碩士學位論文

指導敎授 李斗成

지하 정보 3차원 시각화 시스템 구축에 관한 연구

A Study on Development of 3-D Visualization System for Subsurface Information

2002年 8月 日

漢城大學校 大學院

情報시스템工學科

情報시스템專攻

金 賢 奎

碩士學位論文

指導敎授 李斗成

지하 정보 3차원 시각화 시스템 구축에 관한 연구

A Study on Development of 3-D Visualization System for Subsurface Information

위 論文을 工學 碩士學位論文으로 提出함

2002年 8月 日

漢城大學校 大學院

情報시스템工學科

情報시스템專攻

金 賢 奎

김현규의 공학석사학위논문을 인정함

2002年 8월 일

심사위원장 (인)

심사위원 (인)

심사위원 (인)

목차

표 목차

그림 목차

I. 서론

「시각화 (visualization)」란 일반적으로 사람의 눈으로 인지할 수 없는 정보를 인간의 가장 발달된 감각인 시각에 적합한 형태로 변환시키는 것을 의미한다. 과학적 자료 시각화는 현대의 발전된 컴퓨터 그래픽을 이용하여 과학적 자료를 통찰 (洞察)하고자 하는 시도이다. 이는 주로 공학, 의학, 기상학, 지질학 등의 분야에서 현실의 객체나 현상에 대한 도식적 표시에 더 많은 정보와 상호 작용성 (interactivity)을 부여하기 위해 사용되어왔다 (Fuller and Francis, 1996).

과학적 자료 시각화의 핵심 가운데 하나는 자료의 속성을, 자료가 가진 정보의 내용을 가장 효과적으로 전달할 수 있는 도식적 요소 (graphical primitive)로 표현하는 것이다. 이를 위한 시각화의 기법은 매우 다양하나, 시각화의 대상이 되는 자료의 속성을 이해함으로써 적절한 시각화 방법을 선택할 수 있다.

한편 시각화 시스템 구조와 수단은 도구 지향적 (tool-oriented) 및 객체 지향적 (object-oriented) 프로그래밍에 의한 접근 방법이 일반적이다. 도구 지향적 접근 방식은 과학적 시각화 시스템이 영상만을 제공하는 기본적인 역할을 넘어 완전한 자료 분석 공정을 지원해야 한다는 개념까지 포함하며 각 시각화 및 분석 도구는 객체 지향적으로 프로그래밍 됨을 의미한다 (Manssour *et. al.*, 1997). 아울러 정보 시각화 시스템은 자료의 공간적 분석 및 가공 기능과 함께 검색·생성된 정보의 저장 기능까지 갖춘 DBMS (database management system)가 요구된다.

본 연구의 대상 자료는 지하 공간 정보로서 두 가지 유형으로 구분할 수 있다. 하나는 3차원 격자 형태의 정량적 자료이며, 두 번째는 선형 배열로 구성된 정량적, 정성적 정보를 모두 가지고 있는 자료이다. 본 연구에서 구현하고자 하는 것은 두 가지 유형의 자료를 사용자의 요구에 부응하는, 효과적인 가공 및 시각화 기능을 가진 시스템으로 개발하는 것이며, 이를 위하여 비교적 적은

비용으로 지하 정보를 체계적으로 관리하고 분석할 수 있는 시스템을 개발하고자 하였다.

본 연구에서 설정한 시스템의 기능은 3차원 공간 정보를 자유롭게 탐색하고 분석할 수 있게 하며, 대화형 인터페이스를 통해 사용자가 자료로부터 정보를 추출하고 재구성하는 것이다. 또한, 추출된 정보로부터 새로운 모델을 생성할 수 있는 기능이 부가되며, 다양하고 방대한 정보의 관리를 위해 데이타베이스 시스템이 기저를 이루도록 한다.

이러한 시스템을 독자적으로 제작하는 것은 매우 방대한 과제이므로 본 연구에서는 범용 지리 정보 시스템 (GIS) 소프트웨어 가운데 하나인 「ArcView」를 기반으로 하여 지하 정보 시각화를 구현할 수 있도록 특화하였다. 프로그래밍은 ArcView에 내장된 객체 지향 스크립트 언어인 「Avenue」를 주로 사용하였으며 부분적으로 C 언어를 이용하여 코딩하였다.

II. 3차원 격자배열 자료 시각화

지하를 대상으로 한 정보는 직접 접근에 의한 획득이 어려우므로 대개 원격 탐사의 일종인 지구 물리 탐사에 의한 것이 많다. 여러 가지 물리 탐사 방법에 의해 처리된 3차원 자료는 일반적으로 일정한 간격을 가진 격자 배열 상태로 주어지며, 지하 매질의 물리적 특성에 대한 정보를 가지고 있다. 이러한 물리적 특성 또는 3차원 공간 정보를 이해하고 분석하기 위한 시각화 방법이 volume visualization이다.

Volume visualization을 위한 알고리듬은 크게 direct volume rendering (DVR)과 surface fitting (SF), 그리고 interactive rendering을 위한 방법 등이 있다 (Owen, 1999). DVR은 3차원 개체의 원소를 아무런 기하 요소 (geometric primitive)의 매개 없이 직접 화면에 표시하는 방법으로서 기체나 유체와 같은 무정형 (amorphous) 개체의 시각화에 주로 사용된다. SF 방법은 일정한 값을 가진 점들로부터 등치면 (iso-surface)을 구성하는 것이다. 본 연구에서와 같이 자료의 시각적 탐색이 주 목적인 경우에는 DVR이나 SF 보다는 좀더 단순하고 빠르게 정보를 제공하는 대화식 방법 (interactive methods)이 적당하다.

대화식 방법에는 wireframe contours, multiplanar reprojection, sweeping plane 등이 있다. 이 가운데 wireframe contours는 표현되는 정보가 매우 제한적이므로 본 연구에는 적합하지 않다. Multiplanar reprojection은 slicing이라고도 하며 3차원 자료를 다수의 평면에 투영하여 표시하는 것이다. Sweeping plane은 투명한 상태로 되어있는 자료에 하나 또는 다수의 평면을 통과시키면서 평면에 접하는 원소만을 도시하는 방법이다. 본 연구에서는 multiplanar reprojection 방법을 이용하여 자료를 시각화하고 sweeping plane 방법으로 자료를 탐색하는 복합적 방식을 적용하였다.

1. 단면 시각화

공간 정보의 3차원적 시각화를 위하여 ArcView에서 사용할 수 있는 객체 모델 (object model)은 Shape, TIN (triangulated irregular network), Grid 등이 있으나 TIN과 Grid는 2차원 표면에 높이를 부여한 것에 불과하므로 수직면이나 입방체와 같이 온전한 3차원 개체는 형상화할 수 없다. 따라서 본 연구에서는 Shape만을 이용하여 시각화를 수행하였다.

3차원 물리 탐사 자료의 시각화에 많이 사용되는 방법은 sweeping plane이다. 본 연구에서는 sweeping plane을 구현하기 위해 단면화 (slicing) 기법을 수용(受用)하였다. 즉, 평면 (plane)이 움직이는 축 방향으로 모든 자료에 대하여 단면 (slice)의 배열을 생성한 뒤 특정 위치에 해당하는 단면만을 도시하도록 하였다. 그림 1과 표 1에 단면 생성 과정에 대한 흐름도 (flowchart)와 의사 코드 (pseudo-code)를 제시하였다. 이러한 방식은 단면을 만드는데 시간이 많이 걸린

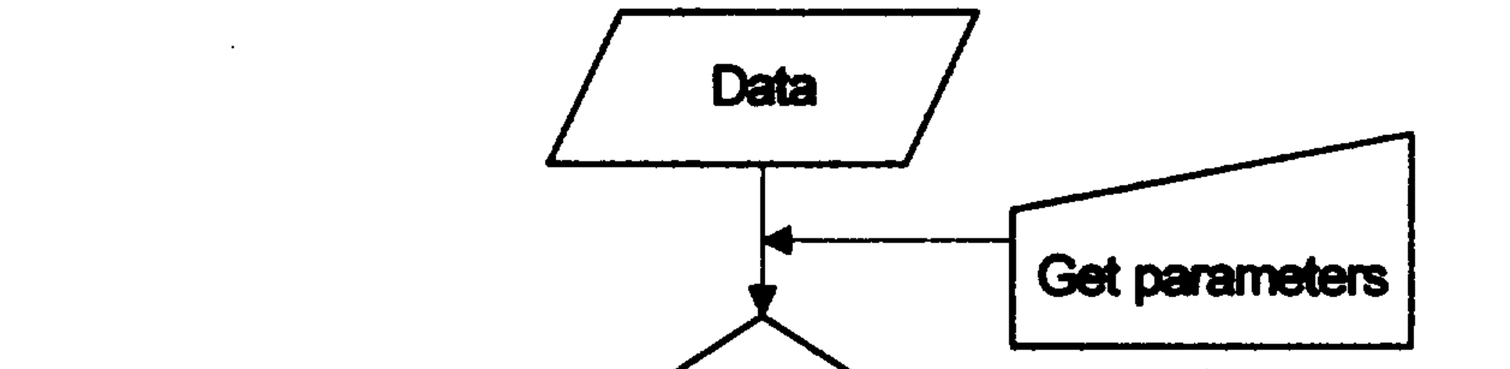

Fig. 1 The flowchart of generating the slice polygons.

Table 1 A pseudo-code for constructing slice shapefiles.

```
INPUT fx: x coordinate of the first sample,
      dx: interval of x coordinate,
      nx: number of samples for x dimension
      fy: y coordinate of the first sample,
      dy: interval of y coordinate,
      ny: number of samples for y dimension
      fz: z coordinate of the first sample,
      dz: interval of z coordinate,
      nz: number of samples for z dimension

For each iy = 0, ···, ny − 1
    For each ix = 0, ···, nx − 1
        For each iz = 0, ···, nz − 1
            (Set coordinates of current sample.)
            x = fx + ix × dx
            y = fy + iy × dy
            z = fz + iz × dz
            If user selects x-plane
                (Create four vertices with constant x coordinate.)
                p1 = (x, y − dy / 2, z − dz / 2)
                p2 = (x, y + dy / 2, z − dz / 2)
                p3 = (x, y + dy / 2, z + dz / 2)
                p4 = (x, y − dy / 2, z + dz / 2)
                (Create a rectangle polygon on y-z plane.)
                Px = (p1, p2, p3, p4)
                Set attributes for Px.
            Else if user selects y-plane
                (Create four vertices with constant y coordinate.)
                p1 = (x − dx / 2, y, z − dz / 2)
                p2 = (x + dx / 2, y, z − dz / 2)
                p3 = (x + dx / 2, y, z + dz / 2)
                p4 = (x − dx / 2, y, z + dz / 2)
                (Create a rectangle polygon on x-z plane.)
                Py = (p1, p2, p3, p4)
                Set attributes for Py
            Else if user selects z-plane
                (Create four vertices with constant z coordinate.)
                p1 = (x − dx / 2, y − dy / 2, z)
                p2 = (x + dx / 2, y − dy / 2, z)
                p3 = (x + dx / 2, y + dy / 2, z)
                p4 = (x − dx / 2, y + dy / 2, z)
                (Create a rectangle polygon on x-y plane.)
                Pz = (p1, p2, p3, p4)
                Set attributes for Pz
            End
        End
    End
End
```

다는 단점이 있으나 일단 단면이 만들어지고 나면 평면의 이동을 신속히 할 수 있고 모든 샘플 위치에서의 속성 정보가 단면 shapefile에 저장되므로 정보의 관리와 분석이 용이하다는 장점이 있다. 그림 2에 시각화된 단면의 예를 도시하였다. 이 자료는 한국석유공사에서 제공한 미국 Boonsville 지역에서의 3차원 탄성파 자료이다 (Gwak and Lee, 2000). 각 평면의 위치 이동은 ArcView의 Dialog Designer (ESRI, 1997)를 이용하여 제작한 슬라이더 대화 상자 (그림 2의 좌상단)를 조작함으로써 이루어진다.

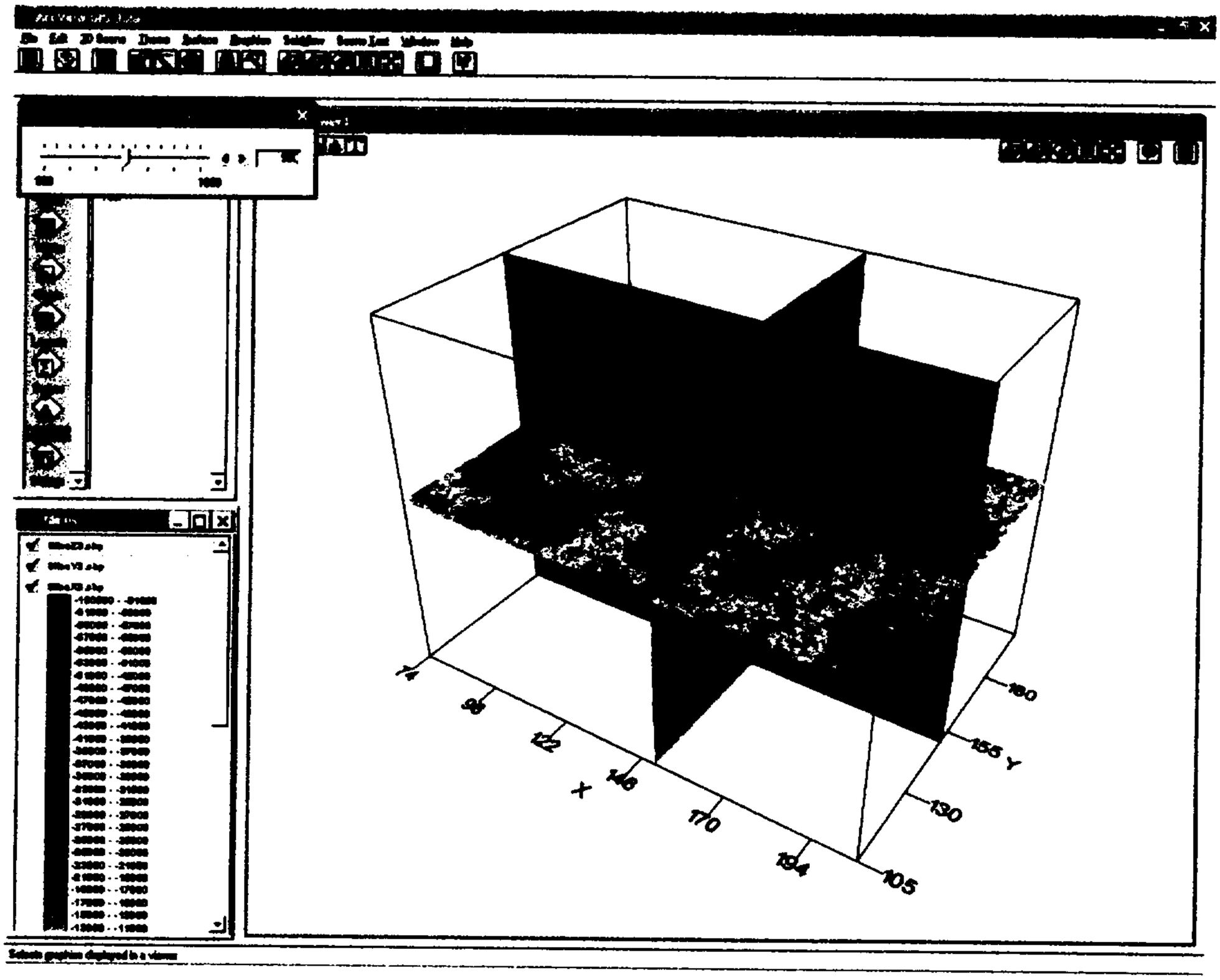

Fig. 2 Three orthogonal cross-sections of the Boonsville 3-D seismic data.

2. 육면체 시각화

3차원 공간 정보를 시각화하기 위한 두 번째 방법으로서 육면체 형태로의 시각화를 수행하였다. 이 방법은 각 샘플을 「복셀 (voxel)」이라고 부르는 작은 입방체로 만들고, 샘플의 속성에 따라 복셀의 색상이나 투명도를 조절하는 것이다. 이렇게 하면 탄성파 자료의 경우 각각의 트레이스는 하나의 복셀 기둥(column)으로 변환된다 (그림 3).

여기서는 단면 시각화 시에 작성한 shapefile의 dBASE 테이블 (ESRI, 1998)을 입력 자료로 사용하며, 육면체의 표면에 해당하는 위치의 샘플들을 복셀 형태의 폴리곤 shape으로 생성한다. 그림 4에 이 방법으로 시각화된 자료의 예를 도시하였다. 육면체의 범위는 그림 좌상단의 대화 상자에 표시되며, 이 곳의 수치를 변경함으로써 표시되는 자료의 범위를 조절할 수 있다.

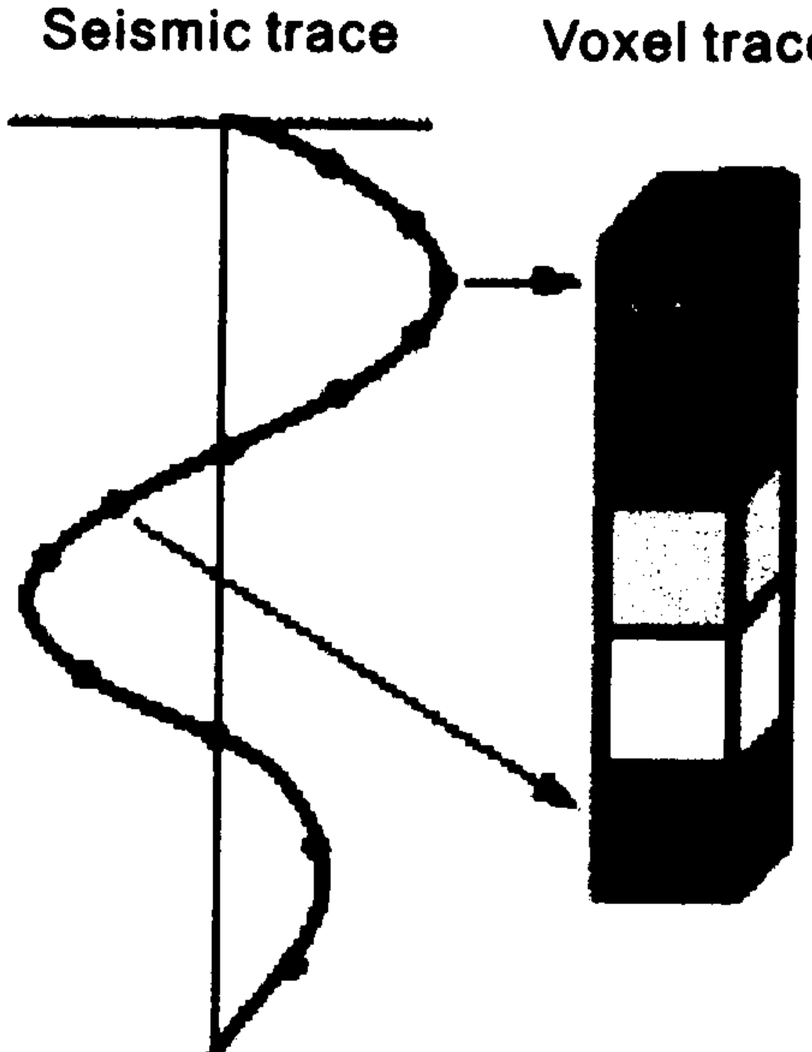

Fig. 3 Converting seismic trace to voxel column. (after Kidd, 1999)

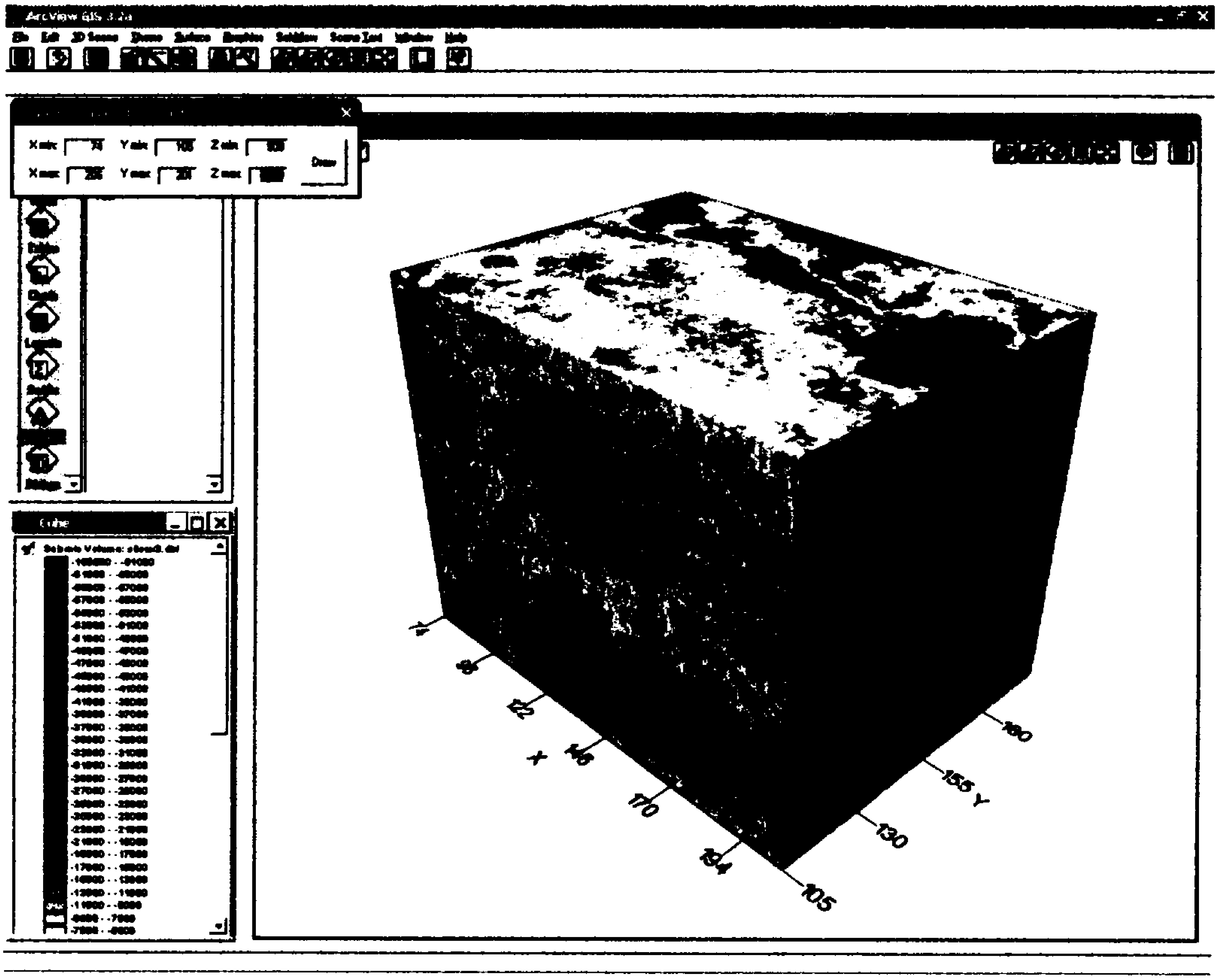

Fig. 4 Three-dimensional view of the Boonsville 3-D seismic data.

III. 선형배열 자료 시각화

지하 정보를 획득하는 다른 방법은 시추공을 이용하여 자료에 직접 접근하는 것이다. 시추공을 이용한 방법 가운데 본 연구에 적용한 자료의 유형은 시추 검층 자료와 시추 주상도 자료의 두 가지이다. 검층 자료와 주상도 자료는 모두 시추공의 경로를 따라 얻어지므로 선형 배열의 형태를 가지고 있다. 이 두 가지 자료는 약간 다른 특성을 가지고 있으므로 구분하여 서로 다른 방식의 시각화를 적용하였다.

1. 시추 검층 자료 시각화

시추 검층 자료는 시추공 안에 측정기를 넣어 일정 간격으로 자료를 획득한 것

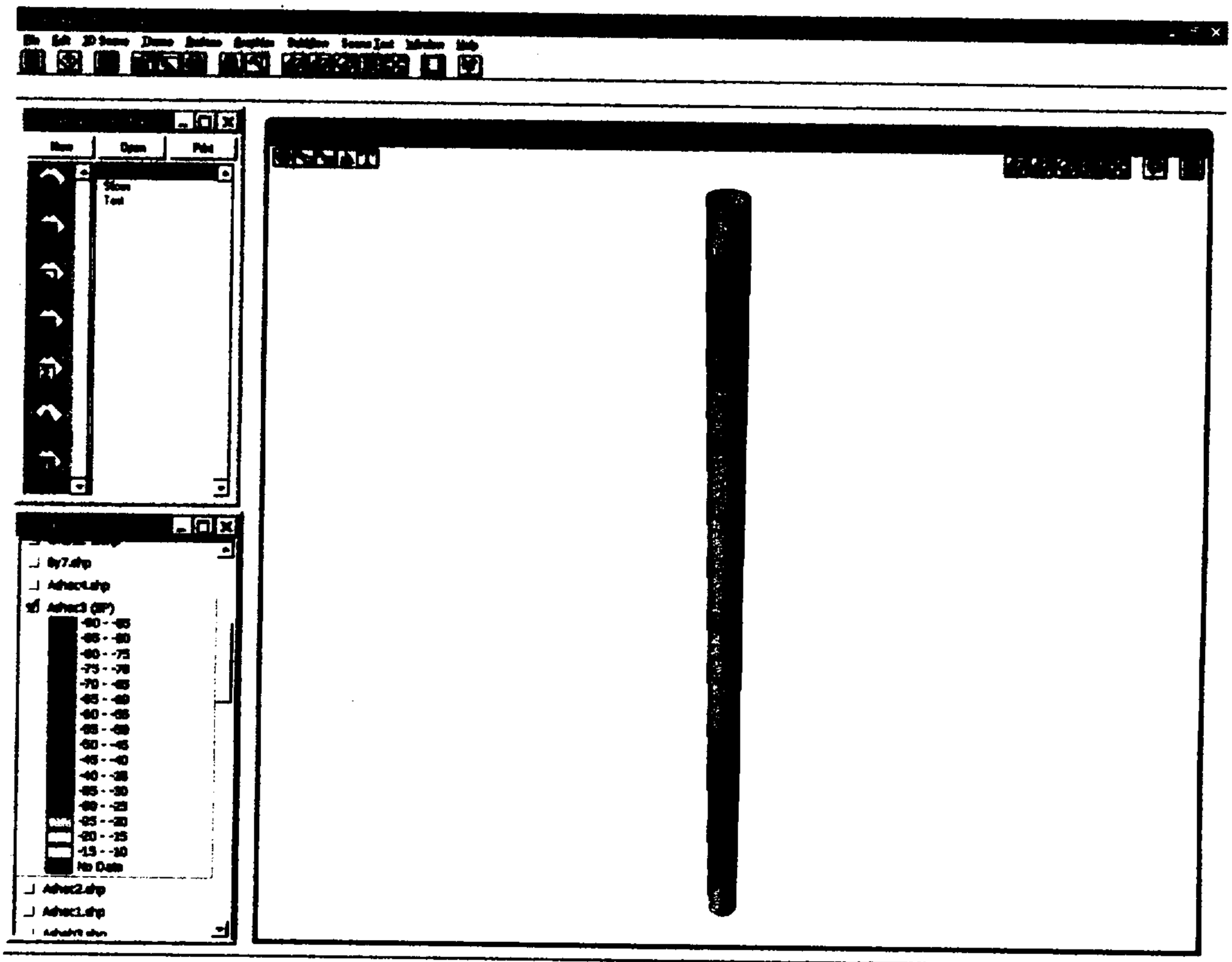

Fig. 5 A well log visualized as a pipe-form shape.

으로서 대개 하나의 시추공에서 여러 종류의 검층 자료를 포함하는 경우가 많다. 그러므로 획득된 자료의 각 샘플은 시추공 경로 상의 한 점에 대한 정량적 정보를 가지고 있다. 본 연구에서는 시추공의 형태에 가깝게 시각화하기 위하여 각 샘플을 샘플 간격만큼의 두께를 가지는 얇은 고리 형태의 shape로 생성하고, 여기에 각 검층 자료의 값을 속성 테이블에 추가하였다. 따라서 한 시추공에 대한 전체적인 자료의 모습은 그림 5와 같이 파이프의 형태를 가지게 되며 검층 값은 색도로서 표시된다. 검층 자료는 수직공에 대해서만 시각화를 수행하였다. 입력 자료의 형식은 단면 시각화에서와 같이 아스키 형식이며, 여기서도 하나의 샘플이 shapefile에서 하나의 레코드가 되므로 DB에 의한 자료 관리가 가능하다.

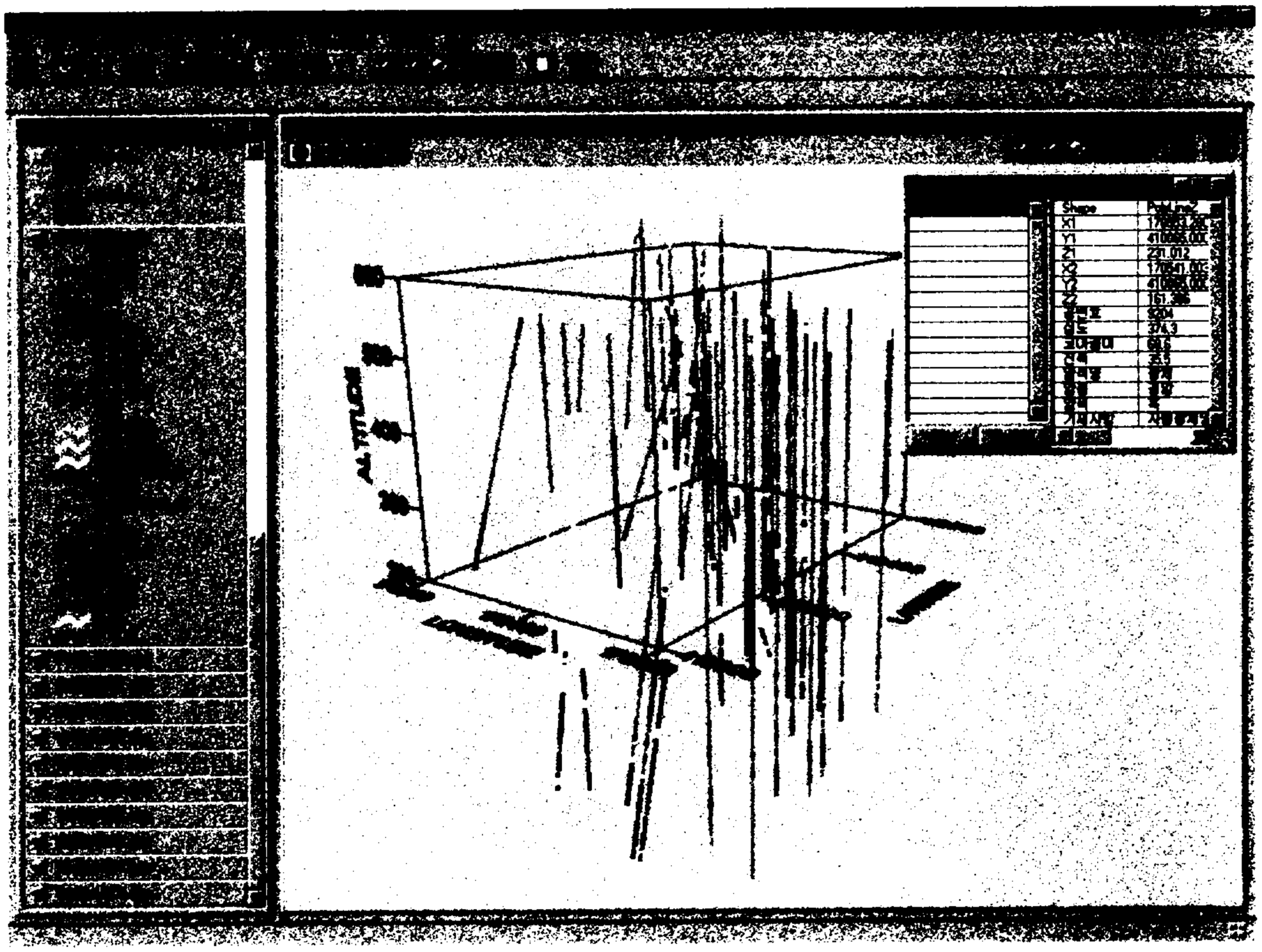

Fig. 6 A 3-D perspective of the vertical and inclined wells with the attributes of a rock interval popped up.

2. 시추 주상도 자료 시각화

시추 주상도 자료는 시추공으로부터 원통형 표본 (core)을 채취한 뒤 암상별로 분석한 기록이다. 따라서 검층 자료와 달리 점이 아닌 특정 구간, 즉 선에 대한 정량적·정성적 정보를 가지고 있다는 특징이 있다. 또한 주상도 자료는 수직공 뿐만 아니라 경사공에 대해서도 시각화였으므로 약간 복잡한 과정을 거쳤다.

먼저 각 시추공 시작점의 3차원 좌표를 측정하였는데, 현장에서 직접 측량한 자료가 없으므로 수치 지도 상에서 좌표를 추출하였다. 이를 위하여 대한광업진흥공사에서 제공한 1:2,000 축척의 전체 시추공 위치도를 스캔하고, 여기에 국립 지리원에서 제작한 수치 지도를 기준 도면으로 사용하여 위치도의 좌표를 설정하였다. 수치 지도의 좌표계는 TM (Transverse Mercator) 도법 (Gauss-Krüger

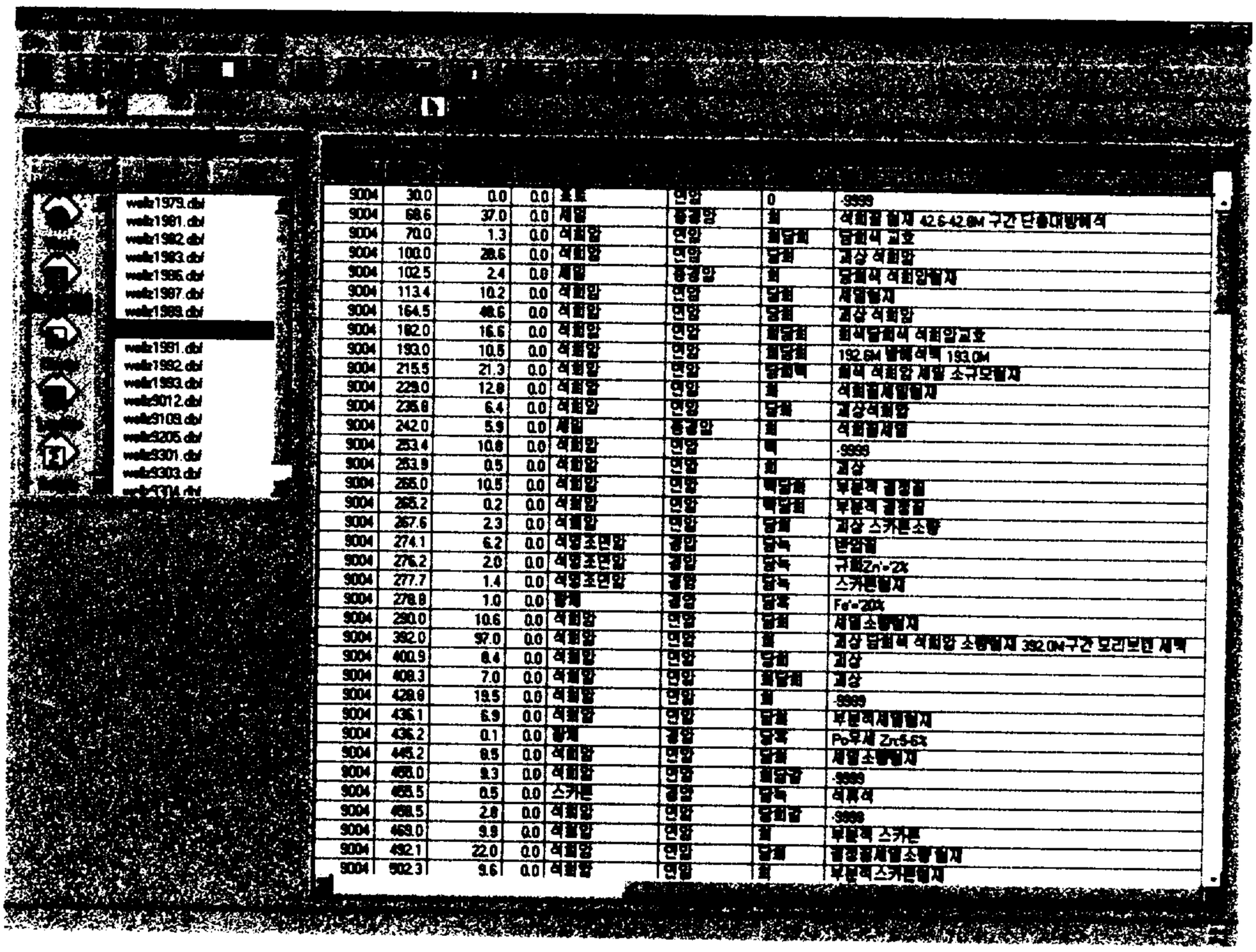

Fig. 7 A database table of core descriptions created by the process of 3-D well shaping.

투영)을 적용하고 있으므로 (국립 지리원, 1998) 본 연구에 사용된 좌표도 모두 이를 따른다.

좌표가 설정된 위치도 영상으로부터 컴퓨터 화면 상에서 디지타이징을 실시하여 모든 시추공들의 좌표를 획득하였다. 이 때 2차원의 위치도 영상만으로 디지타이징 할 경우에는 x, y 좌표만이 표시되므로 z 값을 같이 얻기 위하여 수치 지도를 TIN 형식 (ESRI, 1997)으로 변환하여 위치도 영상과 중첩되게 화면에 도시한 뒤 각 시추공의 삼차원 좌표를 획득하였다.

암상 구간별 정보를 가지고 있는 주상도를 3차원 shapefile로 만들기 위해서는 주상도에서 나타나는 각각의 암상이 하나의 line feature (ESRI, 1996)가 되어야 하고 이들을 연결하여 전체적인 시추공을 완성하게 된다. 하나의 line feature를 만들기 위해서는 두 개의 점이 필요하므로 각 구간마다 시작점과 끝점의 좌표를 계산하여야 한다. 이 계산에 필요한 매개 변수는 시추공 시작점의 좌표와 각 구간의 시작점까지의 거리, 구간의 길이, 시추공의 방위와 경사 등이다. 이 값들을 주상도 자료와 함께 입력 받아 다음 식을 이용하여 계산을 수행하였다.

$$x_2 = x_1 + l\cos\alpha\cos\beta$$
$$y_2 = y_1 - l\sin\alpha\cos\beta$$
$$z_2 = z_1 - l\sin\beta$$

여기서 x_1, y_1, z_1은 구간 시작점의 좌표이고 x_2, y_2, z_2는 끝점의 좌표이다. 또한 l은 구간의 길이이며 α는 방위, β는 경사를 나타낸다.

구해진 두 점으로부터 3차원 line shape을 생성하고 여기에 각 속성 값들을 연결함으로써 하나의 시추공에 대한 정보를 3차원으로 시각화 하였다. 그림 6은 완성된 3차원 시추공으로서, 여기에는 시추 주상도에 기록된 정보가 데이타베이스화되어 속성으로 포함되어 있으므로 (그림 7) ArcView 상에서 각 암석 구간에 대한 정보를 쉽게 확인할 수 있으며 각종 질의 (query)를 통한 분석도 실행할 수 있다.

IV. 공간 정보 해석 결과의 3차원 시각화

1. 특정면 (horizon) 시각화

3차원 탄성파 자료에 의한 지하 특정면의 시각화는 특정 개체로 인지되는 특정 이밴트를 피킹 (picking)함으로써 시작된다. 3차원 자료라 하여도 피킹은 일반적으로 2차원 단면 상에서 이루어지는 경우가 많으므로 본 연구에서도 2차원 환경에서 작업할 수 있도록 프로그램을 개발하였다. 본 모듈에서의 입력 자료는 단면 shapefile의 dBASE 테이블이며, 단면은 Grid 형식으로 생성된다.

탄성파 자료의 피킹에서는 보통 최고점 (peak)이나 최저점 (trough)을 선택하게 되는데, 모든 트레이스 (trace)에 대하여 일일이 작업하는 것은 매우 비효

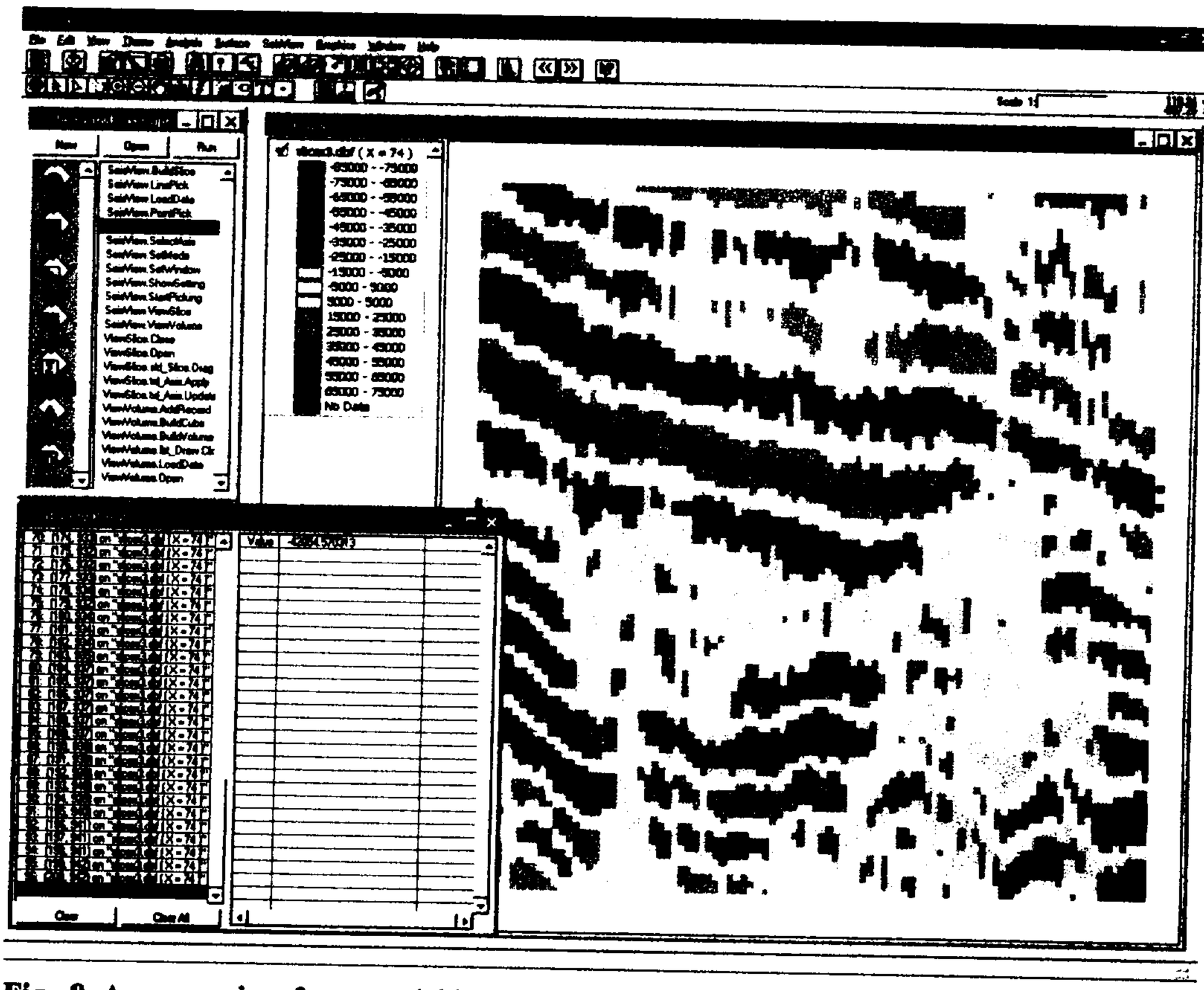

Fig. 8 An example of event picking process.

율적일 뿐 아니라 수작업으로는 정점을 정확히 찾을 수 없다. 따라서 본 연구에서는 사용자가 탄성파 이벤트를 따라 몇 개의 트레이스를 따라 폴리라인(polyline)을 그리면 이 선의 상하로 일정한 윈도우 안에서 최대·최소치를 검색하여 점을 생성하는 유도 피킹 (guided picking) 방식을 적용하였다. 피킹된 결과는 그림 8과 같이 화면에 나타나며 이 점들에 대한 수정도 가능하다. 그림 8에서는 폴리라인 상하로 각각 5개 샘플 이내의 자료 중 최저치를 피킹한 결과이다. 출력 파일은 아스키 형식으로 저장되며, 정의된 패턴들은 ArcView에 의해 3차원 면으로 도시할 수 있다. 그림 9에 Boonsville 자료 (Hardage, 1996)에

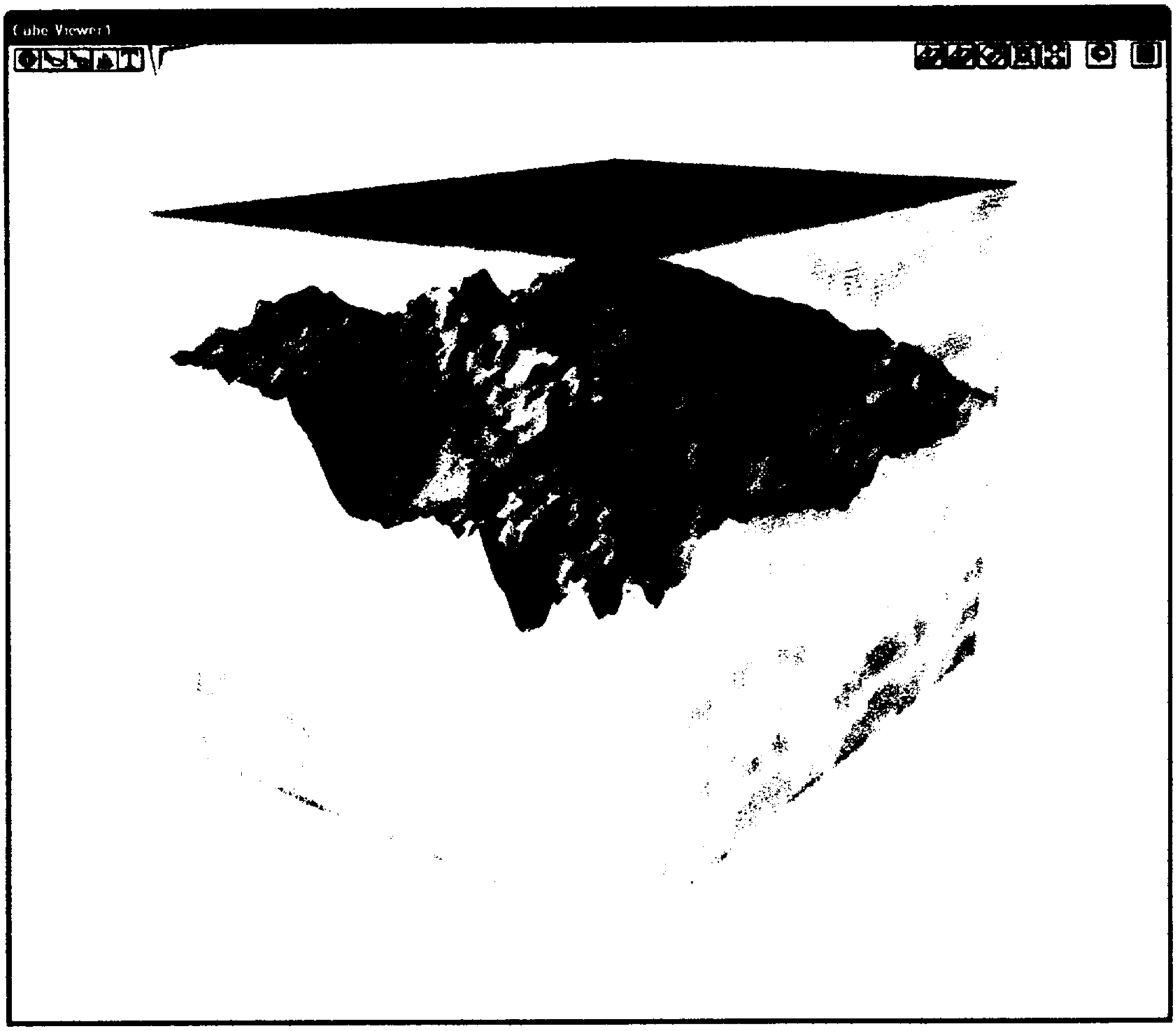

Fig. 9 Three-dimensional view of the top of Davis formation in the Boonsville data.

대해 본 모듈을 적용하여 Davis 면을 시각화한 결과를 도시하였다.

2. 3차원 개체 시각화

그림 10과 같이 이산적인 단면으로 주어지는 정보를 가진 객체로부터 원래의 3차원적 형태를 추론하기 위한 방법으로서 모델링을 도입하였다. 모델링이란 물체를 3차원 컴퓨터 그래픽으로 생성하고 묘사하기 위한 자료 구조나 기법이며, 또는 이를 조작하는 것도 포함한다. 모델링 기법은 여러 가지가 있으나 많이 쓰이는 것으로는 polygonal, bi-cubic parametric patches, constructive solid geometry, spatial subdivision techniques 등이 있다 (Watt, 2000). 본 연구에서 적용한 방법은 가장 일반적으로 사용되며 ArcView에서 유일하게 지원하는 폴리곤 모델링 (polygon modeling)이다. 폴리곤으로 개체를 만드는 방법은 다양하게 개발되

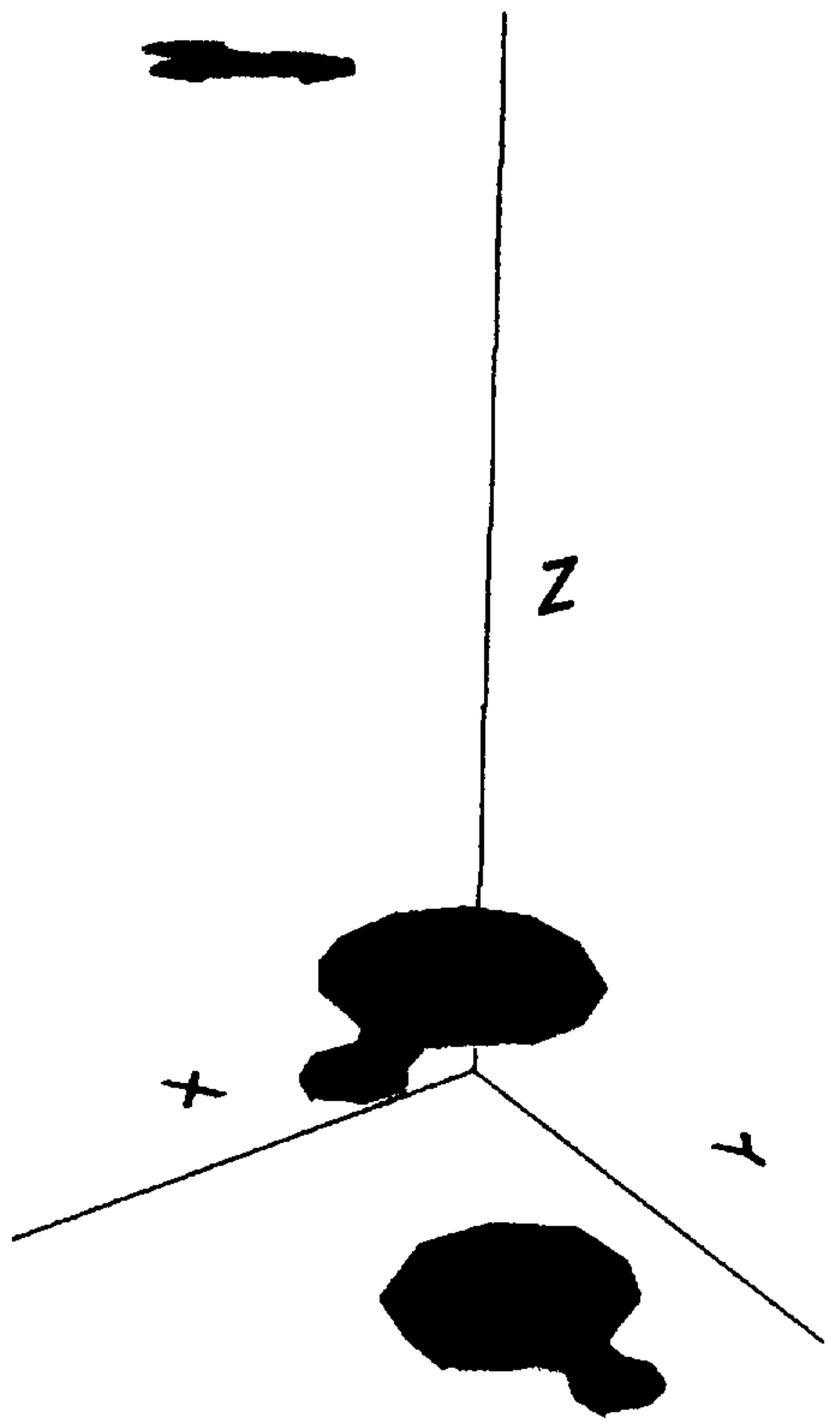

Fig. 10 An example of cross-sections to build a polygonal model by sweeping.

어 있으며, 본 연구에서는 그 가운데 스위핑 (sweeping)을 응용한 단면 연결 방식으로 모델링을 수행하였다.

2.1 개체 모델링

스위핑이란 어떤 단면을 임의의 축 (spine)을 기준으로 하여 이동시켰을 때 나타나는 궤적을 이용하여 면을 생성하는 것이다 (Watt, 2000). 이 때 단면과 축에 여러 가지 변화를 줌으로써 다양한 표면을 만들 수 있다. 본 연구에서는 임의의 모양을 가진 단면들이 임의의 경로를 따라 임의의 간격으로 떨어져 있는 경우, 단면 외곽의 각 점이 이동하는 궤적을 추적하여 면을 생성하였다.

그림 10과 같은 단면들이 주어졌다고 하자. 여기서는 수평 단면들이 수직

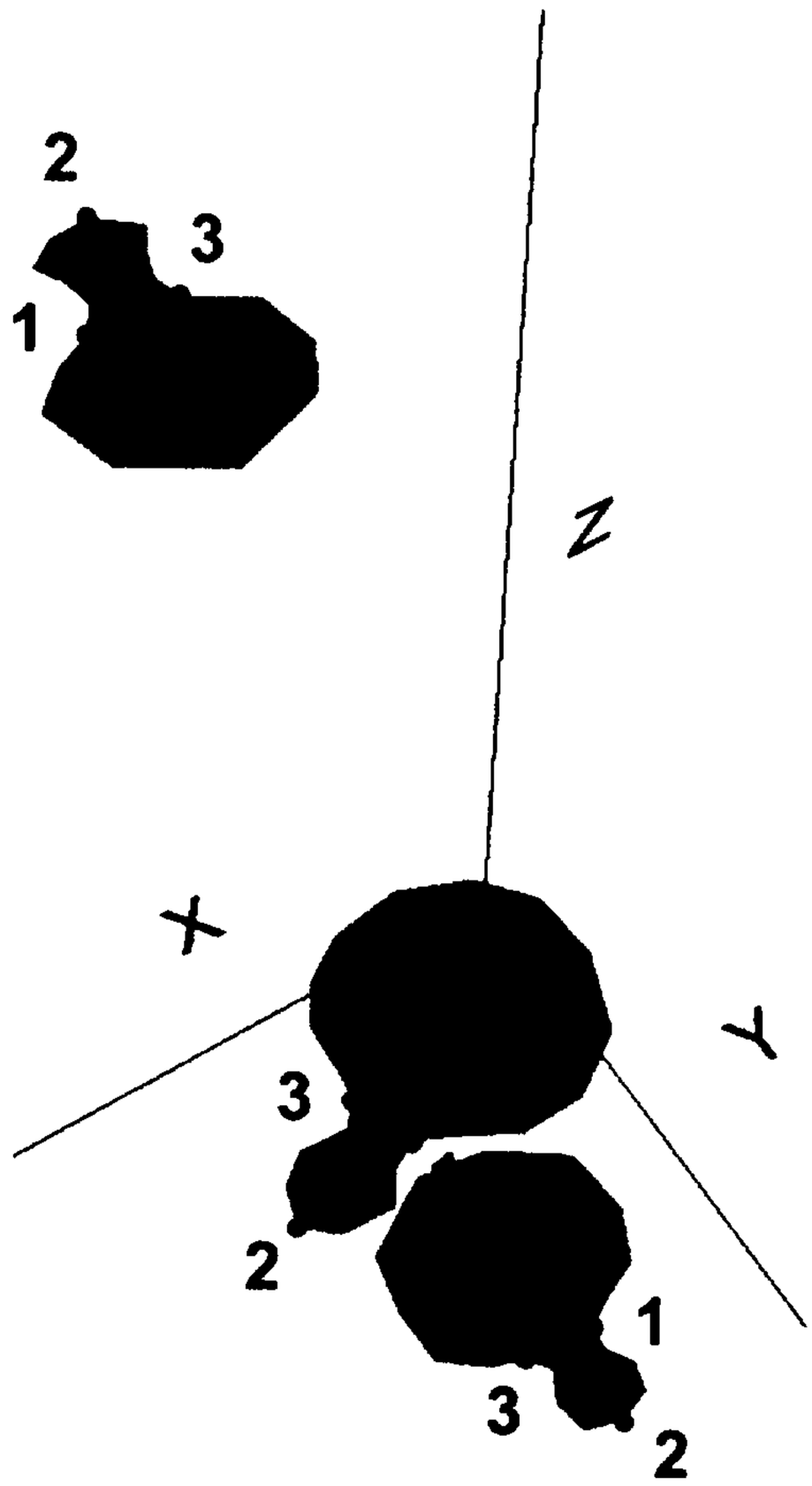

Fig. 11 Definition of the control points on each cross-section to guide the connection of them.

방향의 경로를 따라 이동하는 경우를 가정하였다. 스위핑의 경우 경로를 따라 이동하는 단면의 방향을 정해주어야 하는 문제가 있다. 본 연구에서는 지질학적 개체에 대한 모델링을 상정하고 있으므로 단면이 연결되는 방향을 정하는 것은 지질학적 해석의 한 단계라 할 수 있다. 따라서 이는 사용자와의 상호 작용이 필요한 부분이므로 다음과 같이 입력을 받도록 하였다. 즉, 입력된 각 수

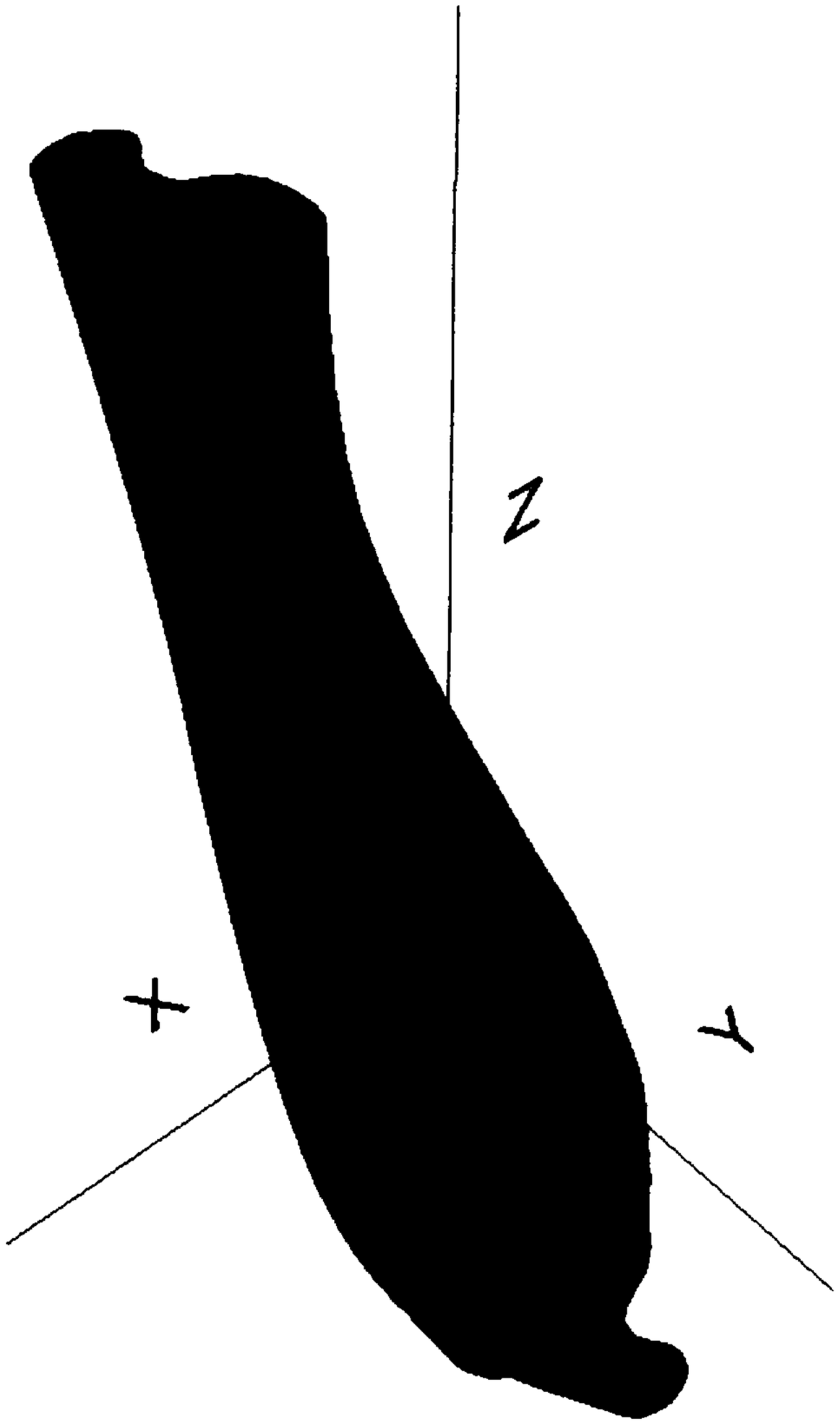

Fig. 12 A polygonal model constructed by sweeping cross-sections.

평 단면을 구성하고 있는 점 (node)들 중에서 인근 단면과 서로 대응하여 연결 가능한 특정점 (control point)들을 정의하고 이 점들에 번호를 부여하여 단면들 간에 서로 어떻게 연결될 것인지를 지정한다 (그림 11).

스위핑에서는 각 단면을 이루는 점들이 서로 대응되어야 하므로 위에 정의한 제어점들 사이를 리샘플링 (resampling)하여 일정 개수의 점을 갖도록 하였다. 이제 단면상의 한 점이 각 단면에서 대응하는 점의 위치를 지나도록 움직이는 궤적을 구해야 하는데 이는 보간의 문제라 할 수 있다.

본 연구에서와 같이 이산적으로 주어지는 3차원 개체의 외곽선을 산출하기 위한 근사 함수로서는 구간별 다항식 (piecewise polynomial) 보간법으로 3차 스

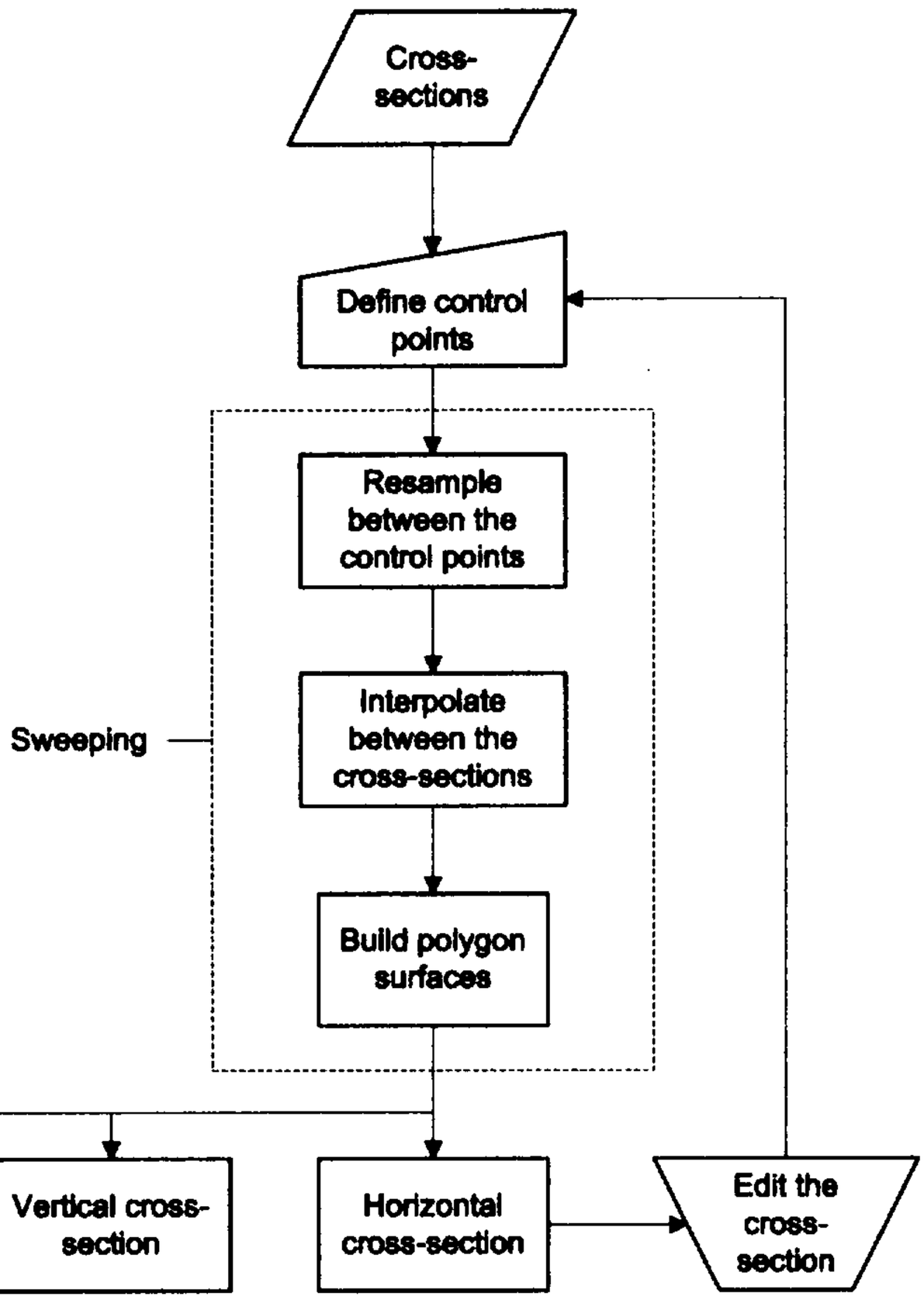

Fig. 13 The flowchart of the modeling system using sweeping.

Table 2 A pseudo-code for sweeping variable cross-sections along a wobbling spine.

Inputs: cross-sections $\mathbf{X} = [x_{nl}]$, control points $\mathbf{P} = [p_{nm}]$, number of resampled points between the control points h, number of resampled point between the cross-sections v. (n: number of cross-sections, m: number of control points on a cross-section, l: number of nodes in each cross-section)

For each $i = 1, \cdots, n$
  For each $j = 1, \cdots, \mathrm{m}$
    For each $k = 1, \cdots, l$
      $\mathbf{K} = [k_{nm}] = \{k \mid \min(\|p_{ij} - x_{ik}\|)\}$ (find indices of **x** that has minimum distance to **p**)
    End
  End
End
For each $i = 1, \cdots, n$
  For each $j = 1, \cdots, m$
    $\mathbf{s} = [x_p]$ $(p = k_{ij}, \cdots, k_{i,j+1})$
  End
  $\mathbf{S} = [s_{mp}]$ (list of nodes arranged by control points on a cross-section)
  For each $j = 1, \cdots, m$
    $\mathbf{r} = [r_h]$ (resampled points obtained by interpolating **s**)
  End
  $\mathbf{R} = [r_{mh}]$ (list of resampled points between the control points)
End
$\mathbf{H} = [r_{nmh}]$ (points for the whole cross-section)
For each $j = 1, \cdots, m$
  For each $i = 1, \cdots, h$
    $\mathbf{v} = [r_n] = \{r \mid r_{nji} \in \mathbf{H}\}$ (resampled points that have constant i and j)
    $\mathbf{t} = [t_v]$ (resampled points obtained by interpolating **v**)
  End
  $\mathbf{T} = [t_{hv}]$ (list of resampled points between the cross-sections)
End
$\mathbf{V} = [t_{mhv}]$ (points for the whole model)

플라인 (cubic spline) (Press *et al.*, 1992)을 사용하여 모델링을 수행하였다. 그림 12는 보간 후 외곽선들로부터 폴리곤을 생성하여 모델을 완성한 모습이다. 폴리곤의 크기는 보간 시 샘플링 개수를 조절함으로써 제어할 수 있다.

그림 13에 본 프로그램의 흐름도를 도시하였고, 이 중 핵심적인 부분이라 할 수 있는 스위핑에 대한 의사 코드를 표 2에 제시하였다. 이 코드에서 중요한 부분은 사용자로부터 주어진 제어점과 기존의 폴리곤 정점을 짝짓는 것과 두 번의 보간을 거치면서 서로 연결되어야 할 점들을 정확하게 유지하는 것이다. 이 것은 한 레벨에 단면이 하나만 있는 경우에는 비교적 단순하나 단면이

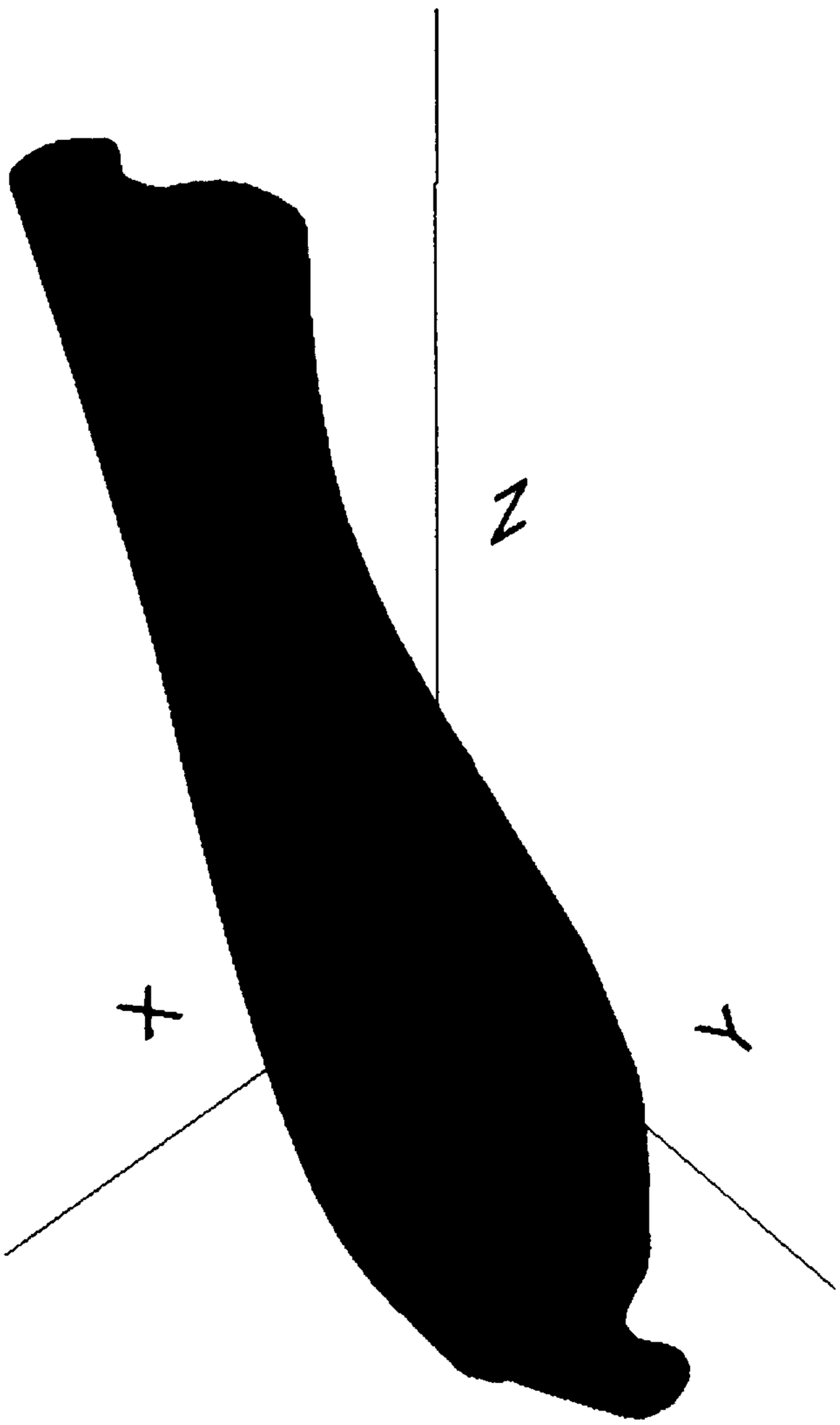

Fig. 14 A horizontal cross-section generated by the software at the user-defined altitude of the model.

여러 개이고 레벨 마다 단면의 개수가 달라지는 경우에는 주의를 필요로 한다. 부록 1에 모델링 과정에 대한 Avenue 코드를 수록하였다.

완성된 3차원 모델로부터 임의의 수직·수평 또는 경사 방향으로 단면을 생성할 수 있다. 수평 단면은 주어진 고도를 포함하는 폴리곤들을 추출하여 선형

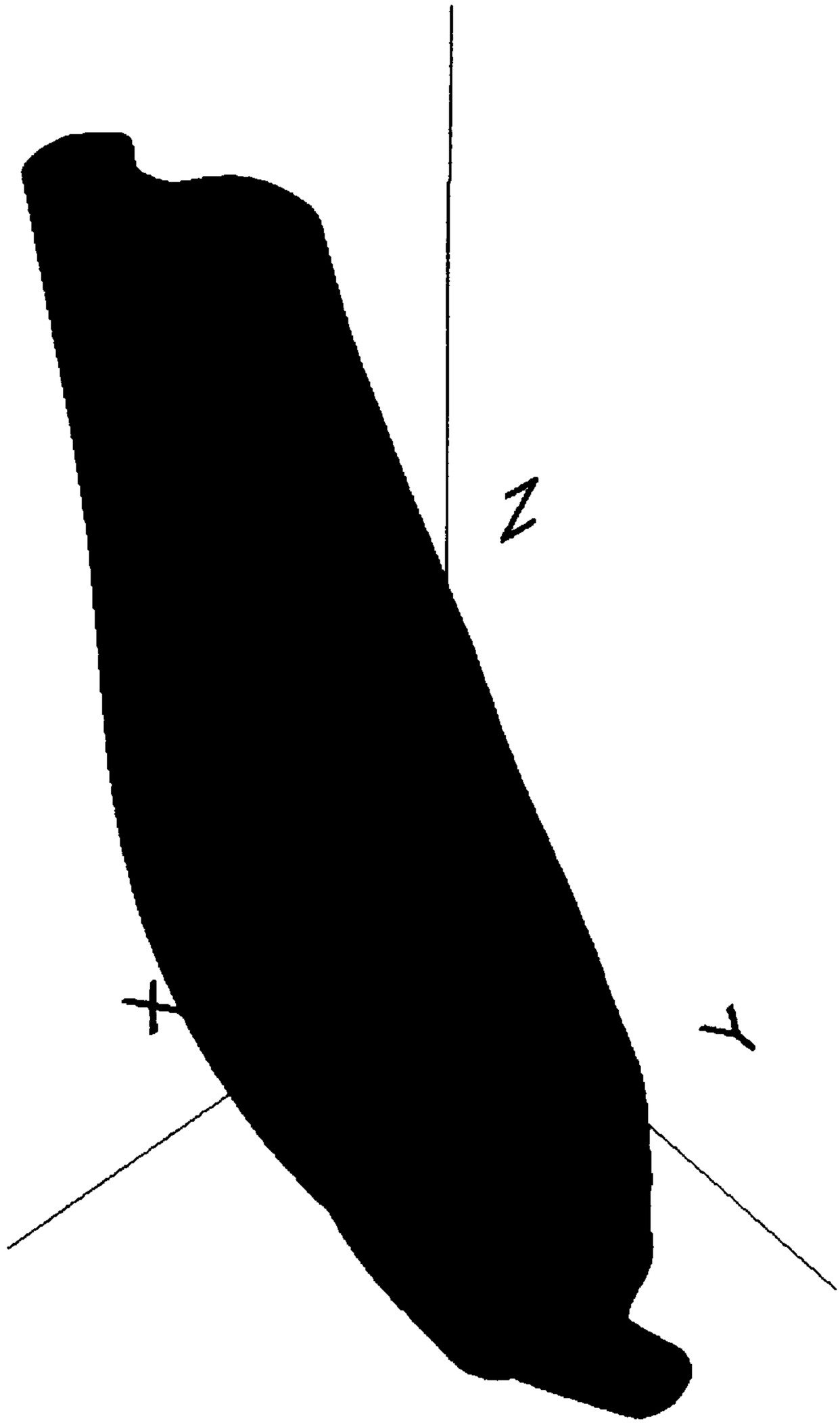

Fig. 15 An updated model with the edited version of the cross-section in Figure 14.

보간에 의해 좌표를 계산한다 (그림 14). 또한 수평 방향의 단면은 사용자의 편집에 의한 수정이 가능하며, 수정된 단면을 포함하여 다시 모델링을 수행함으로써 모델의 형태를 계속적으로 갱신할 수 있다 (그림 15).

수직 단면은 사용자가 X-Y 평면에서 모델을 지나가도록 정의한 선분으로부

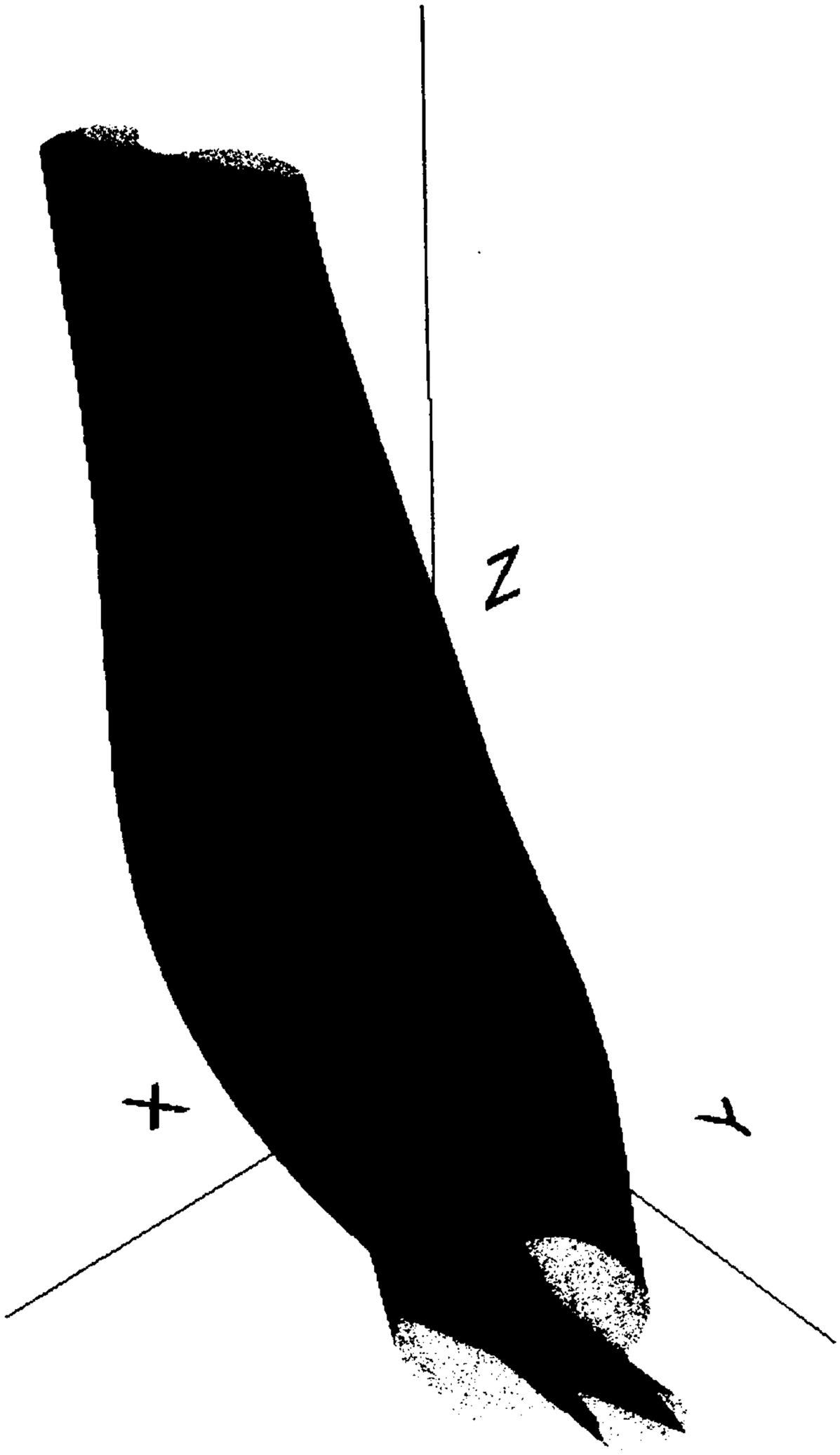

Fig. 16 A vertical cross-section generated by the software from the user-defined line segment.

터 모델 면을 이루는 각 폴리곤의 상하부 경계선이 선분과 만나는 점들을 연결하여 생성한다 (그림 16). 경사 단면도 X-Y 평면에서 모델의 최상부에서의 선분과 최하부에서의 선분을 정의해 주면 그 사이에서 수직 단면과 같은 방법으로 단면을 생성한다 (그림 17). 이 과정 중 선분의 정의와 수평 단면의 편집 부분은 ArcView의 기본 인터페이스를 사용하였다.

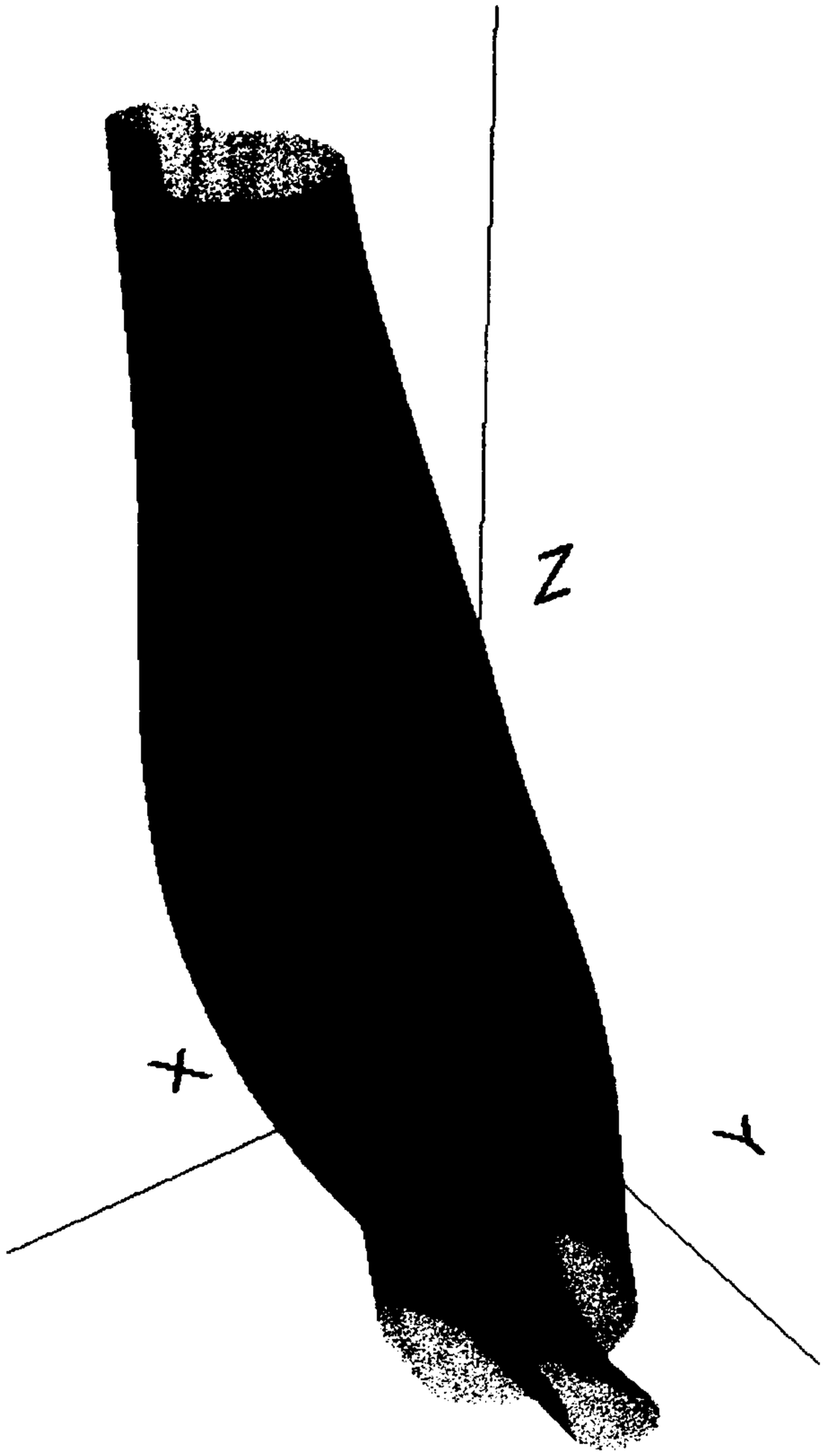

Fig. 17 An inclined cross-section generated by using user-defined quadrilateral.

2.2 광체 모델링

완성된 모델링 프로그램을 그림 18의 실제 자료에 적용하였다. 이 자료는 김현규와 이두성 (2001)의 연구에서 제작한 광체 단면 자료이다. 전체 자료 중 모델링에 사용된 단면은 광체의 하부에 해당하는 -500 ML (30.3 m)부터 -275 ML (255.3 m)까지 총 6개이다. 이들을 택한 이유는 단면의 형태에 비교적 일관성이 있어 추가적인 작업을 하지 않고도 모델링이 가능하기 때문이다. 즉, 그림 18의 상부에 있는 단면들에서 볼 수 있듯이 상하 단면의 개수나 형태에 큰 차이가 있거나 지나치게 복잡한 경우에는 단지 단면 사이의 기하학적 유사성에만 의존하여 모델링을 하기 어렵고 지질학적 분석을 통하여 단면 사이의 연관성을 유

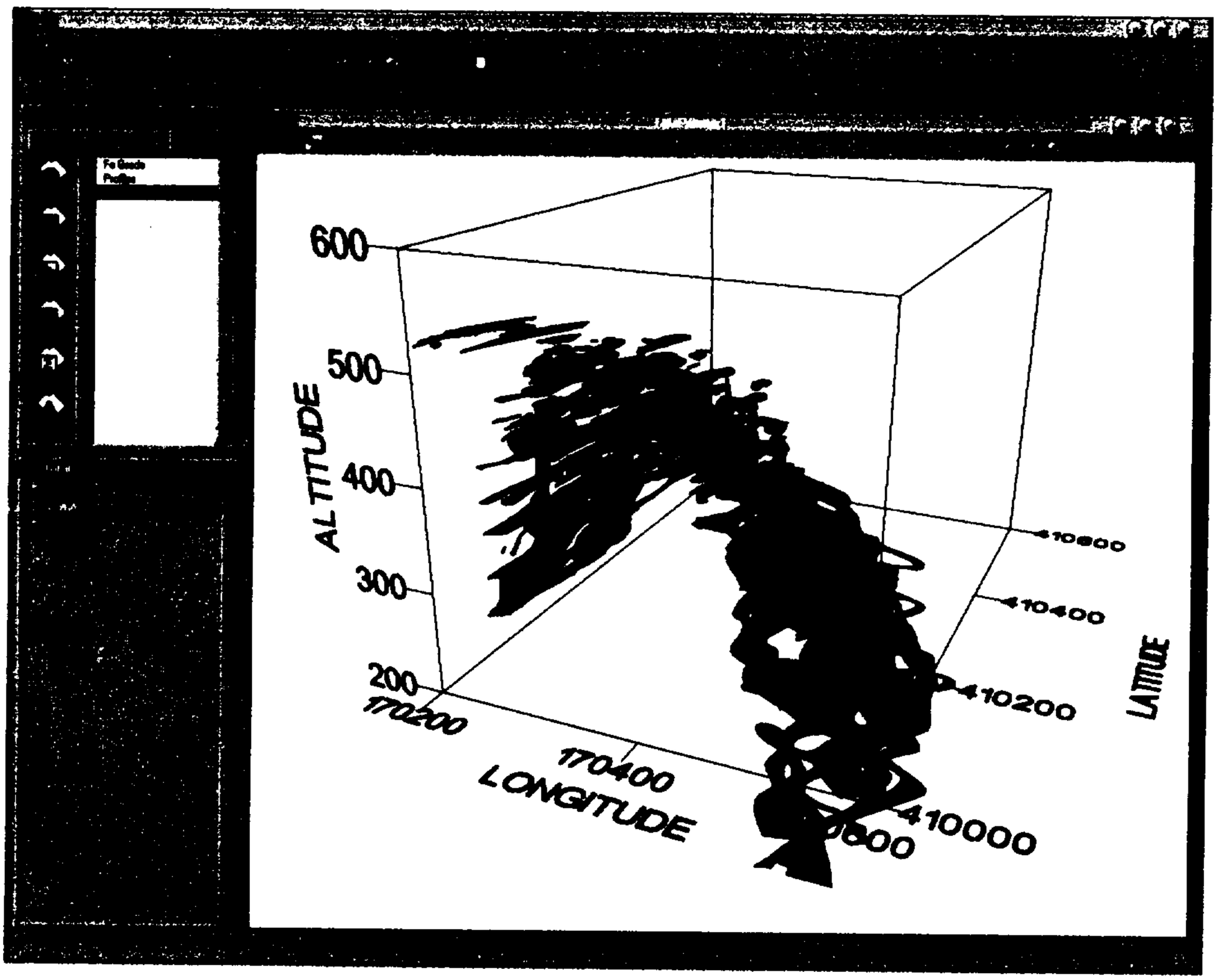

Fig. 18 A 3-D perspective of the ore bodies (dark gray shapes) and the cavities (light gray shapes).

추한 뒤 모델링 프로그램에서 처리할 수 있는 형태로 편집하는 과정이 필요하다. 이러한 과정을 통하여 전체 광체에 대한 완전한 모델을 얻는 것은 이 지역에 대한 전문가의 도움과 함께 상당한 기간을 요할 것이므로, 본 연구에서는 소프트웨어에 대한 지식만으로 처리할 수 있는 구간을 선택하여 모델을 구성하고자 하였다.

모델링 순서는 하부로부터 상부로 향하는 방향으로 수행하였다. 물론 그 반대 방향으로 하여도 무방하며, 모델링 과정을 단계별로 기술하면 다음과 같다.

1) 최하부의 3개 단면에 대하여 제어점을 설정하고 모델링 실시

2) 우선 생성된 모델에서 면이 교차하는 것과 같은 기하학적 오류가 있는지

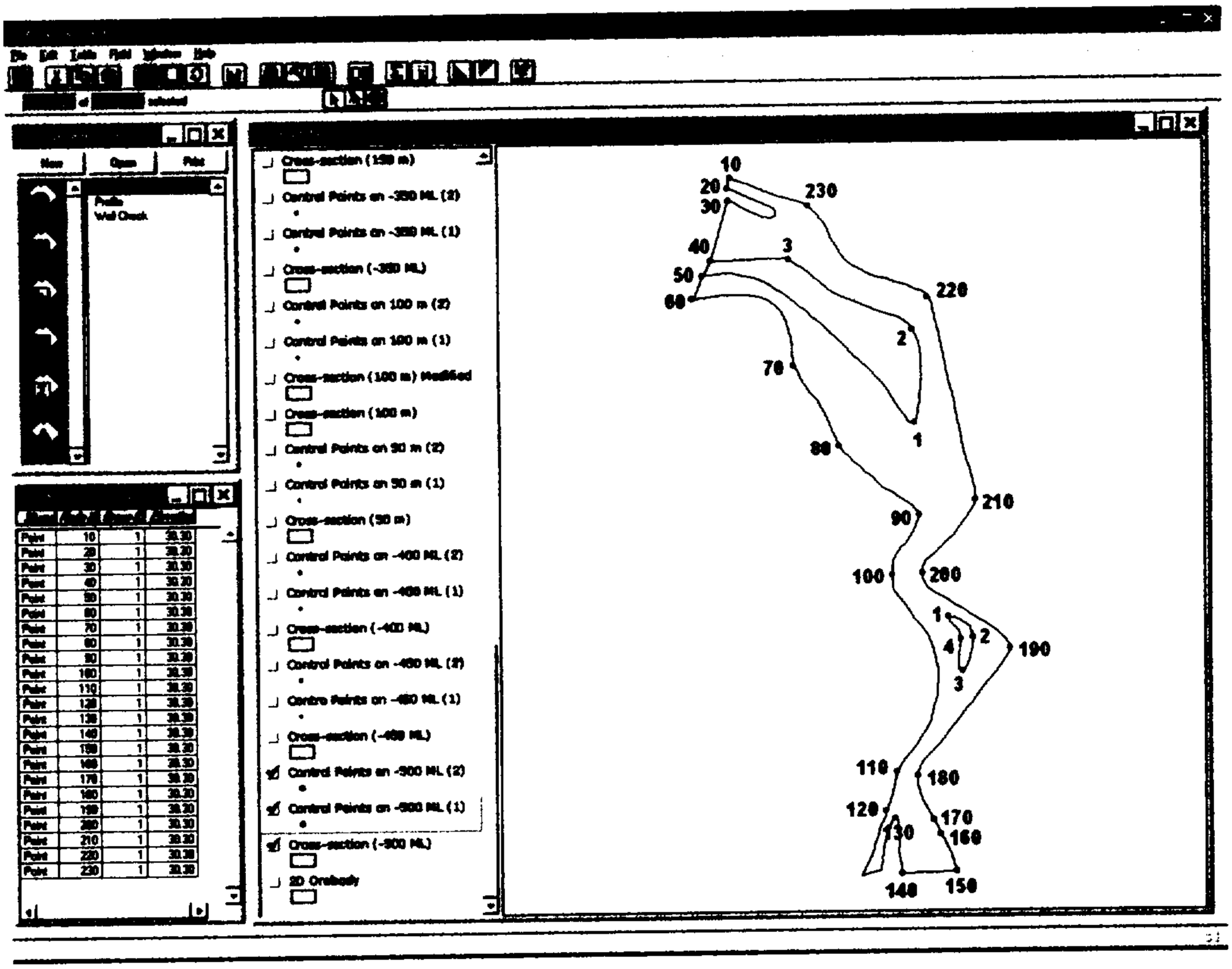

Fig. 19 Allocation of the control points, which are identified by group ID, node ID and elevation, on a cross-section.

검사

3) 오류가 있으면 그 부분의 수평 단면을 추출하여 오류를 수정하고 상하 단면의 제어점들을 고려하여 제어점을 설정

4) 새로 생성된 단면을 기존의 단면과 함께 다시 모델링

5) 수정된 모델에 오류가 제거될 때까지 3 ~ 4 단계를 수행

6) 오류가 없으면 상부의 단면을 하나 추가하여 모델링 실시

그림 19는 하나의 단면에 대하여 제어점을 설정하는 장면이다. 각 단면은 폴리곤 shapefile (ESRI, 1998)로 되어있으며 제어점은 점 (point) shapefile로 생성한다. 하나의 단면은 여러 개의 폴리곤으로 구성될 수 있으며 이는 폴리곤 내부에 공동 (空洞)이 있는 경우로서, 실제로는 광체 내부로 다른 암석이 통과하는 지질 구조일 것이다. 따라서 각각의 폴리곤을 구분하기 위하여 해당 제어점들

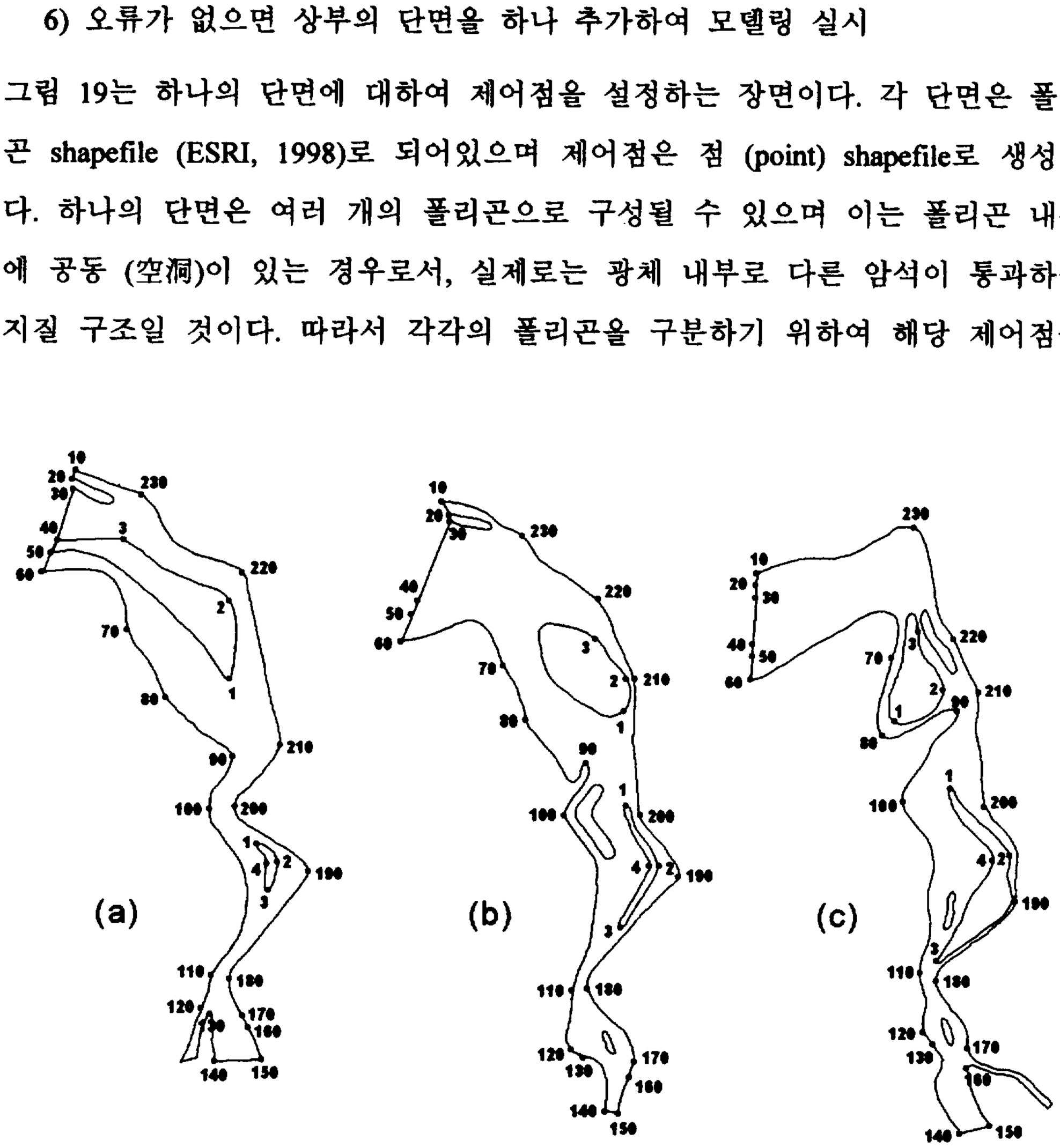

Fig. 20 Allocation of the control points on the cross-sections at the bottom of the orebody.

에 폴리곤 별로 그룹 번호를 할당하며 각 그룹 내에 있는 제어점들에 일정한 노드 (node) 번호를 부여한다. 그룹 번호와 노드 번호는 shapefile의 속성 테이블 (attribute table)을 편집함으로써 설정한다. 그림 19에서는 단면을 3개의 그룹으로 나누어 단면상의 특이점을 중심으로 제어점을 설정하였다. 그룹 1은 최외곽

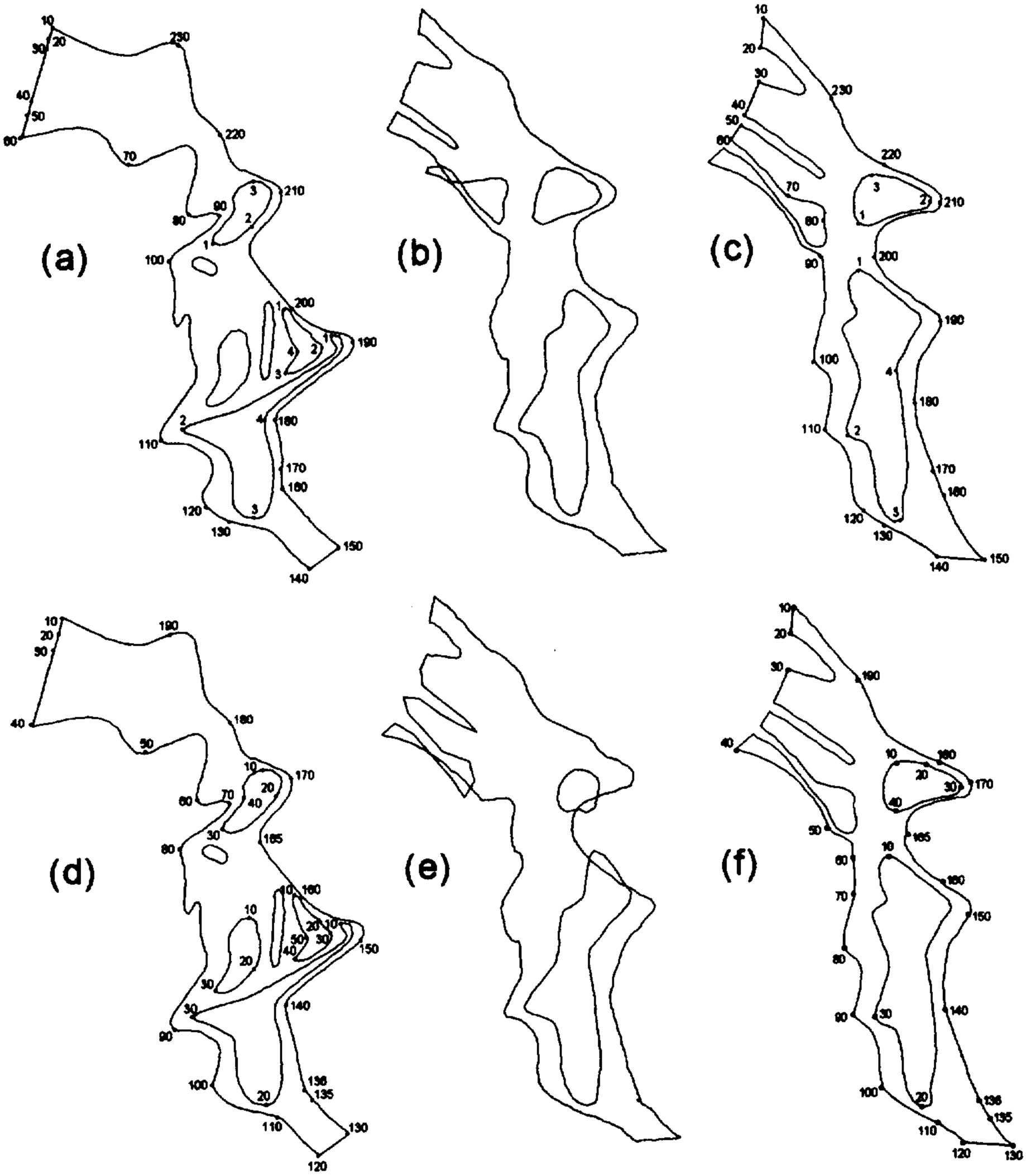

Fig. 21 The cross-sections extracted from two models generated with different configurations of the control points.

선, 그룹 2는 단면 상부의 공동 폴리곤, 그룹 3은 단면 하부의 작은 공동 폴리곤에 각각 대응한다. 제어점의 위치는 현재 단면 및 상·하부의 단면을 비교하여 광체의 형태가 서로 대비되는 곳으로 선정하며, 단면 연결 시 같은 그룹과 노드 번호를 가진 점들끼리 이어지게 된다. 그러므로 제어점의 설정은 모델의 형태를 결정짓는 중요한 요소이며 지질학적 해석이 요청되는 과정이다. 최하부의 3 단면에 대한 제어점은 그림 20과 같이 설정하였다.

그림 21은 두 개의 원시 단면 (심도 180.3 m와 230.3 m) 사이에서 모델링 후 생성된 단면 (215 m)으로서 제어점 설정의 중요성과 그 영향을 보여준다. 그림 상단의 (a) ~ (c)는 본 연구에서 적용한 제어점이며 하단의 (d) ~ (f)는 연구 초기에 시험적으로 적용해본 제어점 구성이다. (b)와 (e) 모두 다른 그룹 혹은 같은 그룹의 다각형과 겹치는 부분이 발생하나 (b)가 훨씬 작은 면적으로 중첩되므로 더 좋은 모델임을 알 수 있다. 이와 같은 표면의 중첩은 원시 단면의 모양이 복잡하고 단면 사이의 수직 거리에 비해 횡적 변화가 클 때 나타날 수 있다. 이런 현상은 실제적으로는 존재하지 않으므로 중첩된 단면을 수정하여 가능한 모델로 개선하는 과정이 필요하다. 그림 22는 그림 21b를 수정하고 새로이 제어점을 설정한 모습이다.

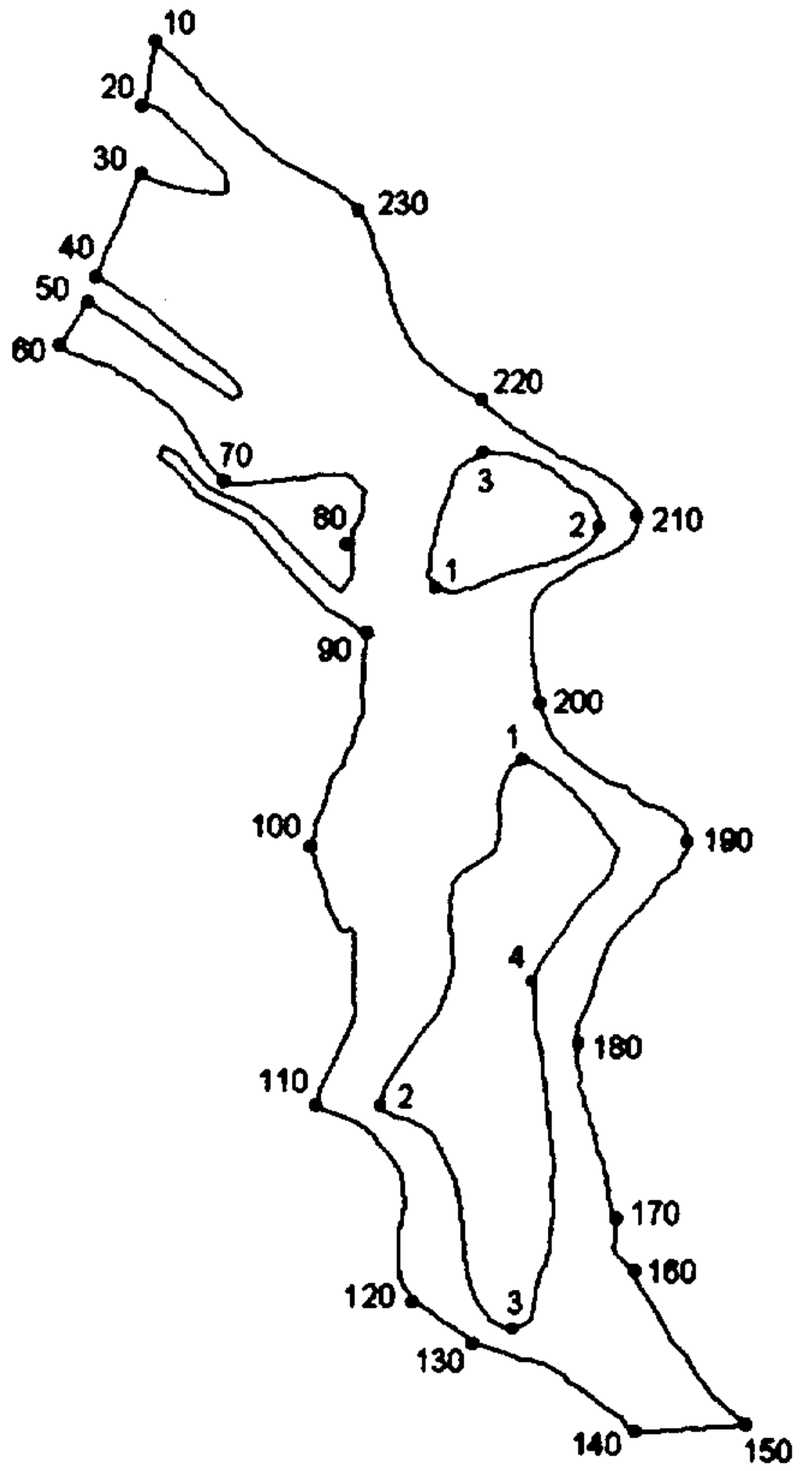

Fig. 22 Modified cross-section from Figure 21b with control points allocated on it.

생성된 모델의 신뢰성은 입력 자료인 원시 단면의 신뢰도와 제어점 설정에 달려 있다. 본 연구에서 사용된 입력 자료는 굴진 갱도와 수평 시추 결과로

부터 작성된 것이다. 이 지역에는 총 63개의 수직 및 경사 시추 조사 자료가 있는데 재래식 방법으로 모든 시추 자료를 체계적으로 사용하여 수평 단면을 작성하는 것은 상당한 시간이 걸리는 작업이다. 본 연구에서는 이를 위하여 시추 자료로부터 특정 고도에서의 정보를 추출하여 시각화하는 기능을 추가하였다. 그림 23에 180.3 m (-350 ML)의 광체 수평 단면도에 수직·경사 시추 조사 자료에 의한 암종을 표시하였다. 그림에서 보는 바와 같이 이들 시추 자료를 사용하면 원래의 수평 단면도는 다소 수정이 필요하다는 것을 알 수 있다.

컴퓨터 모델링으로 제작한 3차원 모델에 의한 단면과 사람의 수작업에 의한 해석 단면을 비교해보기 위하여 김현규와 이두성 (2001)의 연구에서 생성하였던 수직 단면과 같은 위치에서 단면을 추출하였다. 그림 24a는 대한광업진흥공사의 해석에 의한 단면이며 그림 24b는 본 연구에서 산출한 단면이다. 광체의 모양이 비교적 단순한 하부에서는 두 단면이 비슷한 양상을 띄고 있으나, 형태가 복잡하여지는 상부에서는 서로 차이를 보이고 있다. 어느 것이 옳다고 단정짓기는 어려우나, 2차원적인 정보에 의존하여 2차원적으로 해석하였을 가능성이 큰 그림 24a보다는 3차원적인 개념에 의한 그림 24b가 기하적으로는 우수할 것이라 예상할 수 있다. 그러나 지질학적 개체에 대한 해석은 단순히 수리적 (數理的)으로만 따질 수 없고 주변 지

Fig. 23 The lithology of the cross-section obtained from the vertical and inclined well cores.

질을 포함하여 종합적인 지식이 요청되므로 본 지역의 지질에 대한 전문가가 모델링을 수행하는 것이 이상적이라 하겠다.

최종적으로 완성된 3차원 광체 모델은 그림 25에 도시하였다.

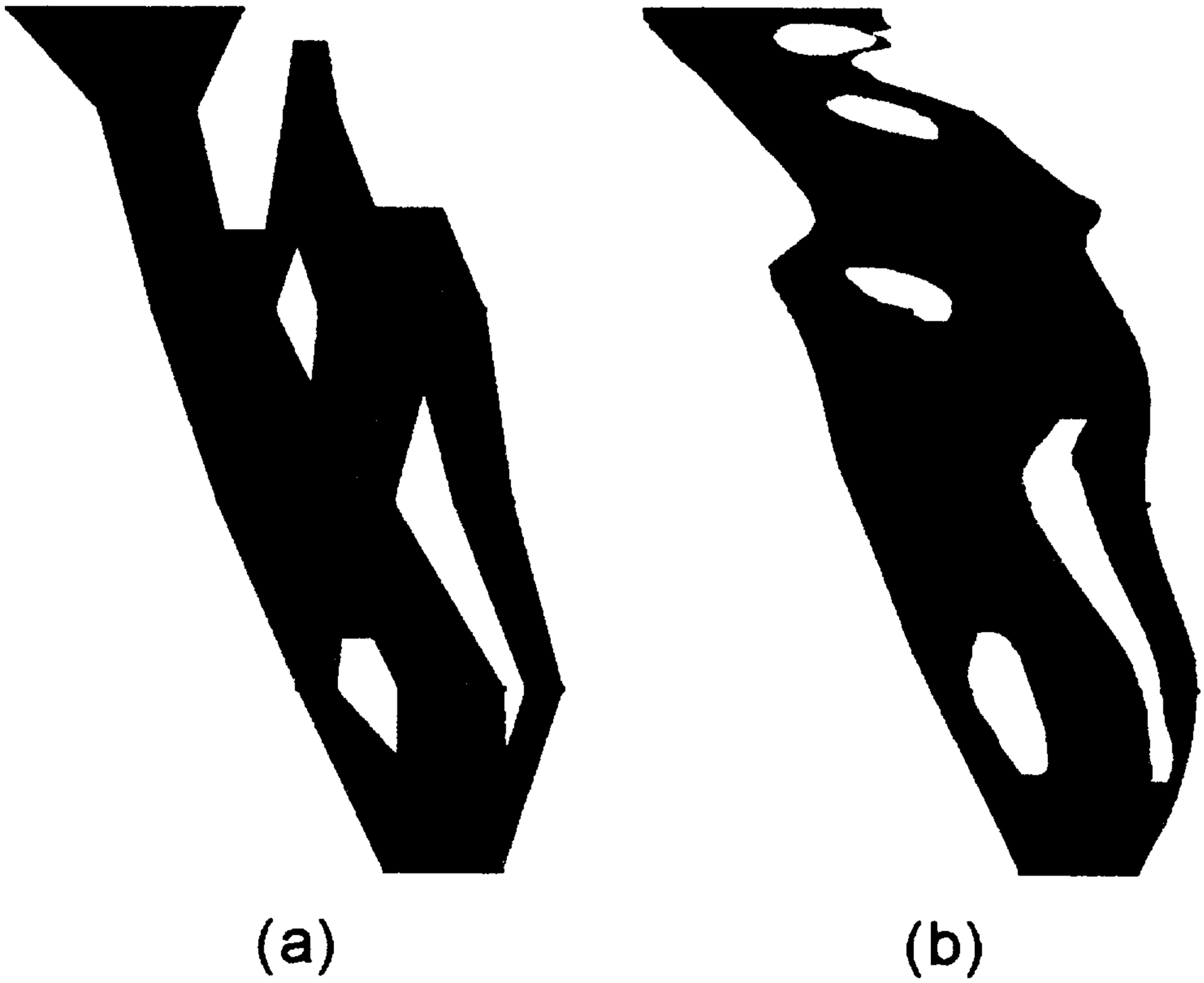

Fig. 24 Comparison of the vertical cross-sections generated by interpretation of human expert (a) and by this computer software (b).

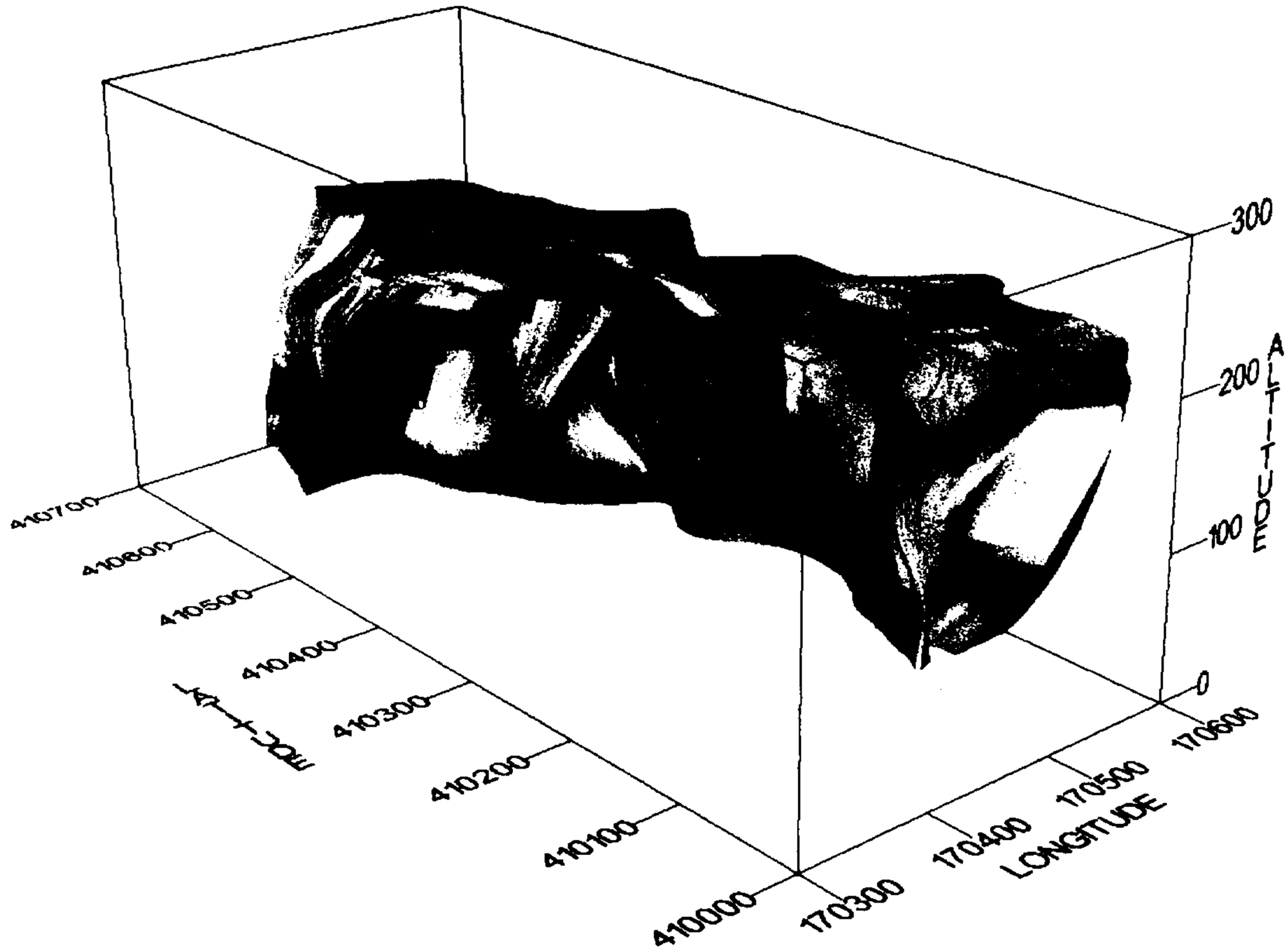

Fig. 25 The final model of the orebody constructed from the horizontal cross-sections.

2.3 함몰대 모델링

3차원 탄성파 자료의 해석 결과로 도출되는 3차원 개체에 대한 해석 또는 이를 기반으로 하는 탐사 계획 수립 등을 위해서는 개체를 3차원적으로 시각화할 필요가 있다. 이러한 시각화는 다음의 2 단계를 통하여 구현할 수 있다. 1) 수평 단면 상에서 개체의 윤곽을 피킹하여 폴리곤으로 정의하고, 2) 모델링 소프트웨어를 사용하여 위에서 정의된 수평 단면을 3차원적으로 시각화한다.

그림 26에 도시한 Boonsville 자료를 보면 사각형으로 표시된 부분의 지층

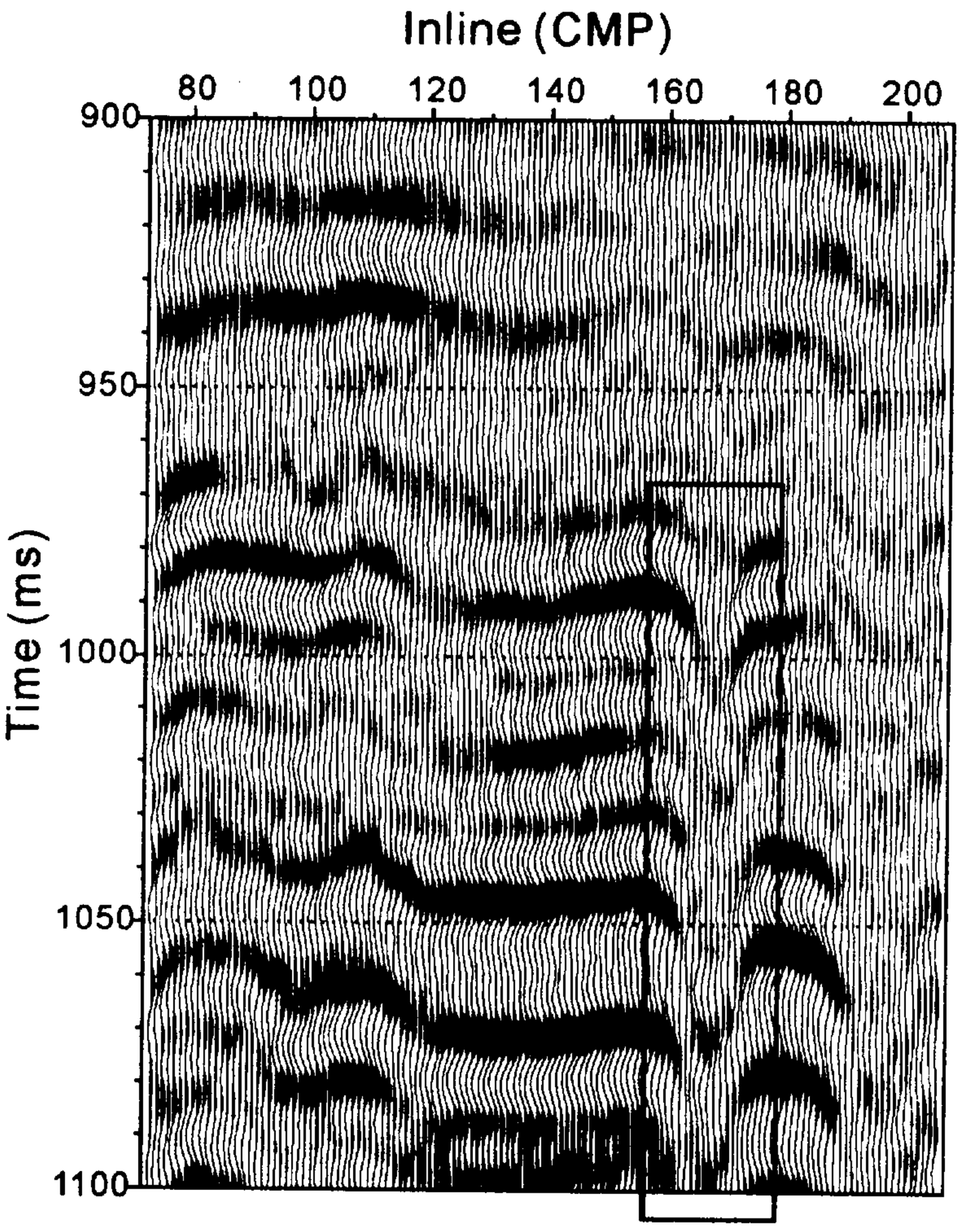

Fig. 26 A vertical cross-section at cross-line 170 of the 3-D seismic data showing a formation distortion.

함몰 형태를 인지할 수 있다. 이 특이 패턴은 석회암 공동 상부 지층의 함몰이 싱크홀 (sinkhole) 형태로 표현된 것이다. 이 구조는 인라인 (inline) 좌표 약 170, 크로스라인 (crossline) 좌표 약 145, 수직 (시간) 좌표 약 980 ms ~ 1100 ms에 나타난다. 980 ms ~ 1100 ms 구간 중 11개의 수평 단면에서 싱크홀 구조의 외곽부를 디지타이징한 뒤 모델링을 거쳐 그 결과를 시각화하였다 (그림 27). 그림 27에서 표시된 3개의 수평 단면은 각각 위로부터 1000, 1050, 1100 ms의 시간 단면 (time slice)이다.

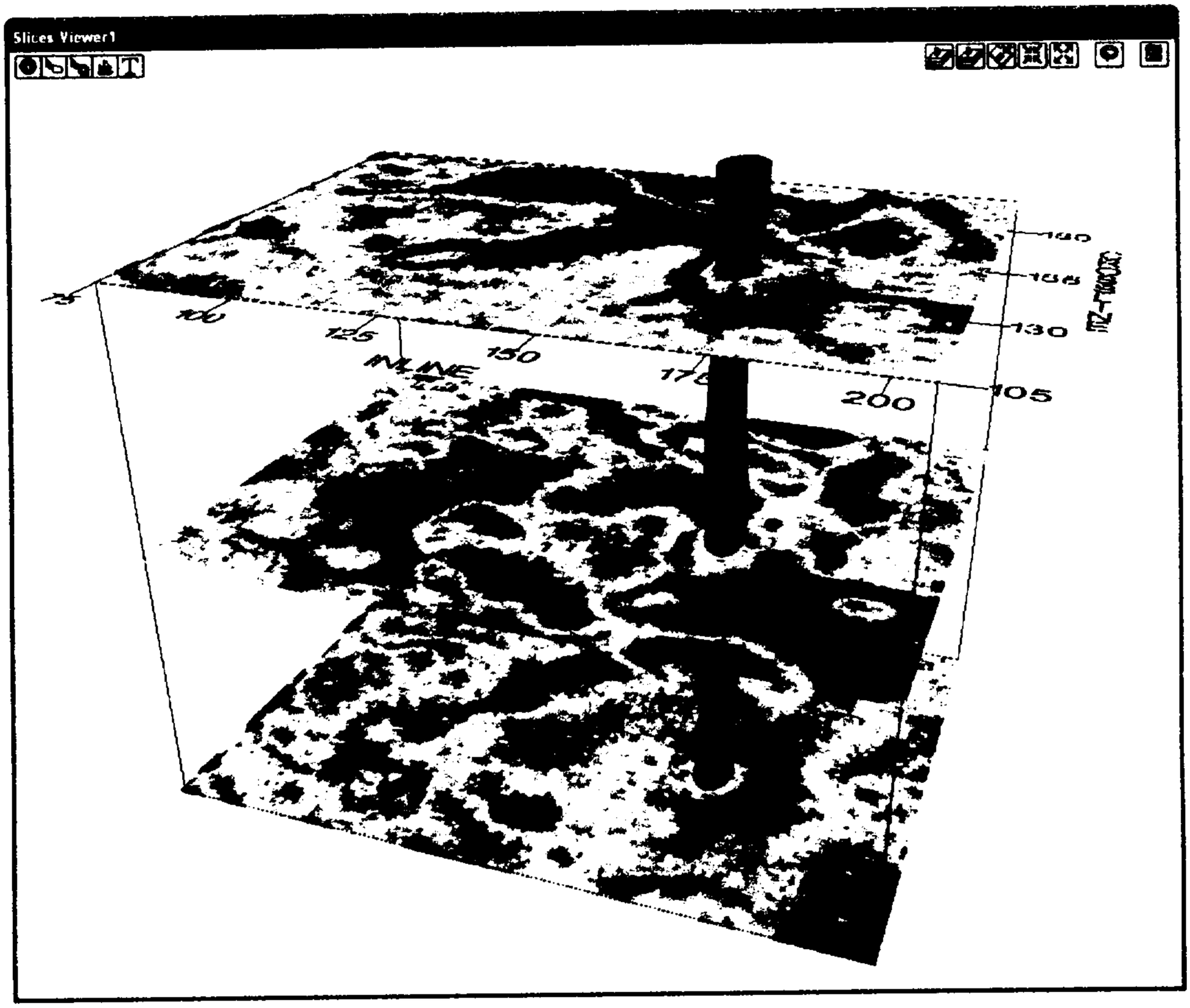

Fig. 27 Three-dimensional view of the sinkhole model.

V. 결론

본 연구는 다양한 지하 정보를 쉽고 빠르게 분석할 수 있는 정보 시스템을 구축하기 위한 기초 연구이다. 본 연구에서 개발한 지하 정보 시스템은 획득된 자료를 데이타베이스화 함으로써 체계적인 분석과 관리를 가능하게 하며 3차원 시각화를 통하여 신속하고 정확한 해석을 수행할 수 있게 한다.

본 연구에서 시각화의 대상으로 설정한 지하 정보의 자료 유형은 3차원 공간 상에서 각 축 방향으로 균질한 분포를 가진 3차원 격자 배열 형태의 자료와, 3차원 좌표를 가지면서 선형으로 획득된 1차원 배열 자료이다. 첫 번째 자료 유형의 예로서 3차원 탄성파 자료를 시각화하였고 두 번째 유형의 예로는 시추 검층 자료와 주상도 자료를 사용하였다.

균질한 3차원 자료의 시각화로는 먼저 전체 자료를 일정 방향 단면들의 집합으로 변환하고 그 가운데 특정 위치의 단면을 사용자의 요청에 따라 표시하는 방법을 사용하였다. 3차원 탄성파 자료를 단면으로 변환하기 위하여 각 원소로부터 3개의 축 방향에 수직인 사각형 폴리곤을 만들고 여기에 탄성파 정보를 속성 값으로 입력하였다. 결과적으로 각 축에 대한 세 개의 폴리곤 shapefile이 생성되며 슬라이더 방식의 대화 상자를 이용하여 특정 위치에서의 단면을 표시하였다.

균질 3차원 배열에 대한 두 번째 시각화 방법은 자료를 육면체 형태로 표시하는 것으로서, 각 원소를 작은 육면체 형태의 화소 (voxel)로 변환하고 이 가운데 주어진 범위에 속하는 복셀만을 도시한다. 이 때 범위 안의 복셀을 모두 화면에 표시하는 것은 그래픽 처리 시 매우 비능률적이므로 최외곽의 복셀만을 표시하게 된다.

선형 배열 자료의 경우에서 시추 검층 자료와 시추 주상도 자료는 약간 다른 특성을 가지는데, 전자의 경우 각 샘플 사이의 공간적 간격이 일정하며 점

에 대한 정보를 가지는 반면 후자는 샘플 간격이 불규칙하고 특정 구간, 즉 선에 대한 정보를 가지고 있다. 따라서 두 가지 경우에 대하여 각각 다른 방식의 시각화를 수행하였다.

검층 자료에 대해서는 수직공의 경우만을 고려하였는데, 시추공의 형태에 가깝게 시각화하기 위하여 각 점마다 샘플 간격만큼의 두께를 가지는 수평 고리 모양의 폴리곤을 생성하고 여기에 검층 값을 속성 (attribute)으로 연결하였다. 샘플 간격이 일정하므로 고리의 배열은 서로 연결되어 원주 형태의 폴리곤 모델을 형성하며 검층 값의 변화는 색도의 차이로서 인지하게 된다.

주상도 자료는 경사 및 수직공 모두를 다루었으며 샘플 간격이 일정하지 않으므로 약간 복잡한 과정을 거쳐 시각화하였다. 먼저 각 시추공의 시작점 좌표와 방위 및 경사를 획득하여 주상도 자료와 동시에 입력받는다. 주상도의 각 구간에 대한 정보에는 지질학적 특성 외에 공간 속성, 즉 시추공 시작점으로부터의 거리와 구간의 길이가 포함되어 있으며, 이들을 이용하여 해당 구간의 시작점과 끝점의 좌표를 계산한다. 마지막으로 두 점의 좌표로부터 선형 shapefile을 생성하고 주상도 정보를 속성으로 연결한다.

시각화된 자료로부터 의미 있는 정보를 추출하고 이로부터 새로운 모델을 구성할 수 있는 도구를 개발하였다. 정보의 추출과 모델링에는 자료와 사용자 사이에 능동적인 상호 작용이 필요하며 이를 위해 다양한 대화형 인터페이스가 요구된다. 정보의 추출 또는 재구성은 기존 정보에 대한 해석의 결과이며 이는 시각화된 정보를 면밀하게 탐색하는 것으로부터 시작한다.

정보 추출 도구는 3차원 격자 배열 자료를 대상으로 동작하며 단면 상에서 추출된 정보를 저장한다. 사용자는 제공된 탐색 도구를 이용하여 자료를 관찰한 뒤 해석을 거쳐 특정한 이벤트를 선정하고 그 정보를 추출할 수 있다. 탄성파 자료에서 이러한 이벤트는 일반적으로 선형으로 나타나므로 본 연구에서는 곡선 상의 정보를 추출할 수 있도록 프로그램을 설계하였다. 즉, 사용자가 단면

상에서 타점 장치 (pointing device)를 사용하여 대상 이벤트를 따라 곡선을 정의하면 추출기는 곡선 주변의 일정 범위 안에서 최소 또는 최대치를 검색하여 해당 정보를 추출한다.

단면 상에서 추출된 정보가 폐곡선 형태를 가지는 경우 유한 개수의 단면을 입력받아 근원 개체를 3차원으로 모델링 할 수 있도록 하였다. 즉, 2차원적으로 획득된 단면들을 연결하여 3차원 공간에서의 원형을 추정하는 도구를 개발하였다. 단면 사이의 연결은 스위핑이라는 모델링 기법을 사용하여 사용자가 정의한 제어점을 따라 부드러운 곡면을 생성하도록 하였다. 또한, 정의된 3차원 모델에 대하여 임의의 수평·수직 및 경사 단면으로 도시가 가능하게 하였고, 해석자가 이 단면을 수정할 수 있는 기능을 부여함으로써 반복적으로 모델의 형태를 개량할 수 있도록 하였다.

모델링 도구를 이용하여 실제 광산에서 획득한 광체 단면도를 토대로 광체의 3차원 모델을 제작하였다. 모델링 과정에서 결과의 품질을 결정하는 가장 중요한 요소는 제어점의 설정이며, 이는 사용자의 경험과 지식에 의해 좌우된다. 실제 자료의 모델링 결과는 만족할 만한 수준인 것으로 판단되며, 이로써 본 시스템의 실용성에 대한 검증으로 삼을만하다.

본 연구에서 개발한 정보 시스템은 지하 정보를 데이타베이스화 함으로써 자료의 저장 및 수정·보관 등의 관리를 용이하게 하며, 기존의 수작업으로는 불가능하였던 방대한 양의 정보에 대한 실시간 검색과 분석을 가능하게 한다. 또한 3차원으로 시각화된 정보는 종래 2차원으로 주어졌던 평면적 정보에 비해 사용자의 공간적 이해력을 향상시키며 지하 개체의 분석 및 해석에 적지 않은 도움을 줄 것이다. 따라서 본 시스템의 이점을 잘 활용하면 지상 및 지하 구조물 설계, 자원 개발 계획의 수립 등 여러 분야에서 적은 비용으로 보다 정확한 판단을 도출하는 데 기여할 수 있을 것으로 기대된다.

참고 문헌

국립 지리원, 1998, 수치 지도 좌표계에 관한 연구: 국립 지리원.

김현규, 이두성, 2001, ArcView를 이용한 지하 정보 및 시각화 시스템 구축 사례 연구: 물리 탐사, **4**, 101-109.

ESRI, 1996a, Using ArcView GIS: Environmental Systems Research Institute, Inc.

ESRI, 1996b, Using Avenue: Environmental Systems Research Institute, Inc.

ESRI, 1997a, Using ArcView 3D Analyst: Environmental Systems Research Institute, Inc.

ESRI, 1997b, Using the ArcView Dialog Designer: Environmental Systems Research Institute, Inc.

ESRI, 1998, ESRI Shapefile technical description: Environmental Systems Research Institute, Inc.

Fuller, M. and Francis, B., 1996, The use of visualization in the examination of work and life histories, *in* Earnshaw, R., Vince, J., and Huw J., Eds., Visualization and modeling: Academic Press Ltd., 63-72.

Gwak, S. H. and Lee, D. S., 2000, Visualization of subsurface information: an interpretation tool: Geosystem Engineering, **3**, 85-89.

Hardage, B. A., 1996, Boonsville 3-D data set: The Leading Edge, **15**, No. 7, 835-837.

Kidd, G. D., 1999, Fundamentals of 3-D seismic volume visualization: The Leading Edge, **18**, No. 6, 702-712.

Manssour, I. H., Freitas, C. M. D. S., Claudio, D. M., and Wagner, F. R., 1997, Visualization and exploring meteorological data using a tool-oriented approach, *in* Earnshaw, R., Vince, J., and Huw J., Eds., Visualization and modeling: Academic Press Ltd., 47-62.

Owen, G. S., 1999, HyperVis - Teaching scientific visualization using hypermedia: ACM SIGGRAPH Education Committee.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 1992, Numerical recipes in C, 2nd ed.: Cambridge University Press.

Watt, A., 2000, 3D computer graphics, 3rd ed.: Addison-Wesley.

부록

1. 이산적으로 주어지는 단면 정보로부터 모델을 생성하는 Avenue 스크립트

```
'=================================================
'
' MODELER.3D: Create 3-D model from cross-sections
'
'-------------
' Initialize.
'
strTitle = "Orebody Modeler"
docScene = av.GetActiveDoc
'-------------------------------------------------
' Create a list of active PolygonZ themes
' and PointZ themes.
'
listActiveThemes = docScene.GetActiveThemes
listPgzThemes = List.Make
listPtzThemes = List.Make
'-----------------------------------------
' Select PolygonZ themes and PointZ themes.
'
FOR EACH thmActive IN listActiveThemes
  IF (thmActive.Is(FTheme)) THEN
    fTabActive = thmActive.GetFTab
    strClassName =
fTabActive.GetShapeClass.GetClassName
    IF (strClassName = "PolygonZ") THEN
      listPgzThemes.Add(thmActive)
    ElseIf (strClassName = "PointZ") Then
      listPtzThemes.Add(thmActive)
    End
  END
END
'-------------------------------------------------
' If there is no polygon or point themes selected
' then exit.
'
If (listPgzThemes.Count < 1) THEN
  MsgBox.Error("Cross-sections in active themes
    required.", strTitle)
  EXIT
End
If (listPtzThemes.Count < 1) Then
  MsgBox.Error("Control points in active themes
    required.", strTitle)
  Exit
End
'-------------------------------
' Ask for the destination scene.
'
listScenes = {} ' look for 3-D Scene
FOR EACH docCurrent IN av.GetProject.GetDocs
  IF (docCurrent.Is(Scene)) THEN
    listScenes.Add(docCurrent)
  END
End
'-------------------------
' Select destination scene.
'
IF (listScenes.Count > 1) THEN
  scnDestination = MsgBox.ListAsString(listScenes,
    "Select destination scene:", strTitle)
ElseIf (listScenes.Count = 1) Then
  scnDestination = listScenes.Get(0)
Else
  MsgBox.Warning("No destination scene to display
    the result.", strTitle)
END
'-------------------------------------------------
' Compute average number of vertices in the polygons
' to inform user.
'
nbVertices = 0 ' number of vertices
nbPolygonZ = 0 ' number of polygons
For Each thmPgz In listPgzThemes
  fTabPgz = thmPgz.GetFTab
  '-----------------------------------------------
  ' Get the sum of vertices from all the polygons.
  '
  fldShape = fTabPgz.FindField("Shape")
  FOR EACH recFTab IN fTabPgz
    pgzIn = fTabPgz.ReturnValue(fldShape, recFTab)
    For Each listPolygonZ In pgzIn.AsList
      nbVertices = nbVertices + listPolygonZ.Count
      nbPolygonZ = nbPolygonZ + 1
    End
  END
End
nbVertices = (nbVertices / nbPolygonZ).Round '
average number of vertices from all the polygons
'-------------------------------------------------
' Get the number of samples to resample from user.
'
listNP = MsgBox.MultiInput("How many points do you
  want for laying out points?", strTitle,
  {"Horizontal:", "Vertical:"}, nbVertices.AsString,
  nbPolygonZ.AsString})
If (listNP.Count = 0) THEN
  Exit
Else
  strNH = listNP.Get(0)
  strNV = listNP.Get(1)
End
If ((strNH.IsNumber.Not) OR (strNH.AsNumber < 2))
THEN
  MsgBox.Error("Invalid number of points.",
strTitle)
  EXIT
End
If ((strNV.IsNumber.Not) OR (strNV.AsNumber < 2))
```

```
THEN
  MsgBox.Error("Invalid number of points.",
    strTitle)
  EXIT
End
'---------------------------
' Set the number as integer.
'
nbNH = strNH.AsNumber.Round
nbNV = strNV.AsNumber.Round
'-----------------------------------------------
' Set an argument list for Modeler.Resampler script.
'
listArgs = {listPgzThemes, listPtzThemes, nbNH,
  nbNV, docScene}
'-----------------------------------------
' Get the resampled points of each polygon.
'
listsGroupID = av.Run("Modeler.Resampler", listArgs)
' a list of the lists of resampled groups and their
IDs
listsGroup = listsGroupID.Get(0)
listGroupID = listsGroupID.Get(1)
If (listsGroup = nil) Then
  MsgBox.Error("Resampling failed.", strTitle)
  Exit
End
'---------------------------------------
' Patch the resampled points as surfaces.
'
thmModel = av.Run("Modeler.Patcher", {listsGroup,
  listGroupID, nbNV}) ' a MultiPath theme of model
If (thmModel = NIL) Then
  MsgBox.Error("Surface patching failed.", strTitle)
  Exit
End
'-----------------------------------------------
' Set new theme's attributes and add the theme
' to destination scene.
'
If (scnDestination <> NIL) THEN
  thmModel.SetVisible(TRUE)
  thmModel.SetName("Orebody Model")
  thmModel.Invalidate(FALSE)
  '-------------------------
  ' Set legend of the theme.
  '
  lgdModel = thmModel.GetLegend ' get the legend
  lgdModel.GetSymbols.UniformColor(Color.GetRed)
  '-----------------------------------
  ' Add the theme and refresh the scene.
  '
  scnDestination.AddTheme(thmModel)
  scnDestination.GetWin.Invalidate
  scnDestination.GetWin.Activate
  scnDestination.GetWin.Close
  scnDestination.GetWin.Open
End
'---------------
' End of script.
'===============
```

```
'=================================================
'
' MODELER.RESAMPLER: Partitions and resamples
' a polygon theme in 3 dimension
' << Called by Modeler.3D
'
'---------------
' Get arguments.
'
listPgzThemes = self.Get(0)
listPtzThemes = self.Get(1)
nbPtsHor = self.Get(2)
nbPtsVer = self.Get(3)
docScene = self.Get(4)
'-------------
' Initialize.
'
strTitle = "Modeler.Resampler" ' title
fnWorkDir = av.GetProject.GetWorkDir
'-----------------------------------------------
' Specify output shape-file of merged polygons.
'
If (listPgzThemes.Count > 1) Then
  fnDefault = fnWorkDir.MakeTmp("NewXS", "shp")
  fnOutput = FileDialog.Put(fnDefault, "*.shp",
    strTitle + "--Gathered PolygonZ")
  If (fnOutput = NIL) THEN
    Exit
  END
  fnOutput.SetExtension("shp")
  fTabPgz = FTab.MakeNew(fnOutput, PolygonZ)
  '------------------
  ' Add some fields.
  '
  fldElevationPgz = Field.Make("Elevation",
    #FIELD_FLOAT, 8, 2)
  fTabPgz.AddFields({fldElevationPgz})
  fldShapePgz = fTabPgz.FindField("Shape")
  '-----------------------------------
  ' Merge all polygon themes into one.
  '
  For Each thmPgz In listPgzThemes
    '----------------------------
    ' Get fields of input theme.
    '
    fTabPgzIn = thmPgz.GetFTab
    fldShapeIn = fTabPgzIn.FindField("Shape")
    fldElevationIn =
fTabPgzIn.FindField("Elevation")
    '---------------------------------
    ' Add each record to the new shape.
    '
For Each recPgz In fTabPgzIn
  objShape = fTabPgzIn.ReturnValue(fldShapeIn,
    recPgz)
  nbElevation =
    fTabPgzIn.ReturnValue(fldElevationIn,
    recPgz)
  nbRec = fTabPgz.AddRecord
  fTabPgz.SetValue(fldShapePgz, nbRec, objShape)
  fTabPgz.SetValue(fldElevationPgz, nbRec,
    nbElevation)
```

```
    End
  End
Else ' if there is only one polygon theme
  fTabPgz = listPgzThemes.Get(0).GetFTab
  fldShapePgz = fTabPgz.FindField("Shape")
  fldElevationPgz = fTabPgz.FindField("Elevation")
End
'-------------------------------------------
' Specify output shape-file of merged points.
'
If (listPtzThemes.Count > 1) Then
  fnDefault = fnWorkDir.MakeTmp("NewCP", "shp")
  fnOutput = FileDialog.Put(fnDefault, "*.shp",
strTitle + "--Gathered PointZ")
  If (fnOutput = NIL) THEN
    Exit
  END
  fnOutput.SetExtension("shp")
  fTabPtz = FTab.MakeNew(fnOutput, PointZ)
  '------------------
  ' Add some fields.
  '
  fldGroupIDPtz = Field.Make("Group ID",
    #FIELD_DECIMAL, 3, 0)
  fldNodeIDPtz = Field.Make("Node ID",
    #FIELD_DECIMAL, 3, 0)
  fldElevationPtz = Field.Make("Elevation",
    #FIELD_FLOAT, 8, 2)
  fTabPtz.AddFields({fldGroupIDPtz, fldNodeIDPtz,
    fldElevationPtz})
  fldShapePtz = fTabPtz.FindField("Shape")
  '-----------------------------------
  ' Merge all point themes into one.
  '
  For Each thmPtz In listPtzThemes
    '---------------------------
    ' Get fields of input theme.
    '
    fTabPtzIn = thmPtz.GetFTab
    fldShapeIn = fTabPtzIn.FindField("Shape")
    fldElevationIn =
fTabPtzIn.FindField("Elevation")
    fldGroupIDIn = fTabPtzIn.FindField("Group_ID")
    fldNodeIDIn = fTabPtzIn.FindField("Node_ID")
    '-----------------------------------
    ' Add each record to the new shape.
    '
    For Each recPtz In fTabPtzIn
      objShape = fTabPtzIn.ReturnValue(fldShapeIn,
        recPtz)
      nbElevation =
        fTabPtzIn.ReturnValue(fldElevationIn,
        recPtz)
      nbGroupID =
        fTabPtzIn.ReturnValue(fldGroupIDIn,
        recPtz)
      nbNodeID = fTabPtzIn.ReturnValue(fldNodeIDIn,
        recPtz)
      nbRec = fTabPtz.AddRecord
      fTabPtz.SetValue(fldShapePtz, nbRec, objShape)
      fTabPtz.SetValue(fldElevationPtz, nbRec,
        nbElevation)
      fTabPtz.SetValue(fldGroupIDPtz, nbRec,
        nbGroupID)
      fTabPtz.SetValue(fldNodeIDPtz, nbRec,
        nbNodeID)
    End
  End
Else ' if there is only one point theme
  fTabPtz = listPtzThemes.Get(0).GetFTab
  fldShapePtz = fTabPtz.FindField("Shape")
  fldElevationPtz = fTabPtz.FindField("Elevation")
  fldGroupIDPtz = fTabPtz.FindField("Group_ID")
  fldNodeIDPtz = fTabPtz.FindField("Node_ID")
End
fldGroupIDPtz.SetAlias("Group ID")
fldNodeIDPtz.SetAlias("Node ID")
'-------------------------------------------------
' Get the number of records in the polygon
' and the point themes.
'
nbPgz = fTabPgz.GetNumRecords ' number of polygons
nbPtz = fTabPtz.GetNumRecords ' number of points
'-------------------------------------------------
' Get the bitmaps of the polygons and the points.
'
bmPgz = fTabPgz.GetSelection
bmPtz = fTabPtz.GetSelection
'-------------------------
' Perform the conversion.
'
av.ShowMsg("Resampling...")
av.ShowStopButton
'-------------------------------------------------
' Make tables from the polygons and the points.
'
tblPgz = Table.Make(fTabPgz)
tblPtz = Table.Make(fTabPtz)
If ((tblPgz = NIL) Or (tblPtz = NIL)) Then
  MsgBox.Error("Unable to create tables.", strTitle)
  Exit
End
'-------------------------------------------------
' Make a list containing the indices of groups.
'
listGroupID = List.Make
'-------------------------------------------------
' Sort the table of points in ascending order
' of group.
'
tblPtz.GetWin.Open
tblPtz.Sort(fldGroupIDPtz, false)
'---------------------------------
' Find the indices of the groups.
'
For Each nbRow In 0 .. (nbPtz - 1)
  nbRecord = tblPtz.ConvertRowToRecord(nbRow)
  nbGroupID = fTabPtz.ReturnValue(fldGroupIDPtz,
    nbRecord)
  If (nbRow = 0) Then ' initial case
    listGroupID.Add(nbGroupID)
    nbGroupIDCurrent = nbGroupID
  ElseIf (nbGroupID <> nbGroupIDCurrent) Then
    listGroupID.Add(nbGroupID)
```

```
    nbGroupIDCurrent = nbGroupID
  End
End
'-------------------------------------------------
' Make a list of the lists containing elevations
' in each group.
'
listsElevation = List.Make
'-------------------------------------------------
' Sort the table of points in ascending order
' of elevation.
'
tblPtz.Sort(fldElevationPtz, false)
'-----------------------------------
' Find the elevations for each group.
'
For Each nbGroupID In listGroupID
  fTabPtz.Query("([Group ID] =" ++
    nbGroupID.AsString + ")", bmPtz,
    #VTAB_SELTYPE_NEW)
  fTabPtz.UpdateSelection
  tblPtz.PromoteSelection
  listGroup = List.Make
  For Each nbRow In 0 ..
    (fTabPtz.GetSelection.Count - 1)
    nbRecord = tblPtz.ConvertRowToRecord(nbRow)
    nbElevation =
      fTabPtz.ReturnValue(fldElevationPtz,
      nbRecord)
    If (nbRow = 0) Then ' initial case
      listGroup.Add(nbElevation)
      nbElevationCurrent = nbElevation
    ElseIf (nbElevation <> nbElevationCurrent) Then
     listGroup.Add(nbElevation)
      nbElevationCurrent = nbElevation
    End
  End
  listsElevation.Add(listGroup)
End
'-------------------------------------------------
' Sort the table of points in ascending order
' of node number.
'
tblPtz.Sort(fldNodeIDPtz, false)
'---------------------------
' Processing on each group.
'
listsGroup = List.Make
For Each nbIndexGroup In 0 ..
  (listGroupID.Count - 1)
  nbGroupID = listGroupID.Get(nbIndexGroup)
  listElevation = listsElevation.Get(nbIndexGroup)
listsCP = List.Make
listsRnP = List.Make
'-------------------------------
' Processing on each elevation.
'
For Each nbElevation In listElevation
  strQuery = "(([Group ID] =" ++
    nbGroupID.AsString + ") AND ([Elevation] =" ++
    nbElevation.AsString + "))"
  fTabPtz.Query(strQuery, bmPtz,
    #VTAB_SELTYPE_NEW)
fTabPtz.UpdateSelection
tblPtz.PromoteSelection
'---------------------------------------
' Select polygons in current elevation.
'
fTabPgz.Query("([Elevation] =" ++
  nbElevation.AsString + ")", bmPgz,
  #VTAB_SELTYPE_NEW)
fTabPgz.UpdateSelection
'---------------------------
' Processing on each point.
'
For Each nbRow In 0 ..
  (fTabPtz.GetSelection.Count - 1)
  nbRecordPtz = tblPtz.ConvertRowToRecord(nbRow)
  ptzControl = fTabPtz.ReturnValue(fldShapePtz,
    nbRecordPtz) ' get current control point
  '-----------------------------------------
  ' Search the closest point in the polygon
  ' to current control point.
  '
  nbCount = 0 ' count for initialization
  For Each nbRecordPgz In fTabPgz.GetSelection
    pgzOrebody =
      fTabPgz.ReturnValue(fldShapePgz,
      nbRecordPgz)
    listsPart = pgzOrebody.AsList
    '--------------------
    ' Loop for each part.
    '
    For Each nbPart In 0 ..
    (listsPart.Count - 1)
    If (nbPart = 0) Then
      nbIndexPart = nbPart
    End
    listPointZ = listsPart.Get(nbPart)
    listPointZ.Remove(listPointZ.Count - 1)
    '----------------------
    ' Loop for each point.
    '
    FOR EACH nbPoint IN 0 ..
      (listPointZ.Count - 1)
      ptzOrebody = listPointZ.Get(nbPoint)
      If (nbCount = 0) Then ' initialize
        nbClosest =
          ptzControl.Distance(ptzOrebody)
        nbIndexPoint = nbPoint
        nbIndexPart = nbPart
        nbIndexRecord = nbRecordPgz
      Else
        nbDistance =
          ptzControl.Distance(ptzOrebody)
        IF (nbClosest > nbDistance) THEN
          nbClosest = nbDistance
          nbIndexPoint = nbPoint
          nbIndexPart = nbPart
          nbIndexRecord = nbRecordPgz
        End ' if closest
      End ' if initial
      nbCount = nbCount + 1
    End ' for nbPoint
```

```
    nbGroupIDCurrent = nbGroupID
  End
End
'-------------------------------------------------
' Make a list of the lists containing elevations
' in each group.
'
listsElevation = List.Make
'-------------------------------------------------
' Sort the table of points in ascending order
' of elevation.
'
tblPtz.Sort(fldElevationPtz, false)
'-----------------------------------
' Find the elevations for each group.
'
For Each nbGroupID In listGroupID
  fTabPtz.Query("([Group ID] =" ++
    nbGroupID.AsString + ")", bmPtz,
    #VTAB_SELTYPE_NEW)
  fTabPtz.UpdateSelection
  tblPtz.PromoteSelection
  listGroup = List.Make
  For Each nbRow In 0 ..
    (fTabPtz.GetSelection.Count - 1)
    nbRecord = tblPtz.ConvertRowToRecord(nbRow)
    nbElevation =
      fTabPtz.ReturnValue(fldElevationPtz,
      nbRecord)
    If (nbRow = 0) Then ' initial case
      listGroup.Add(nbElevation)
      nbElevationCurrent = nbElevation
    ElseIf (nbElevation <> nbElevationCurrent) Then
     listGroup.Add(nbElevation)
      nbElevationCurrent = nbElevation
    End
  End
  listsElevation.Add(listGroup)
End
'-------------------------------------------------
' Sort the table of points in ascending order
' of node number.
'
tblPtz.Sort(fldNodeIDPtz, false)
'---------------------------
' Processing on each group.
'
listsGroup = List.Make
For Each nbIndexGroup In 0 ..
  (listGroupID.Count - 1)
  nbGroupID = listGroupID.Get(nbIndexGroup)
  listElevation = listsElevation.Get(nbIndexGroup)
listsCP = List.Make
listsRnP = List.Make
'-------------------------------
' Processing on each elevation.
'
For Each nbElevation In listElevation
  strQuery = "(([Group ID] =" ++
    nbGroupID.AsString + ") AND ([Elevation] =" ++
    nbElevation.AsString + "))"
  fTabPtz.Query(strQuery, bmPtz,
    #VTAB_SELTYPE_NEW)
fTabPtz.UpdateSelection
tblPtz.PromoteSelection
'---------------------------------------
' Select polygons in current elevation.
'
fTabPgz.Query("([Elevation] =" ++
  nbElevation.AsString + ")", bmPgz,
  #VTAB_SELTYPE_NEW)
fTabPgz.UpdateSelection
'--------------------------
' Processing on each point.
'
For Each nbRow In 0 ..
  (fTabPtz.GetSelection.Count - 1)
  nbRecordPtz = tblPtz.ConvertRowToRecord(nbRow)
  ptzControl = fTabPtz.ReturnValue(fldShapePtz,
    nbRecordPtz) ' get current control point
  '-----------------------------------------
  ' Search the closest point in the polygon
  ' to current control point.
  '
  nbCount = 0 ' count for initialization
  For Each nbRecordPgz In fTabPgz.GetSelection
    pgzOrebody =
      fTabPgz.ReturnValue(fldShapePgz,
      nbRecordPgz)
    listsPart = pgzOrebody.AsList
    '---------------------
    ' Loop for each part.
    '
    For Each nbPart In 0 ..
    (listsPart.Count - 1)
    If (nbPart = 0) Then
      nbIndexPart = nbPart
    End
    listPointZ = listsPart.Get(nbPart)
    listPointZ.Remove(listPointZ.Count - 1)
    '----------------------
    ' Loop for each point.
    '
    FOR EACH nbPoint IN 0 ..
      (listPointZ.Count - 1)
      ptzOrebody = listPointZ.Get(nbPoint)
      If (nbCount = 0) Then ' initialize
        nbClosest =
          ptzControl.Distance(ptzOrebody)
        nbIndexPoint = nbPoint
        nbIndexPart = nbPart
        nbIndexRecord = nbRecordPgz
      Else
        nbDistance =
          ptzControl.Distance(ptzOrebody)
        IF (nbClosest > nbDistance) THEN
          nbClosest = nbDistance
          nbIndexPoint = nbPoint
          nbIndexPart = nbPart
          nbIndexRecord = nbRecordPgz
        End ' if closest
      End ' if initial
      nbCount = nbCount + 1
    End ' for nbPoint
```

```
      End ' for nbPart
    End ' for nbRecordPgz
    listIndexPoint.Add(nbIndexPoint)
  End
  listsCP.Add(listIndexPoint)
  ListsRnP.Add({nbIndexRecord, nbIndexPart})
End

'-----------------------------------------------
' Make partition of the polygon according
' to the control points.
'
listsSection = List.Make
For Each nbIndex In 0 .. (listsCP.Count - 1)
  listsPtz = List.Make
  '---------------------------------------------
  ' Get the corresponding list of points
  ' in the polygon.
  '
  listRnP = listsRnP.Get(nbIndex)
  nbRecord = listRnP.Get(0)
  nbPart = listRnP.Get(1)
  pgzOrebody = fTabPgz.ReturnValue(fldShapePgz,
    nbRecord) ' get the polygon shape
  listPointZ = pgzOrebody.AsList.Get(nbPart)
  listPointZ.Remove(listPointZ.Count - 1)
  '---------------------------------------------
  ' Get the list of corresponding indices
  ' of control points.
  '
  listCP = listsCP.Get(nbIndex)
  nbCP = listCP.Count ' number of control points
  If (nbCP < 3) Then
    MsgBox.Warning("Number of control points must
      be at least 3.", strTitle)
    Continue
  Else
    listCP.Add(listCP.Get(0))
  End
  '---------------------------
  ' Check the sampling order.
  '
  If (listCP.Get(0) < listCP.Get(1)) Then
    bool1 = 1 Else bool1 = 0 End
  If (listCP.Get(1) < listCP.Get(2)) Then
    bool2 = 1 Else bool2 = 0 End
  If (listCP.Get(2) < listCP.Get(3)) Then
    bool3 = 1 Else bool3 = 0 End
  '---------------------------------------------
  ' In case the sampling order is identical
  ' to the control points.
  '
  If ((bool1 + bool2 + bool3) > 1) Then
    FOR EACH i IN 0 .. (nbCP - 1)
      listPtz = List.Make
      '-------------------------------------------
      ' When the first point of the polygon is in
      ' between the index.
      '
      If (listCP.Get(i) > listCP.Get(i + 1)) Then
        For Each j In listCP.Get(i) ..
          (listPointZ.Count - 1)
          listPtz.Add(listPointZ.Get(j))
        End
        For Each j In 0 .. listCP.Get(i + 1)
          listPtz.Add(listPointZ.Get(j))
        End
      Else
        For Each j In listCP.Get(i) ..
          listCP.Get(i + 1)
          listPtz.Add(listPointZ.Get(j))
        End
      End
      listsPtz.Add(listPtz)
    End ' for i
  '---------------------------------------------
  ' In case the sampling order is opposite
  ' to the control points.
  Else
    For Each i In 0 .. (nbCP - 1)
      listPtz = List.Make
      '-------------------------------------------
      ' When the first point of the polygon is in
      ' between the index.
      '
      If (listCP.Get(i) < listCP.Get(i + 1)) Then
        For Each j In listCP.Get(i) .. 0 By -1
          listPtz.Add(listPointZ.Get(j))
        End
        For Each j In (listPointZ.Count - 1) ..
          listCP.Get(i + 1) By -1
          listPtz.Add(listPointZ.Get(j))
        End
      Else
        For Each j In listCP.Get(i) ..
          listCP.Get(i + 1) By -1
          listPtz.Add(listPointZ.Get(j))
        End
      End
      listsPtz.Add(listPtz)
    End ' for i
  End ' if bool
  '---------------------------------------------
  ' Run parametric interpolation for each section.
  '
  listsPtzResampled = List.Make
  For Each listPtz In listsPtz
    listPtzResampled =
      av.Run("Modeler.Parametric", {listPtz,
      nbPtsHor})
    IF (listPtzResampled = NIL) THEN
      MsgBox.Warning("Failed parametric
        interpolation in horizontal dimension.",
        strTitle)
      Continue
    ElseIf (listPtzResampled.Count <> nbPtsHor)
Then
      MsgBox.Warning("Interpolation has not worked
        correctly:" + NL + NL + "Interpolated
        samples =" ++
        listPtzResampled.Count.AsString,
        strTitle)
      Continue
    ELSE
```

```
      listsPtzResampled.Add(listPtzResampled)
    End
  End
  listsSection.Add(listsPtzResampled)
End ' for nbIndex
'---------------------------------
' Processing on vertical dimension.
'
listsPtzResampled = List.Make
For Each nbSection In 0 .. (nbCP - 1)
  For Each nbPt In 1 .. (nbPtsHor - 1)
    '--------------------------
    ' Extract vertical section.
    '
    listPtz = List.Make
    For Each listSection In listsSection
      listPointZ = listSection.Get(nbSection)
      ptzHorizontal = listPointZ.Get(nbPt)
      listPtz.Add(ptzHorizontal)
    End
    '-----------------------------
    ' Run parametric interpolation.
    '
    listPtzResampled =
      av.Run("Modeler.Parametric", {listPtz,
      nbPtsVer})
    If (listPtzResampled = Nil) Then
      MsgBox.Warning("Failed parametric
        interpolation in vertical dimension on
        index" ++ nbPt.AsString, strTitle)
      Continue
    ElseIf (listPtzResampled.Count <> nbPtsVer)
      Then
      MsgBox.Warning("Interpolation has not worked
        correctly in vertical dimension on index"
        ++ nbPt.AsString + NL + NL + "Interpolated
        samples =" ++
        listPtzResampled.Count.AsString, strTitle)
      Continue
    Else
      listsPtzResampled.Add(listPtzResampled)
    End ' if
  End ' for nbPt
  listsGroup.Add(listsPtzResampled)
End
'-------------------
' Close the tables.
'
tblPgz.GetWin.Close
tblPtz.GetWin.Close
'-------------------------------------------------
' Return the list of vertically resampled points
' and the list of their group ID.
'
Return {listsGroup, listGroupID}
'----------------
' End of script.
'================
```

```
'=================================================
'
' MODELER.PARAMETRIC: Resamples polygon by 3-D
' Parametric Interpolation
' << Called by Modeler.Resampler
'
'-----------------------
' Get input parameters.
'
listPointZ = self.Get(0) ' list of given points
nbRspPts = self.Get(1)
strTitle = "Modeler.Parametric" ' string for title
'-------------------------------------------------
' Create a text file to write coordinates of given
' points.
'
fnWrite = "Given.txt".AsFileName
lfGiven = LineFile.Make(fnWrite, #FILE_PERM_WRITE)
If (lfGiven = NIL) Then
  MsgBox.Warning("Creating" ++ fnWrite.AsString ++
    "failed.", strTitle)
  Return NIL
End
'-------------------------------------------------
' Write x, y, z coordinates of the given points to
' the line file.
'
If (listPointZ.Count < 2) Then
  MsgBox.Warning("Too few number of points." + nl +
    "Check the nodes of the polygons.", strTitle)
  Return NIL
Else
  For Each ptzIn In listPointZ
    lfGiven.WriteElt(ptzIn.GetX.AsString ++
      ptzIn.GetY.AsString ++ ptzIn.GetZ.AsString)
  End
End
lfGiven.Close ' close the file
fnRead = "Resampled.txt".AsFileName
'-------------------------------------------------
' Perform resampling by parametric interpolation
' using cubic spline.
'
fnCWD = FileName.GetCWD
FileName.SetSearchPaths({fnCWD})
boolExist =
  FileName.ExistsInPaths("Parametric3D.exe")
If (boolExist) Then
  strCommand = "Parametric3D" ++ fnWrite.AsString ++
    fnRead.AsString ++ listPointZ.Count.AsString ++
    nbRspPts.AsString ++ "2"
  System.ExecuteSynchronous(strCommand)
Else
  MsgBox.Error("The external execution file,
    ""Parametric3D"", does not exist in current
    working directory." + nl + nl + "Current working
    directory is:" ++ fnCWD.AsString.Proper,
    strTitle)
  Exit
End
'-----------------------------------
' Open the resampled file to read.
```

```
'
lfResampled = LineFile.Make(fnRead, #FILE_PERM_READ)
' make a line file to read
If (lfResampled = NIL) Then
  MsgBox.Warning("Opening" ++ fnRead.AsString ++
    "failed.", strTitle)
  Return NIL
End
'-----------------------------------------------
' Read lines containing x, y, z coordinates of the
' resampled points.
'
listPointZ.Empty
For Each nbLine In 1 .. nbRspPts
  strLine = lfResampled.ReadElt ' read a line
  '-----------------------------
  ' Convert the line as a list.
  '
  If (strLine = NIL) Then
    Continue
  Else
    listResampled = strLine.AsTokens(32.AsChar)
  End
  '-------------------------------
  ' Get coordinates from the list.
  '
  nbX = listResampled.Get(0).AsNumber ' x coordinate
  nbY = listResampled.Get(1).AsNumber ' y coordinate
  nbZ = listResampled.Get(2).AsNumber ' z coordinate
  '-----------------------------------
  ' Make a point using the coodinates.
  '
  ptzResampled = PointZ.Make(nbX, nbY, nbZ)
  'MsgBox.Info(ptzResampled.AsString, strTitle)
  listPointZ.Add(ptzResampled)
End
lfResampled.Close ' close the file
Return listPointZ ' return a list of resampled
points
'---------------
' End of script.
'===============
```

```
'=================================================
'
' MODELER.PATCHER: Patches resampled points with
' MultiPatch surfaces
' << Called by Modeler.3D
'
'---------------
' Get arguments.
'
listsGroup = SELF.Get(0)
listGroupID = self.Get(1) ' a list of groups ID
nbSamples = self.Get(2) ' number of samples
strTitle = "Modeler.Patcher"
'---------------------------
' Specify output shapefile.
'
fnWorkDir = av.GetProject.GetWorkDir
fnDefault = fnWorkDir.MakeTmp("Model", "shp")
fnOutput = FileDialog.Put(fnDefault, "*.shp",
strTitle + "--Output Shapefile")
If (fnOutput = NIL) THEN
  Exit
END
fnOutput.SetExtension("shp")
'-----------------------------
' Create an appropriate FTab.
'
fTabModel = FTab.MakeNew(fnOutput, MultiPatch)
IF (fTabModel = NIL) THEN
  MsgBox.Error("Unable to create an output shape
    file.", strTitle)
  RETURN NIL
End
fldShape = fTabModel.FindField("Shape")
'------------------
' Add an ID field.
'
fldID = Field.Make("ID", #FIELD_DECIMAL, 3, 0)
fTabModel.AddFields({fldID})
'-------------------------------------------------
' Rearrange the vertical sections to make horizontal
' sections.
'
For Each nbGroup In 0 .. (listsGroup.Count - 1)
  '-------------------------
  ' Get a group and its ID.
  '
  listsSection = listsGroup.Get(nbGroup)
  nbID = listGroupID.Get(nbGroup)
  listsHorizontal = List.Make
  nbSection = listsSection.Count
  For Each nbPtz In 0 .. (nbSamples - 1)
    listSection = List.Make
    For Each nbIndex In 0 .. (nbSection - 1)
      listVertical = listsSection.Get(nbIndex)
      listSection.Add(listVertical.Get(nbPtz))
    End
    listsHorizontal.Add(listSection)
End
'----------------
' Do conversion.
'
```

```
mpModel = MultiPatch.MakeNull
'-----------------------------------------
' Make surface from the horizontal sections.
'
For Each i In 1 .. (nbSamples - 1)
  '---------------------------------------
  ' Get preceding and succeeding sections.
  '
  listPreceding = listsHorizontal.Get(i - 1)
  listSucceeding = listsHorizontal.Get(i)
  '-----------------------------------------
  ' Add the first point to the end to make
  ' a closed surface.
  '
  listPreceding.Add(listPreceding.Get(0))
  listSucceeding.Add(listSucceeding.Get(0))
  '-----------------------------------------
  ' Merge the two sections in interlaced order of
  ' points.
  '
  listPointZ = List.Make
  FOR EACH j IN 0 .. nbSection
    listPointZ.Add(listPreceding.Get(j))
    listPointZ.Add(listSucceeding.Get(j))
  End
  '-----------------------------------------
  ' Create a MultiPatch shape from the merged list
  ' as triangle-strip.
  '
  mpModel.AddPart(listPointZ,
    #PART_TYPE_TRIANGLESTRIP)
End
  '---------------------------------
  ' Add the MultiPatch to the FTab.
  '
  nbRecord = fTabModel.AddRecord
  fTabModel.SetValue(fldShape, nbRecord, mpModel)
  fTabModel.SetValue(fldID, nbRecord, nbID)
  fTabModel.Flush
  fTabModel.Refresh
End
'-----------------------------
' Make a new theme to return.
'
thmModel = FTheme.Make(fTabModel)
RETURN thmModel
'---------------
' End of script.
'===============
```

ABSTRACT

A Study on Development of 3-D Visualization System for Subsurface Information

Kim, Hyeongyu

Major in Information System Engineering

Dept. of Information System Engineering

Graduate School of Hansung University

This study concerns the development of subsurface information system that can analyze various subsurface information easily and promptly. The data types that I posed to visualize are 3-D latticed array acquired by geophysical explorations and linear array acquired from wells. Volume visualization techniques are applied to visualize 3-D latticed data. The well-core data are digitized to a database and the subsurface objects such as wells and orebodies are visualized three-dimensionally to allow users to extract accurate and realistic information rapidly.

A visualization module to systematically analyze and manage three-dimensional seismic data is developed. The visualization is performed by converting each sam-

ple to vertical or horizontal polygon and assigning seismic information to its attribute. The spatiality of the visualized slice or volume is controlled by dialog boxes of sliders and buttons. What is more for the module is that it can visualize a wireline log as a pipeline form with its log values allotted to a database table. Another more is the picking ability for seismic horizons. The points picked in guide of this module can be delivered to the modeling module developed here to visualize a seismic event.

A modeling software, that constructs 3-D shapes by interpolating 2-D cross-sections, is developed. The software can extract a cross-section in any direction and inclination from the model, then update its form by adjusting the cross-section iteratively. This software was applied to real data to model an orebody. It is found that the most important point for high-quality modeling is the setup of control points, which is influenced by experience and knowledge of the user. The modeling software has additional tool that abstracts lithology on a certain elevation from available core descriptions, which is valuable for verification of the model. This function ascertained that there are some mismatches between the cores and the model, which is caused by exclusion of informations from vertical and inclined wells when producing the input cross-sections.

The database system of geologic columns simplified the implementation of storage, modification, and maintenance of the data, and it also enabled the real-time retrieval for the voluminous amount of data, which was not possible by hand. The seismic cube, well logs, and orebodies that are embodied as 3-D objects will provide spatially expanded information to help us reach a better decision in designing ground/underground structures by giving higher reality than those in 2-D.