한국 CAD/CAM 학회 논문집
제 9권 제 4 호 2004년 12월 pp. 294-305

학술논문

간결하고 효율적인 폴리곤 메쉬의 표현 구조

박상근*, 이상헌**

A Compact and Efficient Polygonal Mesh Representation

Park, S. K.* and Lee, S. H.**

ABSTRACT

Highly detailed geometric models are rapidly becoming commonplace in computer graphics and other applications. These complex models, which is often represented as complex triangle meshes, mainly suffer from the vast memory requirement for real-time manipulation of arbitrary geometric shapes without loss of data. Various techniques have been devised to challenge these problems in views of geometric processing, not a representation scheme. This paper proposes the new mesh structure for the compact representation and the efficient handling of the highly complex models. To verify the compactness and the efficiency, the memory requirement of our representation is first investigated and compared with other existing representations. And then we analyze the time complexity of our data structure by the most critical operation, that is, the enumeration of the so-called one-ring neighborhood of a vertex. Finally, we evaluate some elementary modeling functions such as mesh smoothing, simplification, and subdivision, which is to demonstrate the effectiveness and robustness of our mesh structure in the context of the geometric modeling and processing.

Key words : Geometric modeling, Computational geometry, Surface representation, Polygonal meshes, Data structure

1. 소 개

1.1 연구배경

복잡한 기하학적 형상을 표현하고 모델링하기 위해 등장한 폴리곤 메쉬(polygonal meshes) 곡면 모델은 CAD/CAM 분야를 포함하여, 컴퓨터 그래픽스(computer graphics) 분야 등에서 표현 방식의 표준으로 현재 널리 사용되고 있다. 특히 삼각형 메쉬 형태가 주로 사용되고 있는데, 이는 간결성, 수치적 안정성, 기하학적 직관성, 그리고 효과적인 렌더링(rendering) 측면에서 다른 형태의 곡면 표현식(예를 들어, NURBS곡면)에 비해 우월성을 가지고 있기 때문이다. 또한 관련된 형상 모델링 알고리즘 및 방법 등을 비전문가라 하더라도 직관적으로 쉽게 표현하고 구현할 수 있어 누구나 쉽게 접근할 수 있다는 장점을 가지고 있다. 여기서 형상 모델링 알고리즘 및 방법이란 기존의 곡면 표현식을 가지고 구현할 수 있는 자유형상 모델링(freeform modeling)을 포함하여 새롭게 그 중요성이 부각되고 있는 다중해상도 모델링(multi-resolution modeling)[1,2]을 포함한다.

한편, 폴리곤 메쉬 모델은 임의의 자유형상을 표현하는데 있어 위상학적 제한이 없다. 즉, 사각형 구조의 기존 NURBS 곡면 등이 가지고 있는 곡면 패치(surface patch) 단위의 표현 방식으로 인해 발생되는 패치들 간의 연속성(continuity) 문제 등이 구조적으로 해결된다. 이는 매우 중요한 특징으로 형상 모델링 기능을 구현하는데 있어서 연속성 유지라는 측면의 추가적인 고려 및 계산이 필요하지 않다는 것을 의미한다. 이밖에 폴리곤 모델은 곡면 가시화 측면에서, 렌더링 작업에서 요구하는 면(facet) 데이터를 추가적인 계산 없이 제공하고 있기 때문에, 기존의 NURBS곡면에서 요구하는 다각형화(polygonization) 계산 작업이 필요 없게 된다. 즉 현존하는 곡면 그래픽스 방식에 부합되어 렌더링 측면에서 많은 이점을 가지고 있는 것이다.

이상과 같은 장점에 비해 폴리곤 메쉬에 의한 자유

*정회원, 충주대학교 기계공학과
**종신회원, 국민대학교 자동차공학전문대학원
- 논문투고일: 2004. 02. 24
- 심사완료일: 2004. 06. 18

형상 표현은 표현 방식 자체가 근사적으로 이루어지기 때문에, 원통(cylinder), 구(sphere), 토러스(torus) 등의 2차 곡면(quadratic surface)을 정확하게 표현하지 못한다. 이는 기존의 NURBS곡면에 비해 표현의 한계를 가지고 있다고 말할 수 있다.

한편, 폴리곤 메쉬 모델은 그 표현자체가 기저함수(basis function) 등을 사용한 수학적 표현식이 아니고 삼각형 형태를 가지는 면(face)들의 집합으로 정의한다. 이로 인해 표현 자체가 근사적이고 구현 측면에서도 다양한 형태의 자료 구조(data structure)를 설계할 수 있어, 효율적인 메모리 관리 및 빠른 형상 모델링 기능 등을 고려하여, 사용 목적에 맞게 특정한 형태의 자료 구조를 선택 혹은 설계해야 할 것이다.

또한, 형상 표현에 사용되는 메모리 사용 측면에서도 설계된 자료 구조가 중복 없이 효율적으로 형상 표현을 위해 메모리를 사용하고 관리하는가, 혹은 자주 사용되지 않는 불필요한 데이터 정보를 저장하고 있지 않은가 등에 관한 고려도 자료 구조 설계에 반영되어야 할 것이다.

본 연구에서는 이상의 폴리곤 메쉬 모델을 구현하기 위해 면-중심(face based)의 삼각형 메쉬 모델을 제안하며, 제안된 자료 구조를 기반으로 형상 표현을 위해 필요한 메모리 사용량에 관한 분석 및 계산 효율성을 직접적으로 측정해 볼 수 있는 자료 탐색 알고리즘의 분석을 통하여, 제안한 자료 구조의 간결성, 효율성 및 그 타당성을 보이고자 한다

1.2 관련 연구

일반적으로 폴리곤 메쉬 모델은 꼭지점(vertex), 모서리(edge), 면(face) 등의 집합과 이들 사이의 위상학적 연결 관계(topology connectivity) 등으로 이루어져 있다. 이를 자료 구조화하기 위해 이들 관계를 기반으로 각 위상요소가 어떻게 정의되고, 참조하는 이웃 요소가 무엇인지를 정의해야 한다. 지금까지 발표된 대표적인 자료 구조를 살펴보면, Winged-edge[3] 자료 구조와 Half-edge[4] 자료 구조가 있다. 이상의 자료 구조는 모서리를 중심으로 위상 관계를 정의하고 있다.

본 연구에서 제안하는 구조는 모서리-중심이 아닌 면-중심의 위상 관계를 설명하고 있다. 이상의 자료 구조들은 각기 고유한 특징을 가지고 그 응용 목적에 맞게 활용되고 있으며, 이밖에 LOD(Level of Detail)에 따라 계층 구조를 가지고 다중해상도 모델링 기능을 지원하는 Progressive mesh[5,6]라는 좀더 응용된 자료 구조가 등장하고 있다.

한편, 자료 구조를 설계함에 있어, 자료 구조의 표현 완벽성, 메모리 사용량, 계산 효율성, 사용 편의성 등을 고려해야 하며, 동시에 비다양체(non-manifold) 형상을 표현할 것인가에 대한 결정도 함께 고려되어야 한다. 여기서 비다양체 형상이란 Fig. 1에서 보는 바와 같이, 하나의 모서리에 3개 이상의 면이 존재할 때, 혹은 하나의 꼭지점에 대해서 그 주변의 이웃 요소들이 위상학적 디스크(topological disc), 즉 원뿔(cone) 형상을 이루지 못할 때 등을 말한다. 여기서 해당하는 비다양체 형상을 이루는 모서리 및 꼭지점을 각각 complex edges, complex vertices라고 부른다.

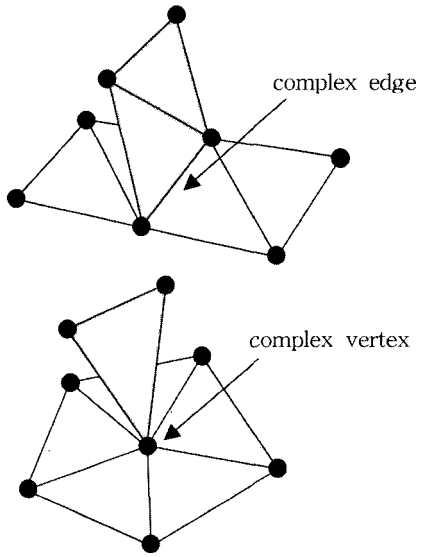


Fig. 1. Non-manifold triangle configurations.

2. 면-중심 폴리곤 메쉬의 표현

2.1 자료 구조

Fig. 2는 본 연구에서 제안하는 삼각형 폴리곤 메쉬 모델의 자료 구조를 설명하고 있다. 그림에서 보는 바와 같이, 하나의 폴리곤 메쉬는 면들의 집합과 꼭지점들의 집합으로 구성되어 있으며, 하나의 면(Fig. 2(a))은 세 개의 꼭지점($\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$)에 의해 삼각형 형상을 정의하며 동시에 이웃 요소로서 세 개의 면($\mathbf{f}_1$, $\mathbf{f}_2$, $\mathbf{f}_3$)을 가리키고 있다. 단, 경계(boundary) 부분에 위치하고 있는 면은 1개 혹은 2개의 면을 이웃 요소로서 가지게 되며, 경계 부분의 면을 찾는데 이웃한 면의 개수가 중요한 단서로서 사용된다. 그리고 하나의 꼭지점(Fig. 2(b))은 형상 정보로서 3차원 좌표값(x, y, z)을 가지고 있으며, 동시에 이웃 요소로서 m개의 면($\mathbf{f}_1$, $\mathbf{f}_2$, …, $\mathbf{f}_m$)을 가리키고 있다. 더불어, 하나의 모서

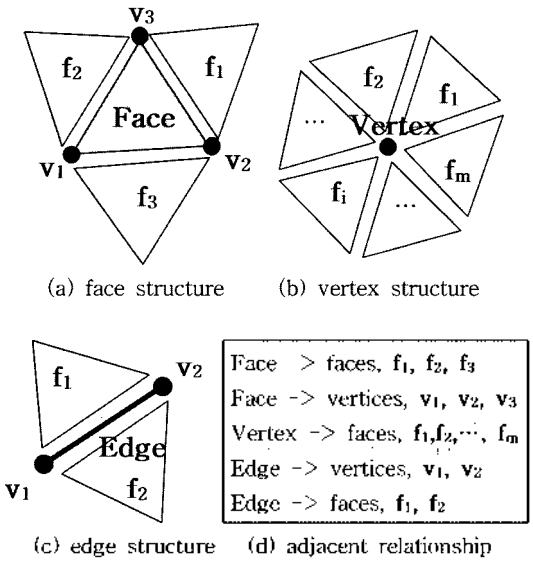


Fig. 2. Topology connectivity of face-based data structure for triangulated polygonal meshes.

```
class Entity {
  int          m_id;
  Attribute*   m_attribute;
};

class Face : public Entity {
  Vertex*  m_vertex[3]; // adjacent vertices
  Face*    m_face[3];   // adjacent faces
};

class Vertex : public Entity {
  int      m_nface;  // number of adjacent faces
  Face*    m_face[]; // adjacent faces
  Point    m_point;  // position (x,y,z)
};

class Edge : public Entity {
  Vertex*  m_vertex[2]; // two end vertices
  Face*    m_face[2];   // adjacent faces
};

class Mesh {
  std::vector<Face*>    m_face_list;
  std::vector<Vertex*>  m_vertex_list;
};
```

Fig. 3. Implementation of face-based data structure with the classes in C++.

리(Fig. 2(c))는 두 개의 꼭지점에 의해 정의하며 필요에 의해 이웃 요소로서 2개의 면을 탐색하여 가지고 있게 된다. 이상의 모서리 요소는 본 연구의 폴리곤 메쉬 자료 구조에 직접적으로 포함되지 않으며, 내부적으로 관련 알고리즘의 구현 및 모델링 기능의 효과적인 수행을 위하여 선택적으로 사용된다. 한편, 이상의 위상학적 연결 구조를 C++언어로 표현하면 Fig. 3과 같다. 참고로 각 위상 요소는 포인터 배열 구조(pointer array)로 인접 위치에 있는 위상 요소를 가리키고 있다.

결국, 본 연구의 자료 구조는 면-중심의 연결 구조와 꼭지점-중심의 연결 구조를 함께 가지며, 추후 설명할 최소의 메모리 사용과 효율적인 계산 성능을 목적으로 하여 설계되었다.

2.2 자료 탐색

자료 탐색(searching)이란 주어진 위상 요소(예, 꼭지점, 면, 모서리)를 중심으로 그 주변에 위치한 위상 요소를 찾아내는 알고리즘을 말한다. 일반적으로 형상 모델링 커널에서 사용되는 자료 구조들은 중복되는 정보를 최소화시키기 위해, 모든 인접 관계에 관한 정보를 직접 저장하지 않고, 일부 인접 관계들만을 저장하고 있다가 필요한 경우 나머지 인접 관계들을 유도해낸다. 꼭지점, 모서리, 면에 의해 정의되는 폴리곤 메쉬 모델의 경우, 크게 다음의 3가지로 인접 관계를 나누어 볼 수 있다.

가. 꼭지점 주변의 이웃 요소인 꼭지점, 모서리, 면을 추출하는 경우

나. 모서리 주변의 이웃 요소인 꼭지점, 모서리, 면을 추출하는 경우

다. 면 주변의 이웃 요소인 꼭지점, 모서리, 면을 추출하는 경우

본 연구의 자료 구조는 이미 인접 관계로서 꼭지점 주변의 면 요소와 면 주변의 꼭지점 및 면 요소들을 가지고 있으므로, 나머지 부분에 대한 탐색 알고리즘을 살펴보면 다음과 같다.

2.2.1 꼭지점 주변의 이웃 꼭지점 찾기 알고리즘

하나의 꼭지점 주변에 인접한 이웃 꼭지점들을 추출하는 알고리즘(VERTICES-OF-VERTEX)으로 Fig. 4와 같다. 꼭지점(*v0*) 주변의 이웃면들은 이미 자료 구조에서 제공되므로 이를 이용하여, 이웃면들의 각 면(*f*)에 대해, 면을 구성하는 3개 꼭지점 중, 처음 주어진 꼭지점(*v0*)의 다음에 위치한 꼭지점(*indx*)을 선택하여 이를 저장(*v_list*)하는 방식으로 이루어져 있다.

```
VERTICES-OF-VERTEX (v0,v_list)
for each face f around v0 do {
  if f->m_vertex[0] is v0, indx = 1;
  else if f->m_vertex[1] is v0, indx = 2;
  else indx = 0;
  add f->m_vertex[indx] to v_list;
}
```

Fig. 4. Query procedure for finding vertices adjacent to a given vertex.

```
EDGES-OF VERTEX (v0,e_list)
do VERTICES-OF-VERTEX (v0,v_list);
for each vertex v in v_list do {
  create an edge e;
  e->m_vertex[0] = v0;
  e >m_vertex[1] = v;
  add e to e_list;
}
```

Fig. 5. Query procedure for finding edges adjacent to a given vertex.

```
EDGES-OF-EDGE (e,e_list)
for each vertex v of e do {
  do EDGES-OF-VERTEX (v,e_list);
  subtract e from e_list;
}
```

Fig. 6. Query procedure for finding edges adjacent to a given edge.

2.2.2 꼭지점 주변의 이웃 모서리 찾기 알고리즘

하나의 꼭지점 주변에 인접한 이웃 모서리들을 추출하는 알고리즘(EDGES-OF-VERTEX)으로 Fig. 5와 같다. 먼저 위에서 기술한 VERTICES-OF-VERTEX에 의해 이웃 꼭지점들을 추출(*v_list*)하고, 추출된 각 꼭지점(*v*)과 처음 주어진 꼭지점(*v0*)을 연결하는 모서리(*e*)를 생성하여 저장(*e_list*)하는 방식으로 구성된다.

2.2.3 모서리 주변의 이웃 꼭지점 찾기 알고리즘

하나의 모서리 주변에 인접한 이웃 꼭지점들을 추출하는 알고리즘은 찾고자 하는 꼭지점들이 모서리를 구성하는 두 꼭지점이므로 특별히 검색할 것이 없다.

2.2.4 모서리 주변의 이웃 모서리 찾기 알고리즘

하나의 모서리 주변에 인접한 이웃 모서리들을 추출하는 알고리즘(EDGES-OF-EDGE)으로 Fig. 6과 같다. 모서리(*e*)를 구성하는 각 꼭지점(*v*)에 대해, 위에서 기술한 EDGES-OF-VERTEX에 의해 이웃 모서리들을 추출(*e_list*)하고, 추출된 모서리들 가운데 처음 주어진 모서리(*e*)를 제외시키는 방식으로 수행된다.

```
FACES-OF EDGE (e,f_list)
for each vertex v of e do {
  for each face f around v do {
    if f is visited at the first time
      make f visited;
    else // already visited
      add f to f_list;
  }
}
```

Fig. 7. Query procedure for finding faces adjacent to a given edge.

```
EDGES-OF-FACE (f,e_list)
v[0] = f->m_vertex[0];
v[1] = f->m_vertex[1];
v[2] = f->m_vertex[2];
v[3] = v[0];
for each i = 0, 1, 2 do {
  create an edge e;
  e->m_vertex[0] = v[i];
  e->m_vertex[1] = v[i+1];
  add e to e_list;
}
```

Fig. 8. Query procedure for finding edges adjacent to a given face.

2.2.5 모서리 주변의 이웃 면 찾기 알고리즘

하나의 모서리 주변에 인접한 이웃면들을 추출하는 알고리즘(FACES-OF-EDGE)으로 Fig. 7과 같다. 모서리(*e*)를 구성하는 각 꼭지점(*v*)에 대해, *v*에 인접한 이웃면들의 각 면(*f*)에 대하여, 처음 방문이면 이를 표시(visited)해 두고, 이미 방문했던 기록이 있다면, 이 면을 저장(*f_list*)하는 방식으로 구성되어 수행된다.

2.2.6 면 주변의 이웃 모서리 찾기 알고리즘

하나의 면 주변에 인접한 이웃 모서리들을 추출하는 알고리즘(EDGES-OF-FACE)으로 Fig. 8과 같다. 면(*f*)을 구성하는 각 꼭지점(*v[i]*, *i*=0,1,2)을 그 순서대로 서로 연결하는 방식으로 이웃 모서리(*e*)들을 생성하고 이를 저장(*e_list*)하는 방식으로 이루어져 있다.

3. 메모리 사용량 및 계산 효율성

3.1 메모리 사용량 분석

본 연구에서 제시한 자료 구조의 메모리 사용량을 살펴보면, 다음과 같이 3가지로 나누어 설명할 수 있다. 첫째, 위상 요소 자체를 저장 및 관리하기 위한 메모리(storage organization), 즉 하나의 메쉬가 nf개의 면들과 nv개의 꼭지점들로 구성되어 있다면, 이를 표현하기 위한 linked list 혹은 STL vector 형태의 $(nf + nv)$개의 포인터(pointer)가 필요하다(여기서 포인터는 4 byte의 크기를 갖는다.).

둘째, 위상 요소들 간의 이웃 관계를 표현하기 위한 메모리(topology connectivity), 즉 하나의 면은 3개의 꼭지점들과 3개의 이웃면들로 구성되어 있으므로 6개의 포인터가 필요하며, 하나의 꼭지점은 이웃면의 개수가 m개일 때, m개의 포인터와 이웃면의 개수를 나타내기 위한 1개의 정수(integer)형 메모리가 필요하다(여기서 정수형은 4 byte의 크기를 갖는다.). 즉,

위상 정보 저장에 필요한 포인터의 개수

$$= nf \cdot 6 + \sum_{i=1}^{nv} (m_i + 1) \tag{1}$$

여기서 nf = 면의 개수, nv = 꼭지점의 개수,
m_i = i번째 꼭지점의 이웃면의 개수이다.

셋째, 형상 정보에 해당하는 점 데이터 저장을 위한 메모리(geometry information), 즉 nv개의 각 꼭지점의 좌표값(x, y, z)을 저장하기 위한 $nv \cdot 3$개의 실수(float)형 메모리가 필요하다(여기서 실수형은 4 byte의 크기를 갖는다.).

한편, 대부분의 메쉬 모델의 경우, 모서리의 개수는 꼭지점의 개수의 대략 3배 정도이며, 면의 개수는 꼭지점의 개수의 대략 2배이고, 꼭지점 주변의 이웃면의 개수는 평균하여 6개이므로, 이를 다음과 같이 쓸 수 있다.

$$ne = nv \cdot 3 \tag{2}$$

$$nf = nv \cdot 2 \tag{3}$$

$$\sum_{i=1}^{nv} m_i = nv \cdot 6 \tag{4}$$

여기서 ne = 모서리의 개수이다.

결국, 본 연구의 자료 구조가 필요로 하는 메모리 사용량은, 식 (2),(3),(4)을 사용하여 포인터 단위의 개수로 나타내면 아래와 같다.

$$(nf+nv)+\left(nf \cdot 6+\sum_{i=1}^{nv}(m_i+1)\right)+(nv \cdot 3)$$
$$= (nv \cdot 2+nv)+(nv \cdot 12+nv \cdot 7)+(nv \cdot 3)$$
$$= nv \cdot 25 \tag{5}$$

한편, 본 연구에서 제안하는 자료 구조가 메모리 사용량 측면에서 대립적 관계에 있는 대표적인 자료 구조인 Half-edge 구조와의 비교를 통하여 상대적으로 우위에 있음을 보이겠다. Fig. 9는 Half-edge 자료 구조의 위상학적 연결 관계를 설명하고 있고, Fig. 10은 이들 위상학적 연결 관계를 C++언어로 표현하고 있다.

그림에서 보는 바와 같이, 하나의 꼭지점은 1개의 half-edge를 가리키고 있으므로 1개의 포인터가 필요하며, 하나의 면도 1개의 half-edge를 가리키고 있으므로 1개의 포인터가 필요하다. 그리고 하나의 half-edge

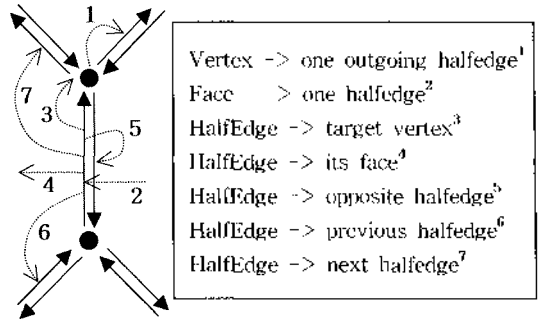


Fig. 9. Topology connectivity of half-edge structure.

```
class Vertex : public Entity {
  HalfEdge*   m_edge;  // outgoing halfedge (1)
};

class Face : public Entity {
  HalfEdge* m_edge; // one of the halfedges
                    // bounding it (2)
};

class HalfEdge : public Entity {
  Vertex* m_vertex; // the vertex it points to (3)
  Face*   m_face;   // the face it belongs to (4)
  HalfEdge* m_opposite; // opposite halfedge (5)
  HalfEdge* m_prev;     // optional (6)
  HalfEdge* m_next;     // next halfedge (7)
};
```

Fig. 10. Implementation of half-edge data structure with the classes in C++.

는 1개의 꼭지점, 1개의 면, 2개의 half-edge(opposite, next)로 구성되어 있으므로 4개의 포인터가 필요하다. 결국, Half-edge 자료 구조가 위상학적 연결 관계를 나타내기 위해 필요로 하는 포인터의 개수는 다음과 같이 나타낼 수 있다.

$$\begin{aligned} & nv+nf+(ne\cdot 2)\cdot 4 \\ &= nv+(nv\cdot 2)+(nv\cdot 3\cdot 2\cdot 4) \\ &= nv\cdot 27 \end{aligned} \tag{6}$$

그리고 위상 요소의 저장 및 관리를 위한 메모리로서 다음의 포인터 개수가 필요하다.

$$\begin{aligned} & nv+nf+(ne\cdot 2) \\ &= nv+nv\cdot 2+(nv\cdot 3\cdot 2) \\ &= nv\cdot 9 \end{aligned} \tag{7}$$

또한 점 데이터의 형상 정보를 저장하기 위한 메모리는 nv · 3개의 포인터가 필요하다. 이는 본 연구의 형상 정보 크기와 같다.

결국, 본 연구의 자료 구조와 Half-edge의 자료 구조를 비교하면 다음의 Table 1과 같고, Half-edge 자료 구조에 대한 본 연구의 자료 구조 메모리 사용량은 $\frac{(memory\ for\ face\ based\ structure)}{(memory\ for\ half\ edge\ structure)}=\frac{nv\cdot 25}{nv\cdot 39}=0.64$ 임을 확인할 수 있다.

이상의 결과를 검증해 보기 위해, Fig. 13, Fig. 15, Fig. 16의 메쉬 모델에 관해, 메모리 사용량을 Byte 단위로 확인해 보았다. 그 결과는 Table 2와 같다. Table 2에서 보는 바와 같이, 본 연구의 자료 구조가 필요로 하는 메모리 사용량은 Half-edge 구조에 비해 Byte 단위의 크기로 대략 0.64배 정도임을 쉽게 확인할 수 있다.

Table 1. Comparison of memory usage between our face based structure and the half-edge structure (*Ratio means the ratio of our data structure to half-edge, and nv the number of vertices.)

Memory Usage Types	Face-based structure	Half-edge structure	Ratio
(1) Storage Organization	nv · 3	nv · 9	0.33
(2) Topology connectivity	nv · 19	nv · 27	0.70
(3) Geometry Information	nv · 3	nv · 3	1.0
Total	nv · 25	nv · 39	0.64

Table 2. Memory usage of sample mesh models (*Ratio means the ratio of proposed structure to half-edge one.)

Mesh model	Number of entities	Memory Usage		Ratio
		Face-based structure	Half-edge structure	
Fig. 13	nf = 69,451 ne = 104,288 nv = 35,947 $\sum_{i=1}^{nv} m_i$ = 208,353	3,415 KB	5,318 KB	0.642
Fig. 15	nf = 6,165 ne = 9,296 nv = 3,132 $\sum_{i=1}^{nv} m_i$ = 18,495	302 KB	472 KB	0.639
Fig. 16	nf = 5,782 ne = 8,694 nv = 2,903 $\sum_{i=1}^{nv} m_i$ = 17,346	283 KB	441 KB	0.640

3.2 계산 효율성 분석

설계된 자료 구조의 계산 효율성은 그 자료 구조에 의해 제공되는 기본적인 탐색 알고리즘의 효율에 의해 결정된다[7,8].

한편, 폴리곤 메쉬 모델을 기반으로 구현되는 여러 종류의 형상 모델링 알고리즘 등에서 가장 많이 사용되며, 그 효율성을 가장 중요시 하는 탐색 알고리즘은 임의의 한 꼭지점 주변의 바로 이웃한 인접 꼭지점들을 찾아내는 알고리즘[9]이다. 이를 흔히 1차 인접 꼭지점(one-ring neighborhood of a vertex)의 탐색이라 부르는데, 본 연구에서 제안하는 자료 구조의 계산 효율성 분석을 위해, 이 탐색 알고리즘이 요구하는 수행 시간을 계산하고, 기존의 Half-edge 자료 구조와의 비교를 통하여 본 연구 자료 구조의 타당성을 보이겠다.

본 연구의 자료 구조에 의한 1차 인접 꼭지점 탐색 알고리즘은 Fig. 4와 같다. 그림에서와 같이, for-문장의 1번 수행 시 작용하는 계산 작업은

- 포인터에 의한 자료 접근 : 4번
- if-문장 수행 : 2번

이다. 여기서, 포인터 접근 시간을 Δt라 정의하면, if-문장의 수행 시간은 대략 Δt로 계산되었다. 본 연구의 모든 계산은 Pentium IV, 512 MB RAM에서 수행되었으며, Δt의 값은 대략, (Δt = 0.012초/10^6번

수행)로 계산되었다. 결국, 꼭지점 주변에 m개의 인접한 이웃 꼭지점들이 존재할 때, for-문장이 m번 수행되므로,

$$m \cdot (4 \cdot \Delta t + 2 \cdot \Delta t) = 6m \cdot \Delta t \tag{8}$$

여기서 m = 꼭지점에 인접한 이웃 꼭지점의 개수
Δt = 포인터에 의한 자료 접근 소요 시간

만큼의 계산 시간이 소요된다.

한편, 비교를 위해 기존의 Half-edge 자료 구조의 1차 인접 꼭지점 탐색 알고리즘을 살펴보면 Fig. 11과 같다. 그림에서와 같이, 꼭지점 주변에 m개의 인접 꼭지점들이 있을 경우, 포인터 자료 접근이 $3m$번이고, if-문장 수행이 m번 일어나므로, 총 $4m \cdot \Delta t$의 계산 시간이 소요된다.

이상의 결과를 통하여, 본 연구의 자료 구조가 Half-edge 구조에 비해 상대적으로 1.5배 느리다는 것을 알 수 있다. 그러나 Δt의 크기가 너무 작아서, 실제로 수행 속도의 차이를 쉽게 느낄 수가 없다. 예를 들어, m = 6인 경우, 1차 인접 꼭지점 탐색의 수행

```
VERTICES-OF-VERTEX (v0,v_list)
e0 = v0->m_edge;
add e0->m_vertex to v_list;
e = e0->m_opposite;
e = e->m_next;
while( e != e0 ) {
  add e->m_vertex to v_list;
  e = e->m_opposite;
  e = e->m_next;
}
```

Fig. 11. Query procedure for finding vertices adjacent to a given vertex in half-edge structure.

Table 3. Computational time complexity of face-based structure
(m means the number of vertices adjacent to a vertex, m_1, m_2 the number of vertices adjacent to edge's vertices, and α the memory allocation time.)

from \ to	Vertex	Edge	Face
Vertex	$6m$	$8m + \alpha$	m
Edge	2	$9(m_1 + m_2) + \alpha$	$2(m_1 + m_2)$
Face	3	$9 + \alpha$	3

Table 4. Computational time complexity of half-edge structure

from \ to	Vertex	Edge	Face
Vertex	$4m$	$3m$	$4m$
Edge	3	$4(m_1 + m_2)$	3
Face	6	3	9

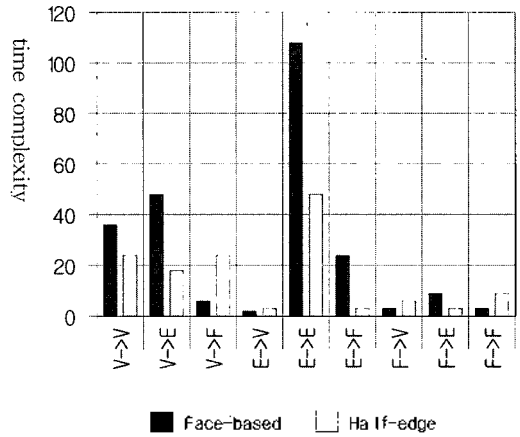


Fig. 12. Comparison of time complexity between our face based structure and the half-edge structure.

시간은 0.432×10^{-6}초이며, Fig. 13의 메쉬 모델의 경우(꼭지점의 개수 nv = 35,947개) 대략 0.0155초 소요된다.

한편, 다른 자료 탐색 알고리즘이 요구하는 탐색 시간을 정리하면 다음의 Table 3과 Table 4와 같다. 여기서 Table 3은 본 연구 자료 구조에 관한 것이고, Table 4는 Half-edge 자료 구조에 관한 것이다. 또한 Fig. 12는 이들의 Table에서 $m = m_1 = m_2 = 6$이라 가정하였을 때의 결과를 비교해 주고 있다.

Fig. 12에서 알 수 있듯이, 총 9개의 인접요소 탐색 알고리즘 가운데, V→F, E→V, F→V, F→F의 4가지 알고리즘은 본 연구의 자료 구조가 더 효율적인 것으로 측정되었고, 나머지 5가지는 Half-edge 구조가 효율적인 것으로 측정되었다. 이러한 탐색 알고리즘들이 특정 모델링 기능의 수행을 위해 동일한 빈도로 사용된다고 가정하면, 본 연구의 자료 구조가 상대적으로 다소 느리다고 판단할 수 있으나, 그 절대값 자체가 너무 작아서 Fig. 13의 메쉬 모델과 같은 크기의 실제 문제에 커다란 영향을 주지는 않는다.

항시 메모리 사용량과 계산 효율성은 서로 이율배반적(trade-off) 관계에 있어 효과적인 수행을 위해서

는 최적 상태의 선택이 필요하다. 만약 메모리 사용 측면에서 아무런 제한을 받지 않는다면, Half-edge구조를 선택할 것이다. 그러나 이러한 가정은 현실성이 없고, 오히려 메모리 사용의 제한으로 인하여, 실제 상황에서 많은 어려움을 받고 있는 것이 현실이다. 주로 과다한 데이터를 읽거나 저장하는 업무가 계산 업무보다 많아지면 기억장치 교통량(memory traffic)이 많아지고 시스템의 효율이 떨어지게 되어, 오히려 탐색 알고리즘 수행을 방해하는 결과를 초래할 수도 있다. 그래서 이러한 메모리 부족으로 인한 어려움을 극복하기 위해, 다중해상도 개념이 소개되었고, 데이터 압축 기술이 등장하게 된 것이다.

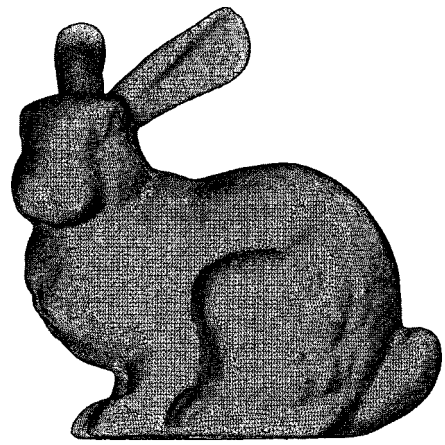
4. 자료 구조의 적용 예제

4.1 메쉬 평활화

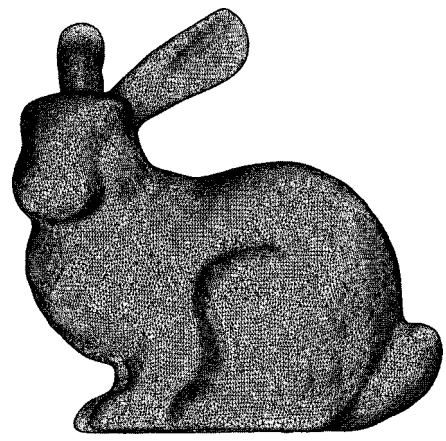
메쉬 평활화(Smoothing)란 폴리곤 메쉬 모델의 전체적인 형상의 모습(global shape)은 유지하면서 작은 잡음(noise)에 해당하는 미소 형상(high frequency details)을 제거하는 알고리즘을 말한다. 대표적인 알고리즘으로 energy minimization[10], diffusion flow[11], PDE discretization[12] 등이 있으며, 흔히 Laplacian smoothing[13] 알고리즘을 자주 사용한다. Laplacian smoothing 알고리즘을 간략히 살펴보면 다음과 같다.

하나의 선택된 꼭지점에 대해, 주변의 이웃한 꼭지점들이 각각 그들의 모서리 방향(선택된 꼭지점에서 주변의 이웃한 꼭지점으로의 방향)으로 선택된 꼭지점을 잡아당길 때, 선택된 꼭지점은 미리 정해놓은 미소양 만큼 이동하게 된다. 이상의 과정을 메쉬를 구성하는 모든 꼭지점에 적용시켜 꼭지점들의 위치를 이동시킨다. 그 다음, 계속해서 주어진 반복 횟수만큼 위에서 설명한 모든 작업을 반복 수행한다. 반복 수행 과정을 살펴보면, 점점 반복 횟수가 증가함에 따라 초기의 거친 표면(잡음 형상)은 점차적으로 제거됨을 확인할 수 있다.

적용 예제로서 Fig. 13을 살펴보면, 본 연구에서 구현한 평활화 알고리즘의 타당성을 확인할 수 있으며, 더불어 본 연구에서 제안한 자료 구조의 타당성을 확인할 수 있다. 또한 자료 구조 자체의 간결성과 효율성으로 인해, 평활화 작업이 안정적으로 빠르게 수행됨을 확인할 수 있다. Fig. 13(a)은 평활화 수행 이전의 초기 모델이고, (b)는 평활화 수행 이후의 결과이다. 초기 모델은 69,451개의 면으로 구성되어 있으며, 평활화 작업에 소요된 시간은 대략 4.4초이다. 참고로 본 연구의 모든 예제들은 Pentium IV, 512 MB RAM 상에서 수행되었다.



(a) initial polygonal mesh



(b) smoothed polygonal mesh

Fig. 13. Mesh smoothing example.

한편, 평활화 작업의 결과를 곡률 분포의 가시화를 통하여 평가해 볼 수 있는데, Fig. 14는 Fig. 13의 메쉬 모델의 곡률 분포 상태를 색깔 매핑을 통하여 가시화한 결과를 보여주고 있다. 그림에서 보는 바와 같이, 평활화 이전의 곡률 분포(Fig. 14(a))가 평활화 이후의 곡률 분포(Fig. 14(b))에 비해 색깔 변화가 심하게 나타남을 통하여 곡률 변화의 정도가 큼을 확인할 수 있다.

4.2 메쉬 간략화

메쉬 간략화(Simplification)란 주어진 폴리곤 메쉬 모델의 특징적 형상을 그대로 유지하면서 폴리곤 모델을 구성하는 면, 모서리 혹은 꼭지점의 개수를 감소

(a) initial polygonal mesh

(b) smoothed polygonal mesh

Fig. 14. Curvature plot of mesh smoothing example shown in Fig. 13.

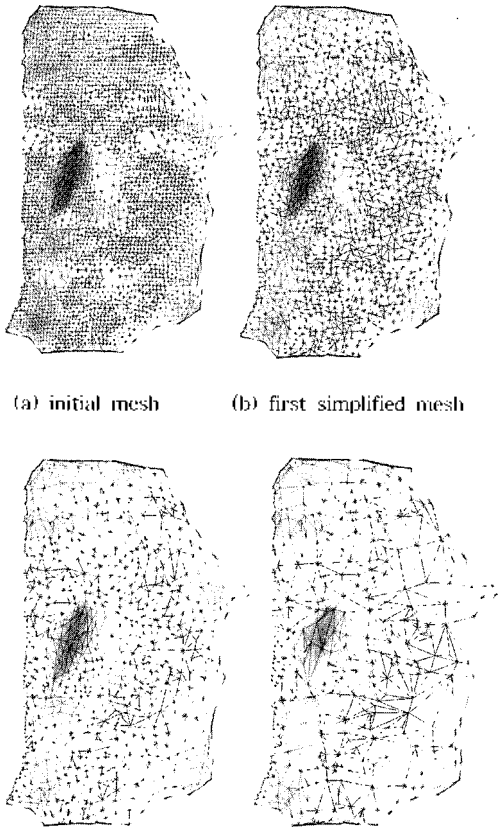
시키는 알고리즘을 말한다. 이는 주로 실시간의 빠른 렌더링 작업을 위하여 주로 사용한다. 대표적인 간략화 알고리즘으로 Turk의 re-tiling approach[14], Schroeder의 vertex removal approach[15], Rossignac과 Borrel의 vertex clustering approach[16], Hoppe의 mesh optimization approach[17], 그리고 Garland의 quadric error metrics approach[18] 등이 있다.

여기서 re-tiling 방식은 곡면의 곡률이 큰 부위에 추가점을 삽입하고 점들 간의 반발력을 사용하여 분포시키는 방식이다. vertex removal 방식은 꼭지점을 제거하고 그 결과 생긴 구멍을 삼각형으로 채우는 방식으로서, 제거된 꼭지점과 새로 채워진 삼각형들 간의 거리를 근사오차로 정의하여 계산한다. vertex clustering 방식은 3차원 격자망(spatial grid)을 대상 모델에 겹쳐놓고, 격자망을 구성하는 각각의 셀(cell)에 대해, 셀 안에 존재하는 모든 꼭지점을 하나의 꼭지점으로 합성(merge)하는 방식으로서, 단순하고 효율적인 알고리즘이나, 곡률 등과 같은 곡면의 성질을 반영하고 있지 않다.

mesh optimization은 에너지 함수(energy function)의 최소화를 통하여 꼭지점의 개수가 최소가 되는 삼각형 메쉬를 찾는 최적화 방식으로서, 꼭지점을 제거하고 재배치하는 방식으로 메쉬의 간략화가 이루어지며, 사용자의 요구에 따라 새로운 구속조건을 추가할 수 있는 특징을 가지고 있다. 그리고 quadric error metrics 방식은 모서리 붕괴(edge collapse)시 새로 생길 하나의 꼭지점과 붕괴 이전 모서리의 이웃면들 사이의 거리 제곱의 합을 에러값으로 정의하고, 모든 모서리에 대해 에러값을 계산한 이후, 최대값을 가지는 모서리부터 먼저 모서리 붕괴 알고리즘에 의해 제거해 나가는 방식이다.

본 연구에서는 quadric error metrics 방식을 기반으로 간략화 알고리즘을 구현하였는데, 빠른 수행 결과를 위하여 모든 모서리에서 quadric error 값을 계산하지 않고 무작위로 8개의 모서리를 선택하여 quadric error값을 계산하고 이중에서 최대값을 가지는 모서리를 제거하는 방식으로 수정 보완하여 구현하였다. 최대 error값을 가지는 모서리를 찾고, 제거할 때마다 모든 모서리에서의 error값을 update해야 하는 기존의 계산적 부담을 줄이기 위해, 통계학적 측면에서 8개의 모서리만을 임의로 추출 선택하여 계산함으로써, 그 결과의 정확성은 원래의 quadric error metrics 방식과 비교될 수 있으나, 커다란 차이 없이 빠른 계산 결과를 기대할 수 있다. 이는 응용 목적에 따라 사용자로 하여금 선택적으로 사용하게 함으로써 응용 분야의 특성을 반영하면 된다.

적용 예제로서 Fig. 15를 살펴보면, 본 연구에서 구현한 간략화 알고리즘의 타당성을 확인할 수 있다. 또한 본 연구에서 제안한 자료 구조의 타당성 및 효율성도 확인할 수 있다. Fig. 15(a)는 간략화 이전의 초기 모델이고, (b)는 (a)의 초기 모델을 간략화한 결과이고, (c)는 (b)을 다시 간략화한 결과이고, (d)는 (c)을 다시 간략화한 결과이다. (a)의 메쉬는 6165개, (b)는 3081개, (c)는 1539개, (d)는 779개의 면들을 가지고 있으며, 각 간략화 작업에 소요된 시간은 각각 0.344초, 0.203초, 0.140초이다.



(a) initial mesh (b) first simplified mesh

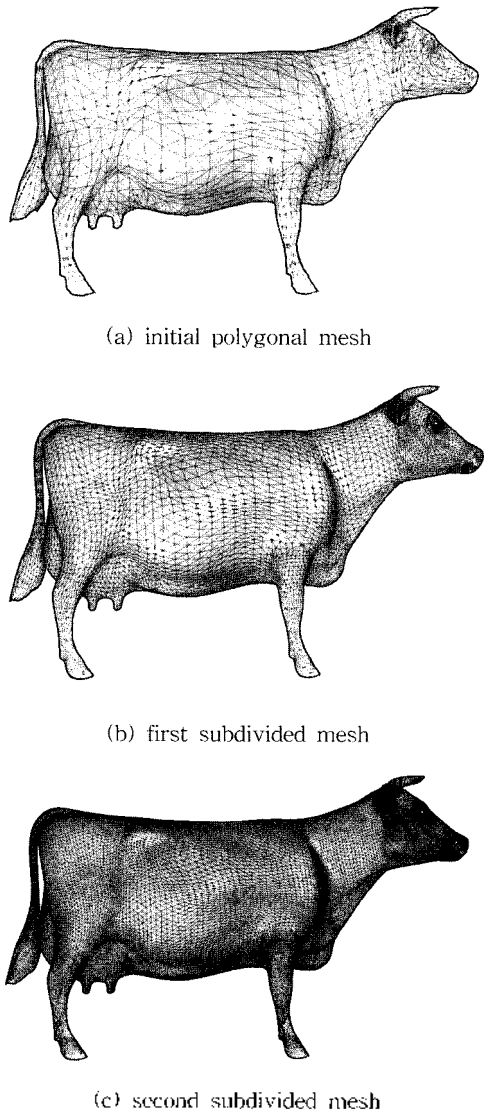
(c) second simplified mesh (d) final simplified mesh

Fig. 15. Mesh simplification examples.

4.3 메쉬 분할

메쉬 분할(Subdivision)[19]이란 폴리곤 메쉬 모델의 면을 혹은 꼭지점을 반복하여 분할하여 매끄러운 곡면을 생성해 내는 알고리즘을 말한다. 대표적인 분할 알고리즘으로 Doo-Sabin[20] 방식, Catmull-Clark[21] 방식, Kobbelt[22] 방식, Butterfly[23] 방식, Loop[24] 방식, $\sqrt{3}$[25] 방식 등이 있다. 이들 방식들은 크게 면-분할 방식(face-split scheme)과 꼭지점-분할 방식(vertex-split scheme)으로 분류할 수 있으며, 및 각 형이냐에 따라 quadrilateral과 triangular으로 분류되며, 연속성에 따라 C^1과 C^2로, 그리고 근사법(approximation)이나 보간법(interpolation)이냐에 따라 분류하여 비교할 수 있다.

본 연구에서는 면-분할 방식, 삼각형 면, C^2 연속성, 근사법으로 분류되는 Loop 방식을 기반으로 분할 알고리즘을 구현하였는데, 분할 작업 수행 시 필요에 따라 유지되어야 하는 특징 형상을 보존하기 위해, 꼭지점을 boundary형, crease형, corner형 등으로 분류하여 Loop 방식의 mask 적용법을 약간 수정 및 보완하여 구현하였다.



(a) initial polygonal mesh

(b) first subdivided mesh

(c) second subdivided mesh

Fig. 16. Mesh subdivision examples.

적용 예제로서 Fig. 16을 살펴보면, 본 연구에서 구현한 분할 알고리즘의 타당성을 확인할 수 있다. 또한 본 연구에서 제시한 폴리곤 메쉬 표현 구조의 타당성 및 효율성을 확인할 수 있다. Fig. 16(a)는 분할 이전의 초기 모델이고, (b)는 (a)의 초기 모델을 분할한 결과이고, (c)는 다시 분할 작업을 수행한 결과이다. (a)의 메쉬는 5782개, (b)는 23128개, (c)는 92512개의 면들을 가지고 있으며, 각 분할 작업에 소요된 시간은

각각 0.282초, 1.281초이다.

5. 결 론

본 연구에서는 면-중심의 삼각형 폴리곤 메쉬 모델의 자료 구조를 제안하였고, 제안된 구조가 주어진 곡면 상의 n개의 측정점에 대해, 실시간의 자료 탐색을 제공하면서 동시에 기존의 Half-edge 구조에 비해 0.64배의 메모리 사용을 통하여, 위상 구조를 충분히 표현할 수 있음을 보였다. 또한 형상 모델링 과정에서 자주 사용하는 메쉬의 평활화, 간략화, 분할 기능 등에 관해 간략히 소개하였고, 동시에 본 연구의 자료 구조에 기반을 두어 수행된 결과를 보여 주었다. 이들 결과의 빠른 수행을 통하여, 자료 탐색의 실시간 수행 여부를 확인할 수 있었다.

결국, 폴리곤 메쉬 모델을 가지고 임의의 형상을 표현하고, 모델링 작업을 수행해 나갈 때, 자료 구조의 성능(performance)을 결정하는 두 가지 요소인 형상 표현을 위한 메모리 사용량 측면과 신속한 모델링 작업 수행을 위한 계산 비용 측면에서, 본 연구 자료 구조의 효율성 및 타당성을 확인할 수 있었다.

또한, 본 연구의 자료 구조는 그 위상학적 연결 구조가 이해하기 쉽고, 이로 인해 자료 구조로서 관련 알고리즘을 구현하기 쉬우며, 사용자에게 자료 구조 커널의 편리한 사용을 위하여 추가로 인터페이스 구조를 설계하고 구현해야 할 필요가 없다.

추후 본 연구를 바탕으로 형상 모델링을 위한 고급 기능의 구현이 필요하다. 예를 들어, 자유형상 모델링(freeform modeling) 기능의 구현을 위해 사용자와 대화식으로 작업하면서 곡면의 형상을 실시간으로 변형해 나가는 모델링 방식, 더 나아가 물리 법칙에 근거하여 형상 변형을 모사 혹은 유도하는 물리법칙 기반의 모델링(physics-based modeling) 방식 등에 관한 연구가 필요하다. 그리고 최근에 데이터의 계층 구조화를 통하여, 빠른 렌더링의 수행 및 빠른 데이터의 전송[26] 등을 목적으로 LOD(level of details) 기반의 다중 해상도 모델링(multi-resolution modeling) 방식이 활발히 연구되고 있는데, 본 연구에서도 제안된 자료 구조의 확장 및 기본 오퍼레이션 기능의 수정 보완을 통하여, 다중 해상도 표현을 위한 간결하고 효율적인 자료 구조의 연구가 필요하겠다.

감사의 글

본 연구는 과학기술부 목적기초연구(R05-2004-000-10448-0) 지원으로 수행되었음.

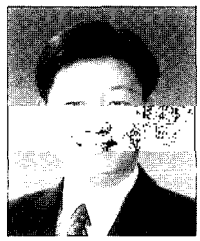
참고문헌

1. Zorin, D., Schroeder, P. and Sweldens, W., "Interactive multiresolution mesh editing," *In SIGGRAPH 97 Conference Proceedings*, pp. 259-268, 1997.
2. Lee, S., "Interactive multiresolution editing of arbitrary meshes," *Computer Graphics Forum*, Vol. 18, No. 3, pp. 73-82, 1999.
3. Baumgart, B. G., "Winged-edge polyhedron representation," *Technical Report STAN-CS-320*, Standford University, Computer Science Department, 1972.
4. Campagna, S., Kobbelt, L. and Seidel, H.-P., "Directed edges: A scalable representation for triangle meshes," *Journal of Graphics Tools*, Vol. 3, No. 4, pp. 1-11, 1998.
5. Hoppe, H., "Progressive meshes," *In SIGGRAPH 96 Conference Proceedings*, pp. 99-108, 1996.
6. Hoppe, H., "Efficient implementation of progressive meshes," *Computer and Graphics*, Vol. 22, No. 1, pp. 27-36, 1998.
7. Lee, S. H. and Lee, K. W., "Partial entity structure: A compact boundary representation for non-manifold geometric modeling," *ASME Journal of Computing & Information Science in Engineering*, Vol. 1, No. 4, pp. 356-365, 2001.
8. Woo, T. C., "A combinational analysis of boundary data structure schemata," *IEEE Comput. Graphics Appl.*, Vol. 5, No. 3, pp. 19-27, 1985.
9. OpenMesh homepage, http://www.openmesh.org
10. Welch, W. and Witkin, A., "Free-form shape design using triangulated surfaces," *In SIGGRAPH 94 Conference Proceedings*, pp. 247-256, 1994.
11. Desbrun, M., Meyer, M., Schroder, P. and Barr, A. H., "Implicit fairing of irregular meshes using diffusion and curvature flow," *In SIGGRAPH 99 Conference Proceedings*, pp. 317-324, 1999.
12. Schneider, R. and Kobbelt, L., "Geometric fair meshes with G1 boundary conditions," *In Proceedings Geometric Modeling and Processing*, pp. 251-261, 2000.
13. Vollmer, J., Mencl, R. and Muller, H., "Improved laplacian smoothing of noisy surface meshes," *In Computer Graphics Forum (Proc. Eurographics)*, pp. 131-138, 1999.
14. Turk, G., "Re-tiling polygonal surfaces," *In Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, Vol. 26, pp. 55-64, 1992.
15. Schroeder, W. J., Zarge, J. A. and Lorensen, W. E., "Decimation of triangle meshes," *In Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, Vol. 26, pp. 65-70, 1992.
16. Rossignac, J. and Borrel, P., "Multi-resolution 3D

approximation for rendering complex scenes," *In Second Conference on Geometric Modeling in Computer Graphics*, Genova, Italy, pp. 453-465, 1993.
17. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., "Mesh optimization," *In SIGGRAPH 93 Conference Proceedings*, pp. 19-26, 1993.
18. Garland, M. and Heckbert, P., "Surface simplification using quadric error metrics," *In SIGGRAPH 97 Proceedings*, pp. 209-216, 1997.
19. Zorin, D., "Subdivision for modeling and animation," In *SIGGRAPH Course Notes*, 2000.
20. Doo, D. and Sabin, M., "Behavior of recursive subdivision surfaces near extraordinary points," *Computer Aided Design*, Vol. 10, pp. 356-360, 1978.
21. Catmul, E. and Clark, J., "Recursively generated B-spline surfaces on arbitrary topological meshes," *Computer Aided Design*, Vol. 10, pp. 350-355, 1978.
22. Kobbelt, L., "Interpolatory subdivision on open quadrilateral nets with arbitrary topology," *Computer Graphics Forum*, Vol. 15, No. 3, pp. 409-420, 1996.
23. Dyn, N., Levin, D. and Gregory, J. A., "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Transactions on Graphics*, Vol. 9, No. 2, pp. 160-169, 1990.
24. Loop, C. T., *Smooth subdivision surfaces based on triangles*, Master's thesis, University of Utah, Department of Mathematics, 1987.
25. Kobbelt, L., "$\sqrt{3}$ Subdivision," *In Computer Graphics Proceedings*, Annual Conference Series, 2000.
26. 공창환, 김창헌, "LOD Rendering과 전송을 위한 Mesh의 Multiresolution 표현," 한국캐드캠학회 학술발표회 논문집, pp. 177-182, 1998.

박 상 근

1991년 포항공과대학교 기계공학과 학사
1993년 서울대학교 기계설계학과 석사
1997년 서울대학교 기계설계학과 박사
1997년~1999년 삼성SDS 정보기술연구소 선임연구원
2000년 서울대 BK21 기계분야사업단 계약교수
2000년~2002년 (주)K&I 테크놀로지 전무기술이사
2003년~현재 충주대학교 기계공학과 전임강사
관심분야: Computational Geometry, CAD/CAM, Scientific Data Visualization, Virtual Engineering



이 상 헌

1986년 서울대학교 기계설계학과 학사
1988년 서울대학교 기계설계학과 석사
1993년 서울대학교 기계설계학과 박사
1993년~1995년 신도리코 기술연구소 책임연구원
1996년 대우 고등기술연구원 선임연구원
1996년~현재 국민대학교 부교수
관심분야: CAD/CAM, 3D Geometric Modeling, Die & Mold Design, Virtual Design and Manufacturing