Chapter 3

Connectionist Model with Large Neighborhood for Line Linking

3.1 Introduction

The previous method considered only 1-neighborhood for a pixel. It can link the line when the distance of any two broken points is small. In the real seismogram, we can often find out that two points are in the same horizon but the distance of two points is too large. Finally, they can not link together.

Therefore, we must take into account large neighborhood to get more information by the points in a larger neighborhood. If large neighborhood is used, it is difficult to determine which neighboring pixels should reinforce each other only depending on the template type. Hence, we consider using line direction and strength to determine which neighboring pixels can give reinforcement. Basak propose another connectionist model [4] with large neighborhood for edge linking. The model regards gradient of each pixel as inputs. The output of the interesting pixel is increased or decreased based on the gradient values input to the pixel in the neighborhood. The detail architecture we discussed in next section. Because the model proposed by Basak base on gray level images. In our approach, we modify the network architecture which proposed by Basak to solve the line linking problem. We give direction and strength to each pixel by its neighboring information and its gray value. Depending on the position relationship of interesting pixel and its neighboring pixels, we can decide how much strength the interesting pixel can get. The size of neighborhood of each pixel decreases with time. When radius of neighborhood radius decreases to zero, we stop adjustment.

After stopping adjustment, the output is the linking result. We observe that some linked lines become thick after linking. In order to solve this problem, we use thinning technique [6] to the final output image. After thinning process, we get a better linking result. The Figure 3.1 shows the diagram of line linking processing.

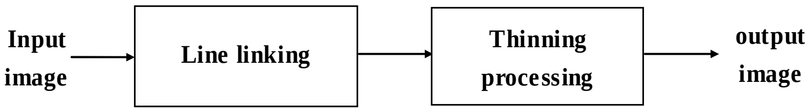
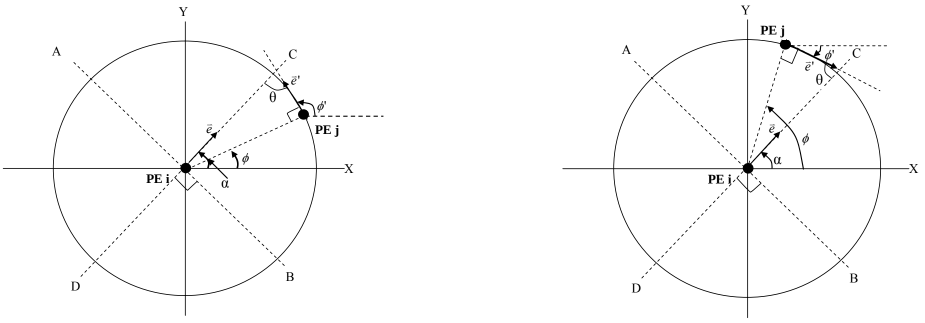


Figure 3.1.Diagram of line linking processing.

3.2**Edge Linking**

In this section, we introduce the network architecture proposed by Basak [4]. In this network architecture, the model is considered to consist of $m \times n$ processing elements (PEs) if the image contains $m \times n$ pixels. Every PE has an edge vector whose direction is the direction of the intensity change, and the magnitude is the absolute value of the first derivative. The direction is normal to the edge. The concept is that each edge vector induces edge points over a neighborhood. In this model, the neighborhood is selected as circular. Figure 3.2 illustrates the concept of edge induction.



(a) (b)

Figure 3.2. Edge induction on a circular neighborhood (Basak [4]).

Consider processing element i (PE i) with edge vector $\vec{e} = (e, \alpha)$, where e is the magnitude and α is the direction of vector $\vec{e}$. The circle around PE i is the circular neighborhood of PE i. Edge points are expects appear along the line $\overline{\mathrm{AB}}$ which is perpendicular to the edge vector. Hence, PE i should try to induce edge vectors at every point on $\overline{\mathrm{AB}}$ and not affect any point on $\overline{\mathrm{CD}}$. PE j denote the neighboring point of PE i. The affect of PE i on the points on $\overline{\mathrm{PEiPEj}}$ increases as PE j close to $\overline{\mathrm{AB}}$ and decreases as PE j close to $\overline{\mathrm{CD}}$.

In a rectangular coordinate system, the edge vector can be written as (e_x, e_y), where e_x and e_y are the components along the x-axes and y-axes respectively. The affect of PE i on the PE j is $\vec{e}' = (e', \phi')$, where e' is the magnitude and ϕ' is the direction of the induced vector. The induced vector $\vec{e}'$ can also be written as (e_x', e_y'). From the Figure 3.2,

$$\phi' = \begin{cases} \pi/2 + \phi & \text{if } \alpha > \phi \\ \phi - \pi/2 & \text{otherwise} \end{cases}$$

where ϕ is the angle subtended by the line $\overline{\mathrm{PEiPEj}}$ with x-axes. The magnitude of induced vector can get by

$$e' = e\cos\theta, \text{ where } \theta = |\alpha - \phi'|$$

The relationship of edge vector and induced vector on the components along x-axes and y-axes can be written as

$$e_x' = e_x \sin^2\phi - e_y \sin\phi\cos\phi$$
$$e_y' = e_y \cos^2\phi - e_x \sin\phi\cos\phi$$

Proof：

Case1: $\alpha > \phi$

(1)
$$\begin{aligned} e_x' &= e'\cos\phi' \\ &= e\cos\theta\cos\phi' \\ &= e\cos(\phi'-\alpha)\cos\phi' \\ &= e\cos(\frac{\pi}{2}+\phi-\alpha)\cos(\frac{\pi}{2}+\phi) \\ &= e\sin(\phi-\alpha)\sin\phi \\ &= (e\sin\phi\cos\alpha - e\cos\phi\sin\alpha)\sin\phi \\ &= e_x\sin^2\phi - e_y\cos\phi\sin\phi \end{aligned}$$

(2)
$$\begin{aligned} e_y' &= e'\sin\phi' \\ &= e\cos\theta\sin\phi' \\ &= e\cos(\phi'-\alpha)\sin\phi' \\ &= e\cos(\frac{\pi}{2}+\phi-\alpha)\sin(\frac{\pi}{2}+\phi) \\ &= \text{-}\,e\sin(\phi-\alpha)\cos\phi \\ &= \text{-}\,(e\sin\phi\cos\alpha - e\cos\phi\sin\alpha)\cos\phi \\ &= e_y\cos^2\phi - e_x\cos\phi\sin\phi \end{aligned}$$

Case2: $\alpha \le \phi$

(1)
$$\begin{aligned} e_x' &= e'\cos\phi' \\ &= e\cos\theta\cos\phi' \\ &= e\cos(\alpha-\phi')\cos\phi' \\ &= e\cos(\alpha-\phi+\frac{\pi}{2})\cos(\phi-\frac{\pi}{2}) \\ &= \text{-}\,e\sin(\alpha-\phi)\cos(\frac{3\pi}{2}+\phi) \\ &= \text{-}\,e\sin(\alpha-\phi)\sin\phi \\ &= \text{-}\,(e\sin\alpha\cos\phi - e\cos\alpha\sin\phi)\sin\phi \\ &= e_x\sin^2\phi - e_y\cos\phi\sin\phi \end{aligned}$$

(2)
$$\begin{aligned} e_y' &= e'\sin\phi' \\ &= e\cos\theta\sin\phi' \\ &= e\cos(\alpha-\phi')\sin\phi' \\ &= e\cos(\alpha-\phi+\frac{\pi}{2})\sin(\phi-\frac{\pi}{2}) \\ &= e\sin(\alpha-\phi)\sin(\frac{\pi}{2}-\phi) \\ &= e\sin(\alpha-\phi)\cos\phi \\ &= (e\sin\alpha\cos\phi - e\cos\alpha\sin\phi)\cos\phi \\ &= e_y\cos^2\phi - e_x\cos\phi\sin\phi \end{aligned}$$

The model regards the normalized gradient component as the inputs and updates the network by the relationship of edge vector and induced vector.

3.2.1 **Network Architecture**

The network is composed of PEs arranged in a two dimensional lattice. Each PE connects with a large neighborhood and self-feedback. Figure 3.3 shows the interconnection between two processing elements i and j in the network.

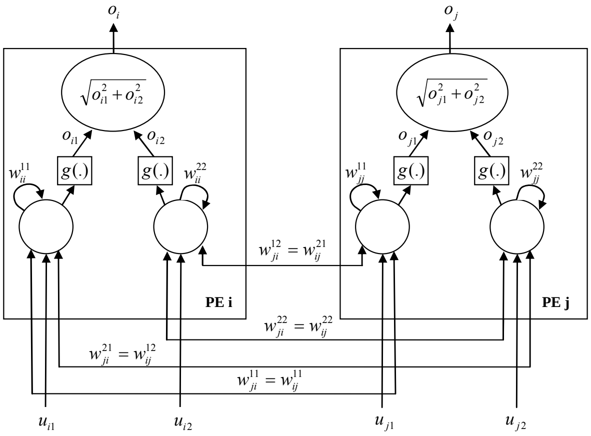


Figure 3.3.Interconnections between two processing elements i and j in the network. (Basak [4])

Initially, the Sobel operator are used to calculate edge response e_x and e_y at every pixel where e_x and e_y is the first partial derivative along the x-axes and y-axes respectively. The resultant response e and edge direction α are found by

$$e=\sqrt{e_x^2+e_y^2}$$

and

$$\alpha=\tan^{-1}(e_y/e_x)$$

The edge responses at all pixels are normalized to (0, 1) by maximum value of e. The normalized edge vector of PE i are denoted by u_i. Then, the components along the x and y directions of normalized edge vector, u_{i1} and u_{i2}, are regarded as network inputs, i.e. $u_{i1}=u_i\cos(\alpha)$, $u_{i2}=u_i\sin(\alpha)$. Each PE has two intermediate outputs. The intermediate outputs o_{i1} and o_{i2} represent the edge responses along the x and y direction of PE i and can be written as

$$o_{i1}=g(u_{i1})$$
$$o_{i2}=g(u_{i2})$$

where $g(.)$ is a transfer function that defined as

$$g(x)=\begin{cases}0 & \text{if } x<-1\\ x & \text{if } -1\le x<1\\ 1 & \text{otherwise}\end{cases}$$

Figure 3.4 shows the transfer function $g(.)$.

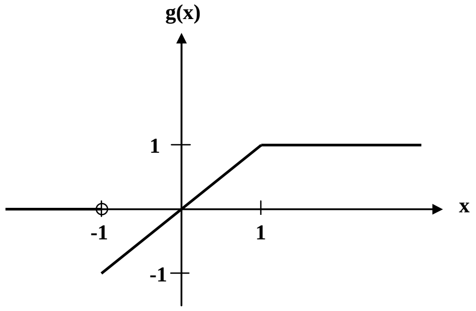


Figure 3.4.Transfer function.

The output of any PE i is gives as

$$o_i=\sqrt{o_{i1}^2+o_{i2}^2}$$

where o_{i1} and o_{i2} are intermediate outputs of the PE i.

Basak propose the update formula by the relationship of edge vector and induced vector. The update formula can be written as

$$u_{i1}(t+1)=\sum_{j\in N_{R(t)}(i)} w_{ji}^{11}o_{j1}(t)+\sum_{j\in N_{R(t)}(i)} w_{ji}^{12}o_{j2}(t)+w_{ii}^{11}o_{i1}(t)$$
$$u_{i2}(t+1)=\sum_{j\in N_{R(t)}(i)} w_{ji}^{21}o_{j1}(t)+\sum_{j\in N_{R(t)}(i)} w_{ji}^{22}o_{j2}(t)+w_{ii}^{22}o_{i2}(t)$$

where the w_{ji}^{rs} is the connection weight from component *r* of PE j to component *s* of PE i. The connection weights are symmetric, i.e. $w_{ji}=w_{ij}$, and be defined as

$$w_{ji}^{11}=w_{ij}^{11}=w\sin^2\phi_{ij}$$
$$w_{ji}^{22}=w_{ij}^{22}=w\cos^2\phi_{ij}$$
$$w_{ji}^{12}=w_{ij}^{21}=-w\sin\phi_{ij}\cos\phi_{ij}$$
$$w_{ii}^{11}=w_{ii}^{22}=-w_sR(t)$$

where w and w_s are constant and ϕ_{ij} is the angle subtended with x-axes by the line $\overline{\text{PEiPEj}}$. The w_{ii}^{11} and w_{ii}^{22} are the self-feedback of PE i. For all PEs the amount of self-feedback are the same and proportional to the radius of the neighborhood of activity.

In this model, the size of neighborhood of each PE decreases with time. When radius of neighborhood radius decreases to zero, we stop adjustment. The radius decreases linearly at ratio γ. The formula can be written as

$$R(t+1) = R(0) - \gamma t$$

where $R(0)$ is the initial radius at $t = 0$.

In the network, each PE has a circular neighborhood. In discrete domain, for any PE i, if PE j belongs to the neighbor of PE i, the distant between PE i and PE j have to satisfy the equation

$$D(i, j) \le R + 0.5$$

where $D(i, j)$ denote the Euclidian distance between PE i and PE j and R is the radius of the circular neighborhood in analog domain.

3.2.2 Experiments

Experiment I: circle

The input of first experiment is shown in Figure 3.5 with size 50×50. The parameters of the $R(0) = 2$, $w = 100$, $w_s = 180$. The linking result shows in Figure3.6.

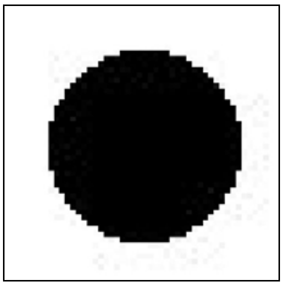


Figure 3.5. Input image.

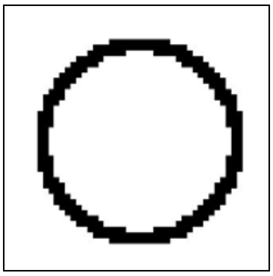


Figure 3.6. Linking result.

Experiment II: personal face

The input of first experiment is shown in Figure 3.7 with size 182×200. The parameters of the $R(0)=2$, $w=100$, $w_s=180$. The linking result shows in Figure3.8.

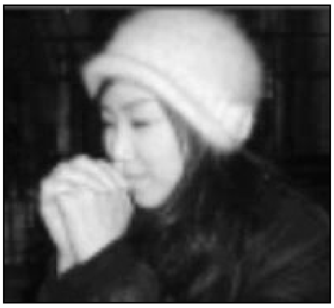


Figure 3.7. Input image.

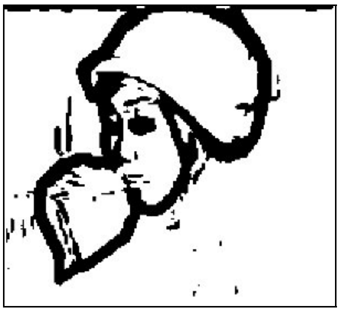


Figure 3.8. Linking result.

3.3 Line Linking

In this section, we reference the architecture on previous section and modify the input information and update formula to solve the line linking problem. Here, the model is considered to consist of $m \times n$ processing elements (PEs) if the image contains $m \times n$ pixels. The concept diagram is shown in Figure 3.9. To consider a small area around a PE on a continuous line, we suppose there has a major line segment inside the small area. We try to find the major line for every PE in the image and use the neighboring points which close to the direction of major line to link broken line segments.

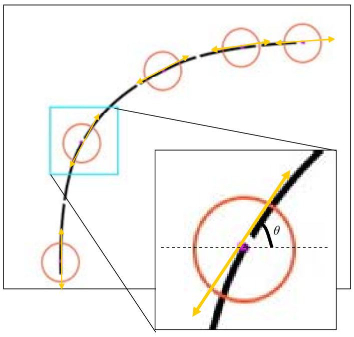


Figure 3.9. Concept diagram of line linking.

3.3.1 Strength and Line Direction

There are many methods can detect line successfully such as Hough transform

[7], line mask that discussed in chapter 2 and so on. In our approach, it will take large cost if we use Hough transform and useless if we use line mask. Because the distance of any two points maybe over 1-neighborhood, we can't detect a line only use a 3×3 line mask.

Hence, we choose linear regression to detect line direction. Linear regression is a method that can get a line which let the sum of distance from points to the line minimize. The relationship of points and line is illustrated in the Figure 3.10.

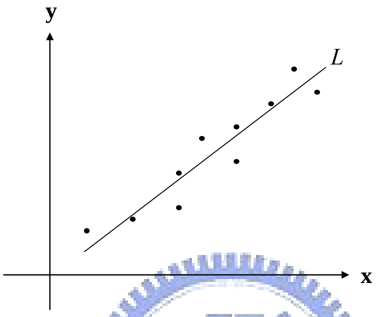


Figure 3.10. Relationship of points and line.

In our approach, for each PE, we set a window to them and the interesting PE is in the center of the window. Then, find a line in the window by linear regression. We regard the line slope as the direction of the interesting PE. Figure 3.11 illustrates the direction of a PEs. The rectangle represents the window, pixel in the center of the window represents the interesting PE, and the line is the result of linear regression, namely direction of interesting PE. The direction of each PE can express as an angle subtended with the line by the x-axis called theta (θ), the range of theta is between $-\pi/2$ to $\pi/2$. If there has less than one point in the defined window, we can not get a line in this window. In this condition, we set the angle equal to infinite for the interesting PE.

ES 1896

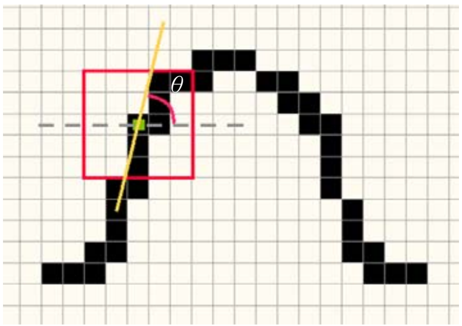


Figure 3.11.Linear regression for a PE.

Direction

Now, we introduce how to get the line's parameters using linear regression and line direction theta θ.

Suppose the line equation of L is $y = ax + b$ and the coordinate of points are represented as (x_i, y_i) $i = 1, \cdots, n$.

The sum of distance can be written as

$$S = \sum_{i=1}^{n} (y_i - y)^2$$

And we know $y = ax + b$,

$$S = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

Because we want S to be minimized, we can get two equations by partial derivative with respect to a and b respectively.

$$\begin{cases} \dfrac{\partial S}{\partial a} = -\displaystyle\sum_{i=1}^{n} 2x_i(y_i - ax_i - b) = 0 \\ \dfrac{\partial S}{\partial b} = -\displaystyle\sum_{i=1}^{n} 2(y_i - ax_i - b) = 0 \end{cases}$$

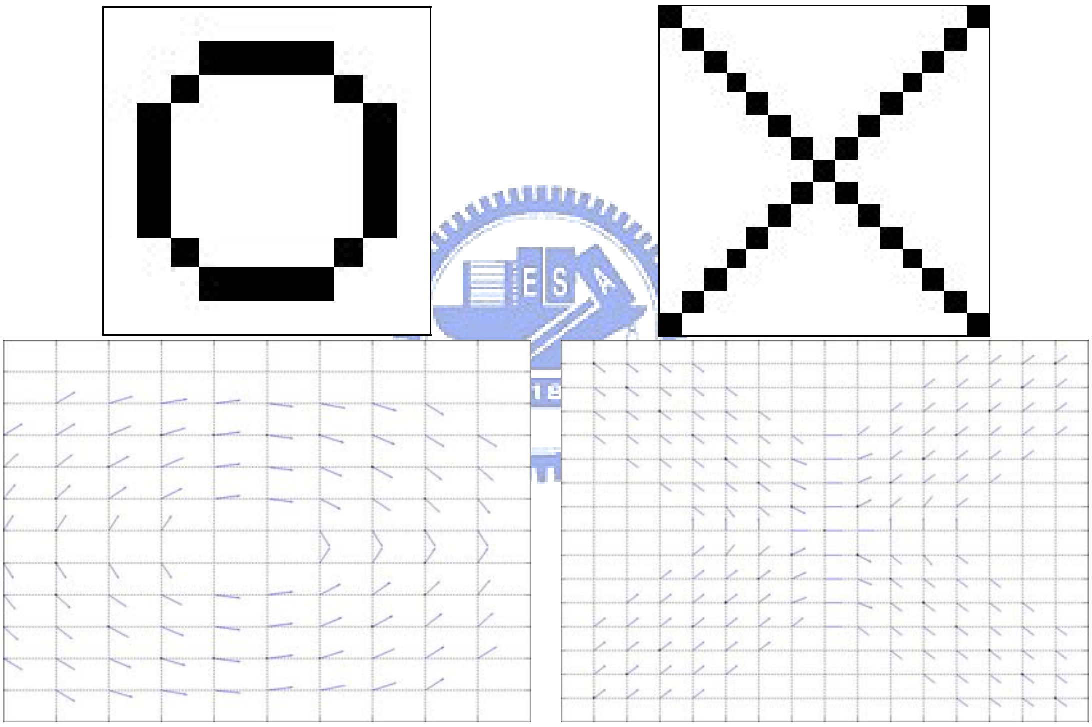
Solve these two equations, we get a and b as follow.

$$\begin{cases} a = \dfrac{n(\sum_{i=1}^{n} x_i y_i) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2} \\ b = \dfrac{(\sum_{i=1}^{n} y_i)(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} x_i y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2} \end{cases}$$

The parameter a represents the line's slope. We can get theta by

$\theta = \tan^{-1}(a)$

Figure 3.12 shows two linear regression results with window size 5×5.



(a) circle(size 10×10) (b) cross line(size 15×15)

Figure 3.12. Linear regression results with window size 5×5.

Strength

In our approach, we set the initial strength as the gray value of the pixel. If the gray value is 1 (black point), the initial strength set to 1, and if the gray value is 0 (white point), initial strength set to 0.

3.3.2 **Network Architecture**

In the implementation of the connectionist model for line linking, the model is considered to consist of $m \times n$ processing elements (PEs) if the image contains $m \times n$ pixels. Each PE connects with its neighboring PEs with a large neighborhood and a self-feedback. The neighborhood is selected as circular. Figure 3.13 shows a PE i connects with its neighboring PEs with radius $\mathrm{R}=2$. In discrete domain, for any PE i, if PE j belongs to the neighbor of PE i, the distant between PE i and PE j have to satisfy the equation

$$D(i,j) \le \mathrm{R} + 0.5$$

where $D(i,j)$ denote the Euclidian distance between PE i and PE j and R is the radius of the circular neighborhood in analog domain.

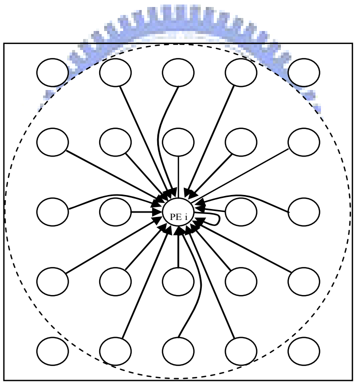


Figure 3.13. A connectionist model with radius $\mathrm{R}=2$.

In our approach, the state can be written as $\vec{u}_i = (u_i, \theta_i)$, where u_i is the state strength, θ_i is the state direction of processing element i. Initially, we calculate direction of each processing element by linear regression and set the strength with its gray value. The output of each processing element can be written as $\vec{o}_i = (o_i, \theta_i)$ where o_i is the output strength and θ_i is the output direction.

The state strength adjusted by update formula. The formula can be written as

$$u_i(t+1) = u_i(t) + \sum_{j \in N_{R(t)}(i)} w_{ji} \cos(\alpha)^p \cos(\beta)^p o_j(t) - w_{ii} o_i(t)$$

The update formula will be discussed later.

The relationship of output strength and input strength can be written as

$$o_i = g(u_i)$$

where $g(.)$ is a transfer function shown in the Figure 3.14.

$$g(x) = \begin{cases} 1 & if \;\; x \geq 1 \\ x & if \;\; 0 < x < 1 \\ 0 & otherwise \end{cases}$$

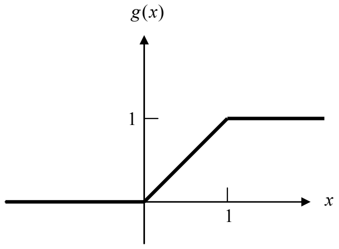
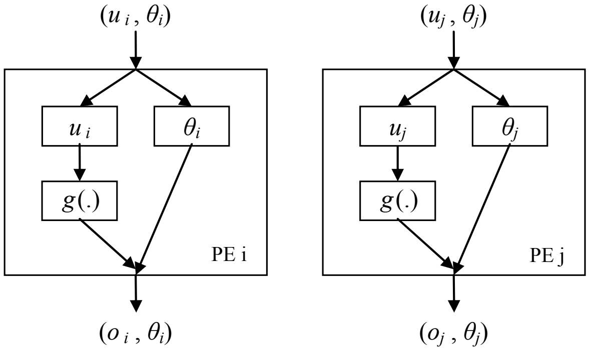
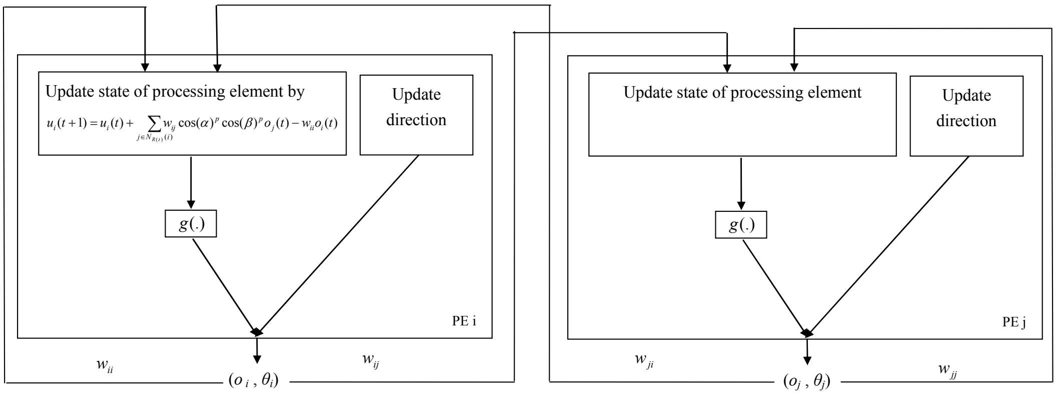


Figure 3.14. Transfer function.

The processing elements connect with each other if they have neighboring relationship. The Figure 3.15(a) shows the initial state of the network. The interconnections between two processing elements i and j are shown in Figure 3.15(b). Each processing element feed back its output strength and output direction information to its neighboring processing elements. We regard the point which getting few support or without getting any support from neighboring PEs as noise. In order to reduce the effect of noise, each processing element has a negative self-feedback.



(a)


Update state of processing element by
$u_i(t+1)=u_i(t)+\sum_{j\in N_{R(t)}(i)} w_{ij}\cos(\alpha)^p\cos(\beta)^p o_j(t)-w_{ii}o_i(t)$
Update direction
$g(.)$
PE i
w_{ii}
(o_i,θ_i)
w_{ij}
Update state of processing element
Update direction
$g(.)$
PE j
w_{ji}
(o_j,θ_j)
w_{jj}

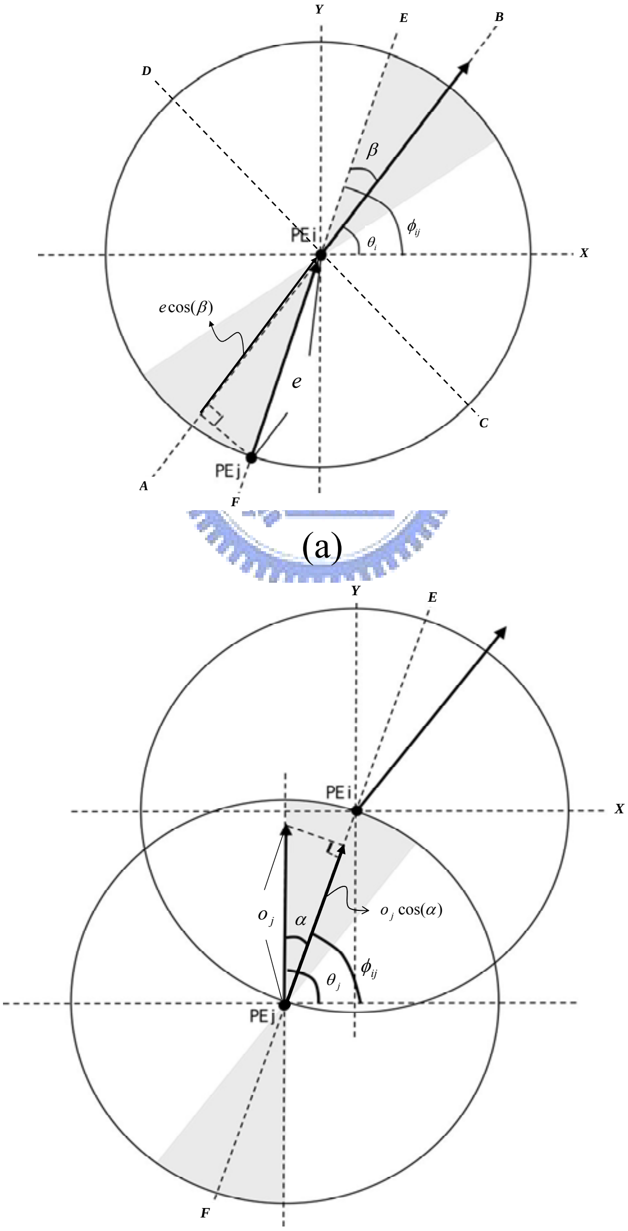
(b)

Figure 3.15. (a)Initial state, (b) Interconnections between two processing element i and j in the network.

In our approach, we reinforce a processing element by its neighboring information. Initially, we set radius for the interesting processing element and reinforce it by processing elements inside the neighborhood.

Figure 3.16 shows the diagram of reinforcement. The interesting processing element is processing element i (PE i) and the processing element j (PE j) represent the neighboring processing elements of PE i. The circle around the PE i is the range of neighborhood. Now, we consider how to use neighboring information to reinforce the PE i. The line direction of PE i can be represented as θ_i, namely the direction of the line $\overleftrightarrow{AB}$. The idea is that the PE i gets large reinforce from the PE j when PE j is close to the line $\overleftrightarrow{AB}$ and gets small reinforce when PE j is close to the line $\overleftrightarrow{CD}$. Furthermore, we consider the position relationship about PE i and PE j. The angle subtended with the x-axis by the line $\overleftrightarrow{EF}$ can be represented as ϕ_{ij}. If the PE j is close to the line $\overleftrightarrow{AB}$, namely the difference of ϕ_{ij} and θ_i, i.e. β, is small, then gives a large strength and small otherwise. Here, we suppose that the PE j has strength e on direction ϕ_{ij} shown in Figure 3.16(a). We can get a projection on $\overleftrightarrow{AB}$ from the PE j. The projection can be written as $e\cos(\beta)$.

Next, how to get strength e on the direction ϕ_{ij} will be discussed. In the Figure 3.16(b), the direction of PE j is θ_j and strength is o_j, we can get a projection on $\overleftrightarrow{EF}$ from the PE j. The projection can express as $e = o_j \cos(\alpha)$, the angle α is the difference of ϕ_{ij} and θ_j. Hence, the reinforcement from PE j to PE i can express as $o_j \cos(\alpha)\cos(\beta)$.

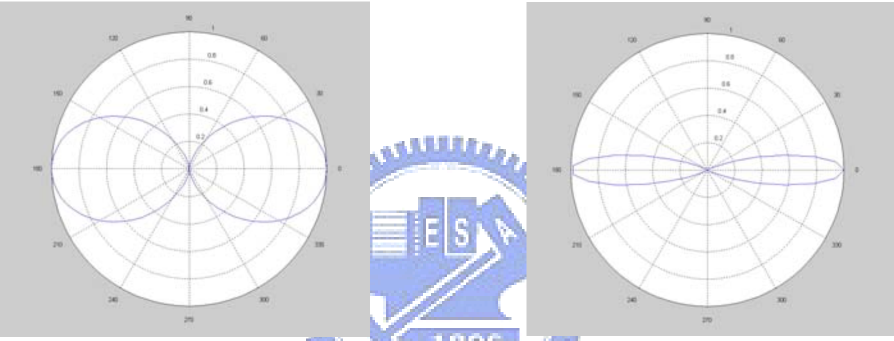


(a)

(b)

Figure 3.16. Diagram of reinforcement concept

In order to make a better result of line linking, we must consider the angle size of α and β. The angle α and β are expected to tend to zero, namely the PE j and PE i mostly on the same line. In this condition, the PE j gives the largest reinforcement to the PE i. If α and β tend to $\pi/2$, that represent PE j and PE i on the different line, the PE j do not reinforce the PE i anymore. Hence, we set a tolerance limit of angle α and β to decrease range of reinforcement. The PE j can reinforce PE i if the angle α and β in the range of tolerance limit. Figure 3.17 shows two order of $\cos(\omega)$ where $0 \le \omega \le 2\pi$. Small order has larger tolerance than high order.



(a) (b)

Figure 3.17. (a) $\cos^2(\omega)$, (b) $\cos^{30}(\omega)$, $0 \le \omega \le 2\pi$

Hence, the state of each processing element is update as

$$u_i(t+1) = u_i(t) + \sum_{j \in N_{R(t)}(i)} w_{ji} \cos(\alpha)^p \cos(\beta)^p o_j(t)$$

where p is a positive integer, w_{ji} is the connection weight of the link from processing element j to processing element i. It is assumed that all interconnections in the network are symmetric, i.e. $w_{ji} = w_{ij}$.

The point which getting few support or without getting any support from neighboring PEs is regarded as noise. In order to reduce noise, a negative self-feedback is added into the update formula. The negative self-feedback helps in removing noise during the link linking process. However, the self-feedback must smaller than reinforcement or the line points will be deleted. The new update formula can rewrite as

$$u_i(t+1) = u_i(t) + \sum_{j \in N_{R(t)}(i)} w_{ji} \cos(\alpha)^p \cos(\beta)^p o_j(t) - w_{ii} o_i(t)$$

where w_{ii} is the weight of the self-feedback and direct proportion to the size of neighborhood. So, the w_{ii} can be written as

$$w_{ii} = w_s R(t)$$

where w_s is a positive constant.

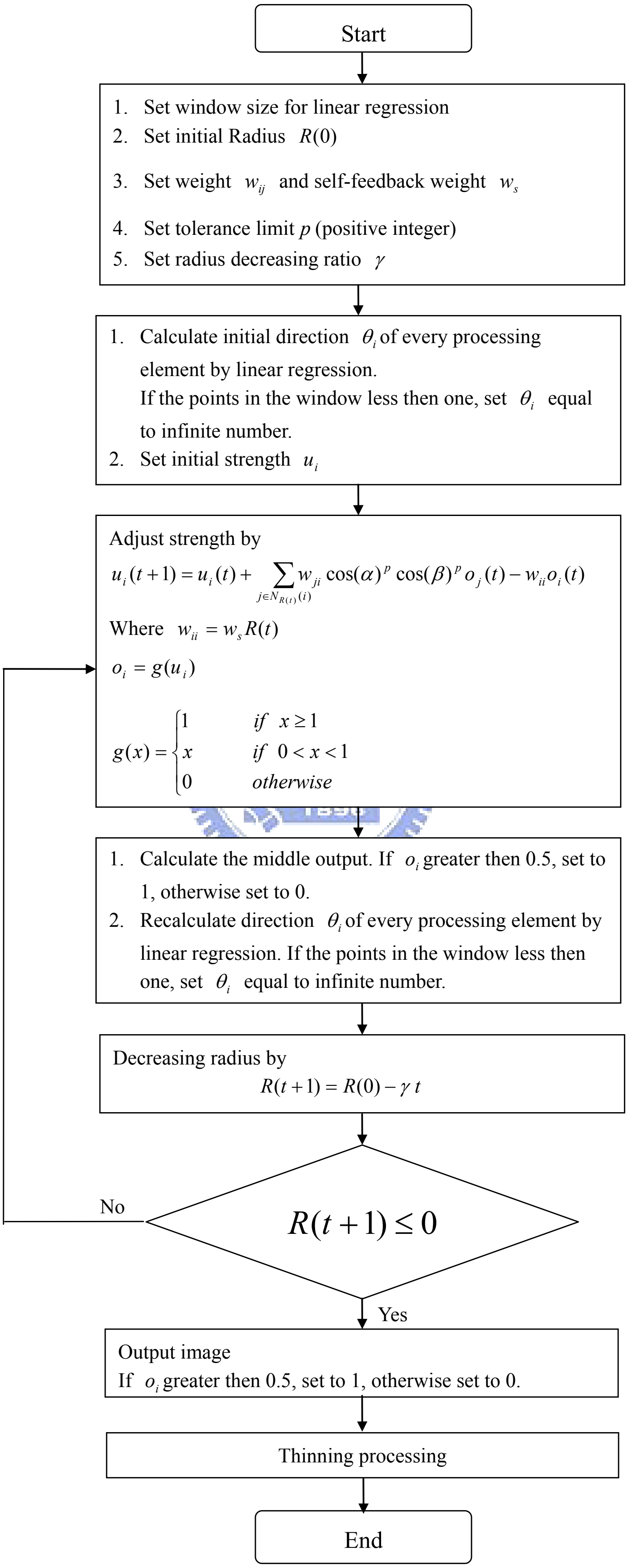
In the implementation of our approach, we recalculate line direction of each processing element after one update. Because some processing elements have no direction in the initial state, they can not get any reinforcement in the beginning. These processing elements maybe get direction after recalculating direction.

The size of the neighborhood of each processing element decreases with time. When the radius of neighborhood reduces to zero, stop the process of update. Since the self-feedback is proportional to the radius of neighborhood, the self-feedback also reduces to zero when radius decreases to zero. We suppose the radius decreases linearly at ratio γ. The formula can be written as

$$R(t+1) = R(0) - \gamma t$$

Where $R(0)$ is the initial radius at $t = 0$.

3.3.3 Programming Flowchart


Start
1. Set window size for linear regression
2. Set initial Radius $R(0)$
3. Set weight w_{ij} and self-feedback weight w_s
4. Set tolerance limit p (positive integer)
5. Set radius decreasing ratio γ
1. Calculate initial direction θ_i of every processing element by linear regression.
If the points in the window less then one, set θ_i equal to infinite number.
2. Set initial strength u_i
Adjust strength by
$u_i(t+1) = u_i(t) + \sum_{j \in N_{R(t)}(i)} w_{ji} \cos(\alpha)^p \cos(\beta)^p o_j(t) - w_{ii} o_i(t)$
Where $w_{ii} = w_s R(t)$
$o_i = g(u_i)$
$g(x) = \begin{cases} 1 & if \;\; x \geq 1 \\ x & if \;\; 0 < x < 1 \\ 0 & otherwise \end{cases}$
1. Calculate the middle output. If o_i greater then 0.5, set to 1, otherwise set to 0.
2. Recalculate direction θ_i of every processing element by linear regression. If the points in the window less then one, set θ_i equal to infinite number.
Decreasing radius by
$R(t+1) = R(0) - \gamma t$
$R(t+1) \leq 0$
No
Yes
Output image
If o_i greater then 0.5, set to 1, otherwise set to 0.
Thinning processing
End

3.3.4 **Experiments**

Experiment I: circle

The input of first experiment is shown in Figure 3.18 with size 50×50. The parameters of the window size of linear regression is set to 5×5, $R(0) = 2$, connection weight $w_{ji} = 0.5 * (1/R(0)), \forall \mathrm{i}, \mathrm{j}$, self-feedback weight $w_s = 0.1$, tolerance limit $p = 30$, radius decreasing ratio $\gamma = 0.4$.

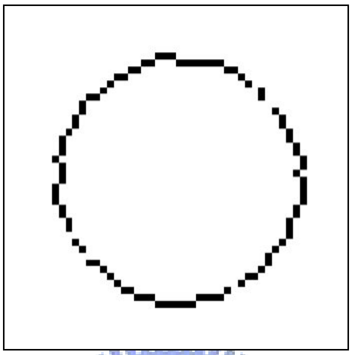
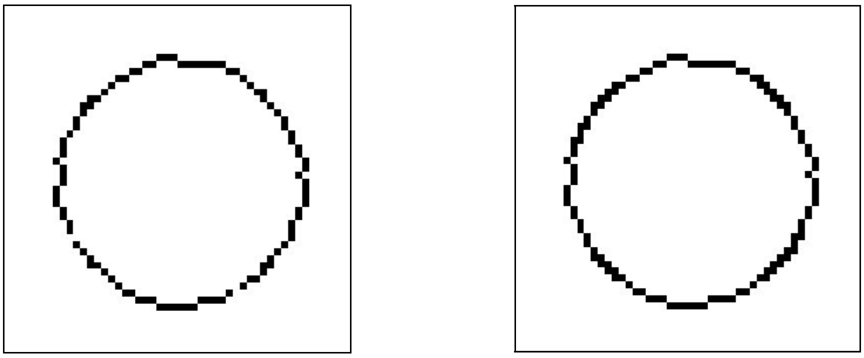
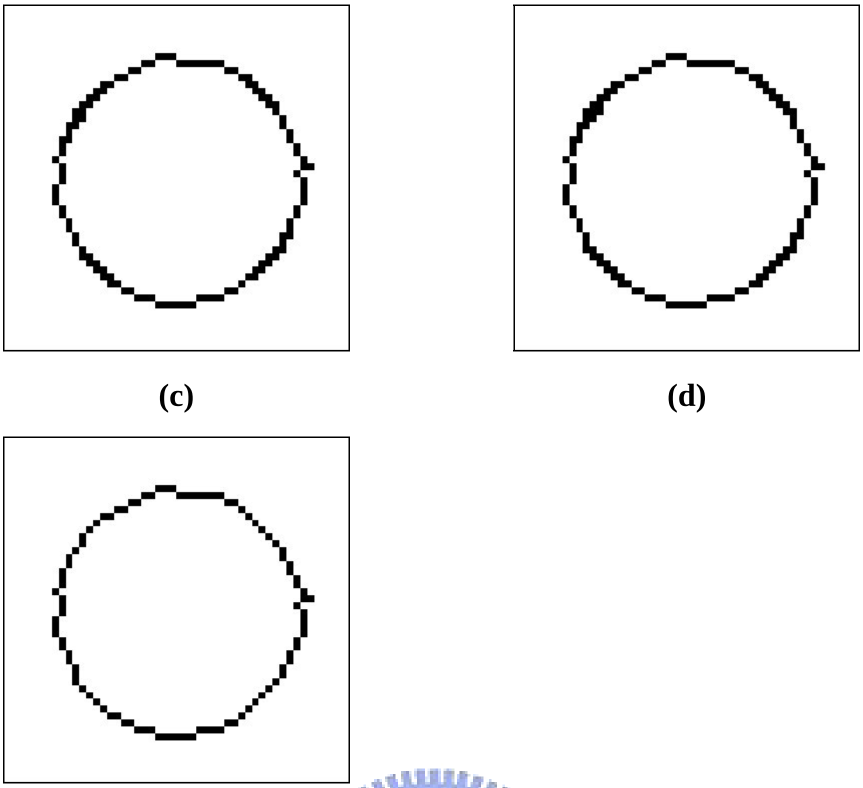


Figure 3.18. Input data of experiment I.

Figure 3.19 shows all the update process with time and result of thinning processing. The thinning result shows in Figure 3.19(e). Compare with original input and final result, the broken segment are linked well.



(a) **(b)**



(c)

(d)

(e)

Figure 3.19.Update process and result of thinning processing (a) $t=1$, (b) $t=2$, (c) $t=3$, (d) $t=4$, (e) thinning result of (d).

Experiment II: personal face

The input of second experiment is shown in Figure 3.20 with size 182×200. The parameters of the window size of linear regression is set to 5×5, $R(0)=2$, connection weight $w_{ji} = 0.5*(1/R(0)), \forall \mathrm{i}, \mathrm{j}$, self-feedback weight $w_s = 0.1$, tolerance limit $p=30$, radius decreasing ratio $\gamma = 0.4$.

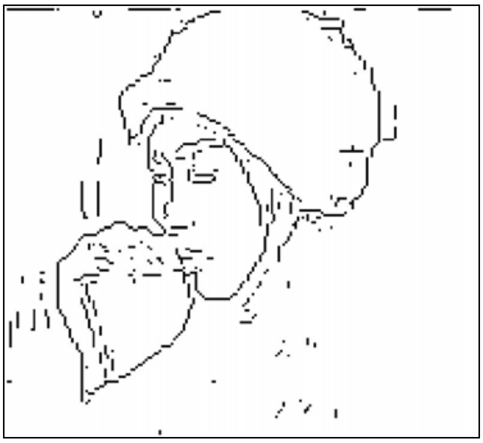
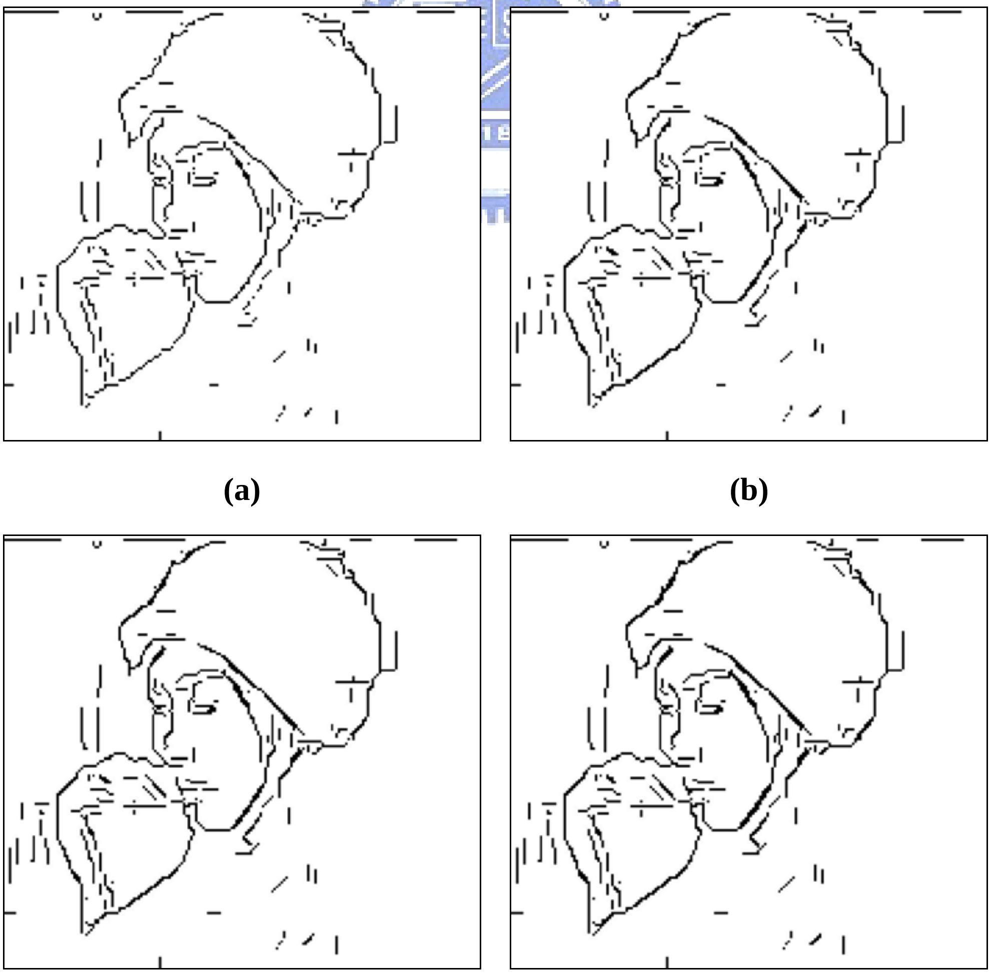


Figure 3.20. Input data of experiment II.

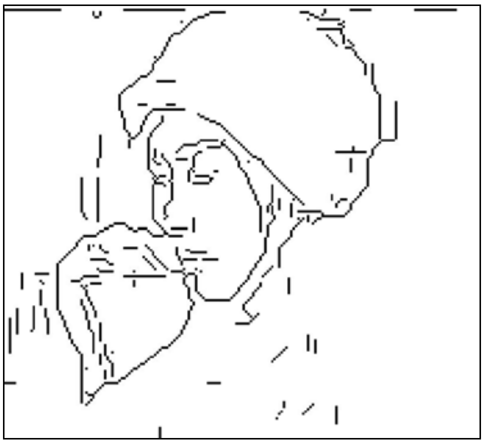
Figure 3.21 shows all the update process with time and result of thinning processing. The thinning result is shown in Figure 3.21(e). Compare with original input and final result, the broken segment are linked well. But the method has bad ability for noise elimination. That can remove one point noise only.



(a)

(b)

(c) **(d)**

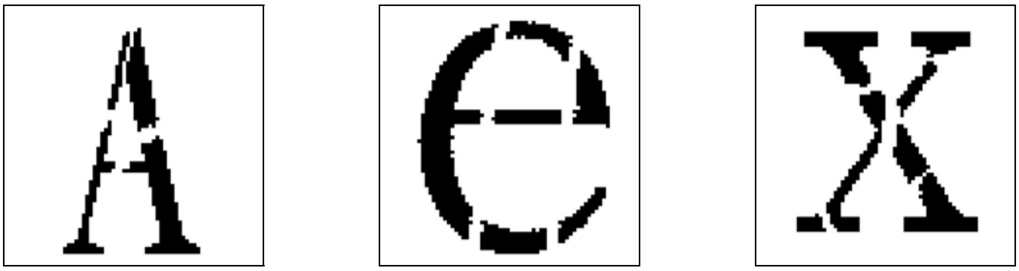


(e)

Figure 3.21.Update process and result of thinning processing (a) $t = 1$, (b) $t = 2$, (c) $t = 3$, (d) $t = 4$, (e) thinning result of (d).

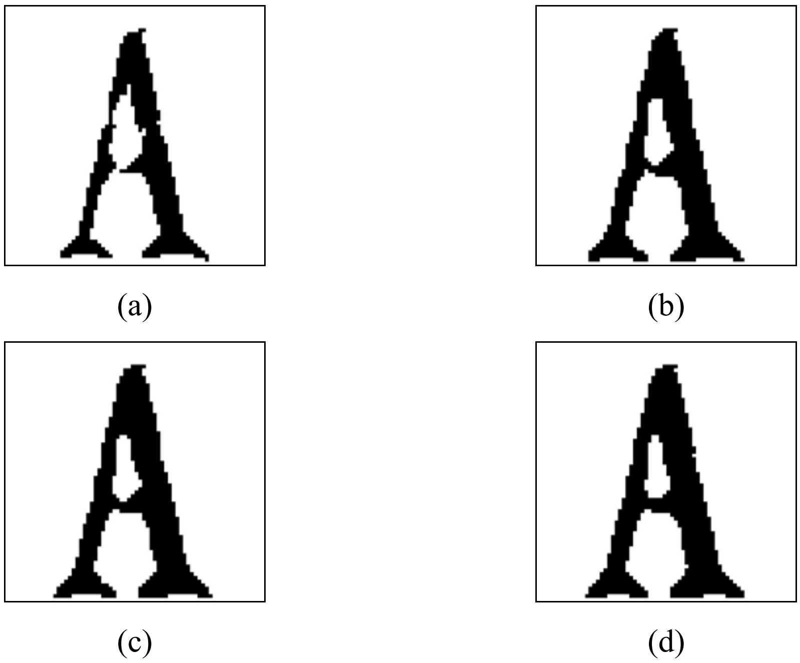
Experiment III: English characters

In the experiment III, we use three English characters for experiments. The input size is all of 70×70. The parameters of these experiments are chosen the same. The window size of linear regression is set to 9×9, $R(0) = 2$, connection weight $w_{ji} = 0.5 * (1/R(0)), \forall \mathrm{i}, \mathrm{j}$, self-feedback weight $w_s = 0.1$, tolerance limit $p = 3$, radius decreasing ratio $\gamma = 0.4$. Figure 3.22 shows three English characters as input data and Figure 3.23-25 shows the update process with time respectively.



(a) (b) (c)

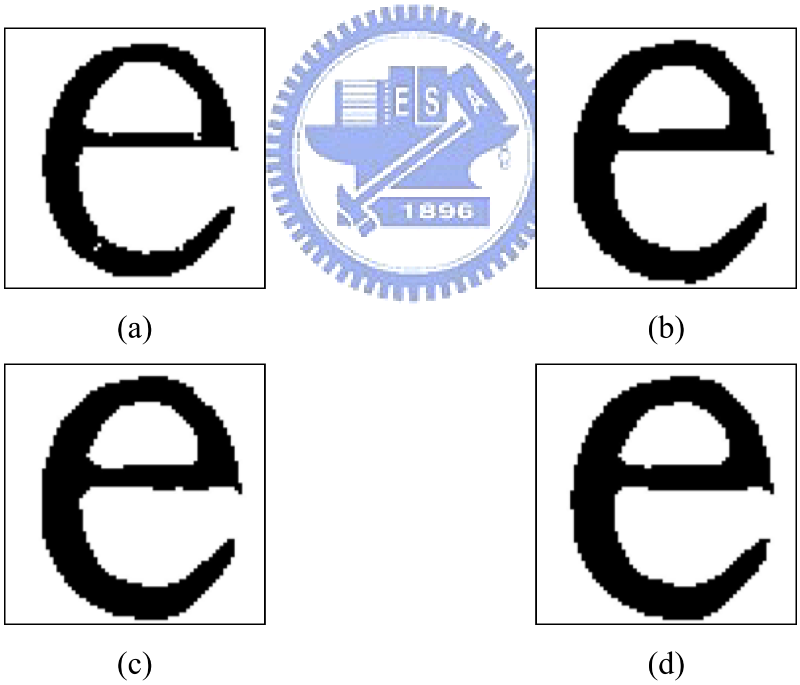
Figure 3.22. Three input characters for experiment III (a) character A, (b) character e, (c) character X.



(a) (b)

(c) (d)

Figure 3.23.Update process of character A (a) $t = 1$, (b) $t = 2$, (c) $t = 3$, (d) $t = 4$.

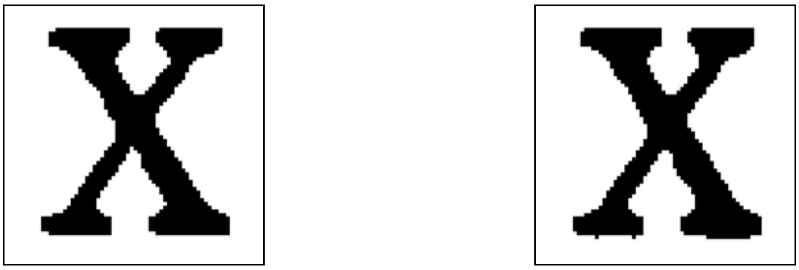


(a) (b)

(c) (d)

Figure 3.24.Update process of character e (a) $t = 1$, (b) $t = 2$, (c) $t = 3$, (d) $t = 4$.

(a) (b)



(c) (d)

Figure 3.25.Update process of character X (a) $t = 1$, (b) $t = 2$, (c) $t = 3$, (d) $t = 4$.