윤곽선을 이용한 다중해상도적 복원
(Multiresolutional Reconstruction from Contours)

민 경 하 † 이 인 권 ††
(KyungHa Min) (In-Kwon Lee)

요 약 본 논문에서는 윤곽선들의 집합으로부터 다각형 곡면을 복원하는 다중해상도적 방법을 제안한다. 그 방법의 첫 단계에서는 방사 조사법을 적용해서 체적 데이타로부터 추출된 여러 장의 영상들로부터 윤곽선을 추출한다. 그리고 이 윤곽선을 구성하는 선분들의 유형들을 분류한 다음, 이를 이용해서 윤곽선을 context-free 문법으로 표현하는 방법을 제시한다. 두 번째 단계에서는 이웃하는 두 윤곽선들 사이에 다각형들을 생성하는 것으로, 이를 위하여 context-free 문법의 유도 트리를 검색함으로써 두 윤곽선들 사이에서 서로 대응되는 꼭지점과 선분을 찾는 방법을 제시한다. 이러한 단계를 거쳐서 생성된 가장 낮은 해상도의 다각형 곡면은 각 윤곽선을 세분화함으로써 더 높은 해상도의 다각형 곡면으로 세분화된다. 여기서 윤곽선은 각 영상에서 더 많은 광선을 방사함으로써 세분화된다. 본 연구에서는 방사 조사법을 이용해서 추출된 다양한 해상도의 윤곽선들의 연결도가 일정하게 유지됨을 이용해서 다양한 해상도의 다각형 곡면들의 연결도는 일정하게 유지됨을 보장한다. 제안된 방법은 효율적인 복원과 복원된 물체의 외형적인 특징을 보장하는 압축 방법을 제공한다.

키워드 : 복원, 윤곽선, 다중해상도, 메쉬, context-free 문법

Abstract A new multiresolutional scheme that reconstructs a polygonal mesh from the set of contours is presented. In the first step, we apply a radial gradient method to extract the contours on the sampled slices from a volume data. After classifying the types of the edges on the contours, we represent the contour using the context-free grammar. The polygons between two neighboring contours are generated through the traversal of the derivation trees of the context-free grammar. The polygonal surface of the coarsest resolution is refined through the refinement of the contours, which is executed by casting more rays on the slices. The topologies between the polygonal surfaces of various resolutions are maintained from the fact that the radial gradient method preserves the topologies of the contours of various resolutions. The proposed scheme provides efficient computation and compression methods for the tiling procedure with the feature preservation.

Key words : reconstruction, contour, multiresolution, mesh, context-free grammar

1. 서 론

윤곽선을 이용한 다각형 곡면의 복원은 의료 영상, 캐드 및 역 엔지니어링 등과 같은 다양한 분야에서 매우 중요한 문제이다. 복원을 위해서는 대응(correspondence), 분지(branching), 타일링(tiling)의 세 가지 문제를 해결해야 한다[1]. 대응 문제는 서로 이웃하는 영상들에 놓인 두 집합의 윤곽선들로부터 서로 대응되는 윤곽선을 찾는 문제이다. 그림 1에서는 대응 문제의 어려움을 보여주고 있다. 분지 문제는 한 윤곽선이 두 개 이상의 윤곽선들로 대응될 경우, 다각형 곡면을 생성하는 방법에 관한 것이며, 타일링 문제는 서로 대응되는 두 윤곽선 사이에서 어떻게 다각형들을 생성할지를 결정하는 것이다. 본 연구에서는 타일링 문제를 해결하기 위한 방법을 제시한다.

타일링 과정을 수행하기 위해서는 두 이웃하는 윤곽선들에 대해서 대응되는 꼭지점(corresponding vertex)이나 선분(corresponding edge)들을 찾는 과정이 수행되는데, 이 과정에서 요구되는 계산량을 최소화하는 것은 타일링에서 매우 중요한 문제이다. 기존의 타일링 방법들은 대응되는 꼭지점이나 선분을 찾기 위해서 두 윤곽선들의 기하학적인 정보를 이용하기 때문에 많은 계산량을 요구하고 있다. 윤곽선들 사이에 다각형을 생성하는 과정에서의 또 다른 요구 사항은 작은 수의 다각

† 비 회 원 : 미국 Rutgers Univ. 박사후연구원
minkh@cs.rutgers.edu
†† 정 회 원 : 연세대학교 컴퓨터과학과 교수
iklee@yonsei.ac.kr
논문접수 : 2001년 10월 23일
심사완료 : 2003년 8월 11일

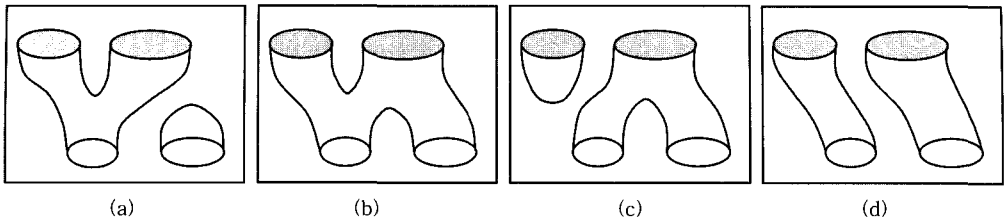


그림 1 대응 문제의 어려움: 동일한 윤곽선의 집합으로부터 서로 다른 네 가지의 대응 관계를 얻을 수 있다.

형을 생성해야 하는 것과 생성된 다각형들을 보기 좋은 형태의 삼각화(fair triangulation)로 표현해야 하는 것이다. 세 번째 요구 사항은 생성된 곡면을 다중해상도적인 표현 방법으로 나타내는 것이다. 몇몇의 연구자들은 가장 높은 해상도의 곡면을 생성하고[2] 이를 단순화함으로써[3,4] 더 낮은 해상도의 곡면을 얻는 2 단계의 다중해상도적 표현방법을 얻는 방법들을 제시하였다[8, 18]. 단계적인 상세화를 통한 다중해상도적 표현 방법은 각 해상도의 다각형 곡면들의 연결도(topology)를 유지하기 힘들기 때문에, 아직까지 제시되지 않고 있다.

본 연구에서는 체적 데이타에서 샘플링된 단층 영상들로부터 복원하고자 하는 물체의 윤곽선을 추출하고, 이 윤곽선들로부터 물체의 3차원 형태를 복원하는 방법을 제시한다(그림 2).

본 연구에서 제안하는 복원 방법의 특징은 가장 낮은 해상도의 물체의 형태(본 연구에서는 이를 기본 형태로 명명한다)를 복원하고, 이를 점진적으로 상세화시켜서 더 높은 해상도의 물체의 형태를 구하는 것이다. 이 방법은 다음과 같은 과정을 통해서 수행된다.

Step. 1: 기본 형태 복원

Step 1.1: 방사 조사법(radial gradient method)을 이용해서 체적 데이타의 단층 영상으로부터 가장 낮은 해상도의 윤곽선(기본 윤곽선)을 추출한다. 방사 조사법은 영상의 관심 영역(ROI:region of interest)의 중심점으로부터 광선을 방사하여 관심 영역의 경계면과의 교차점을 생성하고, 이 교차점들을 연결함으로써 닫혀진 다각형을 생성한다(그림 3의 (a)). 이 방법은 3장에서 제시되어 있다.

Step 1.2: 교차점을 연결하는 선분의 유형을 그림 7과 같이 정의하고, 이를 이용해서 윤곽선을 Context-free 문법을 표현한다. 그리고 이 context-free 문법의 유도 트리를 이용해서 기본 윤곽선으로부터 기본 형태를 복원한다(그림 3의 (b)). 이 방법은 4장에서 설명되어 있다.

Step. 2: 상세화

Step 2.1: 방사 조사법에서 전 단계에서 방사된 기존

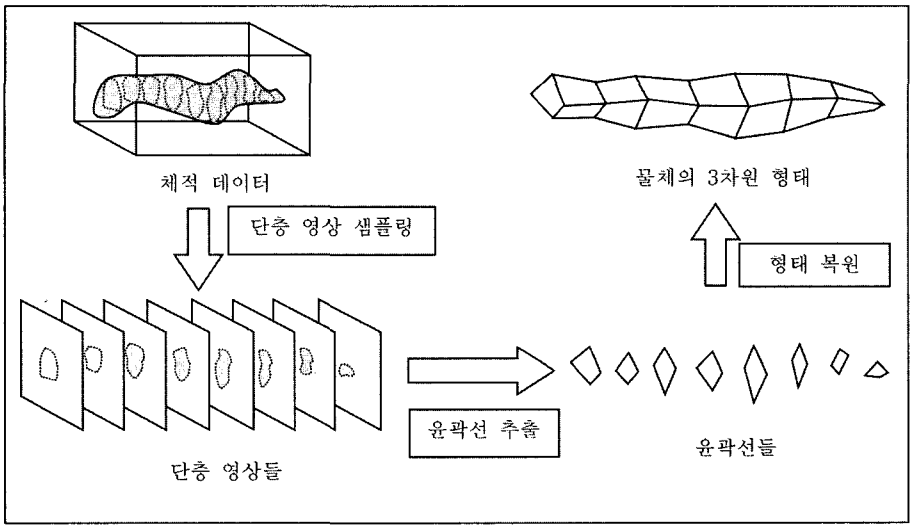
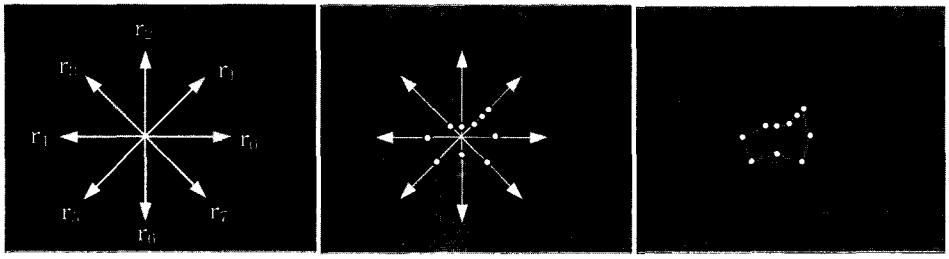
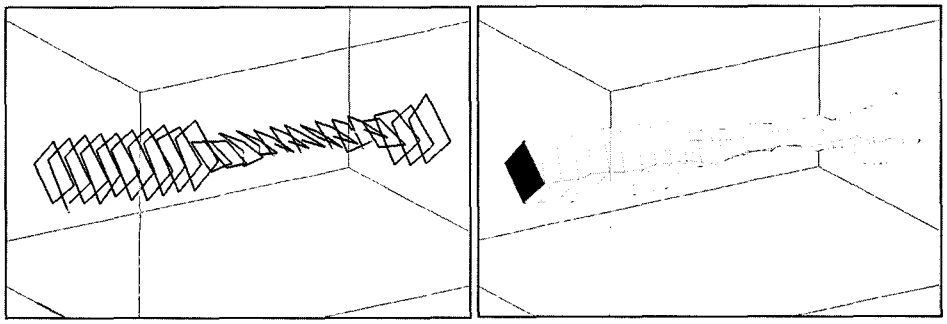


그림 2 제안하는 알고리즘의 개요



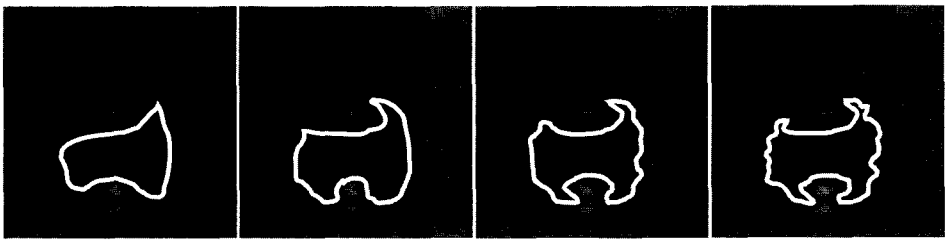
n_r 개의 광선을 방사 교차점의 계산 기본 윤곽선의 추출

(a) Step 1.1 : 방사 조사법 (n_r = 8인 경우)

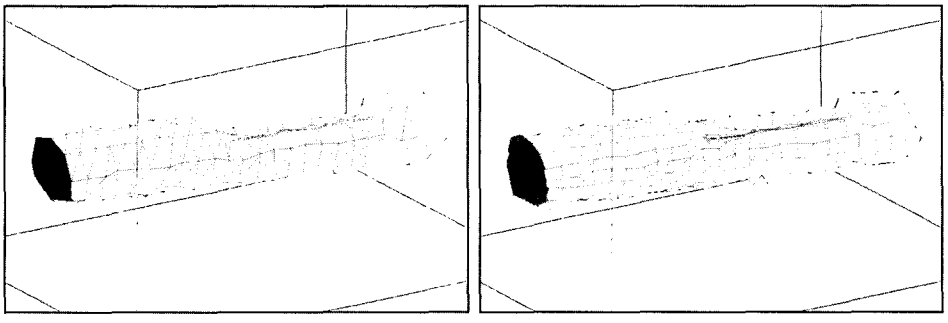


n_r = 4인 경우의 기본 윤곽선들 기본 윤곽선으로부터 복원된 기본 형태

(b) Step 1.2 : 기본 윤곽선으로부터 복원된 기본 형태



(c) Step 2.1 : 윤곽선의 상세화 (좌측으로부터 우측으로)



(d) Step 2.2 : 형태의 상세화 (좌측으로부터 우측으로)

그림 3 윤곽선의 추출과 물체의 형태 복원 과정

의 광선들 사이에 새로운 광선을 방사해서 윤곽선을 상세화한다(그림 3의 (c)). 이 과정에서 상세화된 윤곽선은 전단계의 윤곽선을 표현하는 유도 트리의 잎새 노드에 새로운 자식 노드를 추가함으로써 표현된다. 이 방법은 5장 1절에서 제시되어 있다.

Step 2.2: 상세화된 윤곽선의 유도 트리를 이용해서 물체의 상세화된 형태를 복원한다(그림 3의 (d)). 이 방법은 5장 2절에서 제시되어 있다.

본 논문에서 제시하는 가장 중요한 제안점은 두 윤곽선들 사이의 다각형을 생성하기 위하여 선분의 유형을 이용한다는 점이다. 방사 조사법을 이용해서 생성된 윤곽선들은 윤곽선 상의 선분들의 순서를 제약하는 몇 가지의 성질들을 가지고 있다[4-7]. 따라서 이러한 성질들을 기반으로 하여 각 윤곽선을 인식하는 context- free 문법을 설계한다. 이러한 방법은 타일링 문제에서 제시되는 계산량과 생성된 삼각형의 질이라는 두 개의 요구사항을 해결한다. 제7장에서는 기존의 toroidal 검색 방법과 본 연구에서 제시하는 방법의 성능을 비교함으로써 제시된 방법을 우수성을 증명한다. 그리고, context-free 문법은 곡면의 상세화에도 적용될 수 있다. 윤곽선을 상세화하는 과정은 더 많은 광선을 방사함으로써 수행되기 때문에, 윤곽선의 상세화는 그 윤곽선의 유도 트리를 상세화하는 과정을 통해서 처리되기 때문이다. 그러므로 다각형 곡면의 다각형들은 이 상세화된 유도 트리를 탐색함으로써 상세화 될 수 있다.

본 논문의 공헌점은 다음과 같다.

1. 단계적 상세화: 곡면을 단계적 상세화를 통해서 복원하기 위해서는 가장 낮은 해상도로부터 가장 높은 해상도까지의 연결도 (topology)가 일정하게 유지되어야 한다. 본 연구에서는 다양한 해상도의 윤곽선들의 연결도를 보존하는 방사 조사법을 이용해서 윤곽선을 추출하고 이를 이용해서 다각형 곡면을 생성하기 때문에 생성된 곡면의 연결도는 해상도의 변화에도 일정하게 유지된다.

2. 효율성: 두 윤곽선들에 대해서 대응되는 꼭지점과 선분들이 유도 트리의 탐색을 통해서 결정되기 때문에 본 연구에서 제안하는 타일링 방법은 수치적 계산량을 감소시킨다. 특히, 본 연구에서 이용되는 계산은 정수의 비교 연산이기 때문에 부동 소수점의 덧셈, 곱셈, 제곱근 계산 등을 통해서 거리를 계산하는 기존의 방법에 비해서 분명하게 작은 계산량만을 요구한다.

3. 압축: 두 윤곽선 사이의 삼각형들은, 순차적으로 배열된 윤곽선들의 꼭지점을 연결함으로써 생성되기 때문에 삼각띠(triangular strip)를 이루게 된다. 그리고, 윤곽선의 각 꼭지점들 광선에 의해서 결정되기 때문에 꼭지점의 유클리드 좌표(Euclidean coordinate)는 각과 반지름으로 표현되는 극 좌표계(polar coordinate)로 변형될 수 있다. 이 과정에서 각 꼭지점들과 광선과의 제약 사항 때문에 좌표에서 각을 생략할 수 있다. 이러한 압축 방법을 3:1 이상의 압축비를 제공한다.

4. 특징의 보존: 본 연구에서 제시하는 타일링 방법은 각 영상에서의 광선을 제어함으로써 윤곽선의 특징점을 보존할 수 있다. 그림 18과 그림 19에서 비교되어 있듯이 꼭지점의 기하학적 정보만을 이용하는 기존의 타일링 방법은 물체의 특징을 표현하지 못하는 경우가 있다. 그러나, 본 연구에서 제안된 방법은 다양한 해상도에서 이웃하는 윤곽선의 특징을 정확하게 복원할 수 있음을 보인다.

본 논문은 다음과 같이 구성되어 있다. 제2장에서는 윤곽선으로부터 곡면을 복원하는 방법에 관한 관련된 연구들을 기술한다. 제3장과 제4장에서는 광선을 방사함으로써 윤곽선을 추출하는 개선된 방사 조사법과 이 윤곽선들로부터 곡면을 복원하는 타일링 방법이 각각 소개된다. 그리고, 제5장에서는 곡면에 대한 상세화 방법이 제시되며, 제6장에서는 제안된 알고리즘의 결과를 보여주고, 이를 다른 방법의 결과와 비교하며, 압축과 특징 보존에 대한 방법을 간략하게 기술한다. 마지막으로 제7장에서는 본 연구의 결론과 향후의 연구 방향을 제시한다.

2. 관련 연구

Keppel[8]은 처음으로 2차원 윤곽선으로부터 곡면을 복원하는 방법을 제시하였다. 그의 방법은 윤곽선의 각 꼭지점들에 대해서 toroidal 그래프를 생성하고, 이 그래프를 탐색하는 방법을 통해서 두 윤곽선들 사이에서 서로 대응되는 꼭지점을 찾음으로써 다각형을 생성한다. Fuchs 등[9]은 toroidal 그래프의 탐색 방법에서 분할-정복 방법(divide and conquer)을 도입함으로써 Keppel의 방법을 개선하였다. Christiansen 등[5]은 같은 레벨의 두 윤곽선 사이의 중간점을 계산함으로써 두 개의 윤곽선을 통합하는 방법을 제시하였다. 이 방법에서는 Keppel이 제안한 toroidal 그래프에 대해서 탐욕스러운 탐색 방법(greedy search algorithm)을 적용해서 타일링 문제를 해결하였다. Boissonnat[10]은 서로 이웃한 윤곽선의 쌍에 대해서 딜러니 삼각화(Delaunay triangulation)을 적용함으로써 두 윤곽선 사이의 공간을 삼각뿔화(tetrahedralization)함으로써 다각형을 생성하는 방법을 제안하였다. 즉, 된 삼각뿔들의 내부의 선분들을 제거해서 두 윤곽선 사이를 보간함으로써 곡면을 생성한다. Ekoule 등[11]은 convex가 아닌 윤곽선들을 convex한 여러 개의 보조적인 윤곽선들로 분할하고, 이 보조 윤곽선들에 대해서 타일링 알고리즘을 적용하는 방

법을 제시하였다. Meyers 등[1]은 다각형을 생성할 윤곽선들을 타원을 이용해서 근사하고 효율적인 타일링을 위해서 최소 비용 신장 트리(minimum cost spanning tree)를 생성함으로써 곡면을 생성하는 방법을 구현하였다. Jones 등[12]은 윤곽선을 이산적인 필드 함수 (discrete field function)를 이용해서 표현하고, 이 함수들을 보간함으로써 윤곽선들 사이에 다각형을 생성하는 방법을 제공하였다. Bajaj 등[13]은 복원될 곡면에 대해서 제약 조건을 부여하고 이 제약 조건으로부터 정확한 대응 법칙과 타일링 방법을 유도함으로써 복원을 위한 세 가지의 문제를 동시에 해결하였다. Oliva 등[14]은 윤곽선들을 하나의 공통된 평면에 투사하고 이들의 교차 영역에 일반화된 보로노이 다이어그램(Voronoi diagram)을 적용함으로써 곡면을 생성하는 방법을 제시하였다. 이 방법에서 두 윤곽선들 사이의 곡면은 보로노이 다이어그램을 삼각형화함으로써 생성된 삼각형들을 보간함으로써 생성된다. Cong 등[15]은 그 해가 윤곽선을 나타내는 필드 함수인 편미분방정식을 생성하고, 이 편미분방정식의 해를 구함으로써 윤곽선들에 대한 타일링 방법을 해결하였으며, Klein 등[1]은 두 윤곽선의 교차 영역에서의 중심 축(medial axis)을 계산함으로써 윤곽선들의 꼭지점들 사이의 대등 관계를 찾는 방법을 제시하였다. 그리고 Marsan 등[18]은 morphological dilation 연산자를 적용함으로써 윤곽선들의 보간을 계산하였으며, Fujimura[3]는 타일링을 위해서 윤곽선들을 보간하기 위해서 isotopic 변형 방법을 제안하였다.

몇몇의 연구자들은 윤곽선들로부터 곡면을 복원하는 다중해상도적인 방법을 제시하였다. Meyers[6]는 웨이블릿(wavelet)에 기반한 다중해상도적 방법을 제시하였는데, 이 방법은 두 윤곽선을 복원하기 위해서 $O(n)$의 공간 복잡도(space complexity)와 $O(n\ log\ n)$의 시간 복잡도(time complexity)를 가진 방법을 제시하였는데, Fuchs 등[9]이 제안한 검색 방법은 $O(n^2)$의 공간 복잡도와 $O(n^2\ log\ n)$의 시간 복잡도를 가지고 있었다. Schilling 등[16]은 중심축(medial axis)를 이용한 타일링 방법과 선분 융합(edge collapse)에 기반한 단순화 방법을 제시하였다. 그리고, Fix 등[17]은 Meyer가 제시한 다중해상도적인 방법을 점진적인 상세화 방법으로 확장하였다. 이 방법에서는 먼저, 가장 높은 해상도의 윤곽선에 대해서 웨이블릿 분할 알고리즘을 적용함으로써 더 낮은 해상도의 윤곽선들을 구한다. 그 다음 단계에서는 동적 프로그래밍 기법(dynamic programming)을 적용해서 가장 낮은 해상도의 윤곽선들에 대한 최적의 타일링을 계산한다. 그리고, 낮은 해상도의 윤곽선들의 타일링을 상세화 하는 방법을 통해서 더 높은 해상도의 윤곽선들을 타일링한다. 결과적으로 더 높은 해상도의 곡면은 banded refinement라 불리는 제한된 동적 프로그래밍 기법을 통해서 생성된다.

3. 개선된 방사 조사법

3.1 윤곽선의 추출

물체를 복원할 체적 데이타는 $n \times m \times l$ 크기의 배열로 표현된 정수의 집합인데, 각 원소의 값들은 [0, 255] 내에 들어있다. 만약 z 축을 수집할 영상들에 대한 수직 축으로 설정한다면, 이 체적 데이타로부터는 l 개의 영상들을 수집할 수 있다. 각 영상들로부터 복원하고자 하는 물체와 영상과의 교차 픽셀들의 집합으로 관심 영역을 정의하고, 이 관심 영역의 윤곽선을 추출한다. 이 추출 과정의 전단계로 관심 영역의 중심점을 계산해 기 위해서 다음과 같은 알고리즘을 제시한다.

$(n \times m)$의 크기를 가진 영상에 있는 관심 영역의 중심점 $c = (c_x, c_y)$는 다음과 같이 계산된다.

(1) $[a, b]\ X\ [1, m]$의 영역 내에 포함된 관심 영역의 픽셀의 수를 $\int_a^b RoI\,dx$로 정의하면, c_x는 다음의 조건을 만족하는 값으로 결정된다.

$$\int_1^{c_x} RoI\,dx = \int_{c_x}^{n} RoI\,dx$$

(2) $[1, n]\ X\ [a, b]$의 영역 내에 포함된 관심 영역의 픽셀의 수를 $\int_a^b RoI\,dy$로 정의하면, c_y는 다음의 조건을 만족하는 값으로 결정된다.

$$\int_1^{c_y} RoI\,dy = \int_{c_y}^{m} RoI\,dy, \text{ and } (c_x, c_y) \in RoI$$

윤곽선 추출의 다음 과정으로 관심 영역의 경계면 위의 점들을 수집하기 위하여 광선을 방사한다. 이 광선의 시작점은 중심 픽셀의 네 꼭지점을 평균한 점으로 설정된다. 만약 n_r 개의 광선이 방사된다면, 이 광선들은 각각 r_0, ..., r_{nr-1}로 표시되며 반시계 방향으로 일정한 간격으로 방사된다. 이 때에 r_0의 방향은 x 축의 양의 방향으로 가정한다. 예를 들어, n_r = 4인 경우, 광선의 네 방향은 각각 (1,0), (0,1), (-1,0), (0,-1)이 된다. 여러 영상들에 대해서 유사한 형태의 윤곽선을 추출하기 위해서, 본 연구에서는 각 영상들에 대해서 동일한 집합의 광선들을 방사한다. 그림 3의 (a)에서 제시된 Step 1.1은 n_r = 8개의 광선이 방사된 예를 보여준다.

만약 어떤 광선이 관심 영역의 경계면 위의 픽셀(boundary pixel)과 만나고 있다면, 그 경계 픽셀의 경계 선분(border edge)과 그 광선과의 교차점이 계산된다(그림 4(a)). 그림 4(a)에서 관심 영역의 경계 픽셀들은 더 짙은 회색의 픽셀들로 표시되어 있으며, 경계 픽셀의 경계 선분은 더 두꺼운 선들로 표시되어 있다. 각 교차점들은 IN-OUT 또는 OUT-IN이라는 두 가지의

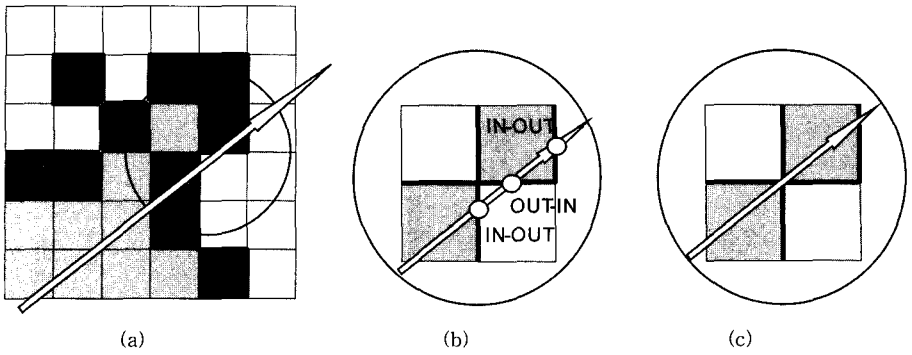


그림 4 광선과 교차점들

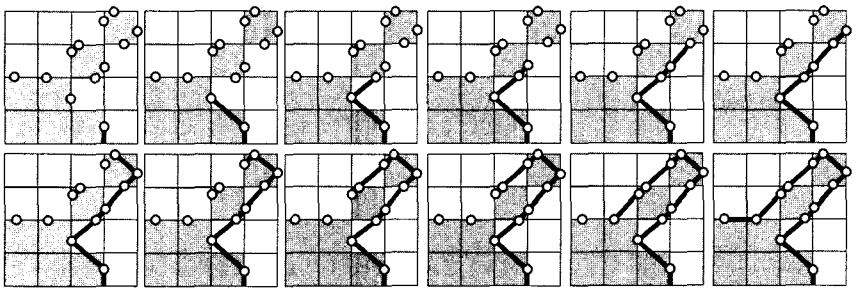


그림 5 교차점을 연결하는 예: 다음 교차점은 다음 경계 선분위에 놓여있다.

유형으로 분류된다. 그림 4(b)에 제시되어 있듯이 OUT-IN 교차점에서는 광선이 관심 영역의 밖으로 벗어나며, IN-OUT 교차점에서는 광선이 관심 영역의 안으로 들어오게 된다. 그림 4(c)의 경우와 같이 만약 광선이 두 경계 픽셀의 구석을 지나가게 될 경우에는 교차점을 계산하지 않는다. 이러한 개선된 방사 조사법을 통해서 수집된 교차점들은 다음의 공리 1을 만족시킨다.

공리 1

i. 각 광선과 관심 영역은 적어도 하나의 교차점을 갖는다.

ii. 한 광선에 의해서 결정되는 교차점의 수는 홀수이다.

iii. 중심점에서 가장 가까운 교차점의 유형은 IN-OUT이다.

iv. 한 광선을 따라서 결정되는 교차점의 유형은 IN-OUT(OUT-IN)과 OUT-IN (IN-OUT)이 반복해서 나타난다.

본 논문에서는 각 교차점들과 경계 선분들의 자료 구조에 서로를 참조하는 포인터를 추가함으로써 교차점들을 연결하는 과정을 효율적으로 처리하였다.

본 논문에서는 체인 코드(chain code) 알고리즘[21]을 이용해서 윤곽선 상에서의 다음 교차점을 결정한다. 체인 코드 알고리즘은 널리 알려진 영상 처리 기술로 관심 영역의 경계 픽셀들을 차례로 방문하는 알고리즘이다. 이 체인 코드 알고리즘은 Bribiesca의 연구[3]와 같이 경계 픽셀의 경계 선분을 차례로 방문하는 알고리즘으로 변경되어서 이용된다. 이 알고리즘을 이용함으로써 관심 영역의 경계 선분들은 첫 번째로 선택된 교차점을 포함하는 경계 선분으로부터 반시계 방향으로 방문된다. 이 과정에서 만약 교차점에 대한 포인터를 포함하고 있는 경계 선분을 방문하게 되면, 이 교차점을 윤곽선에 추가하고 처음의 경계 선분을 다시 방문할 때까지 경계 선분들을 방문하는 과정을 계속한다. 그림 5에는 다음에 방문할 경계 선분을 결정하는 몇 가지의 경우가 제시되어 있다.

결과로 얻어지는 윤곽선은 가장 작은 수의 광선을 방사해서 얻은 윤곽선이기 때문에 가장 낮은 해상도를 가지고 있으며, 이는 기본 윤곽선이라고 표시된다. 그림 6(a)에서 제시되어 있듯이 기본 윤곽선은 단순한 다각형(simple polygon)이 아닌 경우도 존재한다. 이 그림에서는 r1에 의해서 결정된 아래로 향한 선분과 위로 향한 선분이 중첩되어 있다. 이러한 경우를 해결하기 위하여 본 논문에서는 다음과 같은 두 가지의 방법을 제시한다. 그림 6(b)에서의 예와 같이 중첩된 선분을 가진 광선들 사이에 새로운 광선을 방사함으로써 기본 윤곽선을 세분화하거나, 또는 그림 6(c)에서와 같이 위와 같은 경우를 발생시키는 몇몇의 선분들을 제거한다. 이 방법은 윤

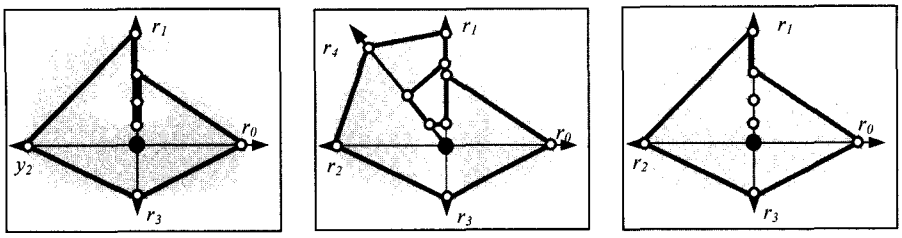


그림 6 기본 윤곽선의 잘못된 예와 그 해결 방법

곽선을 세분화하는 것이 불가능할 경우에 수행된다.

3.2 윤곽선의 분석

윤곽선 위의 선분들은 그 끝점들과 광선과의 관계에 따라서 몇 가지의 유형으로 분류될 수 있다. 선분들의 유형은 다음과 같이 정의된다.

정의 1

윤곽선 위의 선분들은 그 시작점과 끝점 및 그 점들을 결정하는 광선에 의해서 다음과 같은 네 유형 중의 하나로 표시될 수 있다.

1. FOR(forward) 선분: 시작점이 r_i에 의해서 결정되고 끝점이 r_{i+1}에 의해서 결정되는 선분(그림 7(a)).
2. BACK(backward) 선분: 시작점이 r_{i+1}에 의해서 결정되고 끝점이 r_i에 의해서 결정되는 선분 (그림 7 (b)).
3. DOWN(downward) 선분: 시작점과 끝점이 모두 같은 광선에 의해서 결정되며, 그 방향이 광선과 반대되는 선분. DOWN 선분은 $DOWN_I$ (Down-In)과 $DOWN_O$ (Down-Out)으로 더 분류할 수 있는데, $DOWN_I$ 선분은 OUT-IN 교차점에서 IN-OUT 교차점으로 진행하며, $DOWN_O$ 선분은 IN-OUT 교차점에서 OUT-IN 교차점으로 진행한다 (그림 7 (c)).
4. UP(upward) 선분: 시작점과 끝점이 모두 같은 광선에 의해서 결정되며, 그 방향이 광선과 일치하는 선분. UP 선분은 UP_I (Up-In)과 UP_O (Up-Out)으로 더 분류할 수 있는데, UP_I 선분은 OUT-IN 교차점에서 IN-OUT 교차점으로 진행하며, UP_O 선분은 IN-OUT 교차점에서 OUT-IN 교차점으로 진행한다 (그림 7 (d)).

본 논문에서는 ANY 선분의 개수를 \#(ANY)로 표기한다. 윤곽선 위의 선분들은 다음의 성질들을 만족시킨다. 이 성질들의 증명은 부록 A에서 제시되어 있다.

성질 1

FOR 선분과 BACK 선분의 사이에는 홀수개의 UP선분이나 DOWN선분들이 존재한다.

성질 2

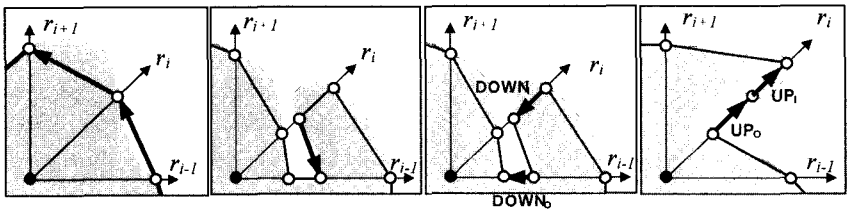
n_r = #(FOR) - #(BACK). 즉, 한 윤곽선 위의 #(FOR)의 가장 작은 값은 n_r이다.

성질 3

만약 $R_0, \ldots, R_{n-1}$로 표시된 새로운 n개의 광선을 r_i와 r_{i+1}이라는 두 광선 사이에 방사할 경우, r_i와 r_{i+1} 위에 있는 DOWN 선분과 UP 선분들은 $2t + 1$개의 선분들로 세분화된다. 이 때에, t는 $t \leq n$을 만족시켜야 한다. 세분화된 선분들은 다음과 같다:

i. r_i 위에 있는 하나의 $DOWN_I$ 선분은 t개의 FOR 선분들과 t개의 BACK 선분들로 둘러싸여 있으며 R_{t-1} 광선 위에 놓인 DOWNI 선분으로 세분화된다 (부록 A의 그림 A-3(a)).

ii. r_i 위에 있는 하나의 DOWNO 선분은 t개의 BACK 선분들과 t개의 FOR 선분들로 둘러싸여 있으며 r_{i-1} 위에 놓인 $DOWN_O$ 선분으로 세분화된다(부록 A의 그림 A-3(b)).



(a) **FOR** 선분 (굵은 화살표) (b) **BACK**edge (굵은 화살표) (c) **DOWN** 선분 (굵은 화살표) (d) **UP** 선분 (굵은 화살표)

그림 7 선분의 유형들

iii. r_{i-1} 위에 있는 하나의 UP_I 선분은 t 개의 BACK 선분들과 t 개의 FOR 선분들로 둘러싸여 있으며 r_i 위에 놓인 UPI 선분으로 세분화된다(부록 A의 그림 A-3(c)).

iv. r_i 위에 있는 하나의 UPO 선분은 t 개의 FOR 선분들과 t 개의 BACK 선분들로 둘러싸여 있으며 R_{t-1} 위에 놓인 UPO 선분으로 세분화된다(부록 A의 그림 A-3(d)).

성질 4

하나의 광선에 의해서 결정된 DOWN 선분들의 집합은 다음의 성질을 만족시킨다.

i. #(DOWN)는 짝수이다.

ii. 같은 집합에서 #($DOWN_I$)와 #($DOWN_O$)은 같다.

iii. 만약 DOWN 선분들보다 짝수 개의 UP 선분들이 선행한다면 한 광선 위에 놓인 DOWN 선분들 중에서 첫 번째 선분은 $DOWN_I$ 선분이다. 그렇지 않을 경우, DOWN 선분들 중에서 첫 번째 선분은 $DOWN_O$ 선분이다.

위의 성질들 중에서 (iii.)을 제외한 다른 성질들은 모두 UP 선분들에 적용된다. UP 선분들을 위해서 개정된 (iii′)은 다음과 같다.

iii′. 만약 한 광선 위에서 짝수개의 DOWN 선분이 UP 선분을 선행한다면, 그 광선 위에 놓인 UP 선분들 중에서 첫 번째는 UP_O이다. 그렇지 않을 경우, 첫 번째 UP 선분은 UP_I이다.

3.3 Context-free 문법을 이용한 윤곽선의 표현

만약 윤곽선 위의 선분들을 정의 1에서 정의된 선분의 분류를 이용해서 표현한다면, 윤곽선은 f, b, d_i, d_o, u_i, u_o의 여섯 문자로 구성된 문자열로 표현된다. 여기서 각 문자들은 각각 FOR, BACK, $DOWN_I$, $DOWN_O$, UP_I, UP_O 선분들을 나타낸다.

정리 1

윤곽선을 추출하기 위해서 n_r 개의 광선을 방사했다고 가정하자. {f, b, d_i, d_o, u_i, u_o}의 여섯 문자들을 이용해서 문자열로 표현된 윤곽선은 다음의 context-free 문법인 $G = (V_N, V_T, P, S)$에 의해서 인식된다.

V_N = {F, B, U, U_R, U_I, U_O, D, D_R, D_I, D_O},

V_T = {f, b, d_i, d_o, u_i, u_o}

P = {
P_1: S --> F ... F (F의 개수는 n_r개임)
P_2: F --> $D\ F\ U$ | $D\ F$ | $F\ U$ | f
P_3: B --> b
P_4: D --> $D\ D$ | $D\ U$ | $U\ D$ | $D_I\ D_O$
P_5: D_I --> $F\ D_I\ B$ | $D_I\ U_R$ | d_i
P_6: D_O --> $B\ D_O\ F$ | $D_O\ U$ | d_o
P_7: U --> $U\ U$ | $D\ U$ | $U\ D$ | $U_o\ U_i$
P_8: U_I --> $B\ U_I\ F$ | $U_I\ D$ | u_i
P_9: U_O --> $F\ U_O\ B$ | $U_O\ D_R$ | u_o
P_{10}: D_R --> $D_O\ D_I$
P_{11}: U_R --> $U_I\ U_O$ }
S = { S }

여기서 V_N, V_T, P, S는 각각 비종료 심벌(non-terminal symbol), 종료 심벌(terminal symbol), 생성 규칙(production rule), 그리고 시작 심벌(start symbol)을 나타낸다.

정리 1의 증명은 부록 A에 제시되어 있다. 위의 심벌들 중에서 같은 형의 선분을 나타내는 심벌들을 동형 심벌(homogeneous symbol) 이라고 칭한다. 따라서 D, D_R, D_I, D_O, d_i, d_o는 DOWN 선분을 나타내는 동형 심벌이며, U, U_R, U_I, U_O, u_i, u_o는 UP 선분을 나타내는 동형 심벌이다.

4. 다각형의 생성

이 장의 목표는 두 개의 이웃하는 윤곽선을 연결하는 다각형을 생성하는 타일링 과정을 기술하는 것이다. 타일링 과정은 두 윤곽선들 사이에서 서로 대응되는 선분 및 꼭지점을 찾는 과정에 그 초점을 맞추고 있다. 기존의 타일링 방법에서는 이러한 대응되는 쌍을 찾기 위해서 기하학적인 계산을 요구하여 왔다. 본 연구에서 제안하는 타일링 방법의 중요한 제안점은 제 3.2 절에서 정의한 선분의 유형을 이용해서 대응되는 꼭지점과 선분을 찾는다는 것이다. 이를 위해서 정의 1에 의해서 윤곽선을 {f, b, u_i, u_o, d_i, d_o}의 여섯 문자들로 구성된 문자열로 표현한다. 이 문자열은 context-free 문법에 의해서 인식되기 때문에, 두 윤곽선들 사이에서 기하학적으로 일치하는 꼭지점과 선분들은 두 문자열의 유도 트리를 동시에 탐색하면서 찾을 수 있다. 이러한 방법은 모호한 경우(ambiguous case)를 해결하기 위한 경우를 제외하면, 기하학적인 계산이 없이 일치하는 꼭지점과 선분을 찾을 수 있다는 장점을 제공한다. 모호한 경우들과 그에 대한 해결 방안에 대해서는 4.4절에서 설명한다.

4.1 대응

윤곽선 위의 한 선분에 대해서 이웃하는 윤곽선 위의 일치되는 선분을 그 선분의 "*대응선분*"이라고 표현하고, 일치되는 꼭지점을 그 선분의 "*대응점*"이라고 표현한다. 그림 8은 두 윤곽선들에 대한 대응선분 및 대응점을 보여주고, 이에 따른 타일링의 예를 제시한다. 다음의 선분들을 서로 대응선분이다.

- 각 윤곽선에서 동일한 쌍의 광선들에 의해서 결정된 두 개의 FOR 선분(그림 8에서의 (e^k_0, e^{k+1}_0), (e^k_3, e^{k+1}_1), (e^k_6, e^{k+1}_4), (e^k_7, e^{k+1}_7)은 이러한 경우의 대응선분이다.)
- 각 윤곽선에서 동일한 광선에 의해서 결정된 UP 또

는 DOWN 선분들(그림 8의 (e^k_4, e^{k+1}_2), (e^k_5, e^{k+1}_3)은 이러한 경우의 대응 선분이다.)

- 각 윤곽선에서 동일한 광선에 의해서 결정된 UP 또는 DOWN 선분들 (그림 8의 (e^k_4, e^{k+1}_2), (e^k_5, e^{k+1}_3)은 이러한 경우의 대응 선분이다.)

각 윤곽선의 선분들에 대해서 대응선분을 찾은 다음, 그 대응선분이 결정되지 않은 선분들을 대상으로 대응점을 찾는 과정을 수행한다. {e^k_x, e^k_1, ..., e^k_a, e^k_y}가 k 번째 윤곽선에서의 연속적인 선분들로 e^k_x와 e^k_y가 각각 $(k+1)$ 번째 윤곽선 위의 선분인 e^{k+1}_x와 e^{k+1}_y를 대응선분으로 가지고 있고, e^k_1, ..., e^k_a는 그 대응선분을 가지고 있지 않다고 가정하자. 이러한 경우, $(k+1)$ 번째 윤곽선에서는 그 대응선분을 갖지 않은 e^{k+1}_1, ..., e^{k+1}_β의 선분들이 e^{k+1}_x와 e^{k+1}_y의 사이에 존재한다고 가정할 수 있다. e^k_1, ..., e^k_a와 e^{k+1}_1, ..., e^{k+1}_β는 서로 대응선분을 가지지 않기 때문에, 이 선분들에 대해서는 대응점을 찾아야 한다. 본 연구에서는 e^{k+1}_x의 끝점을 e^k_1, ..., e^k_a의 대응점으로 설정하고, e^k_x의 시작점을 e^{k+1}_1, ..., e^{k+1}_β의 대응점으로 설정한다. 그림 8의 예에서는 e^{k+1}_1의 시작점을 e^k_1과 e^k_2의 대응점으로 설정하고 있으며, e^k_6의 끝점을 e^{k+1}_5과 e^{k+1}_6의 대응점으로 설정하고 있다. 그림 8의 (b)에는 대응선분들로부터 생성된 다각형들이 제시되어 있고, (c)에는 대응점들로부터 생성된 다각형들이 제시되어 있다.

4.2 유도 트리의 정의

유도 트리는 루트 노드, 1-단계 노드, 내부 노드, 잎새 노드의 네 가지 유형의 노드들로 구성된다. 내부 노드는 G의 비종료 심벌들인 {F, B, D, D_I, D_O, D_R, U, U_I, U_O, U_R} 중의 하나를 저장하며, 잎새 노드는 G의 종료 심벌들인 {f, b, d_i, d_o, u_i, u_o} 중의 하나를 저장한다. 여기서 V라는 심벌을 저장하는 노드를 V-노드로 표기한다. 각각의 노드들은 다음과 같이 정의된다.

- 루트 노드(Root node)

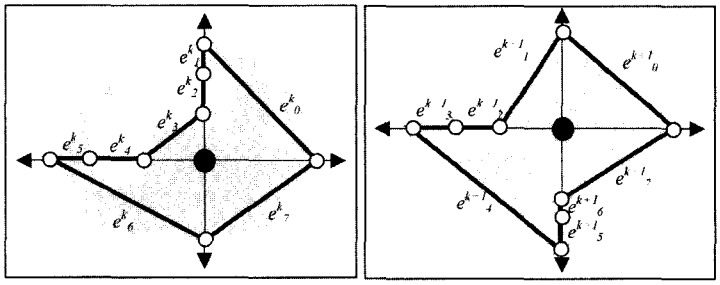
  루트 노드는 G의 시작 심벌인 S를 저장한다. 이 노드는 n_r 개의 1-단계 노드들을 자식 노드로 갖는데, n_r은 처음에 방사된 광선의 수를 나타낸다.

- 1-단계 노드(Level-1 node)

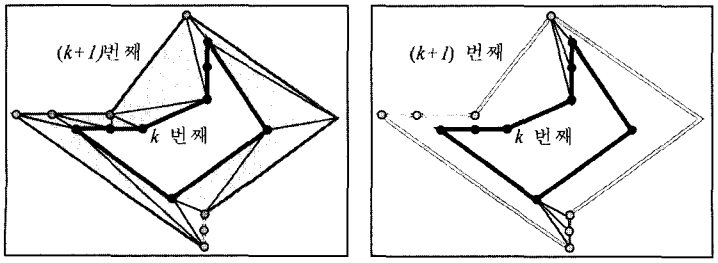
  i 번째의 1-단계 노드는 내부 노드의 일종으로 루트 노드의 자식 노드이며, 비종료 심벌 F와 r_{i-1}과 r_i에 의해서 결정된 FOR 선분을 저장하며, 비종료 심벌 D를 저장하는 좌측 자식 노드와 비종료 심벌 U를 저장하는 우측 자식 노드를 가지고 있다. 예를 들어, 만약 어떤 FOR 선분에 대해서 {F --> D F}를 적용했다면, 그 1-단계 노드는 D를 저장하는 좌측 자식 노드를 가지게 된다.

- 내부 노드(Internal node)

  1-단계 아래의 내부 노드들은 비종료 심벌을 저장한다. 이 노드들의 자식 노드들은 저장하고 있는 심벌에 대해서 적용된 생성 규칙의 우측 항으로부터 결정된다. 예를 들어, {D --> D U}를 적용했다면, D를 포



(a) k 번째 윤곽선(왼쪽) 과 (k+1) 번째 윤곽선(오른쪽)



(b) 대응선분에 의해서 생성된 다각형들 (c) 대응점에 의해서 생성된 다각형들

그림 8 대응선분과 대응점을 이용한 타일링의 예

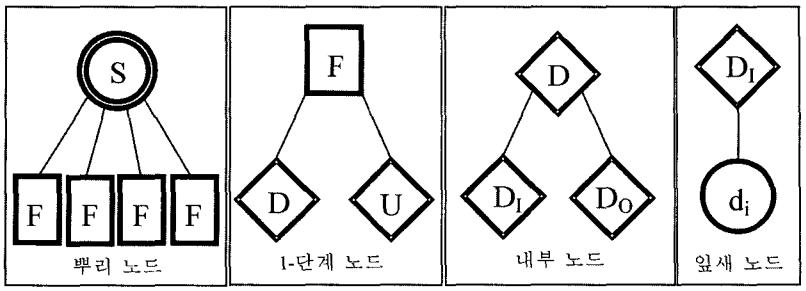


그림 9 유도 트리의 노드의 유형

함하고 있는 내부 노드는 D를 저장하는 좌측 자식 노드와 U를 저장하는 우측 자식 노드를 갖게 된다.

- 잎새 노드(Leaf node)

잎새 노드는 여섯 개의 종료 심벌중의 하나와 그에 대응되는 윤곽선의 선분을 저장한다.

그림 9는 각 노드들과 그 자식 노드들을 보여주고 있다. 이 그림에서 이중 원으로 표시된 것은 루트 노드이며, 사각형은 1-단계 노드, 마름모형은 내부 노드, 그리고 단일 원은 잎새 노드를 각각 나타낸다. 윤곽선의 선분에 대한 정보들은 1-단계 노드와 잎새 노드에 저장되어 있다.

4.3 유도 트리의 생성

본 연구에서는 context-free 문법의 파싱 알고리즘을 적용해서 윤곽선에 대한 유도 트리를 생성한다. 정확한 타일링 과정을 위하여 하나의 윤곽선에 대해서 좌측 유도 트리와 우측 유도 트리라는 두 개의 유도 트리가 생성된다. k번째 윤곽선에 대한 좌측 유도 트리는 (k-1) 번째 윤곽선을 고려해서 생성되며, 우측 유도 트리는 (k+1) 번째 윤곽선을 고려해서 생성된다. 이 생성 과정에서, 각 문자열을 첫 번째 문자로부터 읽어가면서 상향식 (bottom-up) 방법으로 유도 트리를 생성한다. 유도 트리를 생성하는 과정은 다음과 같다.

1. 문자열의 각 문자들에 대해서 G의 생성 규칙을 적용해서 비종료 심벌로 변환한다. 예를 들어, f에 대해서는 생성 규칙 P_2 (F --> f)를 적용해서 F로 변환한다. 이 과정에서 종료 심벌 (문자열의 문자)을 잎새 노드로 하고, 변환된 비종료 심벌을 그 부모 노드로 하는 유도 트리의 일부분이 생성된다. 그리고, 그 문자에 해당되는 선분에 대한 정보는 잎새 노드에 저장된다.
2. 위의 과정에 의해서 생성된 비종료 심벌을 처리하기 위해서, 이 비종료 심벌들을 우변에 갖는 생성 규칙을 적용해서 그 생성 규칙의 좌변에 있는 비종료 심벌을 저장하는 내부 노드를 생성한다. 그리고, 이 노드의 자식 노드로 문자열의 비종료 심벌들을 저장하고 있는 내부 노드들이 설정된다. 예를 들어, {... F D_I B ...}에 대해서 생성 규칙 P_5가 적용된다면, 좌변의 D_I를 저장하는 새로운 내부 노드가 생성되고, 그 노드의 자식 노드로는 F와 D_I와 B를 저장하는 내부 노드들이 설정된다. 여기서, F와 D_I와 B를 저장하는 내부 노드들은 이전 단계에 의해서 이미 생성되어 있다.
3. 문자열에서 f를 만나게 될 경우, 이 f와 그에 해당되는 FOR 선분을 저장하는 1-단계 노드를 생성한다. 이 경우에 발생하는 모호한 경우에 대하서는 5.4 절에서 언급되어 있다.
4. 마지막으로, G의 시작 심벌인 S를 저장하는 루트 노드를 생성한다. 이 루트 노드는 FOR 선분을 저장하는 n_r개의 1-단계 자식 노드들을 가지고 있다.

4.4 모호한 경우의 해결

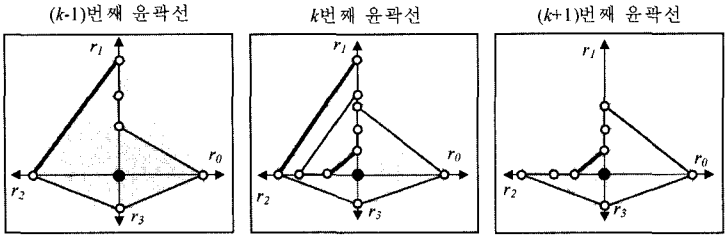
윤곽선을 나타내는 문자열에 대한 유도 트리를 생성하는 과정에서는 다음과 같은 모호한 경우들이 발생할 수 있다.

모호한 경우 1.

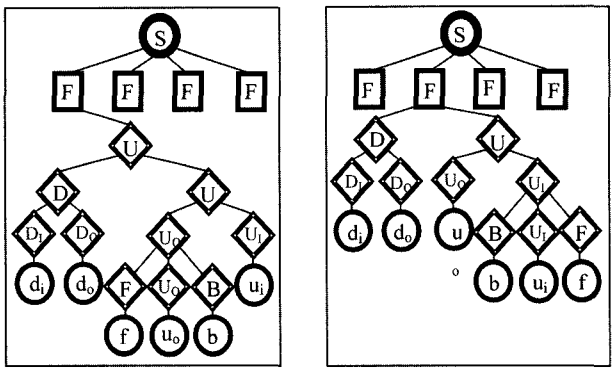
한 쌍의 광선에 의해서 두 개 이상의 FOR 선분들이 결정될 경우, 어떤 FOR 선분을 1-단계 노드에 저장할 것인가를 판단하는 것은 모호한 경우이다. 이 모호한 경우는 타일링 과정을 수행할 이웃의 윤곽선과의 관계를 고려해서 판단하는 과정을 통해서 해결한다. 그림 10 (a)에서 k번째 윤곽선 위에 있는 두 개의 FOR 선분 (중앙의 윤곽선에서의 두꺼운 선)들 중에서 바깥쪽 선분은 $(k-1)$ 번째 윤곽선 위의 FOR 선분 (왼쪽 윤곽선의 두꺼운 선)과 대응되며, 안쪽 선분은 $(k+1)$ 번째 윤곽선 위의 FOR 선분(오른쪽 윤곽선의 두꺼운 선)과 대응되는 것이 적당하다(그림 10(b)).

모호한 경우 2.

하나의 광선에 의해서 여러 개의 동형 선분들이 발생할 경우, 어떤 선분을 선택할 것인지를 판단하는 것은 모호한 경우이다. 그림 11에서 k 번째 윤곽선에서 r_1은 두 개의 UP_O 선분과 UP_I 선분을 생성한다 (r_1위의 두



(a) **FOR**-형 선분에 의한 모호한 경우



좌측유도 트리 우측 유도 트리

(b) 이웃 윤곽선을 고려한 두 개의 유도 트리

그림 10 FOR 선분에 의한 모호한 경우와 그에 따른 두 개의 유도 트리

꺼운 선들). 그리고 k 번째 윤곽선의 이웃 윤곽선들인 $(k-1)$ 번째 윤곽선과 $(k+1)$ 번째 윤곽선에서는 r_1에 의해서 하나의 UP_O 선분과 UP_I 선분이 생성된다. 여기서 이 선분들 사이의 대응 선분을 결정하는 과정에서 발생하는 모호한 상황은 각 선분들의 위치를 계산해서 가장 위치가 가까운 선분들을 대응 선분으로 결정함으로써 해결할 수 있다.

모호한 경우 3.

G의 생성 규칙들 중에는 다른 비종료 심벌에 대해서 동일한 우항을 유도하는 생성 규칙들이 존재하기 때문에 모호한 경우가 발생한다. 선분들의 기하학적인 값을 이용해서 모호한 경우를 해결했던 위의 두 경우와는 달리, 이 경우에는 이웃한 윤곽선에서의 선분의 순서를 고려해서 모호한 경우를 해결한다. 예를 들어, 한 윤곽선 위의 선분들이 $u_o\ u_i\ d_i\ d_o$이고, 이웃하는 윤곽선에서의 해당되는 선분들이 $u_o\ u_i$라면, {$u_o\ u_i\ d_i\ d_o$ --> $U_O\ U_I\ D_I\ D_O$ --> $U\ D$ --> U}가 적용되며, 이웃하는 윤곽선에서의 해당되는 선분들이 $d_i\ d_o$라면, {$u_o\ u_i\ d_i\ d_o$ --> $U_O\ U_I\ D_I\ D_O$ --> $U\ D$ --> D}가 적용된다.

4.5 다각형의 생성

두 윤곽선들 사이에 다각형들을 생성하기 위해서는 두 윤곽선의 유도 트리를 동시에 탐색하면서 윤곽선들 위의 대응점과 대응 선분을 찾아야 한다. 이 과정에서 생성된 다각형 한 윤곽선 위의 선분과 다른 윤곽선 위의 꼭지점으로 결정된 삼각형이다. 그러므로 한 쌍의 대응선분들은 두 개의 삼각형을 생성하며, 대응점은 하나의 삼각형을 생성한다. 다각형을 생성하는 과정은 다음과 같다.

1. 각 유도트리의 루트 노드에서 1-단계 노드들을 왼쪽에서 오른쪽으로 동시에 방문한다.
2. 두 트리의 i 번째의 1-단계 노드에서 이 노드들에 저장된 FOR 선분들은 서로의 대응 선분이 된다. 이 1-단계 노드들에서 같은 심벌을 저장하는 자식 노드들을 동시에 방문한다(그림 12(a)). 그림 12(b)에서는 우측 1-단계 노드가 좌측 자식 노드를 가지고 있지 않기 때문에 좌측 1-단계 노드의 좌측 자식 노드는 방문되지 않는다.
3. 1-단계 아래의 내부 노드들에서는 동일하거나 동형의 심벌을 저장하고 있는 자식 노드들을 동시에 방문한다. G의 생성 규칙에 따르면, 비종료 심벌로부터 유도된 심벌들 중에는 적어도 하나의 동일한 심벌이나 ({D --> $D\ U$}) 동형의 심벌 ({D --> $D_I\ D_O$})

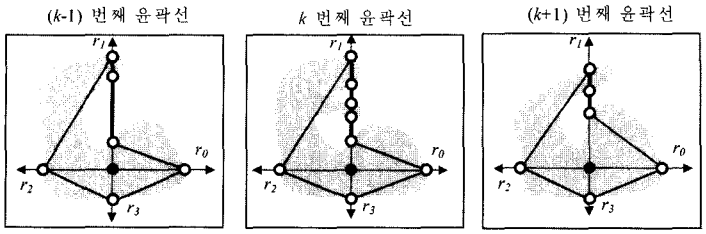
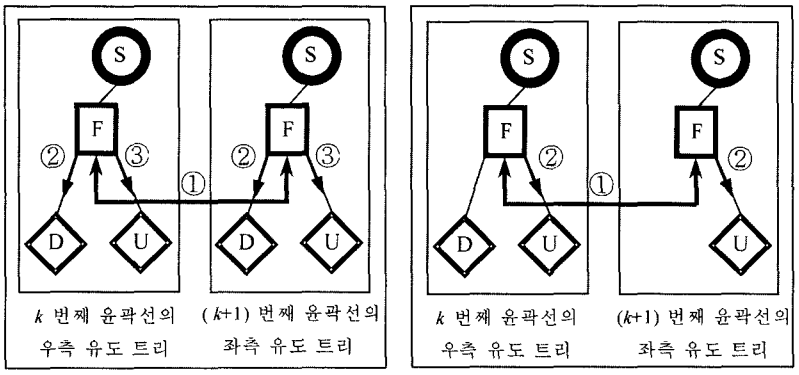


그림 11 동일한 형의 선분에 의한 모호한 경우



(a) 자식 노드들을 모두 방문하는 경우 (b) 자식 노드를 일부만 방문하는 경우

그림 12 1-단계 노드를 방문한 경우

이 포함되어 있기 때문에, 내부 노드에는 동일한 심벌이거나 동형의 심벌을 저장하고 있는 자식 노드가 적어도 하나는 존재한다. 따라서 내부 노드에서 발생할 수 있는 경우는 다음과 같이 정리할 수 있다.

A. 자식 노드들이 일치하는 경우(D --> D U, D --> D U)
   그 자식 노드들을 동시에 방문한다(그림 13(a)).

B. 동일한 자식 노드가 있는 경우(D --> D U, D --> D D)
   동일한 심벌을 저장하고 있는 자식 노드들만을 방문한다. 만약, 어떤 자식 노드를 방문할지가 모호할 경우에는 이를 해결하기 위하여 5.4절에서 제시된 두 번째 모호한 경우에 대한 방법을 적용한다(그림 13(b))

C. 동형의 자식 노드만 있는 경우(D --> D U, D --> D_I D_O)

동일한 심벌을 저장하고 있는 자식 노드를 방문한 다음, 이 자식 노드와 다른 트리의 노드를 비교한다(그림 13(c)).

4. 잎새 노드들에서는 그 잎새 노드에 저장된 선분들이 서로의 대응선분이 된다. 따라서 그림 13(d)에서와 같이 잎새 노드에서는 두 개의 다각형이 생성된다.

5. 방문되지 않은 노드들(그림 13에서 회색으로 표시된 노드들)의 하부 트리에 저장된 선분들은 대응선분 대신 대응점을 갖는다. 대응점을 찾는 방법은 4.1절에 제시되어 있다.

유도 트리를 방문하면서 다각형을 생성하는 과정의 예는 부록 A에 제시되어 있다.

5. 상세화

타일링 과정을 통해서 생성된 다각형 곡면은 그 곡면을 구성하는 윤곽선을 상세화함으로써 상세화된다.

5.1 윤곽선의 상세화

윤곽선을 상세화하기 위해서는 기존의 윤곽선을 추출하기 위해서 방사되었던 광선들 사이에 새로운 광선들을 방사해서 새로운 교차점들을 생성하고, 이를 윤곽선에 추가한다. 새로이 추가되는 교차점들은 다음과 같은 과정을 통해서 기존의 윤곽선의 선분들을 상세화함으로써 기존의 윤곽선을 상세화한다.

1. Context-free 문법의 갱신

하나의 FOR 선분이나 BACK 선분은 두 개의 같은 형의 선분들로 상세화된다. 이 상세화된 FOR 선분과 BACK 선분들을 표현하기 위하여 G에 대해서 다음과 같이 갱신된 생성 규칙인 P_2'와 P_3'를 추가함으로써 새로운 G'를 생성한다.

P_2': F --> F F | D F U | D F | F U | f

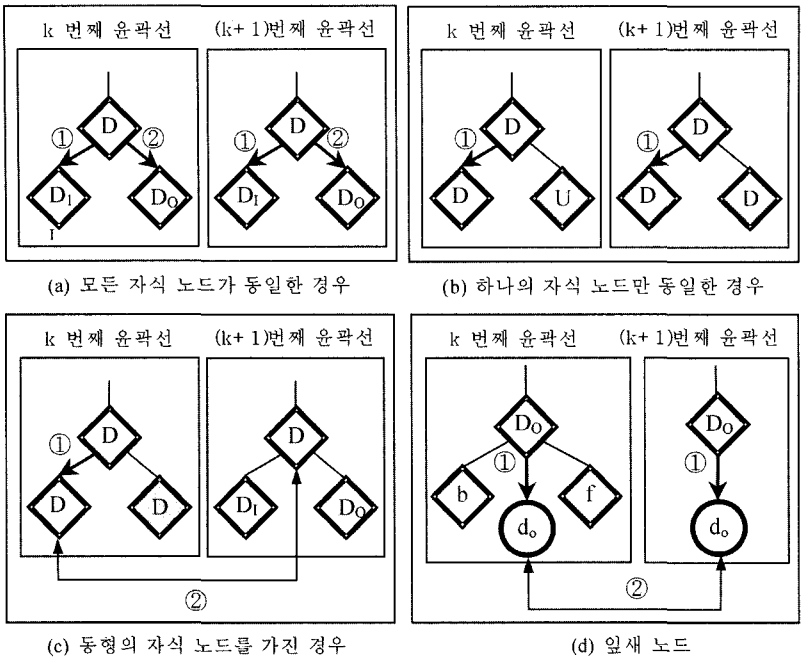


그림 13 내부 노드를 방문한 경우

P_3': B --> $B\ B \mid b$

DOWN 선분이나 UP 선분은 성질 3에 따라서 세 개의 선분으로 상세화된다.

2. 문자열의 변환

기존의 윤곽선을 나타내는 문자열에서 상세화된 선분을 나타내는 종료 심벌을 그에 해당되는 비종료 심벌로 교체한다. 즉, f, b, d_i, d_o, u_i, u_o들은 각각 F, B, D_I, D_O, U_I, U_O들로 교체된다. 그리고, 갱신된 문자열에서의 비종료 심벌들은 G'에 따라서 종료 심벌들로 유도된다.

3. 유도 트리의 갱신

종료 심벌을 저장하고 있는 유도 트리의 잎새 노드들이 제거된다. 잎새 노드들의 부모 노드들은 잎새 노드가 저장하고 있는 종료 심볼로 유도되는 비종료 심볼을 저장하고 있다. 그리고, 유도 트리는 갱신된 문자열의 비종료 심벌들의 유도 과정에 따라서 갱신된다. 예를 들어, 만약 {F -->F F}를 적용했다면, F를 저장하는 두 개의 내부 노드가 우항의 F를 저장하는 노드의의 자식 노드로 생성된다. 이 과정에서 부모 노드의 자식 노드들은 자식 노드에게 상속된다.

그림 14에는 FOR 선분에 대한 두 가지의 상세화가 예시되어 있다. 그림 14 (b)에는 하나의 FOR 선분이 두 개의 FOR 선분으로 세분화되며, 그림 14(c)에는 하나의 FOR 선분이 두 개의 DOWN 선분으로 둘러싸인 FOR 선분으로 세분화된다 (이 세분화의 경우는 G에 의해서 처리가 가능하다). 그리고 그림 15에는 UP 선분에 대한 두 가지의 상세화가 예시되어 있다. 그림 15 (b)에는 하나의 UP_I 선분이 BACK, UP_I, FOR 선분들로 상세화되는 예가 제시되어 있으며, 그림 15(c)에는 하나의 UP_O 선분이 FOR, UP_O, BACK 선분으로 상세화되는 예가 제시되어 있다.

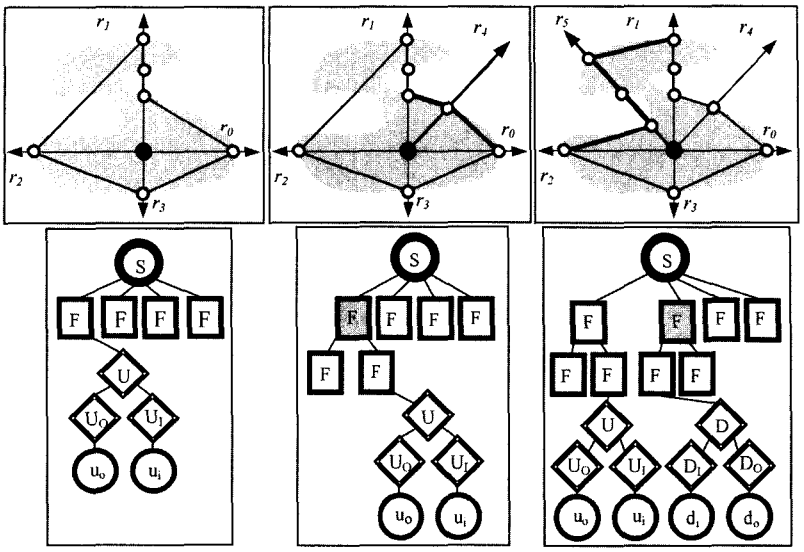
5.2 다각형 곡면의 상세화

5.2.1 윤곽선의 상세화를 통한 곡면의 상세화

윤곽선 위의 선분들을 상세화하는 과정은 그 윤곽선의 유도 트리를 갱신하는 과정을 통해서 구현된다. 이 때에 윤곽선 위의 한 선분은 다각형 곡면의 한 다각형에 속해있기 때문에, 이 선분이 새로운 선분들로 상세화되면, 그 선분을 포함하고 있는 다각형이 여러 개의 다각형으로 상세화된다. 부록 B에는 새로운 광선의 방사에 의한 윤곽선의 상세화 및 이에 따른 다각형 곡면의 상세화가 제시되어 있다.

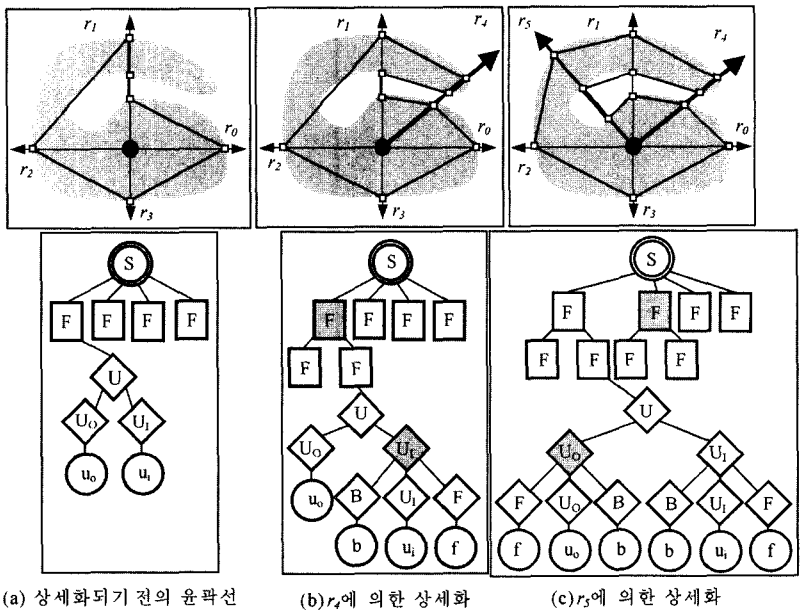
5.2.1 새로운 윤곽선의 추가에 따른 상세화

다각형 곡면을 상세화하기 위하여 새로운 윤곽선을 추가할 경우, 기존에 샘플링된 영상들의 사이에 새로운 영상을 샘플링해야 한다. 만약, k 번째 영상과 $(k+1)$ 번째의 영상들 사이에 새로운 영상을 샘플링한다면, 그 새로운 영상에서의 윤곽선은 가장 낮은 해상도로 추출된다. 이 새로운 윤곽선을 추가함으로써 k 번째 윤곽선과 $(k+1)$ 번째 윤곽선사이에서 생성된 다각형들은 제거되



(a) 상세화되기 전의 윤곽선 (b)r_4에 의한 상세화 (c)r_5에 의한 상세화

그림 14 FOR 선분의 상세화



(a) 상세화되기 전의 윤곽선 (b)r_4에 의한 상세화 (c)r_5에 의한 상세화

그림 15 UP 선분의 상세화

고, 제4장에서 제시된 방법에 따라서 새로운 다각형들이 생성된다.

6. 구현 및 결과

제안된 알고리즘은 800 MHz의 속도를 가진 Pentium III CPU와 256 MB의 주기억 용량을 가진 개인용 컴퓨터에서 Microsoft 사의 Visual C++ 6.0과 OpenGL을 이용해서 구현되었다. 제안된 알고리즘은 팬텀 물체와 목 부위의 경동맥을 복원하기 위해서 적용되었다. 그림 16에는 팬텀 물체를 대한 네 레벨의 복잡도로 복원한 예가 제시되어 있으며, 그림 17에는 목 부위 경동맥을 네 레벨의 복잡도로 복원한 예가 제시되어 있다. 성능과

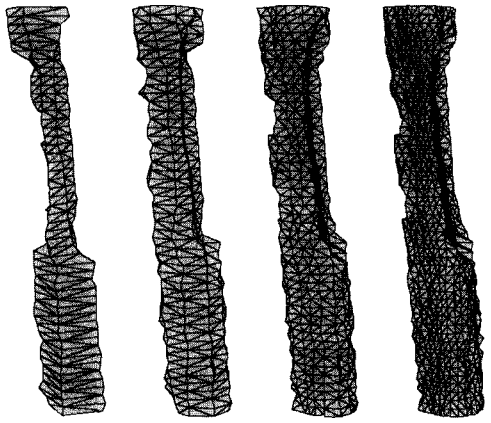


그림 16 네 가지 해상도로 복원된 팬텀 물체

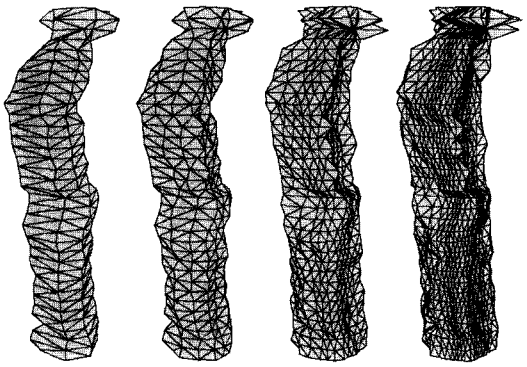


그림 17 네 가지 해상도로 복원된 목 부위 경동맥

표 1

해상도	방법	시간 (Sec)	삼각형의 수	선분의 평균 길이	데이타의 양 (Kbyte)
1	기존 방법	0.340	444	1.202	6.384
	새 방법	0.241	412	1.264	2.684
2	기존 방법	0.761	1,061	0.783	15.126
	새 방법	0.550	867	0.842	6.054
3	기존 방법	1.442	2,424	0.545	24.032
	새 방법	0.992	1,754	0.582	12.652
4	기존 방법	2.904	4,791	0.451	136.986
	새 방법	2.062	3,392	0.465	50.606

품질의 비교를 위해서, 본 연구자들은 toroidal 탐색 방법을 이용하는 기존의 알고리즘[9,10,18]을 구현해서, 계산 시간과 삼각형의 수, 그리고 다각형 곡면의 선분의 평균 길이, 곡면에 대한 데이타의 양 등을 표 1에서와 같이 비교하였다. 여기서 toroidal 탐색 방법은 두 윤곽선들 사이에서 최적의 크기를 가진 삼각형들을 생성하는 방법으로 알려져 있다.

6.1 특징을 보존하는 타일링 방법

본 연구에서 제시된 타일링 방법의 장점 중의 하나는 그 윤곽선의 특징을 보존한다는 것이다. 윤곽선은 물체의 단면이기 때문에, 물체의 기하학적인 특징은 윤곽선의 형태에도 그 영향을 미친다. 그러므로, 타일링 과정에서 윤곽선들의 특징을 추적해서 그를 보존하는 것은 매우 바람직한 일이다. 기존의 타일링 방법들은 윤곽선의 기하학적인 정보에만 의존하기 때문에 윤곽선의 특징을 타일링 과정에 적용하는 것이 어려웠다. 그러나, 본 연구에서 제시하는 방법은 윤곽선을 추출하기 위해서 방사되는 광선을 조절함으로써 윤곽선의 기하학적인 특징을 보존할 수 있다. 그림 18과 그림 19에서 제시된 물체는 X형의 단면을 곡선을 따라서 비틀면서 이동시킴으로써 생성된 물체들이다. 따라서 이 물체는 비틀린 X형의 기둥모양이 된다. 제안된 방법에서는 각 영상들에서 단면이 비틀린 각도만큼 광선을 회전시켜서 방사한다. 그림 19의 물체는 제안된 방법에 의해서 복원된 물체이며, 그림 18의 물체는 기존의 방법에 의해서 복원된 물체이다. 다양한해상도에서, 제안된 방법에 의해서 복원된 물체는 그 외형을 정확하게 보존하는 반면에, 기존의 방법에 의해서 복원된 물체는 그렇지 못하다.

6.2 압축

2차원 영상에서의 윤곽선 위의 꼭지점은 (x, y)와 같이 두 개의 값을 갖는 유클리드 좌표계(Euclidean coordinate)로 표현된다. 여기서 이 유클리드 좌표계를 극 좌표계(Polar coordinate)로 바꾸면, 이 꼭지점은 (r, θ)로 표현되는데, 이 식에서 r은 중심 픽셀로부터의 거리이며, θ는 그 꼭지점을 결정하는 광선의 각도이다. 영상의 대각선의 길이인 d에 대해서 $r \in (0, d)$가 성립하기 때문에, r은 $r' \in (0, 1)$인 r'로 변환될 수 있다. 그리고 i 번째 광선에 대한 θ_i는 $\theta_i = i \Delta\theta + \theta_0$로 표현될 수 있는데, 여기서 $\Delta\theta$는 광선들 사이의 각도이다. 제4장의 정의에 의해서 두 이웃한 꼭지점들 사이의 광

선들에 대한 i의 차는 0 (UP 선분, 또는 DOWN 선분), +1 (FOR 선분), 또는 -1 (BACK 선분)이다. 그러므로, 한 꼭지점은 ar'와 같은 값으로 표현될 수 있다. 여기서 a는 $a \subset [-1,0,1]$을 만족시키며, r'는 $r' = r/d \in (0, 1)$을 만족시킨다. 만약 i 번째 꼭지점이 (θ_i, r_i)로 표현된다면, 그 다음 꼭지점인 $(i+1)$ 번째 꼭지점의 유클리드 좌표인 (x_{i+1}, y_{i+1})는 다음과 같이 계산된다.

$$(x_{i+1}, y_{i+1}) = (r_{i+1}\cos\theta_{i+1}, r_{i+1}\sin\theta_{i+1}),$$
$$where \theta_{i+1} = \theta_i + a_{i+1}\Delta\theta, \text{ and } r_{i+1} = r_{i+1}'d$$

그러므로, 제안된 방법은 두 개의 좌표 값을 갖는 꼭지점을 하나의 실수 값으로 표현한다. 그리고, 두 윤곽선들 사이의 꼭지점들은 하나의 선분을 공유하기 때문에 이 삼각형들은 삼각형 띠(triangular strip)로 표현이 가능하다. 따라서 다각형 곡면은 위의 두 가지의 사실에 의해서 압축될 수 있다. 압축된 데이타의 양은 표 1의 가장 오른쪽의 열에서 비교되어 있다.

7. 결론 및 향후 연구 방향

이 논문에서는 윤곽선들로부터 다각형 곡면을 생성하는 다중해상도적 타일링 방법이 제시되었다. 이 방법의 중요한 특징은 윤곽선들이 개선된 방사 조사법에 의해서 추출된다는 점이다. 이 방법에 의해서 추출된 윤곽선들의 특징은 그 윤곽선을 인식하는 context-free 문법을 정의하는데에 이용된다. 그러므로, 윤곽선들은 그 context-free 문법의 유도 트리들을 탐색하는 과정을 통해서 타일링된다. 이러한 타일링 방법은 타일링 과정에서 요구되는 계산량을 감소시킨다는 장점이 있다. 다중해상도적 표현을 얻기 위해서는 상세화된 윤곽선을 이용해서 다각형 곡면을 상세화하는 점진적인 상세화 방법이 제시되어 있다. 이러한 장점들 이외에도 특징을 보존하는 타일링이나 압축과 같은 장점들은 제6장에 제시되어 있다.

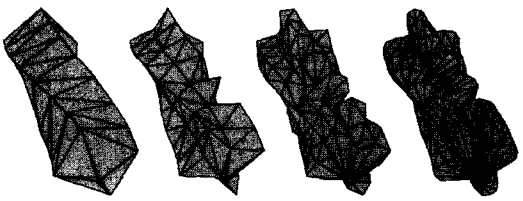


그림 18 기존 방법에 의한 타일링: 특징을 보존하지 못함

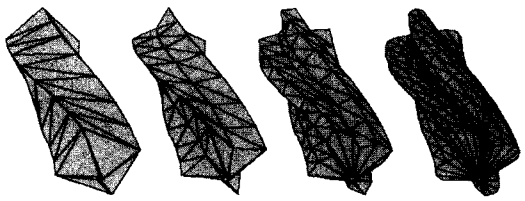


그림 19 제안된 방법에 의한 타일링: 특징을 보존함

본 연구의 향후 연구로는 본 연구에서 취급하지 않았던 복원의 다른 문제, 즉, 분지(branching) 문제를 본 연구에서 제시하는 방법을 이용해서 해결하는 방법을 적용한다. 또한 다른 연구 방향으로는 context-free 문법에 기반을 해서 다각형을 표현하는 방법을 일반화 하여 윤곽선으로부터 생성된 다각형 곡면에 대한 계층적인 표현 방법을 제시하고자 한다.

참 고 문 헌

[1] Klein, R., Schilling, A., and Strasser, W., *"Reconstruction and simplification of surfaces from contours," Proceedings of Pacific Graphics '99*, pp. 198-207, 1999.

[2] Garland, M. and Heckbert, P., *"Surface Simplification using Quadric Error Metrics," Proceedings of SIGGRAPH 97*, pp. 209-216.

[3] Fujimura, K., *"Shape reconstruction from contours using isotopic deformation," Graphical Models and Image Processing*, Vol. 61, No. 3, pp. 127-147, 1999.

[4] Garcia E, Gueret P, Bennett M, Corday E, Zwehl W, Meerbaum S, Corday S, Swan H., and Berman D., *"Real-time computerization of two-dimensional echocardiography,"* Am Heart J 101:783, 1981.

[5] Buda A. J., Delp E. J., Meyer C. R., Jenkins J. M., Smith D. N., Bookstein F. L., and Pitt B., *"Automatic computer processing of digital 2-dimensional echocardiograms," Am J Cardiol* 52:384, 1983.

[6] Meyers, D., Skinner, S., and Sloan, K., *"Surfaces from contours," ACM Transactions on Graphics*, Vol. 11, No. 3, pp. 228-258, 1992.

[7] Pitas, I., *Digital Image Processing Algorithms*, Prentice Hall, 1993.

[8] Lorensen, W. E. and Cline, H. E., *"Marching Cubes: A High Resolution 3D Surface Construction Algorithm," Computer Graphics*, Vol. 21, No. 4, pp. 163 - 169, 1987.

[9] Fuchs, H., Kedem, Z. M., and Uselton, S. P., *"Optimal surface reconstruction from planar contours," Communications of ACM*, Vol. 20, No. 10, pp. 693-702, 1977.

[10] Boissonnat, J. D., *"Shape reconstruction from planar cross sections," Computer Vision, Graphics, and Image Processing*, Vol. 44, pp. 1-29, 1988.

[11] Ekoule, A. B., Peyrin, F. C., and Odet, C., L., *"A triangulation algorithm from arbitrary shaped multiple planar contours," ACM Transactions on Graphics*, Vol. 10, No. 2, pp. 182-199, 1991.

[12] Hoppe, H., *"Progressive Meshes,"* Proceedings of SIGGRAPH 96, pp. 99-108, 1996.

[13] Bajaj, C. L., Coyle, E. J., and Lin, K. N., *"Arbitrary Topology shape reconstruction from planar cross sections," Graphical Models and Image Processing*, Vol. 58, No. 6, pp. 524-543, 1996.

[14] Meyers, D., *"Multiresolution tiling," Computer Graphics Forum*, Vol. 13, No. 5, pp. 325-340, 1994.

[15] Cong, G. and Parvin, B., *"An algebraic solution to surface recovery from cross-sectional contours," Graphical Models and Image Processing*, Vol. 61, No. 4, pp. 222-243, 1999.

[16] Schilling, A. and Klein, R., *"Fast generation of multiresolution surfaces from contours," Proceedings of Eurographics Workshop on Visualization in Scientific Computing*, pp. 35-46, 1998.

[17] Fix, J. D. and Ladner, R. E., *"Multiresolution based refinement to accelerate surface reconstruction from polygons," Computational Geometry*, Vol. 13, No. 1, pp. 49-64, 1999.

[18] Christiansen, H. N. and Sederberg, T. W., *"Conversion of complex contour line definitions into polygonal element mosaics," Proceedings of SIGGRAPH 78*, pp. 187-192, 1978.

부록 A. 증명

성질 1의 증명

FOR 선분과 BACK 선분이 서로 이웃해 있다고 가정하자. 이와 같은 경우는 FOR 선분이 두 개의 IN-OUT 점을 연결하고, BACK 선분이 두 개의 OUT-IN 점을 연결한다는 사실 때문에 모순되기 때문에, 존재하지 않는다. 그리고 UP 선분과 DOWN 선분들은 각각 {IN-OUT, OUT-IN} 점들이나 {OUT-IN, IN-OUT}의 점들을 연결한다. 그러므로 FOR 선분과 BACK 선분들의 사이에는 홀수 개의 UP 선분이나 DOWN 선분이 요구된다. 그림 A-1의 (a)와 (b)에는 홀수 개의 UP 선분이 FOR 선분과 BACK 선분의 사이에 놓여있음을 보여주며, (c)와 (d)에서는 홀수 개의 DOWN 선분이 FOR 선분과 BACK 선분의 사이에 놓여있음을 보여준다. □

성질 2의 증명

이 성질을 증명하기 위하여 수학적 귀납법(mathematical induction)을 적용한다. 먼저, 두 이웃하는 광선들 사이에서는 #(FOR) - #(BACK) = 1임을 증명하고, 이를 n_r 개의 광선에 대해서 확장한다.

[Induction Basis]

먼저, r_i와 r_{i+1} 사이에 두 개의 FOR 선분이 존재한다

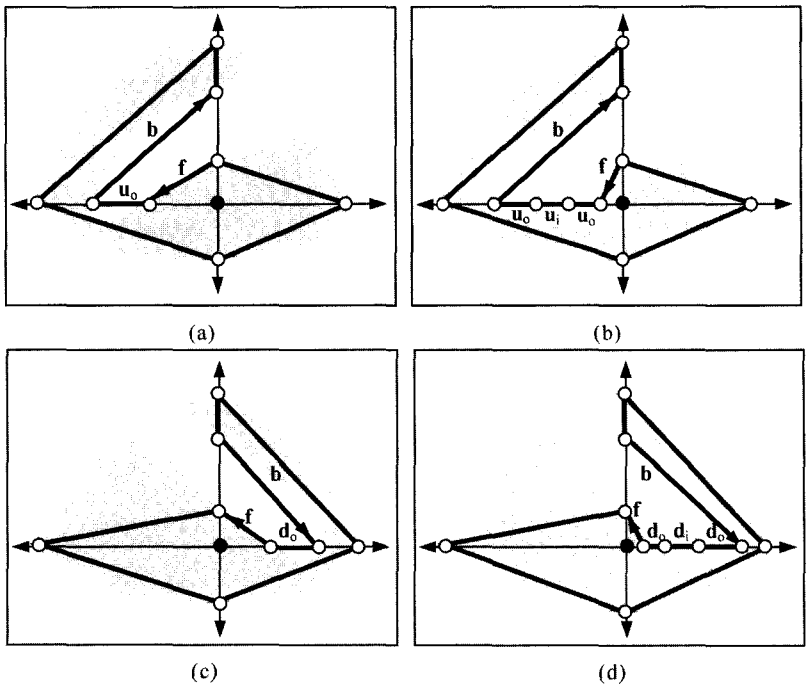


그림 A-1 성질 1의 예: f, b, u_o, u_i, d_o, d_i는 각각 FOR, BACK, UP_O, UP_I, $DOWN_I$, $DOWN_O$ 선분들을 나타낸다.

고 가정하자. 윤곽선은 닫혀진 다각형이기 때문에 한 FOR 선분의 끝점으로부터 다른 FOR 선분의 시작점으로 가는 경로가 존재한다. 여기서 두 FOR 선분들의 끝점은 r_{i+1} 광선 위에 놓여있으며, 시작점은 r_i 광선 위에 놓여있다. 그러므로 한 FOR 선분의 끝점에서 다른 FOR 선분의 시작점으로 가는 경로는 r_{i+1} 광선 위에 놓인 시작점과 r_i 광선 위에 놓인 끝점으로 정의되는 BACK 선분이 존재한다. 그러므로, 두 이웃하는 광선들에 대해서 #(FOR) - #(BACK) = 1이 성립한다.

[Induction Hypothesis]

두 광선 r_i와 r_{i+1} 사이에 m 개의 FOR 선분이 존재할 경우에도, #(FOR) - #(BACK) = 1이 성립한다.

[Induction Hypothesis]

두 광선 사이에 $m+1$ 개의 FOR 선분이 존재한다고 가정하자. Induction Hypothesis에 따르면 m 개의 FOR 선분들에 대해서는 $m-1$ 개의 BACK 선분들이 존재한다. 윤곽선이 닫혀진 다각형이라는 사실로부터, m 번째의 FOR 선분의 끝점과 $m+1$ 번째의 FOR 선분의 시작점을 연결하는 BACK 선분이 존재함을 알 수 있다. 이 BACK 선분을 추가함으로써 두 이웃한 광선들 사이의 $m+1$ 개의 FOR 선분들에 대해서 m 개의 BACK 선분들이 존재한다.

두 이웃한 광선들 사이에서 #(FOR) - #(BACK) = 1이 성립하기 때문에, n_r 개의 광선들에 대해서는 #(FOR) - #(BACK) = n_r이 성립한다. 만약, #(BACK) = 0이면, #(FOR) = n_r이다. n_r = 13인 경우의 예제는 그림 A-2에 제시되어 있다. □

성질 3의 증명

성질 3의 i. 항을 #(FOR)와 #(BACK)을 의미하는 t에 대한 수학적 귀납법을 이용해서 증명한다. 이 i. 번째 항에 대한 증명은 다른 항에 대한 증명으로 확장된다.

[Induction Basis]

t = 1인 경우, R_0로 표시된 새로운 광선은 r_i와 r_{i+1}의 사이에 방사된다고 가정한다. 이 경우에는 r_i 위의 $DOWN_I$ 선분이 세분화된다(그림 A-3(e)의 *(a,d)*). R_0의 첫 번째 교차점(그림 A-3(e)의 c)와 마지막 교차점(그림 A-3(e)의 b)이 경로에 추가된다. 그러므로 다음과 같은 경로가 생성된다: a --> b --> c --> d. 따라서 새로운 광선을 추가함으로써 하나의 $DOWN_I$ 선분은 FOR 선분과 BACK 선분으로 둘러싸인 $DOWN_I$ 선분으로 세분화된다.

[Induction Hypothesis]

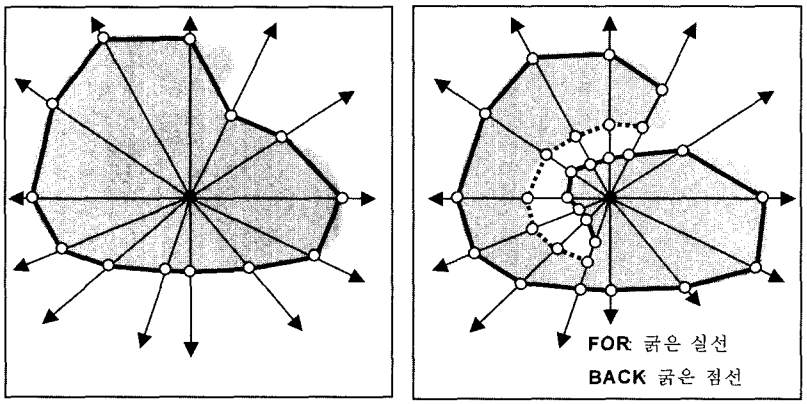
$t = m$이라고 가정하자. 두 광선들 사이에 m 개의 새로운 광선을 삽입한다면, 하나의 $DOWN_I$ 선분은 m 개의 FOR 선분과 m 개의 BACK 선분들로 둘러싸인 $DOWN_I$ 선분으로 세분화된다.

[Induction Step]

$t = m + 1$이라고 가정하자. 두 광선들 사이에 $m + 1$ 개의 새로운 광선을 방사할 경우에, Induction Hypothesis에 의해서 처음의 m 개의 광선으로부터 하나의 $DOWN_I$ 선분은 m 개의 FOR 선분과 m 개의 BACK 선분으로 둘러싸인 $DOWN_I$ 선분으로 세분화된다. 그리고 Induction Basis에 의해서 하나의 $DOWN_I$ 선분은 하나의 FOR 선분과 BACK 선분으로 둘러싸인 $DOWN_I$ 선분으로 세분화된다. 그러므로 $m + 1$ 개의 광선은 하나의 $DOWN_I$ 선분은 $m + 1$ 개의 FOR 선분과 $m + 1$ 개의 BACK 선분으로 둘러싸인 $DOWN_I$ 선분으로 구성된 $2m + 3$ 개의 선분들로 세분화된다. □

성질 4의 증명

i. r_i가 DOWN 선분들을 결정한다고 가정하자. 윤곽



#(FOR) = n_r = 13, #(BACK) = 0 n_r = #(FOR) - #(BACK) = 20 7 = 13

그림 A-2 성질 2의 예

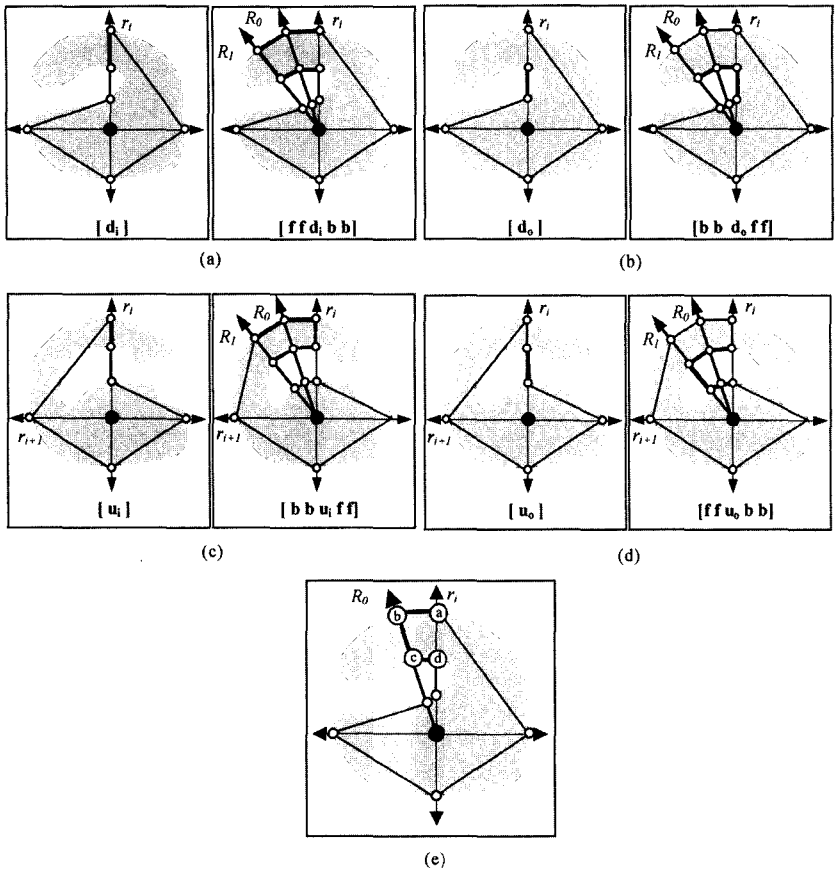


그림 A-3 성질 3의 예: 그림에서 원래의 선분과 세분화된 선분들은 굵은 선으로 표시되어 있다.

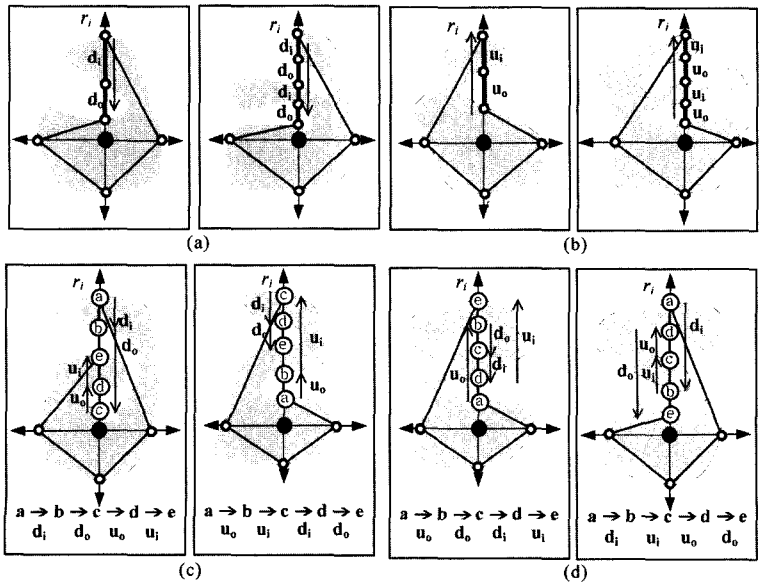


그림 A-4 성질 4의 예

선은 닫혀있기 때문에 이 DOWN 선분들의 집합으로부터 첫 번째 DOWN 선분(r_{i-1}와 r_i를 연결하는 FOR 선분에 연결된)과 마지막 DOWN 형 선분(r_i와 r_{i+1}를 연결하는 FOR 선분에 연결된)을 선택할 수 있다. FOR 선분은 두 개의 IN-OUT 교차점을 연결하기 때문에 첫 번째 DOWN 선분의 시작점과 마지막 DOWN 선분의 끝점은 IN-OUT 교차점이다. 그런데 DOWN 선분은 서로 다른 유형의 교차점을 연결하기 때문에, 홀수개의 DOWN 선분은 두 개의 IN-OUT 교차점들을 연결할 수 없다. 그러므로, DOWN 선분의 수는 짝수개가 된다. 그림 A-4(a)는 DOWN 선분들에 대한 몇 개의 예를 보여주고 있으며(왼쪽 그림에서는 두 개의 선분이며, 오른쪽 그림에서는 네 개의 선분이다), (b)는 UP 선분들에 대한 예를 보여주고 있다.

ii. DOWN 선분은 서로 다른 종류의 교차점을 연결한다. 따라서 DOWN 선분들의 집합은 $DOWN_I$ 선분들과 $DOWN_O$ 선분들의 반복적인 배열로 구성되어 있다. 그리고 DOWN 선분의 집합은 짝수개의 선분들로 구성되어 있기 때문에, $DOWN_I$ 선분의 수와 $DOWN_O$ 선분의 수는 동일하다. 몇 가지의 예제가 그림 A-4(a)와 (b)에 제시되어 있다.

iii. (i.)의 증명에서 r_i의 위에 있는 첫 번째의 DOWN 선분은 r_{i-1}과 r_i에 의해서 결정되는 FOR 선분에 연결되어 있음을 보였다. 그러므로 첫 번째 DOWN 선분의 시작점은 IN-OUT이다. 그리고 DOWN 선분의 방향이 광선의 방향과 반대라는 사실로부터 선분의 끝점은 OUT-IN임을 알 수 있다. 따라서 첫 번째의 DOWN 선분이 IN-OUT 교차점과 OUT-IN 교차점을 연결하기 때문에, 이 선분은 $DOWN_I$ 선분이다(그림 A-4(c)). 같은 광선 위에서 홀수 개의 UP 선분들이 DOWN 선분을 선행하는 경우, 첫 번째 DOWN 선분은 OUT-IN 교차점으로부터 시작된다 (그림 A-4(d)). 그러므로 첫 번째 DOWN 선분은 $DOWN_O$이다. □

정리 1의 증명

윤곽선을 나타내는 문자열은 제3.2절에서 제시된 네 가지의 성질들을 만족시킨다. 따라서 본 절에서는 G는 그 네 가지의 성질들을 표현하고 있음을 증명한다. 여기서 G에 의해서 인식되는 문자열은 그 네 성질들을 만족하는 문자열이 되기 위한 필요 조건이다.

1. 성질 1에 따르면 FOR 선분과 BACK 선분의 사이에는 홀수 개의 UP 선분과 DOWN 선분이 존재한다. 생선 규칙 P_5, P_6, P_8, P_9은 FOR 선분과 BACK 선분들 사이에 홀수 개 UP 선분이나 DOWN 선분이 존재함을 나타낸다.

2. 성질 2에 따르면, n_r 개의 광선에 의해서 추출된 윤곽선은 적어도 n_r 개의 FOR 선분들로 구성된다. 이 성질은 G의 생성 규칙 P_1과 P_2에 의해서 인식된다. P_1은 n_r 개의 F들을 유도해 내는데, 이 각각의 F들은 P_2에 의해서 적어도 하나의 f로 유도된다. 특히, 문자열에서 f의 수에서 b의 수를 뺀 값은 n_r과 일치한다. G는 P_2, P_5, P_6, P_8, P_9의 다섯 생성 규칙에서 f를 유도한다. 이 중에서, P_2에서는 F에 대해서 f만을 유도하고, 다른 네 규칙들은 f와 b를 동시에 유도한다. 그러므로 다른 네 생성 규칙에 의해서 생성된 f의 수와 b의 수는 일치한다. 생성 규칙 P_1에 의해서 S가 n_r 개의 F를 유도하기 때문에, 문자열에서의 모든 f의 수에서 모든 b의 수를 뺀 값은 n_r이 된다.

3. 성질 3에 따르면, 하나의 DOWN 선분은 같은 수의 FOR 선분들과 BACK 선분들에 의해서 둘러 쌓일 수 있다. G에서 P_5는 D_I로부터 d_i를 유도해내며, P_6는 D_O로부터 d_o를 유도해낸다. 이 때에 P_5는 f와 b에 의해서 둘러 쌓인 D_I를 유도한다. 이는 P_5가 같은 수의 f와 b에 의해서 둘러 쌓인 d_i를 유도함을 의미한다. 이와 같이 P_6도 같은 수의 b와 f에 의해서 둘러 쌓인 d_o를 유도한다. P_5와 P_6를 P_8과 P_9으로 교체할 경우, 이와 같은 과정이 UP 선분들에 대해서도 적용된다.

4. 성질 4에 따르면, 하나의 광선에 의해서 결정된 DOWN 선분들은 다음의 세 성질을 가지고 있다.

i. 생성 규칙 P_4에 따르면 D는 D_I D_O를 파생한다. P_5에 의해서 하나의 D_I로부터 하나의 d_i가 파생되고, P_6에 의해서 하나의 D_O로부터 하나의 d_o가 파생되기 때문에 모든 DOWN 선분들의 수는 짝수이다. 이와 같은 과정은 UP 선분들에 대해서도 성립한다.

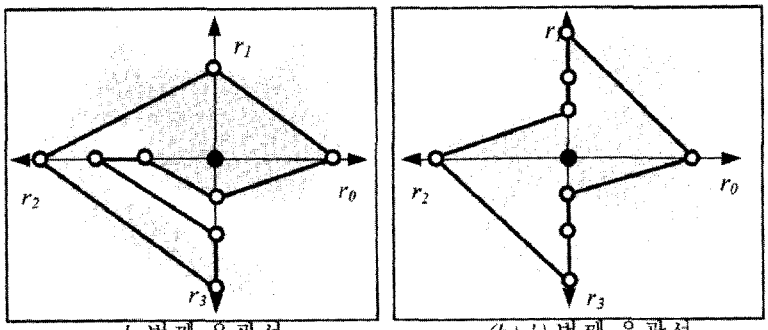
ii. D는 항상 같은 수의 d_i와 d_o를 파생하기 때문에 문자열에서의 d_i의 수와 d_o의 수는 동일하다. 이는 UP 선분들에 대해서도 적용된다.

iii. 생성 규칙 P_4에 의해서 D는 D_I D_O를 파생하기 때문에 문자열에서 d_i는 d_o에 선행한다. UP 선분의 경우에는 P_7에 의해서 U가 U_o U_i를 파생하기 때문에 문자열에서는 u_o가 u_i보다 선행하게 된다.

위의 네 증명들로부터 제3.2절에서 제시한 네 개의 성질을 만족시키는 윤곽선을 나타내는 문자열은 G에 의해서 인식된다. □

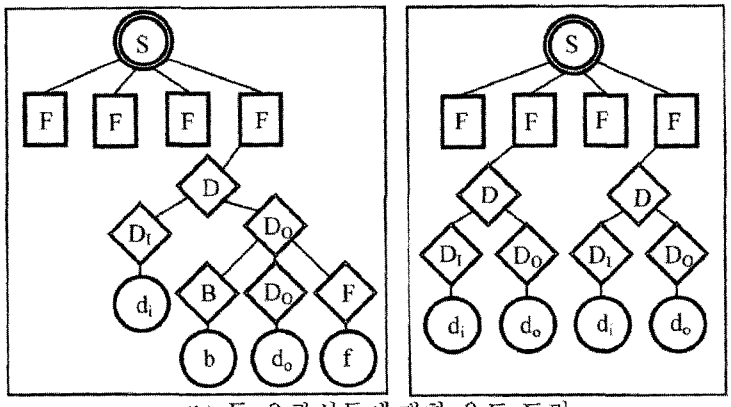
부록 B. 타일링의 예

그림 B-1에는 두 개의 윤곽선과 그 윤곽선들에 대한 유도 트리가 제시되어 있다. 그리고, 이 유도 트리들을 검색하면서 다각형 곡면이 생성되는 예는 그림 B-2와 B-3에 제시되어 있다. 각 그림들에서 두 개 트리에서 방문되고 있는 노드들이 회색으로 표시되어 있으며, 그 방문에 의해서 생성된 다각형들이 제시되어 있다.



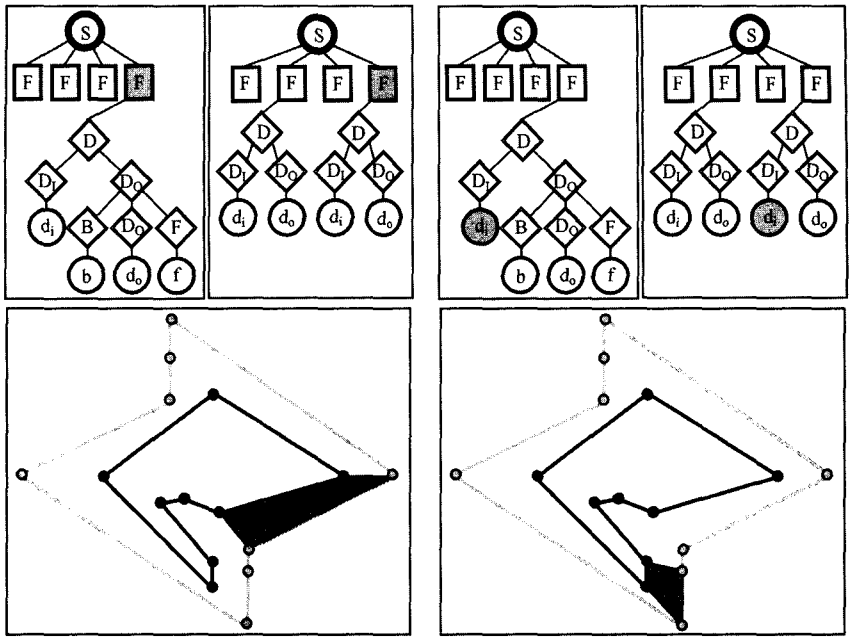
k 번째 윤곽선 *(k+1)* 번째 윤곽선

(a) 두 개의 윤곽선들

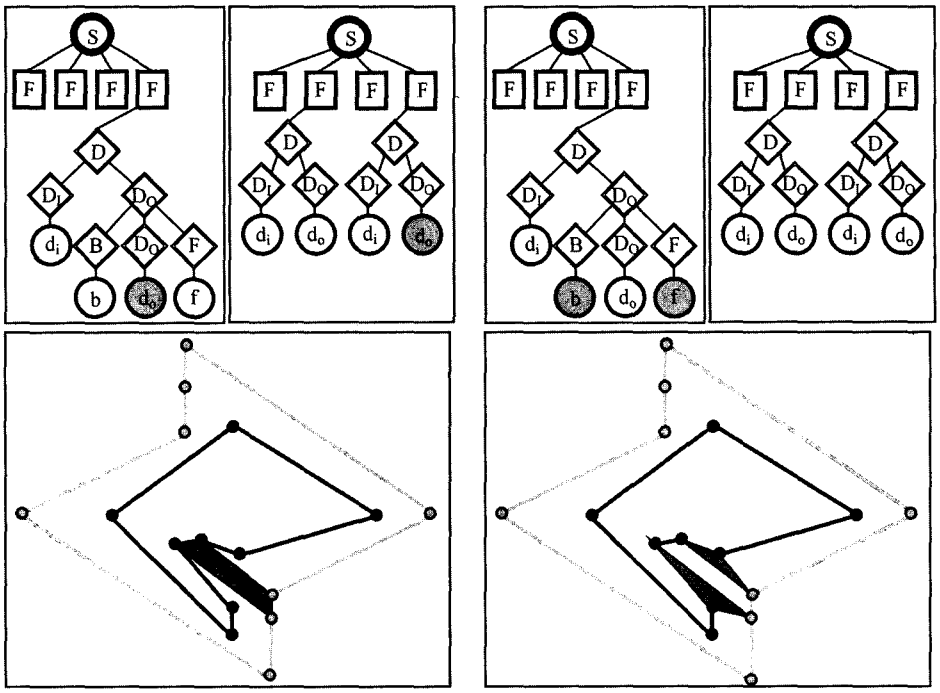


(b) 두 윤곽선들에 대한 유도 트리

그림 B-1 두 윤곽선과 그 유도 트리들

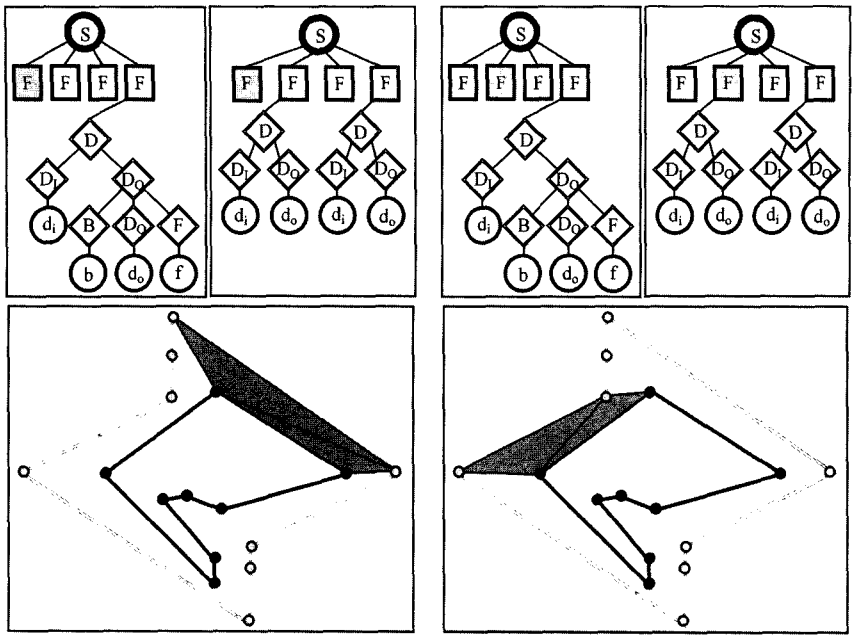


(5) 대응선분에 의한 다각형 (6) 대응선분에 의한 다각형

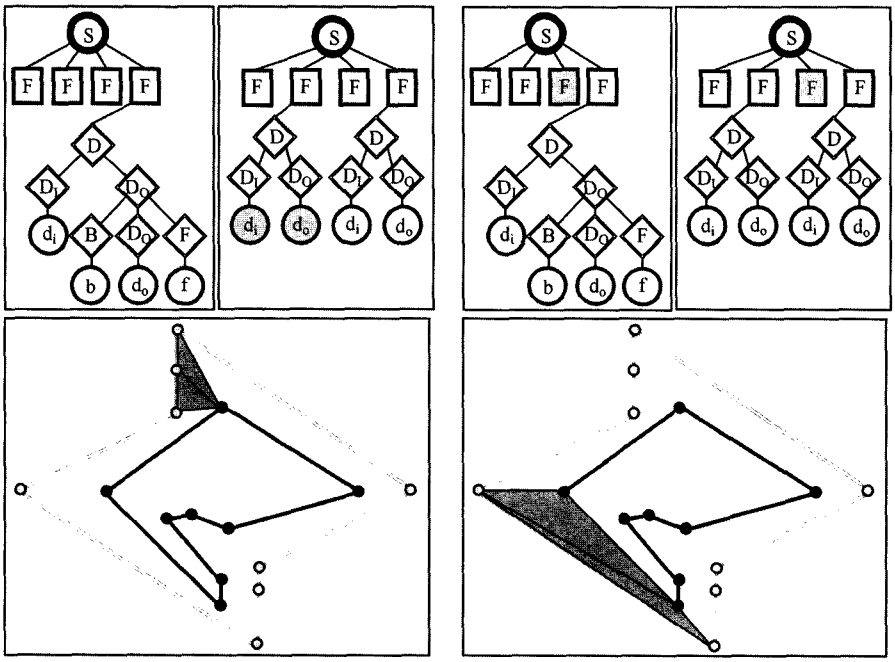


(7) 대응선분에 의한 다각형 (8) 대응점에 의한 다각형

그림 B-2 타일링의 예 (2)



(1) 대응 선분에 의한 다각형 (2) 대응 선분에 의한 다각형



(3) 대응점에 의한 다각형 (4) 대응 선분에 의한 다각형

그림 B-3 타일링의 예 (2)

부록 C. 상세화의 예

그림 C-1 - 그림 C-4에서는 각각 r_4, r_5, r_6, r_7을 방사하면서 발생하는 윤곽선의 상세화 및 그에 따른 다각형 곡면의 상세화가 제시되어 있다.

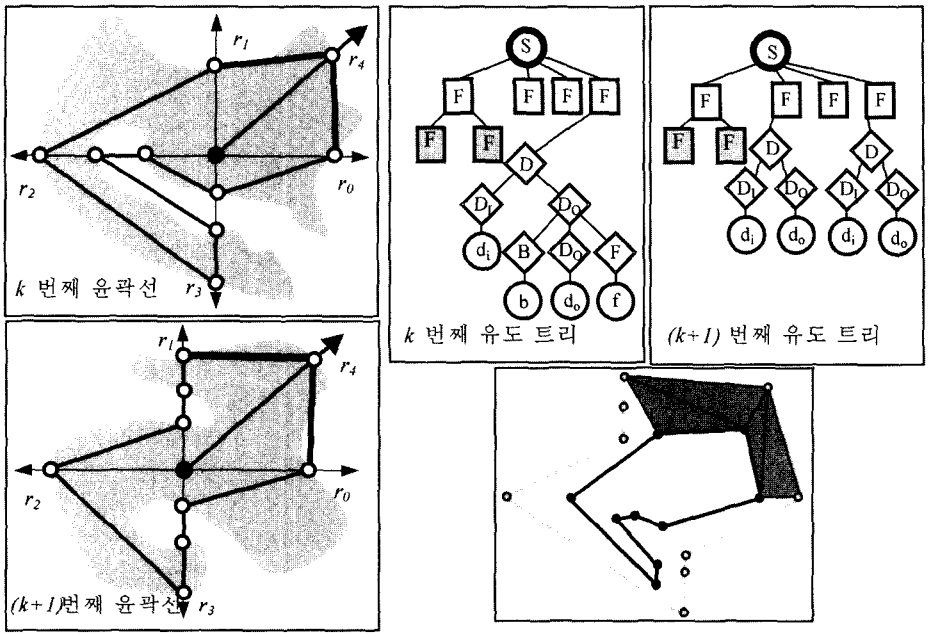


그림 C-1 r_4의 방사에 의한 윤곽선과 다각형 곡면의 상세화

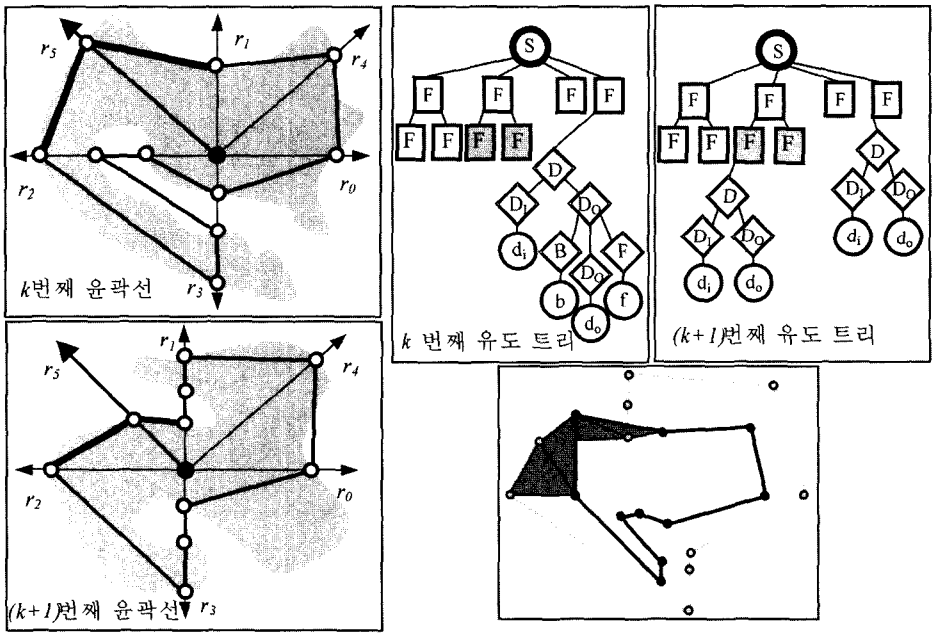


그림 C-2 r_5의 방사에 따른 윤곽선 및 다각형 곡면의 상세화

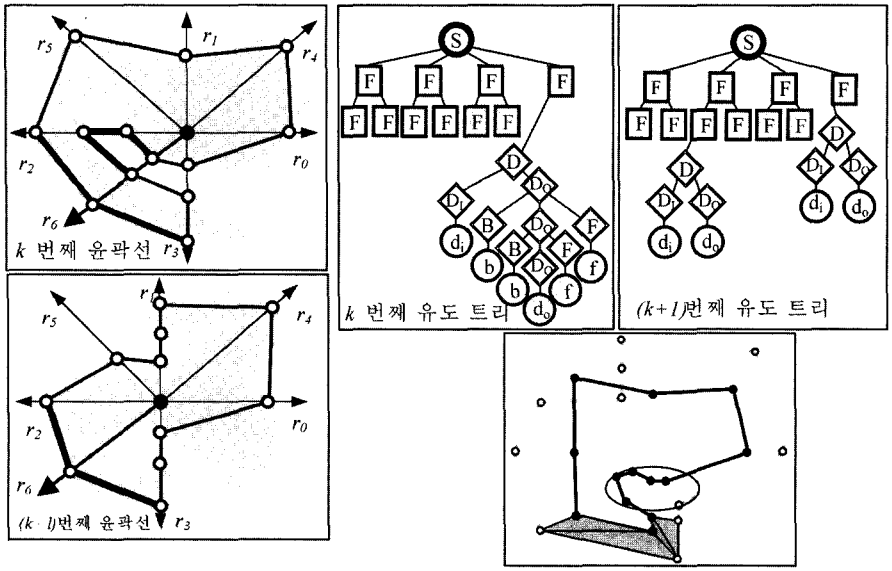


그림 C-3 r_6의 방사에 따른 윤곽선과 다각형 곡면의 상세화

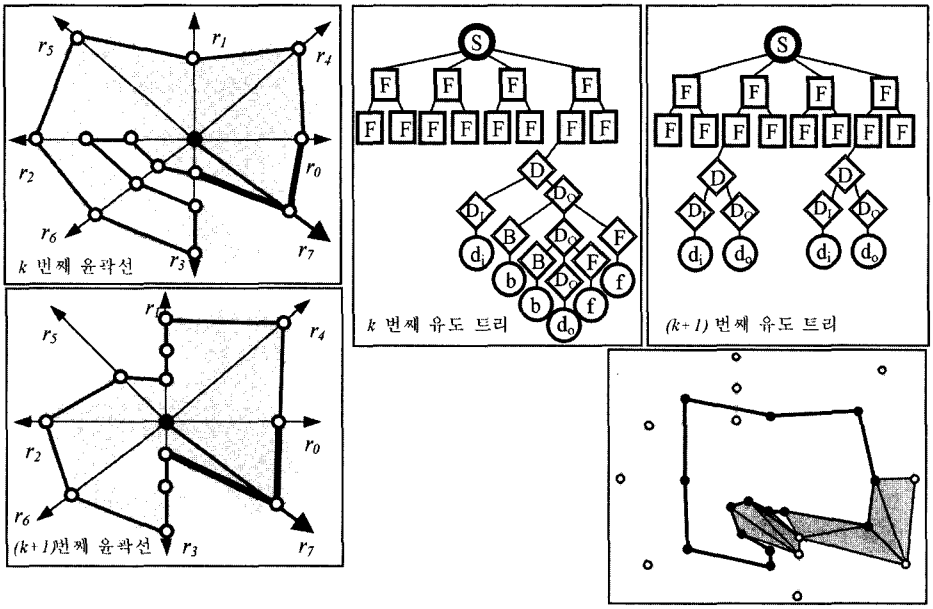


그림 C-4 r_7의 방사에 따른 윤곽선과 다각형 곡면의 상세화

민 경 하

1992년 한국과학기술원 과학기술대학 전산과(학사). 1994년 포항공과대학교 컴퓨터공학과(석사). 2000년 포항공과대학교 컴퓨터공학과(박사). 2000년 8월~2001년 7월 이화여자대학교 정보통신연구소 연구 교수. 2001년 8월~2002년 8월 서강대학교 영상대학원 연구 교수. 2002년 9월~현재 미국 Rutgers Univ. 박사후연구원

이 인 권

1989년 연세대학교 전산과학과 학사. 1992년 포항공대 컴퓨터공학과 석사. 1997년 포항공대 컴퓨터공학과 박사. 1997년~1999년 비엔나 공대 연구원. 1999년~2001년 포항공대 정보통신연구소 선임연구원. 2001년~2003년 아주대학교 미디어학부 조교수. 2003년 9월~현재 연세대학교 컴퓨터과학과 조교수. 관심분야는 컴퓨터 그래픽스, 컴퓨터 게임, 컴퓨터 음악 등