The Journal of GIS Association of Korea, Vol. 2, No. 2, pp.197-205, October 1994

지적도를 위한 자동 지형 객체 인식 및 모델링

유희종* · 정창성*

Automatic Geographical Entity Recognition and Modeling for Land Registered Map

Hee-Jong Yoo · Chang-Sung Jeong

요 약

본 논문에서는 여러 응용 분야의 GIS에서 기본 map으로 사용되는 지적도에 대하여 raster 영상 데이타로부터 벡터(vector) 영상을 구하는 벡터화 알고리즘 및 자동 지역 생성 알고리즘을 제시하고, 이들로부터 직선(edge)들로 이루어진 segment, region(polygon)등의 주요 지형 객체 인식 및 처리를 자동적으로 수행하는 고속 지형 객체 처리 소프트웨어 ARM(Automatic geographical entity Recognition and Modeling)에 대해서 설명한다.

ABSTRACT : In this paper, we present a vectorization algorithm for finding a vector image from a raster image of the land registered map which is used as the base map for various applications, and an automatic region creation algorithm for generating every region automatically from the vector image. We describe an ARM(automatic geographical entity recognition and modeling software) which carries out the recognition and processing of geographical entities automatically using those algorithms.

서 론

현대 사회가 복잡해지고 다양화 되면서 방대한 양의 정보에 대한 효율적인 관리가 절실히 요구되고 있다. 지도 정보(Geographic Information)는 일상 생활뿐만 아니라 기업과 국가 행정 기관 분야에서 많이 사용되고 있으며 전화 시설, 전력 시설, 상하수도, 가스등의 시설물 관리, 산림, 농지, 수로 등의 환경 관리

* 고려대학교 전자공학과 (Electronics, Korea University, 5 Ga, Anam dong, Seongbuk Ku, Seoul, Korea, Tel. (02) 920-1435)

와 소방, 교통, 부동산 등의 다양한 분야의 사업 계획, 국토 계획, 자원 개발 계획, 관리 계획 등에 광범위하게 응용되고 있기 때문에 GIS(Geographical Information System)에 대한 인식의 재고가 이루어지고 있다. 특히 지적도는 통신, 전력, 교통, 상하수도 등 여러가지 분야의 계획에 기본이 되는 도면으로, 지적도와 관련된 여러가지 geometric operation을 효율적으로 수행할 수 있는 geometric model의 설계는 매우 중요하다. 그러나 국외에서 개발된 GIS는 국내 환경에 적합하지 않은 부분이 있어 사용하기가 부적합하고, 국내에서의 연구는 아직 자동 지형 객체 인식 및 데이타 모델링 단계에까지는 이르지 못하고 있다.

본 논문에서는 지적도에 대한 영상 데이타로부터 선들로 이루어진 segment, region등의 주요 지형 객체 인식 및 모델링을 자동적으로 수행하는 고속 지형 객체 인식 및 모델링 소프트웨어 ARM(Automatic geographical entity Recognition and Modeling)에 대하여 설명한다. ARM은 다음과 같은 다양한 기능을 가지고 있다.

- 문자/영상 분리
- 벡터 데이타 생성 및 geometric 모델링
- Error detection / correction 및 raster / vector 편집 기능
- 자동 지역(region) 인식 및 생성

문자 / 영상 분리 프로세스에서는 raster 영상에서 숫자 또는 문자를 제거하여 직선들로 이루어진 영상을 추출하게 된다. 벡터 데이타 생성 및 geometric 모델링 프로세스에서는 문자가 제거된 raster 영상을 전처리하고 이 영상으로부터 직선 segment로 이루어진 벡터 데이타를 생성하고, 이들 segment들 간의 geometric 데이타 구조를 모델링 한다. 벡터 데이타 생성시 문자가 직선과 연결되어 있는 경우에는 직선과 문자를 분리시키는 것이 매우 어렵기 때문에 error detection / correction 프로세스에서는 문자가 직선과 연결되어 있을 가능성이 있는 모든 부분을 detection하고 error를 correction하는 기능을 수행하고 raster / vector 영상 편집기능을 이용하여 error correction 기능을 지원한다. 자동 지역 인식 및 생성 프로세스에서는 geometric 모델을 이용하여 모든 지역(region)을 자동적으로 생성하여 지역 정보에 대한 입출력 및 공간 처리를 효율적으로 수행하게 된다.

본 논문의 구성은 다음과 같다. 우선 문자 분리 및 벡터 데이타 생성을 위한 알고리즘을 제시하고 벡터 데이타의 geometric model에 대하여 설명한다. 그리고 오차 수정과 raster / vector 편집에 대해 간단히 설명하고 자동 지역 생성에 대한 알고리즘을 설명한 후 마지막에 결론을 내린다.

문자 분리 (Character separation)

주어진 지적도는 스캐너를 통하여 이진 영상이나 grey 영상으로 읽어 들인다. Grey 영상으로 읽어 들인 경우에는 이진화(thresholding operation)과정을 거쳐 이진 영상을 만든다. 우리나라의 지적도는 지번을 나타내는 숫자와, 지목을 나타내는 글자, 그리고 실제 대부분의 정보를 담고 있는 직선으로

이루어져 있다. 직선으로 이루어진 vector 영상을 만들기 위해서는 문자(숫자,글자) 분리 과정(character separation operation)을 수행해야 한다. 문자 분리 과정에서는 문자로 추정되는 조건을 만족하는 임의의 점에 대하여 최소 외접 사각형(minimun enclosing rectangle)을 구한 후, 이 사각형의 성질을 이용하여 문자의 존재 여부를 판정한다. 대부분의 숫자나 글자는 직선과 겹쳐져 있지 않지만, 직선과 겹쳐져 있는 부분도 지적도에는 적지 않게 존재한다(S. Bow, R. Kasturi, 1990) (L. A. Fletcher, R. Kasturi, 1990). 이와 같은 부분에서 직선과 문자를 완전히 분리해 내는 것은 매우 어려운 문제이다. 구체적인 문자 분리 과정의 알고리즘은 다음과 같다.

Character—Separation—Algorithm

Input : binary image im[image—size];

Output : binary character image buf[image—size]

binary line image buf2[image—size]

```
1) unsigned char p[9]; /* 3 X 3 mask window */
2) memcpy(buf2,im,image—size);
3) for i=0 to image—size do
4)   if(im[i] == BLACK)
5)     MakeWindow(im,p,i); /* im[i]점에 3 x 3 window를 만든다 */
6)     if(p[2]+p[3]+p[4]==0) /* upper bound is white */
7)       if(IsCharacter(im[i],&x1,&y1,&x2,&y2))
8)         DeleteImage(im,x1,y1,x2,y2);
9)         ImageCopy(buf,buf2,x1,y1,x2,y2); /* copy from buf2 to buf */
10)        DeleteImage(buf2,x1,y1,x2,y2);
```

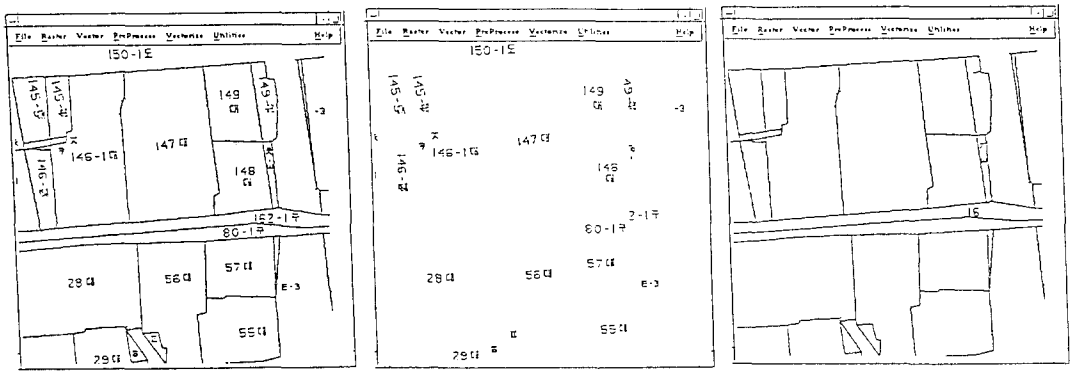
End of Character—Separation—Algorithm

문자 분리 알고리즘에서 사용하는 3×3 마스크는 Fig.1과 같다.

p2	p3	p4
p1	p	p5
p8	p7	p6

Fig. 1 3X3 mask

7)번 라인의 IsCharacter()함수는 주어진 영상을 스캔(scan)하면서 만나는 임의의 점에 대하여 8—connectivity(T. Pavlidis, 1982)를 이용하여 경계선을 따라가면서 최소 외접 사각형을 구한 후, 이 사각형의 존재 여부 및 크기를 포함한 여러가지 성질로부터 숫자 및 글자의 존재 여부를 판정하는 함수이다. (x1, y1)은 최소 외접 사각형의 왼쪽 상단의 점이며, (x2, y2)는 오른쪽 하단의 점을 나타낸다. 처음에 고려한 시작점이 직선의 일부분이라면 최소 외접 사각형은 발산을 하게 되어 구할 수 없어지므로 판정에서 제외한다. 문자가 직선과 겹쳐져 있는 경우는 문자 분리 알고리즘 상에서 제거되지 않고 후에 벡터화 과정에서나 error correction / detection 프로세스에서 제거한다. Fig. 2는 문자 분리 알고리즘을 이용한 예를 보여주고 있다. Fig. 2의 (a)는 스캐너에서 읽어 들인 영상을 나타내고 (b)와 (c)는 이 영상을 문자와 직선으로 각각 분리한 영상을 나타낸다.



(a) original image (b) character image (c) line image

Fig. 2 Example of character separation algorithm

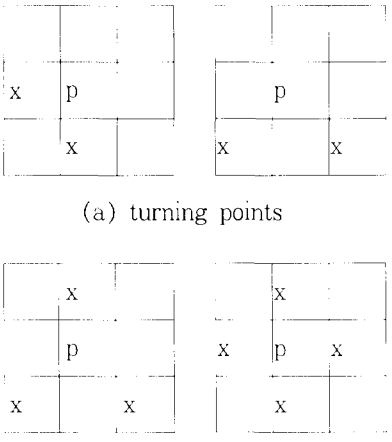
벡터화(Vectorization)

문자 분리 과정을 거쳐 문자가 제거된 영상의 한 직선의 굵기는 여러 pixel의 두께를 가지므로 세선화 과정(Zhang. T. Y, Suen, C.Y, 1984)을 수행하여 한 pixel이나 두 pixel정도의 두께가 되도록 한다. 세선화 과정에서는 연결되어 있는 직선이 끊어지는 것, 직선의 끝점에서부터 직선이 점점 지워지는 것, 그리고 세선화 과정을 수행하기 전과 모양이 크게 바뀌는 것을 피해야 한다.

세선화 과정을 마친 후에는 node point를 구한 후, 이 들 node point에 대한 정보를 이용하여 벡터화 알고리즘을 수행한다. 본 논문에서는 우선 node point 인식에 대하여 설명한 후 벡터화 알고리즘을 제시한다.

Node point의 인식

Node point들은 3개 직선이 만나는 junction point와 2개의 직선이 만나는 turning point들로 이루어져 있다(M. T. Musavi, 1988). Fig. 3 (a)는 turning points를 나타내고 Fig. 3 (b)은 junction points를 나타낸다.



(a) turning points

(b) junction points

Fig. 3 node points

Node point를 구하기 위해서는 3×3 window를 이용하여 crossing number(CN)를 구해야 한다. CN는 다음과 같은 식으로 구한다.

$$CN=\frac{1}{2}\left(\sum_{i=0}^{8}\mid q_i-q_{i+1}\mid\right)$$

여기에서 $q_i=1$ 은 p_i 의 값이 black일 때이고, $q_i=0$ 은 p_i 의 값이 white일 때이다. 그리고 $q_9=q_1$ 이다. CN=3이나 CN=4인 점을 junction point라 하고, CN=1인 점은 직선의 end point를 나타낸다. CN=2인 점들은 직선의 중간점을 나타내며 Fig. 4와 같은 형태가 있다. Fig. 4의 f와 b는 임의의 점 p를 중심으로 한 마스크 윈도우의 내부의 black pixel로써, 1-step forward direction과 1-step backward direction을 나타낸다. Fig. 4 (a)와 같이 f와 b가 일직선 상에 있는 점은 turning point가 될 수 없다.

휜정도(cornerty)를 나타내는 값을 이용하여 turning point의 대략적인 후보점들을 정한 후, 그 후보점들에 대하여 기울기 비교 알고리즘을 수행하여 정확한 turning point를 찾는 방법을 사용한다. 우선 Fig. 4의 (b),(c)와 같은 점들을 대상으로 휜정도(cornerty)를 구하여 대략적인 turning point를 결정한다. Fig. 4 (a),(b),(c)는 CN=2이면서 3×3 window내에 black pixel의 갯수가 3개인 경우만을 나타낸 것이며 실제 지적도의 직선 영상에서는 CN=2이면서도 black pixel의 갯수가 3개 이상인(Fig. 4 (d),(e))경우도 많다.

f와 b에 대하여 forward, backward 2-step 3×3 mask window를 다시 한번 적용한다. 정확한 turning points를 찾기 위해서는 k-step까지 반복할 수 있으나 step수가 많아지면 계산해야 하는 양이 증가하므로 적절한 step수를 결정하는 것이 필요하다. 휜정도를 판별하는 알고리즘(F. H. Cheng, W. H. Hsu, 1988)은 Fig. 4의 (b),(c)와 같이 CN=2 이고 black pixel의 갯수가 3인 경우에는 적용될 수 있지만 Fig. 4의 (d),(e)인 경우처럼 blalck pixel의 갯수가 3 이상인 경우를 고려하지 않았기 때문에 d_1^f와 d_1^b(F. H. Cheng, W. H. Hsu, 1988)를 결정할 때 black pixel의 갯수가 3 이상인 경우도 가능하도록 해야 한다.

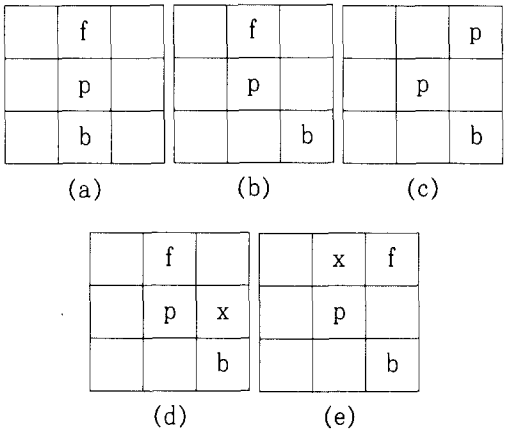


Fig. 4 CN=2 points

한편 휜정도를 구하여 turning point를 구하는 알고리즘에서는 step수를 무한히 늘릴 수가 없기 때문에 turning points의 갯수가 너무 많이 생기는 단점이 있다. 실험적으로 휜정도를 이용한 알고리즘 만으로는 실제의 turning point보다 10-20배 이상의 turning point를 인식한다. 이와 같은 결과를 그대로 사용하기에는 불필요한 데이타가 너무 많기 때문에 본 연구에서는 이렇게 인식된 turning point들을 기울기 비교 알고리즘 (slope matching algorithm)으로 다시 한번 처리하였다. 기울기 비교 알고리즘이란 turning point의 후보

로 선택된 pixel에 대해서 그 점이 위치해 있는 직선 위의(turning point는 항상 CN=2) 두 점을 선택해서 세 점을 연결하는 직선의 기울기(각)를 구해 비교 결정하는 방법이다. 즉 turning point 후보점 P라는 black pixel에서 직선 위에 임의의 λ 만큼 떨어진 S점과, 임의의 λ만큼 떨어지고 방향이 P를 중심으로 S와 반대쪽 직선 위에 있는 black pixel R점의 SPR의 각을 구해서 임계각 δ과 비교해서 turning point를 결정한다. 본 논문의 실제 구현에서는 임계각을 heuristic하게 160도로 정하였다.

벡터화(vectorization) 알고리즘

벡터화 알고리즘은 node point에 도착할 때까지 지적도 영상의 직선을 따라 진행하면서 edge segment를 결정하는 과정을 전체의 영상에 대하여 모든 node point를 방문할 때까지 반복한다. 본 연구에서는 큐(Queue)를 이용한 breadth-first-search 방법을 사용하여 하나의 edge를 큐에서 꺼내어 방문하고 그 edge에 연결된 다른 edge들의 갯수를 결정한 후 그 갯수 만큼의 edge를 다시 큐에 저장하였다. 본 연구에서 제안한 edge segment의 자료 구조는 Fig. 5와 같다.

위의 알고리즘의 5)번 줄에서 큐에서 꺼낸 EDGE는 시작점 P1 만이 저장되어 있다. 6)

```
▶ type of EDGE
POINT   P1      /* start point */
POINT   P2      /* end point */
EDGE    FR      /* forward right edge */
EDGE    FL      /* forward left edge */
EDGE    BR      /* backward right edge */
EDGE    BL      /* backward left edge */
REGION  R       /* forward right region */
REGION  L       /* forward left region */
int     visit   /* number of visit */
int     length  /* number of length */
```

Fig. 5 Data type of EDGE

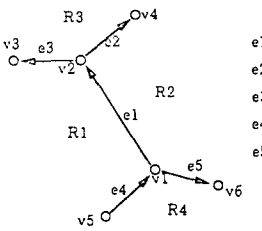
벡터화 알고리즘(vectorization algorithm)은 다음과 같다.

```
Vectorization-Algorithm
Input : binary line image ;
        node points;
Output : EDGE[edge-number];

1) i =0 ;
2) Find an unvisited edge;
3) Enqueue( );
4) while( IsNotEmptyQueue( ))
5)      EDGE[i]=Dequeue( ) ;
6)      num-edge=Find-EndPoint-Of-Edge( ) ;
7)      i++ ;
8)      for(k=0; k<num-edge;k++)
                Enqueue( ) ;
9)      go to 2) if there are unvisited edges ;
10)     Delete-Noise-Edge( ) ;
11)     Link-Adjacent-Edge( ) ;
End of Vectorization-Algorithm
```

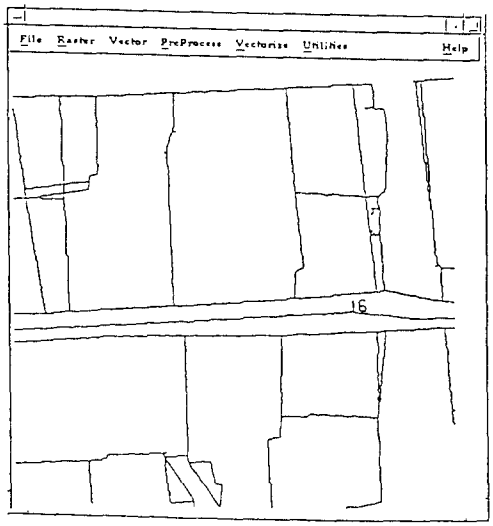
번 줄의 Find—EndPoint—Of—Edge() 함수에서 EDGE의 끝점 P2가 저장된다. Find—EndPoint—Of—Edge() 함수는 EDGE의 시작점 P1에서 출발하여 node point(끝점 P2)에 도착할 때까지 edge를 따라 한번 방문했던 pixel들을 지우면서 진행하고, 끝점 P2에 연결된 edge들의 시작점 P1에 P2값을 대입한다. 이와 같은 과정을 원래의 직선 영상이 모두 지워질 때까지 반복한다. 10)번 줄의 Delete—Noise—Edge() 함수에서는 문자가 직선과 접해 있어서 직선의 edge처럼 분류된 noise edge들을 제거한다. noise edge들은 대부분 시작점 P1과 끝점 P2중 어느 한쪽에 연결된 edge들이 없는 특징을 가지고 있으므로 이와 같은 특징을 이용하여 제거한다. 그러나 noise edge중에서도 문자가 직선과 좁은 영역에서 완전히 겹쳐 있는 경우에 양쪽 끝점에 다른 edge가 연결되어 있는 경우가 대략 1% 내로 존재한다. 이러한 부분은 error correction / detection 프로세스에서 user에게 display 후 user—driven방법으로 수정할 수 있도록 하였다. 11)번 줄의 Link—Adjacent—Edge() 함수에서는 EDGE의 FR, FL, BR, BL의 데이타 영역에 알맞은 값을 대입한다. Fig. 5의 EDGE 모델은 본 논문에서 제시하는 geometric 모델인 DDEL(Doubly Directed Edge List)형태를 나타낸다. DDEL의 형태는 Fig. 6에 나타나 있다.

Fig. 7은 Fig. 2 (c)의 영상에 벡터화 알고리즘을 적용한 결과이다. Fig. 7(a)는 세선화(thinning)된 영상이며, (b)는 세선화된 영상을 벡터 알고리즘을 이용하여 EDGE로 변환하여 얻은 벡터 영상이다.

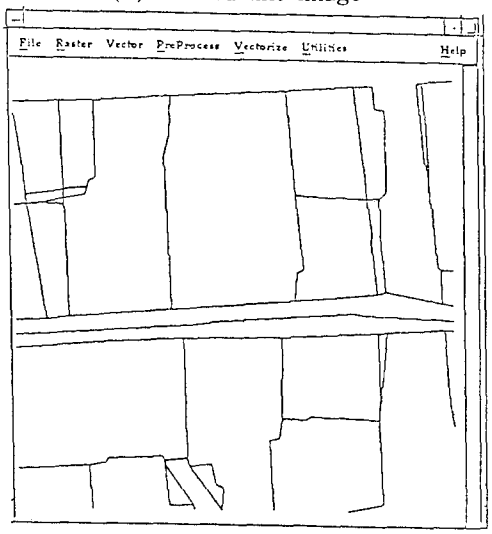


	P1	P2	FR	FL	BR	BL	R	L
e1	v1	v2	e2	e3	e4	e5	R2	R1
e2	v2	v4			e3	e1	R2	R3
e3	v2	v3			e1	e2	R3	R1
e4	v5	v1	e5	e1			R4	R1
e5	v1	v6			e1	e4	R4	R2

Fig. 6 DDEL(doubly directed edge list)



(a) thinned line image



(b) vector image

Fig. 7 Example of vectorization algorithm

오차 수정 (Error detection / correction)

문자 분리 과정을 거친 후에도 문자가 지적도의 직선과 겹쳐 남아 있는 경우에 대부분의 문자는 벡터화 과정에서 제거된다. 그러나 noise edge의 양 끝점이 지적도의 직선과 연결되어 있는 특수한 경우에는 제거가 되지 않으므로 이러한 부분을 error로 간주한다. 또한 지적도를 읽어 들인 영상은 대부분 크기가 매우 크기 때문에 subimage로 분리해서 처리를 할 필요가 있다. 이렇게 커다란 영상을 작은 영상으로 분할하여 처리하고자 할 때에는 영상과 영상의 경계선에 겹쳐져 있는 edge segment가 사라져 버리거나, 세선화 과정에서 경계선을 지나는 edge가 불일치 하거나, 경계선에 위치해 있는 junction point가 사라져 버리는 문제가 새로 발생한다. 이러한 모든 경우에 error가 발생했다고 하며 실험적인 결과로 보면 전체 edge segment의 1%도 훨씬 못 미친다. DDEL구조하에서 이러한 error는 쉽게 발견되기 때문에 user에게 warning을 주어서 벡터 영상 편집기(vector editor)를 이용해 쉽게 수정할 수 있다. 벡터 영상 편집기는 zoom in, zoom out, create, delete, drag, reshape, transform 등의 다양한 기능을 제공하고 있다.

자동 지역 생성 (Automatic region creation)

지적도에서 나타나는 주요 지형 객체로는 점, 선, 그리고 선으로 이루어진 지역(region)을 들 수 있다. 지역(region)은 다각형을 구성하는 edge들을 자동적으로 찾아서 저장하고, 지번, 지목, 지가, 소유주 등의 속성 정보를 가질 수 있게 하여 공간 질의(spatial query) 명령의 수행을 효율적으로 수행할 수 있도록 하였다. 지역(region)의 데이타 구조는 다음의 Fig. 8과 같다.

```
▶ type of REGION

EDGE      *edge-list /* polygon */
String    *address
String    *owner
String    *land-price
String    *usage
```

Fig. 8 REGION의 자료 구조

자동 지역 생성(Automatic region creation) 알고리즘은 다음과 같다.

```
Automatic-Region-Creation
Algorithm
Input : EDGE[edge-number];
Output : REGION[region-number];

1) initialize 'edge[i].visit=0';
2) for i=0 to edge-number do
3)     if(edge[i].visit==0)
4)       March(Clockwise);
5)     else if(edge[i].visit==1)
6)       March(CounterClockwise);
End of Automatic-Region-Creation Algorithm
```

4),6) 번 줄의 March() 함수는 인자로 주어진 방향으로 진행하면서 원래의 시작 edge [i]로 돌아올 때까지 방문하는 모든 edge의 visit영역을 1만큼 증가시키고, REGION.(* edge—list) 데이타 영역에 방문했던 모든 edge들을 기억시킨다. 또한 이 함수는 진행하면서 방문했던 edge들의 왼쪽 region과 오른쪽 region을 나타내는 'L' 과 'R'의 데이타 영역에 Fig.6과 같이 알맞은 region을 가리키도록 한다. 지번, 소유주, 지가, 지목 등의 속성 정보들은 모든 지역을 자동으로 생성한 후에, 그 정보를 이용하여 자동으로 입출력 할 수 있게 하였다.

결 론

본 논문에서는 지적도 응용을 위한 지형 객체 인식 및 데이타 모델링에 대해 설명하였다. 지형 객체를 인식하기 위하여 문자 분리 알고리즘과 지적도의 특성에 맞는 벡터화 알고리즘을 제시하였고, 지적도의 주요 지형 객체인 edge와 region을 자동적으로 인식하기에 적합하도록 데이타를 모델링 하였다. 또한 본 연구에서는 모든 지역을 자동적으로 생성해 주도록 하였기 때문에 각 지역에 관한 정보를 쉽게 관리할 수 있을 뿐 아니라, raster/vector editor를 개발하여 user가 벡터화 과정에서 생기는 error를 쉽게 detection / correction 할 수 있도록 하였다. 지적도는 통신망도, 전력선도, 도로망도 등의 다양한 여러 응용 분야에 기본 도면으로서 사용되기 때문에 앞으로 이에 관한 공간 데이타 베이스 개발을 위해서도 이 분야에 대한 활발한 연구가 요구된다.

참 고 문 헌

F. H. Cheng and W. H. Hsu, 1988, "Parallel Algorithm for Corner Finding on Digital Curves," *Pattern Recognition Letters,* No. 8, pp. 47—53, Aug.

L.A.Fletcher and R.Kasturi, 1990, "A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 12, No. 6, pp. 541—551.

M. T. Musavi et al., 1988, "A Vision—Based Method to Automate Map Processing," *Patern Recognition,* Vol. 21, No.4, pp. 319—326.

S. Bow and R. Kasturi, 1990. "A Graphics Recognition System for Interpretation of Line Drawings," *Image Analaysis Applications,* R. Kasturi and M.M.Trivedi, eds., Marcel Dekker, New York.

T. Pavlidis, 1982, *Algorithms for Graphics and Image Processing,* Computer Science Press, Potomac, MD.

Zhang, T. Y. and Suen, C. Y., 1984, "A Fast Parallel Algorithm for Thinning Digital Patterns." *Com. ACM,* Vol. 27, No. 3, pp. 236—239.

(*본연구는 '93년 산학협동재단 학술연구비 지원에 의해 이루어진 것임)