가변 링크에 의한 도형 패턴의 모서리 결정 방법
(A Vertices-Detecting Algorithm by the Variable Link for Patterns)

李 柱 根*, 朴 鍾 旭**
(Joo Keun Lee and Jong Wook Park)

要 約

본 논문은 도형 패턴에 대한 모서리를 결정하는 방법을 제안하였다. 도형 패턴의 윤곽선에 대한 라인 시그먼트(line segment)의 연결 상태로부터 후보 링크를 설정한다. 그리고 라인 시그먼트의 연결점으로부터 설정된 후보 링크에 대한 편차를 계산하여 모서리의 위치를 결정하는 알고리즘과 凹, 凸 구조의 모서리를 구분하여 식별하는 알고리즘을 제안한다. 그 결과 Freeman의 방법과 비교하여 훨씬 유연하고 효과적임을 확인하였다.

Abstract

A method for the detecting defined vertices is described in this paper. An alogrithm detecting vertex position by means of calculating an error and distinguishing the defined inner and outer corners by the chain code of line segments is described. To calculate the error, virtual links are detected by means of a jointing relations of line segments for a contour. And so an error is calculated by measuring a minimum distance from a jointing point between the line segments to virtual link. As a result, this method is more flexible and effective than conventional Freeman's method.

I. 序 論

도형 패턴의 처리에 있어서 에즈(edge) 픽셀(pixel)을 인식하거나 전송하는 방법은 데이타량이 방대해진다. 따라서 도형 패턴의 특징이 될 모서리(vertex)를 검출한다는 것은 인식과 전송에 있어서 매우 효과적이 된다. 그러나 도형 패턴에서 모서리를 결정하는 문제는 인간이 판단하는 것처럼 간단한 문제는 아니다. 이 문제에 대한 해결 방법으로 직접적인 방법과 간접적인 방법을 들 수 있다.

간접적인 방법으로는 여러 방법[6]~[8]이 있으나, 이들 방법중 스플리트(split), 머즈(merge) 알고리즘이 유력한 방법으로 평가되고 있다. 그러나 이 방법은 임의의 직선에 대한 픽셀 하나 하나에서의 편차를 계산하므로 계산이 복잡하고, 처리과정이 반복될 뿐만 아니라 윤곽선이외의 점에서도 모서리가 생길 염려가 있다.

직접적인 방법으로서는 백타에 의한 k-cosin C_{ak} 를 계산하여 맥시마(maxima)와 미니마(minima)를 검출한 방법[1, 2]과 ADS(angle detection sequential) 알고리즘[5] 등이 발표되었다. 이들 방법도 픽셀 하나 하나를 대상으로 처리하기 때문에 처리과정이 반복되고, 계산이 복잡하다. 가장 간단한 방법으로서는 체인 링크 L_i^s에 의한 인접된 두개의 픽셀에서의 각도차 δ_i^s

*正會員, 仁荷大學校 工科大學 電子工學科
(Dept. of Electronic Eng., Inha Univ.)
**正會員, 圓光大學校 工科大學 電子工學科
(Dept. of Electronic Eng., Won Kwang Univ.)
接受日字 : 1983年 1月 27日

를 구하고, δ_i^*에 의하여 코너리티(cornerity)를 계산하여 모서리를 결정하는 방법[3]이 있다. 이 방법도 픽셀 하나 하나를 대상으로 하고, 체인 링크 L_i^*에 대한 기준(threshold)치 S, △를 일정한 값으로 하기 때문에 한 도형 패턴내에서도 형태에 따라 부분적으로 기준치를 변경시켜야 하는 결점이 있다.

이들 직접방법과 간접방법의 공통점은 픽셀을 중심으로 하여 각도를 계산하거나 또는 편차를 계산하여 모서리를 결정하므로 반복 처리되고 계산이 복잡하다.

본 논문에서는 종래의 픽셀 하나 하나를 대상으로 하지 않고 수 개의 픽셀을 묶어서 하나의 라인 시그먼트로 하고, 라인 시그먼트의 연결 상태에 따라 모서리를 결정하는 새로운 방법을 제안한다.

본 방법은 라인 시그먼트의 연결 상태에 따라 후보 링크를 설정하고, 설정된 후보 링크와 라인 시그먼트의 연결점과의 편차를 계산하여 링크를 구하므로써 모서리를 결정한다. 또한 라인 시그먼트의 체인 코드에 의하여 凹모서리와 凸모서리를 간단히 구하는 방법도 고안하였다. 또한 본 방법의 효과를 평가하기 위하여 Freeman방법[3]과 비교한 결과 일반성을 가지며 유연하고 효과적임을 보였다.

Ⅱ. 링크 결정 알고리즘

종래에 도형 패턴의 윤곽선은 픽셀 하나 하나를 체인 코드로 표현[3]하기 때문에 점마다 반복 처리를 한다. 그러므로 처리가 복잡해지고, 처리 속도가 떨어진다.

본 논문에서는 픽셀 하나 하나를 대상으로 하지 않고, m개의 동일한 체인 코드를 가지는 연속 픽셀은 하나의 라인 시그먼트로 묶고 그것을 체인 코드로 정한다. 따라서 n개의 라인 시그먼트로 구성되는 윤곽선은 다음과 같은 식으로 표시할 수 있다.

$$Q_n = C_{i=1}^{n} S_i = s_1 s_2 s_3 \cdots\cdots s_n \qquad (1)$$

$$S_i = C_{i=1}^{m} P_i = p_1 p_2 p_3 \cdots\cdots p_m \qquad (2)$$

$$|(S_i), (P_i)| \in \{0, 1, 2, 3, 4, 5, 6, 7, |\} \qquad (3)$$

여기서 Q_n은 윤곽선, S_i는 라인 시그먼트이며 P_i는 픽셀이다. 또한 (S_i)와 (P_i)는 그림 1과 같은 체인 코드 매트릭스(matrix)에 의한 체인 코드이며 모듈로(modulo) 8이 되고, 도형의 윤곽선은 단순 폐회로 상에서 $S_i = S_{i \pm n}$이 된다.

또한 윤곽선을 그라프의 모서리(vertex) V_i와 호(arc) A_i로 표현하면 (4)식과 같이 표시할 수 있다.

$$Q_n = C_{i=1}^{k} (V_i, A_i) \qquad (4)$$

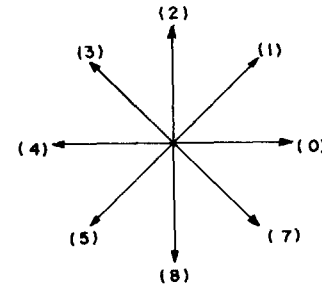


그림 1. 체인 코드 매트릭스
Fig. 1. Chain code matrix.

여기서 그림 2와 같이 호 A_i를 링크 l_i로 대치하고, k개의 링크 체인 L_i로 하면 (4)식은 (5)식으로 표현할 수 있다.

$$Q_n = C_{i=1}^{k} L_i = l_1 l_2 l_3 \cdots\cdots l_k \qquad (5)$$

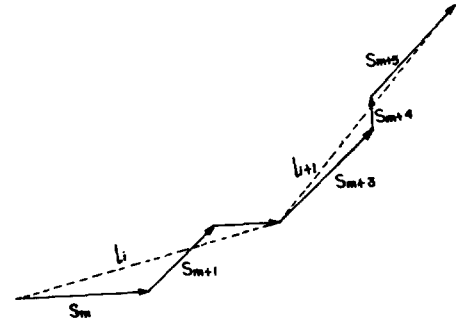


그림 2. 라인 시그먼트 S_m과 링크 l_i와의 연결 관계
Fig. 2. Joint relation of the line segment S_m and link l_i.

(4)식과 (5)식으로부터 두 개의 링크 사이의 연결점은 모서리 V_i가 되고, 반대로 두 개의 모서리를 연결한 직선은 링크가 된다는 것을 알 수 있다. 따라서 링크 l_i를 결정하므로써 모서리를 결정할 수 있다.

또한 (1)식과 (5)식에서 n=k일때 라인 시그먼트 S_n은 링크 l_k와 동일하게 되어 라인 시그먼트의 연결점을 모서리로써 결정할 수 있다. 그러나 라인 시그먼트의 연결점마다 모서리가 정해지므로 모서리는 방대한 량이 된다. 따라서 q개의 라인 시그먼트의 부분체인(subchain) $s_1 s_2 \cdots\cdots s_q$를 하나의 링크 l_i로 하면 (6)식과 같이 표현된다.

$$l_i = C_{i=1}^{q} S_i = s_1 s_2 s_3 \cdots\cdots s_q \qquad (6)$$

$$(1 \leq q \leq n)$$

그러므로 링크 l_i를 결정하면 l_i 양단의 모서리가 결정된다. 따라서 q개의 라인 시그먼트 체인에 대한 (q+1)개의 연결점을 두 개의 모서리로써 감소시킬 수 있다.

그러나 링크를 결정하기 위해서는 링크의 시작점과 끝점을 결정해야 한다.

그런데 (5)식의 Q_n은 k개의 링크 체인으로 표현하였으므로 링크 l_i의 시작점은 링크 l_{i-1}의 끝점이 된다. 그러므로 링크 l_i의 끝점만 결정하여도 링크 l_i를 구할 수 있다.

링크 l_i의 끝점을 결정하기 위한 윤곽선에 대한 라인 시그먼트의 연결 상태는 다음 네 가지 종류로 분류된다. 이 네 가지 성질로부터 링크 l_i의 끝점을 결정할 수 있다.

1) $(s_k)=(s_{k+2})$, $(s_{k+1})=(s_{k+3})$, ……, $(s_{k+2l-2})=(s_{k+2l})$, $(s_{k+2l-1})\neq(s_{k+2l+1})$, $(s_{k+2l})\neq(s_{k+2l+2})$

2) $(s_k)=(s_{k+2})$, $(s_{k+1})=(s_{k+3})$, ……, $(s_{k+2l-1})\neq(s_{k+2l+1})$, $(s_{k+2l})=(s_{k+2l+2})$, ……, $(s_{k+2l+2j-2})=(s_{k+2l+2j})$, $(s_{k+2l+2j-1})\neq(s_{k+2l+2j+1})$, $(s_{k+2l+2j})\neq(s_{k+2l+2j+2})$

3) $(s_k)\neq(s_{k+2})$, $(s_k)=(s_{k+3})$, $(s_{k+2})\neq(s_{k+4})$, $(s_{k+3})\neq(s_{k+5})$

4) $(s_k)\neq(s_{k+2})$, $(s_k)\neq(s_{k+3})$, $(s_{k+1})\neq(s_{k+3})$ (7)

여기서 (s_k)는 라인 시그먼트 s_k의 체인 코드이며 이들 관계는 그림 3과 같이 표시한다.

그림 3 (a)에서 $(s_{k+3})=(s_{k+5})$, $(s_{k+4})=(s_{k+6})$, $(s_{k+5})\neq(s_{k+7})$, $(s_{k+6})\neq(s_{k+8})$이므로 s_{k+5}는 s_{k+3}과, s_{k+6}은 s_{k+4}와 진행 방향이 각각 같고, s_{k+7}부터는 진행 방향이 달라진다. 따라서 라인 시그먼트 s_k부터 추적하여 4개의 서로 다른 라인 시그먼트의 체인 코드가 존재할 때까지 추적한다. 즉 라인 시그먼트 s_k의 시작점으로부터 4개의 라인 시그먼트를 수색하여 두 개의 동일한 방향의 라인 시그먼트가 존재할 때 처음의 s_k를 버리고 다시 s_{k+1}부터 같은 방법으로 반복한다. 이때 m번째 라인 시그먼트로부터 서로 다른 4개의 체인 코드가 존재할 때 형성되는 링크를 결정하기 위하여 다음과 같이 정의 1, 2를 세운다.

정의 1. m번째 라인 시그먼트로부터 서로 다른 4개의 체인 코드가 존재할 때 형성되는 링크를 후보 링크 l_i'로 한다.

정의 2. 후보 링크 l_i'의 끝점을 라인 시그먼트 s_{k+m+2}의 시작점으로 한다.

예 : 그림4에서 체인 코드는 $(s_1)=(s_3)$, $(s_2)\neq(s_4)$,

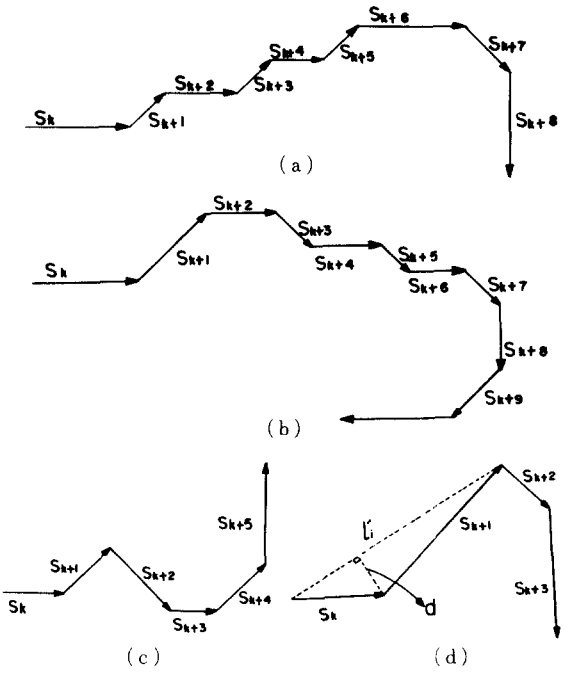


그림 3. 라인 시그먼트의 연결 상태에 대한 예
(a) 1경우 (b) 2경우 (c) 3경우 (d) 4경우

Fig. 3. Example of line segment's joint relation.
(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

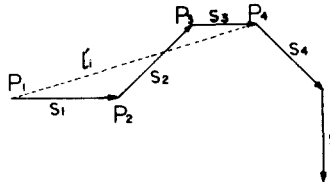


그림 4. 가상 링크 l_i'의 결정 예

Fig. 4. Example of detected virtual link l_i'.

$(s_3)\neq(s_5)$이므로 정의 1, 2로부터 끝점 p_4까지의 직선은 후보 링크 l_i'가 된다.

그러나 그림 3 (d)와 같이 라인 시그먼트 s_k와 s_{k+1}의 길이가 길어지면 s_k와 s_{k+1}의 연결점으로부터 후보 링크 l_i'까지의 수직거리 d가 증가하므로 모서리가 잘못 결정될 염려가 있다. 따라서 후보 링크 l_i'가 타당한 링크인가를 판단하기 위하여 수직거리를 계산하여 판단할 수 있다. 이를 위하여 편차 E_i를 구하는 방법[6]을 도입하면 다음과 같다.

$$x \cdot \sin\rho + y \cos\rho = d$$

$$E_i = |x_i \cdot \sin\rho + y_i \cos\rho - d| \quad (8)$$

(8)식에서 Pavlidis는 임의의 점 $P(x_t, y_t)$로부터 임의의 직선 l까지의 수직거리를 계산하여 편차 E_t를구하였다. 그러나 본 논문에서는 l의 x축, y축과의 교점을 후보 링크 l_i'의 시작점과 끝점으로 정의한다. 그리고 후보 링크를 구성하는 각 라인 시그먼트의 연결점들로부터 편차 E_i'를 구한다. 이렇게 함으로써 보다 간단하게 후보 링크 l_i'가 타당한 링크인가를 판단할 수 있다. 즉 q개의 라인 시그먼트의 연결점으로부터 l_i' 에 수직인 q개의 편차를 구할 수 있다. 여기서 q개의 편차에 대한 최대치를 E_m, 편차의 기준치를 E_t라하면 $E_m > E_t$일 때 E_m을 갖는 연결점 v_i를 끝점으로 하면 링크 l_i를 다시 설정하여 그의 편차를 평가한다. 예를 들면 그림5에서 E_{i+3}'가 편차의 최대치 E_m이 되고, $E_{i+3}' > E_t$이면 v_i가 끝점이 되는 점선 l_i에 대한 편차를 구한다.

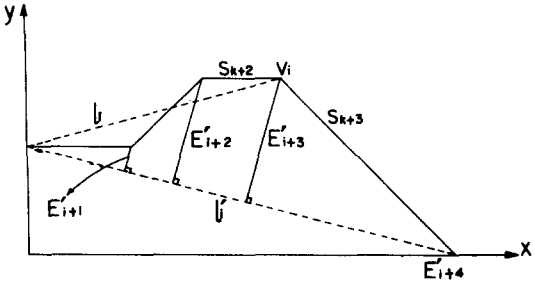


그림5. 링크 l_i'와 편차 E_i'와의 관계
Fig. 5. Link l_i' and error E_i'.

그러므로 링크 체인 L_i를 결정하기 위한 알고리즘 1을 다음과 같이 세울 수 있다.

〔알고리즘 1〕 링크 체인 결정 알고리즘

입력 : 라인 시그먼트의 시작점 S(K)와 체인 코드 C(K)에 대한 리스트

출력 : 링크의 끝점 L(i), 라인 시그먼트의 체인 코드 CS(i), CT(i)

1단계 : i←1 : K←1 : LS(i)←S(K) : CS(i)←C(K)

2단계 : 블럭 10 실행

10.1 블럭 10 시작

10.2 M←K

10.3 C(K)=C(K+2)이면 K←K+1 : 10.3으로 이동

10.4 K←K+1 : C(K)=C(K+2)이면 K←K+1 : 10.3 으로 이동

10.5 C(K−1)=C(K+3)이면 K←K+2 : 10.3으로 이동

10.6 N←K+1 : 10.7로 이동

10.7 블럭 11 실행

11.1 블럭 11 시작

11.2 J←M : L←N

3 K=J부터 L까지 블럭 12 실행

12.1 블럭 12 시작

2 E_m=0 : H=0

3 편차 E(K)계산

4 E(K)>E_t이고 E(K)가 최대치이면 H←K : E_m←E(K)

5 K=K+1 : 12.3으로 이동

11.4 E_m=0이면 LT(i)←S(L) : CT(i)←C(L) : CS(i)←C(L−1) 11.6으로 이동

11.5 L←H : 11.3으로 이동

6 L=N이면 i←i+1 : K←N : 11.8로 이동

7 i←i+1 : J←K : L←N : 11.3으로 이동

8 블럭 11 끝

3단계 : 입력 리스트가 끝나면 터미네이트(terminate)

4단계 : 2단계로 이동

Ⅲ. 모서리 결정 알고리즘

패턴의 윤곽선에 대한 링크 체인 리스트(link chain list)가 결정되었으므로 링크 체인의 연결 관계로부터 2개의 링크 l_k와 l_{k+1}의 연결점을 모서리로 정하고, l_k와 l_{k+1}이 이루는 각 α_v로써 凹, 凸의 모서리를 다음과 같이 정의한다.

정의2. 凹모서리 : $\alpha_v < 180°$일 때 링크 l_k와 l_{k+1}의 연결점으로 한다.

凸모서리 : $\alpha_v > 180°$일 때 링크 l_k와 l_{k+1}의 연결점으로 한다.

정의된 凹, 凸의 모서리를 결정하는 방법은 2개의 링크 l_k와 l_{k+1}을 벡타로 하여 두 벡타의 내적을 구하여 결정할 수 있으나 계산이 복잡하다. 따라서 본 논문에서는 라인 시그먼트에 대한 체인 코드를 사용하여 간단히 결정할 수 있는 방법을 제시한다.

그림6에서 링크 l_i와 l_{i+1}의 연결점 v_i는 라인 시그먼트 s_k와 s_{k+1}의 연결점과 동일하다. 또한 라인 시그먼트 s_{k-1}과 s_k와의 연결점과, 링크 l_i와의 편차 E_i <E_t이고, S_k의 길이 $|s_k| \gg E_i$라 할 때 s_i는 (10)식으로 표현할 수 있다. 그러므로 링크 l_i와 라인 시그먼트 s_k가 이루는 각은 $\delta_i \ll 180°$이고, l_{i+1}과 s_{k+1} 이

$$\delta_i = \tan^{-1}\frac{E_i}{\sqrt{|s_k|^2 - E_i^2}} \approx \tan^{-1}\frac{E_i}{|s_k|} \qquad (10)$$

이루는 각은 $\delta_{i+1} \ll 180°$로 볼 수 있다. 따라서 凹,凸의 모서리를 l_i와 l_{i+1}이 이루는 각 α_v 대신 s_k와 s_{k+1}이 이루는 각으로 결정하여도 무방할 것이다.

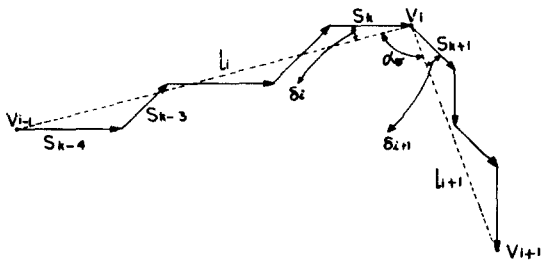


그림 6. 모서리 결정 예

Fig. 6. Example of detected vertices.

그런데 체인 코드(s_k)는 (3)식을 만족하므로 (s_k)와 (s_{k+1})의 차를 구하면 라인 시그먼트 s_k와 s_{k+1}이 이루는 각이 된다. 예를 들면 $(s_k)=0$이고, $(s_{k+1})=2$일 때 $(s_{k+1})-(s_k)=2$ 되어 s_k와 s_{k+1}이 이루는 각은 90도가 됨을 알 수 있다.

따라서 (s_k)와 (s_{k+1})로써 l_i와 l_{i+1}에 의하여 생성되는 모서리 v_i를 결정하는 알고리즘을 다음과 같이 설정할 수 있다.

〔알고리즘 2〕 모서리 결정 알고리즘

입력 : 〔알고리즘 1〕의 출력

출력 : 모서리 종류 VS(i)와 위치 V(i)

1 단계 : i←1 : LC←CT(i)+4 : LC≧8 이면 LC←(LC−8)

2 단계 : CS(i)<LC이면 4 단계로 이동

3 단계 : 凸 모서리로 출력 : 5 단계로 이동

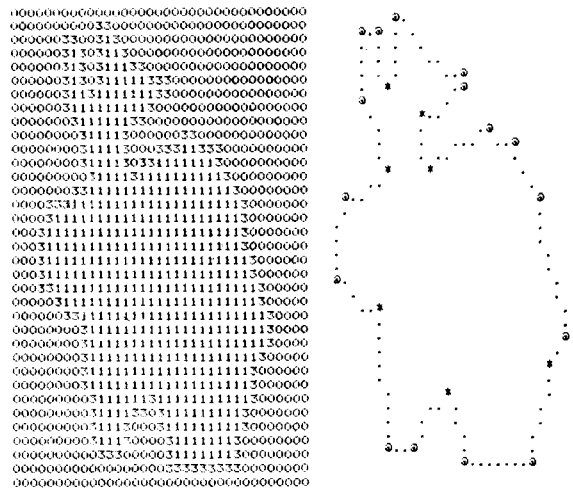
4 단계 : 凹 모서리로 출력

5 단계 : 입력 리스트가 끝나면 끝냄.

6 단계 : i←(i+1) : 2 단계로 이동

이상과 같은 알고리즘 1, 2에 대한 타당성을 보이기 위하여 프로그램을 작성하고 그림 7(a)의 입력 패턴을 처리한 결과, 그림 7(b)와 같다. 여기서 "*"는 凹 모서리이고 "a"는 凸 모서리이며 "•"는 윤곽선을 나타낸다. 또한 부록 표A, B에 모서리와 라인 시그먼트 리스트를 각각 보였다.

그림 7(b)에서 주목되는 것은 도형의 윤곽선 구조를 오목과 볼록으로 구분하는 凹, 凸 모서리로 각각 분리되며 모서리 위치가 정확히 잡히고, 또 극히 세부적인 부분까지 결정된다. 따라서 이들 모서리에 의한 오목과 볼록의 성질을 세이프 패턴(shape pattern)에 이용하면 효율적인 인식을 가능케 한다.



(a) (b)

그림 7. 결정된 모서리에 대한 대
(a)입력패턴 (b)결정된 모서리($E_t=1$)

Fig. 7. Example of detected vertices (a) Input pattern (b) Detected vertices ($E_t=1$).

Ⅳ. 실험 결과 및 검토

본 방법의 알고리즘에 대하여 도형 패턴과 문자 패턴을 대상으로 애플Ⅱ(Apple Ⅱ) 마이크로컴퓨터로써 베이직(basic) 언어를 사용하고, MI. X, Y플로터(plotter)로써 출력하였다. 입력 패턴은 Freeman[3]의 도형 패턴과 Feng의 한자 패턴[9]을 인용하고 한글 패턴을 추가하였다. 그러나 입력 방법은 애플Ⅱ의 화상양자화 인터페이스를 이용하여 2치레벨로 변환하여 실행하였다. 동일한 조건에서 비교하기 위하여 전처리 과정에서 노이즈(noise)를 가급적 제거하여 Freeman과 똑같은 패턴을 만들어서 표본으로 하였다.

Freeman의 방법을 비교한 이유는 그의 발표이후 별다른 발표가 없고, 종래의 방법중 가장 간단한 방법으로 알려져 있다.[3],[10,11] 여기서 Freeman의 방법[3]은 (11)식으로 표현되는 코너리티(cornerity)를 계산하여 모서리를 결정한다.

$$G_j=\sqrt{t_2}\times\sum_{i=j}^{i+s}\delta_i^s\times\sqrt{t_2} \qquad (11)$$

$$t_1=\max\{t:\delta_{j-v}^s\in(-\triangle,\triangle),\ \forall\ 1\leqq v\leqq t\}$$

$$t_2=\max\{t:\delta_{i+s+v}^s\in(-\triangle,\triangle),\ \forall\ 1\leqq v\leqq t\}$$

따라서 Freeman방법과 본 방법을 비교하고, 비교한 결과는 그림8, 9와 같다. 여기서 점선은 윤곽선, 링크

는 실선으로 표시하고, 그림8(a), (b), (c)와 그림 9(a), (b), (c)의 "△"는 Freeman방법에 대한 모서리의 위치를 표시하고 그림8(d), (e), (f), 그림 9(d), (e), (f), 그림10의 "⋈"와 "◇"는 본 방법에 의한 凹 모서리와 凸 모서리를 각각 나타낸다.

Freeman방법은 그림8(b), (c)와 같이 비교적 단순한 도형에서는 거의 유사하게 모서리가 잡히지만 그림8(a)와 같이 부분적으로 세밀한 완곡부분 "A"에서는 모서리가 잡히지 않았다. 또 그림8(a)에서, "B," "C"부분에서는 링크가 윤곽선의 중심을 지나가지 않는다는 것을 볼 수 있으며 "D"부분에서는 필요이상으로 많은 모서리가 결정되어 처리 속도를 저하시키는 요인이 된다. 또한 그림8(c)에서 파라메타 S, △의 값을 그림8(b)와 같은 값으로 택하였을 때 "—·—·" 선으로 표시한 부분에서 링크가 발생하여 도형의 상부가 잘리는 경우가 생긴다. 따라서 모서리 결정에 있어서 S, △의 값에 따라 대단히 민감하다고 볼 수 있다. 그러므로 패턴의 종류에 따라 S, △의 파라메타값을 변경하여야 하는 결점이 있다. 더구나 문자 패턴에서는 매우 적합치 않다는 것을 보여 주고 있다. 즉 그림9(a), (b), (c)에서 작은 내부홀 "B"에 대한 모서리가 결정되지 않았으므로 "ㅂ" 패턴이 형성되지 않고, "C", "D", "E", "F"의 부분에서도 모서리가 잡히지 않으므로 탈락되든가, 변형되는 부분이 발생한다. 그러므로 Freeman방법은 문자 패턴에 대하여 주

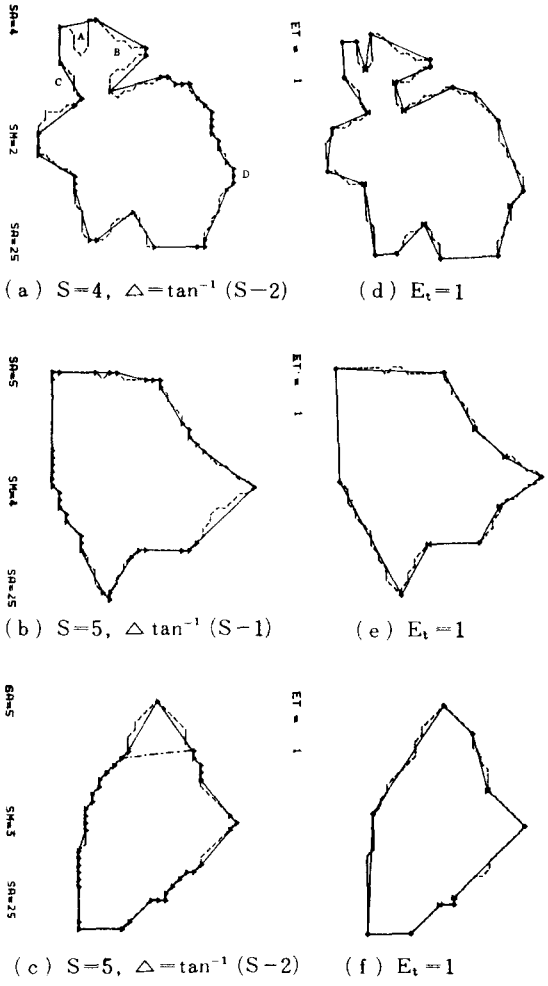


그림 8. 도형 패턴의 모서리 결정에 대한 비교 (a), (b), (c) Freeman방법 (d), (e), (f) 본 방법

Fig. 8. Comaparing detected vertices of Freeman's method with this method for patterns. (a), (b), (c) Freeman's method (d), (e), (f) this method.

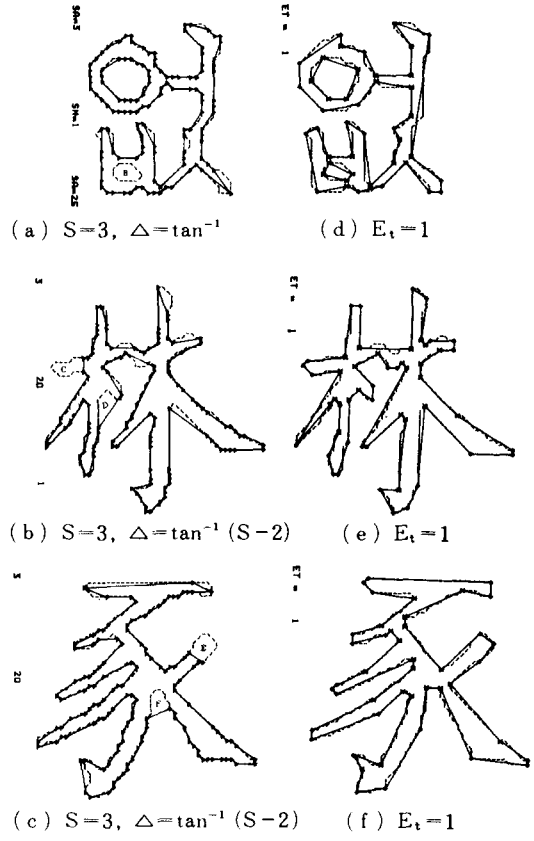


그림 9. 문자 패턴의 모서리 결정에 대한 비교 (a), (b), (c) Freeman방법 (d), (e), (f) 본 방

Fig. 9. Comparing detected vertices of Freeman's method with this method for character patterns. (a), (b), (c) Freeman's method (d), (e), (f) this method.

어진 패턴내의 일부분마다 S와 △값을 수시로 변형하지 않으면 안된다고 보겠다.

이에 반하여 본 방법의 결과는 그림8(d), (e), (f)와 그림9(d), (e), (f)에서 보는 바와 같이 $E_t=1$의 일정한 값으로 모서리를 결정할 수 있을 뿐만 아니라 패턴의 종류에 관계없이 凸, 凹 모서리가 정확히 결정됨을 보이고 있다. 또 Freeman방식은 모서리 결정에 있어서 두 개의 파라메타 S, △에 의하여 결정하나 본 시스템에서는 E_t하나로써 결정되고, 처리시간도 Freeman 방법에 비하여 약 2/3로 절감되었다.

그리고 표1에 그림10(a)의 한자 패턴과 그림10(c)의 도형 패턴의 E_t에 대한 결정된 모서리 수를 보여준다. 표1에서 편차기준치 E_t가 작은 값일수록 윤곽선에 가깝지만 모서리 수가 많아지고, 반면 E_t가 큰 값일수록 모서리수는 적어지지만 윤곽선과의 편차가 크다는 것을 보여 준다.

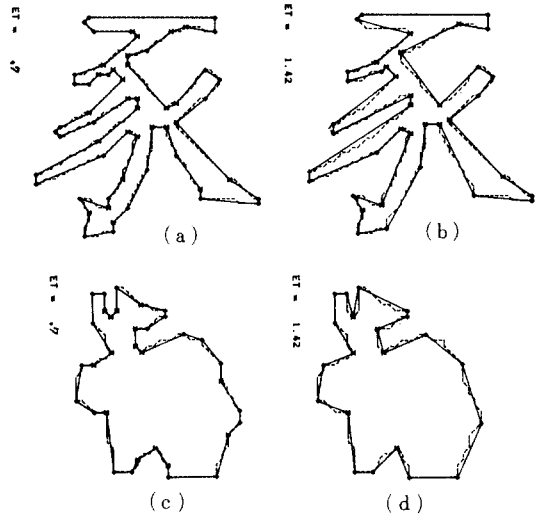


그림10. 본 방법에 대한 모서리 결정예
(a), (c) $E_t=0.7$ (b), (d) $E_t=1.42$

Fig.10. Example of detected vertices of this method for patterns.
(a), (c) $E_t=0.7$ (b), (d) $E_t=1.42$.

표 1. E_t에 대한 결정된 모서리수

Table 1. Detected vertex number of E_t.

E_t \ 패턴	모서리수 A (도 형)	B (문 자)
0	n	n
0.7	34	58
1	23	44
$\sqrt{2}$	21	39

V. 結 論

이상에서와 같이 본 논문을 고찰할 때 다음과 같은 결론을 내릴 수 있다.

1) 파라메타 E_t하나로써 적당한 모서리를 결정할 수 있다.
2) 최소의 모서리로써 도형 패턴의 원형이 구성되므로 통신에 이용할 때 전송 효과가 높아진다.
3) 윤곽선의 구조를 凹, 凸 모서리에 의하여 오목과 볼록으로 구분할 수 있고 Freeman방법에 비하여 처리 시간이 약 2/3로 절감되므로, 세이프 패턴 인식에 있어서 효과적이다.
4) 패턴 종류에 관계없이 이용 분야에 따라 적당한 편차기준치 E_t를 자유로 선택하므로써 효율적인 시스템의 구성을 가능케 한다.
5) 그러나 이용 분야에 따라 적당한 E_t를 선택하여 효율적인 실제의 시스템을 구성하는 문제와 시스템을 구성하였을 때 적당한 처리 시간 측정 문제는 앞으로 좀더 연구되어야 하겠다.

附 錄

표 A. 그림8(a)에 대한 결정된 모서리 표

Table A. Detected vertex list of Fig. 8 (a).

<VERTEX LIST>
****

< ET = 1 >

<VX>	<VY>	<VERTEX>
19	6	5
19	7	5
14	9	4
15	13	4
22	10	5
25	11	5
28	15	5
31	25	5
29	27	4
27	34	5
19	34	5
17	29	4
13	33	5
10	33	5
9	23	4
4	21	5
5	15	5
10	13	4
7	8	5
7	3	5
9	3	5
10	7	4
11	2	5

표 B. 그림 8 (a)에 대한 라인 시그먼트 표

Table B. Line segment list of Fig. 8 (a).

LINE SEGMENT LIST

<CH.C.>	<LX>	<LY>	<LEN.>	<ERROR>
0	11	2	1	0
7	12	2	3	.447213596
0	15	5	1	.894427191
7	16	5	1	.447213596
0	17	6	2	.894427191
6	19	6	1	0
4	19	7	1	0
5	18	7	2	.371390676
4	16	9	1	.742781352
5	15	9	1	.371390676
6	14	9	2	0
7	14	12	1	.727606875
1	15	13	1	0
0	16	12	1	.525225731
1	17	12	1	.131306433
0	18	11	2	.656532164
1	20	11	1	.131306433
0	21	10	1	.393919298
7	22	10	1	0
0	23	11	2	.632455532
6	25	11	1	0
7	25	12	3	.6
6	28	15	3	0
7	28	18	1	.862043656
6	29	19	1	.191565257
7	29	20	1	.478913142
6	30	21	1	.191565257
7	30	22	1	.095782628 3
6	31	23	2	.574695771
5	31	25	2	0
6	29	27	2	0
5	29	29	1	.549442255
6	28	30	1	.137360564
5	28	31	1	.137360564
6	27	32	2	.549442255
4	27	34	8	0
2	19	34	2	0
3	19	32	1	.742781352
2	18	31	1	.185695338
3	18	30	1	.557086014
5	17	29	1	0
4	16	30	1	0
5	15	30	1	.707106781
6	14	31	1	.707106781
5	14	32	1	0
4	13	33	2	0
3	11	33	1	0
2	10	33	3	0
3	10	29	1	.398014876
2	9	28	4	.497518595
3	9	24	1	.099503718 8
4	9	23	1	0
3	7	23	2	.742781352
4	5	21	1	.371390676
2	4	21	4	0
1	4	17	1	.657595949
2	5	16	1	.164398987
0	5	15	2	0
1	7	15	1	.742781352
0	8	14	1	.185695338
1	9	14	1	.557086014
3	10	13	1	0
2	9	12	2	.34299717
3	9	10	2	.68599434
2	7	8	5	0
0	7	3	1	0
7	8	3	1	0
6	9	3	2	0
7	9	6	1	.727606875
1	10	7	1	0
2	11	6	4	.78446454

參 考 文 獻

[1] A. Rosenfeld & E.Johnston, "Angle detection on digital curves," *IEEE Trans. Comput.*, vol. C-22, pp. 875-878, Sept. 1973.

[2] A. Rosenfeld & J.S. Weszka, "An improved method of angle detection on digital curves," *IEEE Trans. Compt.*, vol.C-24, pp. 940-942, Sept. 1975.

[3] H. Freeman & L.S. Davis, "A corner-finding algorithm for chain-coded curves," *IEEE Trans. Comput.*, vol. C-26, pp. 297-303, Mar. 1977.

[4] H. Freeman, "Computer processing of line-drawing images," *Computing Suveys*, vol. 6, pp. 57-97, Mar. 1974.

[5] L.S. Davis, "Understanding shape: angle and sides," *IEEE Trans. Compt.*, vol. C-26, no.3, Mar. 1977.

[6] T. Pavlidis & S.L.Horowitz, "Segmentation of plane curves," *IEEE Trans. Compt.*, vol. C-23, pp. 860-870, Aug. 1974.

[7] T.Pavlidis, "Waveform segmentation through functional approximation," *IEEE Trans. Compt.*, vol. C-22, pp.689-697, 1973.

[8] T.Pavlidis & S.Horowitz, "Piece wise approximation of plane curves," *Proc. IJCPR*, vol. 1, pp.396-405, 1973.

[9] F.Feng & T.Pavlidis, "Decomposition of polygons into simpler component: Feature generation for syntactice pattern recognition," *IEEE Trans. Comput*, vol. C-24, pp. 636-650, June 1975.

[10] L.S. Davis, "Shape matching using relation techniques," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 1, Jan. 1979.

[11] T.Pavlidis, "Hicrarchies in structured pattern recognition," *Proc. of the IEEE*, vol. 67, no. 5, May 1979.