등고선 추출을 위한 효과적인 다음점 결정 방법

이 진 선† 정 성 종††

요 약

고도행렬로부터 등고선을 추출하는데 있어, 가장 중요한 문제는 하나의 격자에 네개의 통과점이 발생하는 이상 상황(degenerate case)의 해결이다. 본 논문은 이상 상황 검사와 다음점 결정을 동시에 수행하는 방법을 기술한다. 이 방법은 최소 횟수의 배열 인덱싱을 필요로 한다. 또한 통과점을 모서리점의 높이차에 따라 LOWER/HIGHER로 표시하고, 이 정보를 이용하여 보다 적은 횟수의 배열 인덱싱을 필요로 하는 방법을 제안한다. 그리고 통과점을 표시하고 이들을 추적하기에 적합한 자료구조를 구체적으로 기술한다.

An Efficient Method to Determine Next Point for Extracting a Contour Line

Jin Seon Lee† and Seung Jong Chung††

ABSTRACT

In extracting contours from elevation matrix, the most important problem is to solve the degenerate case where four cross point arise in a grid. This paper describes a technique which performs the checking of degenerate case and determination of next point simultaneously. It requires minimum number of array indexing. Also this paper proposes a technique which reduces the number of array indexing by using LOWER/HIGHER information of a cross point designated according to height difference of grid vertices. In addition, we describe a data structure which is proper for representing cross points and tracing them.

1. 서 론

과거에는 지형에 관련된 자료를 일일이 사람의 수작업에 의해 입력했다. 그러나 이제는 여러 다양한 분야에서 지형에 관련된 방대한 양의 자료를 필요로 하게 되어 더이상 수작업에 의존할 수 없는 상황이 되었다. 따라서 대량의 지형 자료를 컴퓨터를 이용하여 빠르고 정확하게 추출하고, 이를 효과적으로 처리하여 저장, 출력하기 위한 많은 연구가 진행되어 왔다[1, 2].

지형에 관한 자료를 컴퓨터에 저장하기 위해서는 공간상에 나타나는 연속적인 지형의 변화를 디지탈 공간에서 수치적으로 표현하는 방법이 필요하며, GIS(Geographical Information System, 지리정보시스템) 에서는 이러한 표현 방법을 수치고도모델(Digital Elevation Model, DEM)이라 한다. 수치고도모델이 일반적으로 사용하는 방법으로는, (1) 2차원 공간상에 등간격으로 분포되어 있는 격자점(grid point)에 높이값이 주어지는 고도행렬(elevation matrix), (2) 임의 분포 데이타, (3) 비정규 삼각분할 (TIN; Triangulated Irregular Network), (4) 스플라인 곡면, 그리고 (5) 같은 높이값을 갖는 점을 이어서 2차원 곡선으로 표현하는 등고선 지도(contour map) 등이 있다. Sabin [3]은 이들 표현 각각의 특징을 기술하고, (1)-(4) 표현을 (5)표현으로 변환하는 등고선 추출 방법들을 잘 정리해 놓았다. Petrie와 Kennie [4]는 이들 지형 모델링 방법의 측량 및 토목 공학에의 응용을 기술하는 논문에서, 각 표현을 설명하고 기존의 등고선 추출 패키지(contouring package)들에

† 정 회 원 : 전북대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 전북대학교 컴퓨터공학과 교수
논문접수 : 1994년 8월 22일, 심사완료 : 1994년 10월 20일

대해 소개하였다. 또한 대부분 상업용 GIS들이 이들 지형자료 표현 기법을 제공하며, 이들 표현간에 상호 변환하는 기능을 제공하고 있다[5].

많은 등고선 추출 알고리즘들이 스플라인 곡면을 다루고 있다[6, 7, 8, 9, 10, 11, 12]. 이들 알고리즘의 주안점은 곡면 방정식을 계산하는 횟수를 어떻게 적게 유지하는냐 하는 점과 어떻게 빨리 계산하느냐 하는 점이다. 이들은 급경사같이 변화가 심한 지역은 보다 많은 점을 생성하고 평지와 같은 지역은 점을 적게 생성하는 적응적(adaptive) 방법을 사용한다. 또한 TIN 구조에서 등고선을 추출하는 방법[5], 임의 분포 데이타에서 추출하는 방법 [13] 등이 있다.

본 논문은 고도행렬에서 등고선을 추출하는 문제를 다룬다. 이 문제는 고도행렬을 탐색하여 특정 높이의 등고선이 통과할 점들을 탐지하여 적당한 자료 구조에 저장하는 문제와, 이들 점을 연결하여 등고선을 구성하는 문제로 구성된다[3]. 첫번째 문제를 위해서는 등고선이 통과할 점을 결정하는 구체적인 조건을 정의해야 하며, 이 조건을 검사하고 탐지된 점을 표시하기에 적당한 자료구조를 정의해야 한다. 두번째 문제에서는 등고선을 추적하는데 있어 다음점을 선택하는 규칙을 정하여야 한다. 또한 추적하는 도중에 하나의 격자에 4개의 교차가 발생하는 이상 상황(degenerate case)이 발생할 수 있다. 이 상황에서는 다음점 후보가 세개이므로 적절한 규칙을 사용하여 합당한 선택을 하여야 한다.

Cottafava 와 Moli [14]는 하나의 격자에 대해 상하좌우 변을 고려하고, 등고선이 들어오는 변이 어느 것이냐에 따라 고정된 조사순서에 의해 나갈 변을 결정한다. 이상 상황에서도 같은 조사 순서를 적용하기 때문에 틀린 결과가 발생할 수도 있다. Snyder [15]는 Cottafava-Moli 방법을 여러가지 면에서 개선하였다. 그중 하나는 이상 상황에서 상변과 하변의 교차점의 위치에 따라 등고선을 연결하는 규칙을 사용한 것이다. Robinson [16]은 이상 상황이 발생한 격자의 중점에서의 높이를 네 모서리점의 높이의 평균으로 계산한 후, 대각선과 등고선의 교차점을 결정하여 그 결과에 따라 등고선을 연결하는 규칙을 사용하였다. Feldman [17]은 격자점 변과의 교차를 고려하지 않고 단지 시계방향으로 돌며 같은 높이값을 갖는 점들을 연결하는 알고리즘과 프로그램을 기술하였다.

본 논문은 등고선이 통과하는 점을 탐지하기 위한 조건을 정의하였다. 이 조건은 단지 통과 여부와 통과점 위치만을 기록하는 기존 방법과 달리, 통과점을 주위 상황에 따라 LOWER와 HIGHER로 구분한다. 이 LOWER/HIGHER 정보는 통과점을 추적할 때 이상 상황 여부 및 다음점 결정을 하는데 유용하게 사용된다. 통과점의 정보를 표시하고 등고선 추적 단계에 사용하기에 적절한 자료구조에 대해서 기존 논문에서는 구체적으로 언급되지 않았는데, 본 논문에서는 이를 구체적으로 기술한다.

통과점을 추적하는 도중, 새로운 격자에 들어가면 우선 이상 상황 여부를 검사해야 한다. 이상 상황이 아닌 경우 다음점 후보가 하나 뿐이므로 다음점이 쉽게 결정된다. 하지만 이상 상황인 경우에는 세개의 후보중에 하나를 다음점으로 선택하기 위한 규칙을 사용하여야 한다. 본 논문의 본문에서는 기존에 발표되어 있는 두가지 규칙을 기술한다. 본 논문은 이상 상황 검사와 다음점 결정을 동시에 수행하는 검사 순서를 제안하고, 이 검사 순서가 최소 횟수의 배열 인덱싱 연산을 필요로 함을 보인다. 또한 통과점을 탐지할 때 기록해 놓은 LOWER/HIGHER 정보를 이용하면, 보다 적은 횟수의 배열 인덱싱으로 이상 상황 검사와 다음점 결정을 할 수 있음을 보인다.

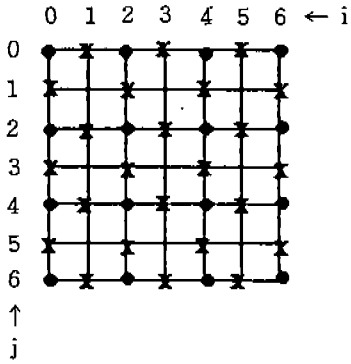
2장에서는 알고리즘에 대해 자세히 기술하며, 3장에서는 실제 지형의 고도행렬을 가지고 실험한 결과를 보여준다. 4장에서 결론을 기술한다.

2. 등고선 생성 알고리즘

보통 고도행렬 화일은 헤더라 불리는 앞 부분과 실제 지형정보를 갖는 뒷 부분으로 구성된다.

헤더에는 한 점에 할당된 비트 수, 공간 해상도, 획득한 지역/날짜/장치, 격자당 실제 길이등의 부가 정보를 갖는다. 우선 고도행렬 화일의 헤더로부터 한 점의 비트 수와 해상도 등을 읽는다. 해상도는 X*Y라 하고, 각 점에 할당된 비트 수에 따라 unsinged char 또는 unsigned int형을 결정한다. 또한 생성하고자 하는 등고선의 최소 높이값과 이웃한 등고선간의 간격을 변수로 입력받는다. 이 값을 사용자가 입력하지 않는 경우 고도 행렬을 조사하여 등고선의 최소 높이값과 최대 높이값을 계산하고, 이 값들을 이용해 등고선이 너무 조밀하거나 드물게 나타나지 않도록 등고선 간격을 자동으로 설정해 준다.

고도자료는 (그림 1)에 있는 이차원 배열 P의 ●-점에 읽어들인다. P 배열의 크기는 (2*X-1)*(2*Y-1)이다. P 배열에서 x-점은 이웃한 (상하로 또는 좌우로 이웃한) 두개의 ●-점 사이에 등고선이 통과하는지의 여부를 나타내는데 사용한다. 우리는 이웃한 두 ●-점 사이에서의 등고선 통과 여부를 결정하여 그 결과를 P배열의 해당 x-점에 표시하여야 한다. P 배열은 통과점을 탐지하고, 통과점의 정보를 저장하고, 나중에 통과점을 추적하는 데 적합한 자료구조이다.



● : 고도 자료점
× : 등고선 통과여부를 나타내는 점

(그림 1) 고도값과 등고선 통과점 정보를 저장하는 자료구조
(Fig. 1) Data structure for storing the height and intersection points.

높이가 C인 등고선의 생성은 크게 두 단계로 이루어진다. 첫째, 이 등고선이 통과할 지점들을 P배열의 x-점에 표시한다. 둘째, P배열을 래스터 순서로 조사하여 등고선 통과지점으로 표시된 점들을 연결하여 체인 코드로 표현되는 등고선을 추출한다. 2.1절에서는 첫번째 단계, 2.2절에서는 두번째 단계를 위한 구체적인 알고리즘을 기술한다.

2.1 등고선 통과 지점 탐지

어떤 x-점에 높이 C의 등고선이 통과하는지 여부를 결정하기 위해 다음 조건을 사용한다. 이 조건식에서 i와 j는 P배열의 인덱스로서 둘다 짝수이다.

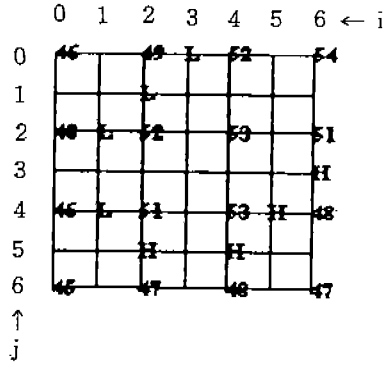
① 좌우로 이웃한 ●-점 사이의 x-점
(짝수 수평 선분상의 x-점):
if(P(i, j) ≤ C < P(i+2, j))
x-점을 LOWER로 표시;
else if(P(i, j) > C ≥ P(i+2, j))
x-점을 HIGHER로 표시;
else x-점을 NOT_CROSS로 표시;

② 상하로 이웃한 ●-점 사이의 x-점
(짝수 수직 선분상의 x-점):
if(P(i, j) ≤ C < P(i, j+2))
x-점을 LOWER로 표시;
else if(P(i, j) > C ≥ P(i, j+2))
x-점을 HIGHER로 표시;
else x-점을 NOT_CROSS 로 표시;

NOT_CROSS는 두 이웃점 사이에 등고선이 지나지 않는다는 것을 나타내며, LOWER나 HIGHER는 두 이웃점 사이에 등고선이 통과한다는 것을 나타낸다. LOWER나 HIGHER 값이 부여된 x-점을 '통과점'이라 부르기로 하자. 기존의 방법에서는 LOWER와 HIGHER 구분없이 단지 그들을 CROSS 로 표시하였다.

예를 들어, P배열에 (그림 2)와 같이 고도자료가 저장되어 있는 경우, C=50 등고선이 통과하는 지점의 정보가 같은 배열 P에 저장되어 있

다. (그림 2)에서 값이 표시되지 않은 지점에는 NOT_CROSS 값이 저장되어 있다. 만일 격자의 모서리 (즉, 고도점)에 등고선이 통과하는 경우에는, 예를 들어 (2,4)점의 높이가 50이라고 하면, 앞의 등고선 통과 여부를 위한 조건식에 따라 (2,3)에 H, (3,4)에 L 값이 주어지게 된다.



L : LOWER H : HIGHER 그외 : NOT_CROSS

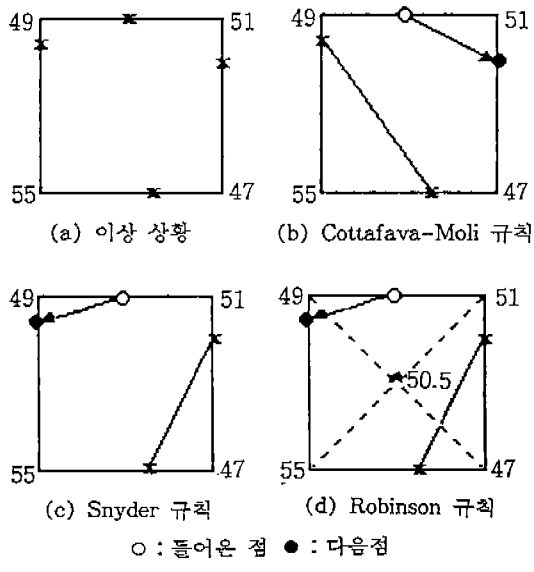
(그림 2) P 배열의 예.
(Fig.2) Example of the matrix P.

2.2 등고선 추출

이 단계는 앞에서 구한 등고선 통과점(P배열에서 LOWER 또는 HIGHER로 표시된 x-점)들을 서로 연결하여 하나의 완전한 등고선을 추출하는 단계이다. 이 단계를 위해서는 앞 단계에서 구한 통과점의 성질을 먼저 파악하여야 한다. 하나의 격자는 다음 세가지 형태 중의 하나이다. 즉 통과점이 하나도 없는 경우, 두개 있는 경우, 그리고 네개가 있는 경우이다. 한개 또는 세개인 경우는 있을 수가 없다 [14]. 추적 도중에 통과점이 하나도 없는 격자로 들어올 수는 없으므로 이 경우는 고려할 필요가 없다. 통과점이 두개인 경우는, 그중 한 점으로 들어왔다고 하면 나머지 점이 한개 뿐이므로 쉽게 다음점을 결정할 수 있다. 통과점이 네개인 경우가 이상 상황(degenerate case)이다.

(그림 3(a))는 이상 상황의 예를 보여준다. 일반성 상실없이 이 격자를 상변의 통과점으로 들어 왔다고 가정하자. 이때 다음점 후보는 하변, 우변, 좌변에 있는 통과점 모두이다. 기존의 방법들은 이러한 경우 다음점을 선택하기 위해 여러가지 규칙들을 사용하였다.

Cottafava와 Moli [14]는 등고선이 들어오는 변이 어느것이냐에 따라 고정된 순서로 조사하여 나갈 변을 결정하는데(예를 들어, 상변으로 들어온 경우에는 우변->하변->좌변의 순서), 이상 상황에서도 같은 조사 순서를 적용한다. 때문에 (그림 3(b))와 같은 결과가 발생한다.



(a) 이상 상황 (b) Cottafava-Moli 규칙

(c) Snyder 규칙 (d) Robinson 규칙

○ : 들어온 점 ● : 다음점

(그림 3) 이상 상황 및 이를 해결하는 규칙 (등고선 높이가 50인 경우).
(Fig. 3) An example degenerate case and several rules to solve it(contour height=50).

Snyder [15]는 상변의 교차점이 하변의 교차점보다 오른쪽에 존재하면 상변-우변, 그렇지 않으면 상변-좌변을 연결하는 규칙을 사용하였다. (그림 3(c))에서 하변의 교차점이 상변 교차점보다 오른쪽에 존재하므로 좌변을 다음점으로 결정한다.

Robinson [16]은 이상 상황이 발생한 격자의 중점에서의 높이를 네 모서리점의 높이를 평균하여 계산한 후, 격자의 대각선과 등고선의 교차점 위치를 가상적으로 생각하여 그 결과에 따라 등

고선을 연결하는 규칙을 사용하였다. (그림 3 (d))에 격자 중점의 평균 높이값과 대각선과의 교차점이 표시되어 있고, 그 결과에 따라 좌변을 다음점으로 결정한다. 추적 도중 새로운 격자점으로 들어가서 다음점을 결정하기 위해서는, 다음과 같은 코드를 수행하여야 한다. 아래의 모든 프로그램 코드는 새로운 격자점을 상변에서 들어갔다는 가정을 하고 있다. 다른 변으로 들어간 경우도 격자를 90도씩 회전하여 비슷하게 생각할 수 있다.

프로그램 코드 1:

```
1) if(하변==CROSS && 좌변==
   CROSS && 우변==CROSS)
   degenerate=TRUE;
   else degenerate=FALSE;
2) if(degenerate) 이상 상황 규칙으로 다음
   점 결정;/* 드물게 발생 */
   else{
   if(좌변==CROSS) 좌변이 다음점;
   else if(하변==CROSS) 하변이 다음점;
   else 우변이 다음점;
   }
```

이 코드에서 단계 1)의 이상 상황 검사는 방문하는 모든 격자점에서 수행해야만 한다. 단계 2)에서 degenerate==TRUE 이어서, 이상 상황 규칙이 사용될 횟수는 비교적 많지 않다.(우리의 실험에서 약 2% 정도이었다.)

이렇게 코딩하는 경우, 단계 1)에서 배열 인덱싱을 세번해야 한다. (크기가 XMAX*YMAX인 2차원 배열 인덱싱 P[x][y]는 해당 요소의 주소를 결정하기 위해, x*XMAX+y 계산을 해야 하고 이는 정수 곱셈 한번과 정수 덧셈 한번을 필요로 한다.) 이러한 이상 상황 검사는 방문하는 모든 격자점에서 수행해야 하고 아주 빈번하게 발생하므로, 이 검사를 빨리 수행할 수 있도록 코딩하는 것은 매우 중요하다.

우리는 이 목적을 위해 다음과 같은 코드를 기술한다. 이 코드에서는 이상 상황 검사와 다음점 결정이 동시에 수행된다.

프로그램 코드 2:

```
if(좌변==CROSS) {
   if(하변==CROSS) 이상 상황 규칙으로
   다음점 결정; /* 드물게 발생 */
   else 좌변이 다음점; /* 우변은 CROSS
   일 수 없다 */ /* 경우 1 */
}
else{/* 이상 상황 아님 */
   if(하변==CROSS) 하변이 다음점;
   /* 우변은 CROSS일 수 없다 */ /*
   경우 2 */
   else 우변이 다음점; /* 경우 3 */
}
```

위 코드에서 경우 1에서는 배열 인덱싱이 좌변과 하변에서 각각 한번씩 총 두번 발생한다. 경우 2와 3에서도 마찬가지로 좌변과 하변에서 각각 한번씩 두번 발생한다. 우리는 이 횟수보다 더 적은 배열 인덱싱으로는 다음점 결정이 불가능하다는 것을 쉽게 알 수 있다.

프로그램 코드 1과 2는 각 변에 통과 여부(CROSS/NOT_CROSS)만을 기록한 기존의 자료구조에서 수행된다. 본 논문에서는 각 변에 대해 통과 여부와 주위 상황(LOWER/HIGHER/NOT_CROSS)을 기록한 우리의 자료구조에서, 보다 적은 횟수의 배열 인덱싱으로 이상 상황 검사와 다음점 결정을 동시에 수행하는 다음 코드를 제안한다.

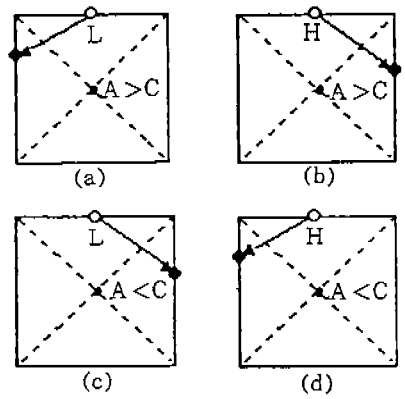
프로그램 코드 3:

```
1. v=하변값;
   /* 들어온 변의 맞은 변을 봄 */
   /* p는 들어온 변의 값으로서, 이전 격
     자에서 결정되어 있음 */
2. if(p==v)
   {하변이 다음점; /* p값은 그대로 */}
   /* 경우 1 */
3. else if(v==NOT_CROSS)
   {/* 이상 상황 아님 */
```

```
4. if (좌변 !=NOT_CROSS)
   {좌변이 다음점; /* p값은 그대로 */}
   /* 경우 2 */
5. else{우변이 다음점; p=~p;}
   /* 경우 3 */}
6. else 이상 상황 규칙으로 다음점 결정;
   /* 드물게 발생 */
```

라인 1에서 하변값을 알기 위해 배열 인덱싱을 한번한다. 라인 2의 v==p 조건은 상변과 하변이 같은 값을 갖는 지, 즉 둘 다 LOWER 또는 둘 다 HIGHER를 갖는 지를 검사한다. 이 조건이 만족되면 좌변과 하변은 NOT_CROSS일 수 밖에 없으므로 하변을 다음점으로 결정한다. (이상 상황에서 상변과 하변이 같은 값을 가질 수 없다는 사실은 쉽게 알 수 있다. 상변과 하변이 같은 값을 가지면 이상 상황이 아니고, 때문에 좌변과 우변은 NOT_CROSS이다.) 이때 p값은 v값과 같으므로 변경할 필요가 없다. 라인 3의 조건이 만족되면 좌변과 우변중 하나가 통과점이다. 라인 4에서 좌변을 먼저 검사하여 NOT_CROSS가 아니면 (즉 통과점이면), 좌변을 다음점으로 한다. 이때 상변과 좌변의 값은 같을 수 밖에 없으므로, p값을 변경할 필요가 없다. 라인 5에서는 우변을 검사해 볼 필요없이 우변을 다음점으로 결정하면 된다. 이때 상변의 값과 우변의 값은 다를 수 밖에 없으므로, p가 LOWER면 HIGHER로 HIGHER면 LOWER로 변경해 준다(p=~p). 결국 프로그램 코드 3은 경우 1에서는 한번, 경우 2와 3에서는 두번의 배열 인덱싱을 필요로 하여 프로그램 코드 2보다 빠르게 동작한다.

이제 위의 코드들에서 이상 상황이 발생한 경우, 이를 어떻게 해결할 지를 살펴보자. 우리는 Snyder규칙과 Robinson규칙이 동일하다고 증명은 하지 못했지만, 많은 상황을 설정하여 비교하여 보았는데 이들 경우에는 모두 동일하게 동작하였다. 본 논문에서는 계산 시간면에서 유리한 Robinson 규칙을 사용한다. 네 모서리의 높이값을 ht, hr, hb, hl이라 하면, A=(ht+hr+hb+hl)/4을 계산하여 현재 등고선 높이 C와 비교한다. 예를 들어, A>C이고 상변의 값이 LOWER이면 좌변을 다음점으로 결정한다. 나머지 경우를 포함하여 다음점 결정 방법을 (그림 4)에서 설명한다.



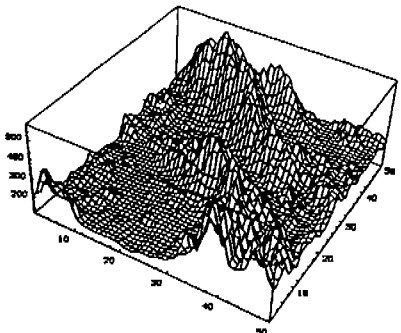
(그림 4) Robinson규칙과 LOWER/HIGHER정보를 이용한 다음점 결정.

(Fig. 4) Determination of next point using Robinson rule and LOWER/HIGHER information.

3. 실험 결과 및 분석

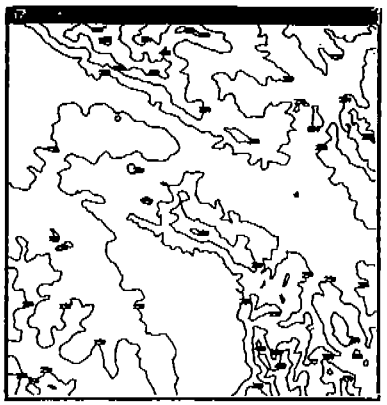
실험 환경은 다음과 같다. 하드웨어로는 SUN SPARC station 10 시스템을 사용하였고, X 윈도우 환경에서 X 라이브러리 함수들과 C 언어를 사용하여 프로그래밍하였다. P배열은 2 바이트 정수 형인 short int형을 사용하였다. 디스플레이 화면은 600*600이다. 실험 자료로는 두개의 지형 고도행렬을 사용하였다. (그림 5(a))의 고도행렬은 200*200 해상도를 갖고, 높이 범위는 약 100m~600m 이다. (그림 6(a))는 295*288의 해상도를 갖고, 높이 범위는 약 70m~220m 이다.

(그림 5(a))에 대해서는 100m와 150m 간격으로 등고선 지도를 추출하였다. (그림 5(b))는 100m 간격으로 추출한 등고선 지도이고, (그림 5(c))는 150m 간격으로 추출한 등고선 지도이다. 프로그램 수행 시간은 100m 간격과 150m 간격에 대해 각각 평균 7.5초와 4.82초이다.



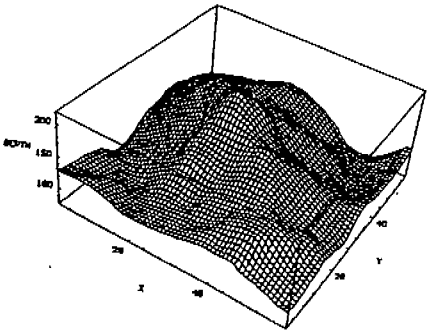
(a) 예제 고도 행렬

(b) 등고선 간격=150m

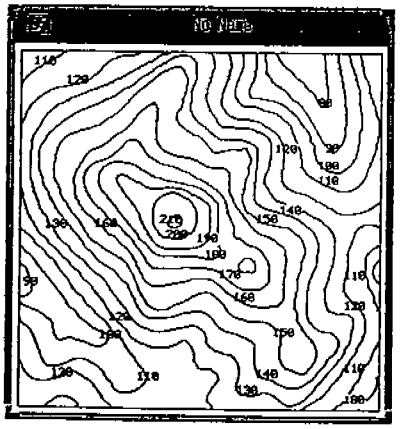


(c) 등고선 간격=100m

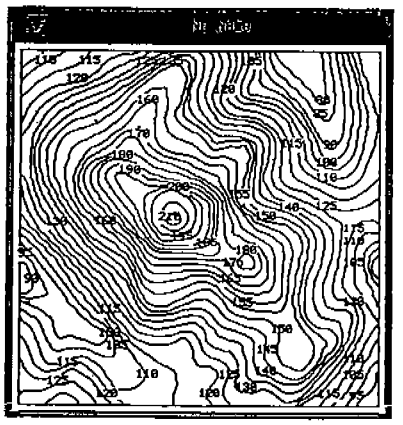
(그림 5) 예제 고도행렬 및 추출된 등고선 지도.
(Fig. 5) Example elevation map and extracted contour map



(a) 예제 고도 행렬



(b) 등고선 간격=10m



(c) 등고선 간격 = 5m

(그림 6) 예제 고도행렬 및 추출된 등고선 지도.
(Fig. 6) Example elevation map and extracted contour map.

(그림 6(a))에 대해서는 5m와 10m 간격으로 등고선 지도를 추출하였다. (그림 6(b))는 5m 간격으로 추출한 등고선 지도이고, (그림 6(c))는 10m 간격으로 추출한 등고선 지도이다.

추출된 등고선은 다음과 같은 방법으로 화면에 디스플레이하였다. 먼저 등고선의 시작점 좌표에서 출발하여, 체인코드를 따라가며 통과점을 화면에 그린다. 각 통과점을 정확한 위치에 그리기 위해, 그 통과점에 이웃한 두 ●-점의 높이값을 P배열에서 찾아, 이 두개의 높이값에 따라 통과점의 실수 좌표를 선형 보간한다. 이 실수 좌표를 화면의 크기를 고려하여 화면 좌표로 변환한다. 이와 같이 구해진 각점의 화면 좌표를 제어점으로 사용하여 직선 또는 큐빅 스플라인 곡선으로 이어서 화면에 디스플레이하였다. 직선을 이용한 방법은 시간이 적게 걸린다는 장점이 있는 반면에 시각적으로 부자연스럽다는 단점을 지닌다. 이러한 점을 보완하기 위하여 곡선을 이용하며, (그림 5)와 (그림 6)의 등고선은 큐빅 스플라인으로 디스플레이한 것이다 [18].

한편 등고선의 높이값에 해당하는 문자열은 등고선을 따라가며 일정한 간격마다 표시하였으며, 이미 표시된 문자열과 겹치지 않도록 하였다.

4. 결 론

많은 GIS시스템들은 여러 형태의 수치고도모델의 표현 기법 및 이들간의 상호변환 기능을 제공해 준다. 이중 고도행렬에서 등고선을 추출하는 문제는 빈번히 발생한다.

고도행렬로부터 등고선을 추출하는 문제에서 가장 중요한 문제는 하나의 격자에 네개의 통과점이 발생하는 이상 상황에서 다음점을 어떻게 결정하느냐에 있다. 본 논문에서는 이상 상황 검사와 다음점 결정을 동시에 수행하는 방법을 기술하였다. 이 방법은 최소 횟수의 배열 인덱싱을 필요로 한다.

또한 통과점을 상하 또는 좌우 높이차에 따라 LOWER/HIGHER 로 표시하고, 이 정보를 이용하여 보다 적은 횟수의 배열 인덱싱을 필요로 하는 방법을 제안하였다. 그리고 통과점을 표시하고 이들을 추적하기에 적합한 자료구조를 구체적으로 기술하였으며, 실제 지형의 고도 행렬로 수행한 실험 결과를 제시하였다.

앞으로의 연구 문제로는 추출된 등고선을 디스플레이 할때, 등고선의 곡률에 따라 적응적(adaptive)으로 큐빅 스플라인을 적용하는 문제가 있다. 즉 곡률이 충분히 큰 지역에서는 직선으로 연결하고 그렇지 않은 지역은 큐빅 스플라인을 적용하여 속도 및 미적인 면을 모두 만족시킬 수 있는 방법을 연구할 필요가 있다.

참 고 문 헌

[1] P. A. Burroughs, *Principles of Geographical Information Systems for Land Resources Assessment*, Oxford University Press, 1986.

[2] 유근배, 지리정보론, 상조사, 1993.

[3] M. A. Sabin, "Contouring-the state of the art," *Fundamental Algorithms for Computer Graphics*, R. A. Earnshaw ed., Springer-Verlag, pp. 411-482, 1985.

[4] G. Petrie and T.J. Kennie, "Terrain modelling in surveying and civil engineering," *Computer-aided Design*, Vol. 19, No. 4, pp. 171-187, May 1987.

[5] ESRI Inc., *ARC/INFO User's Guide: Surface Modelling with TIN*, 1991.

[6] R. Sibson and G. D. Thomson, "A seamed quadratic element for contouring," *The Computer Journal*, Vol. 24, No. 4, pp. 378-382, 1981.

[7] A. Preusser, "Computing contours by successive solution of quintic polynomial equations," *ACM Transactions on Mathematical Software*, Vol. 10, No. 4, pp. 463-472, December 1984.

[8] K. G. Suffern, "Contouring functions of two variables," *The Australian Computer Journal*, Vol. 16, No. 3, pp. 102-106, August 1984.

[9] C.S. Petersen, "Adaptive contouring of three-dimensional surfaces," *Computer Aided geometric Design*, Vol. 1, pp. 61-74, 1984.

[10] G.Sewell, "Plotting contour surfaces of a function of three variables," *ACM Transactions on Mathematical Software*, Vol. 14, No. 1,pp. 33-44, March 1988.

[11] R. R. Dickinson, R. H. Bartels, and A. H. Vermeulen, "The interactive editing and contouring of empirical fields," *IEEE Computer Graphics and Applications*, Vol. 9, No. 3, pp. 34-43, May 1989.

[12] 한순흥, "와이어 프레임 형태의 등가면의 생성과 조작을 위한 O(n) 알고리즘," 1991년도 한국정보과학회 가을 학술발표 논문집, 제18권, 제2호, pp. 707-710, 1991.

[13] I. P. Shagen, "Automatic contouring from scattered data points," *The Computer Journal*, Vol. 25, No. 1, pp. 7-11, 1982.

[14] G. Cottafava and G.E. Moli, "Automatic contour map," *Communications of the ACM*, Vol. 12, No. 7, pp. 386-391, July 1969.

[15] W. V. Snyder, "Algorithm 531:contour plotting," *ACM Transactions on Mathematical Software*, Vol. 4, No. 3, pp. 290-294, September 1978.

[16] A. H. Robinson, et al., *Elements of Cartography*, Fifth eds., John Wiley & Sons Inc., 1984.

[17] T. Feldman, "Generating isovalue contours from a pixmap," *Graphics GEMS III*, Academic Press, 1992.

[18] D. F. Rogers and J. A. Adams, *Mathematical Elements for Computer Graphics*, 2nd Eds., McGraw-Hill, 1990.

이 진 선

1985년 전북대학교 자연과학대학 전산통계학과(이학사)
1988년 전북대학교 대학원 전산통계학과(이학석사)
1992~현재 전북대학교 대학원 컴퓨터공학과 박사과정
1988~1991년 한국전자통신연구소 연구원
관심분야 : 컴퓨터 그래픽스, 영상 처리

정 성 종

1975년 한양대학교 공과대학 전기공학과(공학사)
1981년 Houston 대학교 전자공학전공(공학석사)
1984년 Houston 대학교 전자공학전공(공학박사 수료)
1986~1988년 충남대학교 전자공학과 전산공학전공(공학박사)
1990~1991년 Penn. State University 객원 교수
1985~현재 전북대학교 컴퓨터공학과 교수
관심분야 : 컴퓨터 그래픽스, 영상 처리, 영상 인식