Compositional Approach To Recognition Using Multi-Scale Computations

by

Shih-Hsiu Huang

B.A., Chung-Hsing University, 1987
Sc.M., Chung-Hsing University , 1989

Thesis

Submitted in partial fulfillment of the requirements for
the Degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

PROVIDENCE, RHODE ISLAND

May 2001

This dissertation by Shih-Hsiu Huang is accepted in its present form by the
Division of Applied Mathematics as satisfying the
dissertation requirement for the degree of
Doctor of Philosophy

Date..... .....
Stuart Geman

Recommended to the Graduate Council

Date..... .....
Elie Bienenstock

Date..... .....
David Mumford

Approved by the Graduate Council

Date..... .....

Contents

List of Figures

Chapter 1

Introduction

1.1 General Review

Children can recognize digits, letters and other things when they are very young. It seems that no matter how these objects are displayed– rotated or not– no matter what the quality of the scene is– with background noise or not– all can be easily recognized by children with high accuracy. However, once we want to build a machine with the same capacity, it becomes a very difficult task. Yet, it is very intriguing to "teach" a machine to imitate how people recognize objects. That is why pattern recognition has been a challenging and interesting field of study for decades.

The applications of pattern recognition are broad and extensive. They include character recognition, target detection, medical diagnosis and speech recognition. Therefore, many different techniques have been developed to solve these issues. According to Jain [14], these methods can be grouped into four general approaches: template matching, statistical approach, syntactic approach and neural networks.

Template matching is basically an operation used to determine the similarity between two objects. The assumption is that there are templates, or prototypes, available. The observed entity is then compared against these available templates. During the process of comparison, all allowable translations, rotations and scalings are taken into consideration. Then a correlation, or measure, is created to determine "how similar" they are. Based on this correlation, the identification of an observed object can be rendered. Overviews of this technology are in More [18] and Rosenfeld [26]. An example of this method can be found in Huttenlocher [13].

In the statistical approach, each pattern is represented in terms of d features or measurements, and is regarded as a point in a d-dimensional space. A feature set should be chosen so that pattern vectors, belonging to different classes, occupy compact and disjoint regions in a d-dimensional space. We need the boundaries not only to partition this space, but also to separate patterns belonging to different classes. The purpose of statistical pattern recognition is to determine to which category a given sample belongs. The design of a statistical pattern recognition system consists of two parts. The first part consists of collecting data samples from various classes, and finding the boundaries that separate different classes. This process is called training, or learning. These boundaries are established using

concepts from statistical decision theory. The second part is to test these boundaries by feeding the samples whose classes are known to this system. Introductions and reviews of statistical pattern recognition could be found in Chen [5] and Jain [14]. There are many recent articles devoted to the character/word recognition that basically utilize this approach. ([10],[15],[22],[23],[27])

When the involved patterns are complex and the number of descriptions is very large, it is more practical to adopt the syntactic, or structural, approach. In this approach, a pattern is viewed as a composition of simpler subpatterns, and these subpatterns themselves are furthermore composed of even simpler subpatterns (cf. Fu [11]). The simplest subpatterns are called *primitives*, or *terminals*. The original complicated entity is considered to be the "sum" of these primitives. The criteria for primitive selection can be found in the work of Pavlidis [24]. The term "syntactic" suggests an analogy between the syntax of a language and the structure of patterns. With this analogy in mind, patterns would be composed of primitives and the compositions are governed by rules which are like the grammatical rules in a language. There are many ways (cf. Fu [11]) to derive different grammars. These include induction, heuristic approach, and lattice structure. The recognition process of a pattern is accomplished as follows: after each primitive within this pattern is identified, a syntax analysis, or parsing of the *sentence* (describing this given pattern), is performed to determine whether or not it is syntactically correct with respect to the specified grammar. Through this parsing, a structural description of the given pattern can also be derived. Structural pattern recognition is attractive because, in addition to classification and structural description, a large collection of complex patterns can be described by a small number of primitives and grammatical rules. This approach can, therefore, express an infinite set of sentences in a very compact way. Fu's book [11] provides an excellent introduction and some applications of this field. Chan [4], Nishida [20], [21] and Potter [25] also use this approach.

The last approach is that of neural networks [19]. Serial digital computers and biological systems process information differently than one another. Computers rely on speed, accuracy, and on the ability to execute a vast amount of instructions. However, they are easily crippled by exponential algorithmic jobs. In contrast, the nature of biological systems

is a distributed parallel processing system, made up of large numbers of interconnected elementary processors of rather slow processing speeds. Inspired by biological systems, neural networks can be defined as : *a circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore each element operates asynchronously: thus there is no overall system clock (cf. [19]).* This method can be viewed as massively parallel computing systems consisting of a large number of simple processors with many interconnections. First we have a weighted directed graph, which simulates the artificial neural network, consisting of the nodes (neurons) and directed edges (the connections between neurons). This model tries to use some organizational principles, such as learning, generation, adaptivity, fault tolerance and distributed representation, in this simulated network(cf. [14]). The significance of a pattern is established by associations between a pattern (or a set of patterns) and other patterns (or sets of patterns).

These four approaches– template matching, statistical approach, syntactical approach and neural networks– are not necessarily mutually exclusive. In applications, these methods may be combined. For example, see (Amin [1], Cai [3], Lee [17] and Wong [28]).

1.2 Thesis Introduction

This thesis is devoted to utilizing the syntactic approach. First we are going to explain the concept of compositionality, an important philosophy in recognition.

Compositionality is generally considered to be fundamental to language (Chomsky [7] [8]). However, we believe that it is also crucial to cognition. In his essay on Probability [16], Laplace(1812) discussed the compositional nature of perception. He argues that when people see the string CONSTANTINOPLE, it is highly likely that it will be interpreted as a single word instead of a collection of fourteen letters. His hypothesis is reasonable because people recognize things by compositions. The same thing happens when a letter H is presented. It is more probable that it is perceived as an H, rather than three lines that happen to be there independently with these relative locations.

We need a mathematical background to develop this concept of composition. The tree structure is a useful way to describe the objects constructed from compositions. Furthermore we will put probability distribution on the tree structure. Finally, we want to know the conditions under which objects can group together to form a new object. This is the idea of binding rules.

In addition to the compositionality, there is also an issue concerning resolution. Imagine a very thick line and a very thin line. Both are straight. The major difference between them is the thickness. They are both lines according to our perception. The thick one belongs to low resolution and the thin one belongs to high resolution. But how can we give the machine the ability to recognize the two lines simultaneously in the composition system? The idea of multi-resolution computation is developed to accomplish this goal.

Because the number of this kind of composition objects is huge, we need a very efficient way to produce the more "useful" objects rather than to produce all the objects. A set of algorithms are developed to fit this need.

In order to illustrate these ideas, some handwritten English alphabets are tested. These letters are written on a digital tablet. The tablet samples the points and stores them in the form of x and y coordinates. The order of the input is irrelevant. Figure 1.1 illustrates some examples.

Chapter 2 of the thesis deals with the mathematical framework of the composition

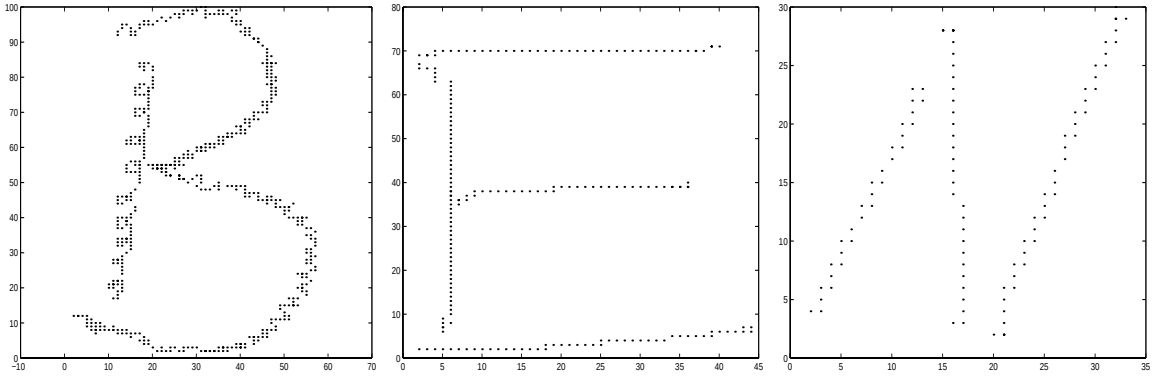

Figure 1.1: Some examples of the handwriting letters

systems. This framework includes the probability on trees, the relative coordinate system and the binding for objects.

Chapter 3 describes the cost function of interpretations of scenes. This cost function plays an important role when the interpretation of an image is needed. Algorithms are developed to optimize this cost function.

Chapter 4 is devoted to the experiments. The figures of the examples are presented. Some weak points of this approach are also discussed.

A summary of the thesis is contained in chapter 5.

Chapter 2

Mathematical Background

In this chapter, we will discuss the mathematical background of the composition system. The probability on the composition objects will be given in section 2.1. We will discuss the recursive definition of the probability on the labeled trees. The nodes in the tree, the binding of nodes, and the method of putting the probability on a tree will be outlined. The relative coordinate system and invariant composition rules, discussed in section 2.2, are very convenient in binding things. We need these to deal with "multi-scale" recognition. They also make the design of composition rules easy. In section 2.3, we will talk about the likelihood ratio when two objects bind together.

2.1 Probability on Labeled Trees

The details of this probability can be found in Geman [12].

The objects constructed from the composition rules can be described in terms of the structure of trees. Any object is either a terminal or a composition of its subtrees. Let's consider a *binary-image* example referring to an $N \times N$ array of binary–valued pixels with the pixels being the terminals. If we are dealing with an object horizontal linelet hl, this linelet hl should be a composition of two "horizontally adjacent" pixels. Thus this linelet in question is a tree with its two subtrees–both of them pixels. Furthermore, if we consider a line L_1, this line could be, but is not necessarily, a composition of a linelet and a pixel. Accordingly, L_1 is a tree with two subtrees– one is linelet, say l_2, and the other is a terminal. That l_2 is also a tree with 2 pixels as the subtrees.

To think about the tree structure, the set of primitives (or terminals) is needed. Different applications will employ different terminal sets. In the *binary-image* example, the set T of terminals could be, like we said, the $N \times N$ locations. In addition to the set of terminals, there are composite objects with labels, or types. Some labels, in our current example, are 'linelet', 'line', 'Letter-T', 'Letter-L', 'Letter-A' etc. We denote M the set of labels of "compound objects."

Needless to say, the composition involves the *binding function* B and *binding support* S. The binding function B and binding support S provide the information about the condition

whether we can combine objects together or not. The domain of B can be any n-tuple of objects. The range of B could be arbitrary as long as it fits the need of application. Through this thesis, we will focus on the binary case, meaning a composite tree has only two subtrees. In the horizontal linelet hl mentioned in the previous paragraph, we might expect the two "horizontally adjacent" pixels to be near one another, say one unit of length apart. One way to define this binding function B_{hl}(the subscript means it's for the horizontal linelet) could be $B_{hl}(p_1, p_2) = \overrightarrow{p_1 - p_2}$. The corresponding binding support $S_{hl} = \{(1, 0), (-1, 0)\}$. So if $B_{hl}(p_1, p_2) \in S_{hl}$, the two pixels p_1 and p_2 can bind together to form a horizontal linelet. By using the same concept, we can define many other binding functions and supports for a vertical linelet, vertical line, horizontal line, and so on.

Composition rules describe the same idea as the binding function and binding support. These rules govern entities to be composed to form composite entities. We rely on these rules to build objects. For example, if there are three binding rules: pixel + pixel $\rightarrow$ linelet, linelet + pixel $\rightarrow$ line, line + pixel $\rightarrow$ line, and their *corresponding* binding supports, we can build a 3-point line from pixel + pixel to linelet and linelet + pixel to a line. An even longer line can be constructed by using the line + pixel recursively. In general, if two objects α and β can bind together to form a composite object ω with label l, with the permission of a composition rule, we would write $\omega = l(\alpha, \beta)$.

Any collection of composition rules together with the set T of terminals define a set of objects, Ω. The set Ω is the set of trees such that for each non-terminal node n with label l there exists a composition rule under which the children of n can bind to form an object of type l. The set T of primitives is characterized as a set of single-node objects. Of course, T is a subset of Ω. We call the system composed of T, M and $\{(B_l, S_l)\}_{l \in M}$ a *composition system.*

Building on this idea of using the labeled trees to describe the objects constructed from the composition, it follows to wed the trees with the probability, because the probability will give us a tool to "measure" all these objects. And this tool is easily handled when we build the object from the bottom up. If the probability for each terminal is given, the probability of a composite object can be defined in terms of its subtree and the probability of each subtree can be defined likewise through the hierarchy of the whole tree structure

down to the very end, the primitive.

The probability P intended to put on the labeled trees needs a *label probability distribution* Q on $T \bigcup M$, $\sum_{l \in M} Q(l) + \sum_{t \in T} Q(t) = 1$, and, for each $l \in M$, a production probability distribution Q_l on S_l.

Since all the materials have been mentioned, we are in pretty good shape to give the definition of the probability on the set Ω.

Definition 2.1 (Probability on the Labeled Trees) *Let Ω be the set of trees and let $T \subset \Omega$ primitives(or terminals).*
Define $P : \Omega \to$ [0,1] a probability by

$$P(\omega) = \begin{cases} Q(\omega) & \text{if } \omega \in T \\ Q(l)Q(B_l(\alpha,\beta)|l)P \times P(\alpha,\beta|B_l(\alpha,\beta)) & \text{if } \omega = l(\alpha,\beta) \end{cases}$$

where B_l is a binding function,S_l is the binding support.
α and β can bind to form $l(\alpha,\beta)$ if $B_l(\alpha,\beta) \in S_l$.
$P \times P$ is the product probability distribution for the pair(α,β).

Remark 2.1 *The existence and uniqueness of a composition measure are not always guaranteed. If Ω is finite, the existence is proved by Chi [6]. One way to guarantee existence(and uniqueness) is to build the measure in the "bottom up" way. The details are in Geman [12].*

So we successfully describe the composite objects by labeled trees and put the probability distribution on these objects. Still, a systematic way to handle the B_l is very crucial. That is the goal of next section.

2.2 Relative Coordinate System and Invariant Composition Rules

From now on, we'll focus on the $N \times N$ grid. All the objects are in this area.

For each object in Ω, it can be assigned by the "absolute" coordinate system. It includes

the location x, size r and orientation θ. The location x can be a particular point in the $N \times N$ square. The size r could be the length if it is a line. It also could be the diagonal length if it is a rectangle. The orientation θ should be in the range of 0 and 2π. Although this coordinate system is very natural in determining an object, it does not offer much advantage in binding things. This is so because when the composition is under consideration, the relative position is much more important and meaningful than the absolute one. If we recall the example, horizontal linelet hl in the previous section, it is obvious that the relative position is given more attention than where(the absolute coordinate) the pixels are. This is the motivation of relative coordinate system. This coordinate was discussed in Potter [25].

Suppose there are two objects α and β, each has its own set of coordinate $\{x_\alpha, r_\alpha, \theta_\alpha\}$ and $\{x_\beta, r_\beta, \theta_\beta\}$ respectively. The relative coordinate system is defined as :

$$s = \frac{r_\beta}{r_\alpha} \tag{2.1}$$

$$v = R_{-\theta_\alpha} \frac{x_\beta - x_\alpha}{r_\alpha} \tag{2.2}$$

$$a = \theta_\beta - \theta_\alpha \tag{2.3}$$

where the R_θ is the rotation matrix $\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$.

So s is a *relative* size; v is a *relative* location; a is a *relative* orientation. Attention is paid to the relative coordinate *with respect to* object α rather than on the absolute coordinate of α and β. This relative coordinate system makes the invariant composition rules possible. Let's see an example which is similar to the one in Potter [25].

Example 2.1 (Letter-L) *Consider a pair of lines* $\{L_1, L_2\}$. *We can define the binding function* $B_{Letter-L}$ *for Letter-L as* $B_{Letter-L}(L_1, L_2) = (\frac{r_\beta}{r_\alpha}, R_{-\theta_\alpha} \frac{x_\beta - x_\alpha}{r_\alpha}, \theta_\beta - \theta_\alpha)$. *Also define the binding support* $S_{Letter-L} = [.8, 1.2] \times \{(x, y) : \sqrt{x^2 + y^2} \leq 0.1\} \times [3\pi/8, 5\pi/8]$. *Line* L_1 *and* L_2 *may be put together to form Letter-L(*L_1*,*L_2*) only if they are approximately equal lengths, have endpoints which lie near to each other, and form an approximately 90 degrees. Let's say s = 1.1, v = (0.1,0.1) and a =* $7\pi/15$ *for this* L_1 *and* L_2. *Then they may come together to form a Letter L according to the binding support. For the sake of invariance,*

consider another pair of lines $\{L_3, L_4\}$. *This new pair would produce another set of* s, v, a. *If these* s, v, a *values make* $B_{Letter-L}(L_3, L_4) \in S_{Letter-L}$, *then this* $\{L_3, L_4\}$ *should be able to combine to form a Letter-L no matter* L_3 *is* n *times* longer *than* L_1 *or* L_4 *is* m *times* shorter *than* L_2.

In the above example, we can see the advantage of the invariant binding rules. As long as we focus on the relative coordinate system, it is very beneficiary and convenient to design the binding functions and binding supports. However, to make the definition more plausible, the labels of the two objects should be taken into consideration. This is obvious because, in some occasions, *not* every pair of objects should be considered.

This leads to the binding function :

$$B_l(\alpha, \beta) \quad = \quad (L(\alpha), L(\beta), s, v, a) \tag{2.4}$$

where $L(.)$ is the label of the object.

In this example of Letter-L, the binding function $B_{Letter-L}$ should be $B_{Letter-L}(\alpha, \beta) = (L(\alpha), L(\beta), s, v, a)$ and the binding support is $(Line, Line) \times [.8, 1.2] \times \{(x, y) : \sqrt{x^2 + y^2} \leq 0.1\} \times [3\pi/8, 5\pi/8]$.

Using the invariant composition rules has some major advantages. Because the original size of objects is not as important as the relative "size" s, we can practice the "multi-scale" composition. It means this kind of composition rules allows us the do the multi-resolution recognition once we employ the relative coordinate and define binding rules accordingly. Another advantage is the the problem of alignment. The invariant composition rules would not treat the objects differently, no matter what the objects are in the middle or on the edge of the $N \times N$ square.

From the above argument, the reasonable conclusion would be : "Every time some new rule needs to be added into the composition system, the designer just has to make sure the binding function with the form of (2.4) and define the binding support appropriately." This is the essential core of *Invariant Composition Rules*.

We will focus on binding rules that are restricted to constituent labels and the relative coordinate, s, v, and a. There is an important disadvantage to this restriction which should be discussed before we move on. In particular, such binding rules may permit inappropriate or unintended compositions. When we consider Figure 2.1 and use the analogy[1] of 'rule 14' in Appendix A, it is apparent that the relative angle, a, is *independent* of the angle ϕ. Nevertheless, in everyday experience, the properness of the composition $\alpha + \beta \rightarrow$ 'F' depends very much on the extent to which angles a and ϕ are approximately the same. Hence in the example in Figure 2.1, there is no penalty for the mismatch in angles between the two horizontal lines. The difficulty is that we want to impose conditions on the angle of β with respect to *both* the vertical and the horizontal line in the L-junction, yet this is not possible if binding function depends only on relative coordinates. Thus although it is very handy to use the relative coordinate system, we must acknowledge that it still has limitations.

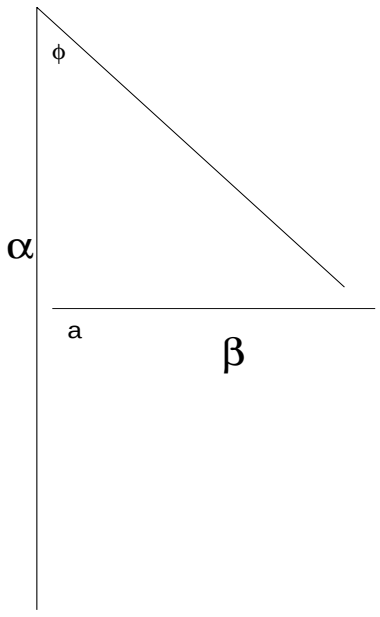

Figure 2.1: Example of limitation in binding rule

[1] We use points instead of disks as primitives in this example.

2.3 Likelihood Ratio and Bits Gained

Now it is a good time to ask the questions: What is the advantage of building the $P(\omega)$ from its subtrees? Does this probability give us the information when two objects bind together? If it does, what kind of information is that? How do the invariant composition rules fit into this picture?

The likelihood ratio would be introduced first. Suppose there are two objects α and β and suppose α and β can bind to form $\omega = l\ (\alpha,\beta)$. The likelihood ratio of the composition $l(\alpha,\beta)$ is the ratio of the probability of ω and the product of $P(\alpha)$ and $P(\beta)$. i.e., The likelihood ratio =

$$\frac{P(\omega)}{P(\alpha)P(\beta)}, \text{ if } \omega = l(\alpha,\beta). \tag{2.5}$$

We will assign encoding to different objects. In a Shannon code, the length of a code word $c(\xi)$ is $-log_2 P(\xi)$. Hence the composition saves $log_2 P(\omega) - log_2 P(\alpha) - log_2 P(\beta)$ bits. And this is exactly $log_2 \frac{P(\omega)}{P(\alpha)P(\beta)}$, the log of the likelihood ratio.

So let's take a look at what it provides. We will argue that this ratio is high above 1 if the recursive definition $P(\omega)$, discussed in section 2.1, is taken into consideration. When α and β combine to form an object ω, this composition saves

$$\begin{aligned}
& log_2(P(\omega) - log_2 P(\alpha) - log_2 P(\beta)) \\
= \ & log_2(\text{likelihood ratio}) \\
= \ & log_2(\frac{P(\omega)}{P(\alpha)P(\beta)}) \\
= \ & (\text{by definition 2.1})\ \frac{Q(l)Q(B_l(\alpha,\beta)|l)P \times P(\alpha,\beta|B_l(\alpha,\beta))}{P(\alpha)P(\beta)} \\
= \ & log_2 \frac{Q(l)Q(B_l(\alpha,\beta)|l)}{P \times P(B_l(\alpha,\beta))} \\
= \ & log_2 Q(l) + log_2 \frac{Q(B_l(\alpha,\beta)|l)}{P \times P(B_l(\alpha,\beta))}
\end{aligned}$$

bits. The first term, $log_2 Q(l)$, represents the cost of coding the label l of ω. As for the second term, it is the observed value of $B_l(\alpha,\beta)$ in the numerator and it is, in the denominator,

the product measure if α and β were to be chosen independently under P. We expect the numerator to be much bigger than the denominator. Consider the example 2.1(Letter-L) in the previous section. α and β are lines, and ω is the Letter-L with the constituents α and β. The binding function $B_{Letter-L}$ restricts α and β to be lines and restricts their relative position so that α and β form a Letter-L. It is clear that the likelihood of observing $B_l(\alpha, \beta)$ is far higher than that of observing α and β *independently* according to the probability P.

So the bits saved(or bits gained), the log of $\frac{P(\omega)}{P(\alpha)P(\beta)}$, provide the information about how "beneficiary" or "profitable" α and β combine together to construct ω. An *interpretation* is the assignment of each element of an image to an object. An *optimal* interpretation is an assignment that achieves the maximum bits gained. So to pursue the optimal interpretation is the driving force to the do the computation.

From now on, we switch to the continuum. The reason we are doing this is because we can rotate, translate and scale objects on the continuum. In the continuum, there is an idea, minimum size m, needed to be introduced. The minimum size m_α of an object α is the minimum allowable size of this object α in the continuum. Once the minimum size m_α is set, r_α is in the range of $[m_\alpha, \infty)$.

Definition 2.2 (Equivalent Class) *Two objects α_1 and α_2, with the same label, are of the same Equivalent Class if they preserve rotation, translation and scaling invariance.i.e, if the origins of α_1 and α_2 are put together, the two objects can match exactly with each other if the appropriate rotation is processed and if the scaling factor is imposed.*

The bits gained in the form of $log_2(Q(l)\frac{Q(B_l(\alpha,\beta)|l)}{P\times P(B_l(\alpha,\beta))})$ has to link with the relative coordinate system . We would rewrite this in term of the $\{s, v, a\}$. It will gives us a handy tool to calculate the bits gained once we are entering the relative coordinate system.

We would like to express $Q(B_l(\alpha, \beta)|l)$ and $P \times P(B_l(\alpha, \beta))$ in terms of the relative coordinate and the labels of α and β. This is reasonable when we remember the example 2.1 (Letter-L). In that example, we concluded that $B_l(\alpha, \beta) = (L(\alpha), L(\beta), s, v, a)$. We will extend this form to a more complete one.

Suppose the binding function B_l is of the form, $B_l(\alpha, \beta) = (G_l(\alpha_{eq}, \beta_{eq}), s, v, a)$. We

also suppose G_l depends *only* on labels. So

$$G_l(\alpha_{eq}, \beta_{eq}) = \sum_{i=1}^{n_l} g_i^l \sum_{j=1}^{m_i^l} 1_{L_{1j}^{l_i}}(L(\alpha_{eq})) 1_{L_{2j}^{l_i}}(L(\beta_{eq})) \tag{2.6}$$

where

$$\begin{aligned}
L(\alpha_{eq}) &: \text{the label of the equivalence class } \alpha_{eq} \\
n_l &: \text{the number of distinct values } G_l \text{ attains,} \\
g_1^l, g_2^l, g_3^l, ..., g_{n_l}^l &: \text{the distinct values attained by } G_l, \\
m_i^l &: \text{the number of label pairs } L(\alpha_{eq}), L(\beta_{eq}) \text{ that achieve } g_i^l, \\
(L_{1j}^{l_i}, L_{2j}^{l_i}) &: \text{the } j\text{-th label pair that achieves } g_i^l.
\end{aligned}$$

Fix $\omega = l(\alpha, \beta)$ with $G_l(\alpha_{eq}, \beta_{eq}) = g_i^l$, likelihood ratio $= Q(l)\frac{Q_l(g_i^l, s, v, a)}{P \times P(g_i^l, s, v, a)}$

The numerator is designed according to the application. We need the explicit form of $P \times P(g_i^l, s, v, a)$.

$$\begin{aligned}
& P \times P(G_l(\alpha_{eq}, \beta_{eq}) = g_i^l, s, v, a) \\
= & \sum_{j=1}^{m_i^l} P \times P(L(\alpha_{eq}) = L_{1j}^{l_i}, L(\beta_{eq}) = L_{2j}^{l_i}, s, v, a) \\
= & \sum_{j=1}^{m_i^l} P \times P(s, v, a | L(\alpha_{eq}) = L_{1j}^{l_i}, L(\beta_{eq}) = L_{2j}^{l_i}) P \times P(L(\alpha_{eq}) = L_{1j}^{l_i}, L(\beta_{eq}) = L_{2j}^{l_i}) \\
= & \sum_{j=1}^{m_i^l} P \times P(s, v, a | L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i}) P(L(\alpha) = L_{1j}^{l_i}) P(L(\beta) = L_{2j}^{l_i}) \\
= & \sum_{j=1}^{m_i^l} Q(L_{1j}^{l_i}) Q(L_{2j}^{l_i}) P \times P(s, v, a | L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i})
\end{aligned}$$

In the last equality, there are $Q(L_{1j}^{l_i})$, $Q(L_{2j}^{l_i})$ and $P \times P(s, v, a | L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i})$. The values of Q terms can be assigned by the label probability distribution. One

way to assign it is to use the equally likely Q on all the labels. The crucial part is the $P \times P(s, v, a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i})$.

Since

$$\begin{aligned} & P \times P(s, v, a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i}) \\ = & \int_{m_\alpha, m_\beta} P \times P(s, v, a|L(\alpha) = L_{1j}^{l_i}, m_\alpha, L(\beta) = L_{2j}^{l_i}, m_\beta) P(m_\alpha|L_{1j}^{l_i}) P(m_\beta|L_{2j}^{l_i}) dm_\alpha dm_\beta \end{aligned}$$

where m_α and m_β are the minimum allowable sizes of α and β respectively.

It is obvious that we need $P(m_\alpha|L_{1j}^{l_i})$ and $P(m_\beta|L_{2j}^{l_i})$ to get $P \times P(s, v, a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i})$. However, to find the distribution on the minimum sizes of all the objects(of the same label) is beyond our current understanding. So we would use $P \times P(s, v, a|L(\alpha) = L_{1j}^{l_i}, m_\alpha, L(\beta) = L_{2j}^{l_i}, m_\beta)$ to approximate $P \times P(s, v, a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i})$ knowing that it may not be perfect.

Now we turn our attention to an $N \times N$ square and let P be denoted by the probability in definition (2.1) on this square. Assume there are two equivalent classes α_{eq} and β_{eq} with $L(\alpha_{eq}) = L_{1j}^{l_i}$ and $L(\beta_{eq}) = L_{2j}^{l_i}$. Also let the (absolute) coordinate system x, r, θ has a universal distribution as follows:

$$\begin{aligned} x &\sim U([0, N] \times [0, N]) \\ r &\sim \frac{2m^2}{r^3}, \\ \theta &\sim U(0, 2\pi). \end{aligned}$$

So the term $P \times P(s, v, a|L(\alpha) = L_{1j}^{l_i}, m_\alpha, L(\beta) = L_{2j}^{l_i}, m_\beta)$ becomes $P \times P(s, v, a|\alpha_{eq}, \beta_{eq})$. Here, m_α is the minimum size of α_{eq} and m_β is the minimum size of β_{eq}.

We need to discuss the $1/r^3$ law, which is used in the previous paragraph. It is claimed (ref. Geman[private communication]) that if the size of constituent follows the $1/r^3$ distribution, then the composed object also follows the same law. The advantage here is this : if we assign the $1/r^3$ distribution on the terminals, then the composition of two terminals also has the $1/r^3$ distribution. By doing so, all of the objects generated by the composition follow the very same rule. Obvious this law is very convenient because the $1/r^3$ can be

inherited from the composition if, in the beginning, the $1/r^3$ rule is given to the members in the set T of terminals. It turns out that this "universal law" makes the computation of bits gained much easier. So if the size of the terminal follows this rule, it is guaranteed that the composition of terminals follows this rule. So all the composed objects follow this law. And all the bits gained can be handled the same way. From the following theorem and the approximation mentioned above, we will get an explicit form of likelihood ratio for composition in terms of the labels and the relative coordinate s, v, a.

Theorem 2.1 *Suppose an $N \times N$ square and let P_N be denoted by the probability in definition (2.1) on this square. Assume there are two equivalent classes α_{eq} and β_{eq}. Also let the (absolute) coordinate system x, r, θ has the distributions as follows:*

$$\begin{aligned} x &\sim U([0,N] \times [0,N]) \\ r &\sim \frac{2m^2}{r^3}, \\ \theta &\sim U(0, 2\pi). \end{aligned}$$

m is the minimum allowable size of r. We assume x, r, θ are independent with each other. Let the relative coordinate system defined by(same as (2.1) - (2.3))

$$\begin{aligned} s &= \frac{r_\beta}{r_\alpha} \\ v &= R_{-\theta_\alpha} \frac{x_\beta - x_\alpha}{r_\alpha} \\ a &= \theta_\beta - \theta_\alpha. \end{aligned}$$

We have

$$N^2 \, P_N \times P_N(s, v, a | \alpha_{eq}, \beta_{eq}) \to \frac{1}{2\pi} \frac{2}{s^3} min\{m_\beta^2, m_\alpha^2 s^2\} \, as \, N \to \infty \tag{2.7}$$

i.e.,

$$P_N \times P_N(s, v, a | \alpha_{eq}, \beta_{eq}) \cong \frac{1}{N^2} \frac{1}{2\pi} \frac{2}{s^3} min\{m_\beta^2, m_\alpha^2 s^2\} \; when \; N \; is \; large. \tag{2.8}$$

Remark 2.2 *According to the distribution of r, it is entirely possible that objects can stick*

out of the $N \times N$ square. However, this theorem is dealing with the limiting case. So when N is getting larger, given the distribution on $r \sim \frac{1}{r^3}$, the mass of "big" r would be smaller and smaller. So this theorem provides a good approximation.

Proof: The density function

$$P_N \times P_N(x_\alpha, r_\alpha, \theta_\alpha, x_\beta, r_\beta, \theta_\beta)$$

is

$$\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\alpha^2}{r_\alpha^3}\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\beta^2}{r_\beta^3}$$

How is this density function linked with the result we are trying to prove? We will take a look at the following equality.

$$\int\int\int\int\int\int \frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\alpha^2}{r_\alpha^3}\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\beta^2}{r_\beta^3} dx_\alpha dr_\alpha d\theta_\alpha dx_\beta dr_\beta d\theta_\beta = 1$$

The above is obvious because it is just the integration of the density function. Using the change of variables, this term turns out

$$\int\int\int\int\int\int \frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\alpha^2}{r_\alpha^3}\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\beta^2}{s^3 r_\alpha^3} J dx_\alpha dr_\alpha d\theta_\alpha ds dv da$$

$$\text{where Jacobian } J = r_\alpha^3$$

$$= \int\int\int\int\int\int \frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\alpha^2}{r_\alpha^3}\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\beta^2}{s^3} dx_\alpha dr_\alpha d\theta_\alpha ds dv da$$

Consider the integrand $\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\alpha^2}{r_\alpha^3}\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\beta^2}{s^3}$. If we integrate this function with respect to $x_\alpha r_\alpha \theta_\alpha$, then the result would be the density function in term of s, v, a. Also, this result is what we need in this theorem. So

$$P_N \times P_N(s, v, a|\alpha_{eq}, \beta_{eq}) =$$

$$\int_{\theta_\alpha}\int_{r_\alpha}\int_{x_\alpha} \frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\alpha^2}{r_\alpha^3}\frac{1}{N^2}\frac{1}{2\pi}\frac{2m_\beta^2}{s^3} dx_\alpha dr_\alpha d\theta_\alpha. \tag{2.9}$$

Equation 2.9 equals

$$\frac{1}{N^2}\frac{1}{2\pi}\frac{1}{N^2}\frac{1}{2\pi}4m_\alpha^2 m_\beta^2\frac{1}{s^3}\int_{\theta_\alpha}\int_{r_\alpha}\int_{x_\alpha}\frac{1}{r_\alpha^3}dx_\alpha dr_\alpha d\theta_\alpha.$$

This involves the area of integration. Let us define the area

$A_{x_\alpha,r_\alpha,\theta_\alpha} = \{(x_\alpha, r_\alpha, \theta_\alpha) : x_\alpha \in [0,N]\times[0,N] \text{ and } x_\beta = (x_\alpha + r_A v R_{\theta_\alpha}) \in [0,N]\times[0,N]\}$.
This area is for the origins of constituents α and β to stay in the $N \times N$ square. So this area $A_{x_\alpha,r_\alpha,\theta_\alpha}$ is definite between the following two regions $A^-_{x_\alpha,r_\alpha,\theta_\alpha}$ and $A^+_{x_\alpha,r_\alpha,\theta_\alpha}$.
$A^-_{x_\alpha,r_\alpha,\theta_\alpha} =$
$\{(x_\alpha, r_\alpha, \theta_\alpha) : x_\alpha \in [|v|\sqrt{N}, N-|v|\sqrt{N}]^2, r_\alpha \in [max\{m_\alpha, \frac{m_\beta}{s}\}, \sqrt{N}], \theta_\alpha \in [0,2\pi]\}$.
$A^+_{x_\alpha,r_\alpha,\theta_\alpha} = \{(x_\alpha, r_\alpha, \theta_\alpha) : x_\alpha \in N^2, r_\alpha \in [max\{m_\alpha, \frac{m_\beta}{s}\}, \infty], \theta_\alpha \in [0,2\pi]\}$.

$$\begin{aligned}
\text{Let } I_1 &= \int\int\int_{A_{x_\alpha,r_\alpha,\theta_\alpha}} \frac{1}{r_\alpha^3}dx_\alpha dr_\alpha d\theta_\alpha. \text{ We have } I_2 < I_1 < I_3.\\
\text{where } I_2 &= \int\int\int_{A^-_{x_\alpha,r_\alpha,\theta_\alpha}} \frac{1}{r_\alpha^3}dx_\alpha dr_\alpha d\theta_\alpha.\\
\text{and } I_3 &= \int\int\int_{A^+_{x_\alpha,r_\alpha,\theta_\alpha}} \frac{1}{r_\alpha^3}dx_\alpha dr_\alpha d\theta_\alpha.
\end{aligned}$$

$$\begin{aligned}
I_2 &= \int_{\theta_\alpha\in[0,2\pi]}\int_{r_\alpha\in[max\{m_\alpha,\frac{m_\beta}{s}\},\sqrt{N}]} \frac{1}{r_\alpha^3}(N-2|v|\sqrt{N})^2 dr_\alpha d\theta_\alpha\\
&= (N-2|v|\sqrt{N})^2\int_{\theta_\alpha\in[0,2\pi]} -\frac{1}{2}\frac{1}{r_\alpha^2}|^{\sqrt{N}}_{r_\alpha=max\{m_\alpha,\frac{m_\beta}{s}\}} d\theta_\alpha\\
&= (N-2|v|\sqrt{N})^2\int_{\theta_\alpha\in[0,2\pi]} \frac{1}{2}[\frac{1}{max\{m_\alpha^2, \frac{m_\beta^2}{s^2}\}} - \frac{1}{N}]d\theta_\alpha\\
&= (N-2|v|\sqrt{N})^2\frac{1}{2}2\pi[min\{\frac{1}{m_\alpha^2}, \frac{s^2}{m_\beta^2}\} - \frac{1}{N}].
\end{aligned}$$

$$\begin{aligned}
&\text{So } P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq})\\
\geq\ & (N-2|v|\sqrt{N})^2\frac{1}{2}2\pi[min\{\frac{1}{m_\alpha^2}, \frac{s^2}{m_\beta^2}\} - \frac{1}{N}]\frac{1}{N^2}\frac{1}{2\pi}\frac{1}{N^2}\frac{1}{2\pi}4m_\alpha^2 m_\beta^2\frac{1}{s^3}\\
=\ & \frac{1}{N^2}(1-\frac{4|v|}{\sqrt{N}}+4|v|^2\frac{1}{N})\frac{1}{2\pi}\frac{2}{s^3}[min\{m_\beta^2, s^2 m_\alpha^2\} - \frac{m_\alpha^2 m_\beta^2}{N}]
\end{aligned}$$

$$\Rightarrow N^2 P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq}) \geq (1-\frac{4|v|}{\sqrt{N}}+4|v|^2\frac{1}{N})\frac{1}{2\pi}\frac{2}{s^3}[min\{m_\beta^2, s^2 m_\alpha^2\}-\frac{m_\alpha^2 m_\beta^2}{N}].....E_1$$

$$\text{Apparently } I_3 = N^2 2\pi \frac{1}{2} min\{\frac{1}{m_\alpha^2}, \frac{s^2}{m_\beta^2}\}$$

$$\begin{aligned} & \text{So } P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq}) \\ \leq\ & \frac{1}{N^2}\frac{1}{2\pi}\frac{1}{N^2}\frac{1}{2\pi}4m_\alpha^2 m_\beta^2\frac{1}{s^3}N^2 2\pi\frac{1}{2}min\{\frac{1}{m_\alpha^2},\frac{s^2}{m_\beta^2}\} \\ =\ & \frac{1}{N^2}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\} \end{aligned}$$

$$\Rightarrow N^2 P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq}) \leq \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}\ .....E_2$$

From E_1 and E_2,

$$(1-\frac{4|v|}{\sqrt{N}}+4|v|^2\frac{1}{N})\frac{1}{2\pi}\frac{2}{s^3}[min\{m_\beta^2, s^2 m_\alpha^2\}-\frac{m_\alpha^2 m_\beta^2}{N}] \leq N^2 P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq})$$

$$\leq \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}$$

$$\text{So } N_2 P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq}) \rightarrow \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\} \text{ as N } \rightarrow \infty$$

$$\text{Thus } P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq}) \cong \frac{1}{N^2}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}$$

Q.E.D.

Now we have proved that if the composition rule depends on s, v, a, then $P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq})$ can be approximated by $\frac{1}{N^2}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}$. Still, there are other cases in which the binding rules do *not* depend on the quantity s.

Consider two lines combine to make an L-junction(not Letter-L). Since L-junction is not Letter-L, it should not be expected that the lengths(r_α and r_β) of two lines are roughly equal. So the binding rule no longer depends on the relative size s.

Furthermore, what is the allowable range of s? Since

$$\begin{aligned} s &= \frac{r_\beta}{r_\alpha} \\ &= \frac{|v| r_\beta}{|x_\beta - x_\alpha|} \text{ because } v = R_{-\theta_\alpha} \frac{x_\beta - x_\alpha}{r_\alpha} \end{aligned}$$

So if v is fixed, knowing that $|x_\beta - x_\alpha|$ can not be that big, the value of s could go to ∞ as r_β goes to ∞. On the other hand, the maximum of $|x_\beta - x_\alpha|$ should be $\sqrt{2}N$ because we are using the $N \times N$ square. So the minimum s is $\frac{m_\beta |v|}{\sqrt{2}N}$. It turns out that s is between $\frac{m_\beta |v|}{\sqrt{2}N}$ and ∞. Now we can discuss the $P_N \times P_N(v, a|\alpha_{eq}, \beta_{eq})$ in the following theorem.

Theorem 2.2 *Given the same assumptions in theorem 2.1 and if the binding function B_l is independent of the ratio of the sizes, we have*

$$\frac{N^2}{lnN} P_N \times P_N(v, a|\alpha_{eq}, \beta_{eq}) \to \frac{1}{2\pi} 2m_\alpha^2 \text{ as } N \to \infty, \text{ for all } a \text{ and all } v \neq 0. \tag{2.10}$$

i.e., for all a and all $v \neq 0$

$$P_N \times P_N(v, a|\alpha_{eq}, \beta_{eq}) \cong \frac{1}{N^2} \frac{1}{2\pi} 2m_\alpha^2 lnN \quad \text{when } N \text{ is large.} \tag{2.11}$$

Proof :

Recall, in (2.9), if s, v and a are fixed

$$\begin{aligned} P_N \times P_N(s, v, a|\alpha_{eq}, \beta_{eq}) &= \int_{\theta_\alpha} \int_{r_\alpha} \int_{x_\alpha} \frac{1}{N^2} \frac{1}{2\pi} \frac{2m_\alpha^2}{r_\alpha^3} \frac{1}{N^2} \frac{1}{2\pi} \frac{2m_\beta^2}{s^3} dx_\alpha dr_\alpha d\theta_\alpha \\ &= \frac{1}{N^2} \frac{1}{2\pi} \frac{1}{N^2} \frac{1}{2\pi} 4m_\alpha^2 4m_\beta^2 \frac{1}{s^3} \int_{\theta_\alpha} \int_{r_\alpha} \int_{x_\alpha} \frac{1}{r_\alpha^3} dx_\alpha dr_\alpha d\theta_\alpha. \end{aligned}$$

$$\text{Let } I_4 = \int_{\theta_\alpha} \int_{r_\alpha} \int_{x_\alpha} \frac{1}{r_\alpha^3} dx_\alpha dr_\alpha d\theta_\alpha.$$

We have the inequality $I_5 < I_4 < I_6$, where $K_N = \frac{N}{(lnN)^{1/3}}$, and

$$\begin{aligned}
I_5 &= \int_{\theta_\alpha\in[0,2\pi]}\int_{r_\alpha\in[max\{m_\alpha,\frac{m_\beta}{s}\},K_N]}\int_{x_\alpha\in[|v|K_N,N-|v|K_N]^2}\frac{1}{r_\alpha^3}dx_\alpha dr_\alpha d\theta_\alpha,\\
I_6 &= \int_{\theta_\alpha\in[0,2\pi]}\int_{r_\alpha\in[max\{m_\alpha,\frac{m_\beta}{s}\},\infty)}\int_{x_\alpha\in N^2}\frac{1}{r_\alpha^3}dx_\alpha dr_\alpha d\theta_\alpha\\
I_5 &= \int_{\theta_\alpha\in[0,2\pi]}\int_{r_\alpha\in[max\{m_\alpha,\frac{m_\beta}{s}\},K_N]}\frac{1}{r_\alpha^3}(N-2|v|K_N)^2 dr_\alpha d\theta_\alpha\\
&= (N-2|v|K_N)^2\int_{\theta_\alpha\in[0,2\pi]} -\frac{1}{2}\frac{1}{r_\alpha^2}|^{K_N}_{r_\alpha=max\{m_\alpha,\frac{m_\beta}{s}\}}d\theta_\alpha\\
&= (N-2|v|K_N)^2\int_{\theta_\alpha\in[0,2\pi]}\frac{1}{2}[\frac{1}{max\{m_\alpha^2,\frac{m_\beta^2}{s^2}\}}-\frac{1}{K_N^2}]d\theta_\alpha\\
&= (N-2|v|K_N)^2\frac{1}{2}2\pi[min\{\frac{1}{m_\alpha^2},\frac{s^2}{m_\beta^2}\}-\frac{1}{K_N^2}]\\
I_6 &= N^2 2\pi\frac{1}{2}min\{\frac{1}{m_\alpha^2},\frac{s^2}{m_\beta^2}\}
\end{aligned}$$

From the inequality ($I_5 < I_4 < I_6$) and multiply $P_N \times P_N$ by N^2, we get

$$I_5 N^2\frac{1}{N^2}\frac{1}{2\pi}\frac{1}{N^2}\frac{1}{2\pi}4m_\alpha^2 4m_\beta^2\frac{1}{s^3} < N^2 P_N\times P_N(s,v,a|\alpha_{eq},\beta_{eq}) < I_6 N^2\frac{1}{N^2}\frac{1}{2\pi}\frac{1}{N^2}\frac{1}{2\pi}4m_\alpha^2 4m_\beta^2\frac{1}{s^3}$$

$\Rightarrow$

$$(1-\frac{4|v|K_N}{N}+4|v|^2\frac{K_N^2}{N^2})\frac{1}{2\pi}\frac{2}{s^3}[min\{m_\beta^2,s^2m_\alpha^2\}-\frac{m_\alpha^2 m_\beta^2}{K_N^2}] \le N^2P_N\times P_N(s,v,a|\alpha_{eq},\beta_{eq}) \le \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2 s^2\}...E_3$$

So if $K_N = \frac{N}{(lnN)^{1/3}}$, $K_N \to \infty$ as $N\to\infty$, we have $N^2P_N\times P_N(s,v.a|\alpha_{eq},\beta_{eq}) \to \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2 s^2\}$ as $N\to\infty$. This is exactly the result of Theorem 2.1.

From E_3

$$(1-\frac{4|v|K_N}{N}+4|v|^2\frac{K_N^2}{N^2})\frac{1}{2\pi}\frac{2}{s^3}[min\{m_\beta^2,s^2m_\alpha^2\}-\frac{m_\alpha^2 m_\beta^2}{K_N^2}] \le N^2P_N\times P_N(s,v,a|\alpha_{eq},\beta_{eq}) \le \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2 s^2\}$$

$\Rightarrow$

$$(1-\frac{4|v|K_N}{N}+4|v|^2\frac{K_N^2}{N^2})(\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}-\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2})$$
$$\leq N^2P_N\times P_N(s,v,a|\alpha_{eq},\beta_{eq})\leq\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}$$

$\Rightarrow$

$$\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}-\frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}+\frac{4|v|^2K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}$$
$$-\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}+\frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}-\frac{4|v|^2K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}$$
$$\leq N^2P_N\times P_N(s,v,a|\alpha_{eq},\beta_{eq})\leq\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}$$

$\Rightarrow$

$$-\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}+\frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}-\frac{4|v|^2K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}+$$
$$\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}-\frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}+$$
$$\frac{4|v|^2K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}\geq-N^2P_N\times P_N(s,v,a|\alpha_{eq},\beta_{eq})\geq$$
$$-\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}$$

$\Rightarrow$

$$\frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}-\frac{4|v|^2K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}+\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}-$$
$$\frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}+\frac{4|v|^2K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2m_\beta^2}{K_N^2}\geq\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2,m_\alpha^2s^2\}$$
$$-N^2P_N\times P_N(s,v,a|\alpha_{eq},\beta_{eq})\geq0$$

$\Rightarrow$

$$|N^2 P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq}) - \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}| \leq \frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\} + \frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2 m_\beta^2}{K_N^2} + \frac{4|v|^2 K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2 m_\beta^2}{K_N^2}$$

$\Rightarrow$

$$|\int N^2 P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq})ds - \int \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}ds| \leq \int \frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}ds + \int \frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2 m_\beta^2}{K_N^2}ds + \int \frac{4|v|^2 K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2 m_\beta^2}{K_N^2}ds$$

As we observed earlier, the range of of s is $[\frac{m_\beta|v|}{\sqrt{2}N}, \infty]$ ($|v| \neq 0$)

$\Rightarrow$

$$|N^2 P_N \times P_N(v,a|\alpha_{eq},\beta_{eq}) - \int_{\frac{m_\beta|v|}{\sqrt{2}N}}^{\infty} \frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}ds| \leq \int_{\frac{m_\beta|v|}{\sqrt{2}N}}^{\infty} \frac{4|v|K_N}{N}\frac{1}{2\pi}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}ds + \int_{\frac{m_\beta|v|}{\sqrt{2}N}}^{\infty} \frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2 m_\beta^2}{K_N^2}ds + \int_{\frac{m_\beta|v|}{\sqrt{2}N}}^{\infty} \frac{4|v|^2 K_N^2}{N^2}\frac{1}{2\pi}\frac{2}{s^3}\frac{m_\alpha^2 m_\beta^2}{K_N^2}ds$$

$\Rightarrow$

$$|N^2 P_N \times P_N(v,a|\alpha_{eq},\beta_{eq}) - \frac{1}{2\pi}2m_\alpha^2[\frac{1}{2} + ln\sqrt{2} - lnm_\alpha + lnN - ln|v|]| \leq \frac{4|v|K_N}{N}\frac{1}{2\pi}2m_\alpha^2[\frac{1}{2} + ln\sqrt{2} - lnm_\alpha + lnN - ln|v|] + \frac{1}{2\pi}\frac{m_\alpha^2 m_\beta^2}{K_N^2}\frac{2N^2}{m_\beta^2|v|^2} + m_\alpha^2 m_\beta^2 \frac{4}{N^2}\frac{1}{2\pi}\frac{2N^2}{m_\beta^2}$$

Divided by lnN

$\Rightarrow$

$$|\frac{N^2}{lnN}P_N \times P_N(v,a|\alpha_{eq},\beta_{eq}) - \frac{1}{lnN}\frac{1}{2\pi}2m_\alpha^2[\frac{1}{2} + ln\sqrt{2} - lnm_\alpha + lnN - ln|v|]| \leq \{\frac{4|v|K_N}{N}\frac{1}{2\pi}2m_\alpha^2[\frac{1}{2} + ln\sqrt{2} - lnm_\alpha + lnN - ln|v|] + \frac{1}{2\pi}\frac{m_\alpha^2}{K_N^2}\frac{2N^2}{|v|^2} + \frac{8m_\alpha^2}{2\pi}\}/lnN$$

$\Rightarrow$

$$|\frac{N^2}{lnN}P_N \times P_N(v,a|\alpha_{eq},\beta_{eq}) - \frac{1}{2\pi}2m_\alpha^2| \leq \{\frac{4|v|K_N}{N}\frac{1}{2\pi}2m_\alpha^2[\frac{1}{2} + ln\sqrt{2} - lnm_\alpha + lnN - ln|v|]$$
$$+\frac{1}{2\pi}\frac{m_\alpha^2}{K_N^2}\frac{2N^2}{|v|^2} + \frac{8m_\alpha^2}{2\pi}\}/lnN + \frac{1}{lnN}\frac{1}{2\pi}2m_\alpha|\frac{1}{2} + ln\sqrt{2} - lnm_\alpha - ln|v||$$

$$= \{C_1\frac{K_N}{N}lnN + C_2\frac{K_N}{N} + C_3\frac{N^2}{K_N^2} + C_4\}/lnN \tag{2.12}$$

So if $K_N = \frac{N}{(lnN)^{1/3}}$, (2.12) $\to 0$ as $N \to \infty$

i.e.,

$$\frac{N^2}{lnN}P_n \times P_n(v,a|\alpha_{eq},\beta_{eq}) \to \frac{1}{2\pi}2m_\alpha^2$$

as N $\to \infty$

i.e. ,

$$P_N \times P_N(v,a|\alpha_{eq},\beta_{eq}) \cong \frac{1}{N^2}\frac{1}{2\pi}2m_\alpha^2 lnN$$

Q.E.D.

We use

$$P_N \times P_N(s,v,a|\alpha_{eq},\beta_{eq})$$

to approximate

$$P \times P(s,v,a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i}).$$

Thus

$$P \times P(s,v,a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i}) \cong \frac{1}{2\pi}\frac{1}{N^2}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}. \tag{2.13}$$

by Theorem 2.1.

Putting it all together, the likelihood ratio $\cong$

$$\frac{Q(l)Q_l(g_i^l, s, v, a)}{\sum_{j=1}^{m_i^l} Q(L_{1j}^{l_i})Q(L_{2j}^{l_i})\frac{1}{2\pi}\frac{1}{N^2}\frac{2}{s^3}min\{m_\beta^2, m_\alpha^2 s^2\}} \tag{2.14}$$

A special case : the binding function B_l *doesn't* depend on the quantity s. We use

$$P_N \times P_N(v, a|\alpha_{eq}, \beta_{eq})$$

to approximate

$$P \times P(v, a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i})$$

$$\text{So } P \times P(v, a|L(\alpha) = L_{1j}^{l_i}, L(\beta) = L_{2j}^{l_i}) \cong \frac{1}{2\pi}\frac{1}{N^2}2m_\alpha^2 lnN. \tag{2.15}$$

by Theorem 2.2. Again the likelihood ratio $\cong$

$$\frac{Q(l)Q_l(g_i^l, v, a)}{\sum_{j=1}^{m_i^l} Q(L_{1j}^{l_i})Q(L_{2j}^{l_i})\frac{1}{2\pi}\frac{1}{N^2}2m_\alpha^2 lnN} \tag{2.16}$$

Since the $Q_l(g_i^l, s, v, a)$ or $Q_l(g_i^l, v, a)$ are designed according to the application, then the bits gained should be easy to implement. Therefore we can use (2.14) and (2.16) to calculate the bits gained for the composition.

Chapter 3

Cost Function and the Algorithms of the Computation

Section 3.1 will be devoted to the cost function of an image. This function is to be optimized when we try to get the recognition for a certain scene. So the optimal interpretation would be the interpretation that optimizes the cost function. We will introduce the data model as well. In this way we can come up with the bits gained for a terminal and the bits gained for an composite can be derived accordingly. In section 3.2, we will discuss the algorithm of multi-scale recognition to optimize the cost function. A set of of lists will be employed to handle different resolutions at the same time. Each scale will be taken care of by a list. Since the *optimal interpretation* is an assignment of objects that achieves the maximum bits gained, it is crucial to derive the "correct" result from our lists. Section 3.2 will also describe the method to accomplish it. If an object has nowhere to grow under the current status, we call it "mature". This phenomenon will make the algorithm in section 3.2 unsuccessfully. Section 3.3 will talk about how to deal with this kind of object and let the algorithm carry on.

3.1 The Cost Function and Data Model

There is a shift that should be mentioned. Since the theorems in chapter 2 are in continuum terrain and we will be working on a discrete space, so we shift away from a continuum.

In chapter 2, we discussed the probability $P(\omega)$ defined on the set Ω of labeled trees. We can use this probability P to define the probability on interpretations. The details of this probability can be found in Geman [12].

Definition 3.1 *Let Ω be the set of trees and let $T \subset \Omega$ primitives. Suppose $P : \Omega \to$ [0,1] the probability defined in 2.1. If $\mathcal{I}$ is the collection of of all finite interpretations $I \subset \Omega$, we define the probability $D : \mathcal{I} \to$ [0,1] by*

$$D(I) = \frac{1}{Z} \prod_{\omega \in I} P(\omega) \tag{3.1}$$

with $Z = \sum_{I' \in \mathcal{I}} \prod_{\omega \in I'} P(\omega)$.

Remark 3.1 *Our goal is to get the largest probability D on a given image Y, i.e., we use this to find the optimal interpretation I of an image. This probability D gives us a good tool to pursue this. We will use this probability D to derive the cost function.*

Given a binary–image Y $= \{x_1, x_2, x_3, \cdots, x_{(N+1)\times(N+1)}\}$, $x_k \in \{0, 1\}$. We want to get the

$$arg\, max_{I\in\mathcal{I}}\, D(I|Y)$$

Since

$$D(I|Y) = \frac{D(Y|I)D(I)}{D(Y)},$$

so we just need

$$arg\, max_{I\in\mathcal{I}}\, D(Y|I)D(I) \tag{3.2}$$

because $D(Y)$ is constant for this given Y. So the cost function is $D(Y|I)D(I)$. The interpretation that maximizes this function is the optimal one.

We use disks as the primitives in the experiment. The centers of disks are on the grid points of an $N \times N$ square. The radii of these disks are from $r_1, r_2, \cdots, r_{max}$. So the set $T = \{(x, y, r) : x, y \in [0, N] \times [0, N],\ r \in \{r_1, r_2, \cdots, r_{max}\}\}$. The size of the set T is $(N + 1)^2 max$.

Let $\mathcal{A}(I)$ = union of the pixels' number in disks in the image I. i.e.,

$$\mathcal{A}(I) = \{\ i \in \{1, 2, 3, \cdots, (N + 1)^2\}\ \ :\ \ x_i \in \text{some disk in } I\ \}$$

The data model with the image is

$$D(Y|I) = \prod_{k\notin\mathcal{A}(I)} q^{x_k}(1 - q)^{1-x_k} \prod_{k\in\mathcal{A}(I)} p^{x_k}(1 - p)^{1-x_k} \tag{3.3}$$

where q is the probability of black points *without* disk and p is the probability of black points *with* disk. The probability q is a value close to 0 because it represents the "background" noise. The probability p, on the contrary, is much bigger because it is the "foreground" probability.

We use this $D(I)$ to derive the bits gained for a single disk. The result we get is the quotient between the values $D(Y|I)D(I)$ with no disk in the interpretation and with *only one* disk in the interpretation. If $I = \{\emptyset\}$, meaning the interpretation is nothing but noise, then according to equation (4.1)

$$\begin{aligned} D(I) &= \frac{1}{Z} \\ D(Y|I) &= \prod_{k=1}^{(N+1)^2} q^{x_k}(1-q)^{1-x_k} \end{aligned}$$

If $I' = \{\ \omega(\text{a disk})\ \}$, meaning the interpretation is the disk ω and the rest of the image is still noise, then

$$\begin{aligned} D(I') &= \frac{1}{Z}P(\omega) \\ &= \frac{1}{Z}P(\omega|T)P(T) \\ D(Y|I') &= \prod_{k \notin \mathcal{A}(I')} q^{x_k}(1-q)^{1-x_k} \prod_{k \in \mathcal{A}(I')} p^{x_k}(1-p)^{1-x_k} \end{aligned}$$

Recall the the label probability distribution Q on $T \bigcup M$, $\sum_{l \in M} Q(l) + \sum_{t \in T} Q(t) = 1$. If we assume there are K labels(including the terminal) and each label has the same mass in terms of this probability Q and we assume each terminal has the same mass, then

$$\begin{aligned} & \frac{1}{Z}P(\omega|T)P(T) \\ = \ & \frac{1}{Z|T|K} \end{aligned}$$

Here $K = |M| + 1$.

So if $I' = \{\ \omega(\text{a disk})\ \}$, we have

$$\begin{aligned} D(I') &= \frac{1}{Z}P(\omega) \\ &= \frac{1}{Z}P(\omega|T)P(T) \\ &= \frac{1}{Z|T|K} \\ D(Y|I') &= \prod_{k\notin\mathcal{A}(I')} q^{x_k}(1-q)^{1-x_k} \prod_{k\in\mathcal{A}(I')} p^{x_k}(1-p)^{1-x_k} \end{aligned}$$

How many bits are gained when we add *only* the disk ω in our interpretation? We use the likelihood ratio between $D(Y|I')D(I')$ and $D(Y|I)D(I)$. So the bits gained for this disk ω are

$$log_2 \frac{D(Y|I')D(I')}{D(Y|I)D(I)} = log_2 \frac{1}{|T|K} \frac{\prod_{k\in\mathcal{A}(I')} p^{x_k}(1-p)^{1-x_k}}{\prod_{k\in\mathcal{A}(I')} q^{x_k}(1-q)^{1-x_k}}$$

Thus the bits gained for a disk are

$$log_2\, p^{n_1}(1-p)^{m_1} - log_2\, q^{n_1}(1-q)^{m_1} - log_2|T| - log_2 K. \tag{3.4}$$

where n_1 is the number of black points and m_1 is the number of white points in the disk.

If $I'' = \{\ \omega_1,\ \omega_2\ \}$, meaning the interpretation is two disks ω_1, ω_2 and the rest of the image is still noise, then

$$\begin{aligned} D(I'') &= \frac{1}{Z}P(\omega_1)P(\omega_2) \\ &= \frac{1}{Z}P(\omega_1|T)P(T)P(\omega_2|T)P(T) \\ &= \frac{1}{Z|T|^2K^2} \\ D(Y|I'') &= \prod_{k\notin\mathcal{A}(I'')} q^{x_k}(1-q)^{1-x_k} \prod_{k\in\mathcal{A}(I'')} p^{x_k}(1-p)^{1-x_k} \end{aligned}$$

How many bits are gained when we add only the two disk ω_1 ω_2 in our interpretation? Still, we use the likelihood ratio between $D(Y|I'')D(I'')$ and $D(Y|I)D(I)$. So the bits gained

for the two disks ω_1 and ω_2 is

$$log_2 \frac{D(Y|I'')D(I'')}{D(Y|I)D(I)} = log_2 \frac{1}{|T|^2 K^2} \frac{\prod_{k \in \mathcal{A}(I'')} p^{x_k}(1-p)^{1-x_k}}{\prod_{k \in \mathcal{A}(I'')} q^{x_k}(1-q)^{1-x_k}}$$

Thus the bits gained for two disks is

$$log_2\, p^{n_2}(1-p)^{m_2} - log_2\, q^{n_2}(1-q)^{m_2} - 2\, log_2|T| - 2\, log_2 K. \tag{3.5}$$

where n_2 is the number of black points and m_2 is the number of white points in the *two* disks. Please notice that the overlapping points are counted only once.

So from equation(3.4), we can calculate the bits gained for each disk. If we have two disks to form a linelet, we use the bits gained in (3.5) plus bits gained in (2.14) to calculate the bits gained for this linelet. In general, the bits gained for a composite object will be computed in the same way.

So far, we have discussed the cost function of an image and the data model. Through the data model and cost function, the bits gained for an object can be derived. The next step would be "optimizing this cost function". This will be discussed in next section.

3.2 Multi-Scale Computation

The idea of Multi-Scale Computation is pretty straightforward. We will present the outline in the next paragraph and give the algorithm after that.

Suppose we have radii r_1,r_2,$\cdots$,r_{max}. These radii are for the terminal set T which was defined in the previous section. We use different lists to store the terminals with different radii. We use list_1 to store the disks of radius r_1 and list_2 the disks of radius r_2 and list_max the disks of radius r_{max}. There is another list, list_cross, for storing and building the cross-resolution objects. First, in list_1, we choose the most "promising" terminals o_{1i}, meaning the biggest bits gained disk. How do we do with this chosen disk o_{1i}?(This disk remains in the current list.) We try to combine every object, which are disks only, in its own list. If they do combine, we get a new object. We put this new object into the current

list. After this object o_{1i} has done every possible composition it can, the algorithm will go over to next list, the list_2, and choose the most promising disk o_{2i} in list_2. This o_{2i} also tries every composition it can for every object in its own list. After the first loop, from r_1 to r_{max}, is over, we go to the list_cross also choosing the object with the largest bits gained if there is any. But this time this object is allowed to combine *every* object in different lists. The newly built objects are stored in the list_cross. Then, we go back to list_1 choosing the object with the biggest bits gained, which could be a linelet. We also combine this linelet with everything it can compose in its own list. Therefore we do the computation through these lists(including list_cross) over and over again. The size of each list keeps growing and growing. In summary, we loop through each list choosing the object with the biggest bits gained, doing all the binding it can in its own list, creating new objects, putting new objects in lists. If, at anytime, a chosen object cannot build any new objects in its own list, this chosen object will be duplicated and the copy will be stored in list_cross. The list list_cross also participates in the loop for compositions.

We will state the algorithm in the following:

Assume there is a finite number of radii. The objects with smallest radius(finest resolution) is stored in $LIST\_1$ and the objects with next-to-smallest radius(coarser resolution) in $LIST\_2$ etc. The coarsest resolution is therefore stored in $LIST\_max$. Another list, $LIST\_cross$ is for the cross-composition objects. Furthermore, we also have a list, $MAIN\_LIST$, for *all* the objects. Therefore this $MAIN\_LIST$ is the union of $LIST\_1, LIST\_2, ..., LIST\_max$ and $LIST\_cross$. Suppose we already have a composition system $T, M, \{B_l, S_l\}_{l \in M}$, we put these primitives into the lists they belong.

$MAIN\_LIST$	$LIST\_1$	$LIST\_2$		$LIST\_max$	$LIST\_cross$
m_1	o_{11}	o_{21}		o_{max1}	o_{cross1}
m_2	o_{12}	o_{22}	$\cdots$	o_{max2}	o_{cross2}
m_3	o_{13}	o_{23}		o_{max3}	o_{cross2}
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$

Algorithm 3.1 *Multi-Scale Computation*

step

1. *Put each terminal in its corresponding LIST_k, k = 1,2,...max.*
2. *Put each terminal in MAIN_LIST.*
3. *Get the size MAIN_LIST_SIZE_OLD for MAIN_LIST.*
   Get the size LIST_k_SIZE_OLD for each LIST_k, k = 1,2,...max.
   Get the size LIST_cross_SIZE_OLD for LIST_cross. (which is zero at first.)
4. *for k = 1 to max, do step 5 to 9*
5. *Find o_{ki}, the object with biggest bits gained in LIST_k.*
6. $o_* \leftarrow o_{ki}$
7. *For j = 1 to LIST_k_SIZE_OLD, do step 8 to 9*
8. *Does there exist any $l \in M$ such that*
   a new object $\omega_{new} = l(o_*, o_{kj})$ or $\omega_{new} = l(o_{kj}, o_*)$?
9. *if yes, put ω_{new} into the LIST_k and MAIN_LIST*
10. *If o_* cannot have anything to compose in step 8, put a copy of o_* in LIST_cross.*
11. *Find o_{cross}, the object with biggest bits gained in LIST_cross.*
12. $o_* \leftarrow o_{cross}$
13. *For n = 1 to MAIN_LIST_SIZE_OLD, do step 14 to 15.*
14. *Does there exist any $l \in M$ such that*
    a new object $\omega_{new} = l(o_*, m_n)$ or $\omega_{new} = l(m_n, o_*)$?
15. *if yes, put ω_{new} into the LIST_cross and MAIN_LIST*
16. *Update LIST_k_SIZE_OLD for each k.*
    Update MAIN_LIST_SIZE_OLD and LIST_cross_SIZE_OLD.
17. *Go back to step 4.*

Remark 3.2 *This algorithm can deal with different resolutions simultaneously because it loops from the finest resolution to coarsest resolution. It is also easy to implement. The designer just has to determine what kind of terminal is desired, what binding rules should be applied, and what hierarchy is needed. Step 8 and 14 will check the availability of the binding very quickly because the number of B_l is not big and because relative coordinate system is easy to handle.*

How do we get the result from this algorithm? As one can see, this algorithm never stops because it can choose the object and build new objects. Even when the chosen object can not produce any new objects, the algorithm can be still running without any progress. We intend to get the result from the $MAIN\_LIST$. So we have to pause this computation when the user wants to get the interpretation from the machine. To pause this computation is easy in implementation. But how do we get the result from this algorithm once it is stopped? The greedy algorithm is used.

The idea of greedy algorithm is pretty simple. This algorithm chooses a subset from its collection, the $MAIN\_LIST$, by choosing successively the next best bits-gained object among those not chosen, until the original image is entirely assigned. The greedy algorithm is not slow, and can be restarted dozens of times. To restart means that we can have a different choice of the *first* chosen object. After the first chosen object, we successively choose the next best bits-gained object and so on. In the following algorithm, let $S = MAIN\_LIST$ and let $R =$ the assignment of the image.

Algorithm 3.2 *Greedy Algorithm*

step

1. *Let* $R = \emptyset$.
2. *Pick an object o with the maximun bits gained among* $S \backslash R$. *Let* $R = R \cup \{o\}$
3. *Recursively do step 2 until the the data in the image have been covered, either by primitives or by constituents.*

Remark 3.3 *In our experiment, the parameters are been set so that the disk with the smallest radius has positive bits gained if this disk has one black point inside. Because of this, step 3 will be processed. Also because of this, a perfect and a finest line can be recognized as such in the experiment.*

There is an issue about "sharing". It means that two objects can have the same part in common. For instance, two intersecting lines can share one disk. Mathematically, the probability that this phenomenon does happen would be zero if the continuum is taken into consideration. However, we do not have the right tools to handle this case. So in step 2,

we are allowed to choose an object that shares some parts in R though it is mathematically impossible. It turns out that it is too optimistic when we get the bits gained when the "sharing" of two (or more) objects happens because we double count the bits gained of the shared object(s).

To save time in executing Greedy Algorithm, it is necessary to do a sort for the $MAIN\_LIST$. A quick sort in Corman [9] is very convenient.

3.3 How to Deal With the "Mature" Object

According to the algorithm in the previous section, a problem will almost always occur. It happens when the object with the biggest bits gained, chosen from a certain $LIST\_k$, can *not* bind with any other object. We call this object "mature" Let's see how it happens.

Example 3.1 *Suppose we have T(the primitives) as the pixels, M={linelt,line,letter-L}, and four binding rules : pixel + pixel → linelet, linelet + pixel → line, line + pixel → line and line + line → letter-L. We also suppose there is an L made of dots in the image. According to the algorirhm, we pick up the pixel and aggregate it to a line, say* L_1*, which is one of the strokes in Lettet-L. Remember we always pick up the the biggest bits gained object, so the code will produce* L_1 exclusively*, leaving the other stroke behind. Now the code definitely picks up this line because it is the most promising one. It would be great if there is another stroke* L_2 *so that* L_1 *and* L_2 *con form a Letter L. However,* L_1 *has no such* L_2 *to combine with because during the process of creating* L_1*,* L_2 *is simply* <u>not</u> *there. So there is no way we can get the object Letter-L. This kind of object is called mature and its consequence will very likely prevent the code from building up the desired object, Letter-L.*

To resolve this problem, we need a method, called suppression, to handle it. It is like we get rid of the area(in the $N \times N$ square) we are working on and start somewhere else. The idea is to suppress the bits gained of any object which is physically contained in the mature

object. By doing so, next time the algorithm chooses the biggest bits gained object, it will choose the object elsewhere, not sticking with the matured object. Even though their bits gained are suppressed, these objects still can be combined with other objects.

Although we suppress the bits gained, it doesn't mean it will change how the greedy algorithm works. Our method is to introduce the "pseudo-bits-gained" of an object. The pseudo-bits-gained is the *original* bits gained we have being using, if no suppression is executed. If the suppression *does* happen, the pseudo-bits-gained is the original bits gained divided by a factor C bigger than 1. So when the greedy alrorithm is searching the result, it still works according to the algorithm 3.2, meaning it chooses the most *original* bits gained object etc. However, in algorithm 3.1, the way to choose the most promising object would be "choosing the most *pseudo-bits-gained* object" instead of the most *original* bits gained one.

We need some notations in presenting the "suppression" algorithm. The set of pixels in an object o is denoted by $S_{points}(o)$. The pseudo bits gained for an object o is $pseudo\_bits(o)$. The original bits gained for an object o is $bits(o)$.

Algorithm 3.3 *Suppression of Object's Bits Gained*

step

1. *If an object o_{mature} is mature, find the number k such that $o_{mature} \in LIST\_k$. This k could be 1, 2, 3, ... , max and cross. So the LIST_cross is handled the same way.*
2. *For every member o_k in LIST_k,*

   if $S_{points}(o_k) \subseteq S_{points}(o_{mature})$,

   then $pseudo\_bits(o_k) = bits(o_k)$ / C

Remark 3.4 *In the algorithm 3.3, the determination of C could depend on the mature object. One can choose the C such that none of the object which is physically contained in the mature object, o_{mature}, can be chosen next time. The determination of C could be, therefore, dynamic. However, one can always make C fixed and big enough so that the multi-scale algorithm would choose an object elsewhere and do the compositions.*

Remark 3.5 *The suppressed objects can still be combined with other chosen objects under*

guidance the binding rules. That is becuase these suppressed ones remain in these LISTS and the chosen objects have to go through the LIST to check whether they can combine or not. Please see steps 7 - 9 in Algorithm 3.1 for details.

Now we can apply algorithm 3.1, 3.2 and 3.3 to the example 3.1. Once the first stroke is built, its pseudo bits gained is suppressed. These objects physically contained in that stroke are also suppressed. So the code will turn to another place, picking the pixel away from the built stroke, building the second stroke. Remember the suppressed object still can bind with other objects. So the second stroke and the first stroke can combine to make a Letter-L.

The set of algorithms is very reasonable and easy to implement. In the next chapter, we will show the results of an experiment using these algorithms. Those results will demonstrate the validity of these ideas.

Chapter 4

Experiment

We implement the experiments by using the algorithms introduced in the previous chapter. The English capital alphabets will be tested. In section 4.1, we will talk about the composition rules for this experiment. These rules are designed for the upper-case letters. Section 4.2 is the results of this experiment. We will show some images we have tested. We will also show how they are interpreted. In section 4.3, we will talk about the cases that this algorithm fails.

4.1 The Composition Rules

The composition system is composed of T, M and $\{B_l, S_l\}_{l \in M}$. T is the set of terminals, M is the set of labels of "compound objects", and $\{B_l, S_l\}_{l \in M}$ are the composition rules. The set T, discussed in Section 3.1, is the set of disks (with different radii) all over the $N \times N$ square. In this experiment, seven radii are used. They are 0.5, 1, 2, 3, 4, 5, and 6. The set M will be presented in the following paragraph.

Since we want to recognize the capital alphabets in this experiment, this set M should be pretty much the labels for the letters. So the set M is {linelet, line, L-junction, T-junction, Letter-A, Letter-B, Letter-b, Letter-C(or curve), Letter-D, Letter-E, Letter-F, Letter-G_1, Letter-G_2, Letter-H, Letter-I, Letter-J, Letter-K, Letter-L, Letter-M, Letter-N, Letter-O, Letter-P, Letter-Q, Letter-R, Letter-S, Letter-T, Letter-U, Letter-V, Letter-W, Letter-X, Letter-Y }. The different Letter-Gs arise from different writing styles. Since our composition rules have the property of rotation invariance, many labels should be considered as the same. Like Letter-N and Letter-Z, they share the same structure. That is why we do not have the label Letter-Z. It is also true that $\dashv$, $\vdash$ and $\perp$ have the same label Letter-T. Similarly, $\vee$ and $\wedge$ have the same label Letter-V. With this rotation invariant property in mind, we discuss the binding rules for this experiment.

The binding rules are :

$$
\begin{aligned}
\text{linelet} &= \text{disk} + \text{disk} \\
\text{line} &= \text{linelet} + \text{disk} \\
\text{L-junction} &= \text{line} + \text{line} \\
\text{T-junction} &= \text{line} + \text{line} \\
\text{Letter-A} &= \text{Letter-V} + \text{line} \ \text{ or } \ \text{T-junction} + \text{line} \\
\text{Letter-B} &= \text{Letter-P} + \text{letter-C} \ \text{ or } \ \text{Letter-b} + \text{Letter-C} \\
\text{Letter-b} &= \text{Letter-C} + \text{line} \\
\text{Letter-C} &= \text{line} + \text{line} \ \text{ or } \ \text{Letter-C} + \text{line} \\
\text{Letter-D} &= \text{Letter-C} + \text{line} \\
\text{Letter-E} &= \text{Letter-F} + \text{line} \\
\text{Letter-F} &= \text{L-junction} + \text{line} \\
\text{Letter-G\_1} &= \text{Letter-C} + \text{line} \\
\text{Letter-G\_2} &= \text{Letter-C} + \text{Letter-L} \\
\text{Letter-H} &= \text{Letter-T} + \text{line} \\
\text{Letter-I} &= \text{T-junction} + \text{line} \\
\text{Letter-J} &= \text{Letter-C} + \text{line} \ \text{ or } \ \text{T-junction} + \text{Letter-C} \\
\text{Letter-K} &= \text{T-junction} + \text{line} \\
\text{Letter-L} &= \text{line} + \text{line} \\
\text{Letter-M} &= \text{L-junction} + \text{L-junction} \\
\text{Letter-N} &= \text{L-junction} + \text{line} \\
\text{Letter-O} &= \text{Letter-C} + \text{line} \\
\text{Letter-P} &= \text{Letter-C} + \text{line} \\
\text{Letter-Q} &= \text{Letter-O} + \text{line} \\
\text{Letter-R} &= \text{Letter-P} + \text{line}
\end{aligned}
$$

$$
\begin{aligned}
\text{Letter-S} &= \text{Letter-C} + \text{Letter-C} \\
\text{Letter-T} &= \text{line} + \text{line} \\
\text{Letter-U} &= \text{Letter-C} + \text{line} \\
\text{Letter-V} &= \text{line} + \text{line} \\
\text{Letter-W} &= \text{L-junction} + \text{L-junction} \\
\text{Letter-X} &= \text{line} + \text{line} \\
\text{Letter-Y} &= \text{Letter-V} + \text{line}
\end{aligned}
$$

In each of the binding rules, the absolute coordinate system $\{x, r, \theta\}$ is chosen. Then the relative coordinate system $s = \frac{r_\beta}{r_\alpha}$, $v = R_{-\theta_\alpha}\frac{x_\beta - x_\alpha}{r_\alpha}$, $a = \theta_\beta - \theta_\alpha$ is introduced. Furthermore, The binding support is designed according to the characteristics of this particular binding. Once these are done, formula (2.14) and (2.16) give us the bits gained for the binding. The total saved bits can be calculated through the cost function discussed in Section 3.1.

Among these binding rules, two are *independent* of the quantity s, the ratio of two sizes. They are the L-junction and T-junction. The reason is pretty straightforward. If we define the absolute coordinates of the line, they could be the end point as x, the length as r and the angle as θ. By introducing the relative coordinates $s = \frac{r_\beta}{r_\alpha}$, $v = R_{-\theta_\alpha}\frac{x_\beta - x_\alpha}{r_\alpha}$, $a = \theta_\beta - \theta_\alpha$ between two lines, it is obvious that the ratio of sizes should not play a role in the binding function because the idea of junction is just two lines connecting in some relative position and at a relative angle. So in the L-junction and T-junction, we use the formula (2.16) to compute the bits gained for the composition.

Furthermore, the parameter C for the mature objects is 10,000. We just choose C big enough that the mature object would not be chosen repeatedly. Eventually, the algorithm can come back to these mature objects. In reality, since the number of all objects increases so fast, a return to these objects has never been observed. The foreground probability p is 0.9. The background probability q is 1.0×10^{-8}. The value of q is pretty small. However, this value is chosen because we want to make sure the isolated point can be interpreted as a disk. If the isolated point can not be interpreted as a disk, it is quite impossible to

recognize a perfect and finest line. As for the greedy algorithm, it only starts once. The Q_l in the likelihood ratio can be found in the appendix.

With all the equipment we have discussed from the chapter 2 to chapter 4, we implement the recognition machine with algorithm (3.1) - (3.3). The lines, junctions, and alphabets are put to test. The results are in the next section.

4.2 Results

We use ten images of each letter to test the code implemented by the algorithms we discussed. The following pages include some samples from these images. Some of these graphs are perfect and some of them are a little bit wild. We also put two or three letters in one image and try to recognize them. Some of these correctly-recognized cases are presented from page 40 to page 52. One of the properties of multiple-letter cases (correctly read) is that the letters are not close to each other. However, some cases are not so good if the letters are close. These failed examples are in the next section.

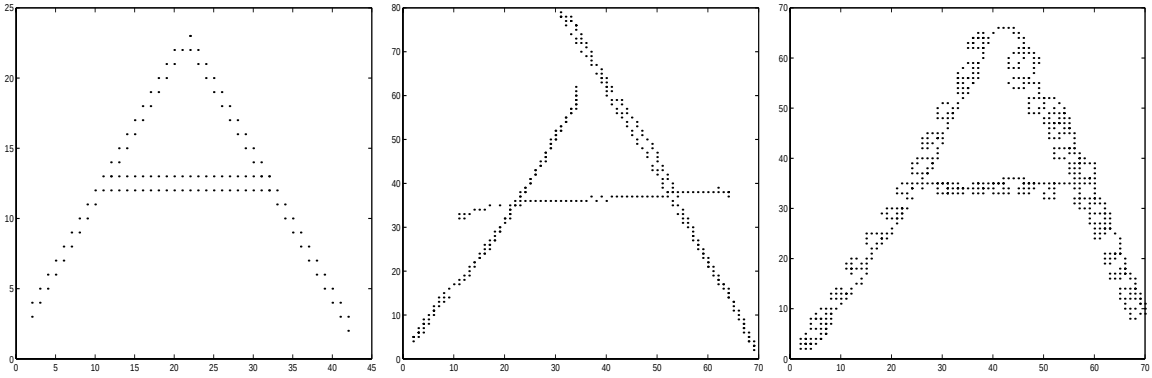

Figure 4.1: Letter A

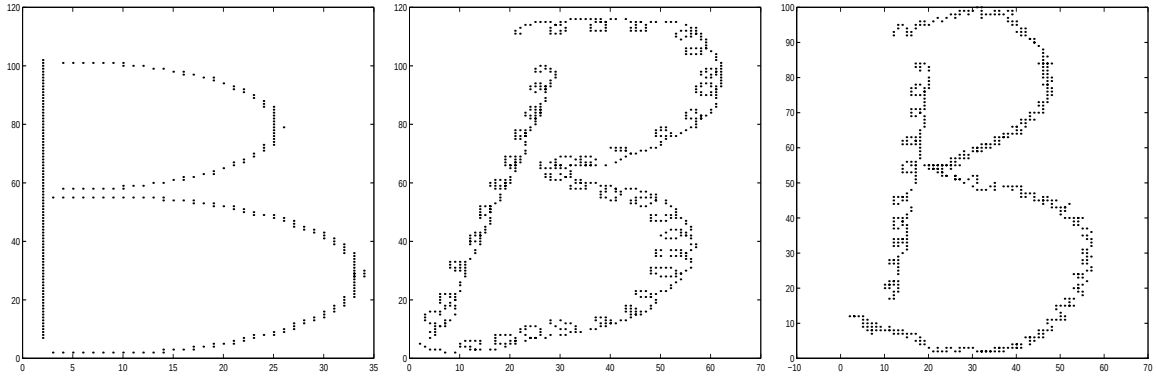

Figure 4.2: Letter B

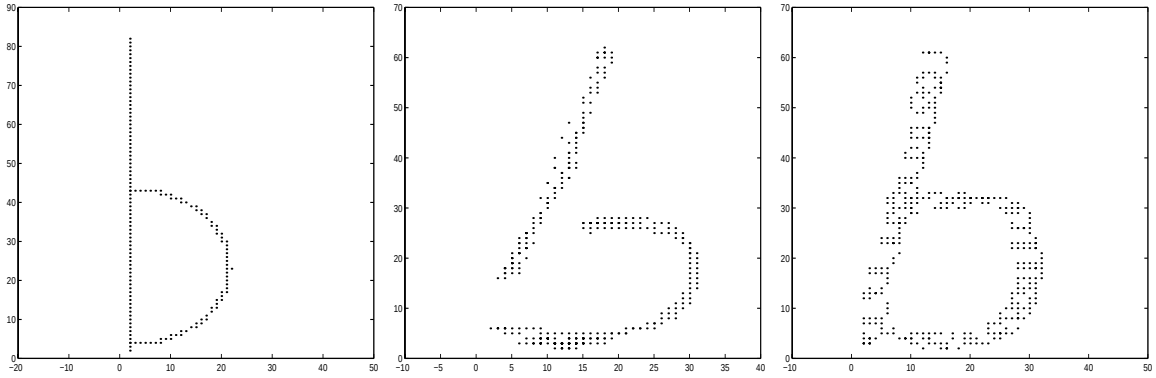

Figure 4.3: Letter B

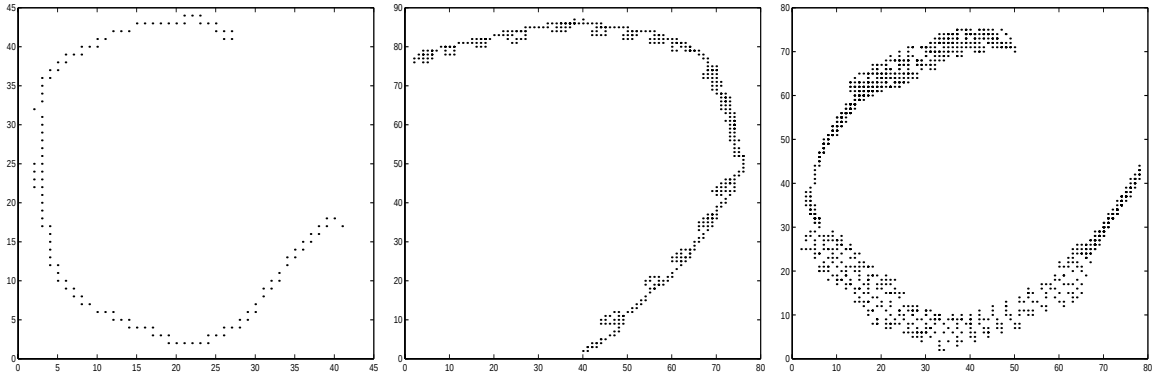

Figure 4.4: Letter C

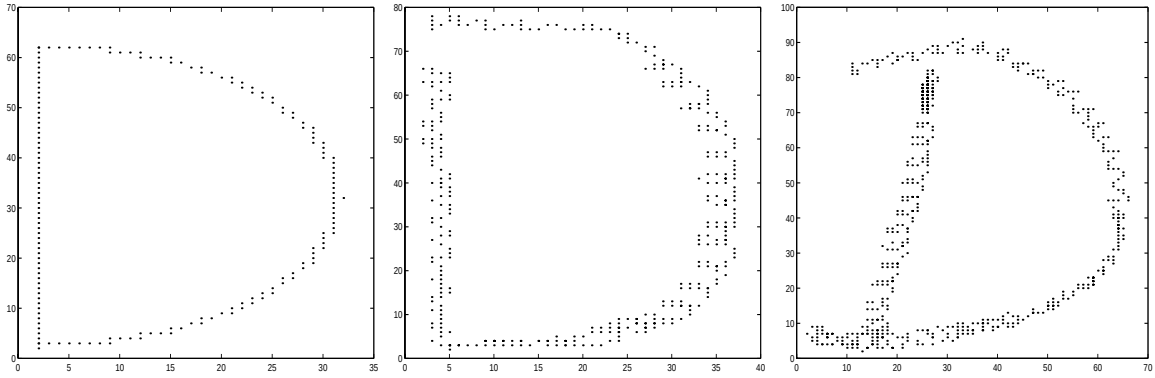

Figure 4.5: Letter D

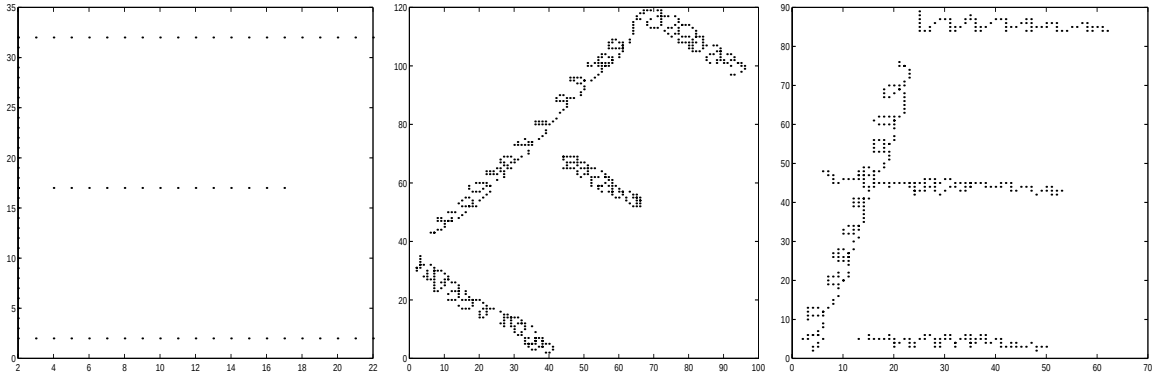

Figure 4.6: Letter E

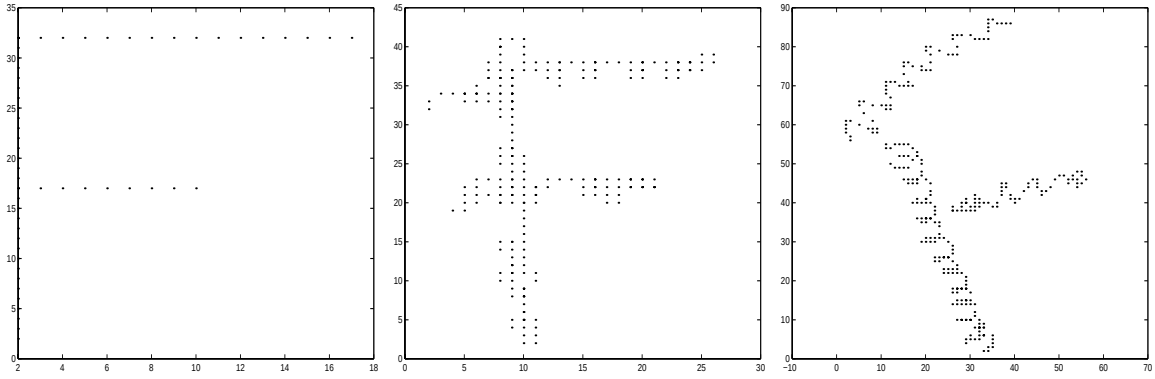

Figure 4.7: Letter F

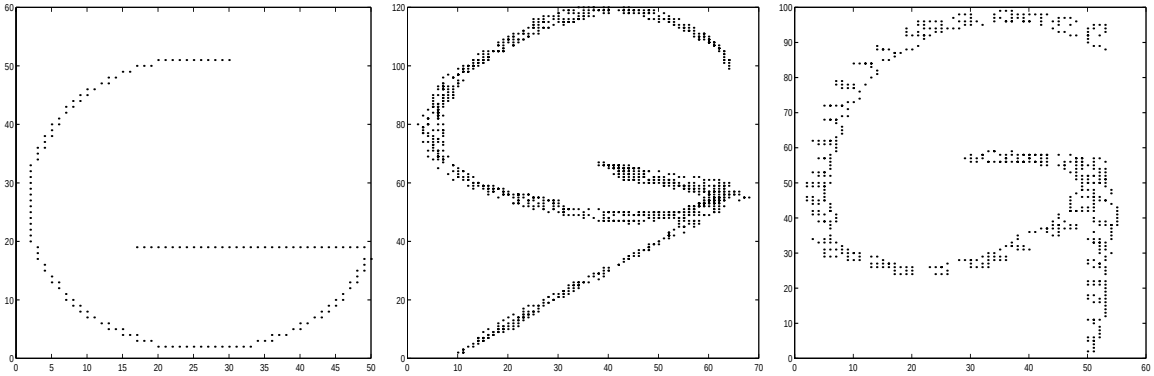

Figure 4.8: Letter G

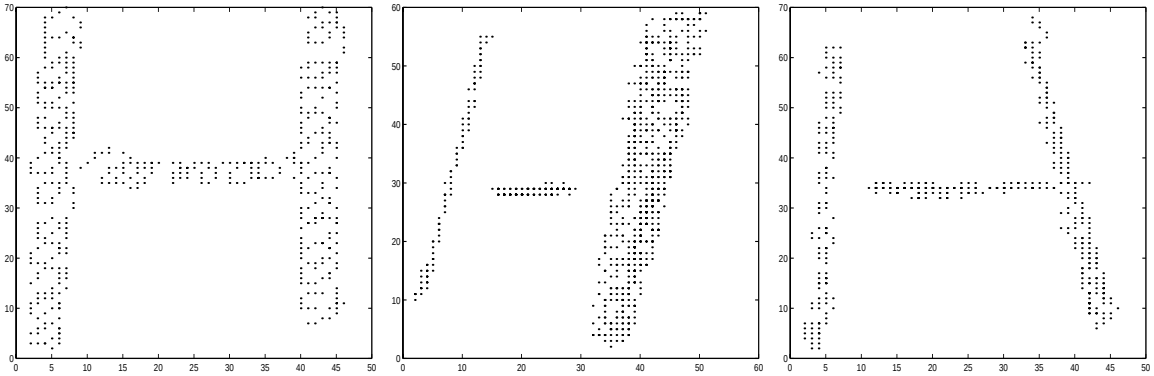

Figure 4.9: Letter H

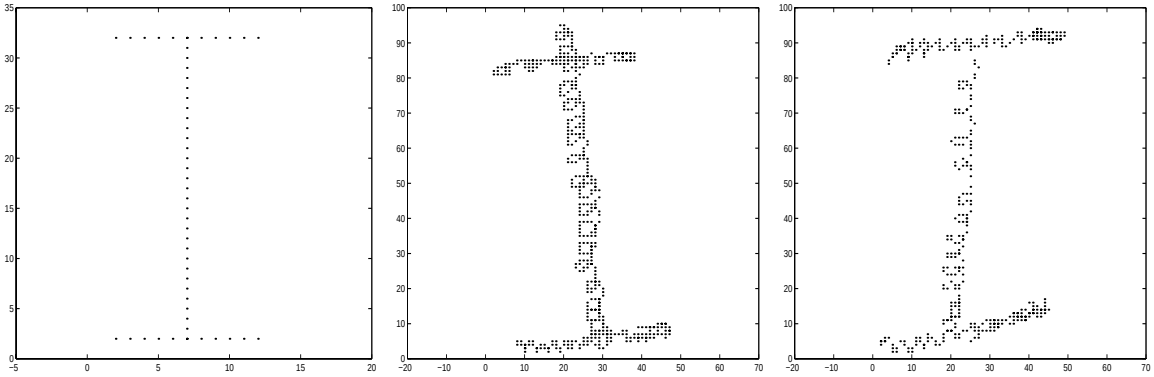

Figure 4.10: Letter I

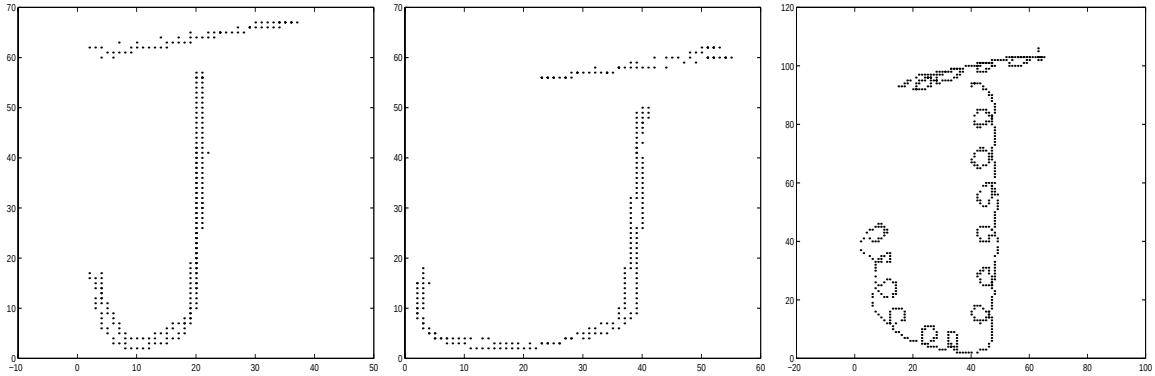

Figure 4.11: Letter J

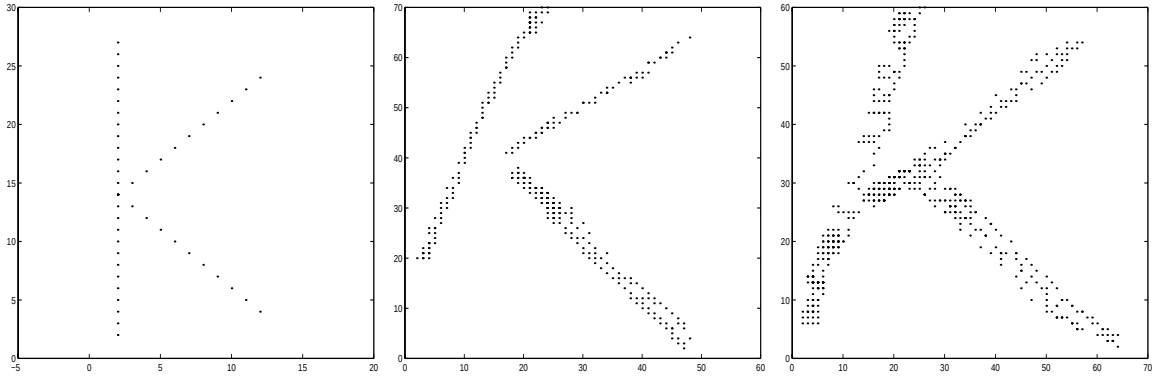

Figure 4.12: Letter K

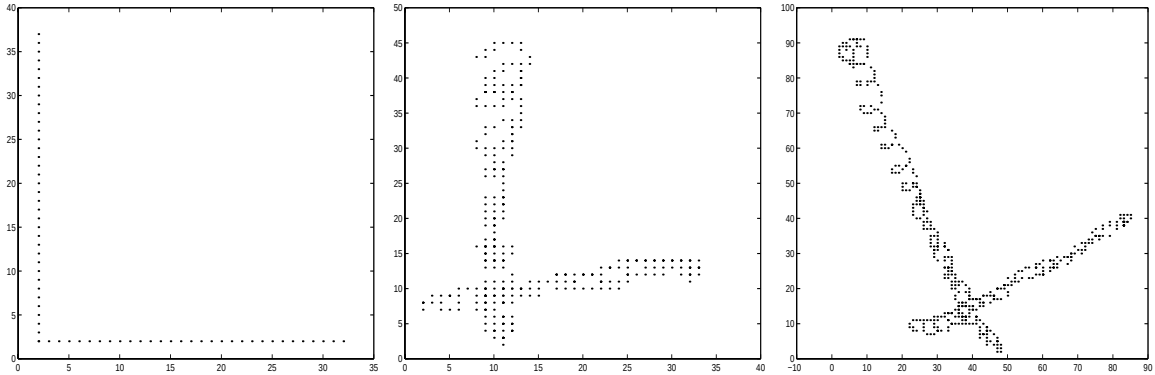

Figure 4.13: Letter L

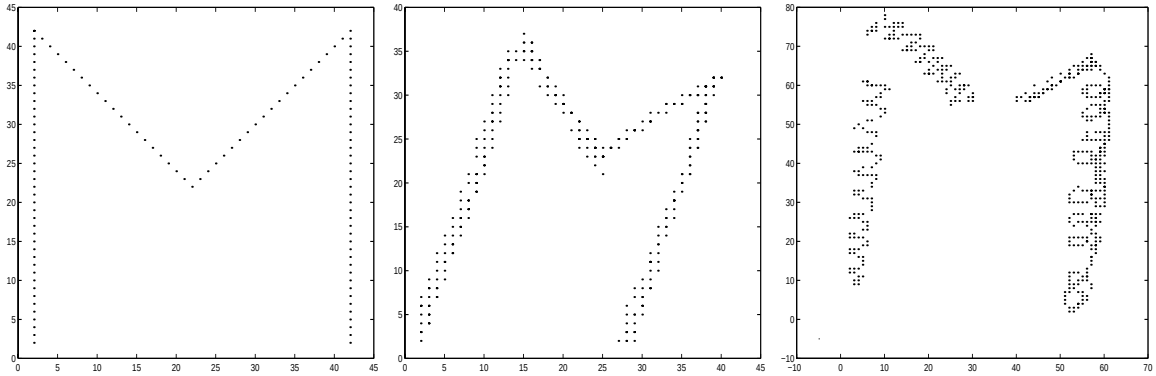

Figure 4.14: Letter M

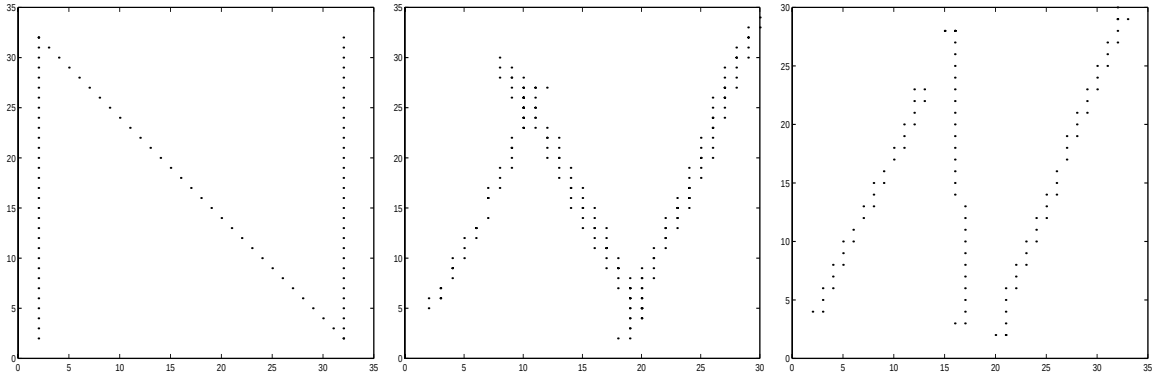

Figure 4.15: Letter N

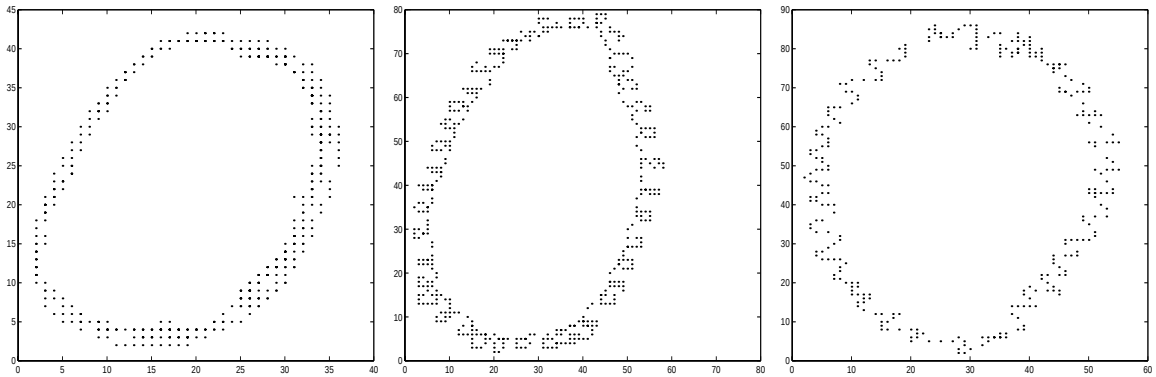

Figure 4.16: Letter O

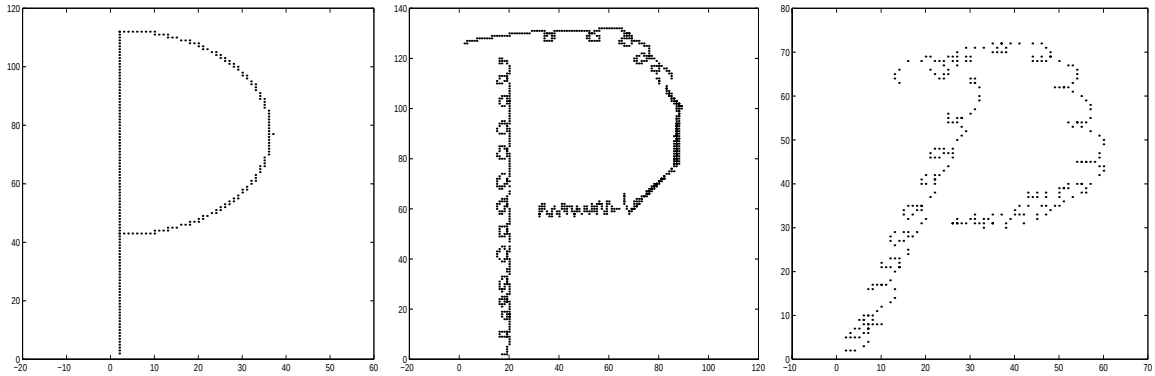

Figure 4.17: Letter P

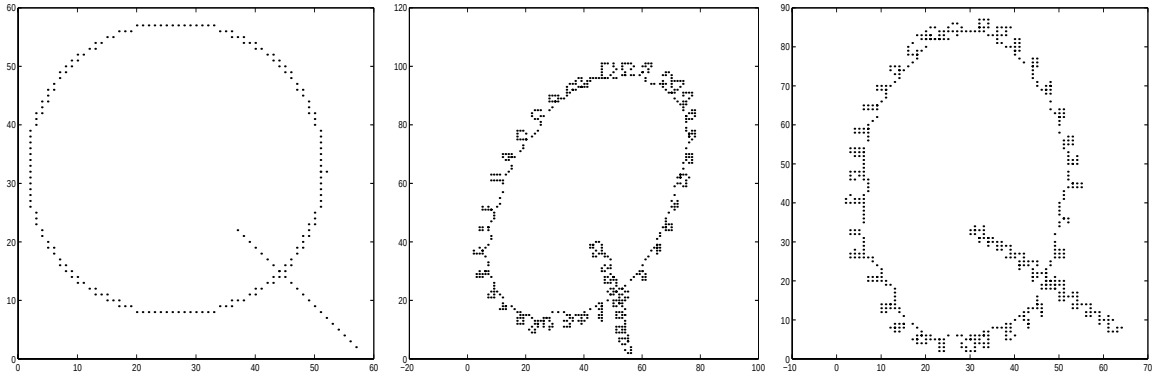

Figure 4.18: Letter Q

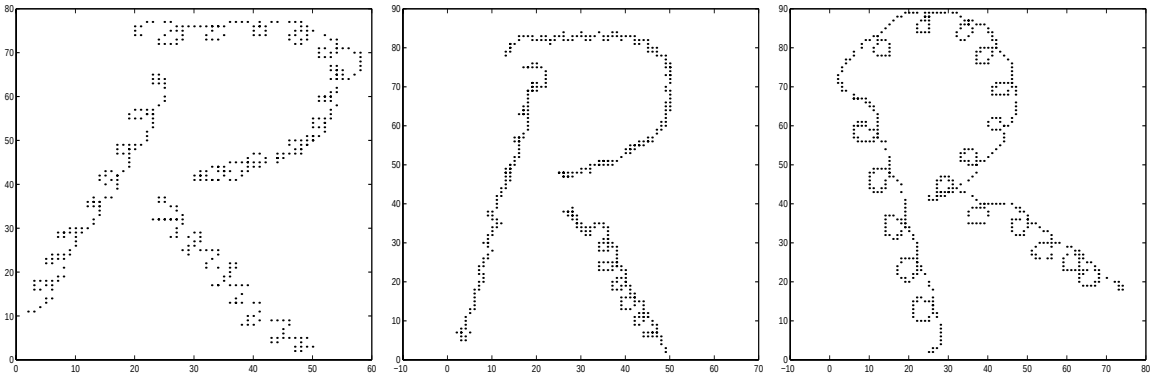

Figure 4.19: Letter R

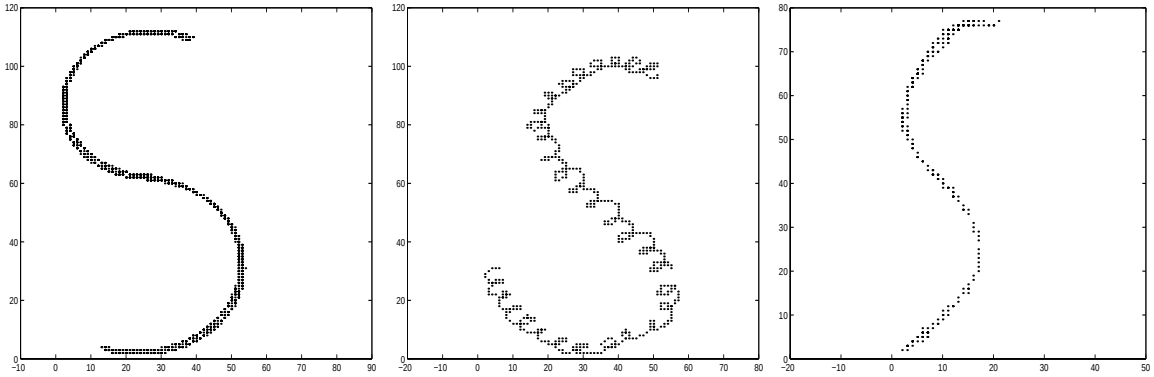

Figure 4.20: Letter S

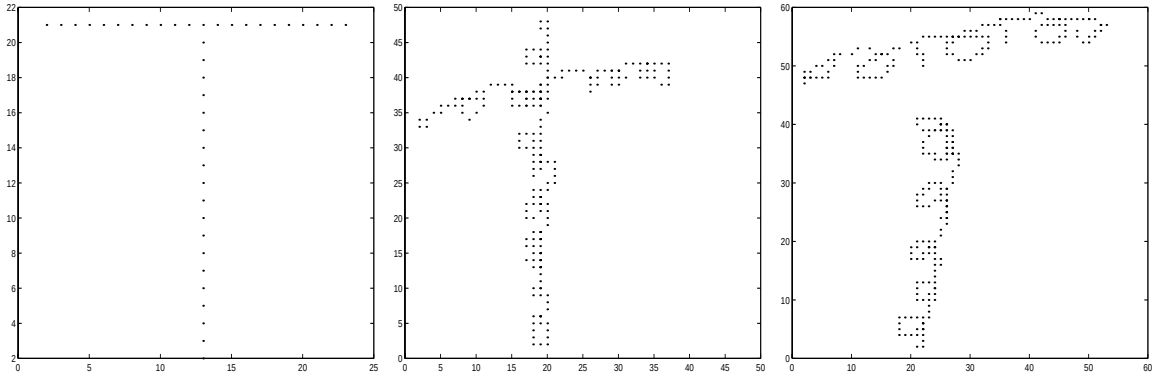

Figure 4.21: Letter T

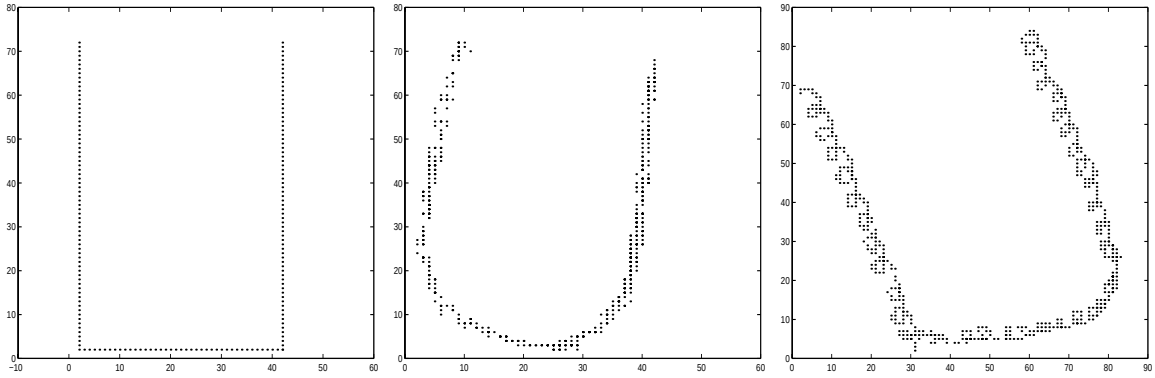

Figure 4.22: Letter U

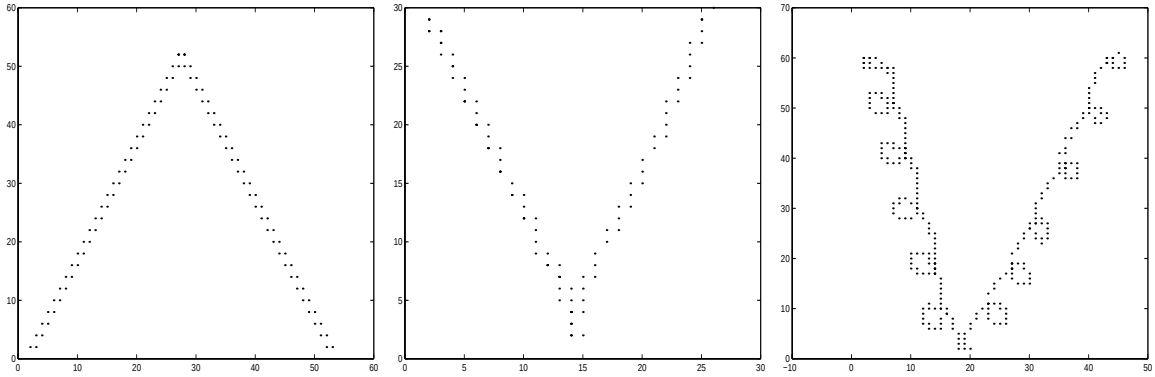

Figure 4.23: Letter V

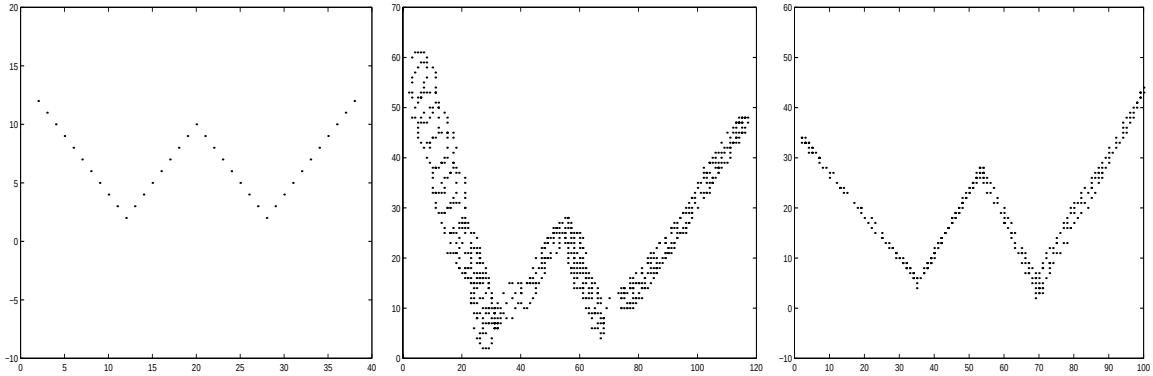

Figure 4.24: Letter W

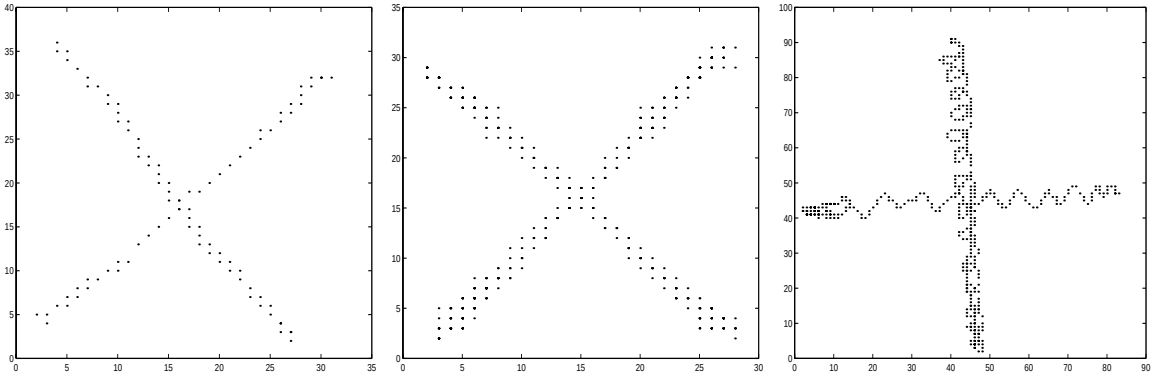

Figure 4.25: Letter X

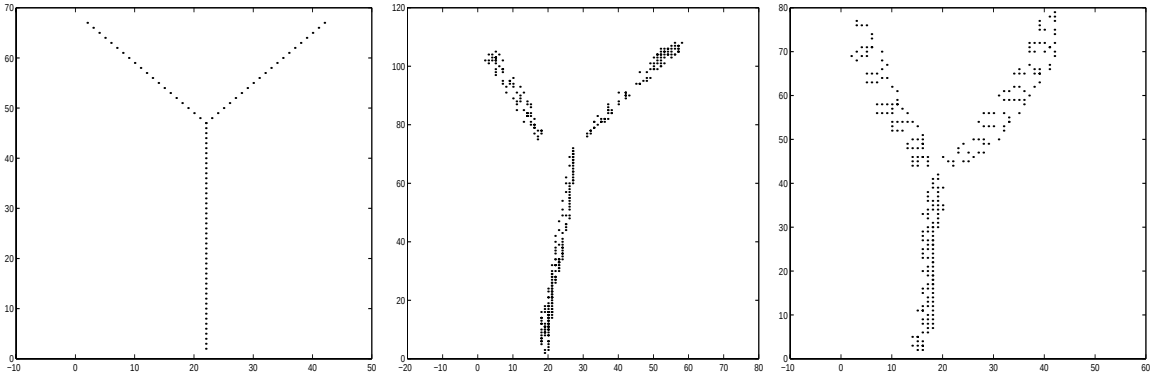

Figure 4.26: Letter Y

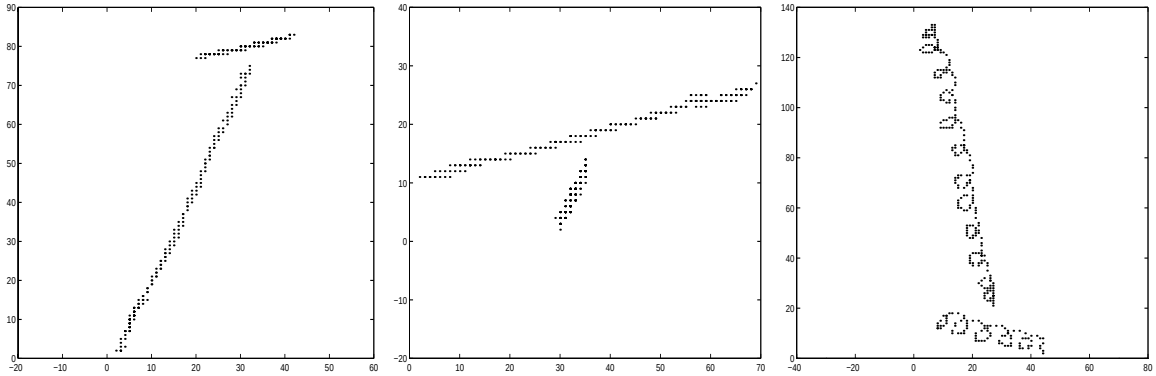

Figure 4.27: T-Junction

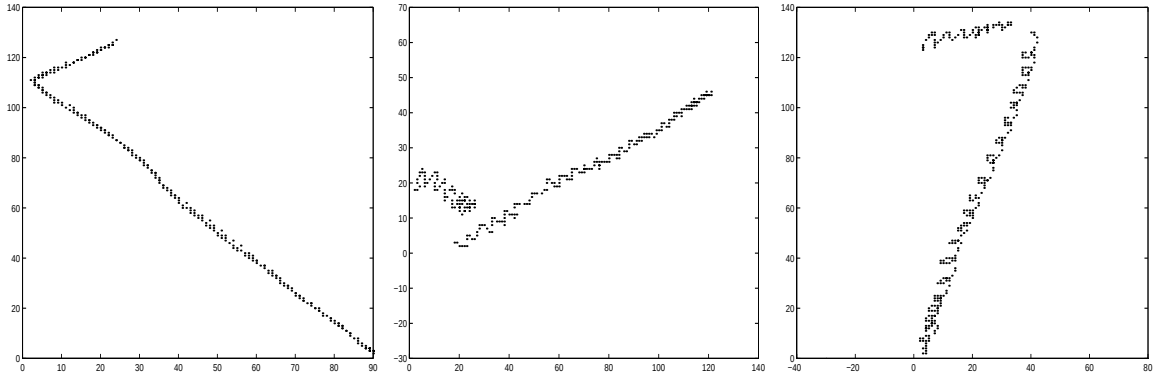

Figure 4.28: L-Junction

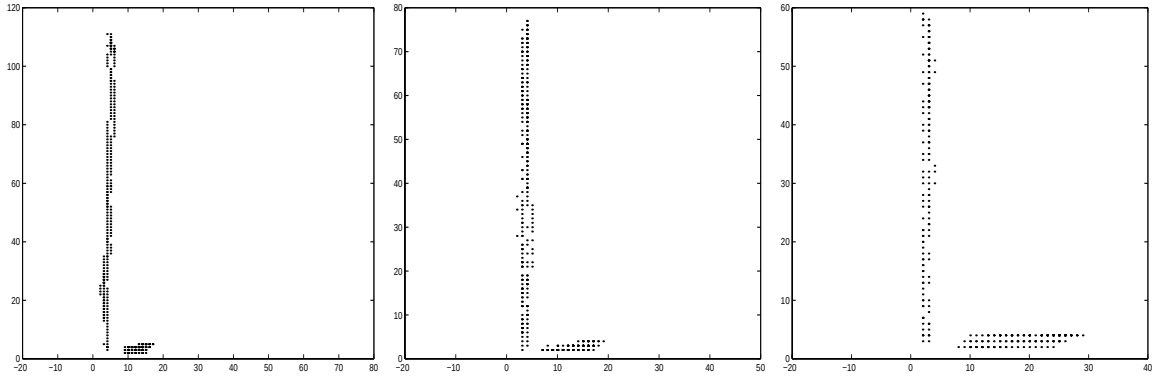

Figure 4.29: L-Junction

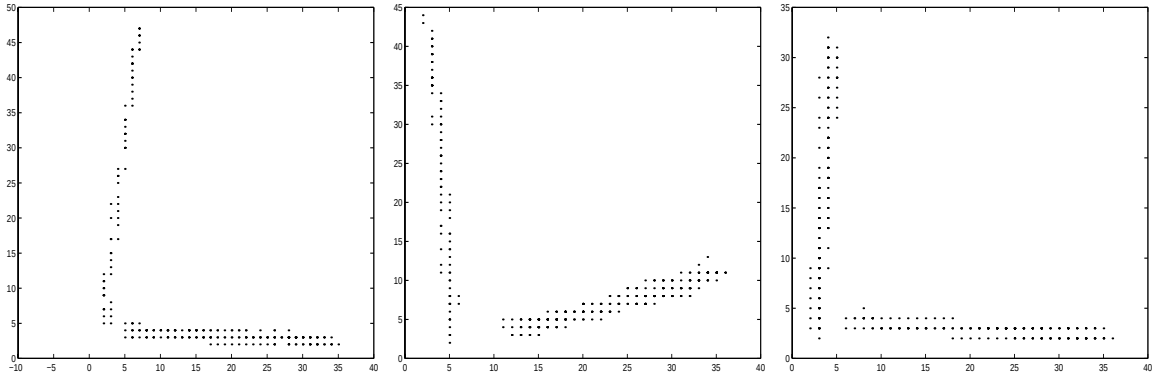

Figure 4.30: Letter L

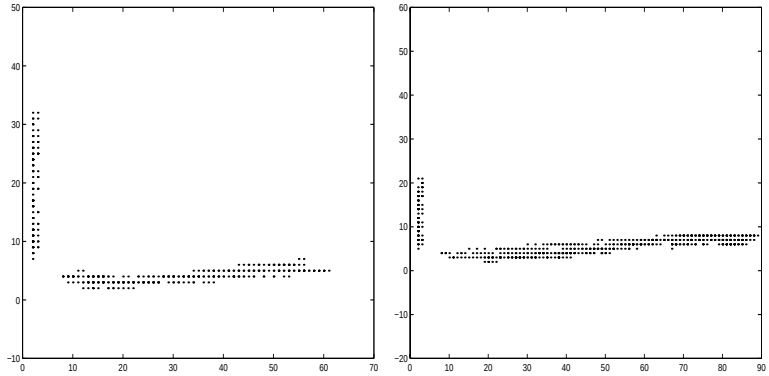

Figure 4.31: L-Junction

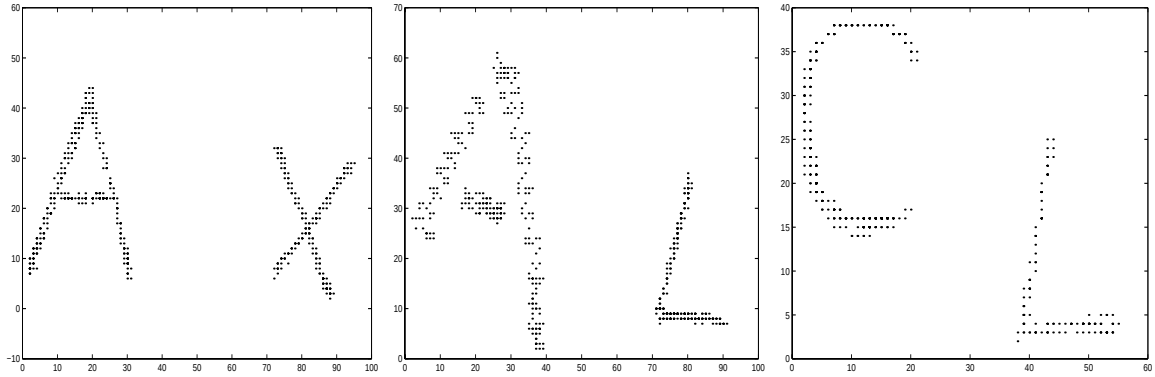

Figure 4.32: Multiple letters

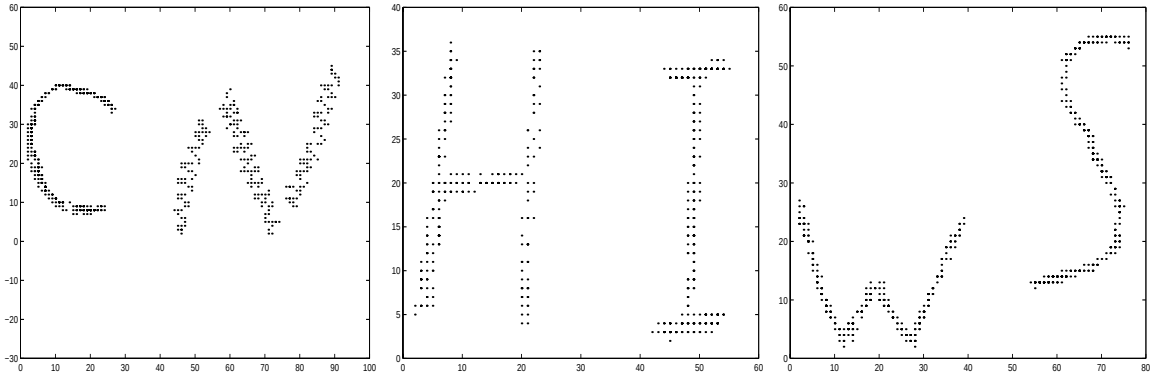

Figure 4.33: Multiple letters

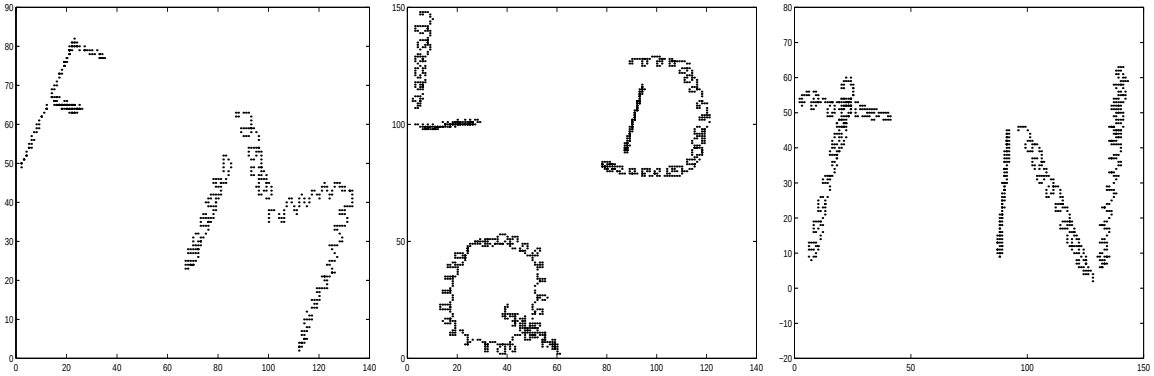

Figure 4.34: Multiple letters

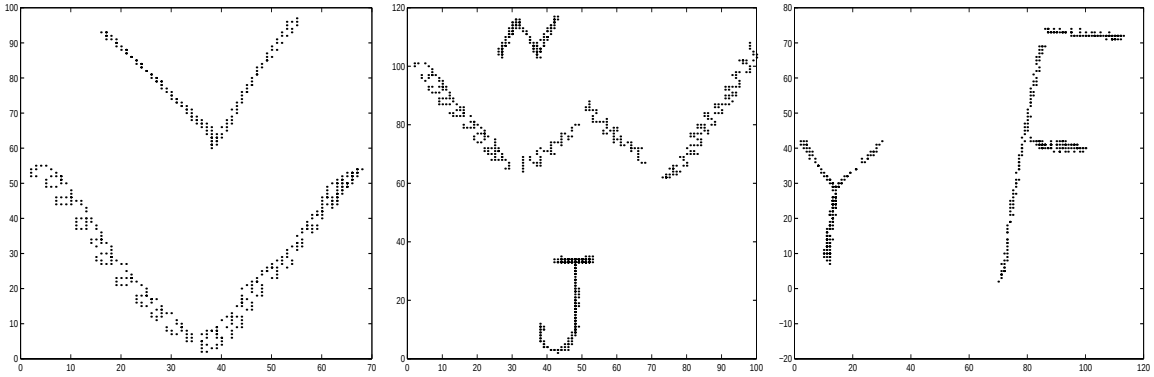

Figure 4.35: Multiple letters

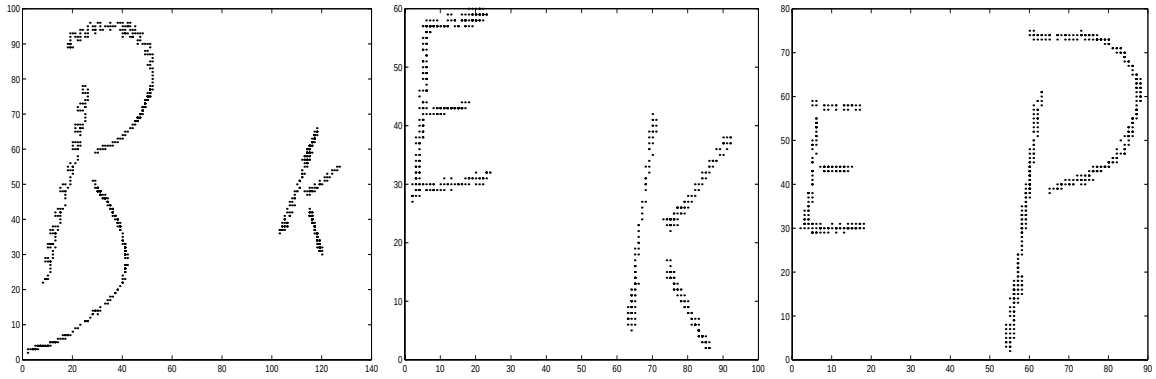

Figure 4.36: Multiple letters

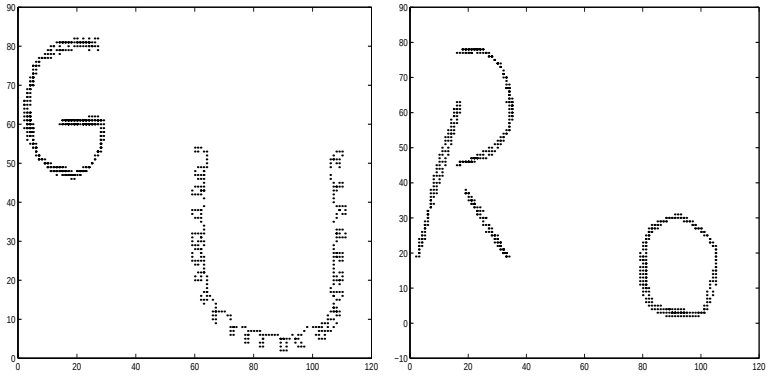

Figure 4.37: Multiple letters

The following figures demonstrate how these images are recognized. The first column is the original image without any interpretation. The next column shows how this image is interpreted. So does the third column. The radii of the disks are getting smaller when we move from left to right. It means we have a interpretation with a finer resolution. It also means as time goes on, we can get finer interpretation. Like the first case, we have an image in the first column. We let the computer run a couple of seconds or ten seconds. We can get the interpretation in the second column by pausing the computer and getting the optimal solution at this particular moment. So the machine tells us it is an A. We can let the machine continue to run for another couple of seconds or ten seconds and get the interpretation, which is also an A but with better fit. This newer interpretation will be in finer resolution because it has a better cost function value than the previous one.

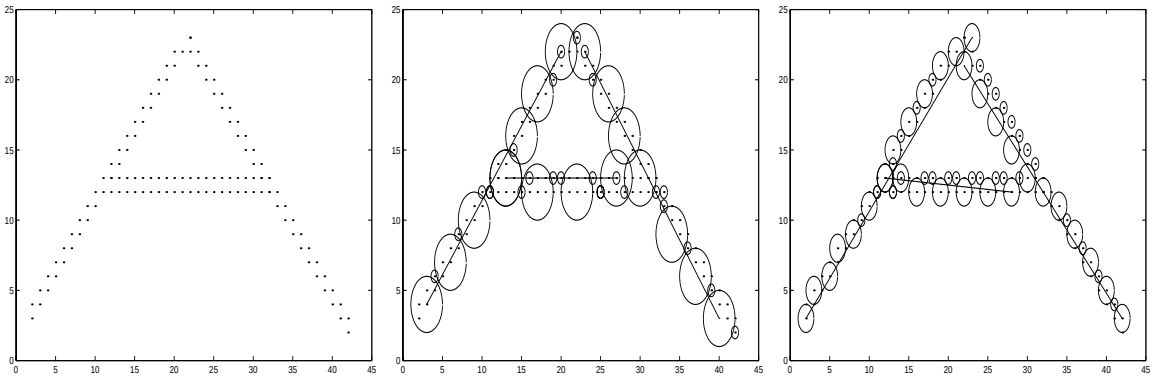

Figure 4.38: Letter A

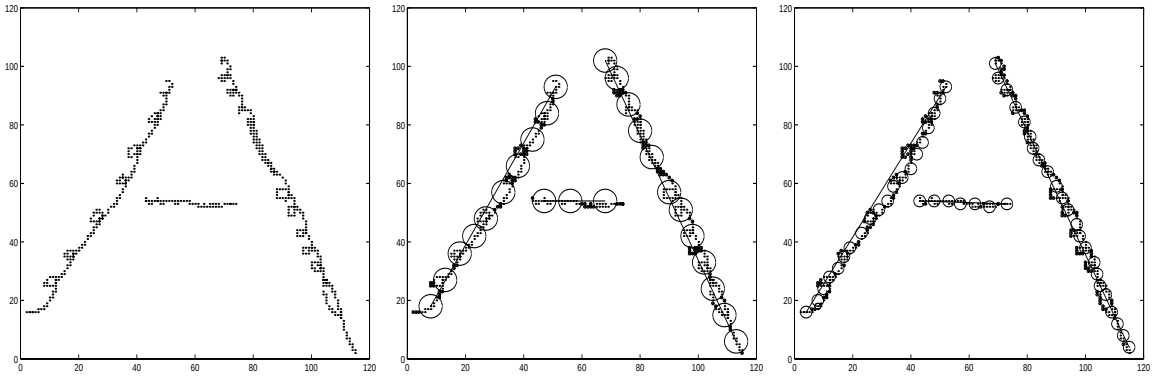

Figure 4.39: Letter A

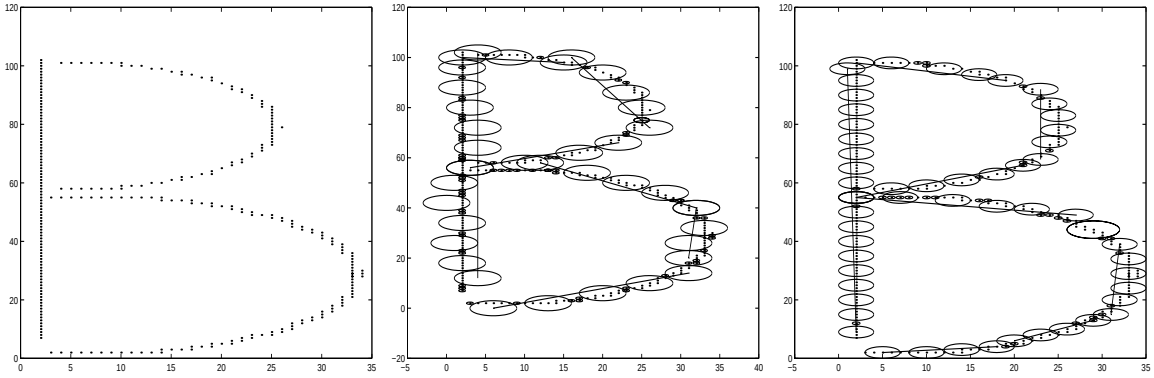

Figure 4.40: Letter B

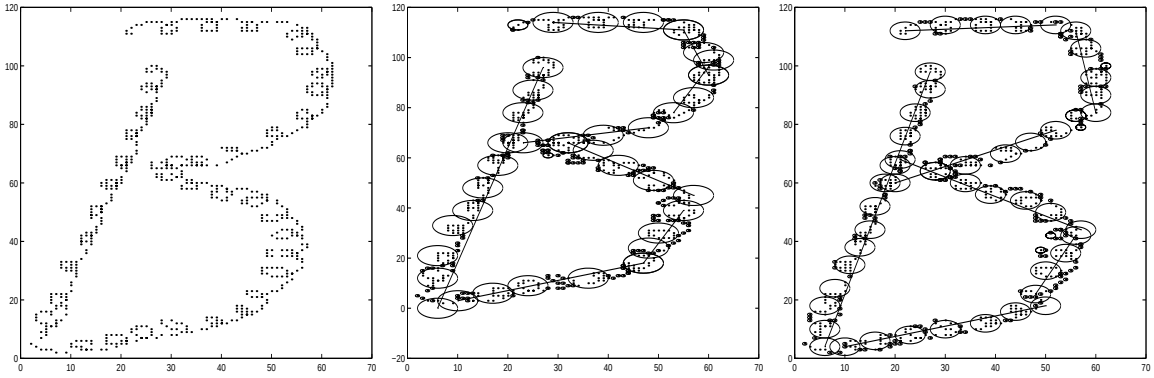

Figure 4.41: Letter B

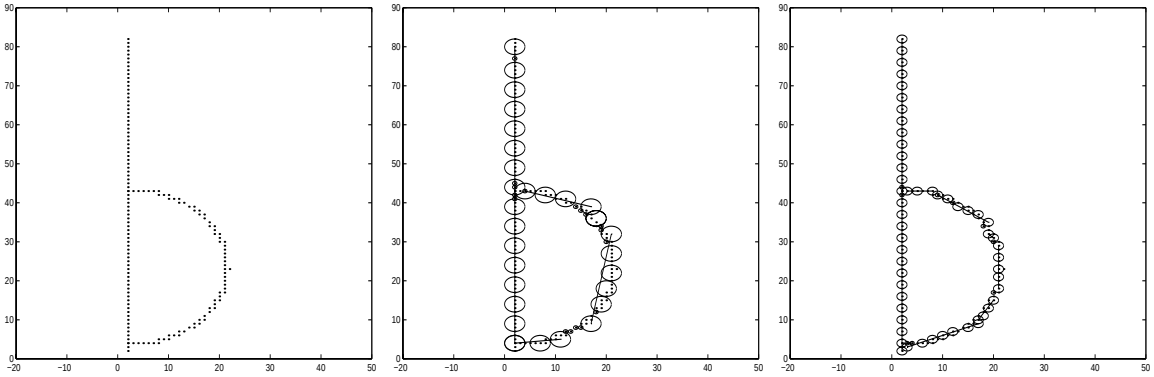

Figure 4.42: Letter b

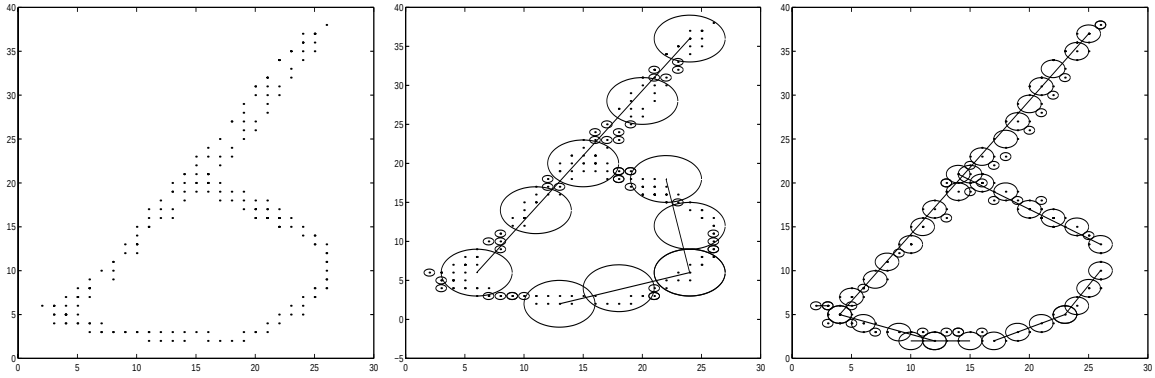

Figure 4.43: Letter b

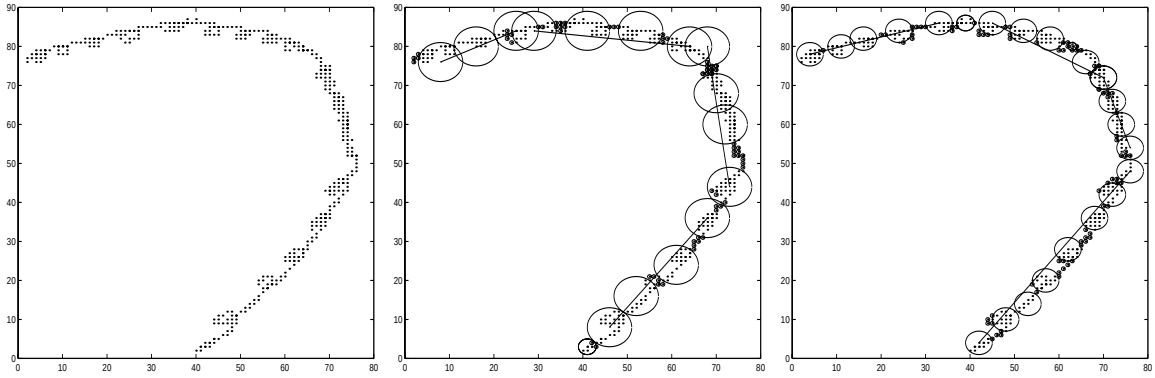

Figure 4.44: Letter C

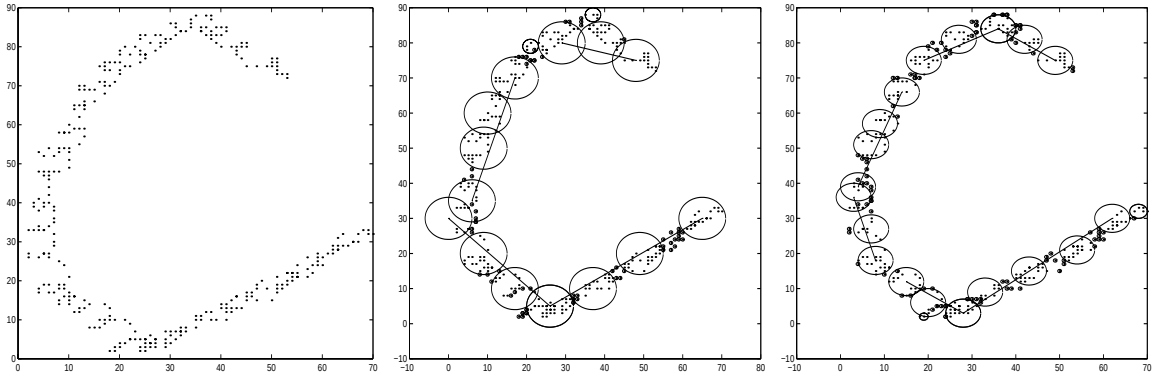

Figure 4.45: Letter C

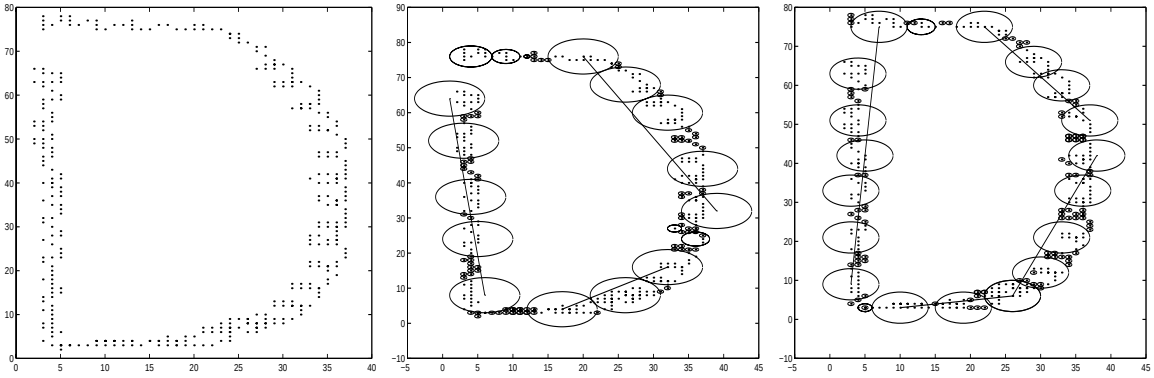

Figure 4.46: Letter D

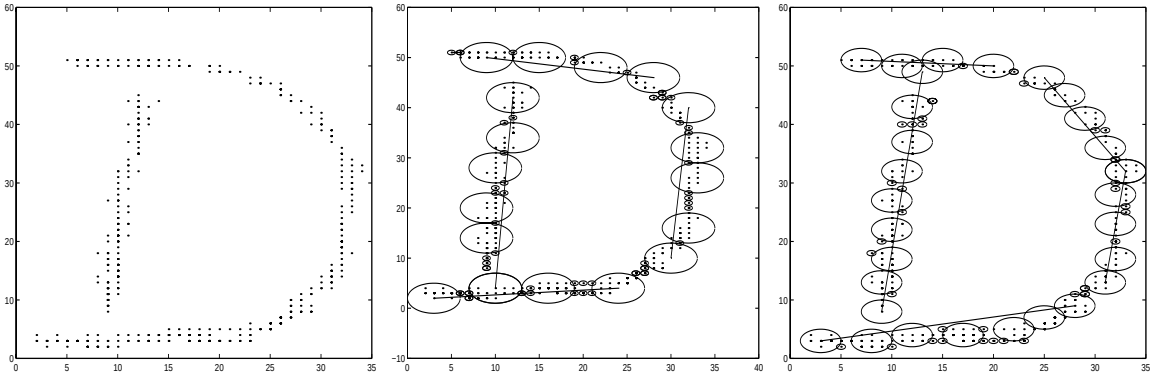

Figure 4.47: Letter D

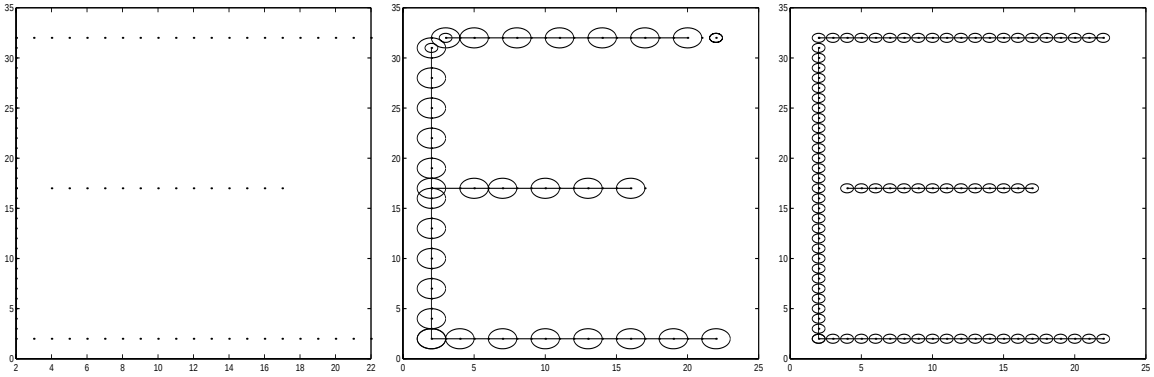

Figure 4.48: Letter E

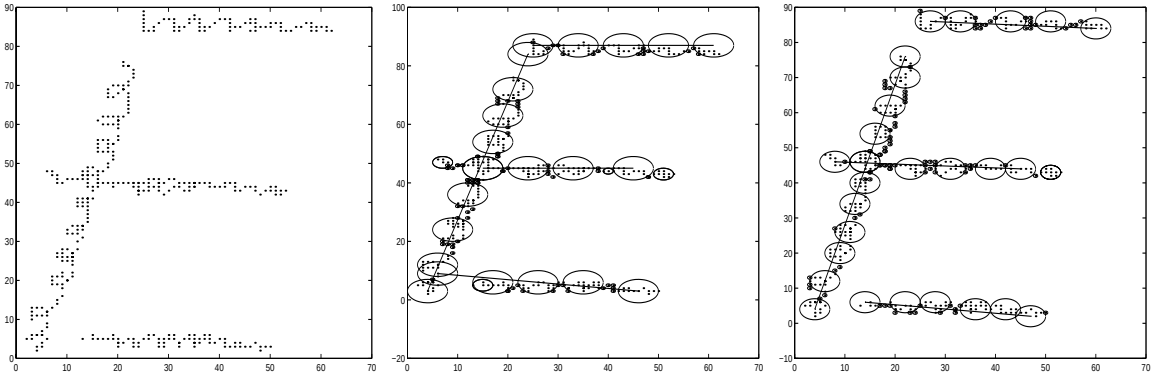

Figure 4.49: Letter E

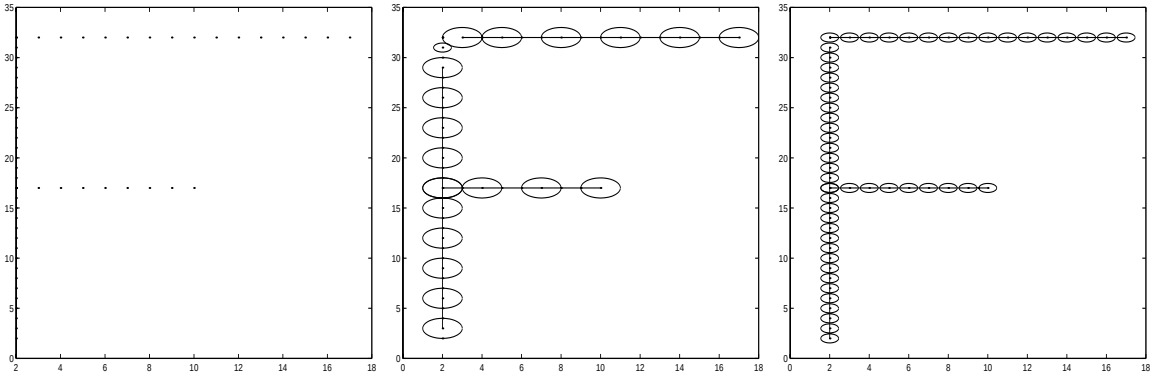

Figure 4.50: Letter F

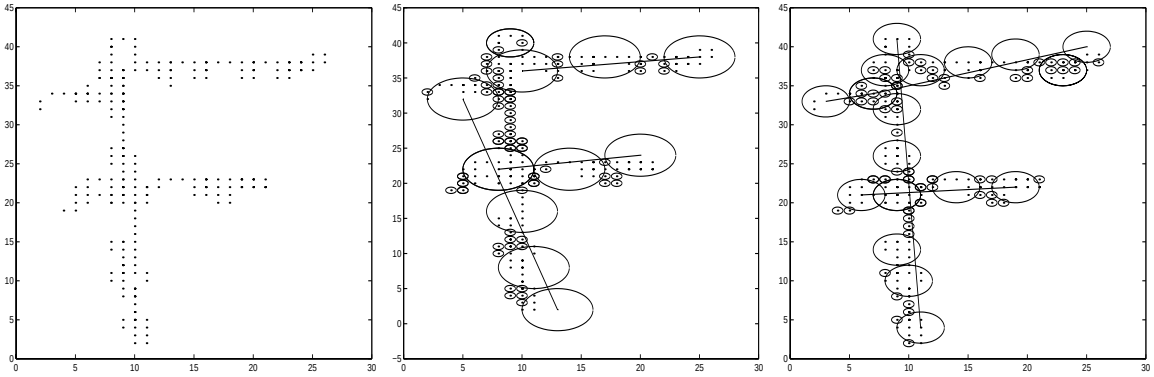

Figure 4.51: Letter F

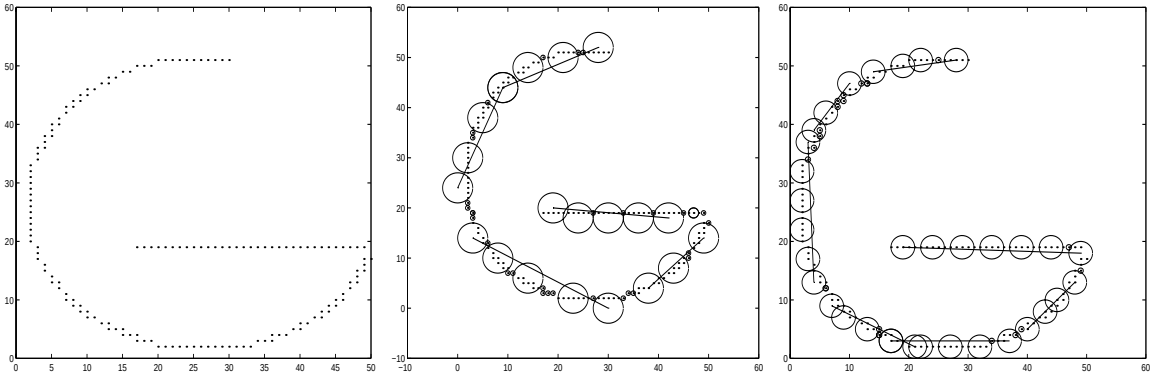

Figure 4.52: Letter G

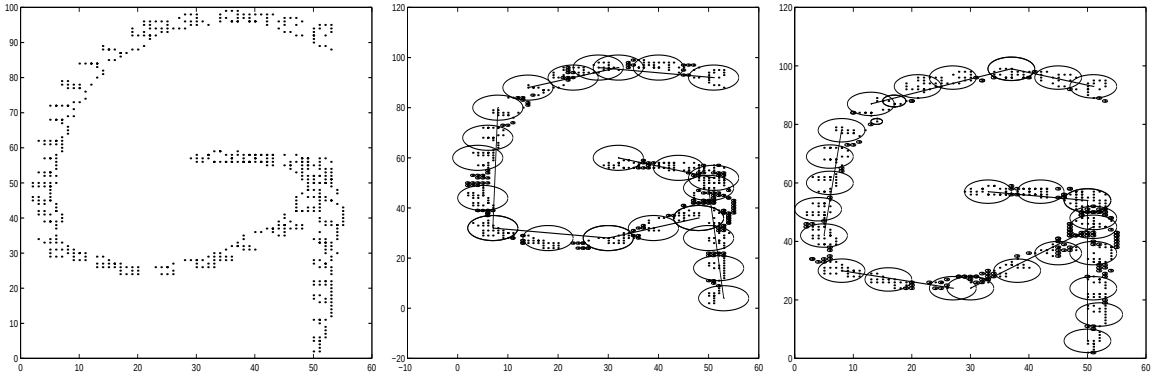

Figure 4.53: Letter G

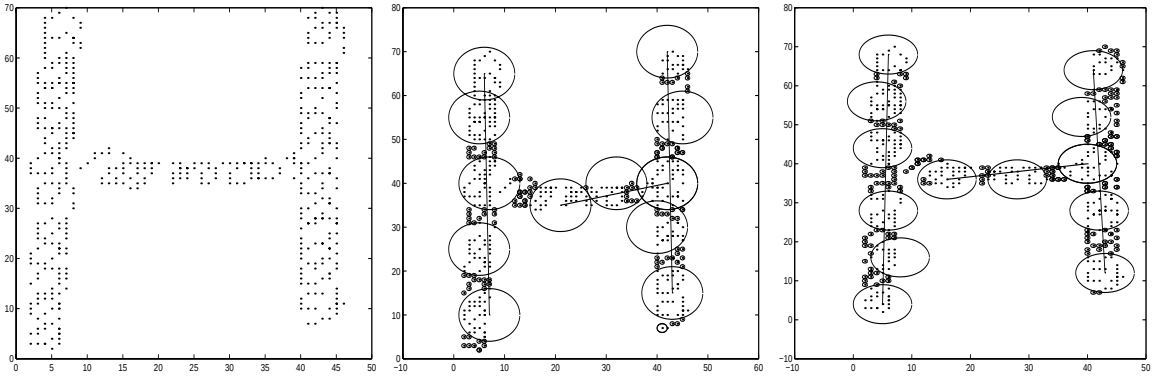

Figure 4.54: Letter H

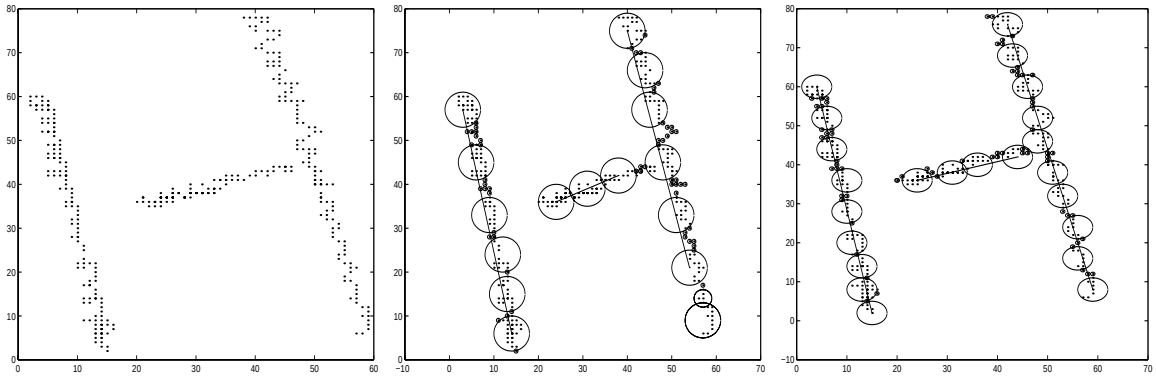

Figure 4.55: Letter H

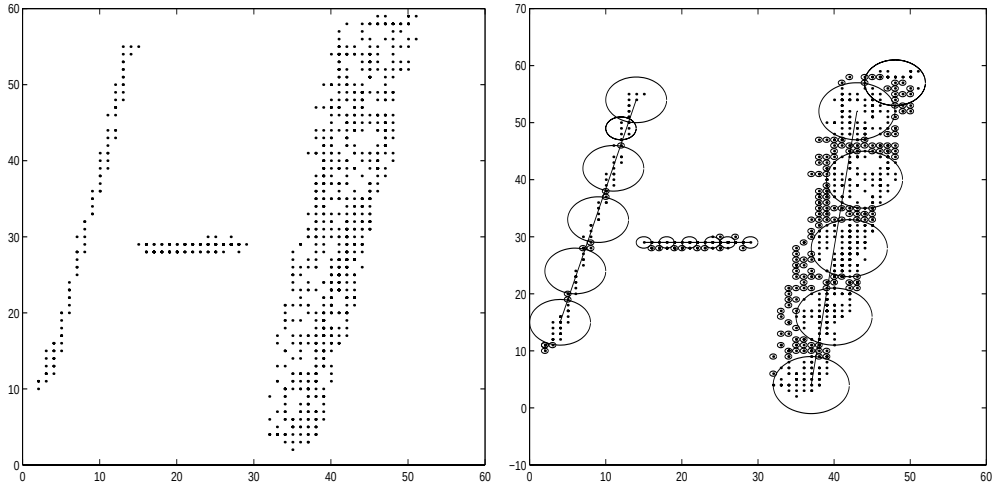

Figure 4.56: Letter H

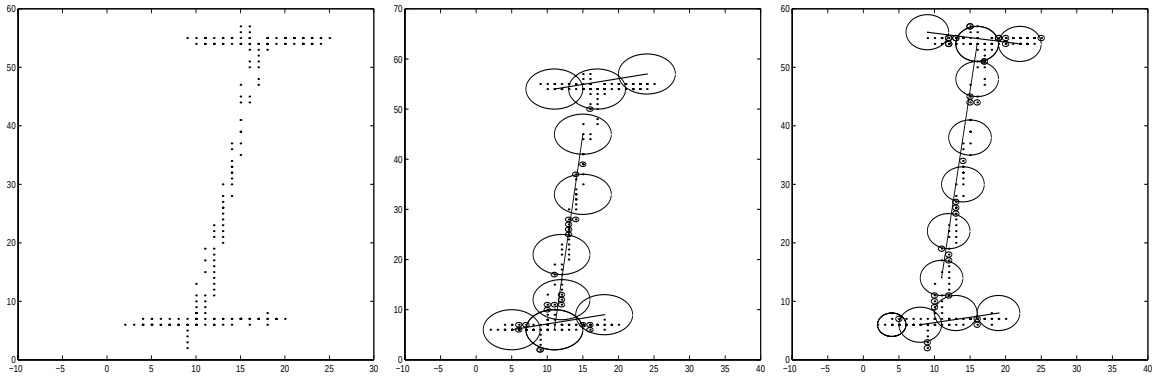

Figure 4.57: Letter I

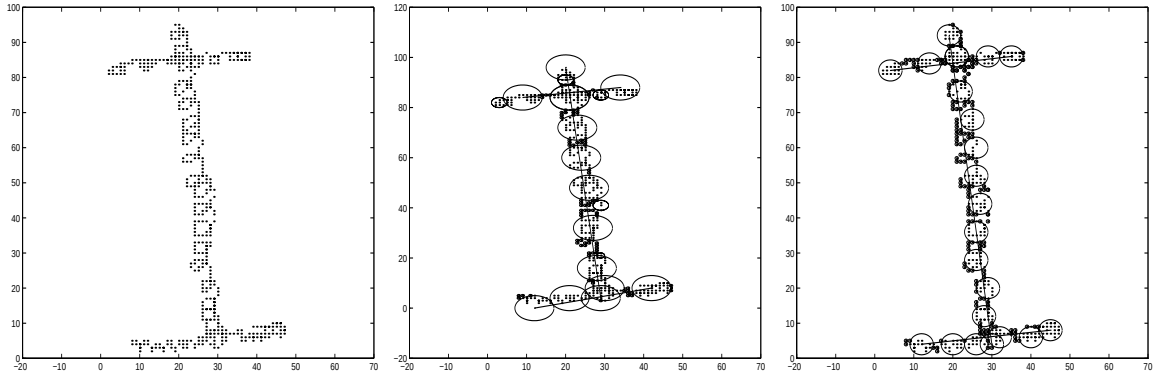

Figure 4.58: Letter I

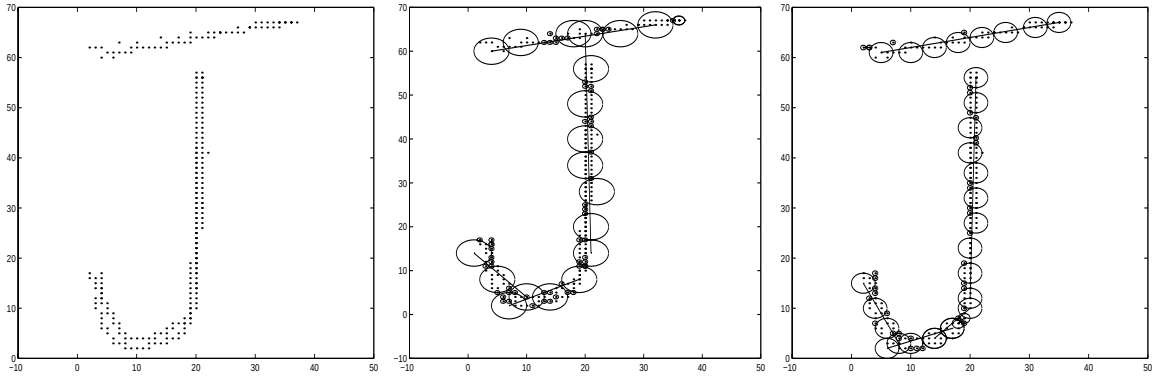

Figure 4.59: Letter J

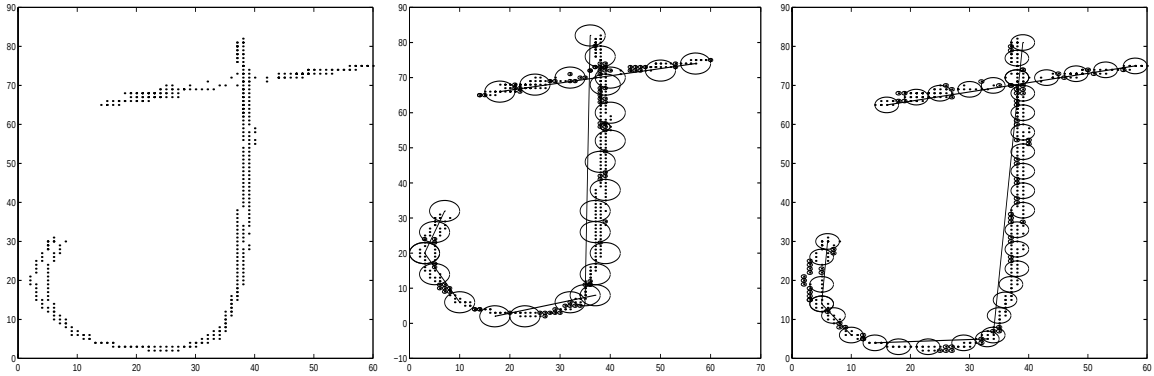

Figure 4.60: Letter J

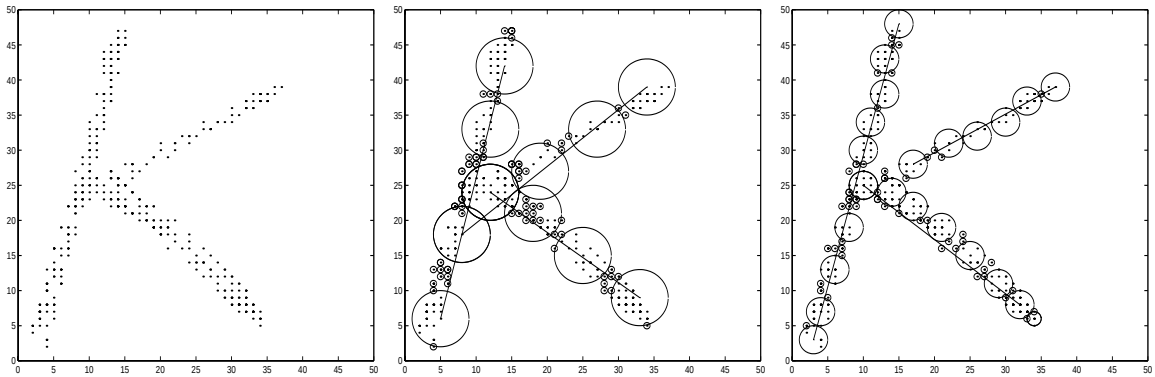

Figure 4.61: Letter K

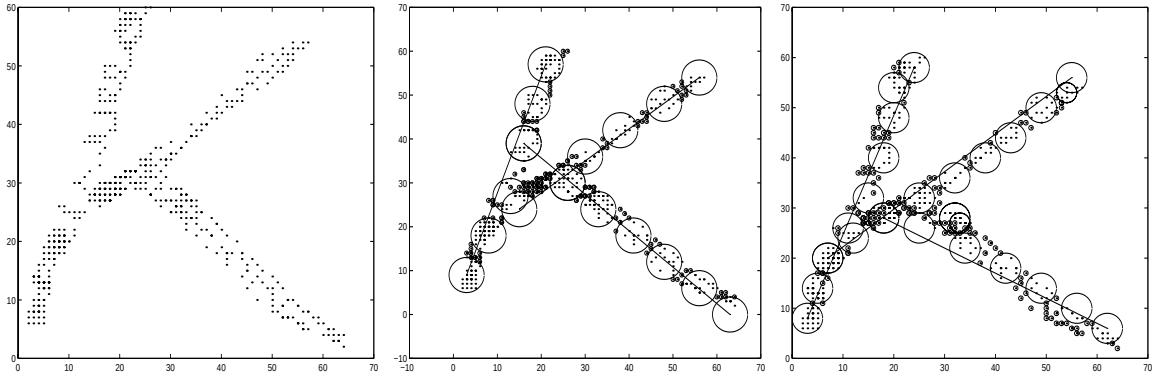

Figure 4.62: Letter K

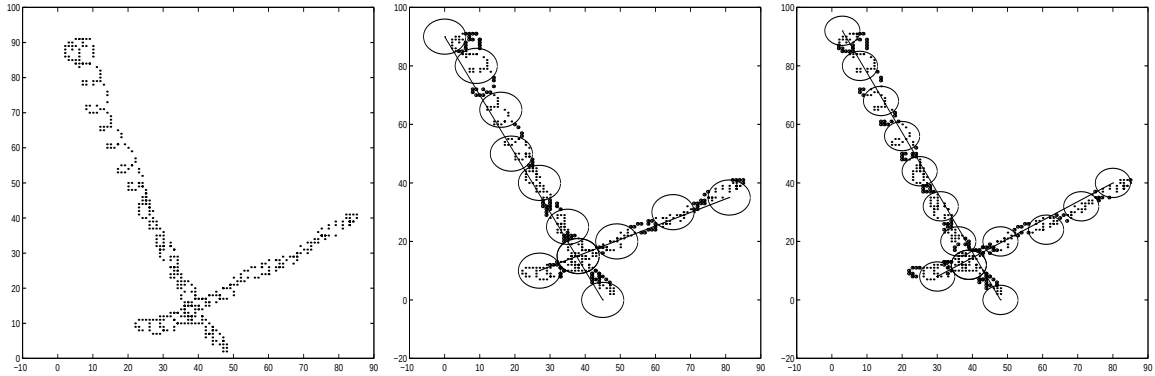

Figure 4.63: Letter L

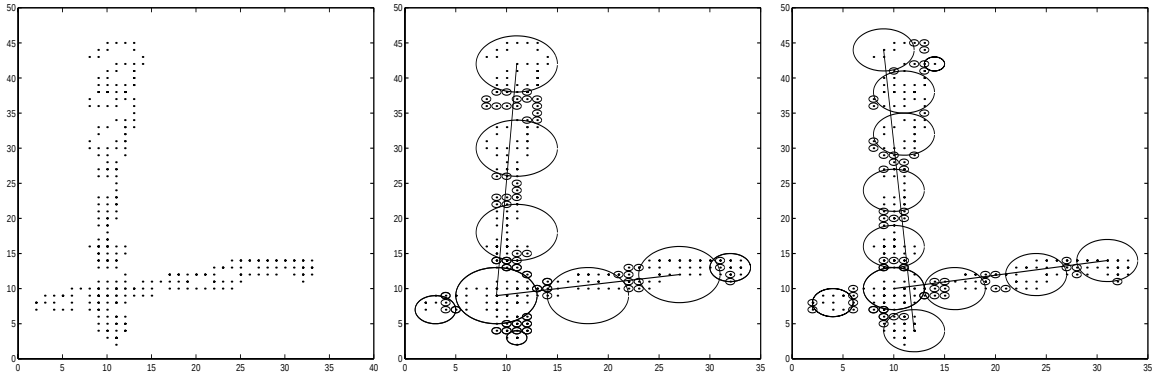

Figure 4.64: Letter L

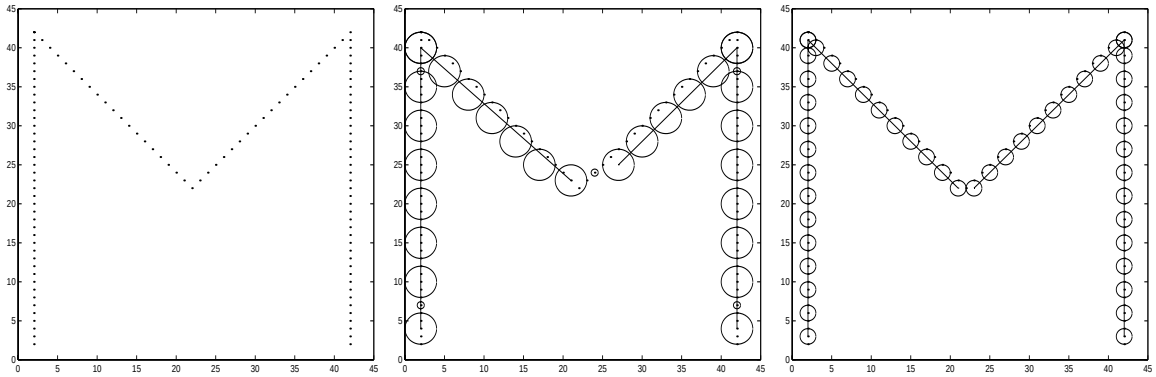

Figure 4.65: Letter M

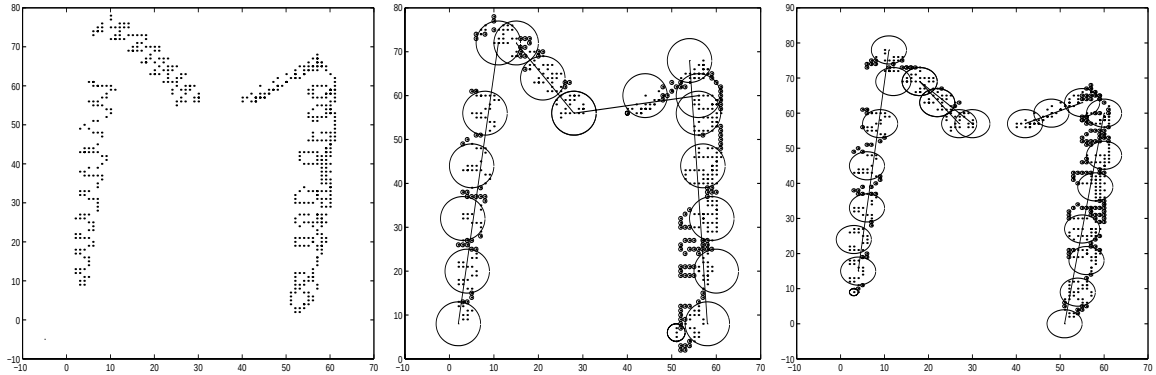

Figure 4.66: Letter M

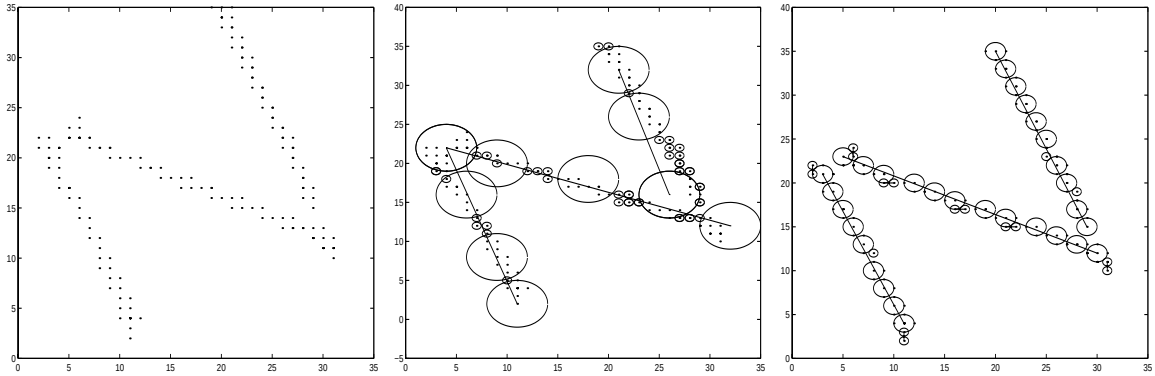

Figure 4.67: Letter N

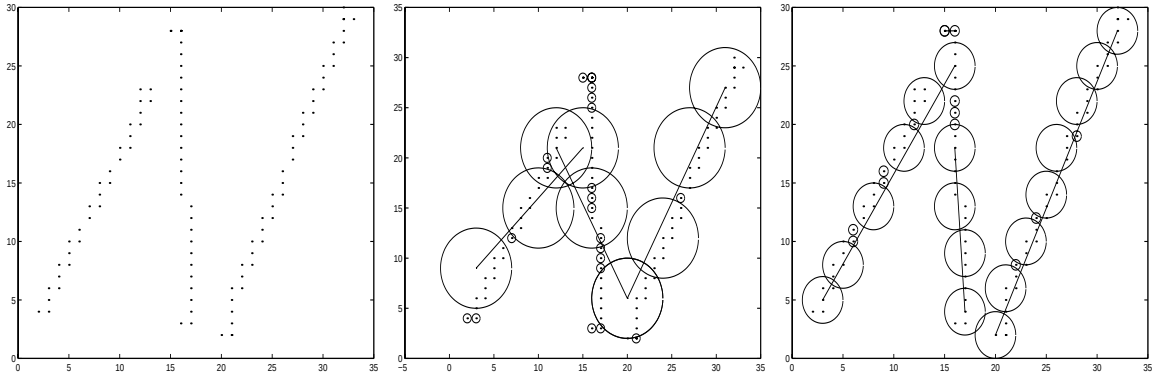

Figure 4.68: Letter N

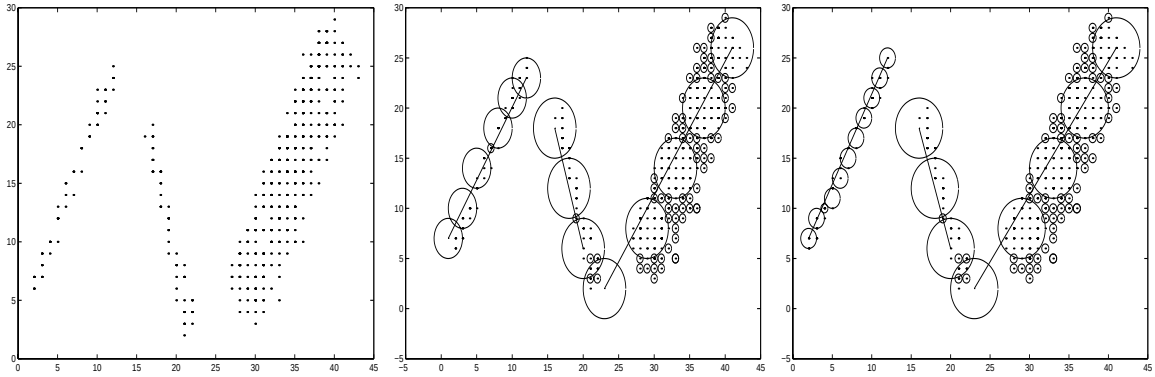

Figure 4.69: Letter N

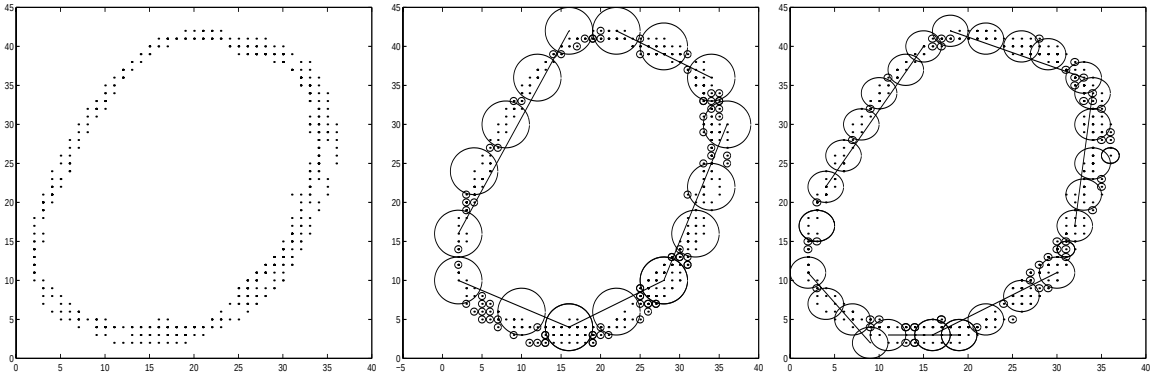

Figure 4.70: Letter O

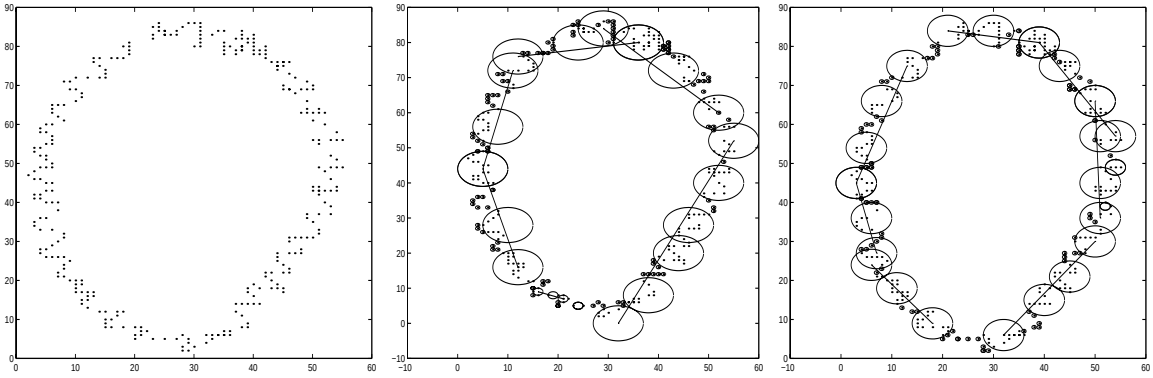

Figure 4.71: Letter O

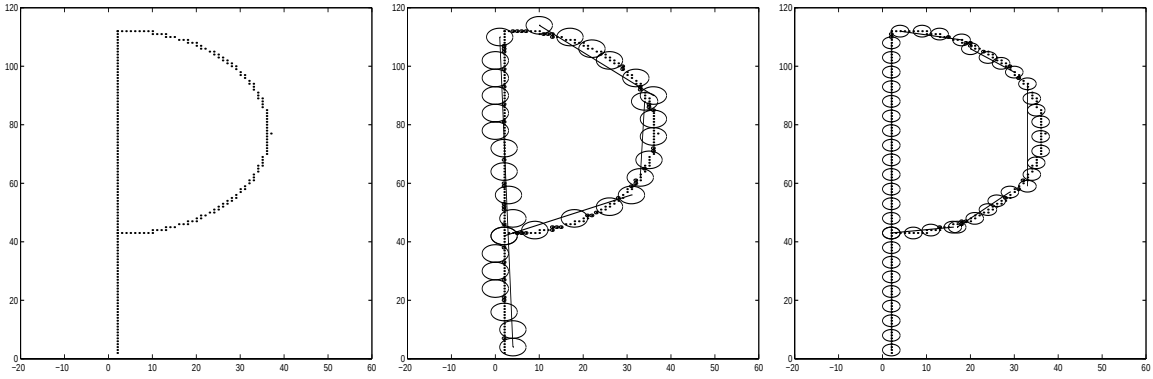

Figure 4.72: Letter P

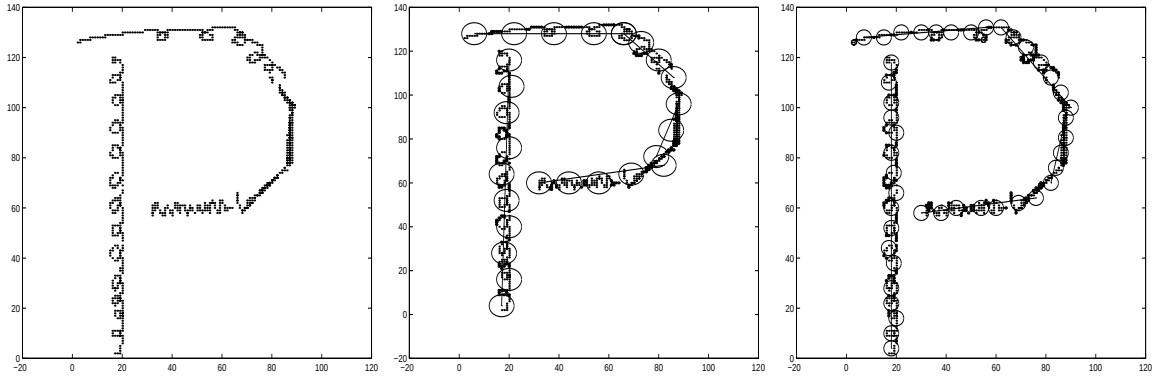

Figure 4.73: Letter P

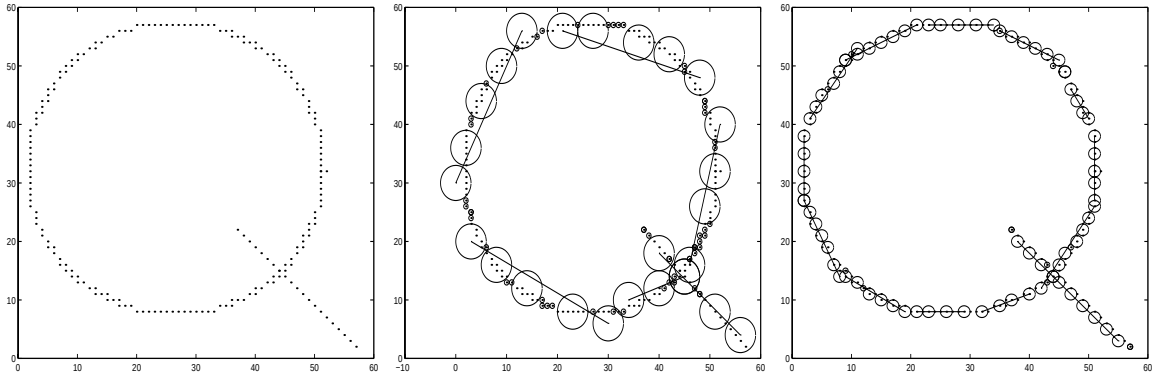

Figure 4.74: Letter Q

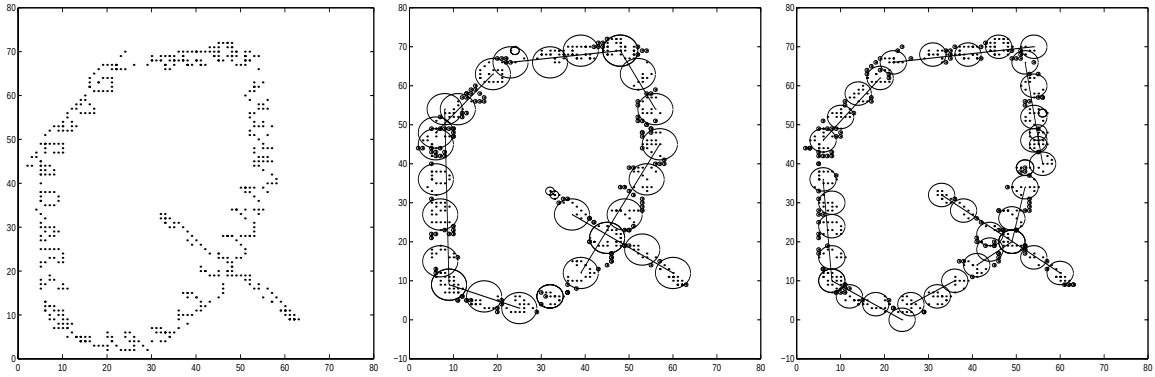

Figure 4.75: Letter Q

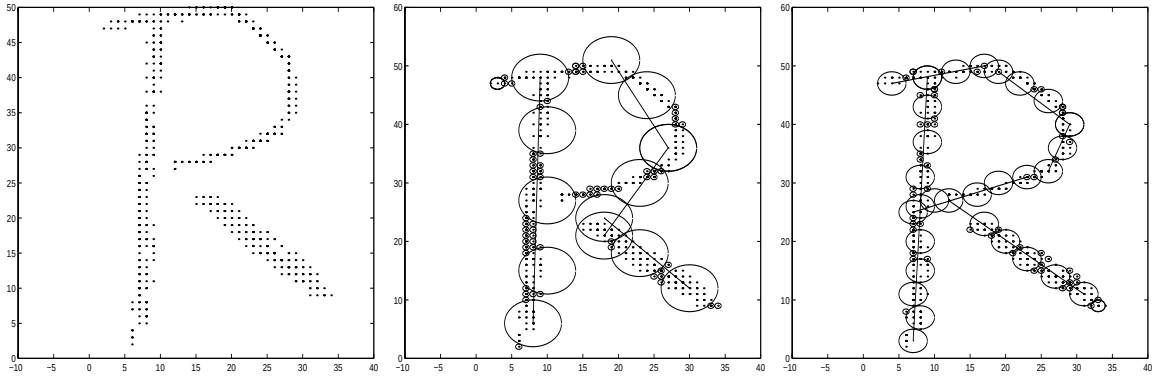

Figure 4.76: Letter R

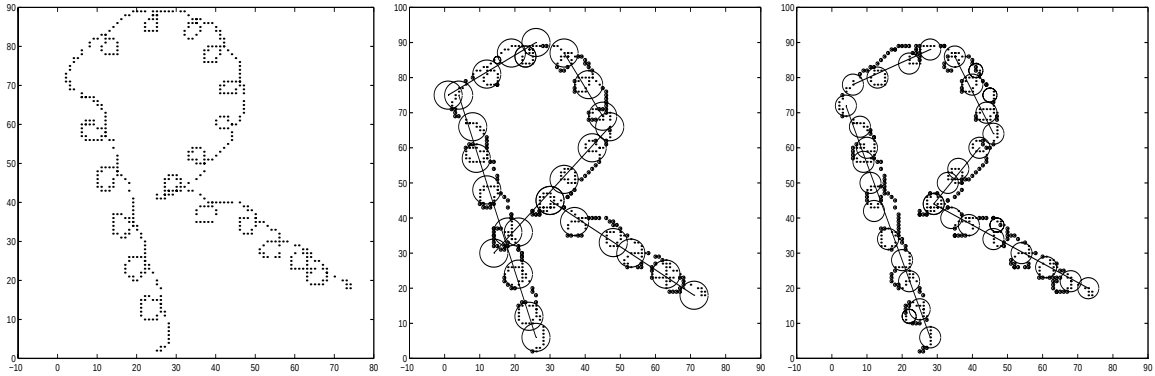

Figure 4.77: Letter R

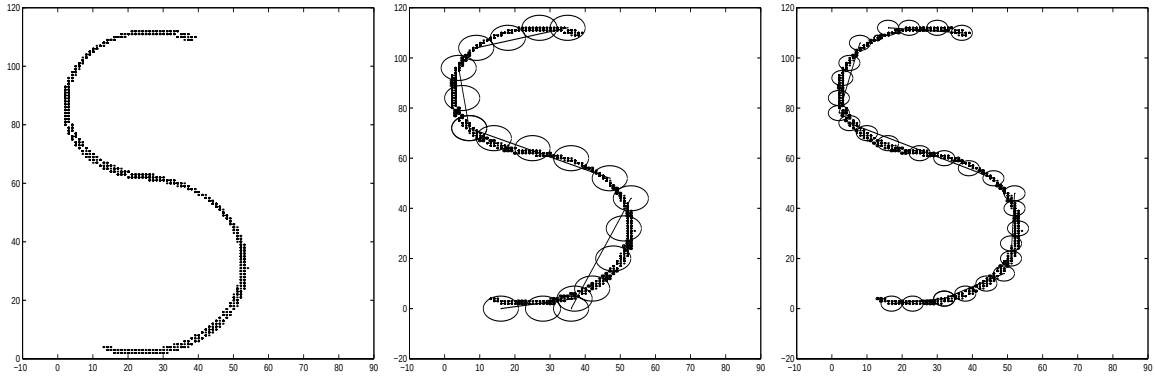

Figure 4.78: Letter S

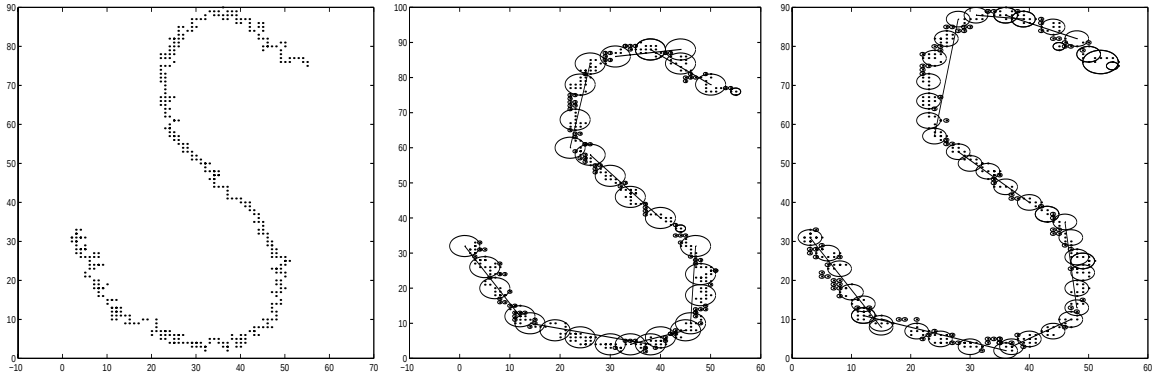

Figure 4.79: Letter S

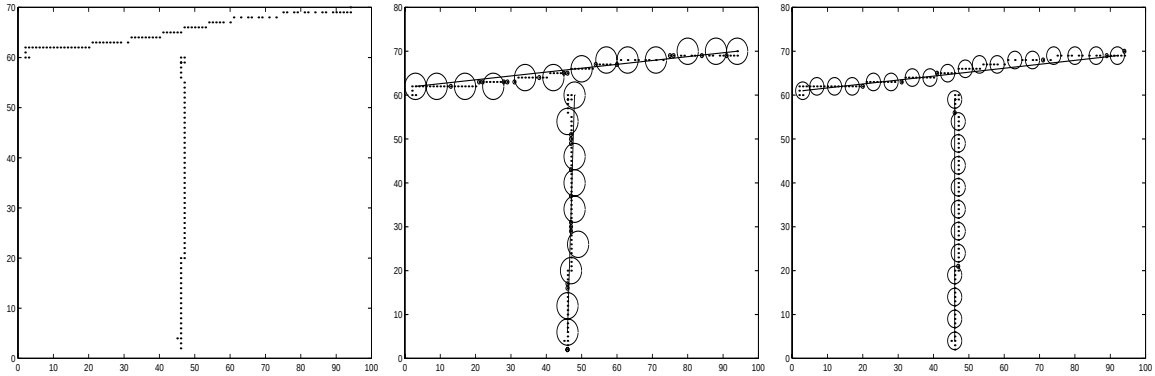

Figure 4.80: Letter T

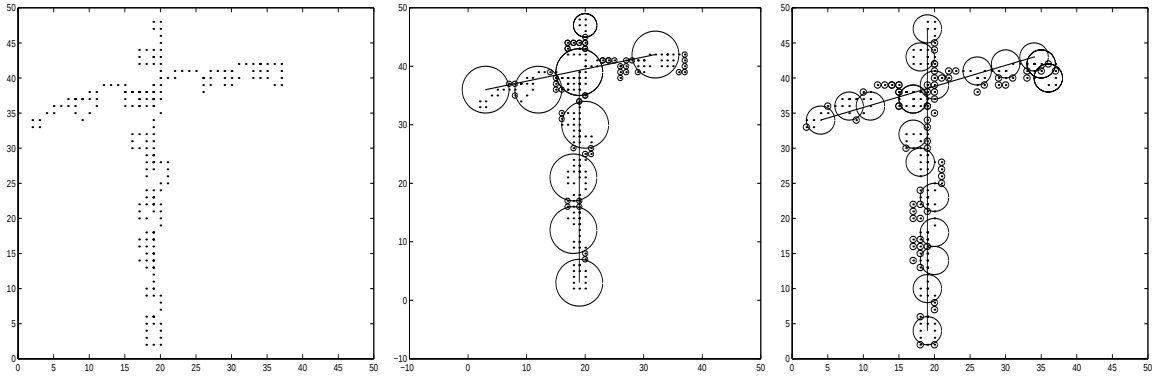

Figure 4.81: Letter T

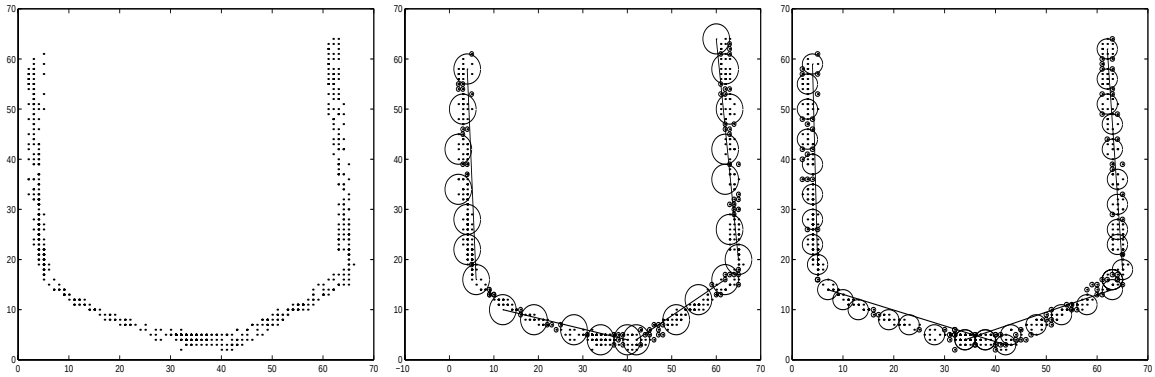

Figure 4.82: Letter U

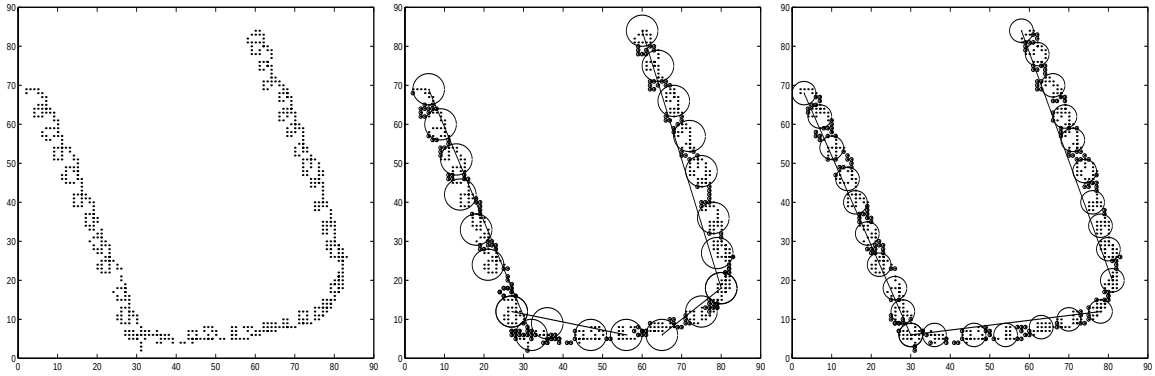

Figure 4.83: Letter U

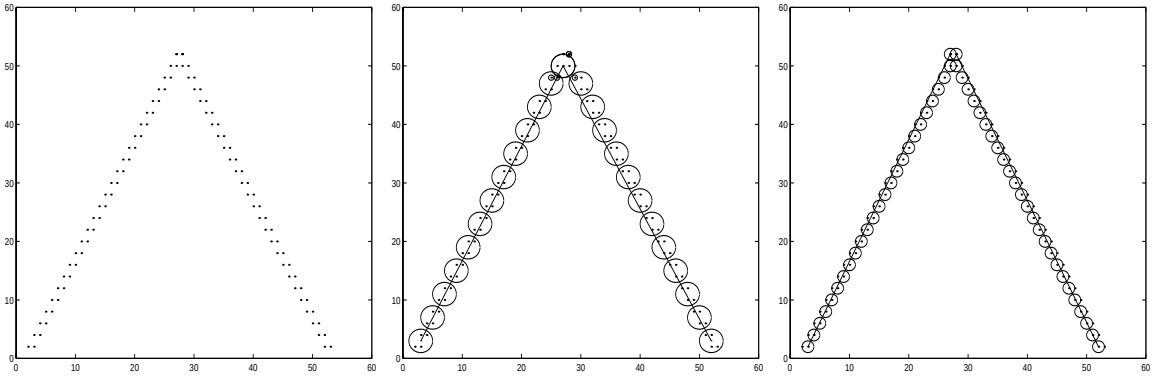

Figure 4.84: Letter V

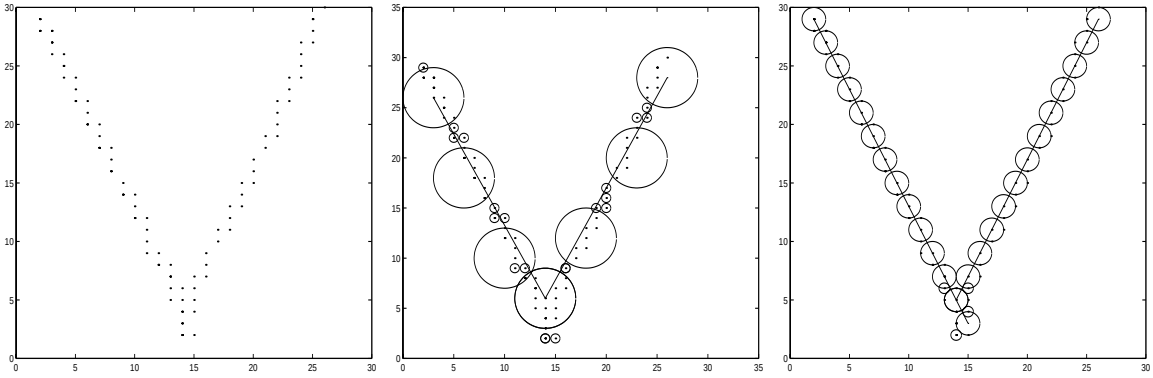

Figure 4.85: Letter V

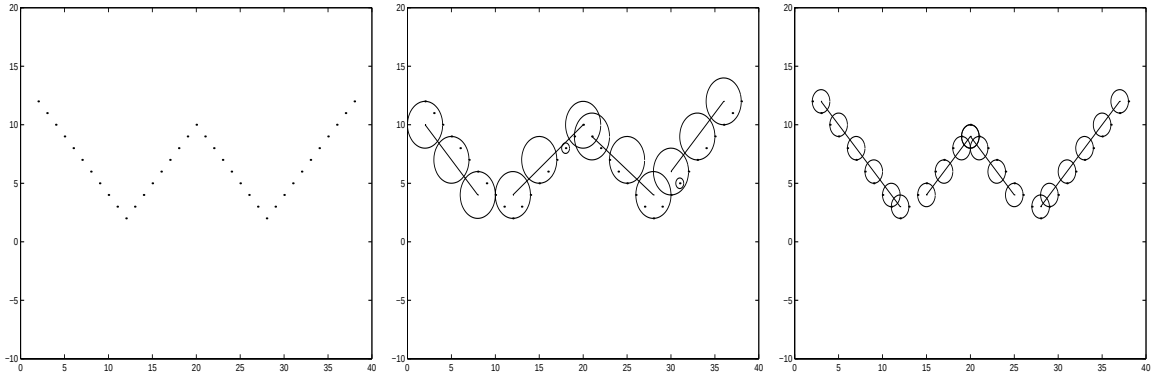

Figure 4.86: Letter W

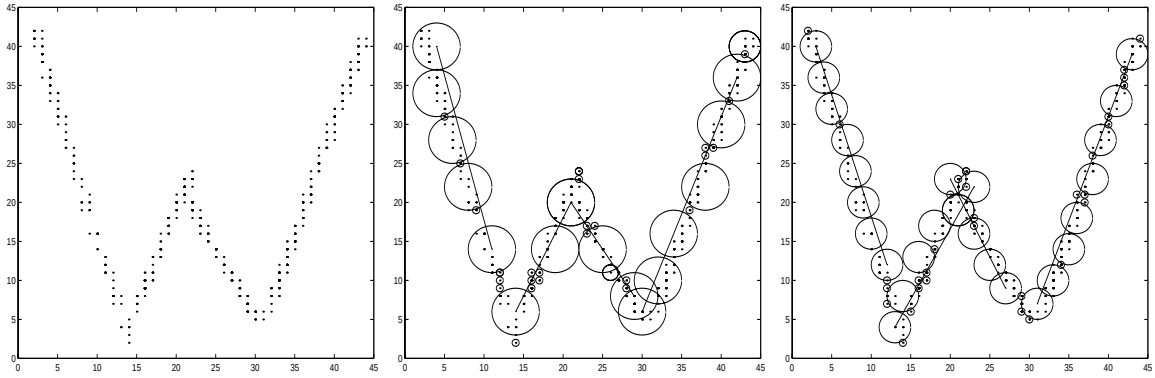

Figure 4.87: Letter W

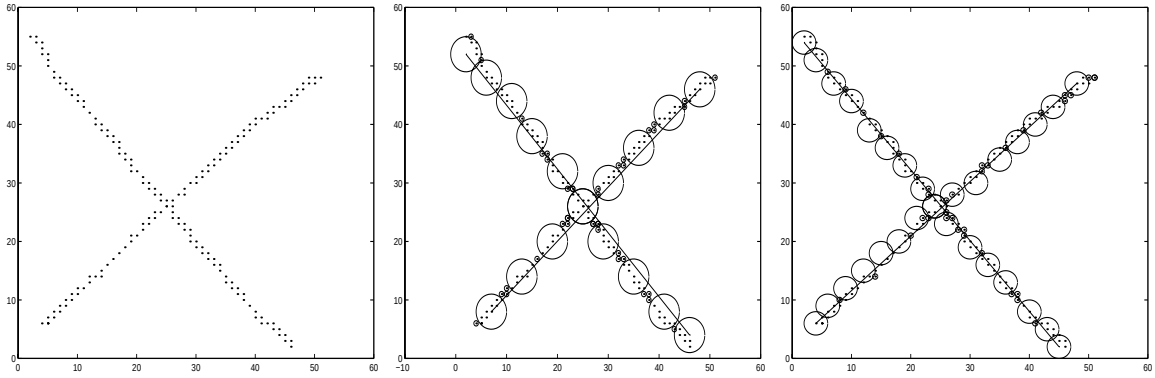

Figure 4.88: Letter X

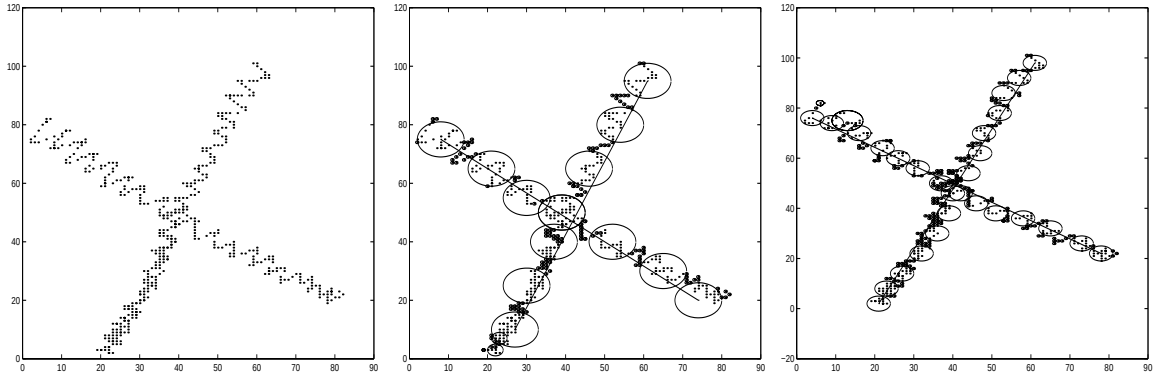

Figure 4.89: Letter X

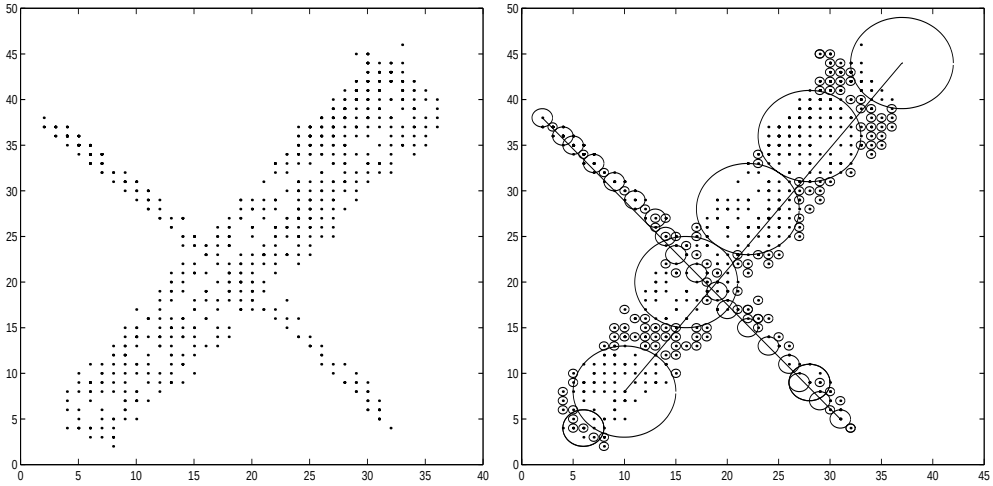

Figure 4.90: Letter X

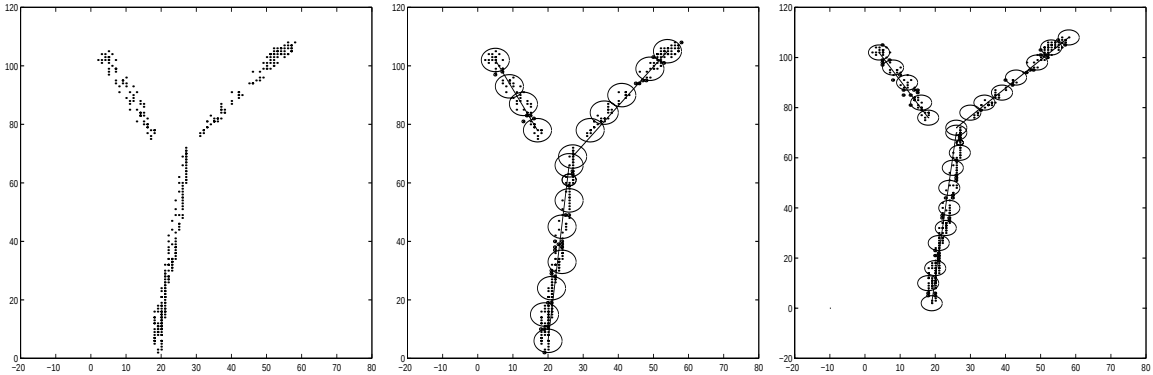

Figure 4.91: Letter Y

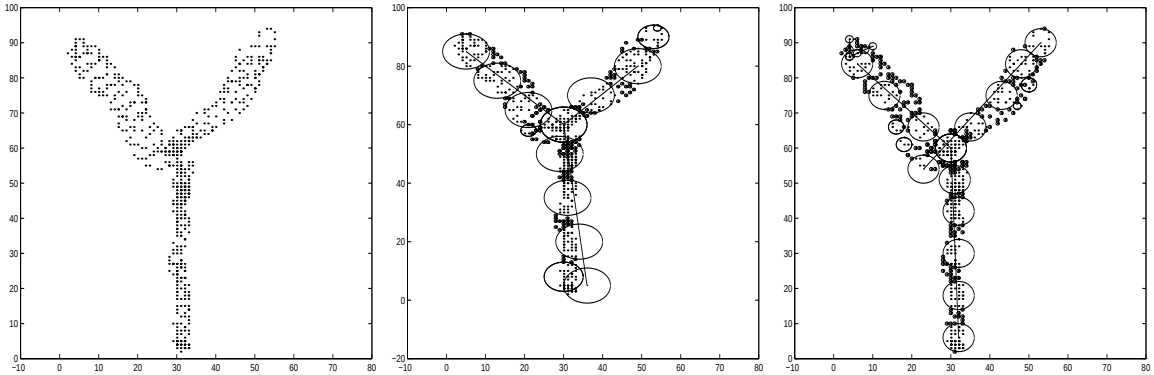

Figure 4.92: Letter Y

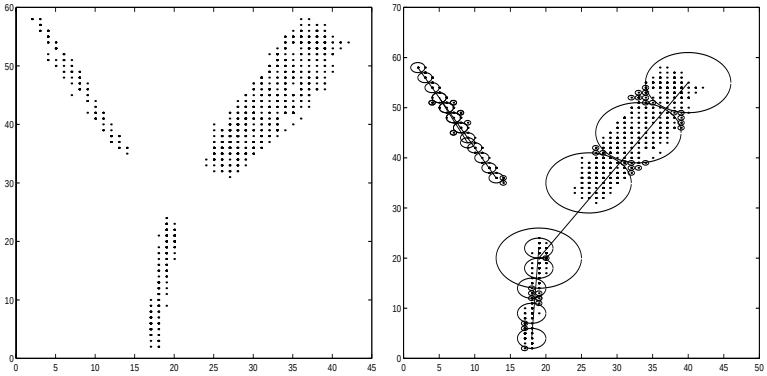

Figure 4.93: Letter Y

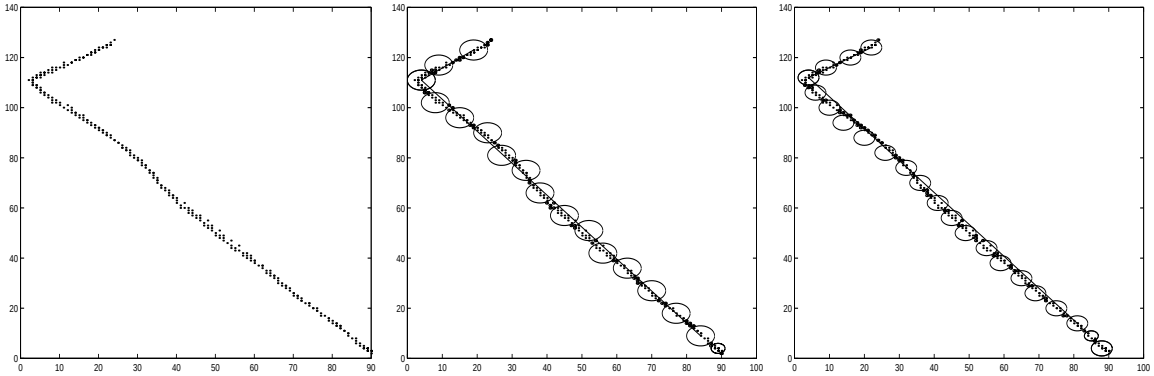

Figure 4.94: L-junction

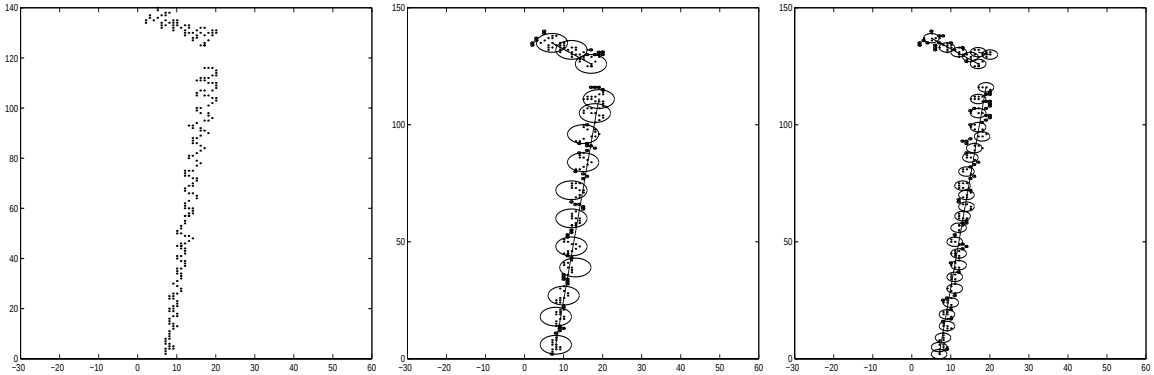

Figure 4.95: L-junction

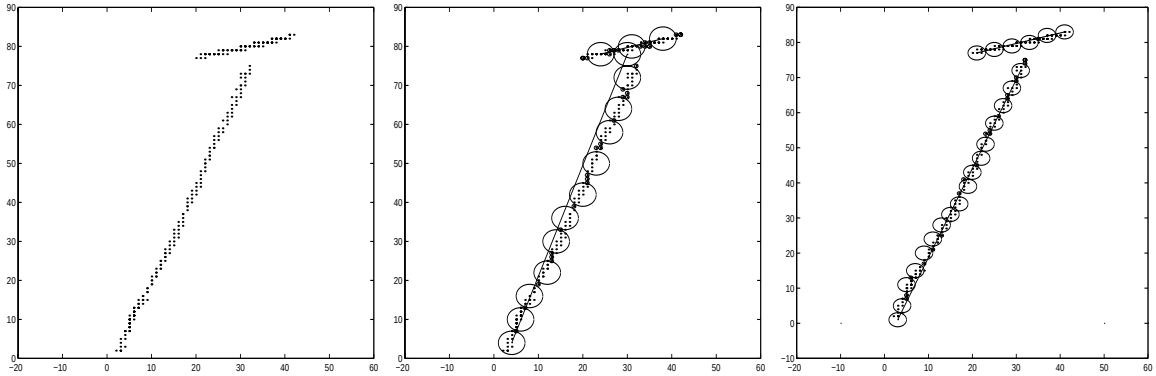

Figure 4.96: T-junction

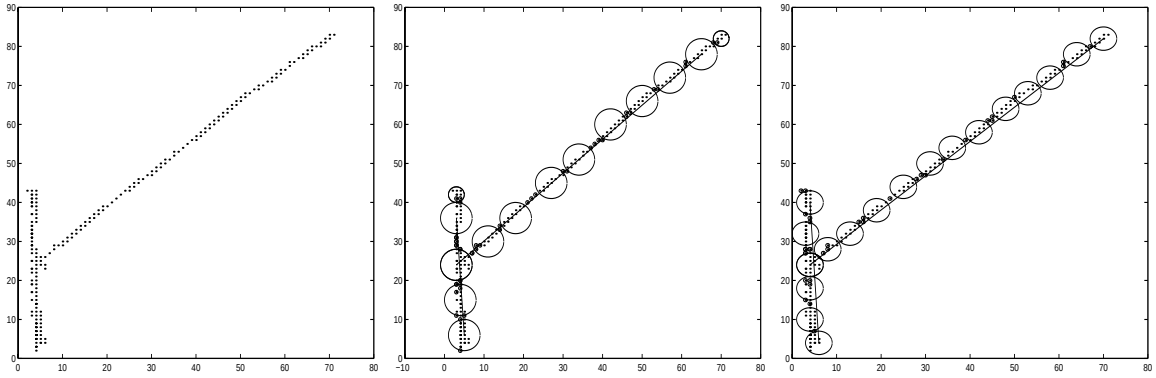

Figure 4.97: T-junction

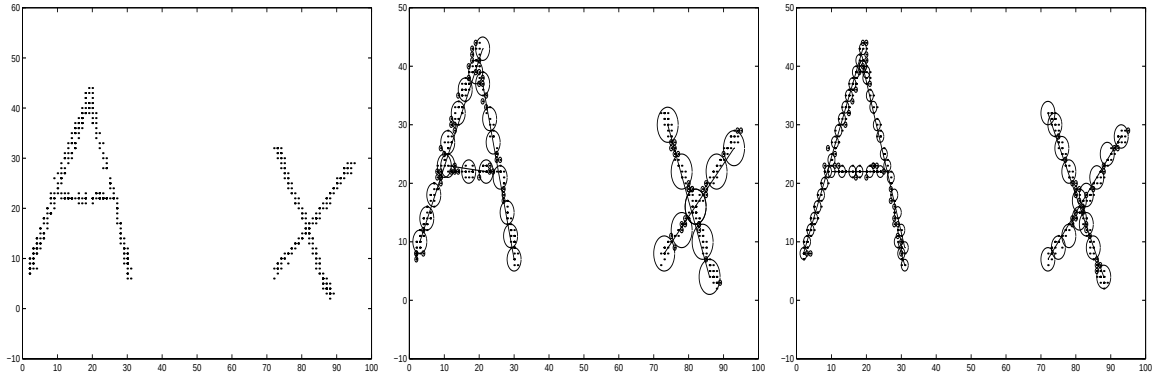

Figure 4.98: Multiple letters

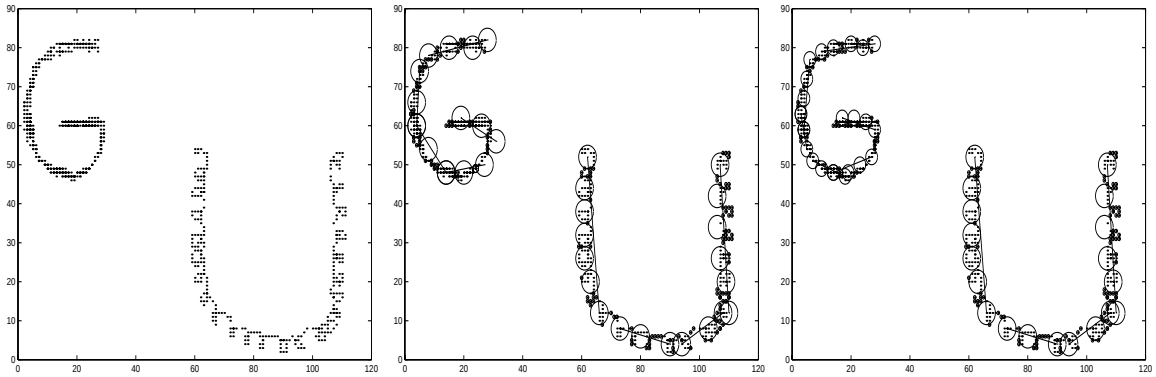

Figure 4.99: Multiple letters

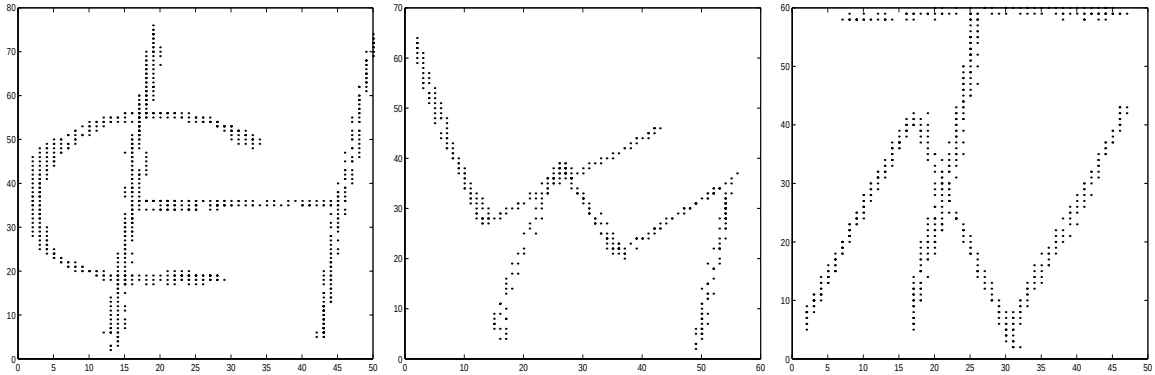

Figure 4.100: Failed Interpreted Multiple letters

4.3 Examples of Failure

Not every character can be correctly recognized. Some letters are easily confused. In our experiment, letter U and letter C, letter V and letter L are the most commonly confused character pairs. Especially when we are dealing with a rotation invariant property, what the orientation is does not matter. Then it is obvious that U and C could be confused. The same thing happens to V and L. One way to clarify this kind of confusion is to implement higher level compositions. It will eliminate these ambiguities. Furthermore, the letter I and the letter J could be confused. This happens when the bottom stroke is not completely perpendicular to the vertical stroke.

There are several examples in Figure 4.100 which cannot succeed in recognition. The common property in these examples is these letters are close or even overlap with each other. It is found that this is a weak point of the code at this current stage. We will talk

about how they fail and will suggest an idea to resolve this problem.

The first example should be a letter C and a letter H. However, The result becomes an upside down and flipped letter P on the left hand side and a T-junction on the right hand side. This is understandable because there is a letter P, though not in the correct orientation from our perspective, and the rest of the image is a T-junction like $\dashv$.

The second image should be interpreted as an L on the left upper side and an M on the right lower part. At least, this is what I thought when I wrote it. It turns out that the machine thinks it is a W and an M. It is because the composition rules are greedy as well as liberal. It tends to connect anything it can compose.

As for the third case, we would expect it is an N and a T. But the result is an F and an L-junction. It is obvious that there is an F if we include the two strokes in letter T and the rightmost stroke of N. And since this letter F has the biggest bit gained, the machine always picks up this F.

From the above examples, we can think of some reasons why this machine fails. One of the reasons is that the composition rules are too greedy. This is unavoidable when we want our machine to be robust. Robustness means that the compositions have more flexibility. This flexibility is very good for many examples. However, it is not so good for these failed examples.

Another reason is the rotation invariant property of the machine. It is very useful when this property is applied in a single letter case (like Figure 4.44). When we go into multiple letter cases, this property has many chances to confuse things (like the third case in Figure 4.100). If we get rid of the rotation invariant property, this problem would disappear. However, this would make the machine less general which means it would have difficulty in recognizing rotated letters.

One way to resolve these failures is to restart the greedy algorithm. It means that we can have a different choice of the *first* chosen object. After the first chosen object, we choose successively the next best bits-gained object and so on. Actually, in Figure 4.100, the intended letters are in the $MAIN\_LIST$. We would assume that the greedy algorithm can get these intended letters for us. However, this method does not always work. For example, in the third image of Figure 4.100, restarting the greedy algorithm would not

give us the intended recognition, T and N. The problem comes from the greedy algorithm itself. This algorithm just picks successively in terms of bits gained. Even though the result is the most bits gained from all of these restart procedures, it still cannot guarantee that the recognition is satisfactory. Sometimes it turns out a more complicated interpretation because of the bits gained. In this example, the output result is an A, an N and a junction. The recognized letter A is composed of the top line of the "original" T, the rightmost line of "original" N and the right part of the middle of the "original" N. The recognized letter N is composed of the leftmost line of "original" N, the left part of the middle of "original" letter N and the upper part of the lower stroke of the letter T. The lower part of the lower stroke of the letter T and some of the middle of the letter N make up the recognized junction.

It is apparent that restart is not the solution to this problem. There is another approach to try. If we can add rules that allow two letters to bind together even if they overlap, we may solve this problem. Because if we combine the C and H, in the first example, to form a *overlapped* string "CH", this object "CH" would be the dominant object in the $MAIN\_LIST$. The greedy algorithm would pick up this object easily.

Chapter 5

Conclusion

In this thesis, we equip labeled trees with probabilities which are recursively defined by their subtrees. We also discuss the "benefit" of treating two constituents as a composite. The quantitative tool, which is a likelihood ratio, to measure this "benefit" is provided.

Besides the mathematical background of the probability and likelihood ratio, to make multi-scale recognition happen, a special algorithm should be developed. Our method is to employ a set of lists taking care of different resolutions. The algorithm will loop through all these lists picking the largest-bits-gained object, doing all the binding it can, and putting the newly created objects in the corresponding lists. If the user wants to derive the recognition from the machine, he/she can stop the (endless) picking-binding procedure and the code will perform the greedy algorithm. This algorithm selects the best (most bits gained) object, and then the second-best with respect to the chosen object(s), and so on, to yield the scene interpretation.

The experiment we run is on the binary-valued $N \times N$ square. The set of terminals is the set of disks. The set of labels is pretty much the linelet, line, junctions and alphabets. With the careful design of the binding rules and binding supports, we get satisfactory results. These results show that the correct recognition (no matter what the resolution is) can be done. So future applications are promising. This pattern-recognition design can be used to recognize even higher-level objects.

Bibliography

[1] Adnan Amin, Humoud AL-Sadoun and Stephen Fischer. Hand-Printed Arabic Character Recognition System Using An Artifical Network. *Pattern Recognition* vol. 29, no. 4 pp. 663-675, 1996.

[2] Elie Bienenstock, Stuart Geman and Daniel Potter. Compositoonality, MDL Priors, and Object Recognition, Division of Applied Mathematics, Brown University, 1996.

[3] Jinhai Cai and Zhi-Qiang Liu. Integration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 263-270, 1999.

[4] Kam-Fai Chan, Dit-Yan Yeung. An Efficient Syntactic Approach to Structural Analysis of On-Line Handwritten Mathematical Expressions. *Pattern Recognition* vol. 33 no. 3, 2000.

[5] C. H. Chen, L..F. Pau and P. S. P. Wang. Handbook of Pattern Recognition and Computer Vision. World Scientific Publishing Co. 1993.

[6] Zhiyi Chi. Probability Models For Complex System. PhD thesis, Division of Applied Mathematics, Brown University, 1998

[7] N. Chomsky. Syntatic Structures. Mouton, 1976.

[8] N. Chomsky. Knowledge of Language: Its Nature, Orogin, and Use. Praeger, 1986.

[9] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest. Introduction to Algorithms. McGraw-Hill Book Company, 1994.

[10] A. EI-Yacoubi, M. Gilloux, R. Sabourin, and C.Y. Suen. An HMM-Based Approach for off-line Unconstrained Handwritten Word Modeling and Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 752-760, 1999.

[11] King Sun Fu. Syntactic Pattern Recognition and Applications. Prentice-Hall, 1982.

[12] Stuart Geman, Daniel F. Potter and Zhiyi Chi. Composition Systems. Division of Applied Mathematics, Brown University, 1998.

[13] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850-863, 1993.

[14] Anil K. Jain, Robert P.W. Duin, and Jianchang Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4-37, 2000.

[15] Nei Kato, Masato Suzuki, Shin'ichiro Omachi, Hirotomo Aso, and Yoshiaki Nemoto. A Handwritten Character Recognition System Using Directional Element Feature and Asymmetric Mahalanobis Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 258-262, 1999.

[16] P.S.Laplace. Esssai philosophique sur les probabilités. 1812. Translation of Truscott and Emory, New York, 1902.

[17] Seong-Whan Lee. Off-line Recognition of Totally Unconstrained Handwritten Numerals Using Multilayer Cluster Neural Network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 648-652, 1996.

[18] Shunji Mori, Hirobumi Nishida and Hiromitsu Yamada. Optical Character Recognition. New York: Wiley, 1999.

[19] Albert Nigrin, Neural Networks for Pattern Recognition. MIT Press, 1993

[20] Hirobumi Nishida and Shunji Mori. Algebraic Description of Curve Structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 5, pp. 516-533, 1992.

[21] Hirobumi Nishida and Shunji Mori. An Algebraic Approach to Automatic Construction of Structural Models *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 12, pp. 1298-1311, 1993.

[22] II-Seok Oh, Jin-Seon Lee, and Ching Y. Suen. Analysis of Class Separation and Combination of Class-Dependent Features for Handwriting Recognition.*IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 1089-1094, 1999.

[23] Jaehwa Park, Venu Govindaraju, and Sargur N. Srihari. OCR in a Hierarchical Feature Space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 400-406, 2000.

[24] T. Pavlidis, Structural Pattern Recognition. New York: Springer-Verlag, 1977.

[25] Daniel Frederic Potter. Compositional Pattern Recognition. PhD thesis, Division of Applied Mathematics, Brown University, 1999.

[26] A. Rosenfeld and A. Kak. Digital Picture Processing, vol. 2. New York : Academic Press, 1982.

[27] Yuan Y. Tang, Lo-Ting Tu, Jiming Liu, Seong-Whan Lee, Win-Win Lin, and Ing-Shyh Shyu. Offline Recognition of Chinese Handwriting by Multifeature and Multilevel Classification.*IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 556-561, 1998.

[28] Pak-Kwong Wong and Chorkin Chan. Off-line Handwritten Chinese Character Recognition as a Compound Bayes Decision Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 9, pp. 1016-1023, 1998.

Appendix A

$Q_l(s, v, a)$ in the likelihood ratio

Q_l is a density function of s, v and a, where $s = \frac{r_\beta}{r_\alpha}, v = R_{-\theta_\alpha}\frac{x_\beta - x_\alpha}{r_\alpha}, a = \theta_\beta - \theta_\alpha$. We assume that s, v and a are independent. So Q_l is the product of the density function of s, the density function of v, and the density function of a, i.e., $Q_l = f_s(s)f_v(v)f_a(a)$, where f_s is the density of s, f_v is the density of v and f_a is the density of a.

1. linelet = disk + disk

x_α = the center of the disk α, x_β = the center of the disk β.
r_α = the radius of the disk α, r_β = the radius of the disk β.
θ_α is arbitrary, θ_β is arbitrary.
$lns \sim$ N(0,1).
$|v| \sim$ Gamma(3,1).
$a \sim$ U(0,2π).

2. line = linelet + disk

x_α = the center of the disk in α that is near β, x_β = the center of the disk.
r_α = the radius of the disk in α that is near β, r_β = the radius of the disk.
θ_α = the angle from α to the x-axis, θ_β is arbitrary.
$lns \sim$ N(0,1).
$v \sim$ N((2,0),$I_{2\times 2}$).
$a \sim$ U(0,2π).

3. line = line + disk

x_α = the center of the disk in α that is near β, x_β = the center of the disk.
r_α = the radius of the end disk in α that is near β, r_β = the radius of the disk.
θ_α = the angle from α to the x-axis, θ_β is arbitrary.
$lns \sim$ N(0,1).
$v \sim$ N((2,0),$I_{2\times 2}$).

$a \sim$ U(0,2π).

4. L-junction = line + line (see Figure(A.1): left side)

x_α = the center of end disk in α that is near β, x_β = the center of the end disk in β that is near α.
r_α = the length of the line, r_β = the length of the line.
θ_α = the angle from α to the x-axis, θ_β = the angle from β to the x-axis.
$v \sim$ N((0,0),$I_{2\times 2}$).
$|a| \sim$ U($\frac{1}{6}\pi$,$\frac{5}{6}\pi$).

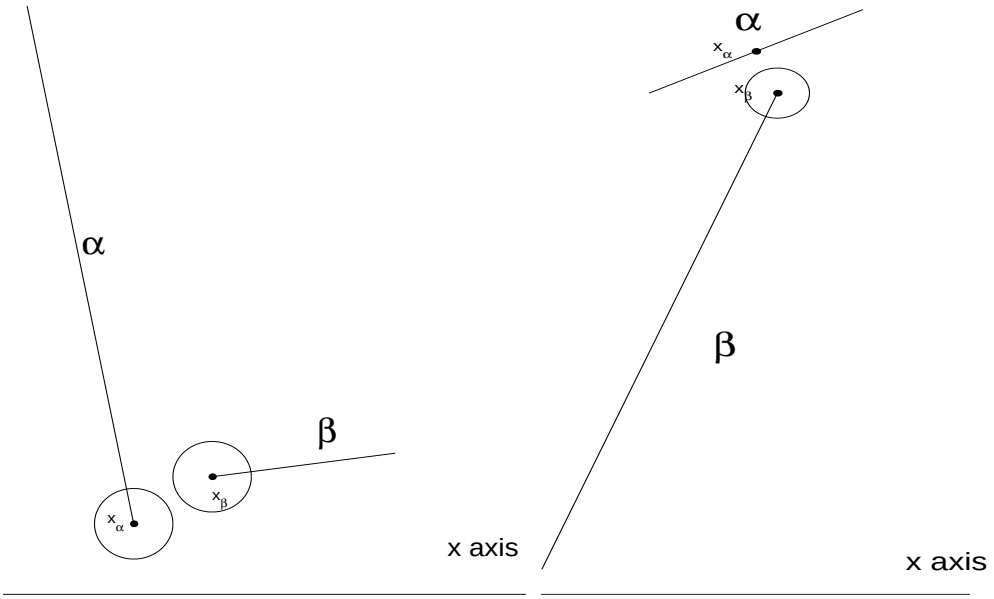

Figure A.1: Rule 4 and Rule 5

5. T-junction = line + line (see Figure(A.1): right side)

x_α = the mid point of the line α, x_β = the center of the end disk in β that is near α.
r_α = the length of the line, r_β = the length of the line.
θ_α = the angle from α to the x-axis, θ_β = the angle from β to the x-axis.
$v \sim$ N((0,0),$I_{2\times 2}$).
$|a| \sim$ U($\frac{1}{6}\pi$,$\frac{5}{6}\pi$).

6. Letter-A = Letter-V + line (see Figure(A.2): left side)

x_α = the mid point of one line in α, x_β = the center of the end disk in β that is near x_α.
r_α = the length between mid points of lines in α, r_β = the length of the line.

θ_α = the angle from the line to which x_α belongs to the x-axis, θ_β = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($-\frac{1}{3}\pi$,$-\frac{1}{6}\pi$).

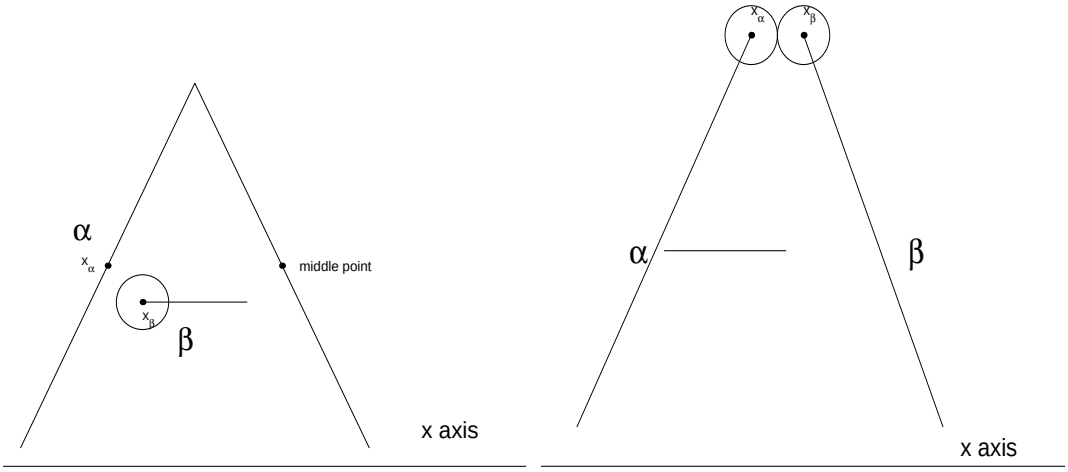

Figure A.2: Rule 6 and Rule 7

7. Letter-A = T-junction + line (see Figure(A.2): right side)

x_α = the center of the end disk in α which should be the top point of letter A, x_β = the center of the end disk in β that is near x_α.
r_α = the length of the line to which x_α belongs, r_β = the length of the line.
θ_α = the angle from the line to which x_α belongs to the x-axis, θ_α = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($\frac{1}{6}\pi$,$\frac{2}{3}\pi$).

8. Letter-B = Letter-P + letter-C (see Figure(A.3): left side)

x_α = the mid point of the straight line in α, x_β = the center of the end disk in β that is near x_α.
r_α = half of the length of the straight line in α, r_β = the distance between two end disks in β.
θ_α = the angle from the straight line in α to the x-axis, θ_β = the angle from the 'end line' of β to the x-axis (see Figure).

$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($-\frac{1}{2}\pi, \frac{1}{6}\pi$).

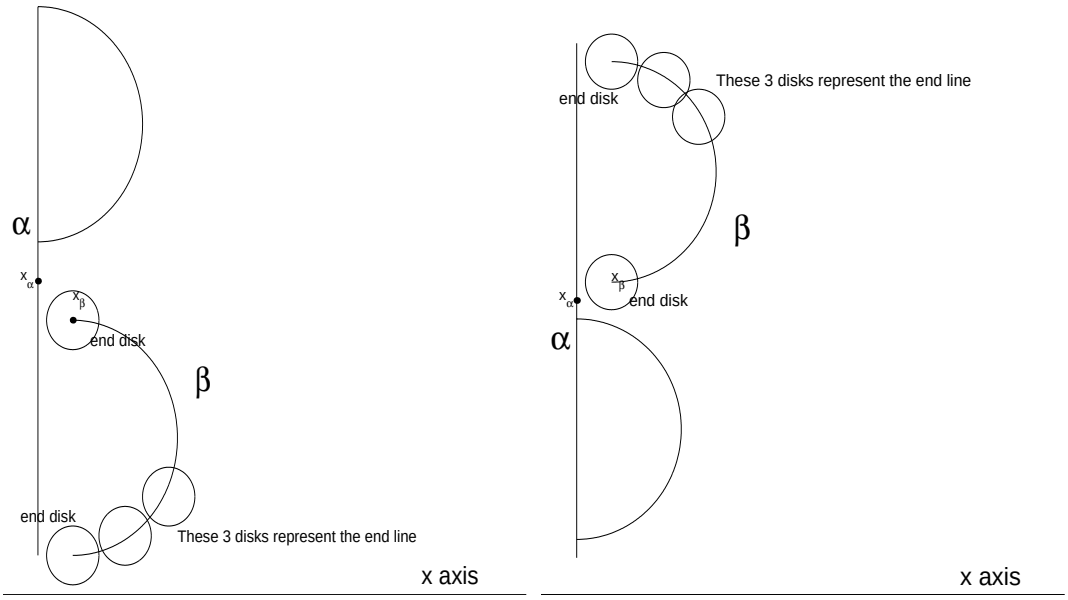

Figure A.3: Rule 8 and Rule 9

9. Letter-B = Letter-b + Letter-C (see Figure(A.3): right side)

x_α = the mid point of the straight line in α, x_β = the center of the end disk in β that is near x_α.
r_α = half of the length of the straight line in α, r_β = the distance between the two end disks in β.
θ_α = the angle from the straight line in α to the x-axis, θ_β = the angle from the 'end line' of β to the x-axis (see Figure).
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($-\frac{1}{2}\pi, \frac{1}{6}\pi$).

10. Letter-C = line + line (see Figure(A.4): left side)

x_α = the center of the end disk in α that is near β, x_β = the center of the end disk in β that is near α.
r_α = the length of the line, r_β = the length of the line.
θ_α = the angle from α to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.5,2).
$v \sim$ N((0,0),$I_{2\times 2}$).

$a \sim \mathrm{U}(\frac{1}{12}\pi, \frac{1}{2}\pi)$.

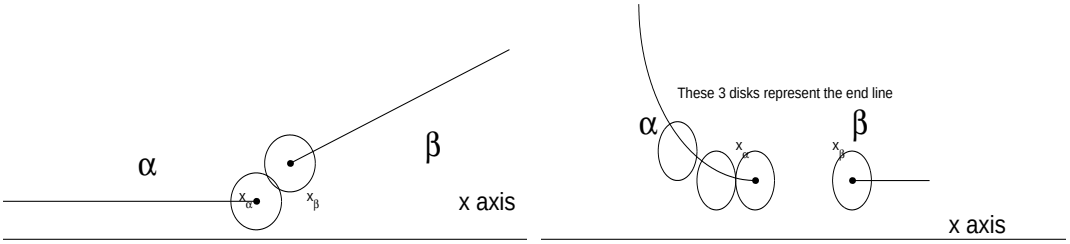

Figure A.4: Rule 10 and Rule 11

11. Letter-C = Letter-C + line (see Figure(A.4): right side)

x_α = the center of the end disk in α that is near β, x_β = the center of the end disk in β that is near α.
r_α = the length of the curve[1], r_β = the length of the line.
θ_α = the angle from the 'end line' in α that is near x_β to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.1,10).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim \mathrm{U}(\frac{1}{12}\pi, \frac{1}{2}\pi)$ or $\mathrm{U}(-\frac{1}{2}\pi, -\frac{1}{12}\pi)$ depends on the orientation.

12. Letter-D = Letter-C + line (see Figure(A.5): left side)

x_α = the center of one of the end disks in α, x_β = the center of the end disk in β that is near x_α.
r_α = the distance between the two end disks of α, r_β = the length of the line.
θ_α = the angle from the line connecting the two end disks of α to the x-axis, θ_β = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).

[1] By 'length of the curve' we mean the sum of the lengths of the lines which make up the curve'

$a \sim$ N(0,1).

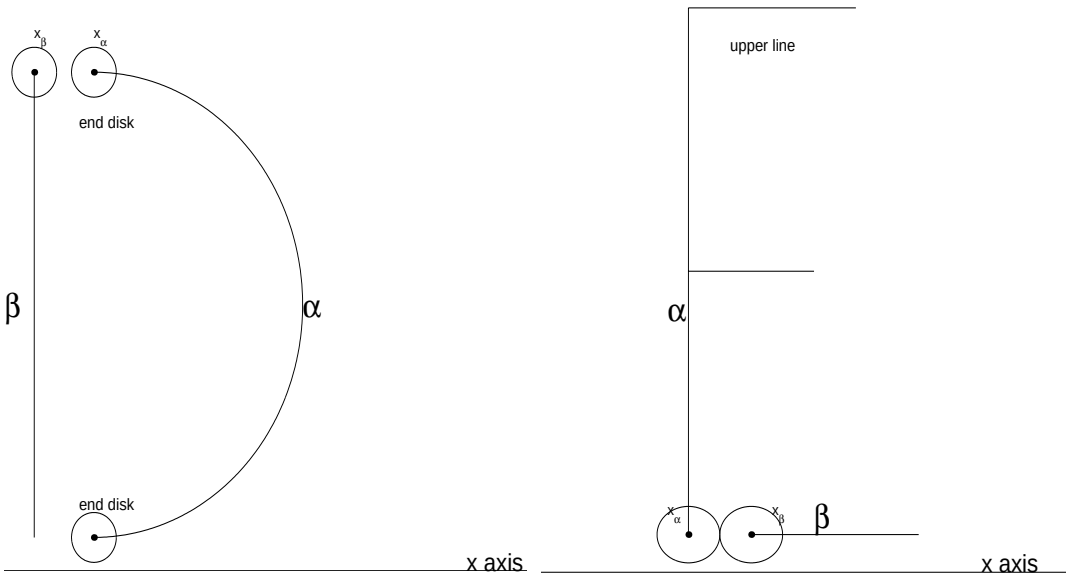

Figure A.5: Rule 12 and Rule 13

13. Letter-E = Letter-F + line (see Figure(A.5): right side)

x_α = the center of the lowest disk in the vertical line of the 'F', x_β = the center of the end disk in β that is near x_α.
r_α = the length of the upper line of the 'F', r_β = the length of the line.
θ_α = the angle from the vertical line of the 'F' to the x-axis, θ_β = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times2}$).
$a \sim$ N(-$\frac{1}{2}\pi$,1).

14. Letter-F = L-junction + line (see Figure(A.6): left side)

x_α = the mid point of one of the lines in α, x_β = the center of the end disk in β that is near x_α.
r_α = the length of the other line in α, r_β = the length of the line.
θ_α = the angle from the line to which x_α belongs to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.5,1.2).
$v \sim$ N((0,0),$I_{2\times2}$).
$a \sim$ N($-\frac{1}{2}\pi$,1).

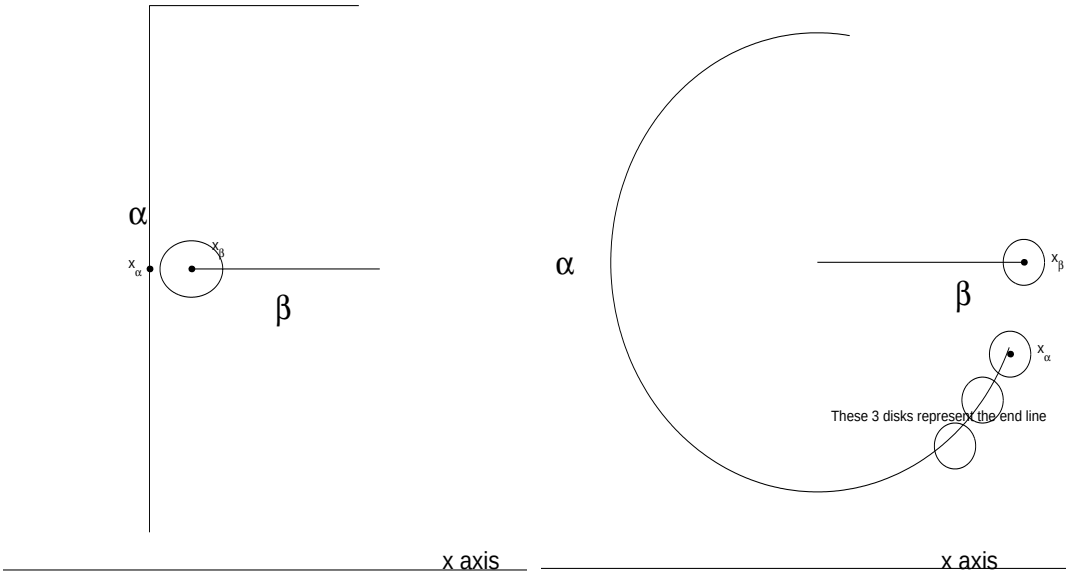

Figure A.6: Rule 14 and Rule 15

15. Letter-G_1 = Letter-C + line (see Figure(A.6): right side)

x_α = the center of the end disk in α, x_β = the center of the end disk in β that is near x_α.
r_α = the length of the curve, r_β = the length of the line.
θ_α = the angle from the end line of α to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.1,0.5).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($-\frac{5}{12}\pi$,$-\frac{1}{12}\pi$).

16. Letter-G_2 = Letter-C + Letter-L (see Figure(A.7): left side)

x_α = the center of end disk, x_β = the point of intersection[2]of two lines in β.
r_α = the length of the curve, r_β = the length of the horizontal line in β.
θ_α = the angle from the end line to the x-axis, θ_β = the angle from the line to which r_β belongs to the x-axis.
$s \sim$ U(0.1,0.5).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($-\frac{5}{12}\pi$,$-\frac{1}{12}\pi$).

17. Letter-H = Letter-T + line (see Figure(A.7): right side)

x_α = the center of the end disk in α, x_β = the mid point of β.

[2]If they do not have the point of intersection, we use the intersection of the extensions of the two lines

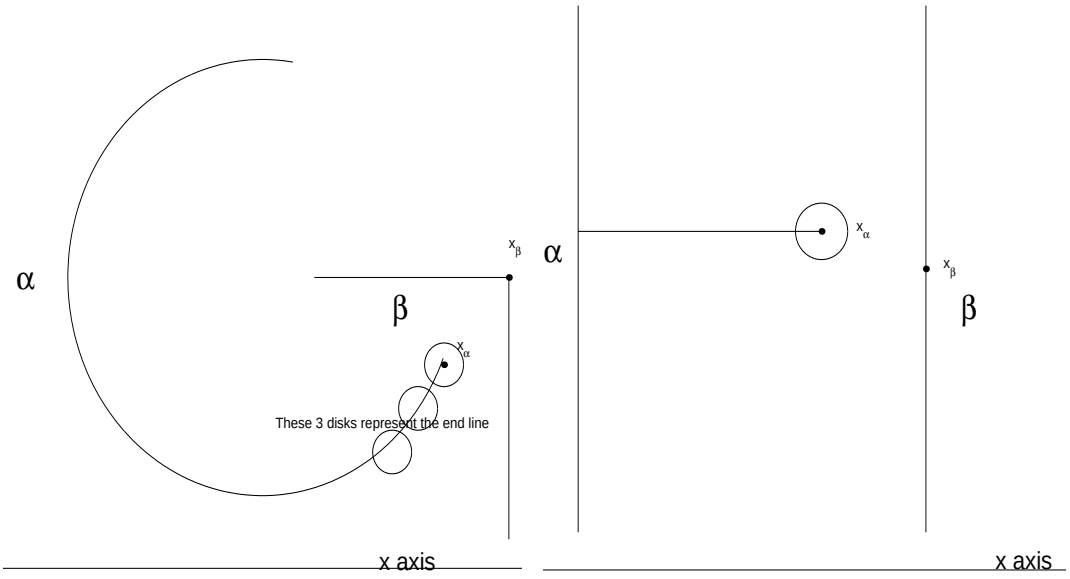

Figure A.7: Rule 16 and Rule 17

r_α = the length of the other line in α, r_β = the length of the line.
θ_α = the angle from the line to which r_α belongs to the x-axis, θ_β = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

18. Letter-I = T-junction + line (see Figure(A.8): left side)

x_α = the center of the end disk in α, x_β = the mid point of β.
r_α = the length of the other line in α, r_β = the length of the line.
θ_α = the angle from the horizontal line to the x-axis, θ_β = the angle from β to x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

19. Letter-J = Letter-C + line (see Figure(A.8): right side)

x_α = the center of the end disk of α near β, x_β = the mid point of β.
r_α = the length of the curve, r_β = the length of the line.
θ_α = the angle from the end line near β to the x-axis, θ_β = the angle from the β to the x-axis.
$s \sim$ U(0.3,1.5).
$v \sim$ N((0,0),$I_{2\times 2}$).

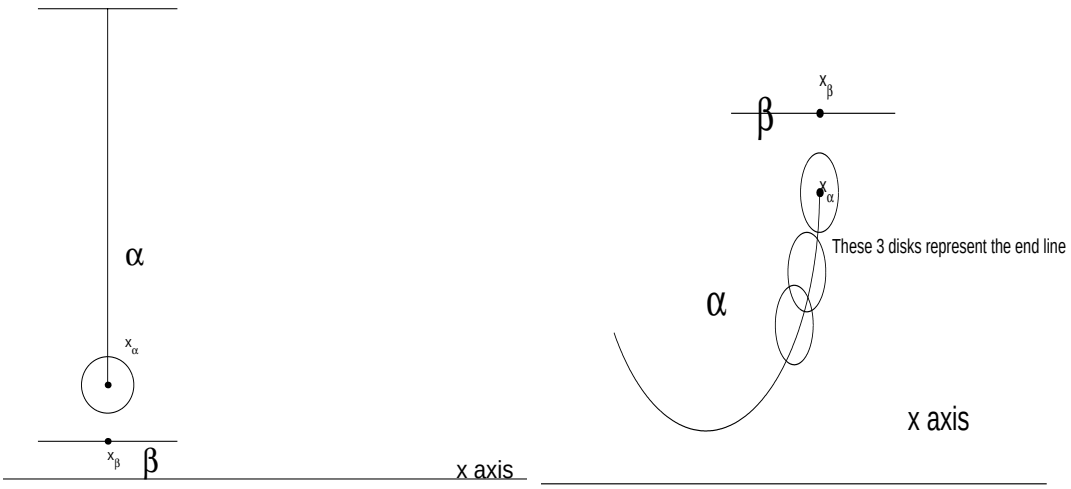

Figure A.8: Rule 18 and Rule 19

$a \sim \mathrm{N}(-\frac{1}{2}\pi,1)$.

20. Letter-K = T-junction + line (see Figure(A.9): left side)

x_α = the point of the intersection of the two lines, x_β = the center of the end disk of β near x_α.
r_α = the length of the line on the right hand side, r_β = the length of the line.
θ_α = the angle from the x-axis to the line on the left hand side of α, θ_β = the angle from β to the x-axis.
$s \sim \mathrm{U}(0.4,1.2)$.
$v \sim \mathrm{N}((0,0),I_{2\times 2})$.
$a \sim \mathrm{N}(\frac{1}{4}\pi,1)$.

21. Letter-L = line + line (see Figure(A.9): right side)

x_α = the center of the end disk of α that is near β, x_β = the center of the end disk of β that is near α.
r_α = the length of the line, r_β = the length of the line.
θ_α = the angle from α to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim \mathrm{U}(.5,2.0)$.
$v \sim \mathrm{N}((0,0),I_{2\times 2})$.
$a \sim \mathrm{N}(-\frac{1}{2}\pi,1)$.

22. Letter-M = L-junction + L-junction (see Figure(A.10): left side)

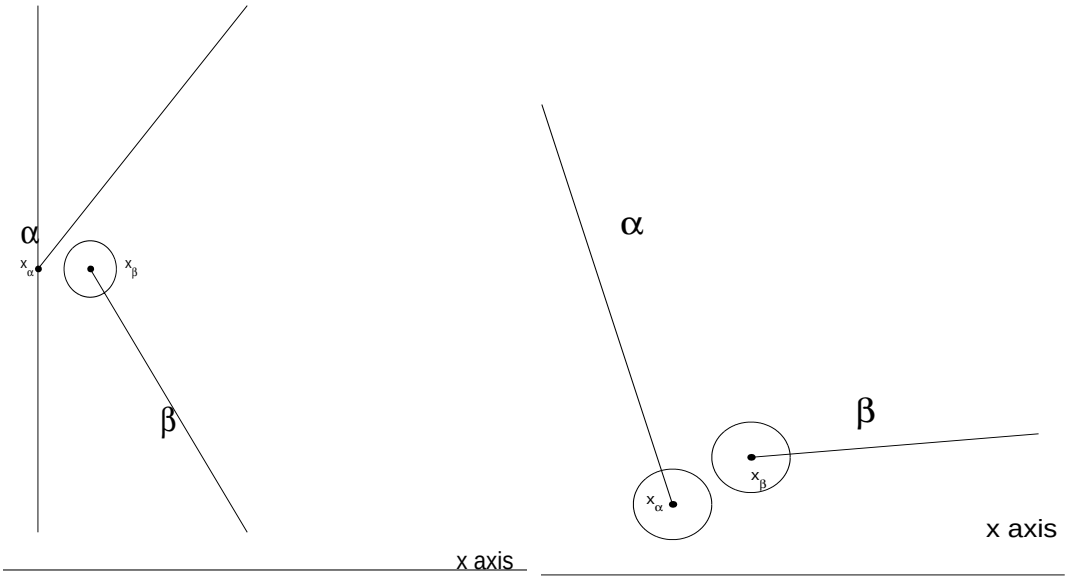

Figure A.9: Rule 20 and Rule 21

x_α = the center of end disk of the line in α that is near β, x_β = the center of end disk of the line in β that is near α.
r_α = the length of the line to which x_α belongs, r_β = the length of the line to which x_β belongs.
θ_α = the angle from the line in α which doen not include x_α to the x-axis, θ_β = the angle from the line in β which doen not include x_β to the x-axis.
$s \sim$ U(0.4,2.0).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

23. Letter-N = L-junction + line (see Figure(A.10): right side)

x_α = the center of the end disk of the line in A that is near β, x_β = the center of the end disk in β that is near α.
r_α = the length of the line to which x_α does not belong, r_β = the length of the line.
θ_α = the angle from the line to which r_α belongs to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.4,2.0).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

24. Letter-O = Letter-C + line (see Figure(A.11): left side)

x_α = the center of an end disk of α, x_β = the center of the end disk of β that is near α.
r_α = the distance between the two end disks of α, r_β = the length of the line.

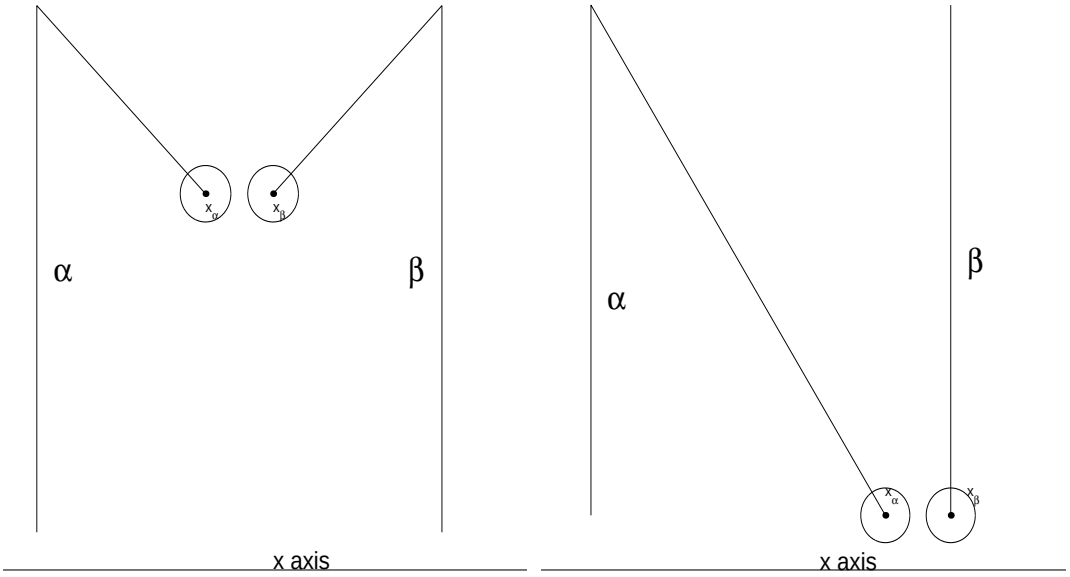

Figure A.10: Rule 22 and Rule 23

θ_α = the angle from the end line to which x_α belongs to the x-axis, θ_β = the angle from β the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($\frac{1}{12}\pi, \frac{1}{2}\pi$).

25. Letter-P = Letter-C + line (see Figure(A.11): right side)

x_α = the center of an end disk of α, x_β = the center of the end disk of β.
r_α = the distance between two end disks of α, r_β = the length of the line.
θ_α = the angle from the line connecting two end disks to the x-axis, θ_β = the angle from the β to the x-axis.
$lns \sim$ N($ln2$,1).
$v \sim$ N((0,0),$I_{2\times 2}$)
$a \sim$ N(0,1).

26. Letter-Q = Letter-O + line (see Figure(A.12): left side)

x_α = the mid point of the line in α to which the intersection point belongs, x_β = the mid point of β.
r_α = the length of the circle, r_β = the length of the line.
θ_α = the angle from the line to which the x_α belongs to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.05,0.2).
$v \sim$ N((0,0),$I_{2\times 2}$).

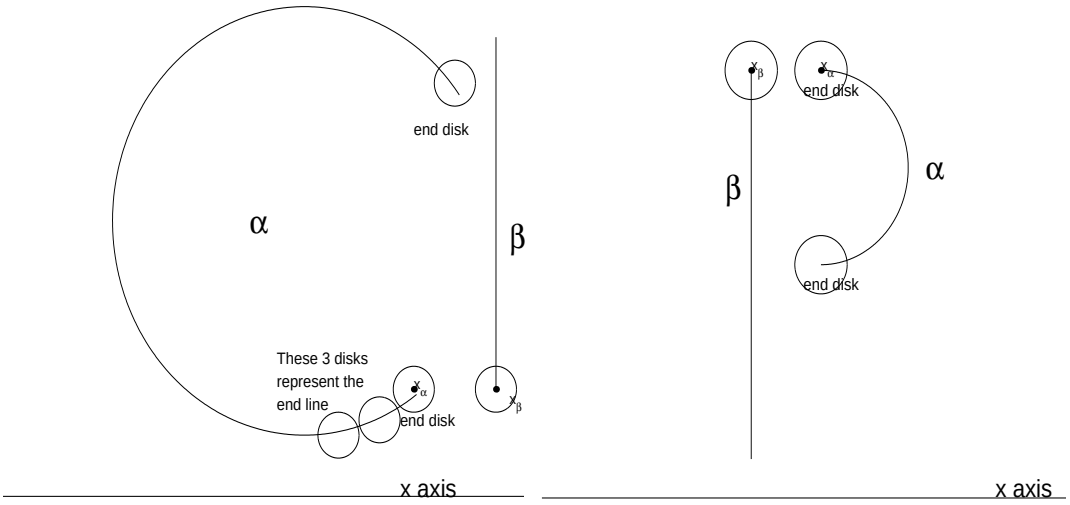

Figure A.11: Rule 24 and Rule 25

$a \sim$ N($\frac{1}{2}\pi$,1).

27. Letter-R = Letter-P + line (see Figure(A.12): right side)

x_α = the mid point of the vertical line of the Letter-P, x_β = the center of the end disk of β taht is near x_α.
r_α = half of the length of the vertical line of the Letter-P, r_β = the length of the line.
θ_α = the angle from the vertical line of the Letter-P to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.4,2.0).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($\frac{1}{6}\pi$,$\frac{1}{3}\pi$).

28. Letter-S = Letter-C + Letter-C (see Figure(A.13): left side)

x_α = the center of one end disk of α, x_β = the center of the end disk in β that is near x_α.
r_α = the length of the curve, r_β = the length of the curve.
θ_α = the angle from the line to which x_α belongs to the x-axis, θ_β = the angle from the end line to which x_β belongs to the x-axis.
$s \sim$ U(0.4,2.5).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

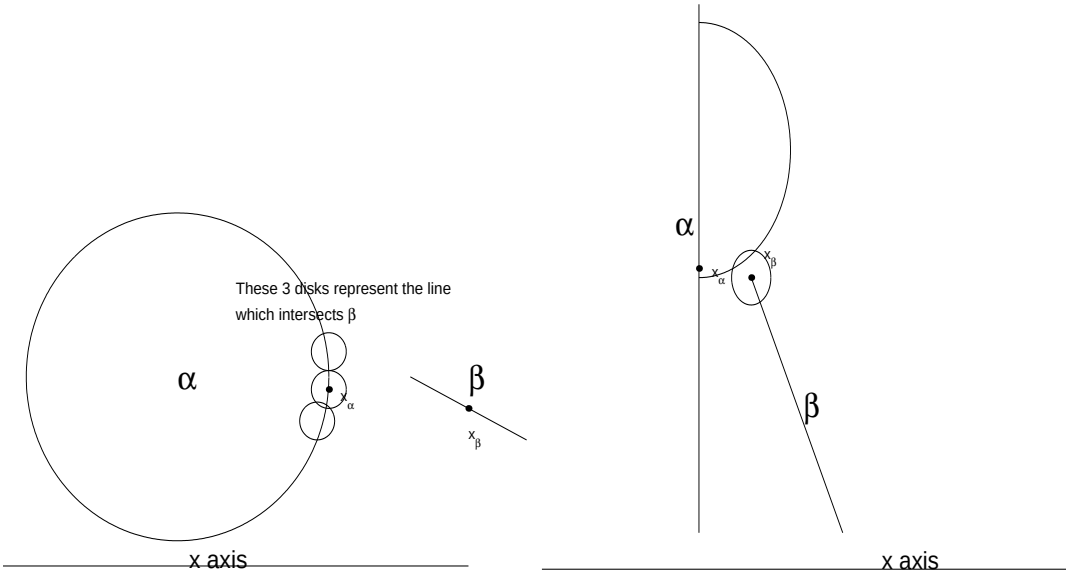

Figure A.12: Rule 26 and Rule 27

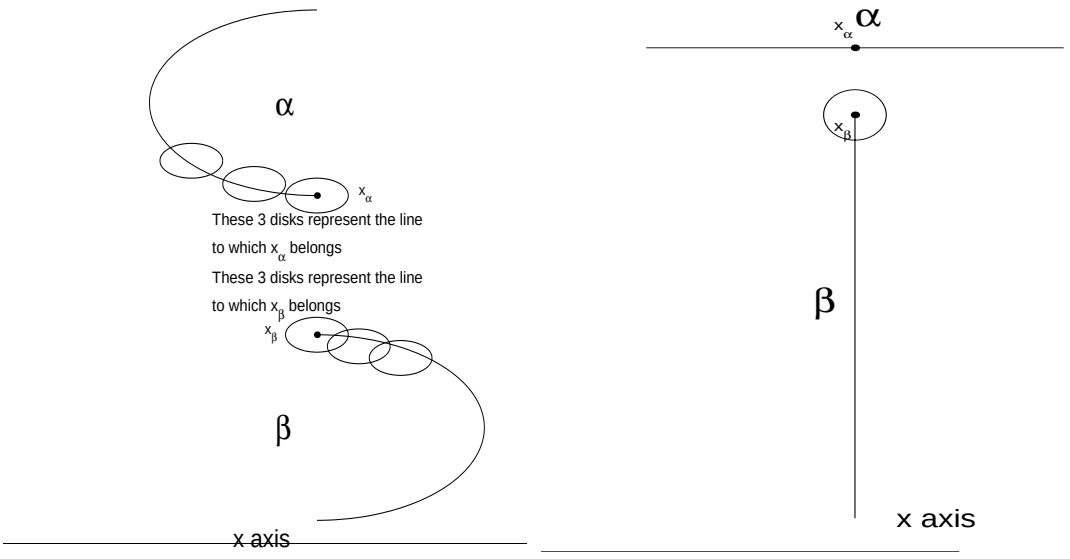

Figure A.13: Rule 28 and Rule 29

29. Letter-T = line + line (see Figure(A.13): right side)

x_α = the mid point of α, x_β = the center of the end disk of β that is near x_α.
r_α = the length of the line, r_β = the length of the line.
θ_α = the angle from α to the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.5,2.0).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N($\frac{1}{2}\pi$,0).

30. Letter-U = Letter-C + line (see Figure(A.14): left side)

x_α = the center of one end disk of α, x_β = the center of the end disk of β that is near x_α.
r_α = the length of the end line of α that is furtherest from x_α, r_β = the length of the line.
θ_α = the angle from the end line to which r_α belongs to the x-axis, θ_β = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

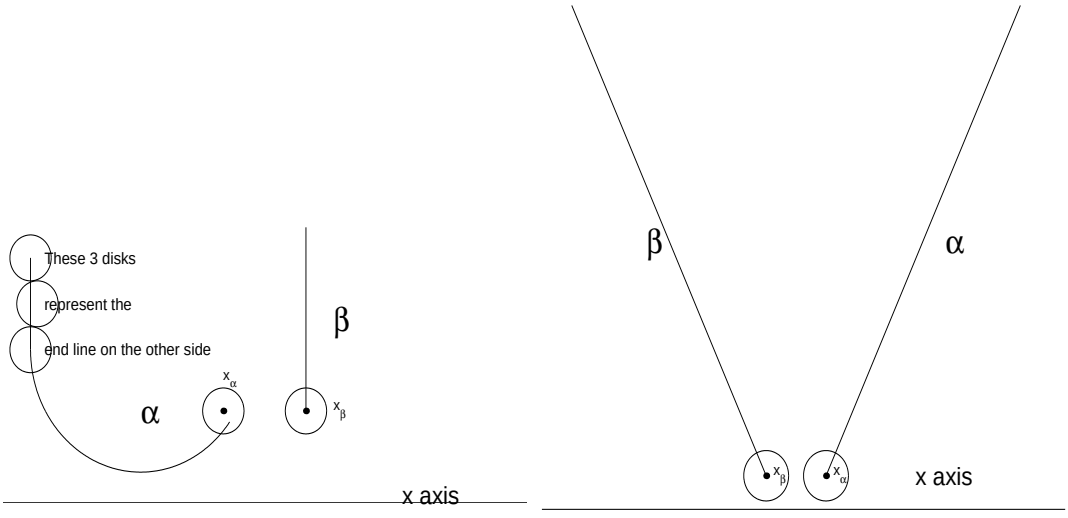

Figure A.14: Rule 30 and Rule 31

31. Letter-V = line + line (see Figure(A.14): right side)

x_α = the center of one end disk of α, x_β = the center of the end disk of β that is near x_α.

r_α = the length of the line, r_β = the length of the line.
θ_α = the angle from α to the x-axis, θ_β = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ U($\frac{1}{6}\pi$,$\frac{5}{6}\pi$)

32. Letter-W = L-junction + L-junction (see Figure(A.15): left side)

x_α = the center of one end disk of α, x_β = the center of the end disk of β that is near x_α.
r_α = the length of the line to which x_α belongs, r_β = the length of the line to which x_β belongs.
θ_α = the angle from the line in α, that does not contain x_α, to the x-axis, θ_β = the angle from the line in β, that does not contain x_β, to the x-axis.
$s \sim$ U(0.4,2.0)
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

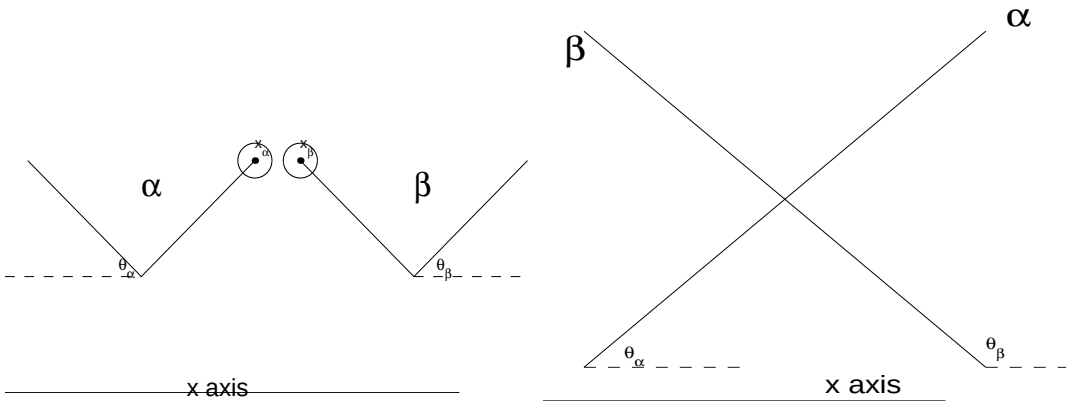

Figure A.15: Rule 32 and Rule 33

33. Letter-X = line + line (see Figure(A.15): right side)

x_α = the mid point of α, x_β = the mid point of β.
r_α = the length of the line, r_β = the length of the line.
θ_α = the angle from α to the x-axis, θ_β = the angle from β to the x-axis.
$lns \sim$ N(0,1).
$v \sim$ N((0,0),$I_{2\times 2}$).

$a \sim$ N($\frac{1}{2}\pi$,0).

34. Letter-Y = Letter-V + line (see Figure(A.16))

x_α = the intersection of the two lines in α, x_β = the center of the end disk of β that is near x_α.
r_α = the length of either line in α, r_β = the length of the line.
θ_α = the average of the two angles of the two lines in α, where each angle is measured from the x-axis, θ_β = the angle from β to the x-axis.
$s \sim$ U(0.4,2.5).
$v \sim$ N((0,0),$I_{2\times 2}$).
$a \sim$ N(0,1).

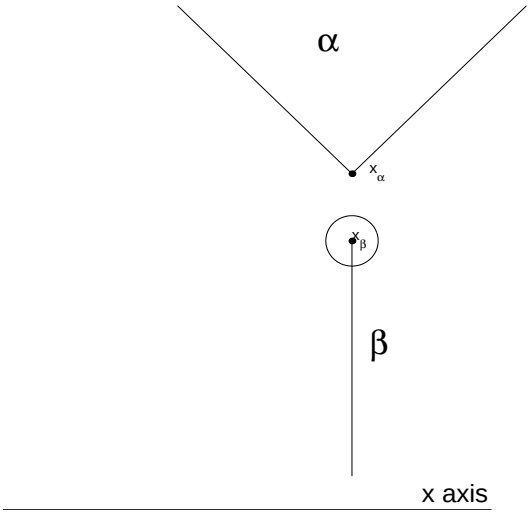

Figure A.16: Rule 34