구조해석에 의한 디지탈 곡선의 분리

(Segmentation of Digital Curves by Structural Analysis)

柳 承 必*, 權 五 錫**, 金 太 均**

(Sung Pil Lyu, O Sok Kwon, and Tae Kyun Kim)

要 約

디지탈곡선을 다각형으로 근사화하는 기술은 화상분석이나, 데이타 압축에 효과적으로 사용될 수 있다.

S. Shilien은 디지탈곡선으로부터 H. Freeman이 제안한 구조적 특성을 만족하는 디지탈선분(Digital straight line segment)들로 분리하는 방법을 제안하였다. 그러나 디지탈선분은 이러한 구조적 특성을 만족하지 않은 경우가 많았으며, 이들은 위의 방법에 의하면 한개 이상의 breakpoint를 갖게 된다.

여기서는 디지탈직선과 디지탈선분의 구조적인 표현에 대해서 서술하고, 이를 이용하여 디지탈곡선을 디지탈선분으로 분리하는 알고리즘을 제시한다.

여기서 표현된 디지탈선분은 디지탈직선의 특성을 만족하는 것과 만족하지 않는 것을 모두 포함한다. 따라서 디지탈 직선의 특성을 만족하지 않는 디지탈선분을 포함하고 있는 디지탈 곡선에 대해서 본 방법은 S. Shilien의 방법에 비해 적은 break point수를 가지게 된다.

Abstract

Techniques for approximating digital curves by polygonal lines are a valuable tool for image analysis and data compression.

S. Shilien proposed a method for segmenting a digital curve into digital straight line segments which agree with the structural properties proposed by H. Freeman. However, there are lots of digital straight line segments which are not satisfied with the structural properties, and have more than one break point by Shilien's method.

Here, the structural representation of digital straight lines and digital staight line segments is described. And a method for segmenting a digital curve into digital straight line segments which may be not satisfied with the structural properties is proposed. The number of break points extracted by this method is less than that by S. Shilien's method from the digital curve which includes the digital straight line segments not satisfied with the structural properties.

*正會員, 韓國에너지硏究所 計測制御 硏究室
(Korea Advanced Energy Research Institute, IC Dept.)

**正會員, 忠南大學校 電子計算機工學科
(Dept. of Comput. Eng., Choongnam Nat'l Univ.)

接受日字 : 1989年 10月 10日

I. 서 론

디지탈곡선을 디지탈 선분들로 분리하는 방법은 화상분석, 패턴인식 또는 데이타 압축등에 효과적으로 사용될 수 있다. 이러한 방법들은 디지탈 곡선으로 부터 디지탈 선분을 분리하는 점(break point 또는 corner)의 수는 화상분석, 패턴인식 및 데이타 압

축의 효율적인 면에서 중요한 변수라 할 수 있다. 이러한 점에서 고려할 때 break point수는 가급적 원래의 형태를 유지하면서 그 수가 적은것이 유리하지만, 형태의 유지정도와 break point수는 서로 반비례하므로, Trade off가 필요하다.

디지탈 곡선을 다각형 근사화 하기 위해 거리 나 각도등을 이용해서 break point를 찾는 방법[7-12]들이 많이 제안되어 왔으나 Freeman[1]의 디지탈 직선의 특성에 대한 발표가 있은 이후, Shilien[3]은 디지탈 직선의 구조적인 특성을 이용하여 패턴인식에 의해, 정수의 연산 및 한화소에 대해 한번의 처리만으로 한 화소미만의 오차를 갖는 디지탈선분을 고속으로 추출하는 방법을 제시한 바 있다.

디지탈 직선의 구조적 특성에 관한 많은 연구[1,2]가 많이 있었는데, 그 특성은 체인코드(8 방향 체인코드)열이 있을 때,

(a) 두개이하의 체인코드로 구성되어 있고, 이들은 서로 이웃하는 체인코드이다.
(b) 두개의 체인코드열중 하나는 항상 단독으로 존재한다.
(c) 단독으로 존재하는 체인코드열은 가능한 균등한 분포를 갖는다.

를 만족하는 경우 이 체인코드열은 디지탈 직선으로 간주하였다. 한편 Brons[4]는 주어진 기울기를 가진 직선을 구조적인 특성을 가지는 체인코드열로 변환시키는 알고리듬(BA)을 제시한 바 있고, Arcelli와 Masarotti[6]는 BA에 의해서 생성된 디지탈 직선이 Rosenfeld[2]가 제안한 디지탈 직선의 특성을 만족함을 증명한바 있다. 한편, 이러한 디지탈 직선의 구조적인 특성을 이용하여, context-sensitive language 형태로 직선을 표현하거나, 생성하는 방법들[4,5]이 제안되어, 임의의 체인코드열이 주어진 실제 직선과 일치하는가를 검사할 수 있게 되었다. 그리고 Shilien[3]은 보다 나아가서, 구조적 패턴인식 방법을 이용하여 임의의 디지탈 곡선으로부터 디지탈 선분을 분리해내는 알고리듬을 제안하였다.

그러나, Shilien[3]이 제시한 방법은 디지탈 직선의 특성을 만족하는 디지탈 선분만을 추출함으로해서 많은 break point가 생성되고, 많은 break point 수는 이들을 정보로 이용하는 시스템의 효율을 떨어뜨릴 수 있다.

디지탈 선분은 시작과 끝점의 위치에 따라 디지탈 직선의 특성을 만족할 수도 있고, 만족하지 않을 수도 있다. 예를들면 그림 1(a), (b)는 모두 디지탈 선분으로 그림 1(a)는 그림 1(b)의 일부이다. 그런데 그림 1(a)는 디지탈 직선의 특성[1,2]을 만족하지 않고 그림 1(b)는 디지탈 직선의 특성을 만족하고 있다. 만약 그림 1(a)의 경우, 디지탈 직선의 특성을 만족하신 선분으로 그림과 분리하면 같이 A, B 두군데의 점(break point)을 기준으로 3개의 디지탈 선분들이 생기지만, 그림 1(b)의 경우는 하나의 디지탈 선분으로 된다.

실제의 디지탈 영상에서 그림 1(a)와 같이 불완전한 구조를 갖는 디지탈 선분이 많이 존재하는데, 이런 경우 Shilien[3]의 방법은 하나 이상의 break point를가지게된다. 만약 그림 1(a)과 같은 디지탈 선분의 일부인 경우, break point가 발생하지 않도록 한다면, 이와 같이 디지탈 직선의 특성[1,2]을 만족하지 않는 디지탈 선분을 포함하고 있는 디지탈 곡선으로 부터 추출되는 break point의 수는 보다 줄어들 것이다.

이 논문에서는 디지탈 직선의 특성[1,2]을 만족하지 않는 디자탈 선분이라도 이를 break point가 없는 하나의 디지탈 선분으로 처리하여 Shilien[3]의 방법에 비해 break point수를 줄이는 알고리듬을 제시한다.

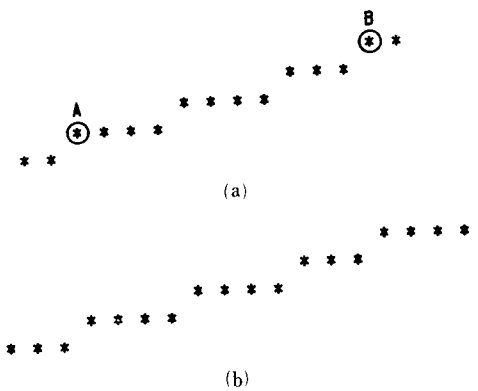


그림 1. 디지탈 선분
(a) 불완전한 구조를 가진 디지탈 선분
(b) 완전한 구조를 가지는 디지탈 선분

Fig. 1. Digital straight line segment.
(a) A digital straight line segment with incomplete structure.
(b) A digital straight line segment with complete structure.

II. 디지탈 직선 및 디지탈 선분의 표현

1. 디지탈 직선의 표현

이 절에서는 BA에 의해서 생성되는 디지탈 직선의 표현을 앞으로 제안할 알고리듬에 쉽게 사용할 수 있

도록 다음과 같은 정의 및 정리를 한다.

체인코드를 그림 2와 같이 0, 1, 2…, 7등의 양자화된 방향코드로 표현할 때,

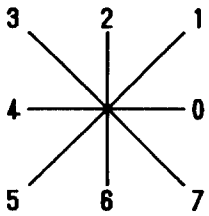


그림 2. 8방향 체인코드
Fig. 2. 8-directional chain code.

정의(1) 어떤 체인코드열의 종류 A가 있고 A가 n개 연결되어 하나의 체인코드열 X를 가 있을 때, X는 다음과 같이 표현한다.

$X=A\ A\cdots A$ 또는 $X=A*n$ 또는 $X=(A)*n$

이 때 만약 $n=0$ 이면 $X=\lambda$ 로 null string을 의미한다.

정의(2) 어떤 체인코드열의 종류 A〔0〕, A〔1〕, …, A〔n〕 그리고 B〔0〕, B〔1〕, …, B〔n〕이 있고, 이들이

$A[n]=A[n-1]*a[n-1]B[n-1]$

이고, $0<i<n$에 대해서,

$A[i]=A[i-1]*a[n-1]\ B[i-1]$

$B[i]=A[i-1]*b[i-1]\ B[i-1]$

(여기서 a[i-1], b[i-1]=0, 1, 2…으로 a[i-1]과 b[i-1]은 서로 연속되는 정수이며, A[0]와 B[0]는 8방향 체인코드 중의 하나로 서로 각도차이가 45이다.)라고 할 때 위의 A[n]과 같이 표현되는 모든 체인코드열에 대해,

i) A[n]이 연속적으로 무한히 반복되는 체인코드열을 L(A[n])이라 한다. 즉, $L(A[n])=A[n]*\infty$이다.

ii) $0\leq i\leq n$ 일때, A[i], B[i]를 각각 i레벨의 A-type 체인코드열, B-type 체인코드열이라 한다. 만약 $i=n$이면, B[i]=A[i]로 간주한다.

iii) A[i], B[i] 내의 A[i-1]이 run-length a[i-1] b[i-1] $(0<i\leq n)$를 각각 i레벨의 A-type run-length, B-type run-length라 한다.

iv) a[0], a[a]… a[i-1] > 0 이고, a[i]=0인 L(A[n])을 L(A[n], a[i]=0), b[0], b[1], …, b[j-1]>0이고, b[j]=0인 L(A[n])을 L(A[n], b[j]=0), a[0], a[1], …, a[k]>0인 L(A[n])을 L(A[n], a[k]>0), 그리고 b[0], b[1], …, b[l]>0인 L(A[n]))을 L(A[n], b[l]>0)으로 나타내며, 이들은 L(A[n], a[i]=0, b[j]=0), L(A[n], a[k]>0, b[j]=0), L(A[n], a[i]=0, b[l]>0), 또는 L(A[n], a[k]>0, b[l]>0)과같이 중복 표현할 수 있다.

v) L(A[n])으로 표현되는 모든 체인코드열의 집합을 T라 하고,

- L(A[n], a[n-1]>0, b[n-1]>0), $n\geq 1$
- L(A[0])

중 어느 하나를 만족하는 모든 체인코드열들의 집합을 T1이라 한다.

여기서, 집합 T1중에서, L(A〔0〕)의 경우 같은 체인코드가 계속되는 체인코드열이므로 디지탈 직선의 특성을 만족한다. 그리고 그이외의 원소들은 BA에 의해서 생성되는 구조적 직선과 같고, BA에 의해서 생성된 구조적직선은 디지탈 직선의 특성을 만족하므로,[6] T1의 원소는 모두 디지탈직선의 특성을 만족한다.

T의 임의의 원소인 L(A〔n〕) 내의 한 주기(A〔n〕)는 시작점의 이동에 따라 그 구조가 변형되는데, 여기서 이를 이용하여 T의 원소가 T1의 원소로 변형되는 것을 보이기위해 다음과 같은 정의 및 정리를 한다.

정의 3) C, D, E가 각각 어떤 체인코드열의 종류이고, C가 E의 내에 존재하는 체인코드열이라 할 때, 함수 SHIFTL, SHIFTR을 다음과 같이 정의한다.

$$\mathrm{SHIFTL}(C,D,E)=\begin{cases} DC';C=C'D\text{이고, C의 좌측에 D가 연결되어 있을 때} \\ \lambda:\text{그 외}\end{cases}$$

$$\mathrm{SHIFTR}(C,D,E)=\begin{cases} C'D;C=DC'\text{이고, C의 우측에 D가 연결되어 있을 때} \\ \lambda;\text{그 외}\end{cases}$$

정리 1) 임의의 L(A〔n〕) 내에서, 임의의 위치에 있는 A〔i〕 및 B〔i〕$(0\leq i\leq n)$에 대해 $A[i]=A'[i]X$이고, $B[i]=B'[i]X$인 체인코드열 종류 A′〔i〕, B〔i〕및 X가 존재하면, $i\leq j<n$에 대해

$\mathrm{SHIFTL}(A[j],X,L(A[n]))\neq\lambda$

$\mathrm{SHIFTL}(B[j],X,L(A[n]))\neq\lambda$

이다.

증명) 같은 레벨에서 체인코드열 종류는 A 또는 B-type 2가지만 존재하므로, 임의의 레벨에서 A-type

및 B-type 체인코드열의 좌측에는 반드시 A-type 또는 B-type 체인코드열중의 하나가 존재한다. 그리고 상위 레벨의 체인코드열은 정의1)에 의해서 반드시 하위 레벨의 B-type 체인코드열을 맨우측에 포함하고 있다. 따라서 위두식은 함수 SHIFTL의 정의에 의해서 성립한다.

정리2) 임의의 L(A[n])내에서, 임의의 위치에 있는 A[i], B[i] $(0 \leq i \leq n)$ 에 대해, A[i]=XA′[i]이고, B[i]=XB′[i]인 체인코드열의 종류 A′[i], B′[i]및 X가 존재할 때, $i \leq j < n$ 에 대해,

SHIFTR(A[j], X, L(A[n]))
SHIFTR(B[j], X, L(A[n]))

이다.

정리2)은 정리2)와 같은 방법으로 증명이 가능하므로 생략한다.

정리3) 임의의 L(A[n])내에서, 임의의 위치에 있는 A[i] 및 B[i] $(0 \leq i \leq n)$에 대해, A[i]=A′[i]X 이고, B[i]=B′[i]X인 체인코드열A′[i], B[ii] 및 X가 존재하고, $0 \leq i < j \leq n$인 모든 A[j], B[j]에 대해서

A′[j]=SHIFTL(A[j], X, L(A[n]))
B′[j]=SHIFTL(B[j], X, L(A[n]))

이라고 하면,

A′[j]=A′[j-1]∗a[j-1]B′[j-1]
B′[j]=A′[j-1]∗b[j-1]B′[j-1]
A′[j]=(SHIFTL(A[j-1], X, L(A[n])))∗a[j-1]
SHIFTL(B[j-1], X, L(A[n]))

이므로,

A′[j]=A′[j-1]∗a[j-1] B′[j-1]

가 된다. 한편 B′[j]에 대해서도 같은 방법으로 증명이 되므로 생략한다.

정리4) 임의의 L(A[n])내에서, 임의의 위치에 있는 A[i] 및 B[i] $(0 \leq i \leq n)$에 대해, A[i]=X, A′[i]이고, B[i]=X B′[i]인 체인코드열 A′[i], B′[i]및 X가 존재하고, $0 \leq i < j \leq = n$인 모든 A[j], B[j] 에 대해서

A′[j]=SHIFTR(A[j], X, L(A[n])

B′[j]=SHIFTR(B[j], X, L(A[n]))

이라고 일 때,

A′[j]=A′[j-1] B′[j-1]

B′[j]=A′[j-1]∗b[j-1] B′[j-1]

이 된다.

정리4)는 정리3)과 같은 방법으로 증명이 되므로 생략한다.

정리5) 임의의 T의 원소인 L(A[n], a[i]=0)에 대해서,

L(A[n], a[i])=L(A″[0])

이거나,

L(A[n], a[i]=0)=L(A″[n″]), a″[n″-1]>0), $n'' < n$

를 만족하는 T의 원소 L(A″[n″])이 존재한다.

증명)

이것을 증명하기 위해서,

(a) L(A[n], a[i]=0)=L(A″[0]),

(b) L(A[n], a[i]=0)=L(A″[n″], a″[n″-1]<0), $n'' < n$, 또는,

(c) L(A[n], a[i]=0)=L(A″[n″], a″[i″]=0), $n'' < n$

중에서 반드시 하나를 만족하는 L(A″[n″])가 존재하는 것을 보인다. 여기서 c)의 경우, 우측항은 좌측항에 비해 레벨수는 작고(n″<n), 우측항은 좌측항과 같은 형태를 가지므로, 좌측항에 대해 다시 (a), (b), (c)와 같은 변환을 거치면 최종적으로 L(A[n], a[i]=0)는 (a) 또는 (b)를 만족하는 L(A″[n″])로 변환된다.

따라서, L(A[n], a[i]=0)에서,

i) i=n-1 일 때,

A[n]=(A[n-1])∗a[n-1] B[n-1]=B[n-1] (1)

이므로 n″=n-1, A″[n″]=B[n-1]이라 하면, 이때, n″=0이면, L(A[n], a[i]=0)=L(A″[0]) 이므로 위의 (a)항을 만족한다.

만약 n″>0이면, a″[n″-1]=b[n-2], $0 \leq j \leq n-2$ 에서 A″[j]=A[j], B″[j]=B[j], a″[j-1]=a[j-1]=b[j-1]이라 두고, 이 때 a″[n″-1]>0 이면, 위의 (b)항을 만족한다.

ii) i<n-1 일 때,

L(A[n])의 정의에 의해 a[i], b[i]는 각각 연속된 정수이므로 b[i]=1이된다.

따라서,

A[i+1]=B[i] (2)

B[i+1]=A[i] B[i] (3)

이다. 한편 (i+2) 레벨에서,

A[i+2]=A[i+1]∗a[i+1] B[i+1] (4)
=[B[i]∗a[i+1] A[i] B[i]

B[i+2]=A[i+1]∗b[i+1] B[i+1] (5)
=B[i]∗b[i+1] A[i] B[i]

가 되므로 만약 $i+1<j<n$인 모든 j에 대해서

$A'[j]=SHIFTL(A[j],\ B[i],\ L(A[n]))$ (6)

$B'[j]=SHIFTL(B[j],\ B[i],\ L(A[n]))$ (7)

이라 두면,

$A'[i+2]=B[i]*(a[i+1]+1)\ A[i]$ (8)

$B'[i+2]=B[i]*(b[i+1]+1)\ A[i]$ (9)

가 된다. 그리고,

$A''[i]=B[i]$ (10)

$B''[i]=A[i]$ (11)

라하고, $0\leq k<i$인 모든 k에 대해서,

$A''[k]=A[k]$ (12)

$B''[k]=B[k]$ (13)

이고, $i\leq m<n-1$에 대해서

$A''[m]=A'[m+1]$ (14)

$B''[m]=B'[m+1]$ (15)

라 두면, $L(A[n])=L(A''[n''])$, $n''=n-1$이 되고, $L(A''[n''])$는 정의에 의해 T의 원소이므로, 식(b) 또는 (c)중의 하나를 만족한다.

〔정리 6〕 T의 원소인 $L(A[n],\ a[n-1]>0,\ b[i]=0)$, $0\leq i<n$에 대해서,

$L(A[n],\ a[n-1]>0,\ b[i]=0)=L(A''[n''],\ a''[n''-1]>0), b''[n''-1]>0$을 만족하는 $L(A[n''])$이 반드시 존재한다.

〔증명〕 이것을 증명하기 위해,

(a) $L(A[n],\ a[n-1]>0,\ b[i]=0)=L(A''[n''],\ a''[n''-1]>0,\ b''[n''-1]>0)$

(b) $L(A[n],\ a[n-1]>0,\ b[i]=0)=L(A''[n''-1],\ a''[n''-1]>0,\ b''[i'']=0), n''\leq n, i''>i$

중에 하나를 만족하는 $L(A''[n''])$가 반드시 존재함을 보인다. (b)에서 우측항의 레벨수는 좌측항의 수 이하($n''\leq n$)이고, 최초로 B-type run-length가 0이 되는 레벨 i''는 i보다 크다. 그리고, $L(A''[n''])$는 $L(A[n])$과 같은 형태로 표현되므로 다시 이를 계속 변환시키면, 최종적으로 (a)와 같이 변환된다.

따라서, $L(A[n],\ a[n-1]>0,\ b[i]=0)$에서, $b[i]$부터 연속해서 적어도 1개 이상의 레벨에서 B-type run-length가 0이 되므로, 연속해서 B-type run-length가 0인 모든 레벨에 대해서, $b[i]=b[i+1]=\cdots=b[j-1]=0$ $(i+1\leq j\leq n-2)$가 되고, $L(A[n])$의 정의에 의하여, $a[i]=a[i+1]=\cdots=a[j-1]=1$이다. 따라서,

$A[j]=A[j-1]\ B[j-1]=A[i]\ B[i]*(j-i)$ (16)

$B[j]=B[j-1]=B[i]$ (17)

이다. 여기서, $k<=i-1$인 모든 k에 대해서,

$A''[k]=A[k]$ (18)

$B''[k]=B[k]$ (19)

라 두고,

$A''[i]=B[i]$ (20)

$A''[i]=A[i]$ (22)

그리고, $a[j]>0$이므로 만약,

i) $j=n-1$이고,

가) $a[n-1]=1$일 때,

$A''[i+1]=SHIFTR(A[n],\ A[i],\ L(A[n]))$ (22)

라 두면,

$A''[i+1]=B[i]*(j-i)*a[n-1]\ B[i]\ A[i]$ (23)

$=B[i]*(j-i+1)\ A[i]$

이 된다. 이 때 $n''=i+1$이라 두면, $L(A''[n''])$는 정의 2)에 의해서, T의 원소이고, $L(A''[n''])$의 A-type run-length는 모두 0보다 크고 i레벨 이하의 B-type run-length도 모두 0이상이므로 위의 (a)식이 성립한다.

나) $a[n-1]>1$일 때

$A''[i+2]=SHIFTR(A[n],\ A[i],\ L(A[n]))$ (24)

라 두면,

$A''[i+2]=(B[i]*(j-i)A[i])*(a[n-1]-1)$
$B[i]*(j-i+1)\ A[i]$ (25)

$A''[i+1]=A''[i]*(j-i)\ B''[i]$ (26)

$B''[i+1]=A''[i]*(j-i+1)\ B''[i]$ (27)

라 두면(여기서 j-i 과 j-i+1은 모두 0보다 크다),

$A''[i+2]=A''[i+1]*(a[n-1]-1)\ B''[i+1]$ (28)

이 된다. 이 때 $n''=i+2$이라 두면, $L(A''[n''])$는 정의 2)에 의해서, T의 원소이고, $L(A''[n''])$의 A, B-type run-length는 모두 0보다 크므로 위의 (a)식이 성립한다.

ii) $j<n-1$이면,

$A[j+1]=A[j]*a[j]\ B[j]=(A[i]\ B[i]*(j-i)\ *a[j]\ B[i]$ (29)

$$B[j+1]=A[j]*b[j] \quad B[j]=(A[i]*(j-1)*b[j] \quad B[i] \tag{30}$$

이다. 여기서,

$$A''[i+1]=A''[i]*(j-i) \quad B''[i] \tag{31}$$

$$B''[i+1]=A''[i]*(j-i+1) \quad B''[i] \tag{32}$$

라 하고, $i+2 \leq k \leq n-j+i$이고, $m=k+j-i-1$인 모든 m, k에 대해서,

$$A''[k]=SHIFTR(A[m],\ A[i],\ L(A[n])) \tag{33}$$

$$B''[k]=SHIFTR(B[m],\ A[i],\ L(A[n])) \tag{34}$$

라 두면,

$$\begin{aligned} A''[i+2] &= SHIFTR(A[j+1],\ A[i],\ L(A[n])) \\ &= (B[i]*(j-i) \quad A[i])*(a[j]-1) \\ &\quad B[i]*(j-i+1) \quad A[i] \\ &= A''[i+1]*(a[j]-1) \quad B''[i+1] \end{aligned} \tag{35}$$

$$\begin{aligned} B''[i+2] &= SHIFTR(B[j+1],\ A[i]\ L(A[n])) \\ &= (B[i]*(j-i) \quad A[i])*(b[j]-1) \\ &\quad B[i]*(j-i+1) \quad A[i] \\ &= A''[i+1]*(b[j]-1) \quad B''[i+1] \end{aligned} \tag{36}$$

이된다. 따라서 정리 4)에 의해 $i+2<m<n-j+i$ 인 모든 m레벨에 대해서 A-type run-length가 0보다 크고, 식 (18) - (21)에서, $L(A[n],\ a[n-2]>0,\ b[i]=0)$이므로, (i+1) 레벨 이하의 A, B-type run-length는 모두 0보다 크다. 이 때,

가) $a[j]>1$이면,

식 (35)에서 (i+2)레벨의 A-type run-length는 0보다 크고, 최상위 레벨 $n-j+i+1$은 n이하 이므로 위의 (b)를 만족한다.

나) $a[j]=1$이면,

$b[j]>0$인 가정에 의해 $b[j]=2$이므로, 식(35)에 따라, (i+2)레벨의 A-type run-length$(a[j]-1)$는 0이다. 이 때 $L(A''[n''])$은

$$L(A''[n''],\ a''[i'']=0, b[i'']>0),\ (i''=i+1,\ n''=n-j+i-1) \tag{37}$$

이 된다. 여기서 $a''[i''-1]>0$, $b''[i''-1]>0$이므로 정리 5)의 식(2)-(12)에 의해 (37)의 레벨수는 감소하고, $(i''+1)$ 레벨 이하 B-type run-length는 0보다 크고, (37)의 A-type run-length는 모두 0보다 커진다. 따라서 이것은 위 (b)항이 성립하는 것을 나타낸다.

〔정리 7〕 T=T1이다.

〔증명〕

i) T의 임의의 원소인 $L(A[n])$은 모든 레벨에서 A, B-type run-length 중 어느 하나는 0이 될 수 있으므로 모든 레벨에서 A, B-type run-length가 0보보 큰 T1의 원소를 포함하는 것은 자명하다.

ii) T의 임의의 원소가 T1의 원소로 모두 변환된다면 T는 T1에 포함한다. 만약 ii)가 증명되면 i), ii)에서 T=T1이 된다.

T의 원소중 모든 레벨에서 A, B-type run-length가 0보다 큰 원소는 T1에 포함된다. 따라서 그렇지 않은 원소는

$$L(A[n],\ a[i]=0) \tag{38}$$

또는

$$L(A[n],\ a[n-1]>0,\ b[i]=0) \tag{39}$$

으로 표현될 수 있다. 그런데, 두 체인코드열의 종류는 정리5), 6)에 의해서

$$L(A[n],\ a[n-1]>0,\ b[n-2]>0 \tag{40}$$

으로 변환이 된다. 따라서 T의 모든 원소는 T1의 원소로 표현되므로 ii)가 성립한다.

다음은 T의 원소가 T1의 원소로 변환되는 예를 보인다.

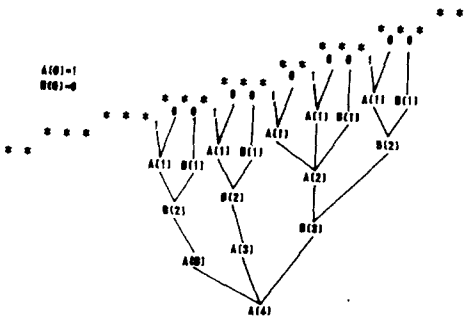
예) 만약 그림 3 a)와 같이 체인코드열 1 0 0 1 0 0 1 0 1 0 0 1 0 0이 무한히 반복되는 체인코드열 $L(A[n])$이 있다고 하면, 이것은 $L(A[4],\ a[2]=0)$와 같이 표현된다. 여기에 정리 5)을 적용하면, 그림 3 b)와 같이 변환되고, 여기에 정리 6)을 적용하면, 그림 3 c)와 같이 변환된다. 즉 원래의 체인코드열 $L(A[4])$은 $L(A''[2])$로 변환 되고, 이 체인코드열은 T1의 원소가 된다.

2. 디지탈 선분의 표현

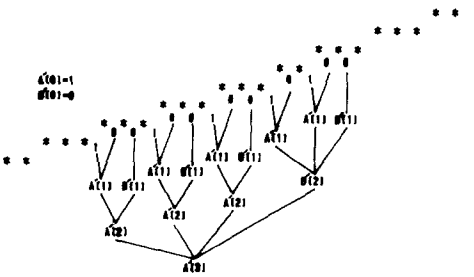
실제적으로 영상분석에 사용되는 양자화된 직선성분은 그림 1 a)의 경우와 같이 일부가 잘려짐에 의해 Freeman[1]이나 Rosenfeld[2]가 제시한 디지탈 직선성분의 범주를 벗어나는 경우가 많다.

T의 원소인 임의의 직선$L(A[n])$의 일부인 임의의 디지탈 선분은 좌우에 끝점을 가지게 되고, 이 끝점들은 임의의 위치에 발생할 수 있다. 따라서 이 점의 위치에 따라 좌우에 Freeman[1]과 Rosenfeld[2]가 제안한 직선의 특성을 상실한, 보다 작은 디지탈 선분이 발생할 수 있다. 다음은 이러한 디지탈 선분을 표현하기 위해 다음과 같은 정의를 한다.

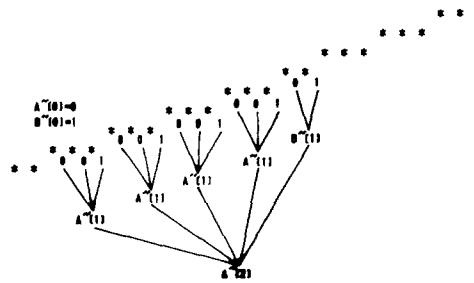
정의 4) 어떤 체인코드열 $H=h_1h_2\cdots h_n$(h_i는 8방향



(a)



(b)



(c)

그림 3. 체인코드열의 변환
(a) T의 원소인 디지탈 직선
(b) 정리5에 의한 (a)의 변환
(c) 정리6에 의한 (b)의 변환

Fig. 3. Transfomations of a chain code string.
(a) a digital straight line in T.
(b) Transtomation of (a) by theorem 5.
(c) Transfomation of (b) by theorem 6.

체인코드, $0 \leq i \leq n$)이 있고, $H' = h_i \ h_{i+1} \cdots h_j$ ($1 \leq i \leq j \leq n$) 또는 $H' = \lambda$라 할 때, H'의 집합 $P(H)$라 하고, $i=1$이고, $j<n$ 이거나 $H'=\lambda$인 H'의 집합을 $PL(H)$, $i>1$이고, $j=n$이거나 $H'=\lambda$인 H'의 집합을 $PR(H)$이라 한다. 이 때 $PL(H)$, $PR(H) \subset P(H)$이다.

정의 5) $L(A[n])$의 임의의 레벨 $i(0<i \leq n)$의 A 또는 B-type 체인코드열의 우측일부 체인코드열을 $S[i]$라 하고 여기서,

$$S[i] \in PR(A[i]) \tag{41}$$

또는

$$S[i] \in PR(B[i]) \tag{42}$$

로 정의한다. 이 때 $S[i]$를 i 레벨의 S-type체인코드열이라 한다. 이 때 $S[i]$는 $L(A[n])$과 PL의 정의에 따라서,

$$S[i] = S[i-1] \ A[i-1] * s[i-1] \ B[i-1] \tag{43}$$

또는,

$$S[i] = S[i-1] \tag{44}$$

로 표현되므로, 이는 계층구조를 가지게 된다.

한편, $L(A[n])$의 임의의 레벨 $ii(0<i<=n$의 A 또는 B-type 체인코드열의 좌측일부 체인코드열을 $C[i]$라 하고, 여기서,

$$C[i] \in PL(A[i]) \tag{45}$$

또는

$$C[i] \in PL(B[i]) \tag{46}$$

로 정의한다. 여기서 $C[i]$를 i 레벨의 C-type 체인코드열이라 한다. 이 때, $C[i] = A[i-1] * c[i-1]$

$$C[i] = A[i-1] * c[i-1] \ C[i-1] \tag{47}$$

로 표현되고, 이는 계층구조를 가진다.

이 때 임의의 레벨 i 에 대해서 $S[i]$에 포함되어있는 $A[i-1]$의 run-length $s[i-1]$를 i레벨의 S-type run-length라 하고, $C[i]$에 포함되어 있는 $A[i-1]$의 run-length $c[i-1]$를 i레벨의 C-type run-length라 한다.

T의 임의의 원소인 $L(A[n])$이 있고, X는 $L(A[n])$의 일부인 임의의 디지탈 선분이라고 하면,

$$X = S[i] \ A[i] * c[i] \ C[i] \tag{48}$$

로 표현할 수 있다. 이 때, $i<n$ 이면, (48)은

$$X = S[i+1] \ C[i+1] \tag{49}$$

로 나타낼 수 있으며 여기서, 만약 $i+1<n$이면 $S[i+1]$과 $C[i+1]$은 동시에 $P(B[i+1])$의 원소가 될 수 없다.

다음 그림은 디지탈 선분을 계층구조로 표현한 예이다.

Ⅲ. 알고리듬 및 적용

본 알고리듬에 의한 디지탈 선분의 추출 방법은 임의의 체인코드열이 주어지면, 좌측체인코드열부터 순

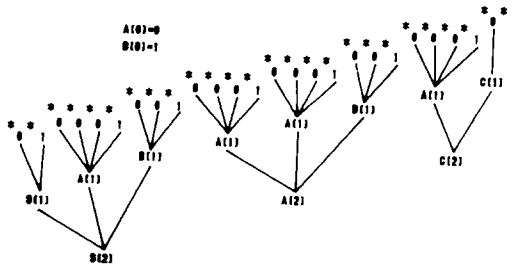


그림 4. 디지탈 선분의 구조적 표현
Fig. 4. Structural representation of a digital straight line segment.

차적으로 입력된다. 입력된 체인코드열은 식(48)과같이 좌측 S-type 체인코드열부터 순차적으로 체인 코드열을 결정한다. 그리고, 좌측체인코드들에 의해 만들어진 체인코드열의 종류에 따라 우측체인의 종류가 결정된다. 만약 식(8)에서 X의 우측에 새로운 체인코드D(8방향 체인코드 중의 하나)가 있다고 하고 X와 D를 합한 체인코드열을 X′라 하면,

$$X' = X\ D \tag{50}$$

이다. 한편 즉(50)식은, (48)에 의해,

$$X' = S[i]\ C[i] * c[i]\ D \tag{51}$$

가 된다. 이 때 X′가 디지탈 선분이 되기 위해서는, C′[i]=C[i] D라 두면,

$$C'[i] \in PL(A[i]) \tag{52}$$

또는,

$$C'[i] \in PR(B[i]) \tag{53}$$

중의 하나가 된다.

본 방법에서는 C′[i]가 (52) 또는 (53)를 만족하는 D의 집합을 구하고, X다음에 연결되는 체인 코드가 이에 포함이 되면 X′를 디지탈 선분으로 하고,아니면 X의 마지막점이 dreak point가 된다.

1. 디지탈 선분이 되기 위한 C-type 체인코드열의 집합

임의의 체인코드열 X가 있고, X는

$$X = S[i]\ A[i] * c[i]\ C[i]\ \ (i=1,2,3,\cdots) \tag{54}$$

로 표현될 때, X가 디지탈 선분이 되기 위한 C[i]는 여러가지로 존재할 수 있다. 임의의 $k(0<k\leq i)$ 레벨에서, C[k]는 식 (47)에서,

$$C[k] = A[k-1] * c[k-1]\ C[k-1] \tag{55}$$

이므로 여기서, C[k]는 A[k] 또는 B[k]의일부 체인코드열이므로, C[k-1]이 PL(B[k-1])의 원소가 되기 위한 c[k-1]의 최대값은 a[k-1] 또는 b[k-1]이 된다. 그런데 만약 $C[k] \in PL(A[k]) \cup PL(B[k])$라고 하면, $C[k-1] \in PL(B[k-1])$이 되기위한 최대 및 최소 c[k-1]은 각각 MAX(a[k-1], b[k-1]), MIN(a[k-1], b[k-1])이 된다.

〔정의 6〕 $X = S[i]\ A[i] * c[i]\ C[i]\ \ (i=0,1,2,\cdots)$인 임의의 디지탈 선분이 있을 때, 임의의 레벨 $k(0<k\leq i)$에 대해, 반드시 C[k] PL(B[k])을 만족하는 c[k-1]의 최대값을 cmax[k-1], 최소값을 cmin[k-1]이라 한다.

한편, 식 (54)에서 c[i]=0이면, i 레벨의 A, B-type 체인코드열은 미정이므로, a[i-1], b[i-1]도 미정이다. 따라서, X가 디지탈 선분이 되기위한 a[i-1], b[i-1]는 이중 적어도 하나는 식 (41), (42)에 의해서 s[i-1]이상이고, 이 때, a[i-1], b[i-1]은 0이상인 연속된 정수이므로, 적어도 s[i-1]-1이상(만약 s[i-1]=0이면 0이상)이 된다. 따라서 cmin[i-1]은 이들 중에서 최소값 즉, MAX(s[i-1]-1, 0)이 되고, cmax[i-1]=∞가 된다.

그런데 만약 c[i]>0일때, a[i-1]은 결정되어 있으나 b[i-1]은 결정되어 있지 않으므로, b[i-1]은 a[i-1]+1 또는 a[i-1]-1중의 하나로서, cmax[i-1]=a[i+1]+1이 되고, cmin[i-1]=MAX(a[i-1]-1, s[i-1]) 가 되는데, cmin[i-1]의 경우 만약 s[i-1]=a[i-1]+1이면, s[i-1]=b[i-1]이 될 수 밖에 없고 이 때, cmin[i-1]은 a[i-1]=s[i-1]-1이 된다(그림5 의 step {6}).

그리고 $k(0<k<i)$ 레벨에서, cmax[k] 및 cmin[k]가 결정되어 있다고하면,

$$C[k+1] = A[k] * c[k]\ C[k]$$

에서,

(a) c[k]<cmin[k] 이면, $C[k] \in PL(A[k])$

(b) cmin[k] ≦ c[k]< cmax[k]이면, $C[k] \in \{PL(A[k]) \cup PL(B[k])\}$

(c) c[k]=cmax[k] 이면, $C[k] \in PL(B[k])$

가 된다. 따라서, 이 때 cmax[k-1], cmin[k-1]은 위의 각각의 경우

(d) cmax[k-1]=a[k-1], cmin[k-1]=a[k-1]

(e) cmax[k-1]=MAX(a[k-1], b[k-1]), cmin[k-1]=MIN(a[k-1], b[k-1])

(f) cmax[k-1]=b[k-1], cmin[k-1]=b[k-1]

이 된다(그림 5의 step {7}).

```
Global variables
  s,a,b,c,cmax,cmin : array of integer;
  break : boolean  { break point } ;
  top : integer { the highest level };
  A0,B0 : integer { A[0], B[0] };

Constants
  ND=-1 { for any not-decided number };
  INF=65535  { as infinitive number };


procedure HLP(c1,k:integer);
begin
  i:=k-1;
  if top<k then begin                                  {1}
     top:=k; s[i]:=c1; a[i]:=ND; b[i]:=ND; end
  else if a[i]=ND then  begin                          {2}
     a[i]:=c1; c[i]:=1;
     if a[i]<s[i]  then begin
        HLP(0,k+1); b[i]:=s[i] end
     end
  else if a[i]=c1 then                                 {3}
     c[k]:=c[k]+1
  else if b[i]=ND then begin                           {4}
     b[i]:=c1;
     if (a[i]=s[i]) and (a[i]>b[i]) then c[k]:=c[k]+1;
     HLP(c[k],k+1); c[k]:=0
     end
  else begin                                           {5}
{    HLP(c[k],k+1); c[k]:=0 end;

{ Decide cmax[i],cmin[i] }
  if b[i]=ND then                                      {6}
    if a[i]=ND then begin
       cmax[i]:=INF; cmin[i]:=MIN(s[i]-1,0) end
    else begin
       cmax[i]:=a[i]+1; cmin[i]:=MAX(a[i],s[i])-1 end
  else begin                                           {7}
    if c[k]<cmin[k] then begin
       cmax[i]:=a[i]; cmin[i]:=a[i] end
    else if c[k]=cmax[k] then begin
       cmax[i]:=b[i]; cmin[i]:=b[i] end
    else begin
       cmax[i]:=MAX(a[i],b[i]);
       cmin[i]:=MIN(a[i],b[i]) end
  end
end;  {of HLP}
```

그림 5. HLP 알고리듬
Fig. 5. Alorithm of HLP.

즉 상위 레벨의 cmax, cmin은 하위레벨의 C-type 체인코드열의 종류를 결정하고, 어떤 레벨에서 C-type 체인코드열의 종류가 결정되면, 이들에 의해서 그 레벨의 cmax, cmin 값이 결정된다. 이와같이 하여 최상위레벨로 부터, 최하위 레벨까지 cmax, cmin 값이 결정되면, C[0]의 종류가 결정되고, 식 (51)에서 D가 C[0]의 종류에 해당하면, X′가 디지탈 선분이 되고, 아니면 D의 시작점이 break point가 된다(그림 6).

2. 알고리듬

본 알고리듬은 체인코드열로 부터 좌측에서 우측으로 scanning하며, 이들 체인드열로 부터 디지탈 선분을 찾기위해 0레벨 parser(Parser0)와 0보다 큰 레벨을 위한 parser(HLP:High Level Paser) 등 두가지 procedure가 있으며, parser0는 HLP를 procedure를 사용하며, HLP는 recursive program이다. 그리고 HLP는 cmax, cmin 값을 return한다.

한편, 각각의 HLP에서는, B-type 체인 코드열이 발생하면, 그 이전까지 입력된 A-type 체인코드의 run-length(C-type run-length)를 다음 레벨의 높이와 함께 HLP로 보낸 다음, 수행이 끝나면 현재의 C-type run-length를 0로 한다. 끝으로 HLP는 상위레벨로 부터 주어지는 cmax 및 cmin과 C-type run-length로 부터 하위 레벨의 cmax와 cmin을 구한다(그림 5).

여기서, 각레벨의 체인코드열은 A[0], B[0] 또는 각 type의 run-length로 표현된다.

그리고 Parser0는 HLP와 같이 B-type 체인코드열이 입력되면 C-type run-length와 현재 레벨수와 함께 HLP로 보내지고, HLP의 수행이 끝나면 C-type run-length는 0가 된다.

Parser0에서는 최초에 입력되는 체인코드열을 A[0]라하고, S[0]는 nullstring으로 취급하여, 이를 무시한다. 한편 두번째 이상의 체인코드는

(a) 첫 번째 코드와 일치하거나, 45도 차이가 있는 체인코드로서, 세번째 종류의 체인코드가 아니어야 하고,

(b) 이 코드를 C[0]라 할 때, 3.1절의 (a),(b),(c) 중의 하나를 만족해야 한다.

만약 위의 (a), (b)를 모두 만족하지 못하면, 이 체인코드의 시작점을 break point로 간주한다(그림 6).

3. 알고리듬의 적용

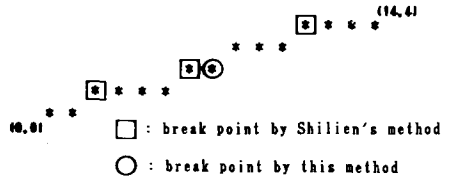
그림 7은 위의 알고리듬에 의한 디지탈 선분 추출

```
procedure  parser0(D:integer);
begin
 if top=ND then begin                                  {1}
   initialize; {A0:=D; B0:=ND; c[0]:=1; break:=false;
                cmax[0]:=INF; cmin[0]:=NA; top:=0;}
   end
 else if A0=D then begin                               {2}
   c[0]:=c[0]+1;
   if (cmax[0]<>INF) and (c[0]>cmax[0])
      then break:=true end
 else if B0=ND then begin                              {3}
   if (ABS(A0-D)=1) or (ABS(A0-D)=7) then begin
      B0:=D; HLP(c[0],1); c[1]:=0; end
   else  break:=true
 else if B0=D then                                     {4}
   if c[0]<cmin[0]  then  break:=true
   else begin
      HLP(c[0],1); c[0]:=0; end
 else break:=true;                                     {5}

 if break then begin                                   {6}
    initialize;
    MARK-current-point
    {Mark current point as break point}
 end;
end;  {of parser0}
```

그림 6. 0레벨 parser(Parser 0)의 알고리듬
Fig. 6. Algorithm of 0 level paser(Parser 0).

예이다. 그림 7(a)에서 본 방법에 의하면 break point가 생기는 것을 보인다. 한편 그림 7(b)는 본 방법에 의해서 그림 7(a)의 디지탈 곡선으로 부터 디지탈 선분을 찾아내는 과정을 보이고 있다. 그림 7(b)에서 좌측 체인코드가 입력되면 Parser0와 HLP에 의해 그림의 우측과 같은 결과가 된다. 그리고 결과의 생성 순서는 상단, 좌측순으로 진행된다. 그림의 점 (6, 2)에서 체인코드 0가 입력되면 그림 7(b)와 같이 c[0]=1이 되고, 이 때 계속해서 cmax[0]=4, cmin[0]=2이 되는데, 다음 점 (7, 2)에서 체인코드는 1이 되고 이것은 B[0]와 같다. 그런데 이때 c[0]=1이므로 Parser0의 step {4}에 의해서 이점은 break point가 된다.



(a)

INPUT	RESULTS		
(x,y) code	level 0	level 1	level 2
(0,0) 0	A0=0,B0=NA		
	c[0]=1		
(1,0) 1	B0=1	s[0]=1,a[0]=b[0]=NA	
		cmax[0]=INF	
		cmin[0]=0	
	c[0]=0		
(2,1) 0	c[0]=1		
(3,1) 0	c[0]=2		
(4,1) 0	c[0]=3		
(5,1) 1		s[0]=3	
		cmax[0]=4	
		cmin[0]=2	
	c[0]=0		
(6,2) 0	c[0]=1		
(7,2) 1	A0=1,B0=NA	-----> break point	
	c[0]=1		
(8,3) 0	B0=0	s[0]=1,a[0]=b[0]=NA	
		cmax[0]=INF	
		cmin[0]=0	
	c[0]=0		
(9,3) 0		s[0]=1,a[0]=0,b[0]=1	s[1]=0
			a[1]=b[1]=NA
			cmax[1]=INF
			cmin[1]=0
		c[1]=1	
		cmax[0]=1	
		cmin[0]=0	
	c[0]=0		
(10,3) 1	c[0]=1		
(11,4) 0			a[1]=1,c[2]=1
			cmax[1]=2
			cmin[1]=0
		c[1]=0	
		cmax[0]=1	
		cmin[0]=0	
	c[0]=0		
(12,4) 0		c[1]=1	
		cmax[0]=1	
		cmin[0]=0	
	c[0]=0		
(13,4) 0		c[1]=2	
		cmax[0]=1	
		cmin[0]=1	
	c[0]=0		

(b)

그림 7. 디지탈 선분의 추출 예
(a) Break point
(b) 디지탈 선분의 추출과정

Fig. 7. An example of the extraction of digital straight line segments.
(a) Break point.
(b) Exraction of digital straight line segments.

Ⅳ. 결 론

본 논문은 Freeman [1] 및 Sosenfeld[2]가 제안한 디지탈 직선의 특성을 만족하는 BA에 의해서 표현되는 디지탈 직선의 T의 원소로 변환됨을 증명하였고, 본 논문에서는 이를 이용하여 디지탈 선분을 구조적 표현을 하였다.

본 논문은 디지탈 선분이 비록 디지탈 직선의 특성 [1, 2]을 만족하지 않더라도 이를 추출할 수 있는 방법을 제시하여 오직 디지탈 직선의 특성을 만족하는 디지탈 선분을 추출하는 Shilien의 방법[3]에 비해 break point의 수를 줄일 수 있다.

한편 Shilien [3]이 제시한 방법은 break point를 0 이상의 상위레벨에서 판정하게 되므로 break point 를 찾기 위해, 각 레벨의 체인코드열(벡터)의 크기(x 및 y의 증분)을 계산하므로 이에 따라 정수의 곱셈이 필요한데, 본 방법은 0레벨에서 정수의 비교만으로 break point 판정이 가능하다. 따라서 본 알고리듬은 정수의 가감산 및 비교만으로 디지탈 선분의 추출이 되므로 low level 하드웨어에서 실현 가능하여, 고속 화상처리에 이용될 수 있을 것으로 생각된다.

본 방법과 Shilien[3]방법에 의한 구조해석적 디지탈 직선 추출 방법은 오차의 범위가 한 화소 이내이므로 화상의 윤곽선으로 부터 정밀한 근사화 다각형을 추출하는데 효과적이다.

參 考 文 獻

[1] H. Feeman, "On the encoding of abitary geometric configuration," *IRE Trans. on Electronic computers 10*, pp. 260-268, 1961.

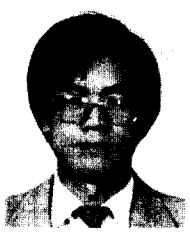
[2] A. Rosenfeld, "Digital straight line segment," *IEEE Trans on. Comput. C-23*, pp. 1264-1269, 1974.

[3] S. Shilien, "Segmentation of digital curves using linguistic techniques," CVGIP 22, pp. 277-286, 1983.

[4] R. Brons, "Linguistic methods for the description of a straight line on a grid," CGIP 3, pp. 48-72, 1974.

[5] J. Rothstein and C. Weinman, "Parallel and sequential specifications of a context senstive language for straight lines on a grid," CGIP 5, pp. 106-124, 1976.
[6] C. Arcelli and A. Massarotti, "On the parallel generation of straight digital lines," CGIP 7, pp. 67-83, 1978.
[7] T. Pavlidis and S. Horowitz, "Segmentation of plane curves," *IEEE Trans. on Computers, C-23*, pp. 860-870, 1974.
[8] H. Freeman and L.S. Davis, "A corner-finding algorithm for chaincoded curves," IEEE Trans. on Computers, C-26, pp. 297-303, 1977.
[9] T. Pavlidis, "Polygonal approximations by Newton's method," *IEEE Trans. on Computers, C-26*, pp. 800-807, 1977.
[10] J. Sklansky and V. Gonzalez, "Fast polygonal approximation of digitized curves," PR, vol. 12, pp. 327-331, 1980.
[11] Y. Kurozumi and W.A. Davis, "Polygonal approximation by the minimax method," CGIP 19, pp. 248-264, 1982.
[12] K. Wall and P.E. Danielsson, "A fast sequential method for polygonal approximation of digitized curves," CVGIP 28, pp. 220-227, 1984.

著 者 紹 介



柳 承 必(正會員)
1979年 서울대학교 전자공학과 졸업. 1987年 충남대학교 전자공학과(공학석사). 1980年~현재 한국에너지 연구소 선임연구원 주관심분야는 패턴인식, 화상처리, 인공지능등임.

●

權 五 錫(正會員)
1977年 서울대학교 전자공학과 졸업. 1980年 한국과학기술원 전기및 전자공학과(공학석사). 1980年~현재 충남대학교 전자계산기 공학과 근무(조교수). 연구분야는 컴퓨터구조, 컴퓨터 그래픽스, 컴퓨터 비젼등임.

金 太 均(正會員)
1971年 서울대학교 공과대학 졸업. 1978年 일본동경공업대학 물리정보학과(공학석사). 1985年 일본동경공업대학 물리정보학과(공학박사). 1979年~1980年 미국 Purdue 대학 객원 연구원. 1974年 ~ 현재 충남대학교 전자계산기 공학과 근무중(부교수). 연구분야는 패턴인식, 컴퓨터 에니메이션, 오토마타이론등임.