SmartDust: Distributed Behavior-Based Mobile Agents for Thinning Segmentation and Feature Extraction of Hand-Written Arabic and Chinese Words

Ashraf Nijim

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

May 2004

Your file Votre référence
ISBN: 0-612-90981-6
Our file Notre référence
ISBN: 0-612-90981-6

ABSTRACT

SmartDust: Distributed Behavior-Based Mobile Agents
for Thinning Segmentation and Feature Extraction
of Hand-Written Arabic and Chinese Words

Ashraf Nijim

Image processing is generally acknowledged as a computationally demanding task. Distributed mobile agents is a form of parallel processing that is gaining attention as a means of increasing the efficiency of many image processing tasks. In addition, a swarm of highly autonomous agents is more reliable in satisfying a task than one sequential process; the failure of a couple of agents is unlikely to stop the rest of the swarm from achieving the over all task.

This report discusses a specific application of autonomous agents to the problem of pattern pre-processing. Specifically, we design and build a system that uses a number of behaviour-based agents to detect, segment and thin, as well as describe a hand-written patter (character or word). Each agent has a subsumption type architecture that employs a number of behaviors. These behaviors are activated when certain conditions are satisfied. Every agent has a local coordination unit that arbitrates between the various behaviors to decide which one of the behaviors is active at any one time. The system has a whole a global coordination unit that creates and spreads the various agents over the image, then acts as a conduit of (light) communication between the agents, until the overall task is done, and all last agent dies.

In this work we have introduced a number of innovations including a new method of simultaneous thinning and segmentation and a simple but effective development of the

classical Fourier descriptor method that works better for open curves. We have tested our system on Arabic characters and Chinese glyphs. The agents were able to find patterns and then work on parallel to extract and segment the centerline in a few seconds. A fixed feature vector of ten descriptors is computed for each segment. The reconstruction process showed that this vector is capable of retrieving the whole shape of the signature.

Acknowledgment

I would like to thank my supervisor, Dr. Nawwaf Kharma, for his patient and valuable guidance.

I would also like to thank Donghai Zan for helping with the coding and long discussions. And finally, special thank you for Hussein Moghnieh for helping with the GUI coding, and Dr. Yaser AbuLebdeh for helping in reviewing this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The purpose of this study is to explore a new and expanding approach to digital image processing. More specifically, we apply a distributed agent based approach to the problem of image processing. A number of behavior-based agents are spread over an image of a pattern such as a handwritten word, and work largely independently of each other to detect, segment and thin, and describe the pattern. The main advantage of this approach is increased over-all speed and efficiency of processing, especially when parallel hardware or robot swarms are available.

The motivation for using a large number of independent agents in image processing is the great number of pixels (in the millions) that represent a digital image. Any kind of mechanism that processes the image sequentially (one pixel at a time) will

be executing multiples of millions of computations per image. In many cases, such as image segmentation, the required computations need access only to local information, and hence, can function independently of each other. A number N of identical computational processes (agents) can potentially process an image in 1/N of the total time required for a sequential mechanism. This is the case for multiple agents.

The motivation for using mobile agents is straight forward. In most cases the majority of the information contained in an image is redundant. For example, most of a scanned image of a typed page is white background. An agent or swarm of agents that seek areas of relevant information (typed text) and then process it will act more efficiency than a single *blind* sequential algorithm, going over the image from start to end, this is the case for giving agents the ability to move.

In addition to efficiency, there is the issue of adaptation to local conditions. An engineer who is viewing an image from the local point of view of an agent is more likely to develop algorithms that consider and exploit local information, than some one who is developing a single algorithm for the whole image. There are many cases of new solutions in science born out of a new point of view. This encourages the development of adaptive algorithms that are suited to images of objects that appear in a high variety of states (such as images of living cells). It is a given that decisions that only require local information should be made locally, and decisions that require global information should be made globally.

One unexpected advantage of agent swarms is fault tolerance, if one of one thousand agents processing an image is dead locked or simply fails, then the over-all image processing task is unlikely to be (significantly) compromised. And, if the designer

of the distributed image processing application takes into account this failure, then it is generally easy to amend agent architecture in ways that make such failures inconsequential. This fault tolerance ability makes the over all mechanism more reliable.

In summary, the processing of digital images using a swarm of autonomous agents makes the process faster, more reliable, and affords the designers a different point of view which among other things will lead to more locally adaptive algorithms.

1.2 Objectives

In this study we adopt a thoroughly distributed agent approach to the pre-processing of handwritten Arabic and Chinese characters. Pre-processing includes the following:

- Detection of the pattern in the image map *via* edge detection.
- Segmentation of the pattern into a number of segments.
- Generation of a thinned skeleton for each of those segments.
- Description of the thinned skeletons using augmented Fourier descriptors.

The input to the process is a black and white image of any size and the output is two groups of numbers. The first group represents the relative position of the segment and the second group describes the shape of the segment.

1.3 Contributions

The main contributions that we make to the fields of agent-based image processing and pattern recognition are:

- A subsumption type architecture for image processing agents;
- A new fast behavior that simultaneously segments and thins patterns;
- Improved Fourier descriptors suitable for describing open curves.

1.4 Organization

Chapter 2 provides a review of robots and agents, their different architectures, and collaboration techniques. Chapter 3 discusses thinning, along with some of its important features and problems. It also talks about segmenting skeletons and computing Fourier descriptors for each segment. Chapter 4 presents the distributed image pre-processing agent architecture, and the various behaviors of an agent. Chapter 5 concludes the system and its results. It also provides an idea about further work that can be done to utilize the full potential of our approach.

Chapter 2

Background and Review

2.1 Robots and Agents

2.1.1 Architectures

There are four basic [1] approaches to autonomous agent control: reactive, planner-based (deliberative), hybrid and behavior-based. Reactive systems embed the agent's control strategy into programmed condition-action pairs; it is a bottom-up approach always used for achieving real-time performance. Usually it employs a simple functional mapping between stimuli and responses in the form of a look-up table. The reactive approach is usually used for well-defined predictable systems.

The deliberative approach is the traditional top-down strategy that relies on a centralized model; the information available to the system is used to produce the most

appropriate sequence of actions to achieve the goals. Any changes in the information, errors in sensing or different environments would require recoding large parts of the system.

The hybrid approach is a compromise between the deliberative and reactive approaches. It usually uses a reactive system for low-level control, relating, for example to the safety of the agent, and a deliberative approach to high-level decision making, such as selecting the sequence of actions for achieving a given goal.

The behavior based approach falls between reactive and deliberative type architectures. Behavior based systems rely on various forms of distributed computations and representations instead of a centralized processor. Behavior based systems employ non-serialized behaviors controlled by a predefined activation and selection mechanism. Most practical systems use built-in fixed-priority control mechanisms, such as the subsumption architecture [2]. In general, the constrains on behaviors in any behavior-based system mandate that the behaviors be:

a) relatively simple;

b) incrementally added to the system;

c) executed when ready (non-serially);

d) more time-extended than simple atomic actions of a particular agent;

e) interactive with other behaviors through the world rather than internally through the system.

Scaling up in agent systems has two types: scaling in the number of behaviors within a single agent, and scaling up the number of interacting agents. Going from single-agent domains to multi-agent domains expands the state space G to include the state of each agent. This global state space grows exponentially with the number of agents as $G = s^a$, where s is

the state of a single agent, and (a) is the number of agents. Many problems prevent using deliberative-based systems for multi-agent domains, such as the difficulty of tracing a system that grows as the number of agents increases; communication between agents; and uncertainty in perceiving state. Reactive and behavior-based approaches used for multi-agent domains produces non-centralized systems that scale well with the number of agents and require no global communication. Nerd Herd [3] is a good example of a multi-robot system that uses basic behaviors as the building blocks of the system. Those basic behaviors are not further reducible to each other, and can be combined to generate higher-level behaviors. The set of basic behaviors were chosen by combining constrains in two ways: bottom-up using the dynamics of the robot and the environment, and top-down by goals. The architecture of the system allows two combination operations: summation that generates higher-level behaviors and switching that inhibits all but one of the behaviors at a time.

Learning behavior selection [4] was tested using the basic behavior set discussed in the previous section with another group of robots called the Don Group. A simple reinforcement learning based on behaviors instead of actions was used for (a) choosing the best (highest reinforcement) behavior for each condition, and (b) learning the switching circuit for achieving certain goals. A special kind of reinforcement called shaped reinforcement was used as feedback during the learning phase.

Learning can also be extended to other levels of behavior-based systems. It can be used at the evolutionary scale for automatically generating basic behaviors, at the behavior level for tuning parameters and functions, and furthermore, at the behavior combination and selection level.

2.1.2 Agents

2.1.2.1 Initialization

Initialization is the process of placing agents inside the image. This process has two parts; the first part is to find out the number of agents (the initial number of population) to be placed in the image at the beginning of the program. And the second part is to place (locate the initial coordinates) of each one of those agents in the image.

Different initialization techniques have been used in different image processing tasks, those techniques could be summarized in the following methods: -

a) Uniform distribution of a variable number of agents proportional to the size of the image (one agent placed in a fixed n x n area) [5];

b) Fixed number of agents placed in random positions [6,7,8];

c) Only one agent placed at the origin of the image [9,10];

d) Prior-knowledge; where a variable number of agents proportional to the complexity of the image are placed at certain feature points;

e) Random number of agents placed randomly in the image.

Each technique has its own advantages and drawbacks. Using a fixed number of agents works well for small images, but its performance will start to deteriorate with the increase of the size of the image. Placing one agent in a fixed area after dividing the image can work well for certain tasks but also can miss feature points located in smaller size areas (smaller than the fixed area). Using only one agent could be seen as using a filter with fixed dimension. The agent applies this filter to all the points of the image to locate feature points.

This process is time consuming, works in a sequential way and can only do certain fixed tasks. Randomized number of agents and placement can give good results for certain tasks, but not for every image-processing task, especially when all the feature points in a certain image are required.

Uniform and random distribution of agents in an image does not guarantee the extraction of all feature points in a given image, because the agents are not guaranteed to visit every location in that image. To solve this problem, long life spans were given to those agents to increase their chances of locating the feature points. Also the search (movement of agents) was biased by examining the directions of all successful agents in locating feature points [6]. Even after introducing those two elaborations, random or uniform distributions with random movements are hard to predict, and can only give reliable results if agents are assigned very long life spans. By doing this, the program will run for a long time, and the multi-agent system will lose its main advantage, which is speed. On the other hand, giving the multi-agent system prior-knowledge (like certain starting points), will direct the search for feature points, but only at the beginning. The initial population size will be directly proportional to the complexity of the image, as it increases, there would be more initial points, and the number of agents initiated would also increase. The previous method introduces another problem, which is the finding of general complexity assessment technique that would be applicable to all sorts of images.

2.1.2.2 Architecture

G. Butler *et al.*, described an object-oriented design methodology based on the subsumption architecture for the Truckin' project [11]. Guidelines for building such systems were given. The guideline concerning the architecture of object-oriented subsumption like software systems are:

- Guideline#1; decompose the system into task achieving behaviors. Start from the behavior that deals with survival in a changeable environment. Each added behavior would increase the capability of the agent for doing more complex tasks.

2.1.2.2.1 Behaviors

Conceptually speaking, agent behaviors are the same as robot behaviors. The main difference between the two classes of behaviors is the goal. While robots do real world real time tasks, agents are simply programs designed to do some specific computer tasks. Four guidelines are provided describing the nature of behaviors and the tests every behavior should pass in the Truchin' project [11]:

- Guideline #2; the input to a behavior is the environment: the only input to any behavior in the system is the environment. There are no connections between the different behaviors. Therefore, the action of the behavior is coupled to its input. The behavior should be able to translate its input to actions fast enough for it to be a positive part in the system.
- Guideline #3; suppression and inhibition: Higher behaviors can suppress or inhibit lower behaviors. Suppression is the process of changing the input of a

certain behavior, whereas inhibition is the process of ignoring the output of a lower behavior.

- Guideline #4; testing environment: Each behavior should be tested in all possible environments. There should be no simple or specific test environment. Every behavior should be able to operate properly in all possible environments.
- Guideline #5; testing behaviors: Before adding any new behavior to a certain system, this system should be tested alone in all possible and dynamic environments. Also, after adding a behavior, the system should be tested again. Since a bug in a lower behavior may lead to unpredictable and magnified errors in higher behaviors.

2.2 Agent Coordination

In [12] a new message passing behavior was added to the four-robot system to introduce cooperation between the identical robots for doing specific tasks (puck delivering). The main function of the message passing behaviors is to send the status of the robot when it is executing a task and monitor the status of other robots. When a robot concludes the execution of a task successfully, it starts to negotiate the next step with the other agents by using a radio communication channel. The robot will wait for a positive signal, but if one is not received for a pre-fixed wait period, the robot will proceed to execute its intended task alone. When the robot fails to do a task, and the robot is aware of that fault, it will send a message to the other robots informing them, and allowing the other three robots to complete the task. But if the robot is not aware of that fault, then the robot will activate a restart

process, during which all the other robots will receive no messages from the re-booting robot, effectively treating it as a "dead" robot.

An auction-based task allocation system called MURDOCH is introduced in [13], the system uses a publish/subscribe communication model of cooperation between robots. A minimum communication overheard was taken under consideration when designing the system. Also the robots were assumed to be heterogeneous and capable of communication, but messages may be lost. Furthermore, the robot, or part of it, may fail at any time, and it may not even be aware of its own failure. Those constrains made the system more complicated to design and test. On the other hand, the system was positively tested on practical tasks (box pushing). The robots would be able to cooperate to achieve certain tasks if there were communication channels between them; if there were none (due to failure of the other robots, high noise in the environment, or the inactivity of all the robots but one) then an agent would still be able to achieve (or try to achieve) the task by itself, but usually taking longer times to do so.

2.3 Review of Distributed Agents-Based Image Processing

Wang and Yuan, described a color-sensitive behavior-based multi-agent system for the detection of face location within an image [5]. An initial number of agents are initially spread around the target image; this initial number depends on the size of the image and the minimum size of face (assumed constant). Then, each agent examines its location by computing the HSV color values and compares them to a predefined range of values concerning human skin-like color. If the location belongs to a face-like region, the agent will mark it, and then produce four other child agents (with the same characteristics and family

index). After reproduction, the parent agent will be inactive. If the location does not belong to a face-like region or is visited by another agent, the agent will randomly diffuse to one of the eight neighborhood pixels and its life will increase by one. If the agent life exceeds its life span, the agent will be removed. The agents program stops when the number of agents reaches zero (no agents are looking for face-like regions in the image). The agents work in a natural way, therefore more than one agent can work on locating one face-like region. So after locating all the face-like regions, a region-merging program joins the different regions of one face together. It counts the number of locations visited by two different agents (using the family index), and depending on this number, the two regions may be merged into one or left separate. After locating all the face-like regions, the system analyzes the shape of each region to make sure that it belongs to a face-like shape (which is typically elliptical). The orientation, height and aspect ratio (height/width) of every candidate region are combined to form a decision function that is used to locate different human faces in the image using a predefined threshold value.

Liu *et al.* [6,7,8] described a system that uses evolutionary autonomous agents for extracting different image features. A fixed initial number of agents are spread randomly over a gray-level image. Each agent checks its local neighborhood, which is a circular region centered at the pixel were the agent resides and with a fixed radius. It looks for other pixels with grey-level values sufficiently different from the grey-level value of the pixel where the agent is located. Sufficient difference is defined by a pre-fixed positive threshold . If the number of such pixels is within a predefined range, the agent will place a mark at the agent's pixel, and trigger the reproduction behavior of the agent. On the other hand, if the number is out of range, the agent will diffuse, to one of eight directions, for a random number of steps

within a predefined diffusion region. Each agent has a fixed life span, when the agent age (number of diffused steps) exceeds that life span, the agent will vanish from the environment. The direction of reproducing offspring agents (placing the offspring agents) and the direction of diffusion are calculated using a probability vector derived from all the successful (high fitness) agent directions of the current and previous agent generations. Thus, the agent will inherit the most effective direction from its successful predecessors. When reproduction is triggered, a predefined number of offspring agents are spread in a predefined reproduction region; the distance of each offspring from the parent agent is randomly generated.

The authors took a border-tracing agent as an example after defining the previously mentioned attributes. The effect of each behavior was examined in that example. First the direction of reproduction and diffusion was examined, after fixing the direction region to 5x5. The effect of random direction selection for each behavior was examined, then compared with the results derived from the direction vector selection method (mentioned above). The later method gave better experimental results for cases with fixed agent lifespan and a fixed number of offsprings per reproducing agent. Varying the initial population size did not affect the dynamics of evolution. On the other hand, varying agent lifespan did have a direct effect on evolution: increasing the lifespan of agents increases the chances of agents on finding feature points.

The authors also examined multiple classes of agents. Three different agents were applied to a complex image. Each class of agents was looking for a certain feature without any kind of interaction between the different types of agents. The rate of feature extraction for each class of agents varies due to the fact that agent behaviors are more effective in searching for high-connectivity features. Tracking image features for multiple frames

(moving image) was also tested by the evolutionary agents. After the agent locates a feature pixel in the first frame, it resides in that location, instead of simply marking it. In the next frame, some or all of the feature points will move. At this time, the agent is activated again, and will start looking for feature pixels by diffusion, reproduction and directional adjustment. At the end of the experiment, the authors observed that inheriting the direction of diffusion and reproduction hinders the ability of the agent to adapt to new local areas. Increasing the lifespan of agents will partially solve this problem by allowing agents more time to learn and adjust. By introducing a measure of randomness to diffusion and reproduction, the agent was able to explore different areas instead of being totally dependent on previous experience.

The authors of [8] added two more criteria to the process of locating feature points. These are the regional mean and regional standard deviation of the gray-level intensity of local pixels. One or more of the various methods (described above) may be used to identify feature points. The fitness function used to evaluate the movement of the agent measures the number of steps the agent takes to find a feature pixel. This function returns a values between 1 (most fit) and -1(least fit).

Angelotti *at al.* [9] described a system for the treatment of bank checks called Multicheck system. The system consists of three types of agents, a segmentation agent, a recognition agent, and an analysis agent. The segmentation agent works on the check image, to extract the different logical fields of the check like the numerical value, literal value, date, and signature. After extracting those logical fields, the segmentation agent applies a local segmentation algorithm to each field; the results are sent to 4 specialized agents. Each one of these four agents (signature, date, numerical, and literal) has three parts: interpretation, recognition, and communication part. The segmentation agent has more than one local

segmentation algorithm for each logical field, but can only give the results of one algorithm at a time to the recognition agents. If the recognition values found by the recognition agent do not achieve a certain degree of certainty then the agent will return a request to re-segment the field (using another segmentation algorithm) to the segmentation agent. The recognition agent of each field classifies the segments taken from the segmentation agent; each part recognized by the classifier has two values: favorable evidence, and opposing evidence. The former is a percentage that represents how positive the classifier is about the result, while the latter is another percentage that represents how negative the classifier is towards the same result. The degree of certainty of the result is simply the percentage difference between the two previous values. The analysis agent has the responsibility of accepting or rejecting a check, by examining all the results taken from the other agents (with their certainty degrees). The interaction between the numerical agent and the literal agent was taken as an example to show how the two agents communicate and negotiate their individual results to find the best classifier that would increase the degree of certainty of both agents.

The pattern algorithm inside each agent investigates the results of the classifier, if the certainty degree of any one of its local fields did not pass a certain predefined threshold, the agent can reject the check, or request a re-segmentation of the logical field using another algorithm. If the agent has more than one classification result, the different values are compared. If the same local field found by both classifiers, bearing in mind that the agent can combine only two results at a time, the highest favorable value would be chosen, if the classifiers gave different results, the recognition agent would have to communicate with the other agent using the communication channel to choose between those results but giving the literal value a higher degree of decision. After all the agents finish the recognition phase, the

results are sent to the analysis agent to decide whether to accept the check or reject it. The agents use a special communication language to send and receive results and other decision making data. A special communication channel is used between the numerical and the literal agents to combine the two results.

Alhajj *et al.* [10] described a multi agent system for identifying touching in handwritten Hindi numerals. Two agents are presented; each of them extracts the possible touching points in the image using its own method. When the two agents finish extracting those touching points, they start negotiating and resolving the conflicts, to identify the actual touching points. The first agent is a thickness-based agent that utilizes the width of handwriting in the raw input image. The thickness of the line is assumed to be homogenous except in the touching points between two numbers. The agent looks for sudden changes (increases or decreases) and reports them as possible touching points. The other agent is a segmentation agent. Its duty is to resolve the touching numbers and then recognize (classify) each one. After thinning and normalizing, the agent segments the image into four categories, vertical, horizontal-top, horizontal-middle, and horizontal-bottom segments. The segment points (the points were a segment starts or finishes) are either an end point (has only one adjacent pixel) or a peak point (a point where the direction of the contour starts to change in either the x or y-direction for a certain number of pixels). After segmentation, the agent starts to detect the touching points, examining segments one-by-one, and trying to fit each one with its neighbors, to build the most appropriate number. If there are two possibilities corresponding to two neighbor segments, an angle related rule is used to resolve this conflict, by choosing the appropriate segment with angle. The remaining possible-touching points are negotiated and resolved between the two agents. The results of the thickness agent are taken

by the segmentation agent to resolve any unsolved conflicts, or as advice on choosing the appropriate segmentation.

Chapter 3

Pattern Recognition and Handwritten Characters

3.1 Thinning

Thinning is the process of reducing a pattern into a one-pixel width skeleton, that (in some fashion) retains the character of the original pattern. The ideal skeleton approximates the centerline of a pattern. Practical thinning algorithms react differently to different kinds of noise, and they produce artifacts with different degrees.

3.1.1 Thinning Artifacts and Noise

3.1.1.1 Salt and Pepper Noise

Salt and peeper noise is scattered black pixels inside the image white areas and scattered white pixels inside the black areas.

3.1.1.2 White Areas inside the Pattern

White areas inside the pattern black area is also called white islands. It consists of more than one white pixel beside each other inside an area of black pixels. This kind of noise will produce two segments for a pattern instead of one, or will produce a circle between two segments.

3.1.1.3 Elongation

When two lines converge to a point with a small angle, they merge at a certain point along the two lines [14]. Therefore the two lines became one. Moreover, the smaller the convergence angle between the two lines the bigger this area of convergence is. This merged area produces an elongation artifact by adding non-existent segments. Moreover, this artifact may separate important points from each other.

3.1.1.4 Bifurcation

The same causes that lead to elongation artifacts also produce bifurcation in cases where two lines cross each other [14]. The bifurcation could be considered as two elongation artifacts combined in one area. This artifact introduces a new artificial segment inside the

intersection point. Therefore the same intersection point is broken into two; those two points are connected *via* the artificial segment. The smaller the convergence angle between the two lines and the thicker those lines, the longer the artifact is.

3.1.1.5 Touching Segments

Touching segments is a common problem with handwritten patterns. As the writer tends not to remove his/her hand while writing, more touching lines would occur. Two or more lines could touch to form a thick line. This kind of problem results in the elimination of important information, by getting the skeletonization procedure to produce one segment (instead of two) for both touching lines.

3.2 Basic Concepts

Many basic points have been extracted from pattern skeletons. Common points include: intersection points, end points and curvature points. These points are relatively easy to extract from a skeleton, therefore they have been used as the basis for many segmentation processes (see section 3.3). Extracting these points from the original pattern is not as easy as extracting them form the skeleton of the pattern. The whole area should be examined and the effect of noise should also be considered.

3.2.1 Intersection Points

The intersection point is any point along the skeleton that has more than three neighbor pixels. Usually intersection areas are subject to a lot of noise. This noise is either a result of the thinning process or is caused by the complexity of the intersection branches.

The number of black-to-white transitions is used for testing the pattern areas for possible intersection points. An expandable circle centered at the tested point with an initial radius of one is used for the transition count. The radius of the circle is increased by one, iteratively, until a fixed number of transitions is reached. At any point, if the number of transition from black-to-white (BWT) around a circle centered at that point with radius (r) is greater than 2, it will be considered an intersection point.

$$BWT > 2,\ r = 1,2,.....,R_T \tag{3.1}$$

where R_T is the maximum radius allowed for the test circle.

3.2.2 End Points

An end point is any point in a pattern skeleton that has only one neighbor pixel. On the other hand, for a point inside the original pattern to be identified as one of the end points, a bigger view of the end area should be tested. The same test used in the previous section (section 3.2.1), is used for testing for end points, but with a different threshold.

An expandable circle is used to count the number of BWT. At any test point, if the number of BWT around a circle centered at that point with radius (r) is equal to 1, then this point will be considered an end point.

$$BWT = 1,\ r = 1,2,.....,R_T \tag{3.2}$$

where R_T is the maximum radius allowed for the test circle.

3.2.3 Curvature

The curvature of a continuous curve is defined as the change in angle of the tangent of a point p in a small segment of the curve,

$$\text{Curvature} = \lim_{\Delta s \to 0} \frac{\Delta \alpha}{\Delta s} \tag{3.3}$$

where, s is the distance to the point p.

Finding the curvature of a curve in discrete digital images is difficult, because of the discrete nature of the representation.

Different methods were used to find the curvature [15,16]. By using the pixel as the smallest unit in Equation 3.3, we let $\Delta s = 1$, the curvature of a point i along the curve will simply be the change of the angle $\Delta\alpha_i$,

$$\text{Curvature}_i = \Delta\alpha_i \tag{3.4}$$

To find the above curvature, two vectors ($\mathbf{V}_{ic}$ and $\mathbf{V}_{ia}$), one in a clockwise direction to the reference point i, and the other in the anticlockwise direction (representing the two sides of the curve reference to point i) were computed. The vectors ($\mathbf{V}_{ic}$ and $\mathbf{V}_{ia}$) are the summation of n vectors. Each vector is computed form point i to the next pixel up to n pixels away from point i.

$$V_{ic} = \sum_{k=1}^{n} v_i(k,c) \tag{3.5}$$

$$V_{ia} = \sum_{k=1}^{n} v_i(k,a) \tag{3.6}$$

Where,

$$v_{i(k,c)} = (x_i - x_{k,c})i + (y_i - y_{k,c})j \tag{3.7}$$

$$v_{i(k,a)} = (x_i - x_{k,a})i + (y_i - y_{k,a})j \tag{3.8}$$

The angle $\Delta\alpha_i$ can be computed from the dot product of the two vectors:

$$\Delta\alpha_i = \cos^{-1} \frac{V_{ic} \bullet V_{ia}}{| V_{ic} || V_{ia} |} \tag{3.9}$$

For a fixed number of neighbor pixels *m*, we can define the curvature point of a discrete continues curve using a threshold T as follows:

$$\text{if } \Delta\alpha_i \leq T, \quad 0 \leq T < 180 \tag{3.10}$$

then, the point *i* is an inflection point.

In practice, if we consider no digitization or quantization noise, threshold T would only take values between 0 for a straight line, and 90 for a right angle curvature point.

3.2.4 Summary and Discussion

Simple but effective algorithms have been introduced for extracting important feature points. Those algorithms are applicable to skeletons of patterns. The simplicity also implies speed, which is always a positive characteristic.

3.3 Segmentation of Handwritten Character Skeleton

3.3.1 Review

Lin and Tang [17] described a system for offline stroke extraction from handwritten Chinese characters. The system works in two stages; the first stage is to extract the feature points from the character's skeleton, after removing the thinning artifacts around the fork points. The second is to extract the strokes using the feature points found in the first stage. For extracting the feature points, a "crossing number" is used to find the end points, connected points, and fork points (intersection points) from the skeleton. The thinning algorithm used does not produce a perfect 1-pixel wide skeleton, especially when there is a fork point. Those artifacts were grouped into two main problem categories. The skeleton was rescanned to identify and remove those problems. This process continues until all the problem points are removed. After locating all the feature points (end points and fork points) in the skeleton, the stroke extraction stage starts by tracing every stroke between two feature points. Strokes passing through fork points are now broken due to the artifacts produced by the thinning algorithm. To determine which two strokes need to be connected (or not), a bi-directional graph method was proposed. The directions of the various stroke segments around the fork point are computed (after ignoring the points adjacent to the fork point), then each of

these strokes tries to connect itself to another stroke that has an opposite direction. The two segments with the closest angle differential to 180 are chosen for connection; but only if that differential greater than 135°. Otherwise the segments are considered as individual strokes, and are hence left separate.

For certain 3-fork points, the algorithm always fails to produce the appropriate connection, mainly due to the small angle between the two possible strokes. The segment that has the smaller angle with the orientation of those segments was used as the input for a set of heuristic rules derived from the general convention of Chinese writing. Those rules are used to decide which two segments are to be connected or not.

3.3.2 Introduction

Segmentation is the process of dividing a certain shape into useful, well-defined segments. We can use those objects to identify the shape and/or classify it. It can also act as a pre-process for advanced image processing techniques.

There are two types of segments, open-segments and closed-segments. Lines and arcs are two types of open-segments. Triangles, ellipses and polygons are considered as closed-segments.

3.3.3 Definition of Segments

For the purpose of our study, a number of definitions were asserted and hence used to segment patterns. Also, results produced were judged relative to these definitions. Given that handwritten pattern is composed of one or more strokes, we present these definitions below:

Definition 1 (Intersection Area): An Intersection Area is the area of the smallest circle covering the touching of two strokes.

Definition 2 (End Area): An End Area is the area of the smallest circle covering the beginning or the end of a stroke, if it does not belong to an Intersection area).

Definition 3 (Segment): A Segment is a number of continuous pixels between two Intersection Areas and/or End Areas.

Here, it should be noted that the segmentation process can suffer from two kinds of problems, over-segmentation and under-segmentation. Both problems are defined below:

Definition 4 (Over-segmentation): Over-segmentation occurs when a part of the pattern which is actually forms one Segment is wrongly divided into two or more Segments.

Definition 5 (Under-segmentation): Under-segmentation occurs when two or more actual Segments are wrongly considered as a single segment.

(Table 1) shows the relation between over-segmentation, under-segmentation and normal segmentation, in terms of Segments.

Table 1.Types Of Segmentation

Type	Component (part)	Segment
Normal segmentation	1	1
Over-segmentation	Many	1
Under-segmentation	1	Many

3.4 Feature Extraction

3.4.1 Review

The use of Fourier Descriptors (FD) for pattern recognition tasks started in the early seventies. FDs were used as features for recognition systems for both handwritten characters [18] and numerals [19]. Granlund [18] used a small number of lower-order descriptors for his classification system. Those descriptors were insensitive to translation, rotation and dilation. Because of the small computational power available at that time, the system could not be examined to give the suitable number of descriptors. The classification system was applied to a small number of characters. Nevertheless the system was able to produce a very good recognition rate of 98%.

Zhan and Roskies [19] started computing the FDs by translating the contour of handwritten numeral into a change of angle curve. The process of translation starts by choosing a point along the handwritten numeral contour (start point). The angle of the start point with respect to the horizontal axis is computed. The direction of travel along the contour is selected, by choosing either a clockwise direction or an anti-clockwise direction.

Starting from the point beside the start point and in the direction of travel, the angle with respect to the horizontal axis is computed iteratively, until reaching the point beside the start point from the other direction of travel. The difference between each angle and the start point angle is the angle-difference signature of the given handwritten numeral. Fourier transformation is then applied to the signature (see section 3.4.2.4). A large number of Fourier coefficients are produced. The lower order coefficients capture the general shape of the signature, whereas the higher order ones reflect finer details. Fourier descriptors, in their totality, are a normalized version of the Fourier coefficients. For each Fourier coefficient two kinds of FDs are computed, the harmonic amplitude and the phase angle. Those pair of FDs is invariant under translation, rotation and change of size of the original handwritten numeral. The complete set of FD pairs fully describes the original signature. On the other hand, for the purpose of discriminating between different handwritten numerals, only the first ten FD pairs are usually used.

A problem appeared in the start and end points of the reconstructed shape from its FDs. The reconstructed shape is not a closed curve. And, the start and end point of the new curve are not the same. To solve this problem, very strict conditions are imposed in the process of computing the FDs to insure that the reconstructed shape would be a closed curve [19]. A practical algorithm was proposed [20] for adjusting the offset between the starting and end points of a reconstructed contour shape for the purpose of constructing 3D Fourier descriptors from stereo images.

Fourier descriptors were also used to describe open curves in an online character recognition system [21]. The one pixel thick strokes were taken *via* a tablet. Then twenty FDs were computed and used for classification.

Different kinds of FDs were evaluated along with different shape signatures [22] to fix the number of FDs for classification purposes. Different simple and complex shapes were tested, and only ten FDs were found to be sufficient for representing the different shapes using specific high-convergence shape signatures. The best converging shape signature was found to be the centroidal distance. Centroidal distance signature is the distance of contour points from the center of mass (CoM) of the contour. First the CoM of the contour is computed, and then a point along the contour is chosen to be a start point. Using one of the two traveling directions mentioned before, the distance of each point to the CoM is computed until the end point is reached (the point beside the start point from the other chosen direction). This distance is called the centroidal distance signature of the contour.

3.4.2 Introduction

There is a trade-off between accuracy of representation using a fixed number of FDs and the computational resources required to compute these descriptors. There is a balance to be set between necessary accuracy and acceptable performance.

The general shape of an open-curve is easily reconstructed from a small number of lower-order FDs. Moreover, most of the high frequency components and noise are eliminated also. But on the other hand, using a small number of FDs some times introduces a problem at the start and the end areas of the open curve after reconstruction. The points belonging to the two previously mentioned areas are far away from the original ones unlike the other points along the rest of the curve. One of the reasons for this problem is due to the special way that the Fourier Transform (FT) deals with sudden changes (pulse functions). Since the points constituting the curve are discreet, and at both the start and end points of the curve could be

considered as a pulse function. The reconstruction problem increases (the offset between the original and the reconstructed curves) by decreasing the number of FDs used.

3.4.3 Shape Signature

Shape signature is a 1-D function that represents a 2-D area [22,23]. Applying the Fourier transform on a 1-D shape signature derives the FDs. Several shape signatures have been used in the area of FDs; following are the most common signatures.

3.4.3.1 Position Function

The position function is the complex coordinates of the relative coordinates to the centroid of the shape. First the centroid is computed by averaging the two coordinates of the shape:

$$x_c = \frac{1}{N}\sum_{t=0}^{N-1} x(t) \qquad y_c = \frac{1}{N}\sum_{t=0}^{N-1} y(t) \tag{3.11}$$

where N is the number of points along the shape, and (x_c, y_c) is the centroid of the shape.

Afterwards, the position function is computed using the following formula:

$$z(t) = [x(t) - x_c] + j[y(t) - y_c] \tag{3.12}$$

3.4.3.2 Centroidal Distance

The centoridal distance is the distance of the points along the shape from the cenrtoid of the shape found by equation 3.11. The formula for computing this function is:

$$r(t) = \sqrt{[x(t) - x_c]^2 + [y(t) - y_c]^2} \tag{3.13}$$

3.4.3.3 Chord Length Signature

The cord length is the distance between a point along the curve and another boundary point such that the tangent at the current point is perpendicular to the line connecting the two points.

3.4.3.4 Cumulative Angular Function

The cumulative angular function is the tangent angle difference between every point along the curve and the first angle. The formula for computing it taking under consideration the continuity of the signature is:

$$\varphi(t) = [\theta(t) - \theta(0)] \bmod (2\pi) \tag{3.14}$$

where, $\theta(t)$ is the tangent at point t.

3.4.3.5 Curvature Signature

Curvature signature is the curvature values at very curve point. As mentioned in section 3.2.3, there are many ways of computing the curvature of a discrete curve. Despite

the problems involved in computing curvature at certain points, the curvature signature has been used in many applications.

3.4.3.6 Area Function

The area between two consecutive points along the curve and the center of mass of the shape (centroid) is called the area function. The formula for computing this function is:

$$A(t) = \frac{1}{2}\left|x_1(t)y_2(t) - x_2(t)y_1(t)\right| \tag{3.15}$$

3.4.4 Fourier Series

The discreet Fourier function for a periodic polynomial function $f(t)$ is,

$$F(k) = \frac{1}{N}\sum_{t=0}^{N-1} f(t)\exp(-j2\pi kt/N) \tag{3.16}$$

where N is the total number of points along $f(t)$.

Equation 3.16 could be rewritten in another form,

$$F(k) = \frac{1}{N}\sum_{t=0}^{N-1} f(t)\{\cos(2\pi kt/N) + j\sin(2\pi kt/N)\}, \quad k = 0, 1, 2, \ldots, N\text{-}1 \tag{3.17}$$

The Fourier coefficients of the above two equations are,

$$a_k = \frac{2}{N}\sum_{t=1}^{N-1} f(t)\cos\left(\frac{2\pi kt}{N}\right)$$

$$b_k = \frac{2}{N}\sum_{t=1}^{N-1} f(t)\sin\left(\frac{2\pi kt}{N}\right) \qquad (3.18)$$

3.4.5 Fourier Descriptors

One of the commonly used FDs are the harmonic amplitude A_k and phase angle θ_k of the Fourier coefficients a_k and b_k (see equation 3.18),

$$A_k = \sqrt{a_k^2 + b_k^2}$$

$$\theta_k = \tan^{-1}\frac{b_k}{a_k} \qquad (3.19)$$

The harmonic amplitude A_k is a pure shape feature and does not contain information about the position or the orientation of the shape. But, on the other hand the phase angle θ_k has those two features. Furthermore, those two descriptors are insensitive to shifting, resizing and small shape changes in the overall shape.

The fixed length feature vector would be:

$$\{A_k, \theta_k\}_{k=1}^{M}, \qquad (3.20)$$

where M is a fixed integer number.

3.4.6 Shape Reconstruction

The original polynomial could be reconstructed from its FDs by using the following equation:

$$f(t) \approx A_O + \sum_{k=1}^{N-1} a_k \cos\frac{2\pi kt}{N} + b_k \sin\frac{2\pi kt}{N}, \quad t = 0, 1, 2, \ldots, N\text{-}1 \tag{3.21}$$

where, A_o is the DC component of the function, and has no effect on the shape description.

a_k and b_k are the Fourier coefficients of the shape, and could be derived from (Equation 3.19).

Chapter 4

Distributed Image Pre-processing Agents

4.1 Introduction

A collection or swarm of mobile pre-processing agents will generally have the following advantages over typical image processing algorithms:

a) The agent will be able to seek areas of relevant information within the image; this allows a number of agents, much smaller in number than the total number of pixels in the image, to process the image. This reduces computational time considerably.

b) Agents will sense and act locally; they will read a small amount of local information of the area around them, and they will take actions that only affect that area. Hence, they are not only simple, but also more locally adaptive.

c) The agents being largely independent of each other can be allocated to separate computing nodes, and as such will minimize the overall computational time required to process a complete image.

4.2 Behaviors

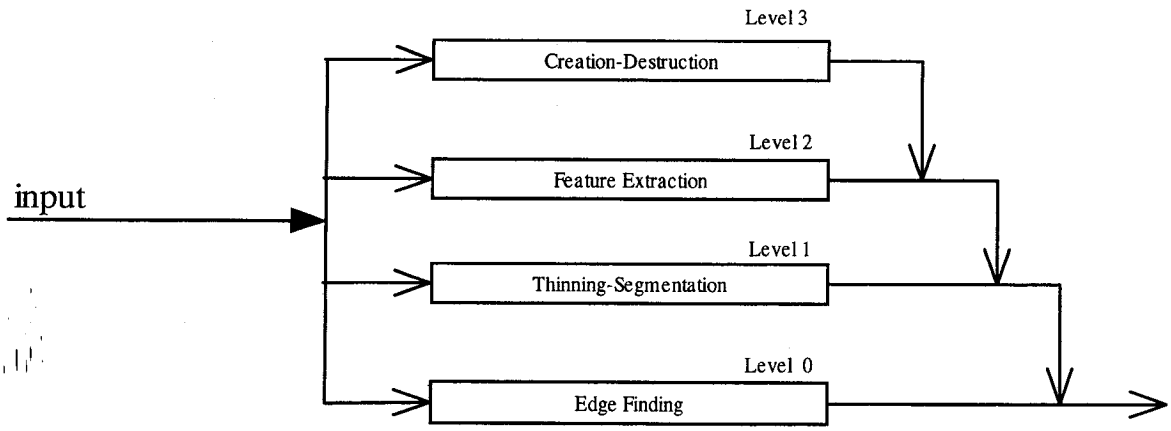


Figure 1. Agent Behaviors

An agent is made of two components; a set of behaviors and local a coordination unit. Every behavior has access to local input of the agent, and decides when the behavior is going to be active or not. Zero or more behaviors may be active at any moment in time (Figure 1). Higher behaviors subsume lower ones.

Below is a detail description of the behaviors, the order of presentation of behaviors mirror the likely order of their application during the lifetime of an agent.

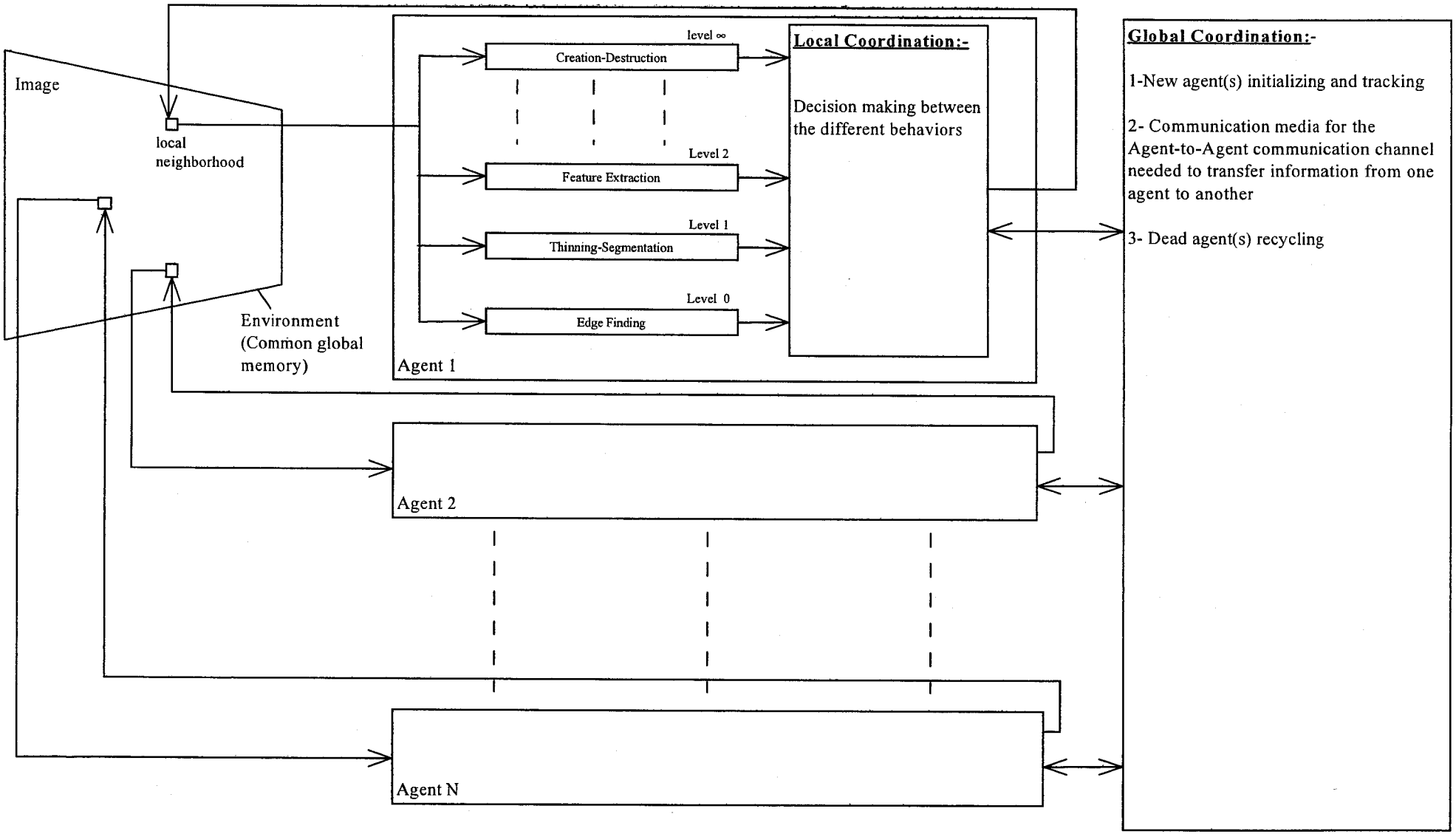


Figure 2. Multi-Agent Architecture For Image Pre-processing Tasks

4.2.1 Creation-Destruction Behavior

This behavior has the highest level among the behaviors inside an agent (Figure 1 and 2). This behavior takes care of any initialization needed when placing the agent for the first time. It is also responsible for sending a message to the global coordination unit (*via* the local coordination unit) to inform it of the death of the agent.

4.2.2 Edge Finding Behavior

The edge finding behavior is the lowest level behavior. The objective of this behavior is to get the agent to travel to reach an edge of a pattern and mark it. The image is a black and white image. When an agent detects a change of color from black to white or *vice versa*, it will label the black pixels as edge points (see section 4.7). After marking the edge point, the agent's task is achieved and it will stay in the same place waiting for other behaviors to be activated. Once this is done, other behaviors are likely to get active, and one or more of them will eventually express itself.

4.2.3 Thinning-Segmentation Behavior

The thinning-segmentation behavior has two image processing tasks. The first task is to find the centerline of any pattern inside the image. And the second task is to segment this centerline into simple segments. The two processes operate simultaneously. The output of this behavior is a number of one-pixel-wide segments that represent the skeleton of the pattern. Two end and/or intersection points bind each segment.

4.2.4 Feature Extraction Behavior

This behavior accepts the output of the thinning-segmentation behavior, which is a sequence of pixels representing a segment. It then generates (a) a point representing the center of mass (CoM) of that segment, and (b) a fixed number of Fourier descriptors. The coordinates of the CoM of the segment is relative to the coordinate of the center of mass of the whole pattern. The Fourier descriptors for each segment are computed using a local system of coordinates with the center of mass of that segment as its origin. Those numbers describe the segment and can later be used for classification or reconstruction. Furthermore, the Fourier descriptors comprise of a fixed number of harmonic amplitude and phase angle pairs (see section 3.4.4).

4.3 Local Coordination

Local coordination is the conflict-resolution mechanism of an agent. The different behaviors inside the agent are connected to local coordination unit. The local coordination unit determines which of the active behaviors will be expressed. This unit embodies a static regime of priorities: if one behavior is active, then this behavior expresses itself; if more than one behavior is active, then the behavior with a higher priority (level) is allowed to express itself. Behaviors work independently of each other. However, any/all access to the image (environment) goes through the local coordination unit.

4.4 Global Coordination Unit

This unit is the place where any/or communication and coordination between varies agents takes place. The main role of this unit is the creation, initialization and deletion of

agents. It also keeps a record of the number of agents on the image at any given time. Communication between agents should be kept to a minimum. After creation and initialization, an agent will communicate with the global coordination unit for the following reasons: (a) reproduction of an agent, (b) the removing of dead agents (see section 4.5), and (c) the merging of the results of two agents.

4.5 Creation-Destruction Behavior

The creation-destruction behavior is responsible for initializing agents and removing them from the environment (image). The process of initializing an agent consists of many steps. First an agent is placed at a specific location. Then the agent is assigned a unique identification number (index).

Agent removal is the reverse of the agent initialization.

4.5.1 Initial Agent Population

The first event that takes place right after starting the program is the birth of the initial agent population. The purpose is to spread a number of agents in a given image so that each one of those agents can perform its task in an efficient way. First, the image is subdivided into equal squares. This is followed by the placement of one agent in each square. The size of the square in a given image depends on the size and complexity of that image. It was empirically determined that a windows size of 10 x 10 works best for our purposes.

4.5.2 Agent splitting

After placing the initial agent population, each agent will start looking for edges (see section 4.6). When it finds a new edge and after selecting a center point inside the segment (see section 4.7), the agent splits into two agents. The new agent is given the same index as the older agent. Agents with identical indices are set to have the same "family". One agent will move inside the segment following the segment centerline, and the other will move in the opposite direction. The process of splitting the agent into two in the thinning-segmentation behavior increases the efficiency of the thinning-segmentation behavior.

4.5.3 Agent Reproduction

One agent family (one or more agents with the same index) works on only one segment of the pattern. When an agent reaches an intersection point it marks it as one end of the segment. To enhance overall efficiency, if one of the nearby branches of the current intersection point (where the agent is currently located) has not been segmented (meaning that no other agent has worked on the branch), the agent will produce another agent at the beginning of that un-segmented branch. The offspring agent will retain all the parameters from its parent agent, but will have a new index number. The offspring agent is treated as a different agent with no operational relation to its parent.

4.5.4 Agent Death

Agents are like workers performing a specific job. When a worker finishes its task, it starts looking for other work. If it does not have a new job, then it will eventually die, and be removed from the environment. There are two main conditions that would activate the agent

removing process. Two conditions cause the removal of an agent from the environment (a) if an agent moves, looking for a pattern, for a number of steps greater than its maximum allowed life (predefined constant of the simulation), and (b) when an agent does not change its location for three consecutive cycles. This condition for removing an agent can only happen when an agent finishes working on a specific segment, and does not find other segments to work on.

4.5.5 Death Algorithm

a) Edge finding behavior: if the current active behavior is the edge finding behavior, then,

   i) Count the number of movements (unsuccessful steps)

   ii) Compare the number of unsuccessful steps with Life (the maximum number of unsuccessful steps), if both are equal then go to (c), otherwise go to (a).

b) Thinning-segmentation behavior: if the current active behavior is the thinning-segmentation behavior, then if the agent did not change its current location for three consecutive cycles, then go to (c), other wise go to (b).

c) Delete agent: Send a message to the global coordination unit through the local coordination unit, requesting the removal of the current agent identified by its index number. The message has two requests:

   i) The index of the agent is removed

   ii) The number of active agents is decreased by one.

4.5.6 Summary and Discussion

The creation-destruction behavior takes care of the initialization of new agents and the removal of elderly and jobless agents (see section 4.5.4). The birth of additional agents in areas of interest in the image increases the efficiency of the over all system. Also, the removal of agents that were not successful in finding segments after a certain number of steps prevents the system from allocating too many resources to unproductive agents.

4.6 Edge Finding Behavior

4.6.1 Description

An edge is where a change of pixels (from white to black or from black to white) occurs in a continuous area of pixels. After placing the agents on an image, each agent will start looking for edges.

Edge finding behavior has two processes. The first process is the detection of pixels in an area for possible edges, and the second is moving to another area if the first process was unsuccessful. Each agent examines its new location for possible edges. The surrounding area inside a circle of radius one centered at the agent's current location is examined. The count of black pixels including the agent's current pixel is evaluated. If the Black Pixels Count (BPC) was between two predefined numbers, the agent's current location is marked as an Edge Point (EP),

$$\mathrm{BPC} = \sum_{i=0}^{8} x_i \tag{4.1}$$

where, i is the index of the pixels in a 3x3 window,

$i = 0$ represents the center of that window,

$x_i = 1$ if the pixel is black, and 0 if the pixel is white

Afterwards,

$$3 \leq \mathrm{BPC} \leq 7 \Rightarrow \mathrm{EP} \tag{4.2}$$

This edge point could be a white pixel or a black pixel. After locating an edge point the, agent will move to a nearby black pixel if it is not currently on one. The agent selects one of four directions as a default direction of movement (Figure 3). This default direction of movement is not changed during the lifetime of an agent. The agent moves one step in the default direction from its current location if it did not find an edge. If the agent reaches one of the four boundaries of the image, it will select one of two directions (either a clockwise or anti-clockwise) 45° apart from the default direction (Figure 4), and will move in that direction for a single pixel, crossing into the other side. After crossing, the agent reverts to its default direction, and continues to look for edges in that direction. The image is treated as a tauras with the top boundary connected to the bottom one, and the left boundary connected to the right one.

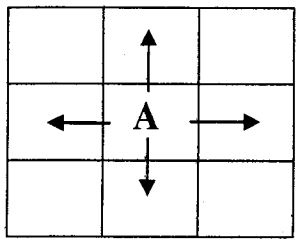


Figure 3. Agent's Four Possible Movement Directions

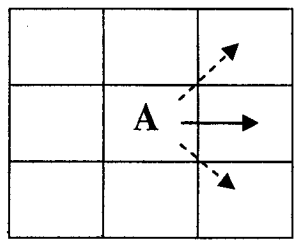


Figure 4. Agent's Default Direction With The Two Neighbor Directions (45°)

4.6.2 Algorithm

a) Examine the local area for possible edges:

   i) Count the number of black pixels in a window of 3x3 centered at the current location,

   ii) If the number of pixels is less than the pre set threshold, then go to (c), else go to (b).

b) Move:

   i) Randomly choose any movement direction (agent default direction) from the four main directions (north, east, west and south)

   ii) Move one step: if the agent current location does not belong to a pixel in the boundary of the image, then move one step (1 pixel) in the default direction from this location

   iii) Boundary displacement: if the agent is located in the boundary of the image, then move one step (1 pixel) 45° apart from the default direction (if the default direction is north, then randomly choose either north west or north east), and keep this direction as the default displacement for next possible boundary displacement

iv) Go to (a).

c) Edge pixel:

i) White pixel: if the current pixel is a white pixel, then move to any nearby black pixel

ii) Black pixel: mark this pixel (location) as an edge pixel.

4.6.3 Reliability of the Algorithm

The edge finding behavior depends on three main parameters, the number of agents working in a specific area, the agent's Life (maximum number of steps), and the pattern edge limits (see section 4.6.1).

4.6.3.1 Discovery of Small Pattern Component

The agent can locate any pixel in a given image provided that a sufficient number of agents are spread over that image and each has a suitable Life.

The worst case for small pattern components is an image that has only one pixel, and one agent was placed in that image. Both the pixel and agent locations are selected randomly. Two conditions should be met for this specific scenario; the agent's Life should be greater than or equal to the total number of pixels of the image, and the threshold for the edge detection behavior should be set to allow the detection of one pixel edge,

$$\begin{aligned} \text{Life} &\geq \text{image dimension(hight} * \text{width)} \\ \text{BPC} &\geq 1 \end{aligned} \tag{4.3}$$

After examining its current location, the agent will move using its default direction (see section 4.6.1). At every new location, the agent will look for that pixel. If the agent reaches one of the four boundaries of the image without finding the pixel, it will move to the

other boundary opposite to it. The agent will start moving in the series of locations parallel to, and adjacent to, the agents previously visited locations (Figure 5). The agent movement guarantees that it will visit all the locations between the agent's initial location and the pixel edge. Furthermore, if there were no edges in the whole image, the agent will continue moving and will return to its initial location.

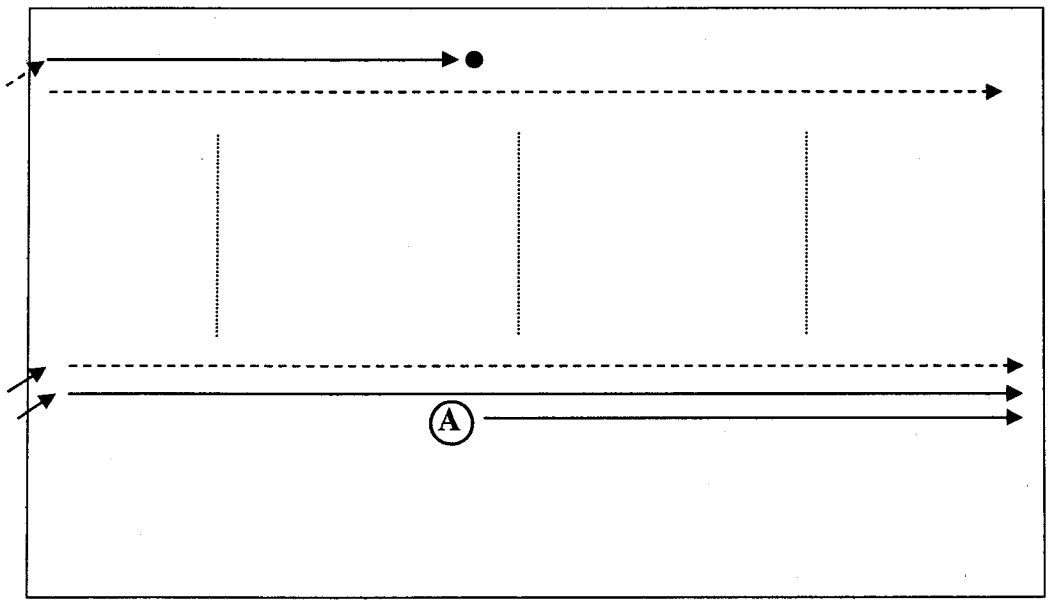


Figure 5. The Worst Case: One Agent And One Pixel, The Agent (A) Moves Around The Image Until Locating The Pixel (The Black Dot)

4.6.3.2 Noise

The edge detection threshold insures that the agent will locate real edges. The agent will ignore noise consisting of one or two adjacent pixels. Pepper noise is one kind of noise that will not affect the agent's edge detection behavior.

4.6.4 Limitations

As stated in the previous section, two of the main parameters that affect the results of the edge detection behavior are the agent's Life and the number of initial agents placed in an image. The number of agents needed for detecting all the segments in a given image depends on many parameters. The complexity of the image is one of those parameters. As the complexity increases (in terms of number of segments) more agents are needed to detect all the possible segments. The contour length of a given segment is another parameter. The number of agents should increase in inverse proportion to the average length of segment contour. Small segments completely surrounded by other segments are an important problem. When facing such a problem, the agent placement becomes an important issue. At least one agent should be placed inside that area to increase the probability of locating the small enclosed segment. But still, this does not ensure that the agent placed inside the closed area will find the small segment. As a matter of fact, the agent will probably find the other large segments as the contour of those segments are bigger than the enclosed one.

An agent's Life has an affect on its edge detection performance. Therefore, the agent will be able to find any black pixel if it is allowed to live forever. If Life = ∞ the system would never terminate execution, even after all the segments have been found and described (using Fourier descriptors). But this makes the system impractical. However, the agent's Life should be long enough for an agent to locate any segment at an image if placed in any location inside that image.

4.6.5 Results

Two handwritten databases (Arabic words [24], Chinese official numbers [25]) were used to test the edge finding behavior. The image is first divided into equal squares of 10 x 10. Then one agent is placed inside each one of those squares. After spreading the agents over the image, both the time and CPU clock were recorded. As soon as one (or more) agent(s) locate any and all the segments in the image, time and CPU cycle recording is terminated. This time in seconds and CPU cycles are viewed as indicators of the efficiency of the edge detection algorithm. The average time in seconds was 1.6 seconds and the CPU cycle average was 113. The results indicate how fast the algorithm was when it applied to our databases. (Figure 6) shows a sample of the edge finding behavior results from each database.

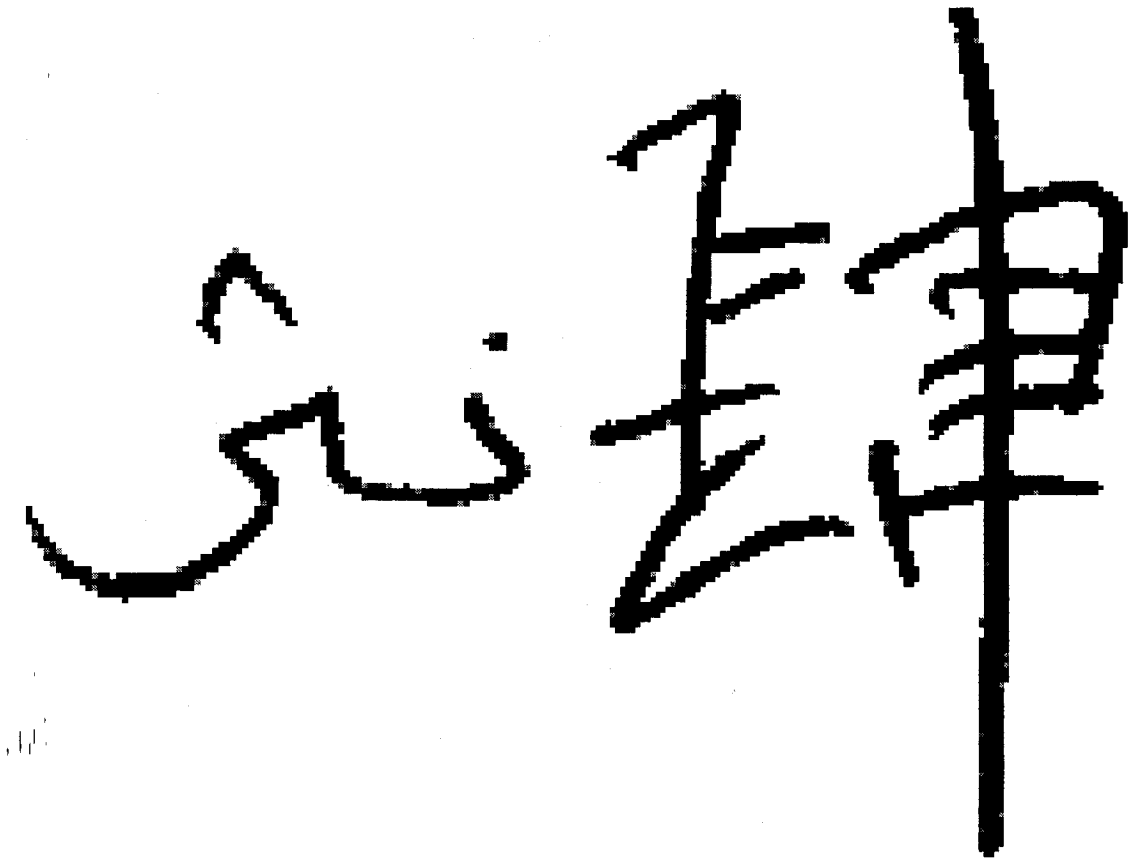


Figure 6. Edge Finding Behavior Results, Gray Dots Represents The Agent's Current Location

4.6.6 Summary and Discussion

The edge detection behavior gave good results in locating all the segments using two different data sets. Moreover, the detection process was fast. There are many reasons for these results. The agent's sight is limited to a window of 3x3 pixels. Each agent is placed at a different location inside the image. All of those reasons along with the edge detection algorithm mentioned before made the location of edges a very easy task to the agents. Most of the time ten steps or less were enough for most of the agents to locate a segment. But on the other hand, there would be one agent or more working on empty areas trying to locate

any possible edges. The only termination for those agents would be either finding an edge that has been found before, or exceeding the maximum number of steps allowed for an agent to move while looking for edges. The time and CPU cycle were measures of speed for the algorithm when applied to the two handwritten numbers/words databases.

4.7 Thinning-Segmentation Behavior

The thinning-segmentation behavior has two tasks. The first is to find a skeleton for a given pattern, whereas the second task is to segment that skeleton. The agent uses a circular window to locate the center point of a given pattern. The agent is located at the center of the circular window. The size of this window is made to change so as to ensure that at least one of the edges of the segment is tangential to the circumference of the pattern. Segmentation and thinning operate simultaneously. When an agent locates a center point that is either located in an end area of a pattern or an intersection area (see section 3.3), the point is added to the centerline as an end of that segment.

4.7.1 Description

The thinning part of the behavior has two functions, while the segmentation part has only one. Finding the first center point and locating the next best center point are the two functions of the thinning part. Whereas counting the black-to-white transitions is the function of the segmentation part. All of this results in locating a centerline of a segment of a pattern.

4.7.2 Joining Segments

One or more agents can have the information of a pattern segment. Most of the times a family of two agents work on the same segment, each moving in a different direction from the edge point. Therefore, the information of the two agents should be merged.

When one of the two agents finishes adding points to a certain pattern segment, it sends a request-to-join message to the global coordination unit (see sections 4.4 and 4.5). The global coordination unit will look at its agent's index list for other agents with the same indices. The results of the search are sent back to the inquiring agent. If the agent receives a positive massage (another agent with the same index is found) it will start transferring its information to the other agent through the global coordination unit. However if the agent receives a negative massage it means that all the information of a segment is inside this one agent. The transfer of information from one agent to another is achieved through the global coordination unit after passing through the agent's local coordination area. When all the information is passed to one agent, the other agent will delete the duplicate information inside it.

4.7.3 Algorithm

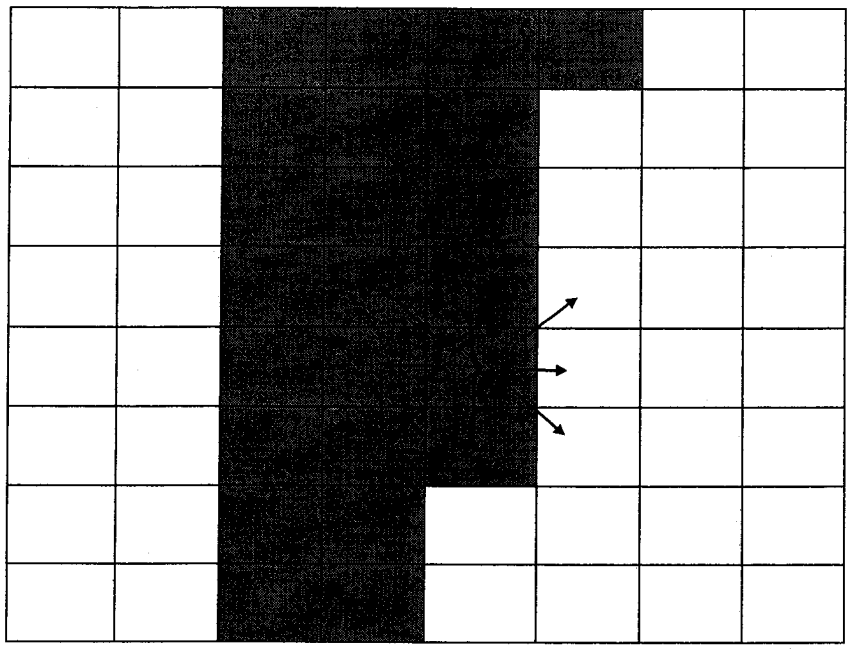
4.7.3.1 Finding the First Center Point

a) Black pixels count: count the number of black pixels in the eight neighbor directions.

b) Center point direction: find the direction of the second highest black pixel count found at step (a) that has an opposite direction of no black pixels.

c) Locate the center point: find the middle point of the direction found at step (b) by dividing the number of black pixels by two.

d) Move the agent: move the location of the agent from the edge point to the first center point found at step (c), see (Figure 7 and 8).



(a)

2	*10*	0
2	A	0
2	1	0

(b)

Figure 7. The First Center Point; (a) The Agent's Eight Directions; (b) Black Pixels Count Result (Italic Number Indicates Unseen Black Pixels)

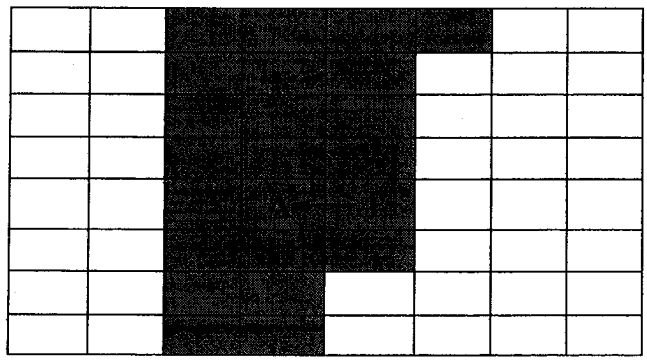


Figure 8. Agent Moving To The First Center Point

4.7.3.2 Splitting

a) Send a message to the global coordination unit *via* the local coordination unit to initiate an agent at the same location as the requested agent.

b) Send a message to the global coordination unit to increase the number of agents, and to add the index of the new agent to the index list.

4.7.3.3 Black-To-White Transitions

a) Find a suitable test circle: fix a circle at the agent's current position,

   i) Starting with a radius of one, count the white pixels inside the circle

   ii) If the white pixels count is equal to zero, then increase the radius by one and count again

   iii) Continue increasing the size of the circle at (ii) until the count is greater than or equal to one.

b) Black-to-white transitions: traveling in a clockwise direction,

   i) Count the black pixel to white pixel transition at a distance equal to the radius of the circle found at step (a) centered at the agent's current position

   ii) Increase the radius by one, and repeat the count

   iii) Keep increasing the radius of the circle until the two previous values at (i) and (ii) are equal

   iv) This value is considered as the black-to-white transition, see (Figure 9).

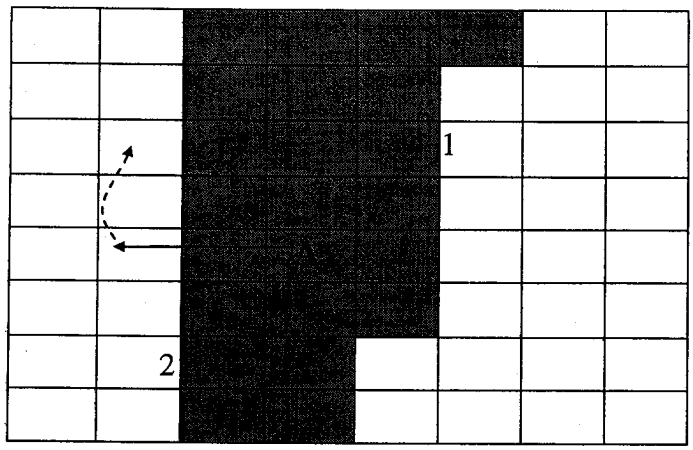


Figure 9. BWT Count

4.7.3.4 The Next Best Center Point

c) Find a suitable test circle: fix a circle at the location beside the agent's current position using its previous direction:

   i) Starting with a radius of one, count the white pixels inside the circle

   ii) If the white pixels count is equal to zero, then increase the radius by one and count again

   iii) Fix the size of the circle to the one found in the previous step with the first white pixel count of greater than or equal to one.

d) Test the other two directions:

   i) Using the same circle found at step (a), count the number of white pixels inside the circle in the direction +45° apart from the direction used in step (a)

   ii) Count the number of white pixels inside the circle in a direction -45° from the one used in step (a).

e) Select the next best center point: select the smallest white pixel count among the three values found in the previous two steps. Give the direction at step (a) more weight when two directions happened to have the same white pixels count, see (Figure 10).

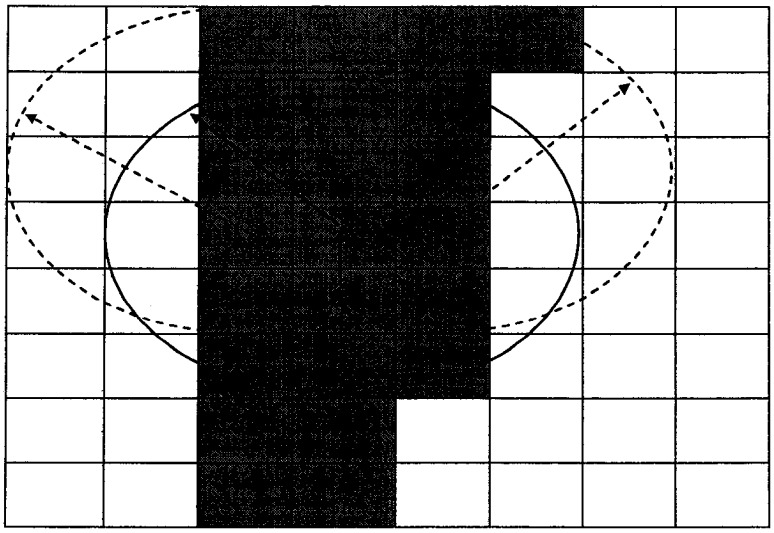


Figure 10. White Pixels Count Inside The Circle Centered At The Three Possible Next Locations

4.7.4 Agent Reproduction

The intersection area of a pattern has three or more branches as mentioned before. Therefore, when an agent reaches an intersection area by traveling inside a pattern segment, only its own branch is already segmented. However, other branches may or may not have been segmented by other agents. As the current agent has only local access, it would start producing other agents. The production process starts by scanning the other branches to find the number of unsegmented branches (see section 4.5.3). The agent will produce a number of agents equal to the number of un-segmented branches. The new agents do not retain any parameter values from their parent agent. However, each of those agents would be given an initial direction by the parent agent.

4.7.5 Results

4.7.5.1 Segmentation Results

For each of the two databases, the Chinese official numbers and the Arabic words databases, the number of perfectly segmented patterns, and the number of imperfect ones are counted. The imperfect segmented patterns include the under-segmented and over-segmented patterns. The results for 250 Chinese official numbers are shown in (Table 2), and the results of another 250 Arabic words are shown in (Table3). The "both under and over-segmentation" results represents the number of patterns that had the two problems at the same time. (Figure 11) and (Figure 12) shows a plot of the percentage of the results from (Table 2) and (Table 3) respectively. The 250 sample patterns from the two databases were chosen to cover all the different classes and writers in each database. The results of segmentation are presented in Appendix A.

Table 2. Segmentation Results: Chinese Official Handwritten Numbers

Perfectly segmented patterns	**Imperfect segmented patterns**		
	Under-segmentation	**Over-segmentation**	**Both under and over-segmentation**
233 (93.2 %)	14 (5.6%)	3 (1.2 %)	1 (0.4 %)
	17 (6.8 %)		

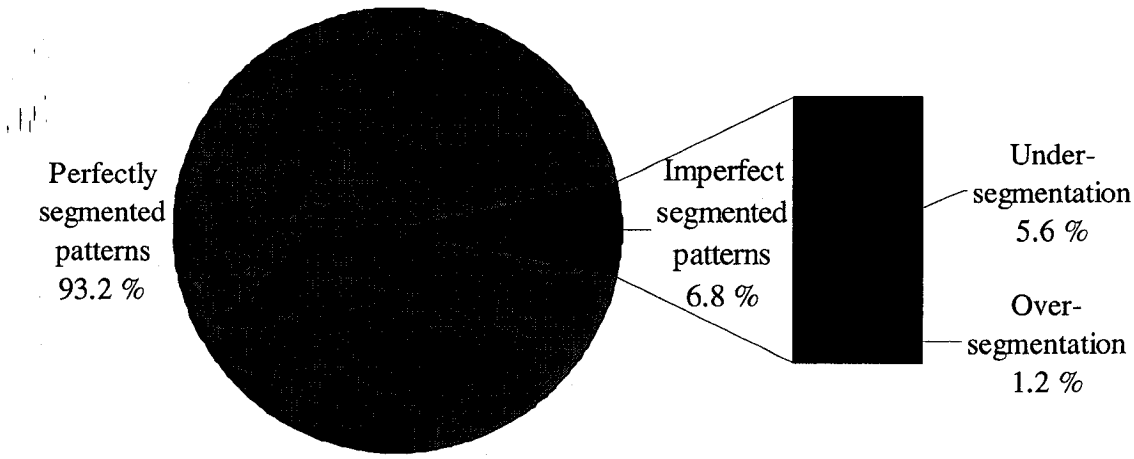


Figure 11. Chinese Handwritten Official Numbers Segmentation Results

Table 3. Segmentation Results: Arabic Handwritten Words

Perfectly segmented patterns	**Imperfect segmented patterns**		
	Under-segmentation	**Over-segmentation**	**Both under and over-segmentation**
234 (93.6 %)	12 (4.8 %)	4 (1.6 %)	1 (0.4 %)
	16 (6.4 %)		

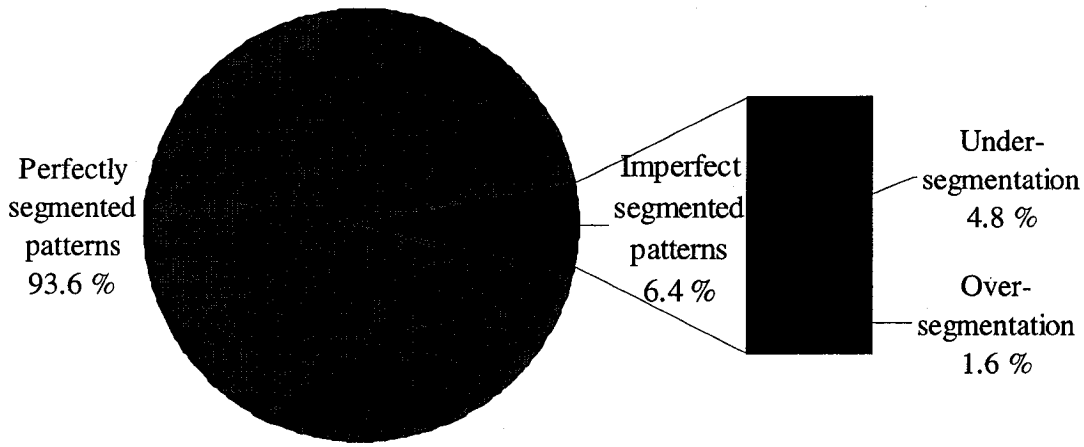


Figure 12. Arabic Words Segmentation Results

4.7.5.2 Thinning-Segmentation Results

A sample from the two databases, the Chinese official numbers and the Arabic words databases, were used to test the performance of the thinning-segmentation behavior. A sample of those results are presented in (Figure16) and (Figure 17) for the Chinese numbers database and the Arabic words database respectively. The segments and centerline are shown in those two figures. The intersection and end areas are labeled. The centerline between any two labeled areas represents one segment skeleton.

The performance of the thinning algorithm is measured relative to a number of well known thinning problems and artifacts (see section 3.1.1). The number of occurrence of those thinning artifacts and problems are reported in (Table 4). The results from each database are reported in a different column. The percentage of thinning artifacts/problems for each database are plotted in (Figure 13) for the Chinese number database, and in (Figure 14) for the Arabic words database. The test sample patterns from the two databases were chosen to cover all the different classes and writers in each database. The complete results of thinning-segmentation behavior are presented in Appendix A.

Table 4. Thinning Artifacts And Problems

Thinning problem	**Average number of problem occurrence**	
	Chinese official handwritten numbers	**Arabic words**
Salt and pepper noise	0 %	0 %
White areas inside the pattern	12 %	1 %
Elongation	5 %	11 %
Bifurcation	14 %	9 %
Touching segments	0 %	0 %
Total average	6.2 %	4.2 %
Total STD	0.0657	0.0535

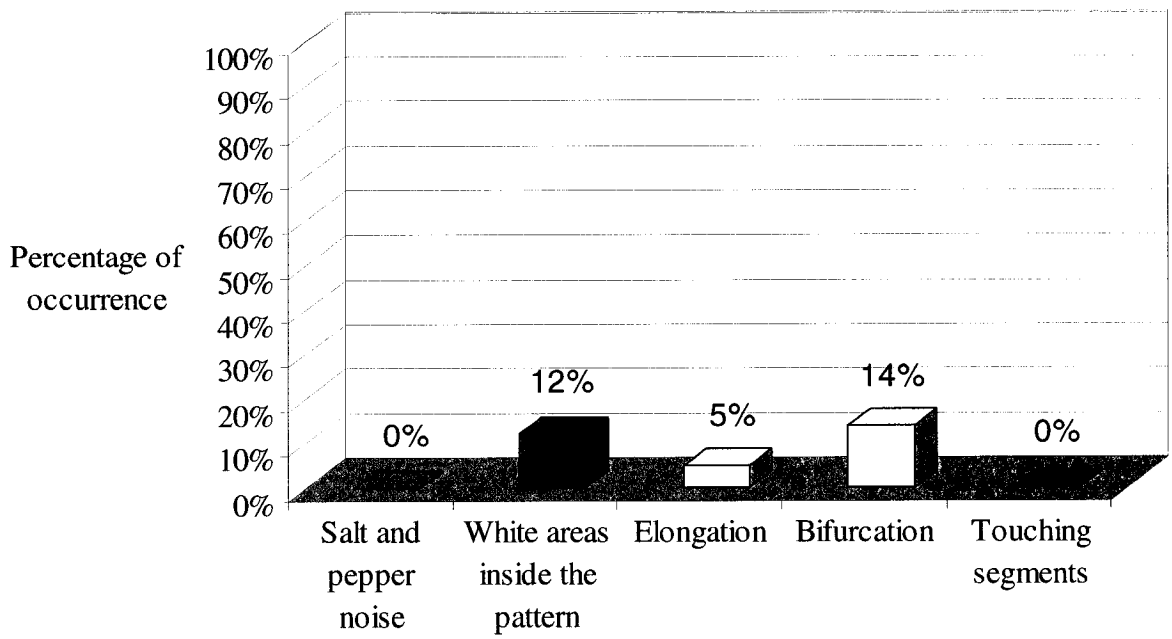


Figure 13. Chinese Handwritten Official Numbers Thinning Results

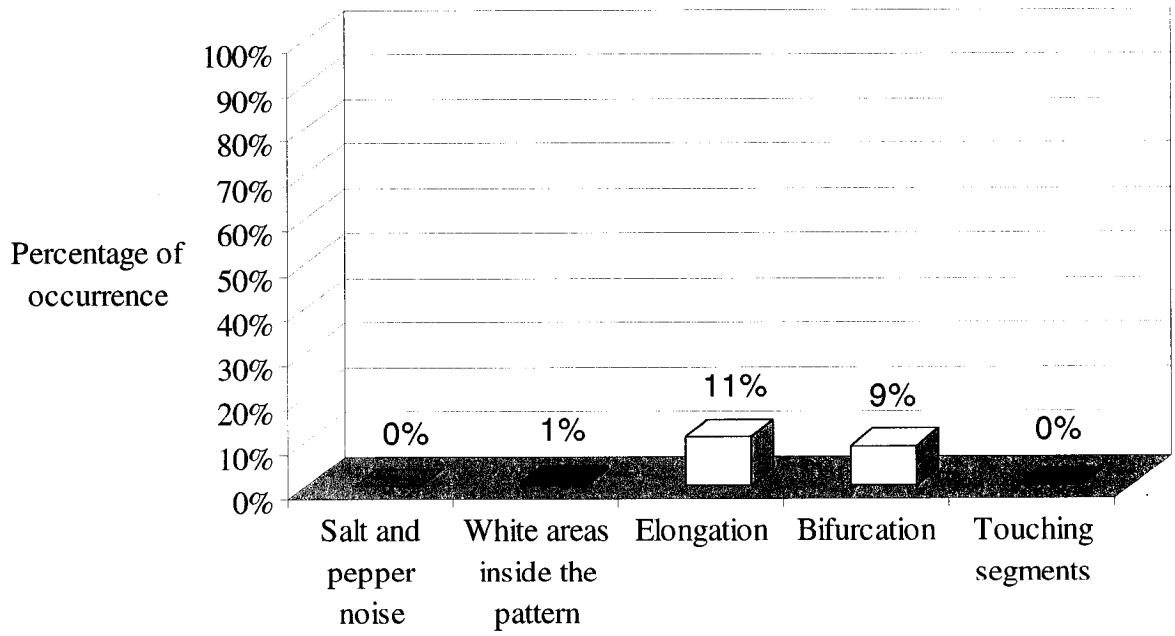


Figure 14. Arabic Words Thinning Results

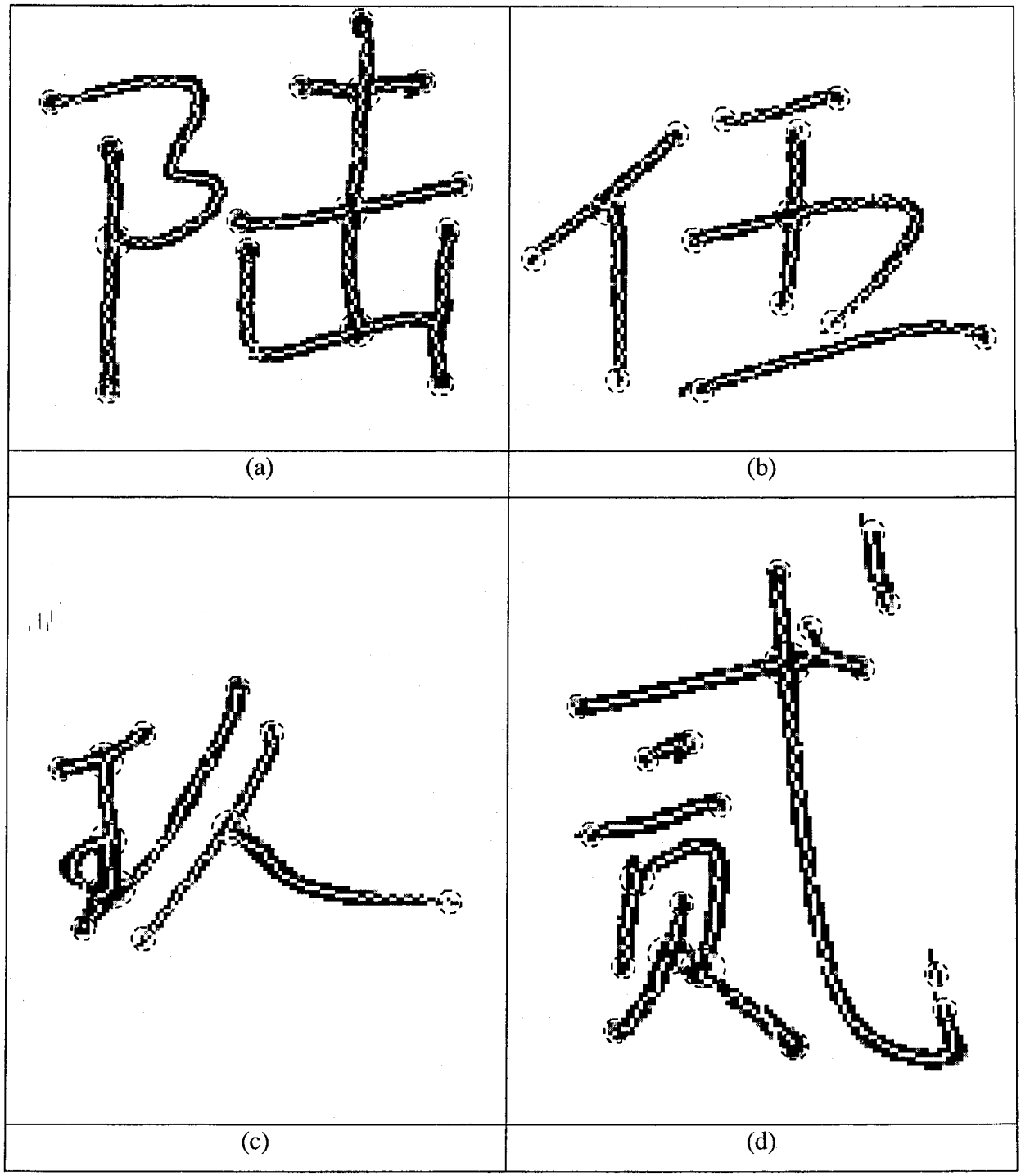


Figure 15. Chinese Handwritten Official Numbers Thinning-Segmentation Results

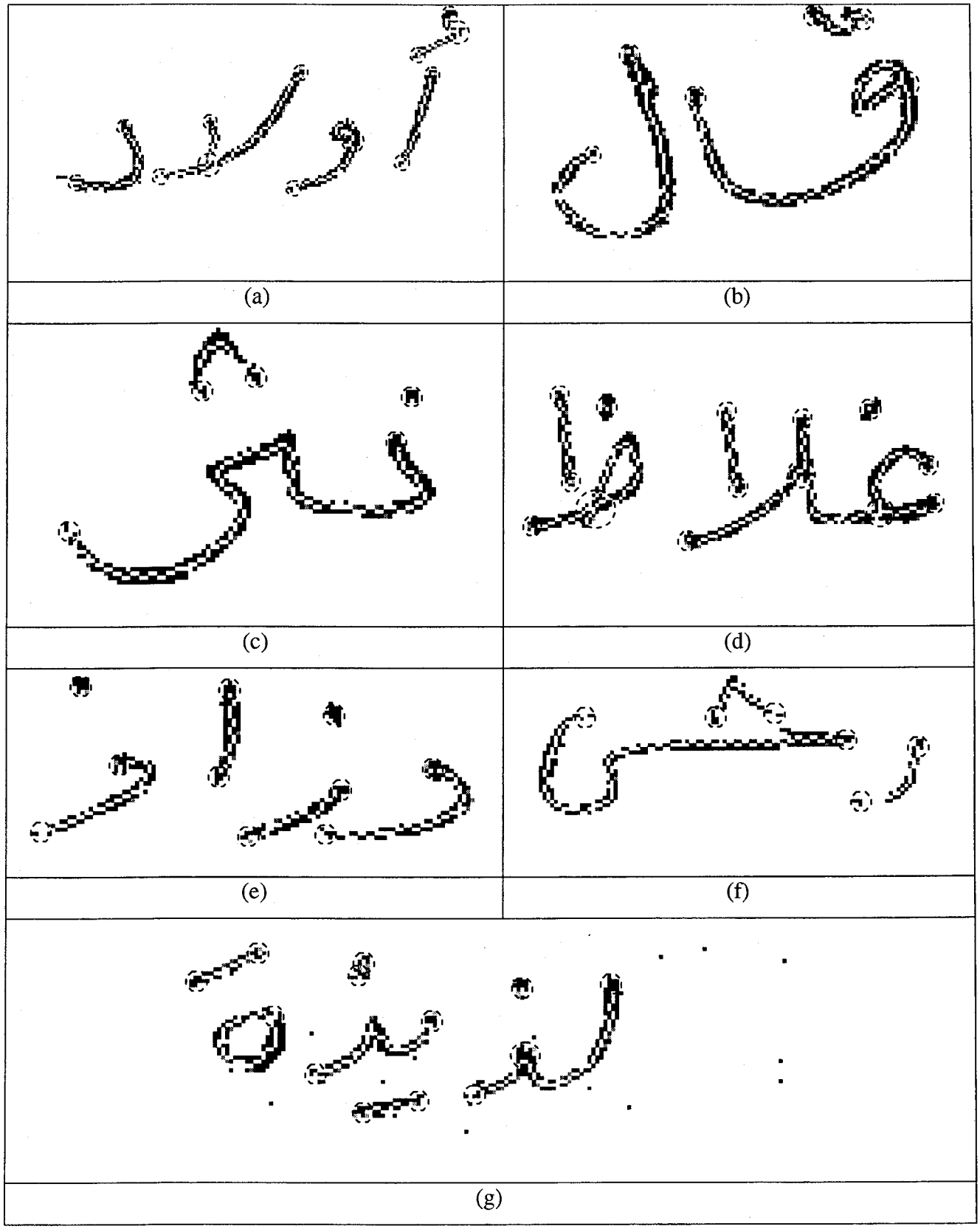


Figure 16. Arabic Words Thinning-Segmentation Results

4.7.6 Summary and Discussion

4.7.6.1 Segmentation Results

The results of the segmentation showed that the adapted strategy is consistent for the two databases in question. The perfectly segmented patterns were 93.2% and 93.6% for the Chinese official numbers and Arabic words, respectively. Most of the imperfectly segmented patterns happened to be under-segmented patterns. The most common problem among those under-segmented patters was the noise around intersection areas. This noise led to an early detection of intersection point, especially with three-branch intersection areas. On the other hand, the under-segmented patterns are not that important to our system as the over segmented patterns. If the resulted segments from under-segmented patterns kept a relatively simple shape, the system would still be able to reconstruct those segments. The over segmented patterns are the main segmentation problem. Noise is the main cause of this problem. While the percentage of the over segmented patterns is low (around 1 % for the two databases), still this problem can cause unpredictable errors for other databases having high amount of noise.

4.7.6.2 Thinning Results

The thinning-segmentation behavior carries out two pattern-recognition processes concurrently. The thinning part uses a resizable rotating circle to find the centerline of a given pattern segment. The agent is biased to move in one direction. After locating the first center point and the first candidate direction, the agent will focus its search on the area laying in the direction of movement. The agent's splitting and reproduction processes speed up the system by allowing more than a single agent to work on the segment in question. As the

agent has access only to the local area, the agent examines only the part of the image that has information. Therefore the thinning and segmentation processes will operate faster than the traditional thinning/segmentation algorithms.

The thinning-segmentation was able to reduce some of the well known thinning artifacts/problems (see Table 4, Figure 13, and Figure 14). The elongation and bifurcation artifacts were greatly reduced without adding any post-processing algorithms. Noise segments produced by white areas inside a given pattern were also reduced. The agent tends to ignore small areas of change inside a given pattern including small noise segments. As a consequence, only one segment skeleton is produced for a given pattern segment that has a noise white area inside it. The two behaviors, the thinning-segmentation behavior and the edge finding behavior, managed to eliminate the salt-and-pepper noise. The thing-segmentation behavior is insensitive to white pixels inside a given pattern.

4.8 Feature Extraction Behavior

4.8.1 Description

Once a segment has been isolated, an agent initiates the process of describing it using our augmented FDs. When the agent does not change its location (does not move) for two consecutive cycles, this process starts. The first step is to test the information inside the agent. If the agent does not have any information the behavior will release the control imposed on it. On the other hand, if it has information about a segment, the agent will go to the next step, which is the feature extraction. Certain threshold is used to prevent the

behavior from processing tiny segments. Those segments could be noise or some other background artifact however; it could be a small but very important feature like a dot.

The centriodal-distance signature and the angle difference signature [19,22] were used to transfer each segment into a polynomial that is necessary for computing the Fourier coefficients.

4.8.2 Fourier Descriptors for Open Curves (FDOC)

Consider an open-curve consisting of N pixels. Using one of the shape signatures [22], convert the curve into a polynomial curve. Every pixel along the shape is converted to a point along the polynomial curve. The next step is to extend this polynomial by ΔN points starting from the two end points. Each end point is extended by (ΔN / 2). The new extended polynomial curve length is equal to (N+ΔN). (ΔN / 2) should be a positive even number. The process of feature extraction goes through a number of steps. First extending the original curve, then computing the FDs described in the previous section. After that, the process of reconstructing the original shape starts by computing the inverse Fourier transform (Equation 3.21) using a small number of lower order FDs. Afterwards, the area associated with the extended parts of the original curve is removed leaving a curve that has the same number of points as the original curve and is very close in shape to it.

The extension of the original curve involves three steps, first shifting the curve along the horizontal axis, then computing the average slope of each end area, and finally adding a fixed number of points to each end area starting from the end points. Those steps are explained in detail below:

The original polynomial curve *f(t)* is shifted along the horizontal *t*-axis, (ΔN / 2) steps,

$$f'(t+\Delta N/2) = f(t), \qquad t = 0, 1, 2, \ldots, N\text{-}1. \tag{4.4}$$

where, f' is the extended-polynomial curve.

The average slope of the end segment is equal to the average of slopes of the various lines connecting the pairs of consecutive points making up the end segment. Since the number of points between any two consecutive points is equal to 1, then the difference in the vertical direction between two points is equal to the slope of the line connection these two points.

$$\Delta S_1 = \frac{1}{\Delta N} \sum_{t=t_1}^{t_2} \Delta f'(t) \tag{4.5}$$

where $t_1 = \frac{\Delta N}{2}$, and $t_2 = \frac{3\Delta N}{2}$.

And the slope for the right most portion of the polynomial (ΔS_2) would be:

$$\Delta S_2 = \frac{1}{\Delta N} \sum_{t=t_3}^{t_4} \Delta f'(t) \tag{4.6}$$

where $t_3 = N - \frac{3\Delta N}{2} - 1$, $t_4 = N - \frac{\Delta N}{2} - 1$.

The curve extension starts at the end points. (ΔN / 2) points are added to each end segment. The added point's value to the left most portion are computed from the value of the point

adjacent to the place of the added point (one unit apart at the t-axis), and the value of the average slope of the left most portion (ΔS_1):

$$f'(t) = f'(t+1) + \Delta S_1 \tag{4.7}$$

where, $t = \Delta N / 2, (\Delta N / 2) - 1, \ldots, 0$.

The value of the added points to the right most area are computed using the same previous method but using different limits and the average slope of the right most portion (ΔS_2):

$$f'(t) = f'(t-1) + \Delta S_2 \tag{4.8}$$

where, $t = N - 1 - (\Delta N / 2), N - (\Delta N / 2), \ldots, N - 1$.

When the original segment signature is reconstructed using (Equation 3.21), only the original (non-extended) signature pixels are retrieved. Therefore, only the original curve limits are considered (Figure 14):

$$f(t) \approx A_O + \sum_{k=1}^{N-1} a_k \cos\frac{2\pi kt}{N} + b_k \sin\frac{2\pi kt}{N}, t = \frac{\Delta N}{2}+1, \frac{\Delta N}{2}+2, \cdots, N - \frac{\Delta N}{2} - 1 \tag{4.9}$$

4.8.2.1 Fixed Feature Vector Size

The feature vector has two parts. The first part represents the relative location of the segment, whereas the second represents the shape of the segment.

The relative location of the segment is the relative center of mass for segment pixels to the pattern center of mass. The shape representation is ten Fourier descriptors (M = 10); each of those descriptors has two values (see section 3.4).

4.8.2.2 End-Points Problem

The trade-off between the number of FDs necessary for shape retrieval and how close the reconstructed shape is to the original one has always been an issue. Since an accurate reconstructed shape is needed using a few number of descriptors. This issue is not of great importance when dealing with classification and recognition systems. Only the lower-order FDs were sufficient for those systems. The general shape of an open-curve is easily reconstructed from a small number of lower-order FDs. Most of the high frequency components and noise are eliminated also. But on the other hand, using a small number of FDs occasionally introduces a problem at the start and the end segments of the open curve after reconstruction. The end segment of the reconstructed curve is greatly diverging from the end segment of the original curve. One of the reasons for this problem is the special way that Fourier Transformation (FT) deals with sudden changes (pulse functions). Since the points constituting the curve are discrete at both the start and end points, then it could be considered as a pulse function. The reconstruction problem increases (the offset between the original and the reconstructed areas) by decreasing the number of FDs used.

4.8.3 Algorithm

a) Segment signature: convert the open-curve into a polynomial curve using one of the signature functions [5].

b) Select an even positive number ΔN.

c) Curve shifting: shift the polynomial curve computed at step (a) (ΔN / 2) points along the positive axis (t-axis).

d) Slope calculation:

i) Compute the average slope ΔS_1 of the first ΔN points at the left extreme portion of the polynomial curve

ii) Compute the average slope ΔS_2 of the last ΔN points at the right extreme portion of the polynomial curve.

e) Curve extension:

i) Left portion extension:

- Compute the value of the first extended point beside the extreme point at the beginning of the curve by adding the average slope ΔS_1 computed at step (d) to the value of that point
- Repeat the previous step for (ΔN / 2) number of times after shifting the calculations one step to the left.

ii) Right portion extension:

- Compute the value of the first extended point beside the extreme point at the end of the curve by adding the average slope ΔS_2 computed at step (d) to the value of that point
- Repeat the previous step for (ΔN / 2) number of times after shifting the calculations one step to the right each time.

f) Fourier descriptors:

i) Compute the first ten Fourier coefficients using (Equation 3.18), and the curve values computed in the previous steps

ii) Then compute the ten FDs pairs using (Equations 3.19).

g) Signature reconstruction: reconstruct the curve from the ten FDs computed at the previous step using (Equation 3.21).

h) Trimming: ignore the first and last (ΔN / 2) points of the result of the previous step (Equation 4.9).

i) Shifting back: shift the resulted curve (ΔN / 2) points along the negative axis (t-axis).

4.8.4 Results

Using the traditional FDs, the number of descriptors was increased until reaching an accurately reconstructed signature (compared to the original signature curve). The centroidal-distance signature gave better results compared to the angle difference signature in terms of the number of FDs needed to reconstruct the original signature shape [5]. For this reason the centroidal-distance signature method was used in the experiments that follow.

Among the different segments taken from both the databases (more than 100 segments from each database, and more than 250 segments in total), 10 Fourier descriptors were sufficient to accurately reconstruct 65 percent of the different segments. Whereas the other 35 percent of the segments needed more than 10 descriptors for an accurate reconstruction. Depending on the complexity of the segment signature shape, some of those segments needed about 100 Fourier describers. Those segment signatures that failed to be described by ten Fourier descriptors were fed to the augmented FDs (see Figure 17). The reconstructed signature of each was evaluated by computing the error distance between that

signature and the original signature. This error distance was computed for the reconstructed signature before and after the augmented FDs method.

4.8.4.1 Distance Error (DE) Comparison

The original signature and the reconstructed one have the same number of points. Measuring the gap between each two points - from the original curve and the reconstructed one –indicates how close the reconstructed point is to the original one. If there were no gap the results would be zero, and as the gap increases the results would consequently increase. When adding the results of all the points along the two curves the result would be a measure of accuracy on how far the reconstructed signature is compared to the original one. This measure is called the Distance Error (DE)

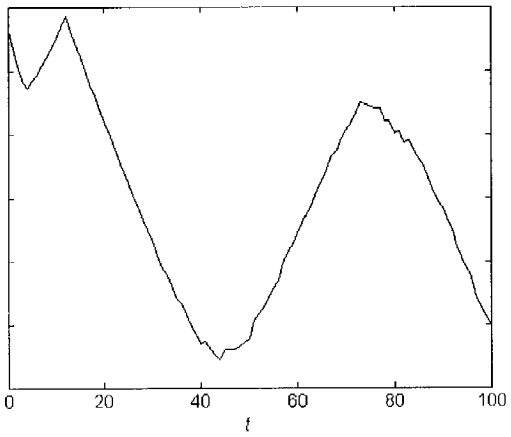
$$DE = \sum_{i=1}^{n} \left(x_i - x_i'\right)^2 \tag{4.10}$$

where, n is the total number of points in a given curve; x_i is the i^{th} value of the original signature curve; and, x'_i is the i^{th} value of the reconstructed signature curve.

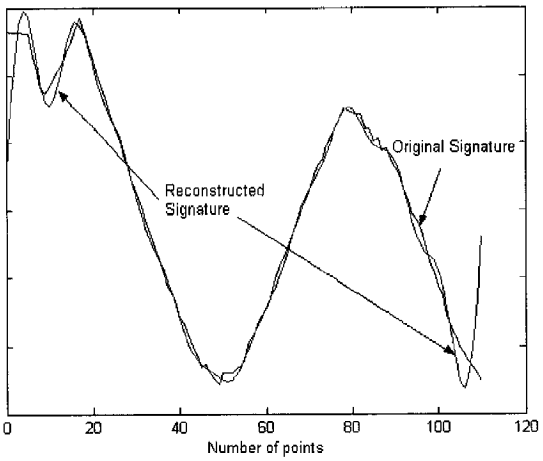
The DE between the original signature of the handwritten segment and the reconstructed signature curve before and after applying the augmented FDs method were computed. Then the difference between the two DEs was computed to compare the two results. The DE comparison results showed that about 96.5 percent of the segments used showed a decrease in the error after applying our method. The DE results prove that the reconstructed signature shape is changing towards the original signature shape after applying the augmented FDs method. (Figure 18) shows the histogram of the normalized DE for the two reconstructed

signatures for the dataset before and after using the augmented FDs method. (Table 7) shows the average DE for both signatures over the dataset with the standard deviation.

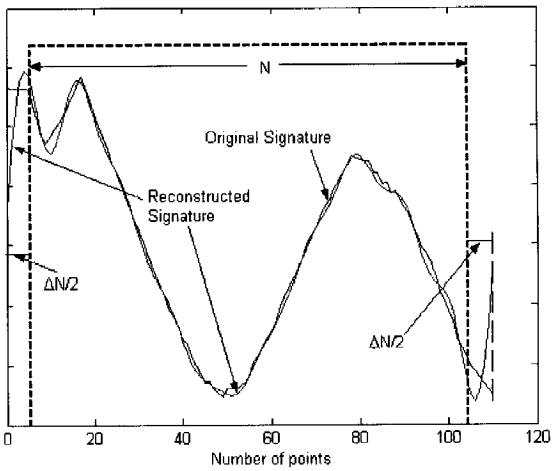
(a)



(b)



(c)



(d)

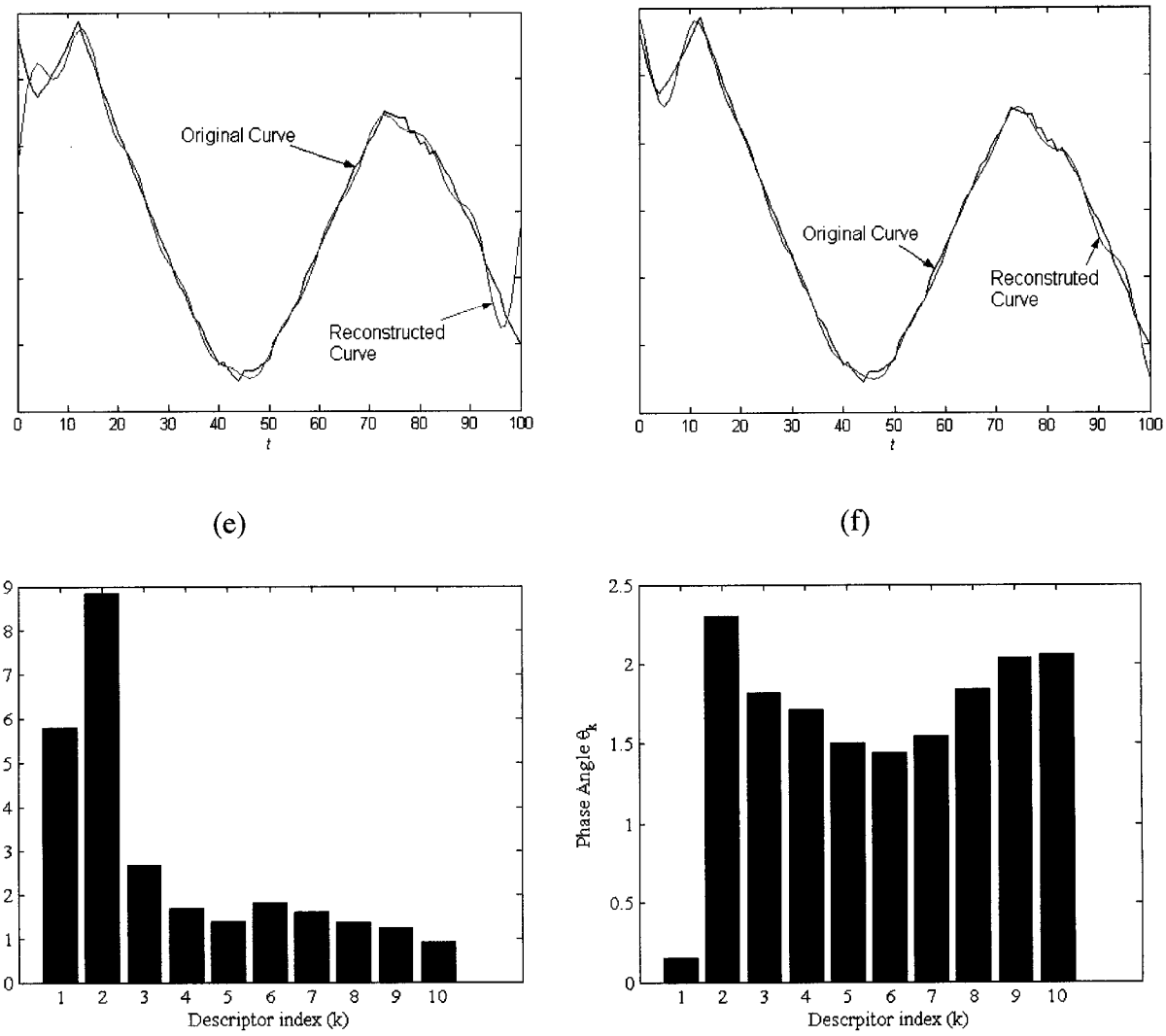


Figure 17. The Process Of Reconstructing The Signature Of A Handwritten Segment Signature Using Limited Number Of Fourier Descriptors; (a) The Handwritten Character Segment; (b) It's Centroidal-Distance Signature [5]; (c) The Extended Signature With The Reconstructed One From 10 Fourier Descriptors; (d) The Selection Processes; (e) The Reconstructed Signature Along With The Original Signature Using The Classical FD Method ;(f) The Modified Reconstructed Signature Along With The Original One; (f) And (g) Are The Ten Harmonic Amplitudes And Phase Angles Of The Fourier Descriptors Vector

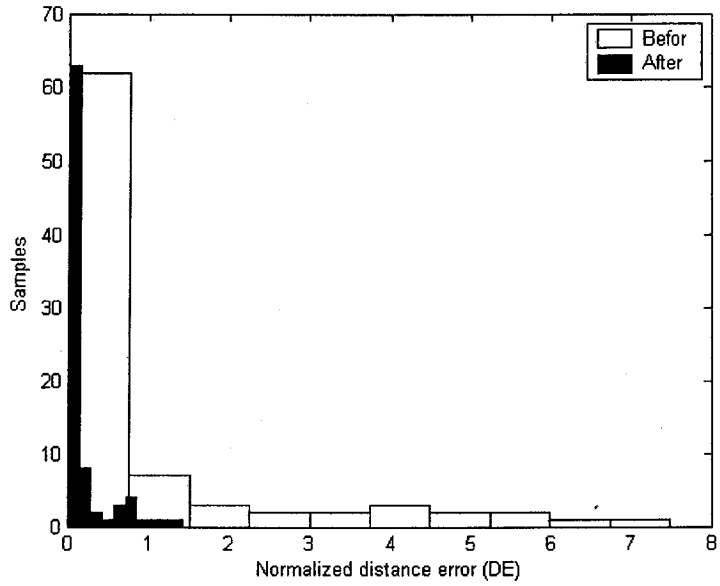


Figure 18. Histogram For The Normalized DE For The Reconstructed Signature Before Applying The Augmented FDs Method And After

Table 5. A Comparison Of The Average Normalized DE For The Reconstructed Signatures Before And After Applying The Augmented FDs Method

Method	**Average DE over the dataset**	**STD of the average DE over the dataset**
Before applying the augmented FDs method	1.028568	1.671667
After applying the augmented FDs method	0.193762	0.295818
Percentage Improvement	81 %	

4.8.5 Summary and Discussion

Fourier descriptors are very useful as features for reconstructing the shape of curves. Small number of descriptors was sufficient for reconstructing complex handwritten segments. The problem of reconstructing the curve end points were solved by extending the original curve, then the reconstruct curve were trimmed to extract the modified reconstructed curve. This modified FDs method proved to produce curves that are closer to the original shape than the ones produced by the traditional method. The results were evaluated both visually and using the distance error (DE) between the reconstructed curve and the original one in both cases. The DE comparison results showed improvement in reducing the distance between the two curves after using the augmented FDs.

Chapter 5

Conclusion and Final Thoughts

In this study, a swarm of autonomous agents is used to process handwritten Arabic characters and Chinese formal numerals. Each agent has local decision making capability and works independently and in parallel with other agents (see section 4.1). Furthermore, every agent is composed of a number of behaviors. Each behavior satisfies one or more task. The input to the behaviors is the local neighborhood of the agent; the output is an action executed by one of its behaviors. The behaviors are largely independent of one another: each one evaluates its own conditions (for activation), and decides to activate (or not) its own output.

Initially, a number of agents are created and initialized. Each agent starts to look for (edges of) patterns inside an image, using its edge detection behavior. Once an edge is found, the thinning-segmentation behavior becomes active and asserts its own output (see section 4.6). After segmentation and thinning (which are simultaneous processes) are carried out to completion, the feature extraction behavior becomes active. It will compute a fixed number

of Fourier descriptors to be used later for classification or reconstruction. Some time after completing its tasks, an agent will naturally die and be removed from the image.

Agents have subsumption architecture. Each behavior receives the same information from the local environment. The behaviors inside the agent have fixed priorities (or levels). These levels determine which of the active behaviors will execute at any given moment. Higher-level behaviors can subsume lower-level behaviors, and impose their own output on all the other lower level behaviors. The architecture of the agent was able to manage the different behaviors efficiently. The agents acted fast in locating patterns, thinning and segmenting them, and then extracting their features (see section 4.5).

The segmentation process of the handwritten skeletons is based on intersection and end points (see section 3.3). Both points are considerably easy to extract from skeletons and even from patterns. The result of this segmentation strategy is a number of open curves. The segmentation process is consistent, except when the intersection area has a lot of noise. The intersection area of two strokes suffers from noise more than the other parts of the handwritten pattern.

The thinning and segmentation processes were combined into one behavior. The two processes work together to produce thinned segments from the image pattern. The thinning-segmentation behavior is only applied to the areas of interest inside the image. Only after finding a pattern in the image does the behavior start to extract the segments. This process made the thinning-segmentation behavior effective, by acting only on those areas of the image that have patterns. Moreover, combining the two processes, of thinning and segmentation, made the system even faster. Many of the thinning artifacts and problems were partially solved. Bifurcation and elongation artifacts were reduced. Salt and pepper noise is

another type of noise that the system as a whole manages to eliminate. The edge finding behavior ignored all the pepper noise on the image, and the thinning-segmentation behavior was insensitive to white noise inside the pattern. The thinning-segmentation behavior dealt with the white noise area inside the pattern in different ways. As white areas get larger, the agent tends to treat the surrounding black areas as two separate segments (see section 3.1 and 4.7).

The feature extraction behavior uses a specially augmented form of Fourier Descriptors (FDs) to describe the shape of each segment. Only ten pairs of descriptors were used to describe a segment, regardless of its complexity or length. The normalized square distance between the original and reconstructed signatures (of a segment) was used to judge the improvement gained through the use of augmented FDs. A large gain in accuracy was predicted and then confirmed by the empirical results (see section 4.8)

We see adaptation *via* evolution is the next major step for this project. The individual agents should be able to act on different classes of patterns- not just Arabic and Chinese characters and numbers. More behaviors will need to be added, for sure. But more importantly, the various parameters of the system can be automatically tuned by a Genetic Algorithm (or similar mechanism). This is not a trivial process and it is one that demands the definition of an accurate and (at the same time) not very complex fitness function. This fitness function must reflect the quality of the solution as whole: quality of segmentation, quality of thinning, and quality of feature extraction. There is not, yet, a set of universally accepted measures of objective quality of any one of these three processes. Existing measures of quality seem to be related to fuzzy concepts (e.g. centerline) or the ultimate objective of accurate classification.

In addition, the form of the solution (a set of parameter values) of different types and meanings, may force us to devise special representations and genetic operators that are more efficient than the "usual suspects" in dealing with the specific problems of pattern recognition.

References

[1] M. Mataric, "Behavior-based control: Examples from Navigation, Learning, and Group Behavior", *Journal of Experimental and Theoretical Artificial Intelligence*, Special Issue on Software Architectures for Physical Agents, Vol.9, Nos.2-3, Hexmor Horswill Kortenkamp eds., 1997.

[2] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, RA-2, pp.4-23, 1986.

[3] M. Mataric, "Interaction and Intelligent Behavior", *Technical Report AI-TR-1495*, MIT Artificial Intelligence Lab, 1994.

[4] M. Mataric, "Reward Functions for Accelerated Learning", *In Proceedings of the Eleventh International Conference on Machine Learning, San Francisco*, Morgan Kaufman, 1994.

[5] Y. Wang, and B. Yuan, "Fast method for face location and tracking by distributed behaviour-based agents," *IEE proc.-VIS, Image Signal Processing*, Vol.149, No.3, pp. 173-178, 2002.

[6] J. Liu, Y. Y. Tang, and Y. C. Cao, "An evolutionary autonomous agents approach to image feature extraction," *IEEE Transactions on Evolutionary Computation*, Vol.1, No.2, pp.141–158, 1997.

[7] J. Liu, D. Maluf, and Y. Y.Tang, "Toward evolutionary autonomous agents for computational perception," *Proceedings of the 1997 IEEE International Conference on System, Man, and Cybernetics*, Vol.4, pp.3096-3101, 1997 (invited paper).

[8] J. Liu and Y. Y. Tang, "Adaptive segmentation with distributed behavior-based agents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.21, No.6, pp. 544-551, 1999.

[9] E. Angelotti *et al*, "A Paraconsistent System of Autonomous Agents for Brazilian Bank Check Treatment," *XX International Conference of the Chilean Computer Science Society*, Vol.16, No.8, pp.89-98, 2000.

[10] R. Alhajj *et al*, "Employing Multi-Agents to Identify Touching of Adjacent Digits in Handwritten Hindi Numerals," *Proceedings of the IEEE International Conference on Systems,* Man, and Cybernetics, Tennessee, USA, pp. 2725-2730, 2000.

[11] G. Butler, A. Gantchev, and P. Grogono, "Object-Oriented Design of the Subsumption Architecture," *Software --- Practice and Experience*, Vol.31, pp.911-923, 2001.

[12] D. Goldberg , and M. Mataric, "Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks," in *Robot Teams: From Diversity to Polymorphism*, Tucker Balch and Lynne E. Parker, eds. AK Peters, 2002.

[13] B. Gerkey, and M. Mataric, "Sold!: Auction Methods for Multirobot Coordination," *IEEE transactions on robotics and automation*, Vol.18, No.5, 2002.

[14] S. Lee, and J. C. Pan, "Offline Tracing and Representation of Signatures," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.22, No.4, pp.755-771, 1992.

[15] R. Plamondon, and C.M. Privitrea, "The Segmentation of Cursive Handwriting: An Approach Based on Off-Line Recovery of the Motor-Temporal Information," *IEEE Transactions on Image Processing*, Vol.8, No.1, pp.80-91, 1999.

[16] X. Li, M. Parizeau and R. Plamondon, "Segmentation and reconstruction of on-line handwritten scripts," *Pattern Recognition*, Vol.31, No.6, 1998.
[17] F. Lin and X. Tang, "Off-line handwritten Chinese character stroke extraction," *Proc. Int'l Conf. Pattern Recognition*, pp.249 –252, 2002.
[18] G. H. Granlund, "Fourier preprocessing of hand printed character recognition," *IEEE Trans. Comp.*, Vol.C-21, pp.195-201, 1972.
[19] C. T. Zhan, and R. S. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. Comp.*, Vol.C-21, pp.269-281, 1972.
[20] H. T. Sheu, and M. F. Wu, "Fourier descriptor based technique for reconstructing 3D contours from stereo images, Vision, Image and Signal Processing," *IEE Proceedings*, Vol.142, No.2, pp.95-104, 1995.
[21] L. Yang, and R. Prasad, "Recognition of line-drawing based on generalized Fourier descriptors," *International Conference on Image Processing and its Applications*, pp.286-289, 1992.
[22] D. Zhang, and G. Lu, "A comparative study of Fourier descriptors for shape representation and revival," *In Proc. of the Fifth Asian Conference on Computer Vision*, Melbourne, Australia, pp.646-651, 2002.
[23] D. Zhang, "Image Retrieval Based on Shape," *PhD Thesis, Monash University*, Churchill, Victoria, Australia, 2002.
[24] N. Kharma and R. Ward, "A Novel Invariant Mapping Applied to Handwritten Arabic Character Recognition", *Pattern Recognition*, Vol.34, No.11, 2001.
[25] Chinese official numbers database, available at: http://www.ece.concordia.ca/~kharma/ExchangeWeb/DataBases/ChineseNo/

Appendix A

Segmentation and Thinning-Segmentation Results CD

The attached CD has the segmentation results and the Thinning-Segmentation results (see section 4.7.6). The CD has two main folders. One folder has the name "Segmentation_Results". This folder has inside it two other folders with the names "Chinese_Numbers" and "Arabic_Words". Those two folders have the figures of the segmentation algorithm used in the course of this study for the Chinese official numbers database and the Arabic words database respectively. The figures are of different sizes and dimensions, but all are Windows® bit map color files.

The second main folder has the name "Thinning_Segmentation_Results". It also has inside it two other folders with the names "Chinese_Numbers" and "Arabic_Words". Those two folders have the figures of the thinning-segmentation behavior used in course of this study for the Chinese official numbers database and the Arabic words database respectively. The figures are of different sizes and dimensions, but all are Windows® bit map color files.