Handwritten Character Recognition using the Minimum Description Length Principle

by

Anant Sahai

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996

Author .....
Department of Electrical Engineering and Computer Science
May 28, 1996

Certified by .....
S.K. Mitter
Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Frederic R Morgenthaler
Chairman, Departmental Committee on Graduate Students

Handwritten Character Recognition using the Minimum Description Length Principle

by

Anant Sahai

Submitted to the Department of Electrical Engineering and Computer Science
on May 28, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Off-line handwritten character recognition is a hard problem. A hierarchical formalization of Rissanen's Minimum Description Length Principle for use in recognition problems is developed. The formalism is shown to be NP-complete. It is then applied to the specific problem of character recognition and the ideas are then tested in a specially designed software workbench environment on real isolated handwritten numeral data from postal zip-codes.

Thesis Supervisor: S.K. Mitter
Title: Professor of Electrical Engineering

Acknowledgments

I would like to express my deep gratitude to my advisor, Sanjoy Mitter, for his encouragement and guidance. In addition, I would like to thank Irvin Schick for very helpful discussions at the start of this work and Sekhar Tatikonda, Upendra Chaudhari, S.R. Venkatesh, Stefano Casadei, and Mike Branicky for putting up with my vague ramblings and acting as a good sounding-board for ideas as I developed this work.

Finally, I would like to thank the U.S. Office of Naval Research for supporting me through their graduate fellowship program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The basic problem of off-line handwritten character recognition is easy to state. Given a scanned image of someone's handwriting, we wish to generate an ASCII (or other suitable computer readable format that is also easy to manipulate) representation of the written text. This problem is important in many practical contexts ranging from automatic sorting of physical mail to more automated check clearing systems. Of course, many of us would also prefer to be able to write our calculations with paper and pencil and have them converted into LaTeX for us automatically!

The difficulty of the problem arises primarily from the great variability in handwritten characters. Even a given individual rarely writes the same character twice in such a way that it looks the same. On the level of the scanned-in picture, it always looks different. This difference is also not due to just simple additive noise. Rather, the observed characters can differ by affine domain deformations such as scaling, slanting, translation, and rotation. They also differ by non-linear domain deformations such as changing the curvature of certain strokes. [1]

Most of the existing approaches to character recognition attempt to first extract some *features* from the raw scanned image and then run these through some type of *classifier* which actually decides which character is present. The main intuition behind this approach is to choose features which distill the relevant information in the raw image. These features are chosen so as to be as invariant as possible under the set of possible deformations. Then, the classifier needs only to consider the feature vector

which is generally of lower dimension than the raw scanned image. [1]

Unfortunately, most of these methods inherently rely upon having the input data segmented into regions which contain at most one character. It is clear that if the segmentation is done improperly, then these methods fail. In general, for handwritten data, the segmentation problem is very difficult. [1] In addition, there is a serious problem in picking feature sets. The choices are usually made on empirical grounds and it is often hard to utilize *a-priori* knowledge of the handwriting process in a principled way to guide feature selection.

A few years ago, Mohamed Akra proposed to think of the character recognition problem in a different way. He suggested that we apply metaphors and ideas from information theory. He also developed a particular method for doing character recognition involving deformable templates and a criterion which involved the local minimum of the single-sided Haussdorf distance.[1] In particular, Akra's method did not require segmenting the raw scanned data before the recognition process could begin. He tested this method on scanned laser-printed material and it performed quite well. It also appeared promising on some simple handwritten test examples. He also proposed that the recognition problem inherently involved representation in a hierarchy of different levels, though these ideas were never fully developed in his work.

We studied the application of Akra's method to a real database of scanned individual handwritten characters (from the Postal Service Zip Code Database). The performance was very poor (less than 40% correct matches). Moreover, our attempts to apply his method to this problem showed us that there were certain unanswered theoretical questions as well.

In this thesis, we study a way of formulating the information-theoretic approach to recognition so that it is more rigorous and fills in the gaps that we noticed while trying to apply Akra's work. Rissanen's Minimum Description Length (MDL) principle [6] (itself a formalization of Occam's Razor), provides us with some of the tools required to do this. While Akra had used the MDL principle as a motivation behind his work, he did not use it explicitly in his method. We feel that to get a more complete theory of recognition, it needs to be used explicitly as it is in the recent work of Geman [4].

The basic idea that we pursue is that *the problem of recognition is actually the problem of data representation*, viewed at an appropriate level of abstraction.[2] This gives us a natural cost function, the length of the representation, that we can use to judge the goodness of a particular representation. Thus, the problem of recognition can, conceptually at least, be reduced to a minimization of the length of the representation. All the specifics regarding what it is that we wish to recognize, etc. is captured in the design of the language in which the representation is expressed.

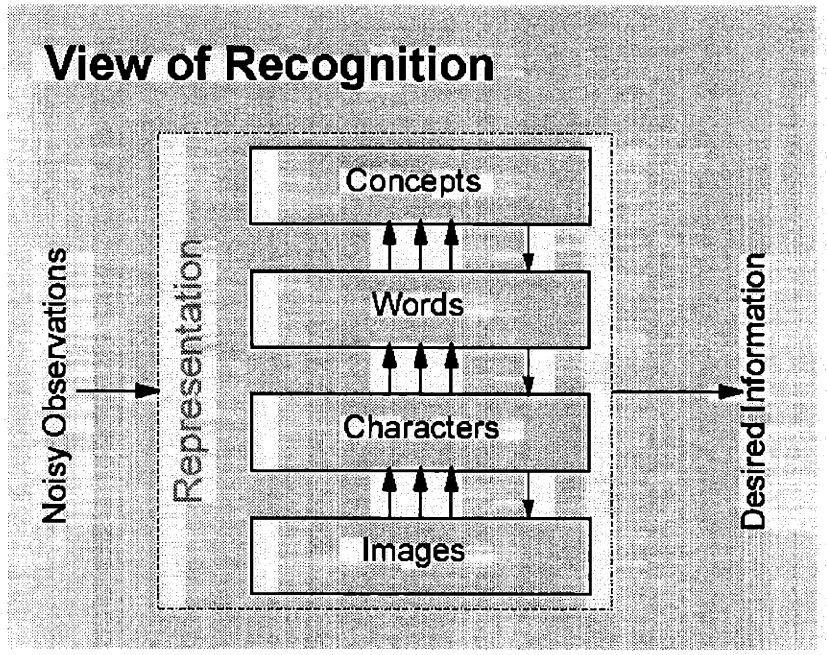

Figure 1-1: A picture of character recognition

It is important here to stress that in the above idea, the design of the language is critically important since *all the problem specific parts of the recognition problem* in principle must be addressed in the language. This is made especially interesting because unlike the statistical parameter estimation problems for which the MDL principle was originally developed, in most recognition problems we have an "outside the model" understanding of what constitutes a correct answer, namely whatever a human would say. Thus, great care must be given to make sure that the models of the process (from which the language follows in a natural way) are designed in a way that ensures that the human's answer and the MDL formulation's answer agree most of the time.

With such a formalization in hand, we implement it on a computer program and then apply it to the same real database of scanned handwritten characters that was difficult for Akra's method and discuss the results.

Chapter 2

The Big Picture Of Recognition

In this section, we present what we think is a useful formal framework in which to pose recognition problems.

2.1 Goal

Generally, the goal of recognition is to find the best acceptable representation for the observed data in a manner consistent with all other relevant information at our disposal.

2.2 The Basic Framework

We have an underlying set E_0 of elementary observations.

We have an operation + which acts on subsets of E_0 and returns other subsets of E_0. + can be considered to be the composition operation. For convenience, we assume some simple properties for +. + should be commutative and associative. The simplest, and prototypical, example for + is just the set union operator.

Definition 1 *A* model *of E_0 is a function m which takes a single positive integer argument θ and returns a subset of E_0.*

While we formally can think of it as a single positive integer argument, we will

actually think of it as a self-punctuated[1] into the finite number of positive integer arguments.

A model class is just a set of models. Note that a finite model class can be turned into a single model by creating a new model m' which has a first argument that tells which model from the model class to use.

Definition 2 *A* representation R *is a finite set of* θ *values.* R *is said to represent* O *iff* $I(R) \doteq \sum_R m'(\theta) = O$ *where the sum uses the appropriate operator* $+$ *mentioned above.*

Definition 3 *The* cost $C(R)$ *of a representation* R *is defined as follows:* $C(R) = \sum_R L(\theta)$ *where the* $L(\theta)$ *function is just the "bit-length" of* θ *when it is written as a binary string.*[2]

In addition, we may wish to place a further restriction on what we consider acceptable representations. So, we define a set G which consists of all the acceptable representations R.[3]

So, what is the MDL principle? It is quite simple. It says for a given observation O, we should pick that representation $R \in G | I(R) = O$ that has the minimal cost $C(R)$. We can express this in terms of a function MDL which is defined as follows:

Definition 4

$$\mathrm{MDL}(O|m, G) = \underset{R \in G | I(R) = O}{\operatorname{argmin}} C(R) \tag{2.1}$$

This can be understood as follows. Ignore G for a moment. Each realization R or O in effect covers O with $m'(\theta_i)$ sets relative to the composition operation $+$. So, we can conceptually imagine covering O with a very redundant cover $C = \bigcup R$ where the R range over all those R which represent O. Now, the MDL cover is the minimum cost sub-cover that still represents O.

[1] This allows the single integer to be uniquely decodable

[2] At first glance it may seem that this is a little dishonest since we are neglecting to tell the number of elements in R. But, since we have assumed that the θ_i are each self punctuated, we can represent the set R as just a single long bit-string consisting of the θ_i arranged in any order. We can then unambiguously break up this string.

[3] In the present work, we only care about G defined in extension. For real problems, we don't anticipate having G enumerated for us — rather it will probably be defined implicitly.

2.2.1 Example - Line Segments In The Plane

To give ourselves a concrete example of what we are talking about, let us think of the following problem:

We observe a scanned-in black and white image that is NxN pixels in size. (Assume N is an integral power of two for simplicity.) We know that it was generated in the following manner: Someone used a computer drawing program that could only draw line segments to create the image. This was laser printed on a somewhat dirty printer, and then scanned back in. We wish to recover the original set of lines.

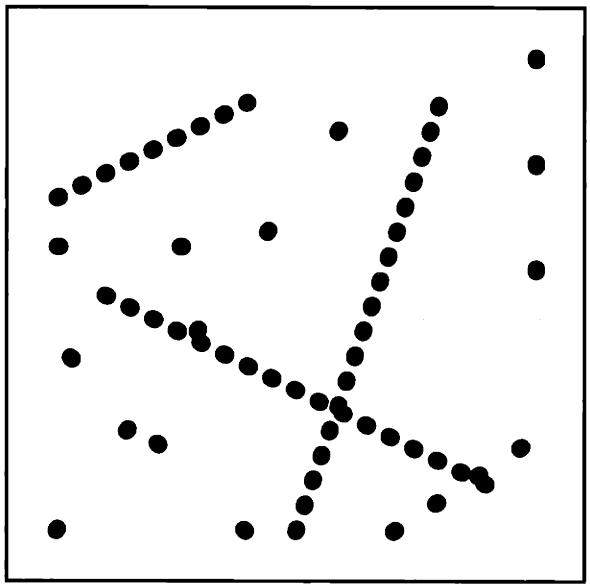

Figure 2-1: The dirty image

This is how we set up the recognition problem in our framework:

The set E_0 consists of ordered triplets of the form (i, j, k) where $0 \leq i < N$, $0 \leq j < N$, and $k \in \mathcal{R}$.

The $+$ operation is also pretty natural. We define $\{(i, j, k_1)\} + \{(i, j, k_2)\} = \{(i, j, k_1 + k_2)\}$ while $\{(i_1, j_1, k_1)\} + \{(i_2, j_2, k_2)\} = \{(i_1, j_1, k_1), \{(i_2, j_2, k_2)\}$ as long as $(i_1, j_1) \neq (i_2, j_2)$. Everything else follows from commutativity and associativity. Basically the $+$ operation is just the normal function adding operation.

The model consists of a function $m(\theta)$ that can be written in the following way: $m(i_1, j_1, i_2, j_2, n_{black}, i'_1, j'_0, \ldots, i'_{n_{black}}, j'_{n_{black}}, n_{white}, i''_1, j''_1, \ldots, i''_{n_{white}}, j''_{n_{white}})$ returning a

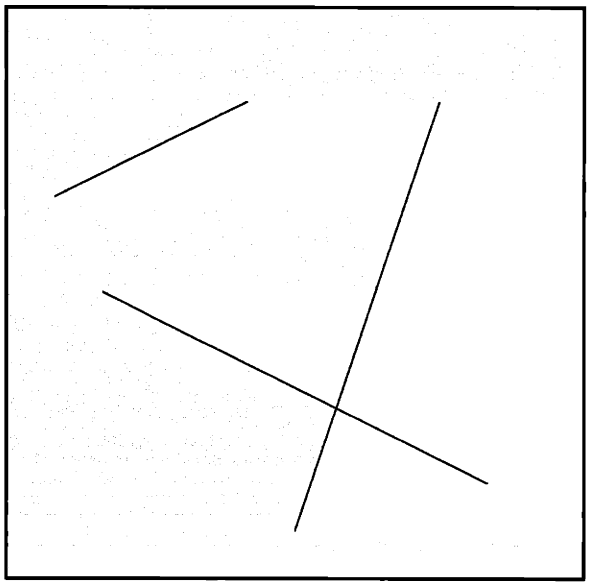

Figure 2-2: The ideal image

1 valued line from (i_1, j_1) to (i_2, j_2) plus (in the sense of the + operation defined above) n_{black} additional 1 valued points at $(i'_0, j'_0), \ldots, (i'_{n_{black}}, j'_{n_{black}})$ plus n_{white} additional -1 valued points at $(i''_0, j''_0), \ldots, (i''_{n_{white}}, j''_{n_{white}})$ plus $\{(i, j, 0) | i \in [0, N-1], j \in [0, N-1]\}$. Here all of the i and j values are encoded using $\log_2(N)$ bits in the natural fashion. The n values are encoded using a standard self-punctuating code for the positive integers. We just let G be all of 2^{Z^+}, so we accept any representation.

Now, we encode our observed scanned in image into a $\{0, 1\}$ valued function on the domain $[0, N-1] \times [0, N-1]$. Call this our observation O.

Then, the representation $R = \text{MDL}(O|m, G)$ corresponds well to our intuitive idea of what the best representation is. We can see that usually $R = \{\theta_1, \theta_2, \ldots\}$. Here each of the θ_is corresponds to the lines in the picture and we can see that all of the other black and white pixels will be crammed into one of the θ_i terms. In other words, we will represent the image as a set of lines in the plane, one of which may be corrupted by additive noise.

2.3 Models And Their Parameters

In the above Basic Framework, all of the details are thrown into the construction of the models themselves. So, at this point, We would like to describe some useful properties of models.

For convenience, let us think in terms of many parameters rather than just the one integer. Assume that everything is nicely self punctuated.

Now, at each level of abstraction, there are two different kind of parameters to a model. We call them *style parameters* and *fit parameters.*

Definition 5 Style parameters *are those parameters for which all possible values take the same number of bits to encode.*

These can be thought of as fixed-cost parameters or as the parameters which parameterize "costless deformations." No particular value for these parameters is any better than any other. For example, in our single character recognition case, these will be translation, small rotation, shear, etc. Hence the fixed cost. Probabilistically viewed, these are the parameters over which the probability density is uniform.

Definition 6 Fit parameters *are all those parameters which are not* style parameters.

These can be thought of as the variable-cost parameters. These characterize "costly deformations". For example, the in our single character case, these are things like small displacements of single points, added noise pixels, removed noise pixels, etc. Certain parameters (like no or less noise) are "better" than others and hence take fewer bits to encode. Probabilistically viewed, these are the parameters over which the probability density is non-uniform.

For convenience, we will usually rearrange the parameters so as to mostly write: $\theta = \theta^{style}\theta^{fit}$.

These two different kind of parameters have a nice interpretation if we think of the model as representing a data channel. This channel has two inputs. One is the user. He wishes to convey a particular message to us, we have no prior over what that

message is. For example, he might command the source to generate "the numeral '5' in italics, 10pt size, 3 pixels thick, located at (5,6) on the paper." The second input comes from "nature" or "noise." This represents the random distortions introduced by the channel to the information being sent. It need not simply be an additive process. For example, if a part of the channel is a human hand, nature could make the hand shake slightly, inducing domain deformations, etc. The important thing is that these are modeled probabilistically. See Figure 3-3 for a graphical depiction.

So, we see that there is often a natural correspondence between models (with their embedded decoding of the parameters) and our physical intuition regarding the true source of the phenomenon. In fact, it is this natural correspondence that enables us to make principled use of our intuitions and *a-priori* knowledge in designing models.

2.3.1 Example - Line Segments In The Plane

Let us return to our existing model for lines in the plane. What are the the *style parameters*? What are the *fit parameters*?

It is clear that the *style parameters* are the first four arguments to m, namely the i_1, j_1, i_2, j_2 which tell where the line is supposed to go. All the rest, the parameters corresponding to where we should add single pixel noise, are *fit parameters*. This is understandable in terms of our intuitive model as well. The user decides where each line is going to start and finish. Nature, in this case the noise introduced by the printer and the scanner, adds white and black pixels to the whole thing.

2.4 Hierarchy

Hierarchy often occurs naturally in most recognition problems — phenomena can be viewed at many different levels. We believe that the key to understanding hierarchy in this framework is to understand how to abstract models, and how this abstraction affects MDL.

First, we know that we can abstract things by creating a new observation class: E_1. The elements of E_1 are model-parameter pairs (m, θ) (or equivalently, values for θ arguments to m'). The + operation is simple set union. We can now consider models over this E_1. This process can be continued indefinitely creating hierarchies upon hierarchies of models. Now, it is important to remember that for every model m that models the observation class E_i there exist natural models m^{i-j} that model the observation class E_j for $0 \leq j < i$.

Definition 7 $m^k(\theta) \doteq I(m^{k-1}(\theta))$, *with the base case being defined as follows:*

$$m^0(\theta) \doteq \theta \tag{2.2}$$

We can similarly define a natural tower of representations as well.

Definition 8 *Given a representation R which exists at the level i, we set $R^0 \doteq R$, and recursively define $R^k \doteq I(R^{k-1})$*

Now, let us look at the nature of the models that exist on these higher levels. Suppose that $m(\theta) = \{\theta_0, \theta_1, \ldots, \theta_n\}$.[4] Now, what needs to be in θ? We claim that it is reasonable to expect that the θ should contain copies of all of the *fit parameters* that are inside the θ_i. We need some new notation to handle what is left.

Definition 9 $\theta^{metastyle}$ *and* $\theta^{metafit}$ *are those* style parameters *and* fit parameters *respectively which are used to determine the* style parameters *of the model directly below the one in question.*

[4]Remember here that the θ_i are the parameters for the model on the immediately lower level

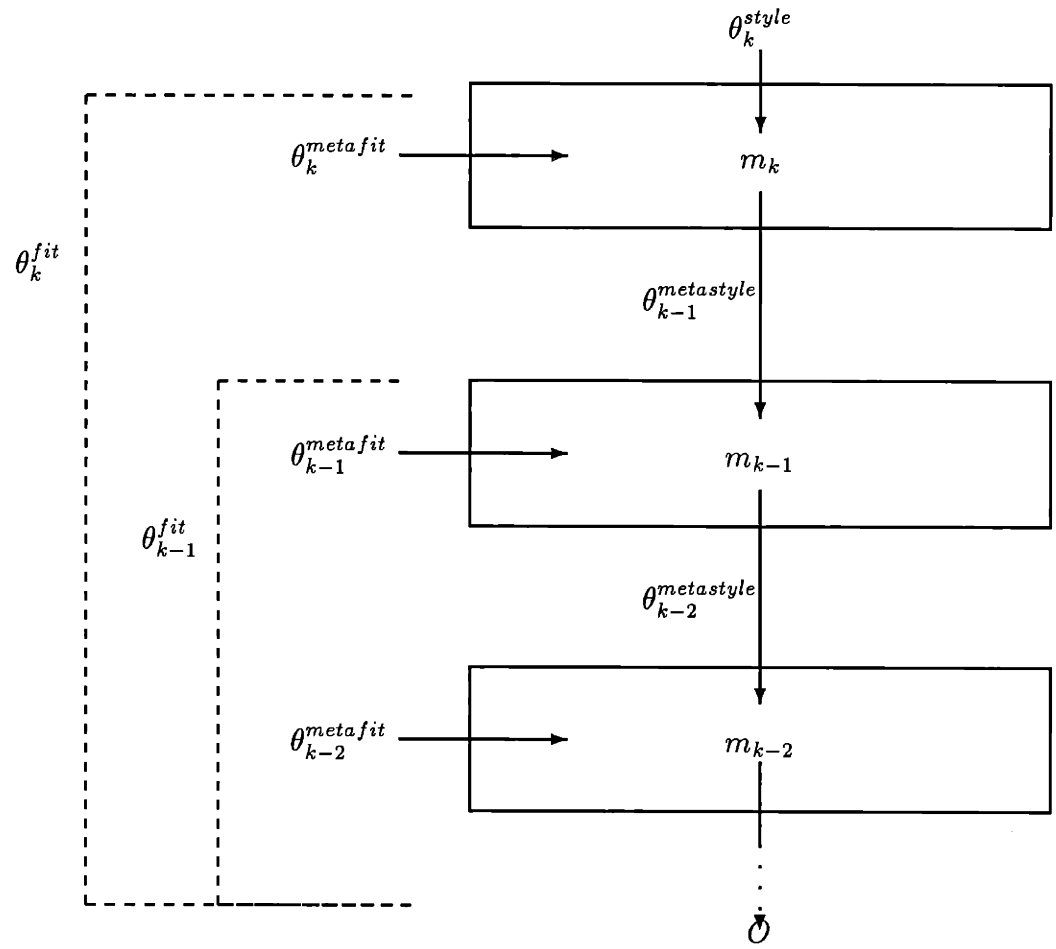

Figure 2-3: A picture of parameters.

Basically, if we think of each θ_i as being decomposable into θ_i^{style} and θ_i^{fit}, then θ contains a virtual copy of each of the θ_i^{fit} parameters while the θ_i^{style} are functions of the other parts of θ. So, we can say that $\theta = (\theta^{metastyle}, \theta^{metafit}, \theta_1^{fit}, \theta_2^{fit}, \ldots)$. So, the model generates the θ_i^{style} as a function of $\theta^{metastyle}$ and $\theta^{metafit}$. All of the *fit parameters* are just copied from θ_i^{fit}. We say that a model with this property *respects* the lower level. Furthermore, we should restrict the G_i sets so that if something is allowed in the *fit parameters* on level i, it should be allowed on the higher levels as well. In other words, we assume that if $R \in G_i$, then $I(R) \in G_{i-1}$. (Otherwise we just restrict G_i such that it is true)

The key idea here is that *what looks like* style *(ie. it looks arbitrary) on one level,*

may actually be understandable (non-arbitrary) on a higher level. Also, what is valid "noise" on a given level should not be disallowed on a higher one.[5] This kind of abstraction is in general the reason to consider a higher level in the first place!

Looking at the above kind of hierarchy, it seems strange that we are carrying around all of the extra *fit parameters.* This is not really a problem since we are free to mentally ignore these extra *fit parameters* since they don't really make any difference to our understanding of what the model does. We can think of them as only existing on the lower levels. See Figure 2-3 for a visualization. To avoid even that, we make another kind of abstraction, one that collapses all those *fit parameters* into a single more compact *fit parameter.*

2.4.1 Hierarchy's Interaction With MDL

To understand this second kind of abstraction, we have to first establish the relationship between MDL and abstraction. Consider a model m_k for level k of the hierarchy and a particular observation O which lives at the base level E_0. Suppose $\hat{R}$ is the best possible representation of O relative to m^k. Now, we intuitively see that if $\theta \in \hat{R}$, then $\{\theta\}$ is the best possible representation for $m_k^k(\theta)$. Moreover, this sort of sub-optimality holds on lower levels as well. So, if $\theta \in \hat{R}^j$, then $\{\theta\}$ is the best possible representation of $m_{k-j}^{k-j}(\theta)$ — but only relative to the *fit parameters.* The *style parameters* need to be taken as a given.

Let us make the above more precise.

Definition 10 *Suppose we have a hierarchy of the type specified above. We have a tower of observation classes* $E_0, E_1, \ldots$ *and corresponding models* $m_0, m_1, \ldots$ *and constraint sets* G_i*, all of which* respect *their lower levels. Suppose we also have an observation* O_i *at some level* i*.*

Now, define $\hat{R} = \text{MDL}(O_i | m_k^{k-i}, G_k)$.

In plain English, this means that $\hat{R}$ is the best possible representation of O_i relative

[5] This is not a harsh restriction. We can always concoct additional *style parameters* that vary what kind of noise we allow. Then, these can be controlled from above.

to the model m_k (when viewed as representing things in E_i) that satisfies "constraint" G_k.

Now, fix $i < j \leq k$. Consider any $\theta' \in \hat{R}^{k-j}$. This θ' can be written as $(\theta'^{style}, \theta'^{fit})$. Now, we define a G'_j as follows. $G'_j = G_j \cap \{\{\theta\} | \theta = (\theta'^{style}, A), A \in Z^+\}$ This G'_j is the constraint which is established by the higher levels. We claim θ' is optimal relative to this constraint set.

Theorem 1 $L(\theta') = C(\text{MDL}(m_j(\theta')|m_j, G'_j))$.

Proof: *Suppose otherwise. So, we have a* θ'' *such that* $m_j(\theta'') = m_j(\theta')$, $\theta'' = (\theta'^{style}, A)$, *and* $L(\theta'') < L(\theta')$. *Then, we know that* $L(A) < L(\theta'^{fit})$. *But, since all the models respect the hierarchy, we know that a copy of* θ'^{fit} *is sitting inside at least one* $\hat{\theta}_l \in \hat{R}$. *We can construct a new* $\hat{\theta}'_l$ *simply by substituting* A *for* θ'^{fit} *in the appropriate place. Then, we know that since* $L(A) < L(\theta'^{fit})$, $L(\hat{\theta}'_l) < L(\hat{\theta}_l)$. *So, we can construct a new* $\hat{R}'$ *with* $\hat{\theta}_l$ *replaced with* $\hat{\theta}'_l$. *It is obvious that* $C(\hat{R}') < C(\hat{R})$ *while* $\hat{R}'^{k-i} = \hat{R}^{k-i} = O_i$. *But this violates our definition of* $\hat{R}$ *as* $\text{MDL}(O_i|m_k, G_k)$. □

Actually the previous result can be extended to apply not just to a single $\theta' \in \hat{R}$ but to any subset of $\hat{R}^{k-j}$. We can use similar arguments to establish it.

Moreover, notice that the only aspect of the θ'^{fit} that we used was that a copy of it existed in the representation at the highest level. So, if there is a *style parameter* with the same property, the identical result will hold.

We can see that since there is this recursiveness to the optimal solutions, we know that not all of the sub-solutions will be used. In other words, at any level i, there are θ values that are *never* the MDL solution to anything. So, why should we bother even considering them? We can take any model m_i with associated constraint G_i and construct a new constraint $\hat{G}_i^j$ as follows.

Definition 11 $R' \cong R$ *whenever*

$$(\forall \theta \in R, \exists \theta' \in R' | \theta^{style} = \theta'^{style}) \wedge (\forall \theta' \in R', \exists \theta \in R | \theta^{style} = \theta'^{style}) \qquad (2.3)$$

Intuitively the equivalence relation $\cong$ captures everything that any higher level

that *respects* this one can possibly care about. Now, we can define the $\hat{G}_i^j$ using the following.

Definition 12 $\hat{G}_i^j = \{R \in G_i | C(R) = C(\text{MDL}(R^j | m_i^j, (G_i \cap \{R' \text{s.t.} R' \cong R\}))\}$

In particular, it is clear that replacing G_i with $\hat{G}_i^l$ will not change the behavior with respect to MDL on either this or on higher levels. By the recursiveness property that we already know, it suffices to look at the level immediately above i. First we define a new induced constraint set $\tilde{G}_{i+1}(A)$ as follows.

Definition 13 $\hat{G}_{i+1}(A) \doteq \{R \in G_{i+1} | I(R) \in A\}$

Now, it is easy to see that if for $j \leq i - l$, $R = \text{MDL}(O_j | m_{i+1}^{k-j}, G_{i+1})$, then $R \in \tilde{G}_{i+1}(\hat{G}_i^l)$ as well. So, as far as MDL is concerned, we have not really further restricted G_{i+1} at all!

If we wish, we can actually describe another more restrictive constraint set $\hat{G'}_i^j$ as follows:

Definition 14 $\hat{G'}_i^j \doteq \{R \in G_i | R = \text{MDL}(R^j | m_i^j, (G_i \cap \{R' \text{s.t.} R' \cong R\})\}$

The key difference here is that we are arbitrarily choosing only one of the minimal representations R.

Now, we have possibly confused the MDL function on higher levels. But, not by much. In fact, it is easy to that if for $j \leq i - l$, $R = \text{MDL}(O_j | m_{i+1}^{k-j}, G_{i+1})$, and $R' = \text{MDL}(O_j | m_{i+1}^{k-j}, \tilde{G}_{i+1}(\hat{G'}_i^l))$, then $C(R) = C(R')$ as well. Once again, so far as MDL is concerned, we have not really further restricted G_{i+1} in any important way!

Notice that in general, by restricting the G_i we are making much of the information in the *fit parameters* redundant. After all, we know that at level $i+1$ that the multitude of *fit parameters* that we are carrying around must be such that they are consistent with $\hat{G'}_i^l$. That means that we can save some bits by taking advantage of that fact. So, we can construct a new model $\hat{m}_{i+1}$ such that it can be viewed as taking a parameter of the form: $\theta = (\theta^{metastyle}, \theta^{metafit}, \overline{\theta^{fit}})$ where the $\overline{\theta^{fit}}$ captures all of the information in the old $(\theta_1^{fit}, \theta_2^{fit}, \ldots)$ parameters relative to the now known $\hat{G'}_i^l$. This is the more compact *fit parameter* we were referring to earlier.

All we need to do to specify $\hat{m}_{i+1}$ is give a function $f_{i+1}(\theta^{metastyle}, \theta^{metafit}, \overline{\theta^{fit}})$ that returns the old $(\theta_1^{fit}, \theta_2^{fit}, \ldots)$ parameters. We define it as follows. Consider a non-empty set $A = (m_{i+1})^{-1}(R)$ where $R \in \hat{G'}_i^l$. Now, let $B = (m_{i+1})^{-1}(\{R' \in \hat{G'}_i^l | R' \cong R\})$. Now, pick a $\theta \in A$. Write $\theta = (\theta^{meta}, \theta^{lower})$ where $\theta^{meta} = (\theta^{metastyle}, \theta^{metafit})$ and $\theta^{lower} = (\theta_1^{fit}, \theta_2^{fit}, \ldots)$. Let $C(\theta^{meta}) = \{\phi \in B | \phi^{meta} = \theta^{meta}\}$. Finally, let $D(\theta^{meta}) = \{\phi^{lower} | (\theta^{meta}, \phi^{lower}) \in C(\theta^{meta})\}$ So, what we need $\overline{\theta^{fit}}$ to do is to select an element out of D in such a way that is proportional to the length of the element of D that is being selected. We can do this by building an appropriate Huffman-like code.

It is important to notice that $\hat{m}_i + 1$ does not *respect* the lower level since it does not carry a copy of the lower level's parameters. The interesting question is whether or not replacing m_{i+1} with $\hat{m}_{i+1}$ substantially changes the behavior of MDL. We do not yet have a good answer to this question in general and it is the subject of current investigation.

2.5 Complexity

So far, we have not discussed the computational complexity of recognition in the above framework. Obviously, we would like recognition problems posed in this framework to be tractable.

For the system to be tractable, we need to first make a few assumptions.

1. The composition operation + on every level needs to be polynomial-time in its two arguments. This is the case for most useful + operations.

2. All the models m^k must be such that computing $m^k(\theta)$ is polynomial-time in the length $L(\theta)$ and the size of $m^k(\theta)$ itself.

3. Checking membership in G_k can be done in polynomial time in the length of R and the description of G.

With these basic restrictions in place, we will first show that in general, the decision problem associated with the optimization problem above (minimizing the description length) belongs to NP. First, let us formally state the decision problem.

Problem 1 *Given as input an observation O, a model hierarchy $(m_k, G_k), k \in [1, l]$, and an upper bound D, does there exist a representation $R \in G_l$ such that $R^l = O$ and $L(R) \leq D$?*

To see that the above decision problem is in NP, all we need to do is exhibit the efficient witness of a YES answer. This witness is clearly the representation R itself. By the assumptions 1 and 2, we can efficiently compute what this represents at every level. So, we can see that it indeed does have $R^l = O$ in polynomial time. Membership in G_l is also easy to check by assumption 3. So, this decision problem has been shown now to be in the class NP.

Now, the question remains in the mind whether this framework allows us to consider intractable problems, i.e. is it NP-Hard? The answer to this question is again YES. We show this by a reduction from `MINIMAL-SET-COVER` [3], a known NP-Complete problem.

The MINIMAL-SET-COVER problem (Figure 2-4) is as follows

Problem 2 *Given a finite set X and a family $f_i \in F$ of subsets $f_i \subseteq X$ satisfying $X = \bigcup F$, Find a subset $C \subseteq F$ such that $X = \bigcup C$ and $|C|$ is minimized.*

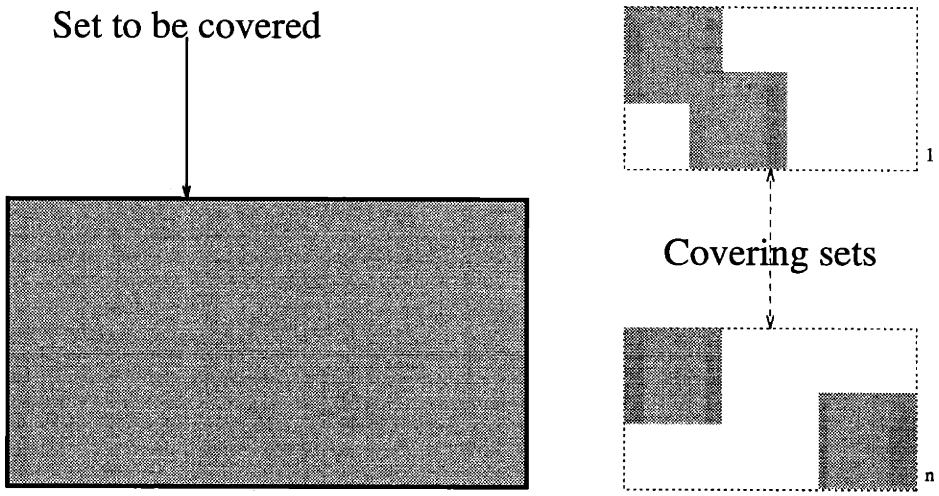

Figure 2-4: The MINIMAL-SET-COVER problem

We map into a hierarchy with only 1 level as follows. Let $E_0 = X$ and let the composition operation $+$ just be ordinary set union. Let m be defined as follows: $m(\theta) = f_\theta$ where θ is just encoded as a fixed $\log_2|\mathrm{F}|$ length string[6]. We just let G be all of 2^{Z^+}, so we accept any representation. Finally, we set $O = X$.

Now, it is clear that a valid solution $R = \theta_1, \theta_2, \ldots, \theta_n$ to this problem, is just an enumeration of indices of the $f_i \in C$s such that $\bigcup C = X$. Since the length of the representation R is just $\log_2|\mathrm{F}|$ times $|C|$, an optimal solution of one tells us the optimal solution to the other. The reduction clearly takes polynomial time and so we have succeeded in showing that the framework we have given is NP-complete in general.

2.5.1 Complexity Discussion

Now that we have established that solving problems in this framework is in general as hard as solving any other NP-complete problem, should we be disappointed?

[6]So we see that all we have is a *style parameter*, there are no *fit parameters*

It seems the answer to this question is a definite NO. As Ristad shows in [7], the most important of recognition problems, the understanding of human natural language, must contain NP-complete elements. In particular, he argues that determining the reference of anaphoric pronouns is NP-complete. So, if our framework were not capable of handling NP-complete problems we should reject it as being too weak to tackle interesting problems.

The important part of the complexity analysis is that under the relatively mild conditions given above, the complexity of problems in this framework will be no more than NP. This is intuitively plausible as well. We would like recognition problems to possess efficient witnesses, this squares well with the fact that once a human recognizes something, the now known answer becomes clear.

2.6 General Discussion

Now that we have sketched the broad outlines of our general framework, it will be good to review what is good about it. The first advantage is its ability to incorporate *a-priori* knowledge of the problem domain in a clean and principled manner. We simply build it in to the models. Second, the framework has a strongly generative feel to it. This is desirable because it corresponds well to the results being obtained in Linguistics. It holds out the potential of being able to unify natural language understanding and acquisition with all other human perceptual tasks. Furthermore, on the "input" end of the framework, we can use partial information, at whatever level, whenever it is available to us by incorporating it into the constraint sets G and the base-level observation. On the "output" end, we are free to take as output the representation at any level of the hierarchy, not just the top.[7]

Finally, it is appealing on philosophical grounds. This framework takes as a given that things appear to be "what they are" not because they "are *really*" that thing and they possess the distinctive, if sometimes hidden, features, that mark that thing. Rather, it says that the recognition of things is a purely mental process the outcome of

[7]In other words, we will make the choice of what is relevant output based on other considerations.

which is crucially determined by the (mostly) innate and learned mental models. We avoid postulating the existence of special "-ness" qualities like "fiveness, chairness, etc..." in the world, keeping the abstractions where they belong, in the minds of the recognizers.

In this general approach, the idea is to make the "innate" models do most of the work. The role of what is traditionally called learning is in comparison, minor. Rather than focusing on training systems with thousands if not millions of samples, the focus is on building in the right generative models. This accords well with the empirical facts about how children can acquire a word in their language upon a single exposure. Closer to home, when children are taught how to read and write, it doesn't take a thousand or even a hundred examples of a number for them to be able to recognize it. Show them how to make a few fives and they can do recognize most all of what adults will call fives with little further help.

What we have built in this section is a theoretical tool to *formally pose recognition problems in a way that incorporates our* a-priori *knowledge.* The proper way of evaluating the usefulness of this tool is to try and pose a real problem using it — and then to see how well it performs.[8] This is what is covered in the remainder of this thesis.

[8]In other words, how well do the right answers in the framework, the MDL answers, agree with the answers given by a human.

Chapter 3

Our Picture — Character Recognition

Now that we have seen some general properties of the framework that we are proposing, we introduce the problem on which we have chosen to test this framework: handwritten character recognition. In particular, we will look at a problem with only one level in the hierarchy - identifying single isolated handwritten numerals.

The data we used is from the Postal Service Zip-Code Database. It consists of 16x16 centered gray-scale images of isolated handwritten numerals[1] taken from real human-written[2] zip-codes and labeled with the "correct"[3] value. The database was already divided into two parts, a 7291 sample "Training set" and a 2007 sample "Test set."[4] We ignored the gray-scale information and decided to just pick an arbitrary threshold and work with the resulting binary-valued black and white images.[5] You can see a few sample characters and the effect of thresholding by examining Figure

[1] 0,1,2,3,4,5,6,7,8,9

[2] By many different writers. The data is quite variable.

[3] From what the humans thought the zip-code was.

[4] This division was not really used by us except that we developed the algorithm looking only at a few images from the "Training set" and present our results on images from the "Test set." The distinction is more relevant to more traditional approaches, such as connectionist models, that rely on extensive training.

[5] We did this because Akra's approach, based on domain-deformations only, could not handle gray-scale information. This work in this thesis has its roots in trying to test Akra's approach on a real-world difficult data set. In principle, our framework can handle gray-scale data as we discuss in a later section.

3-1 and Figure 3-2.

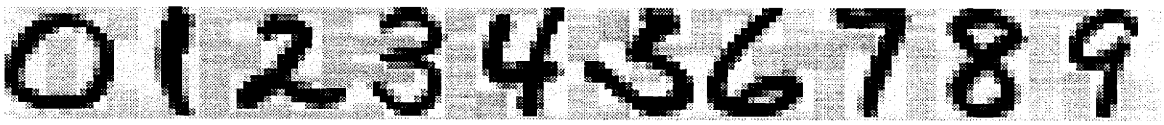

Figure 3-1: Sample gray-scale characters

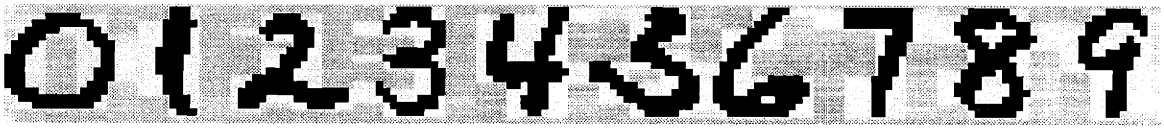

Figure 3-2: Sample thresholded characters

3.1 *A-priori* Knowledge

As we have discussed before, the advantage of our approach is that it enables a principled use of a-priori knowledge of the problem domain. In our case, our intuitive understanding of the handwriting process was as follows: We know that the writer will be writing exactly one character, which one is up to her. The writer has an idea of the form of the character that she will be tracing out. This can be thought of as an "ideal form" which is scaled, translated, rotated, and sheared[6] so as to fit the particular purposes of the writer. The writer also uses a reasonably fixed-width writing implement (pencil or pen) to write out the character.

In the process of writing out the character, the hand carries out the commands of the mind imperfectly, introducing some jitter as the character is traced out. This jitter has the general property that smaller displacements are more common than large ones.

Finally, in the process of scanning in the character, some pixels are turned on, and others are turned off. Once again, this noise is such that fewer pixels are flipped more often than many ones are.

[6]Of course, the writer can not use any affine transformation, only ones within a known limited range. But within that range, we have no reason to prefer one over the other.

3.2 The Model

We translated the above intuitions almost directly into the model we used. Our objective was to see how well we can do with just the simple straightforward way of translating these intuitions.

The first task is to identify the underlying set of elementary observations E_0. In our case, it is clear that this should be the set of binary-valued functions on a 16x16 grid, in our case identified with the lattice points of $[0, 15] \times [0, 15]$. The choice of $+$ is not that important in our problem, but to make things definite, let us choose the simple function `XOR` as usually defined on images.

The second task is to identify what will be a part of the *style parameters* and what will be a part of the *fit parameters*. From the above discussion, we know that the *style parameters* are the ones that are arbitrary at this level of understanding – we have no reason to prefer one value for them over another.[7] So, in this case, the *style parameters* are:

1. : The character to be drawn

2. : The "style" of the character — which affine transformation to use

3. : The width of the writing implement

As we know, all that remains are the *fit parameters*[8]:

1. : Where and how-much to jitter

2. : Which extra pixels were turned off

3. : Which extra pixels were turned on

[7]We can also think of these as being costless deformations. It is important to remember that while they are costless at this level, they will be costly at some higher level of the hierarchy that takes advantage of more information about how these particular characters were chosen. But at this level, there is no need to put a cost on them.

[8]We can also think of them as costly deformations. Then, we can think of the more costly deformations as representing less likely events.

Now that we know the basic parameters, it just remains to spell out how each of them is encoded. It should be clear that the encoding of the *style parameters* is mostly uninteresting since whatever it is, the code length remains constant.[9] So, we just used a straightforward uniform code. For example, we used 4 bits to decide which character (0–9) was to be drawn since $3 < \log_2 10 < 4$. The model then used this parameter to pick out an *ideal form* for the given character. This *ideal form* consisted of an ordered set of points in the Euclidean plane. This *ideal form* is then deformed by the specified affine transformation.

So, let us focus on the interesting *fit parameters*. First, let us address the easier numbers 2 and 3 above. Since there are 256 pixels in the 16x16 image, we can specify any one of them uniquely using $8 = \log_2 256$ bits. So, we used a simple two part code for these parts. First, we used a simple self-punctuated code for the natural numbers to tell how many pixels we were turning off. If there were n such pixels, this could be done in $2 * \log_2 n$ bits using a simple code using a 0 to mark the least significant bit and 1s to mark all other ones. For example, the number 3 can be represented as 1101. After the number n was encoded, it just remained to use $8 * n$ bits to tell which pixels to turn off. The identical code was used for which pixels to turn on[10].

The more interesting part is *fit parameter* number 1, the "jitter parameter." We decided to encode it as follows. We thought of the *ideal form* as hitting particular points in order. The jitter was just a displacement of each of these points. So, to tell the "jitter parameter," we had to tell the displacement vectors for each of the points in the *ideal form*. To do this, we first told the "maximum magnitude" of the displacement. Then, for every point in the *ideal form*, we told what displacement vector to apply to it.

Since we are talking about pixels here, we can think of everything as happening on the lattice points of the normal Euclidean plane. So, if we identify the zero dis-

[9]Since we know that in the current application there is only one character, the length of the *style parameters* is a constant added to the cost of each representation of the observed image — it does not affect MDL at all! If we were to consider things at a higher level which allowed multiple characters in a single image, then it would be much more relevant.

[10]alternatively, we can just think of both together as which pixels to flip — `XOR` with the image

placement vector with the point $(0,0)$, we can naturally associate every other lattice point with another displacement vector. Suppose that the maximum displacement is the vector (x,y)[11]. Now, if we draw a shaded closed disk in the plane, centered at the origin, with its circumference touching the point (x,y), then we know that every other displacement vector for the points of this *ideal form* must be associated with one of the M lattice points that is in this disk. So, in order to communicate the "maximum magnitude," all we need to encode is which disk we are talking about.

But, this is simple since the possible disks are all linearly ordered by radius and so they can be neatly enumerated. For example, disk 0 is the disk of radius 0, disk 1 is of radius 1, disk 2 is of radius $\sqrt{2}$, etc. For these disks, the Ms are also easy to compute. For example, $M_0 = 1$, $M_1 = 5$, $M_2 = 9$, etc. So, we just give the number r of the radius using the same simple self-punctuating code we use above for telling the number of pixels to flip in parts 2 and 3, and then can just use $N * \log_2 \mathrm{M_r}$ bits to tell all of the jitter vectors where N is the number of points in the *ideal form* and $\log_2 \mathrm{M_r}$ is the number of bits needed to tell the displacement vector of any one point given that we know the maximum displacement.

With the above encoding, all that we have to specify is the constraint set G. In our case, we merely say that it is all the representations R that consist of only a single valid θ since we know *a-priori* that we are only recognizing a single character.

So, in this formulation, the right answer to the handwritten character recognition problem is just the *style parameter* encoding which character to be drawn (number 1 in the list of *style parameters* above) in the shortest θ such that $m(\theta) = O$, where O is the binary-valued image to be recognized. So,

$$\mathrm{RightAnswer}(O) = \mathrm{WhichCharacter}(\mathrm{MDL}(\mathrm{O}|\mathrm{m}, \mathrm{G})) \tag{3.1}$$

[11]So, $\sqrt{x^2 + y^2}$ is the single sided Haussdorf distance from the deformed *ideal form* to the observed image

3.2.1 Extensions And Refinements

Many refinements are possible to this simple model. To show how these can be addressed, we consider a simple one — extending this model to deal with gray-scale images.

The first thing to do is to change the underlying elementary observation set so as to make it possible to represent gray-scale. Instead of binary-valued functions, we allow functions that take values in a range, say 0 to 255 inclusive.

Now, we just take our existing model and raise it to the next higher level of the hierarchy. Then, to go from binary-valued images to gray ones, we just implement a simple model with a single 256 bit *style parameter*[12], and one *fit parameter* for every pixel in the image, 256 *fit parameters* in total. If a given pixel's binary value (given in the *style parameter*) is a 0, we think of it as being a light colored pixel and we use a self-punctuating code (the corresponding *fit parameter*) for the gray value of that pixel. If it is a 1, we think of it as being a dark colored pixel and we use (255 - a self punctuating code) for the gray value. Basically, this expresses a preference for light-colored pixels to be light and dark-colored pixels to be dark.

This may not be the optimum extension to gray-scale[13], but it shows how it can be done in principle.

[12]Since the higher level model produces as its output a 16x16 binary-valued image.

[13]In fact, it probably isn't. It does not take into account the *a-priori* knowledge that we have that says that pixels at the center of the stroke will tend to be darker than pixels at the outer edges.

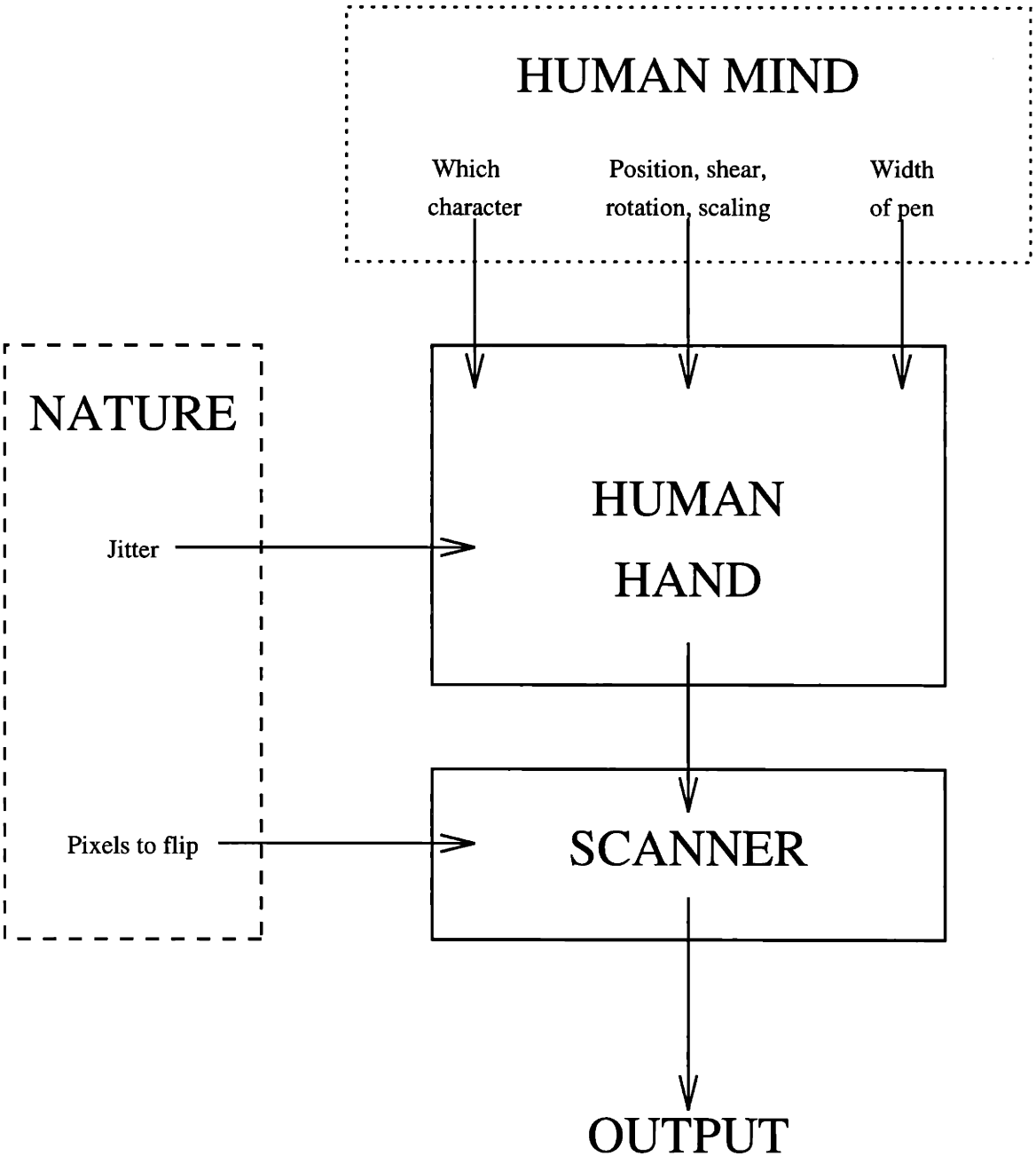

Figure 3-3: Our knowledge of the writing process

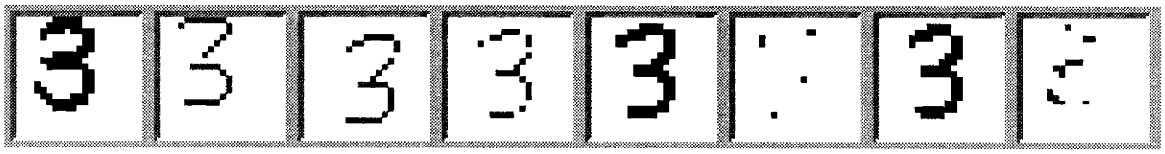

Figure 3-4: A picture of the encoding in action: The first image is the image to be recognized. The second represents the *ideal form* (style 1). Next, the affinely deformed *ideal form* (style 2). Then, the "jittered" deformed *ideal form* (fit 1). The next one takes into account the width of the pen (style 3). Next, the pixels to be turned off (fit 2). Then, what happens when we turn them off. Finally, the pixels to be turned on (fit 3).

Chapter 4

Implementation and Results

4.1 Experimental Set Up

To implement the model presented in the previous section, a recognition-workbench shown in Figure 4-1 was developed on a Pentium workstation running Linux 1.2.13 and X11R6. The tool was written in C++ and used TCL as its command language with TK as its graphical widget set. It was designed with ease of use and extensibility in mind. The code was not optimized for running-time and "brute-force" exhaustive search algorithms were used for the optimization of description length. The goal of the experiment was merely to establish a kind of "proof of concept" and to build a work-bench software environment that would allow easy exploration of possible models and optimization techniques.

The *ideal forms* shown in Figure 4-2 of the characters were hand generated in an *ad-hoc* fashion after brief visual examination of some of the training data in the data set. In keeping with the philosophy of this approach, we only used 14 of these *ideal forms.*[1] As you can see, we had only one *ideal form* each for the 0,1,5,6,7,9 and two each for the 2,3,4,8. The range of acceptable affine transformations was also determined in an ad-hoc fashion after brief visual examination of the training data. No

[1] We draw inspiration from two sources. The first is our anecdotal knowledge that one does not need to show a child a thousand copies of the number "1" for the child to be able to recognize it, one or two examples are enough for the child to do quite well. The second is the well-documented ability of children to acquire words of natural-language in a single hearing of a word. [5]

systematic use was made of the training data in the generation of models or acceptable transformations.

The experiment was run against the first 420 samples in the test data.

4.2 Results

The overall results of the experiment were promising. In comparison with Akra's recognition rate of 40%, we were able to achieve recognition rates of 86% with only the 14 *ideal forms* shown in Figure 4-2. We can see the result of the algorithm on a few selected samples in Figure 4-3. Each row shows the output for a different sample character image. Within each row, the first (leftmost) image is of the image together with its label. The rest of the entries in the row are for the given models, sorted by the minimal number of bits needed to represent the image using that model. For each of the models, three numbers are given. The first is the label of the model. The second is the single-sided Haussdorf distance from the affinely deformed *ideal form* representing the model and the observed image. The third is the number of bits needed to encode the observed image using this model. In the picture, you can see the deformed *ideal form* superimposed on the image of the character to be recognized.

If we don't just look at the single best model and instead allow some ambiguity, the performance numbers become more interesting. We will consider ambiguity in the following manner — rather than just selecting the best model (in terms of bits needed to represent the given image), we will select a set of models. Presumably, the ambiguity at this level will be resolved by a higher level. So, we will consider ourselves to have correctly recognized the image if the "true" model[2] is a member of the ambiguous-set for the image. The first way that we will define the ambiguous-set is to fix its cardinality, k. Then, we will pick the best k models and say that they are in the ambiguous-set. This is a traditional way of looking at ambiguity and it just uses the bit-length as a tool for defining a total-ordering over the models relative to the image. Graphically, we see its performance in Figure 4-4. The dashed line represents

[2]The one selected by the human

the best existing algorithm's performance on this data set.

Defining the ambiguous-set in this manner is not our only choice. We can make better use of the information at hand. Rather than just choosing a fixed number k of models, we can consider the set of models relative to whom the representation of the image is within N bits of the minimum length representation. The recognition performance using this definition of the ambiguous-set is shown in Figure 4-5 and in Table 4.1. We can compare this to the previous method of defining the ambiguous-set by looking at Figure 4-6 which shows us that the resulting ambiguous-sets are not too big on average as shown in Table 4.2[3]. In fact, we found that just by going to $N = 60$ bits, we were able to boost recognition performance (in that we were getting the right answer within that 60 bit window) to 97% while still only having an average of 1.7 candidates within this 60 bit window! This is particularly impressive when you consider that some of the best existing algorithms for this problem, statistical algorithms that used all 7291 training points and full gray-scale information, were also performing in the 95% range.[4] The best existing algorithm achieved a recognition rate of 97.4%[5] using all 7291 training points, gray-scale information, and *a-priori* knowledge[6] in the form of the distance-measure used — the Tangent distance[7].[8]

But, is taking a "window" of N bits meaningful? It seems the answer is a clear yes. To see why, we must think of this problem as existing within a hierarchical framework. We are looking at the lowest level models of the presumed hierarchy — the level of individual characters. Because there presumably exist higher levels (words, sentences, etc.), it does not make sense to only consider the single representation at this level with minimal description length, but rather a set of possible ones. Why should these "windows" be measured out in bits and not in other ways like "top M choices" and

[3]It is not clear that taking the average is actually meaningful in this case. We suspect that perhaps some notion of the "typical" set-size will be more useful. This is a topic of ongoing work.

[4]Humans perform in the 98% range given the gray-scale data. Human performance data on the binary-valued images is not available.

[5]To be fair, this is when the algorithm was asked to produce what it thought the one best answer was. How well it would do with ambiguity sets is unknown.

[6]They also used the fact that characters' classifications are mostly invariant under small affine transformations

[7]Basically, this approximated small affine transformations by a hyper-plane tangent to the 256 dimensional vector representation of the image

Recognition Performance											
Ambiguity	Total	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
0 bits	0.86	0.94	0.93	0.85	0.96	0.80	0.52	0.98	1.00	0.77	0.85
10 bits	0.87	0.94	0.95	0.88	0.96	0.83	0.57	0.98	1.00	0.77	0.85
20 bits	0.90	0.95	0.95	0.90	0.96	0.87	0.71	0.98	1.00	0.77	0.87
30 bits	0.92	0.95	1.00	0.90	1.00	0.87	0.71	1.00	1.00	0.83	0.92
40 bits	0.93	0.95	1.00	0.95	1.00	0.90	0.71	1.00	1.00	0.86	0.92
50 bits	0.96	0.95	1.00	0.98	1.00	0.97	0.86	1.00	1.00	0.89	0.92
60 bits	0.97	0.97	1.00	0.98	1.00	0.97	0.90	1.00	1.00	0.91	0.95
70 bits	0.98	0.97	1.00	0.98	1.00	0.97	1.00	1.00	1.00	0.97	0.97
80 bits	0.99	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	0.97	0.97

Table 4.1: Recognition performance (fraction correct) for each individual character as the ambiguity window varies.

Average Ambiguous-Set Size											
Ambiguity	Total	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
0 bits	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10 bits	1.1	1.0	1.0	1.1	1.1	1.2	1.2	1.0	1.0	1.2	1.0
20 bits	1.2	1.1	1.0	1.2	1.1	1.3	1.5	1.0	1.0	1.5	1.1
30 bits	1.3	1.1	1.1	1.3	1.1	1.5	1.6	1.1	1.0	1.8	1.2
40 bits	1.4	1.1	1.1	1.6	1.2	1.7	1.7	1.1	1.0	2.3	1.3
50 bits	1.6	1.2	1.1	1.9	1.2	1.9	2.0	1.2	1.0	2.7	1.5
60 bits	1.7	1.3	1.1	2.2	1.2	2.1	2.2	1.3	1.0	3.1	1.7
70 bits	1.9	1.3	1.3	2.4	1.3	2.4	2.7	1.4	1.0	3.6	2.0
80 bits	2.2	1.4	1.4	2.8	1.4	2.6	2.9	1.4	1.0	4.3	2.4

Table 4.2: Mean ambiguous-set size for each individual character as the ambiguity window varies.

the like? The answer comes down again to the fact that it is bits that are the common currency that we have in this framework. The choice of N comes from the specifics on the models (or presumed models) at the higher level. For example, suppose that we were trying to identify postal zip-codes. Suppose that we know that this higher level is expected to have up to a 60 bit variation in the length of the encoding of the *fit parameters* that are local to this level.[8] This tells us that it is certainly safe to only look at answers for individual characters from the lower level that are within 60 bits of the minimum description length value. This suggests a way that these formulations can be used in practice to "prune" choices in a provably safe way.

We now take a closer look at the classification errors that we made. Typical examples of classification errors can be seen in Figure 4-7 and Figure 4-8. When we look at the performance in Table 4.1, we see that our errors were due mainly to poor performance on a few numerals — in particularly a dismal 52% performance on the number 5 (for which we had only 1 model). In contrast, our performance on the numeral 7 was perfect at 100%. It seems that this problem was just due to the fact that we had too few *ideal forms* for certain characters. When we had used only one form for 8, its performance was also a dismal 51%. The addition of a single other form for '8' boosted performance up to 77%. It seems quite likely that the addition of even modestly more *ideal forms* will cause performance to improve dramatically.

[8]The *fit parameters* local to this level are the ones that help to define the *style parameters* at the next lower level — not the ones that are just copies of the *fit parameters* at the lower levels. In the language of this thesis, these truly higher level *fit parameters* are the θ_i^{fit}.

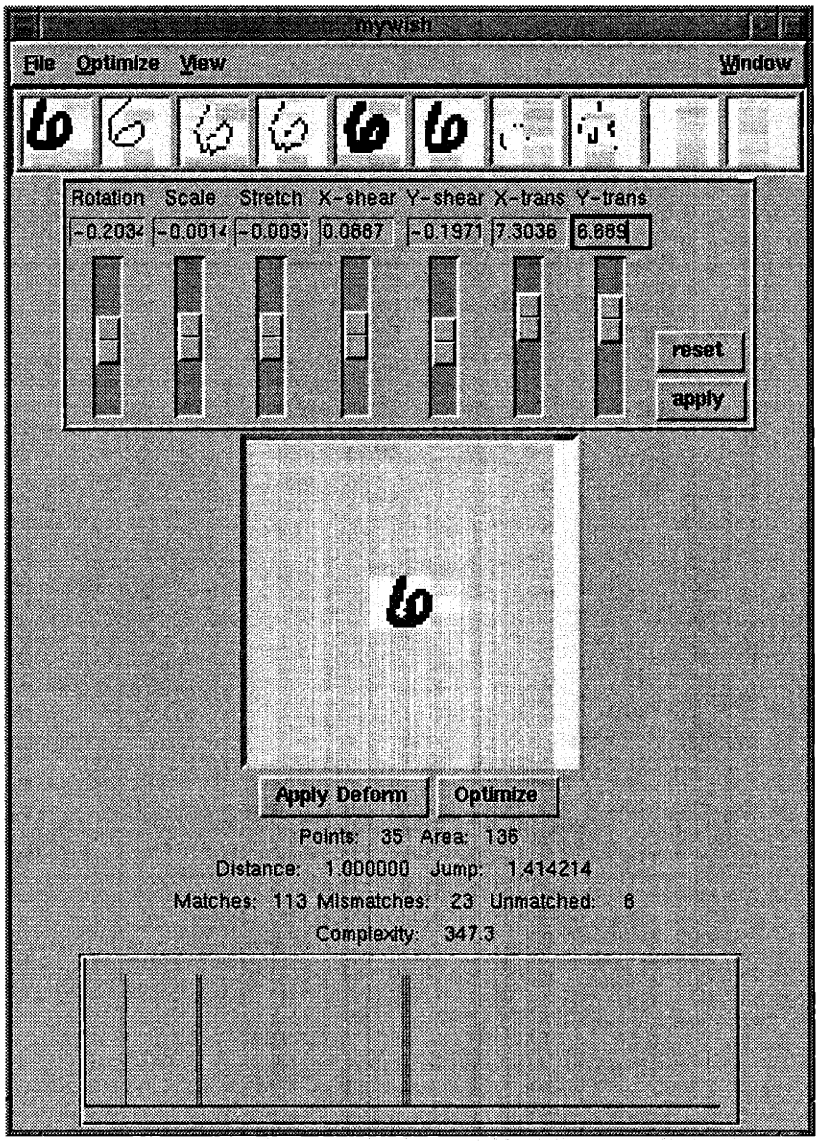

Figure 4-1: Our character recognition workbench

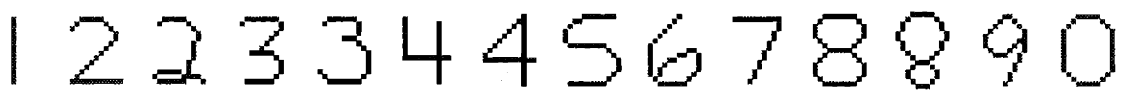

Figure 4-2: The ideal forms

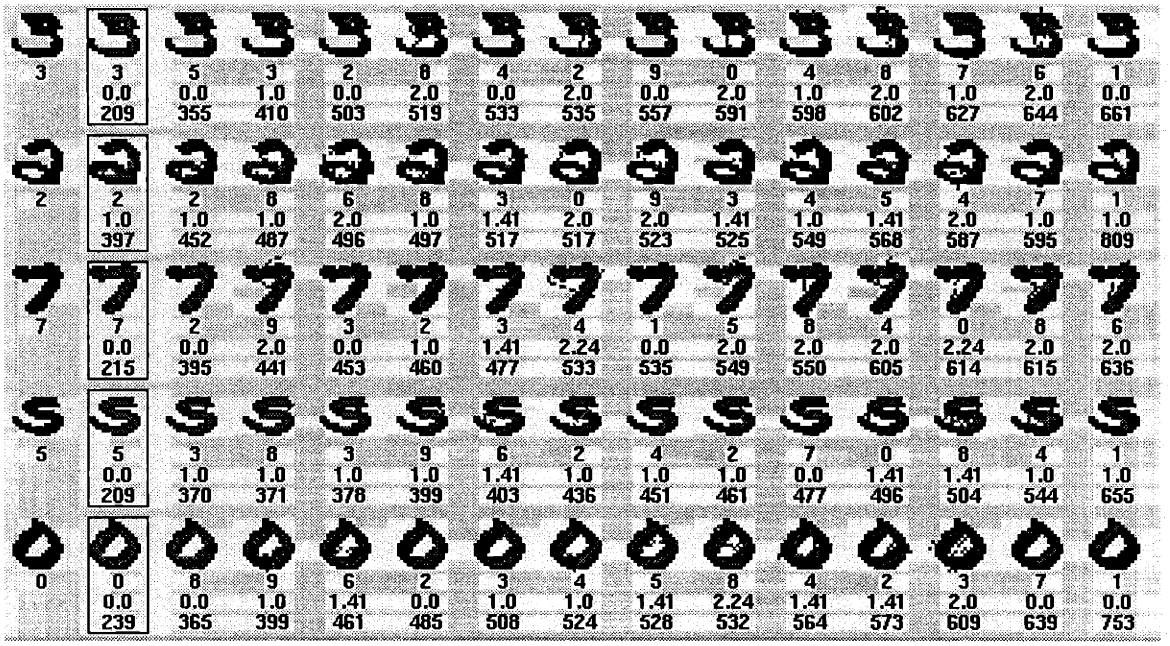

Figure 4-3: Recognition of selected characters

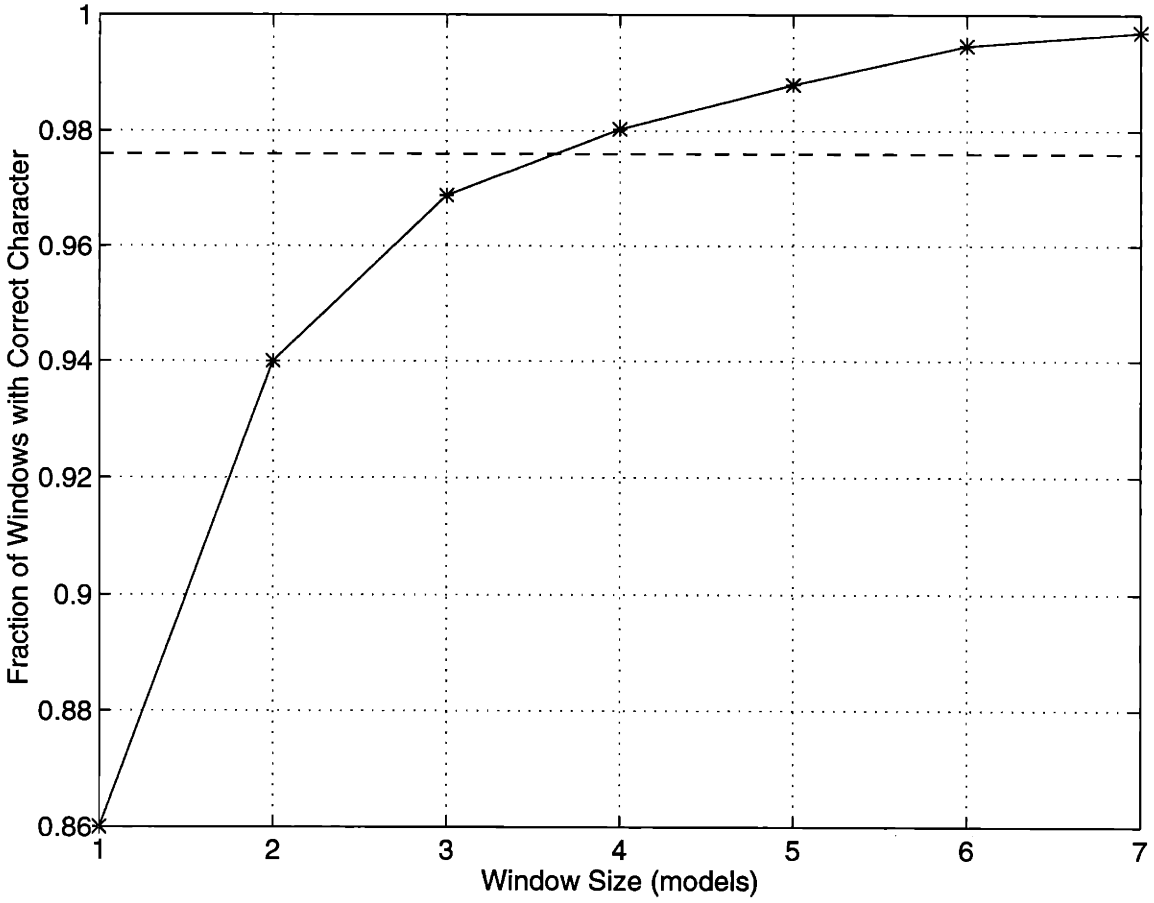

Figure 4-4: Recognition performance vs ambiguous-set size

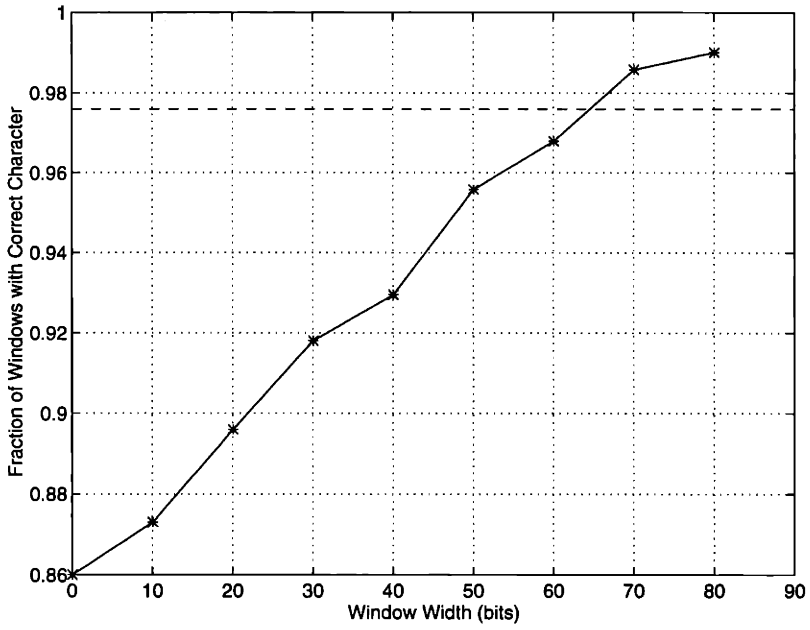

Figure 4-5: Recognition performance vs bit-length ambiguity

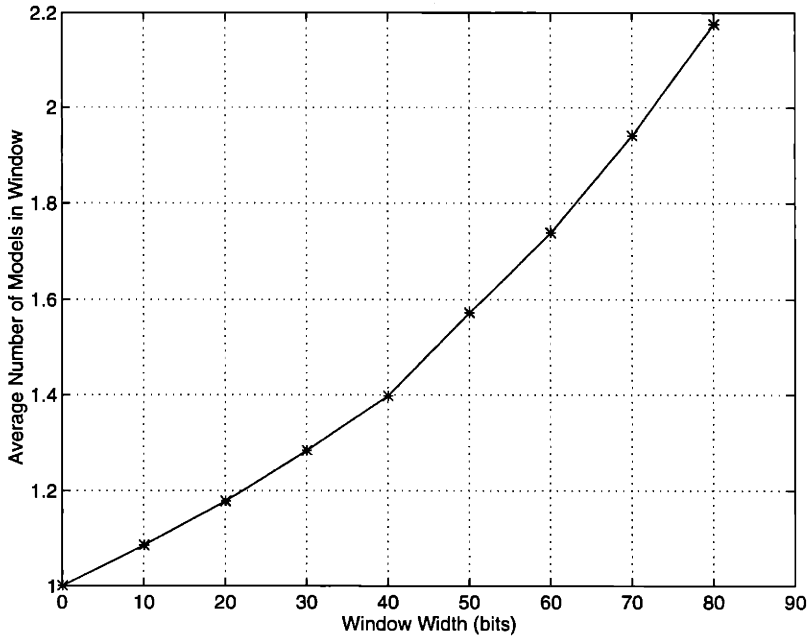

Figure 4-6: Average ambiguous-set size vs bit-length ambiguity

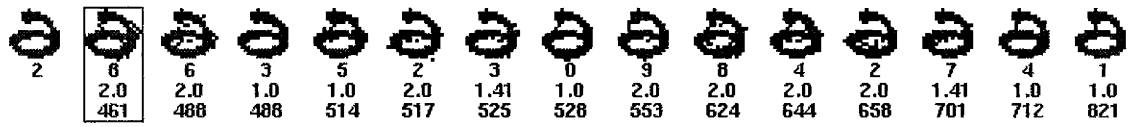

Figure 4-7: A sample 2 that was misclassified

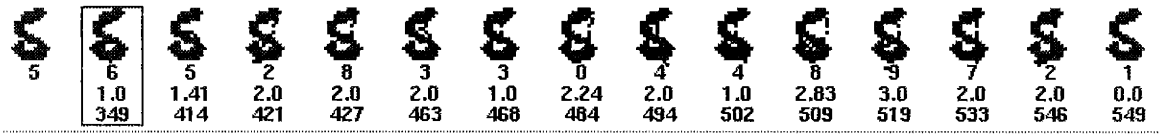

Figure 4-8: A sample 5 that was misclassified

Chapter 5

Conclusions

In this thesis, we have presented a unified framework in which to pose recognition problems. This framework has the advantage of allowing easy use of *a-priori* knowledge of the problem domain and it is compatible with the results coming out of the generative Linguistics[1] enterprise. We showed that the problems in the framework, were in general, NP-complete by nature. We then attacked the problem of handwritten character recognition in this framework and showed how the results were quite promising.

In future work, we expect to address the problem of recognition from another angle. While the approach in this thesis allows the systematic use of *a-priori* knowledge of the problem domain, it has no explicit way of dealing with any knowledge of the specific application[2] for which the recognition problem is being solved. This is currently under investigation.

In addition, we hope to further understand the framework and in particular come up with definite conditions under which the problem must be NP-complete and under what conditions a polynomial time deterministic algorithm is available. In those cases where a polynomial time algorithm is not available, approximation methods should be

[1] We intend to pursue these intriguing connections in future work.

[2] An example of such an application is machine vision for robotic navigation. One may want to avoid obstacles which are visually detected or navigate by landmarks. Presumably, knowledge of this application should affect the formulation of the recognition problems.

explored. Furthermore, the nature of the ambiguous-sets[3] and their potential impact on performance[4] needs to be better understood.

We also intend to systematically attack the problem of "learning" in this coding-based approach. The general idea is that what is usually called "learning" is just a matter of recognizing appropriate *style* parameters at a higher level in the modeling hierarchy. Once developed, the ideas of "learning" should allow us to make systematic use of noisy "training data."

So, along with the above theoretical work, the framework should be applied in a problem with more hierarchical levels. For example, we might try to attack the problem of handwritten word recognition. In addition to being able to study hierarchical modeling[5], we can then explore the interaction between hierarchical levels and the MDL principle. In particular, we notice that in our formulation, there is no communication from the higher levels to the lower levels in a way that depends on the particulars of the data being "recognized." It seems clear that to achieve any kind of computational efficiency, such "feedback" may be necessary to do early pruning and hence avoid a combinatorial explosion. This is something which needs to be studied in the context of a real application.

[3]Ambiguous-sets are defined in the discussion in 4.2.

[4]They may be used as a disciplined way to prune choices in a hierarchical formulation.

[5]How to do such modeling and quantifying the effect of simplifications in the models (undermodeling), is an interesting problem in its own right.

Bibliography

[1] Mohamed Akra. *Automated Text Recognition.* PhD thesis, Massachusetts Institute of Technology, 1995.

[2] Stefano Casadei. *Robust Detection of Curves in Images.* PhD thesis, Massachusetts Institute of Technology, 1995.

[3] M R Garey and D S Johnson. *Computers And Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman And Company, New York, New York, 1979.

[4] Stuart Geman. Compositional vision. Talk at MIT — Center for Intelligent Control Systems Review, October 1995.

[5] Jacques Mehler and Emmanuel Dupoux. *What Infants Know, The New Cognitive Science of Early Development.* Blackwell Publishers, Cambridge, Massachusetts, 1994.

[6] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, Singapore, 1989.

[7] Eric Sven Ristad. *The Language Complexity Game.* MIT Press, Cambridge, Massachusetts, 1993.

[8] P Y Simard, Y LeCun, and J Denker. *Advances in Neural Information Processing Systems*, chapter Efficient pattern recognition using a new transformation distance, pages 50–58. Morgan Kaufman, San Mateo, California, 1993.