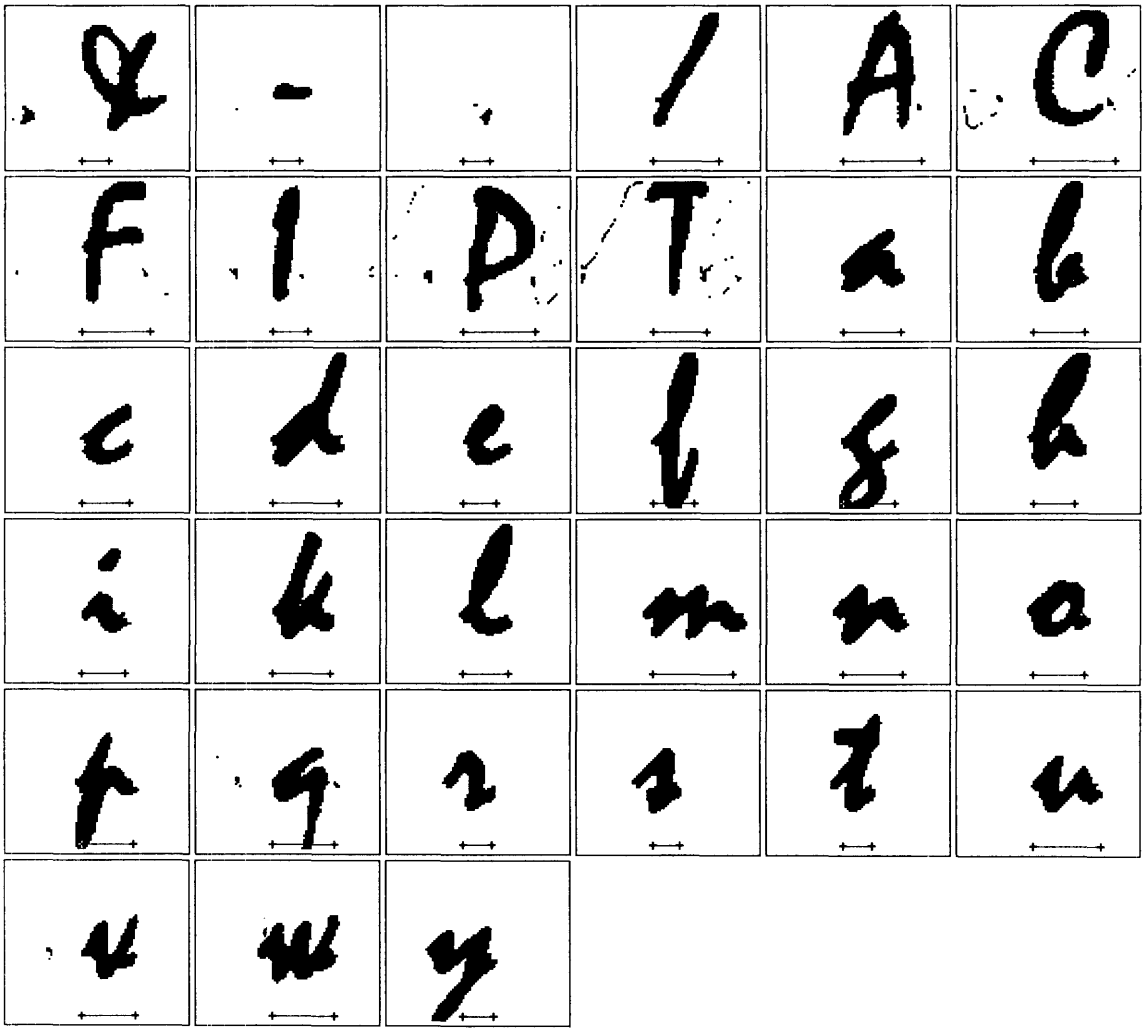


Figure 4-29: Mistral Document LTR Templates after 8 Iterations

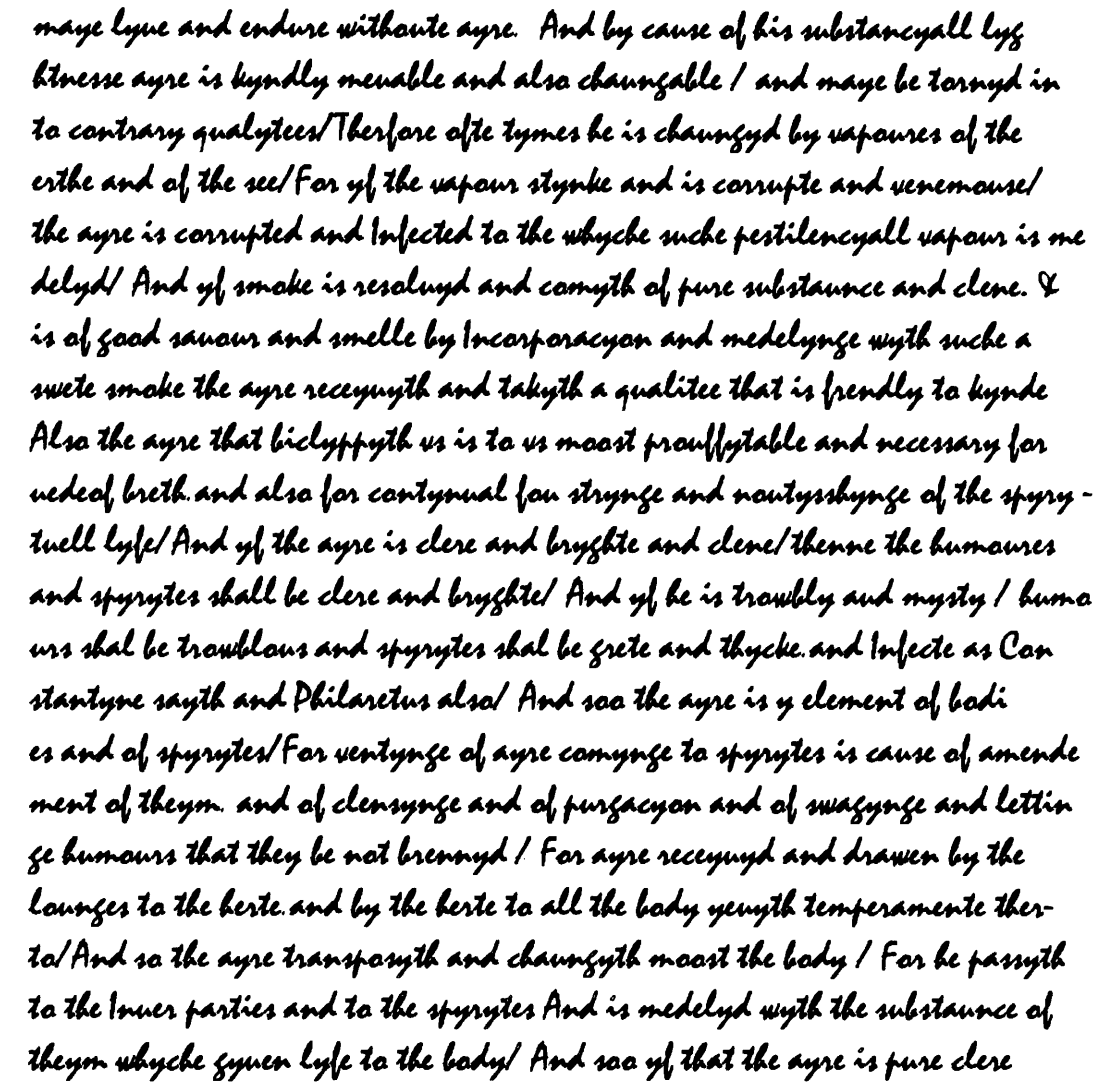


Figure 4-30: Mistral Document Template Reconstructed Image after 8 Iterations

Chapter 5

Summary and Further Directions

5.1 Summary

The objective of most document image analysis systems is to convert paper or microfilm based documents into a form that may be processed, searched, and stored by a computer. Template based OCR systems rely on accurate character template models for decoding document images and converting them into electronic documents. One of the drawbacks of these systems is that image quality differs across a wide range of document images, and the templates used by an OCR system might not be well suited to recognize document images where each observed instance of a character has been similarly degraded.

In this thesis we developed a procedure for training a set of character templates from a set of sample images and the corresponding transcriptions. The goal of such a procedure is to capture the image quality due to the physical degradation process, so that the estimated templates can be used to more accurately recognize unknown images of the same type.

Under the framework of DID, the problem of estimating character templates when the character origins on an image are given was shown to be NP-complete, and a combination of greedy+local search algorithms was provided that yielded very good results. The problem was then generalized to instances where all we have is a document image, the text baselines, and the transcription of each textline. Under the DID

framework, the baselines and transcription were easily merged into a separable page model. Using this page model and a modified image decoder, an iterative procedure was then developed for estimating character templates. In this iterative procedure we alternate between aligning a set of templates on the image and estimating the set of character templates using the current alignment.

This iterative template estimation procedure was applied to three very different images. In the Typed and Old English documents, the algorithm produced the proper models for the character shapes. The Mistral document demonstrated the procedure to be error prone. The font metrics of these templates were then modified by correlating the LTR and RTL templates, which resulted in the proper template models.

The work presented in this thesis has by no means been completed. There are a lot of extensions to consider by extending the analyses we have presented to a wider class of images, such as musical scores, or grayscale images.

5.2 Further Directions

In this section we list a few natural extensions of the work presented in this thesis.

Relaxing Overlapping Criterion

In a wide class of images, the non overlapping criterion is not strictly enforced by the imaging model. The Mistral document is one example where the typefaces are designed to overlap slightly so that the characters in a word appear connected. The extension from the greedy+refinement algorithms to allow overlapping should involve very minor modifications to the refinement algorithm. Specifically, since the greedy algorithm yields such good results, the refinement procedure should consider assigning pixels ON if they would contribute a positive likelihood score given that other templates have pixels that also contribute to the likelihood score of the reconstructed image.

One could easily find examples where this approach might not work well. However, given the earlier experimental results this sounds like a very attractive approach.

Font Mislabelings

The problem to a lot of recognition errors using trained templates seems to be bad templates as a result of font labeling problems, specifically lumping together dissimilar fonts into the training procedure.

When similar characters of the same font are lumped together into the same training data there is no clear method to determine that two or more different fonts for a given character exist. The character estimation model needs to be more robust to font mislabelings. One proposed approach is to use Minimum Description Length principles to "separate" the instances of a given character into a fixed number of different font templates. The likelihood of each of these font templates could be measured with the observed instances of its corresponding images, and compared to the likelihood of the original template, given that there is some measure of penalty for a large number of font templates. This is to avoid identifing each observed instance as a unique character template. How this "separation" can be carried out remains to be examined.

Noise Model

There are a lot of effects that are not entirely captured by the simple asymmetric bit flip noise model, such as thinning, stretching, or blurring. In a realistic situation, it is more likely to encounter blurring near edges of a character, and to find that black pixels inside the body of a character stay black. This indicates that a more realistic noise model would condition the probability that a pixel is flipped on the number of similarly colored pixels in its neighborhood. The use of grayscale images for template training is another area that can be examined further.

For stretching, thinning, or thickening of glyphs we propose examining the use of transformation functions for each template with an appropriate metric to measure large distortions from the nominal template image. Rotations could be esily incorporated into the decoding routine similar to the "jitter" used earlier for baseline outliers.

Template Representation

In this thesis we generated template images from its observed instances by aligning the instances from each other, forming a composite, and then applying a binarization threshold to the composite image so that the template had values of 0 or 1.

A character template could be similarly represented as a probability distribution matrix, where every pixel has a value proportional to the probability that it is ON in an arbitrary instance. The shift to this representation would require modification to the noise model, since it would now be incorporated into every template, and to the decoding routines, where the matching score would now be a probabilistic function of each template. This representation would have the advantage of easily incorporating edge effects of characters, such as blurring, where pixels corresponding to the true support of the template would have a very large probability, and pixels close to the edges that are susceptible to fade would have lower probabilities.

Character Outliers

A subtle problem we find with characters that have few instances is the presence of edge pixels from other characters being assigned to them. This problem can be found in the Mistral example for the "T" template in figure 4-29. Notice the presence of "edge" pixels corresponding to the template "/". In the reconstructed image this choice of pixel assignment yields a higher score than assigning these pixels to "/" because otherwise they would incurr a higher number of mismatches. Clearly, the notion of connected components for characters should play a role in the template estimation routines. It is not unreasonable to think that an added criterion for template images is the number of connected components, which has some connection with Minimum Description Length and Entropy principles, which were not addressed in this thesis.

Generalization to a 2-D model

Intuitively, a Markov source has the capability of generating a vast number of images without restriction on the positioning of imaged characters on the image plane.

In our character template estimation routines, we examined the 0-D problem where we are given the exact locations of all imaged characters on a page, and the 1-D problem where we search along a one dimensional image path for the origin locations of all imaged characters on a page. The extension of this work into a 2-D problem would need to search for the origins in all the image plane. In other words, estimate the character templates when we are only given a document image and its corresponding transcription.

This problem has already been addressed in this thesis. Specifically, we used some imaging processing techniques to estimate the baselines of a given document. This estimate can be regarded as a heuristic on the position of characters on the page, where we reduce the 2-D problem to a 1-D model. The difference is that the baseline estimation procedures were supervised. The algorithms to estimate the baselines on each of the three types of images used in this thesis were different, each one chosen because they resulted in better estimates.

In a structured approach to the 2-D problem a total search through the entire image plane needs to be carried out in order to estimate the templates. The use of heuristic estimates of the baselines matching the transcription would require some user intervention.

Appendix A

Cluster Assignment Problem is NP-complete

In this appendix we prove that the Cluster Assignment Problem, or CAP for short, under strict observation of the non-overlapping criterion is NP-complete. We also prove that the variant of the CAP where we relax the non-overlapping requirement is also NP-complete.

NP-complete problems are a class of problems that cannot be solved by any known polynomial algorithm. The significance of the identification of NP-complete problems is the belief that they are not susceptible to efficient algorithmic solution; and that any algorithm that correctly solves an NP-complete problem will require in worst case an exponential amount of time, and hence will be impractical for all but very small instances [2]. Therefore, proving that the CAP is NP-complete gives us some vindication for designing an algorithm that yields a good solution to the CAP without guarantee on its optimality.

The theory of NP-completeness is designed to be applied only to decision problems, which have only two possible solutions, either a "yes" or a "no". Since we are dealing with an optimization problem that asks for a certain structure to have a maximum "score" among all such structures, we associate this optimization problem with a decision problem that includes a numerical bound B as an additional parameter and asks whether there exists a structure of the required type having a score *at least* B.

We present two known NP-complete problems that are essential in the proofs, Independent Set (IS) and Minimum Cover (MC). Showing that IS or MC are special cases of each particular formulation of the CAP shows that the CAP is **at least as hard** as IS or MC, and since both IS and MC are NP-complete, so is the CAP.

Cluster Assignment Problem

Given a cluster $\mathcal{C}$, choose a subset $\mathcal{C}' \subseteq \mathcal{C}$ of pixels to turn ON such that cluster likelihood is maximized.

Independent Set

Instance: Graph $G = (V, E)$, positive integer $K \leq |V|$.
Question: Does G contain an independent set of size K or more, i.e., a subset $V' \subseteq V$ such that $|V'| \geq K$ and such that no two vertices in V' are joined by an edge in E?

Minimum Cover

Instance: Collection C of subsets of a finite set S, positive integer $K \leq |C|$.
Question: Does C contain a cover for S of size K or less, i.e., a subset $C' \subseteq C$ with $|C'| \leq K$ such that every element of S belongs to at least one member of C'?

Theorem A.1 *Cluster Assignment under strict observation of the non-overlapping criterion is NP-complete.*

Proof: We transform Independent Set to CAP. Let the graph $G = (V, E)$ and positive integer $K \leq |V|$ constitute an arbitrary instance of IS. We must construct a cluster $\mathcal{C}$ such that $\mathcal{C}$ contains a subset $\mathcal{C}'$ whose likelihood score $\mathcal{L}(\mathcal{C}')$ is K or more if and only if G has an independent set of size K or more.

The construction of $\mathcal{C}$ merely replaces for each vertex $v \in V$ a template indexed pixel $v \in \mathcal{C}$ with likelihood score $\mathcal{L}(v) = 1$, and for each edge $\{u, v\} \in E$ a conflict between cluster elements u and v. The image locations of the template indexed pixels, $\mathcal{S}_{\mathcal{C}}$, are irrelevant in this proof. The instance of CAP is completely specified by:

$$\mathcal{C} = \{v \mid v \in V\} \tag{A.1}$$

$$\mathcal{D}_{\mathcal{C}} = \bigcup_{v \in V} \{D_v\} \tag{A.2}$$

where $D_v = \{u \mid \{u, v\} \in E\}$. This instance can easily be constructed in polynomial time. Note that under strict observation of the non-overlapping criterion, a valid solution to the CAP is a subset $\mathcal{C}' \subseteq \mathcal{C}$ such that there exist no conflicts between any two distinct template indexed pixels $u, v \in \mathcal{C}'$; $D_u \cap D_v = \emptyset$. In terms of the graph G, it is easy to see that a valid solution equivalently picks a subset $V' \subseteq V$ such that no two vertices in V' are joined by an edge in E; V' is an independent set.

Suppose $V' \subseteq V$ is an independent set of G with $|V'| \geq K$. The corresponding subset $\mathcal{C}' \subseteq \mathcal{C}$ is given by the elements in V' and has a likelihood score of $\mathcal{L}(\mathcal{C}') = \sum_{w \in V'} \mathcal{L}(w) = \sum_{w \in V'} 1 = |V'| \geq K$.

Conversely, if $\mathcal{C}' \subseteq \mathcal{C}$ is a solution to the CAP with likelihood score $\mathcal{L}(\mathcal{C}') \geq K$, the corresponding independent set is given by the elements in $\mathcal{C}'$, with size $|V'| = \sum_{w \in \mathcal{C}'} 1 = \sum_{w \in \mathcal{C}'} \mathcal{L}(w) = \mathcal{L}(\mathcal{C}') \geq K$. □

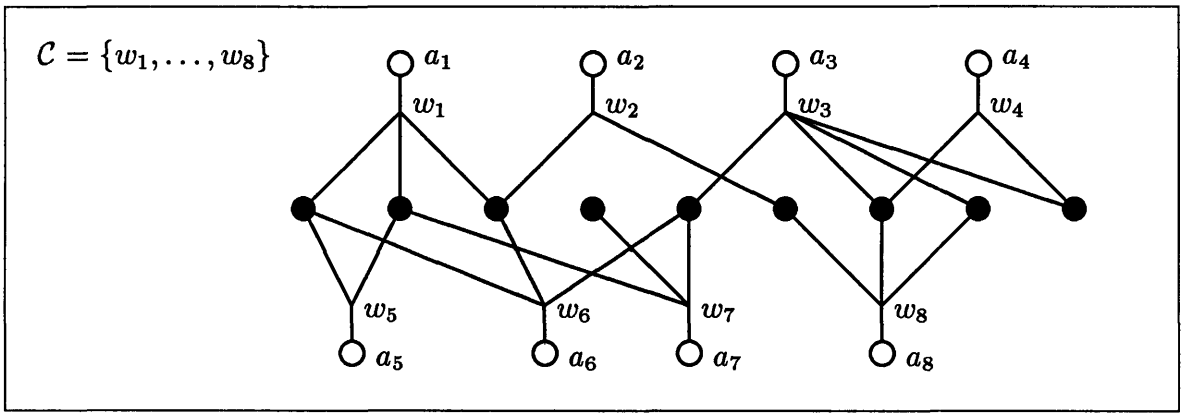


Figure A-1: CAP Construction from Minimum Cover

Theorem A.2 *Cluster Assignment with no restrictions is NP-complete.*

Proof: We transform Minimum Cover to CAP. Let the collection C of subsets of a finite set S, and positive integer $K \leq |C|$ constitute an arbitrary instance of MC. We must construct a cluster $\mathcal{C}$ such that $\mathcal{C}$ contains a subset $\mathcal{C}'$ whose likelihood score $\mathcal{L}(\mathcal{C}')$ is $|S| - K$ or more if and only if C contains a cover for S of size K or less.

Let $\{a_1, a_2, \ldots, a_{|C|}\}$ be new distinct elements not in S. The construction of $\mathcal{C}$ simply replaces for each $c_i \in C$ a corresponding template indexed pixel $w_i \in \mathcal{C}$. The construction of the image indexed pixels of each cluster element, S_{w_i}, replaces each member of c_i with an image indexed pixel that is ON and one unique OFF image indexed pixel a_i; $S_{w_i} = c_i \cup a_i$, where $c_i \xrightarrow{\alpha} \{\text{ON}\}$ and $a_i \xrightarrow{\alpha} \text{OFF}$. Assume the likelihood score of a matching pixel is +1, and of a mismatching -1. The conflict dependencies between cluster elements, $\mathcal{D}_\mathcal{C}$, are irrelevant in this proof. The instance of CAP is completely specified by:

$$\mathcal{C} = \{w_i \mid 1 \leq i \leq |C|\} \tag{A.3}$$

$$\mathcal{S}_\mathcal{C} = \{\{S_{w_i}\} \mid 1 \leq i \leq |C|\} \tag{A.4}$$

where $S_{w_i} = a_i \cup c_i$. This instance can be easily constructed in polynomial time. One such construction is illustrated in figure A-1, where each cluster element w_i is

represented by a tree with one branch that maps to one unique OFF pixel (white filled circles), and other branches that map to ON pixels (black filled circles).

Suppose $C' \subseteq C$ forms a cover for S with $|C'| \leq K$. The corresponding subset $\mathcal{C}'$ is given by the elements in C', and has a likelihood score

$$\begin{aligned}
\mathcal{L}(\mathcal{C}') &= \mathcal{L}(\bigcup_{w \in C'} S_w) && \text{(A.5)} \\
&= \mathcal{L}(\bigcup_{w \in C'} (c_w \cup a_w)) && \text{(A.6)} \\
&= \mathcal{L}((\bigcup_{w \in C'} c_w) \cup (\bigcup_{w \in C'} a_w)) && \text{(A.7)} \\
&= \mathcal{L}(\bigcup_{w \in C'} c_w) + \mathcal{L}(\bigcup_{w \in C'} a_w) && \text{(A.8)} \\
&= |S| - |C'| && \text{(A.9)} \\
&\geq |S| - K. && \text{(A.10)}
\end{aligned}$$

Conversely, if $\mathcal{C}' \subseteq \mathcal{C}$ is a solution to the CAP with likelihood score $\mathcal{L}(\mathcal{C}') \geq |S| - K$, there is a corresponding subset $C' \subseteq C$ given by the elements of $\mathcal{C}'$. Note that the set C' **might not be a cover** for S. Suppose C' **is a cover** of S, then

$$\begin{aligned}
\mathcal{L}(\mathcal{C}') &= \mathcal{L}(\bigcup_{w \in C'} S_w) && \text{(A.11)} \\
&= \mathcal{L}(\bigcup_{w \in C'} c_w) + \mathcal{L}(\bigcup_{w \in C'} a_w) && \text{(A.12)} \\
&= |S| - |C'| && \text{(A.13)} \\
&\geq |S| - K && \text{(A.14)} \\
\Rightarrow |C'| &\leq K. && \text{(A.15)}
\end{aligned}$$

Now suppose C' **is not a cover** of S, then

$$\begin{aligned}
\mathcal{L}(\mathcal{C}') &= \mathcal{L}(\bigcup_{w \in C'} S_w) && \text{(A.16)} \\
&= \mathcal{L}(\bigcup_{w \in C'} c_w) + \mathcal{L}(\bigcup_{w \in C'} a_w) && \text{(A.17)} \\
&= |S'| - |C'| && \text{(A.18)} \\
&\geq |S| - K. && \text{(A.19)}
\end{aligned}$$

Take C^* to be a cover of S such that $C' \subseteq C^* \subseteq C$, and every element $c \in C^* - C'$ includes at least one new element in S not included by any other element in $C^* - C'$, then

$$\mathcal{L}(\bigcup_{w \in C^* - C'} S_w) \geq 0 \tag{A.20}$$

because adding a subset that includes at least one new element in S will increase the likelihood by at least zero. Now, since,

$$\begin{aligned} \mathcal{L}(\bigcup_{w \in C^* - C'} S_w) &= \mathcal{L}(\bigcup_{w \in C^* - C'} c_w) + \mathcal{L}(\bigcup_{w \in C^* - C'} a_w) && \text{(A.21)} \\ &= |S - S'| - |C^* - C'| && \text{(A.22)} \\ &= |S| - |S'| - |C^*| + |C'| && \text{(A.23)} \end{aligned}$$

if we substitute (A.23) into (A.20) we have

$$\begin{aligned} |S| - |S'| - |C^*| + |C'| &\geq 0 && \text{(A.24)} \\ |S| - |C^*| &\geq |S'| - |C'| && \text{(A.25)} \end{aligned}$$

and since $|S'| - |C'| \geq |S| - K$ from (A.19),

$$\begin{aligned} |S| - |C^*| &\geq |S| - K && \text{(A.26)} \\ \Rightarrow |C^*| &\leq K. && \text{(A.27)} \end{aligned}$$

□

Bibliography

[1] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, McGrawHill, New York, 1991.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

[3] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[4] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[5] R. E. Blahut, *Principles and Practice of Information Theory*, Addison-Wesley, Reading, MA, 1990.

[6] R. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, 1991.

[7] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific, New York, 1989.

[8] W. H. Press, B. P. Flannery, S. A. Teukolsy and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1988.

[9] R. Rubenstein, *Digital Typography*, Addison-Wesley, Reading, MA, 1988.

[10] G. E. Kopec, "Least-Squares Font Metric Estimation From Images," *IEEE Transactions on Image Processing*, vol. 2, no. 4, October 1993.

[11] G. E. Kopec and P. A. Chou, "Document Image Decoding Using Markov Source Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, June 1994, pp. 602-617.

[12] G. E. Kopec and P. A. Chou, "A Stochastic Attribute Grammar Model of Document Production and its use in Document Image Decoding," *SPIE Proceedings on Document Recognition*, vol. 2422, February 1995, pp. 66-73.

[13] G. E. Kopec and Anthony C. Kam, "Separable Source Models for Document Image Decoding," *SPIE Proceedings on Document Recognition*, vol. 2422, February 1995, pp. 84-97.

[14] G. E. Kopec, P. A. Chou, and David A. Maltz, "Markov Source Model for Printed Music Notation," *SPIE Proceedings on Document Recognition*, vol. 2422, February 1995, pp. 115-125.

[15] P. A. Chou, "Recognition of Equations Using a Two-Dimensional Stochastic Context-Free Grammar," *Visual Communications and Image Processing IV*, November 1989.

[16] L. R. Rabiner and B. H. Juang, An Introduction to Hidden Markov Models, *IEEE Transactions on Acoustics, Speech and Signal Processing*, January 1986.

[17] L. R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, vol. 77, no. 2, February 1989.

[18] L. R. Bahl, F. Jelinek and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 2, March 1983.

[19] A. C. Kam, *Heuristic Document Image Decoding Using Separable Markov Models*, Master's thesis, Massachusetts Institute of Technology, 1993.

[20] R. M. Karp, *Reducibility Among Combinatorial Problems*, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, 1972.

[21] J. H. Shapiro and A. S. Willsky, *Probability Review and Random Vectors*, Supplementary notes for MIT course Stochastic Processes, Detection, and Estimation (6.432), Fall 1991.