技術論文 한국정밀공학회지 제13권 제10호 (1996년 10월)
Journal of the Korean Society of Precision Engineering Vol. 13. No 10. OCTOBER. 1996.

급속조형시스템을 위한 STL 포맷의 오류 검증에 관한 연구

최 홍 태*, 이 석 희**

A Study on Error Verification of STL Format for Rapid Prototyping System

Hong-Tae Choi*, Seok-Hee Lee**

ABSTRACT

As industrial standard data, the STL format which approximates three dimensional CAD model to triangular facets, is used for RP(Rapid Prototyping) system in recent days. Because most RP systems take the only form of two dimensional line segments as an input stream inspite of its imperfectness while converting into STL format, a CAD model is converted into a standard industrial format which is composed of many triangular facets. The error verifying process is composed of four main steps, and these are 1) Remove facets with two or more vertices equal to each other. 2) Fix overlapping error such as more than three facets adjacent to an edge. 3) Fill holes in the mesh by using Delaunay triangulation method. 4) Correct the wrong direction and normal vectors. This paper is concerned with searching the mentioned errors in advance and modifying them.

Key Words : Rapid Prototyping system(급속조형시스템), STL format(STL 포맷), STL translator(STL 변환기), triangular facet(삼각형 면), normal vector(법선 벡터), overlapping error(중복 오류), Delaunay triangulation method(Delaunay 삼각화 기법)

1. 서 론

급속조형시스템을 이용해서 제작할 부품은 일반적으로 3차원의 부피를 가지는 CAD 객체로 모델링되어야 한다. 이들 CAD 모델은 급속조형장치가 인지할 수 있는 STL 포맷으로 불리워지는 산업 표준 데이터로 변환된다. 오늘날 대부분의 급속조형시스템의 입력 데이터로 자리 잡고 있는 STL 포맷은 Stereolithography 기술로서 잘 알려진 미국의 3D Systems사를 위해 Albert Consulting Group이 개발한 것으로 3차원의 닫혀있는 서피스모델을 삼각형 facet으로 근사화 시킨 것이다. 개발자 Albert는 모든 CAD 시스템에 공통 인터페이스를 쉽게 제공하기

* 부산대학교 생산기계공학과 대학원
** 부산대학교 생산기계공학과 및 기계기술연구소

위해 이러한 포맷을 선택하였다[1].

현재 대부분의 상용 CAD 시스템이 STL 포맷을 지원하고 있으며, Table 1의 IGES, DXF, VDAFS, STEP 등과 같은 각종 CAD 교환 데이터를 STL 파일로 변환시키는 소프트웨어들이 개발되고 있다. 이러한 STL 변환기들은 대부분 CAD 판매자와 관련 분야 중간 개발자들이 제공하고 있지만, CAD 모델에서 변환된 STL 포맷에서 하나의 facet에 이웃한 facet이 두개 이상 존재하는 중복 오류(overlapping error)와 facet 데이터가 빠져 구멍이 생긴 오류(hole error) 등이 발생하는 문제점이 지적되어 왔다.

현재 STL 포맷 변환 및 오류 검증용으로 상용화된 소프트웨어는 다음과 같은 종류가 있다. Brock Rooney는 보다 신뢰성 있는 STL 포맷 출력을 위하여 IGES를 STL로 변환시키는 STL 변환기인 Brockware를 개발하여 판매하고 있다. 여기서는 서피스모델 변환에 효과적이며 삼각형 facet이 누락되어 있거나 두 이웃한 facet 사이의 틈과 같은 오류가 있는 STL 파일도 수정할 수 있는 특징을 가지고 있다. C-TAD 시스템에서는 Ford PDGS와 IGES 파일을 STL 포맷으로 변환하는 소프트웨어를 판매하고 있고, Autodesk사에서는 AutoCAD를 위해 AME 2.0과 같이 STL 변환기를, Pro/Engineer와 I-DEAS 시스템에서도 STL 변환기를 제공하고 있다[1].

Imageware의 Rapid Prototyping Module(RPM)은 급속조형 데이터를 생성, 편집, 수정할 수 있는 7가지 Tools들이 있다. 이 중 Repair Tools에서는 인접한 다각형 사이의 간극을 연결하고, 중첩 또는 교차되는 다각형뿐만 아니라 서피스 법선벡터의 방향이 거꾸로된 것을 수정하는 기능을 제공하고 있다[2].

DeskArtes의 Rapid Tools는 급속조형 사용자를 위한 유틸리티로 IGES 또는 VDA-FS 모델의 위상을 검사하여 이들의 설계 오류를 자동적으로 또는 대화식으로 고친다. 특히 이들 모델을 STL 포맷으로 변환하기 전에 중첩된 면을 제거하고, 기타 오류 등을 찾아내는 기능을 제공하고 있다[3]. 이상과 같은 상용 소프트웨어는 개발비 및 상업적인 이유로 인해 상당히 고가로 시판되고 있는 실정이다

표면 정도가 좋은 급속조형물을 얻기 위한 가장 좋은 방법은 정확한 곡면 수학식으로 부품을 표현하는 것이다. 그러나 대부분의 급속조형시스템은 2차원 직선 세그먼트(line segment)만을 받아들이기 때문에 삼각형 facet의 모서리 중복 및 실제 CAD 모델과의 근사 오차 등과 같은 STL 포맷의 단점에도 불구하고, CAD 모델을 삼각형 facet으로 근사시킨 STL 포맷을 산업 표준 파일로 사용하고 있다.

따라서 본 연구에서는 여러 STL 변환기를 통해서 CAD 모델을 STL 포맷으로 변환할 때 발생하는 오류를 사전에 찾아내고 이를 수정함으로써 이후의 데이터 변환 공정이 일관되고 신뢰성이 있도록 하는데 그 목적이 있다.

Table 1 Current standards of neutral file format of CAD data

데이터 포맷	설 명
IGES	Initial Graphics Exchange Specification, 서로 다른 CAD 시스템간에 그래픽 데이터를 교환하는 표준, 1979년 미국에서 시작, 현재 ISO 표준으로 제정되어 있음
DXF	Data Interchange File, AutoCAD 데이터 호환용 표준, AutoDesk사 제품
VDAFS	Verband der Deutschen Automobilindustrie Flächen Schnittstelle, 1983년 독일 자동차공업회에서 제정한 표준으로 Sculptured Surface에 대한 인터페이스가 주 내용임
STEP	STandard for the Exchange of Product model data, 1983년 ISO 제184 기술위원회의(산업 자동화 시스템) 제4소위원회 Working Group 1에서 수행, 도면 데이터 뿐만 아니라 제품 데이터까지도 호환될 수 있는 국제적으로 사용될 차세대 중립형태 표준

2. 관련연구

국외에서는 Fumiki Tanaka[4]등이 STL 파일에서 얻어진 점군 데이터를 이용하여 STL 포맷의 두가지 오류 즉, 삼각형 패치 사이에 facet 데이터가 빠져있는 경우와 하나의 모서리(edge)에 2개 이상의 facet이 중복되게 겹쳐 있는 경우의 점군 데이터를 2차원 평면에 투영하고, 이를 Delaunay 삼각형 분할을 기초로한 삼각패치 재구성법을 제안하였다. 그러나 위와 같은 오류가 있는 facet을 찾아내는 알고리즘 제시가 없고, 삼각형 facet의 법선벡터(normal vector) 오류에 관한 검증은 고려하지 않았다.

M.J.Wozny[5]등은 기존의 STL 파일의 문제점을 지적하고 이를 개선하기 위한 STL 포맷의 중복된 정점(vertex)을 줄이기 위해 위상정보가 부족한 STL 파일을 기초로 하여 정점, 모서리, 그리고 면의 색인 리스트(index list)를 가진 facet solid entity로 표현된 RPI(Rensse-

laer Polytechnic Institute) 포맷을 제안하였다. 그러나 대부분의 급속조형장치들이 STL 포맷을 표준으로 지원하기 때문에 새로운 포맷의 제안은 데이터 변환 공정의 호환성에 문제가 있으며, 새로운 RPI 포맷도 입력 데이터로 STL 파일을 사용하기 때문에 오류가 없는 완벽한 STL 파일을 대상으로 하였다.

A.Dolenc[6]등은 기존의 CAD 데이터에서 STL 포맷으로의 변환의 결과가 만족스럽지 못하기 때문에 IGES 파일을 와이어프레임 모델의 VDAFS 포맷으로 변환하고, 이를 면 표현(Facetted Representation)방식의 STL 포맷으로의 변환 절차를 제시하였지만, 최종적으로 생성된 STL 파일의 신뢰성을 확인하기 위한 오류 검증에 관한 부분은 언급하지 않고 있다.

P. Vuyyuru[7]등은 기존의 STL 파일은 CAD 모델을 3차원 삼각형 facet으로 근사하기 때문에 실제 CAD 모델과 삼각형 facet 사이에 오차가 생긴다. 따라서 facet 수를 많게 하면 이런 오차를 줄일 수 있지만, 상대적으로 파일 크기가 커지게 되어 단면 슬라이스 작업과 같은 후공정이 길어지는 문제점 및 급속조형물의 표면 정도를 개선하기 위해서 NURBS(Non-Uniform Rational B-Spline) 곡선을 SLA(StereoLithography Apparatus)가 받아들일 수 있는 2차원 직선 세그먼트로 분할하는 방법을 제시하였다.

국내에서는 김준안[8]등이 STL 파일을 z축으로 일정한 간격 만큼 절단한 단면 데이터를 이루는 loop 정보에서 offset 데이터를 생성하는 알고리즘에 관한 연구를 하였다. 이 연구에서는 오류가 있는 STL 파일을 입력을 하게 되면 open loop가 발생하기 때문에 이를 해결하기 위해 레이저빔의 직경을 오차구간으로 정하고 open loop된 길이가 오차구간 이내에 들면 이웃 선분과 연결하여 close loop을 만들지만 오차구간 이상이 되면 사용자가 직접 가장 가까운 선분과 연결하고 아래층 또는 위층과 확인하는 방법을 제시하였다. 그러나 이는 offset 정보생성 이전 공정이 길어지고 사용자가 각 층별로 loop 데이터를 일일이 검사해야 하는 문제점이 있다.

3. 본 론

몇몇 CAD 시스템에서 지원하고 있는 STL 변환기들이 만족할 만한 결과를 주지 못하기 때문에 급속조형에 필요한 파트의 최적자세 결정, 지지대 구조 생성, 슬라이싱 등과 같은 CAD 프로세스에서 많은 문제점을 야기시킨다.

따라서 본 연구에서는 STL 파일의 중요한 오류의 원인을 찾아서 분류하고, 이를 수정하여 신뢰성 있는 STL 파일을 생성하고자 한다.

STL 포맷의 오류 검증 프로세스를 단계별로 도식화하면 Fig. 1과 같다.

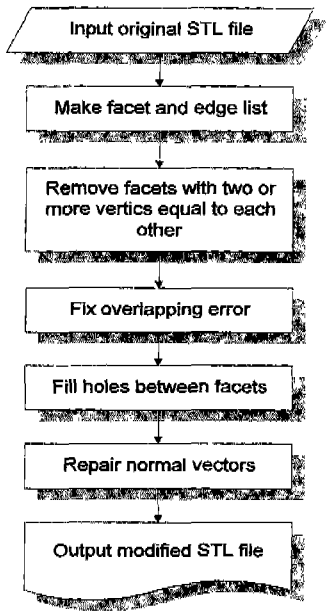

Fig. 1 Error verifying process of STL format

3.1 STL 포맷의 구성 및 오류 발생 문제점

급속조형기술에서는 3차원 모델을 삼각형으로 근사화시킨 STL 포맷 파일이 표준 입력 데이터로 널리 사용되고 있다. 이러한 STL 파일은 저장 방식에 따라 ASCII 포맷과 Binary 포맷 두 종류가 있다. Binary 포맷은 ASCII 포맷에 비해 파일 크기가 작고, 변환 속도가 빠르기 때문에 많이 사용되고 있다. Fig. 2는 STL 파일의 ASCII 포맷을 나타낸 것이고, Table 2에서는 STL 파일의 Binary 포맷을 보여주고 있다. 헤더 레코드(header record)는 84byte로 이루어져 있다. 처음 80byte는 작성자 이름, 설명문 등의 정보를 담고 있고, 마지막 4byte는 삼각형 facet의 수를 나타낸다. 다음부터 하나의 삼각형 facet을 표현하기 위해 50byte를 사용한다. 여기서 법선벡터와 정점의 x, y, z값을 표현하기 위해 48byte만이 사용되고, 나머지 2byte는 사용되지 않는다. 따라서 STL 파일의 Binary 포맷 크기는 다음식과 같이 구할 수 있다.

$$STL\ \ file\ \ size = (Number\ \ of\ \ facet) \times 50 + 80$$

위와 같이 CAD 모델로부터 ASCII 또는 Binary 포맷의 STL 데이터로 변환할 때 문제점은 CAD 데이터로부터 얻어지는 각 곡면 단위에서 삼각형 facet을 생성할 때 자유곡면의 접합부에서 하나의 모서리를 두 개 이상의 facet이 공유하여 중복되거나, facet이 누락되어 구멍이 생기는 현상이 발생한다. 이런 경우의 STL 파일에서는 정점과 삼각형 facet의 연결을 나타내는 정확한 위상정보를 가지지 못하기 때문에 단면정보의 도출에서 문제가 발생한다.

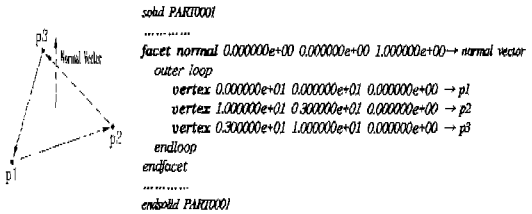

Fig. 2 Typical ASCII format of STL file

Table 2 Typical Binary format of STL file

Byte	Type	Description
80	string	Head information such as the CAD system used
4	unsigned long integer	Number of facets
	First Triangle Definition	
4	float	normal x
4	float	normal y
4	float	normal z
4	float	vertex1 x
4	float	vertex1 y
4	float	vertex1 z
4	float	vertex2 x
4	float	vertex2 y
4	float	vertex2 z
4	float	vertex3 x
4	float	vertex3 y
4	float	vertex3 z
2	unsigned integer	Number of attributes bytes should be set to zero
	Second Triangle Definition	
4	float	normal x
4	float	normal y
4	float	normal z
4	float	vertex1 x
4	float	vertex1 y
4	float	vertex1 z
4	float	vertex2 x
4	float	vertex2 y
4	float	vertex2 z
4	float	vertex3 x
4	float	vertex3 y
4	float	vertex3 z
2	unsigned integer	Number of attributes bytes should be set to zero
	:	
	:	

Fig. 3은 완벽한 STL 포맷의 삼각형 facet을 보여주고 있으며, Fig. 4에서는 STL 포맷의 중요한 오류인 구멍 오류와 중복 오류가 생긴 삼각형 facet을 나타내고 있다.

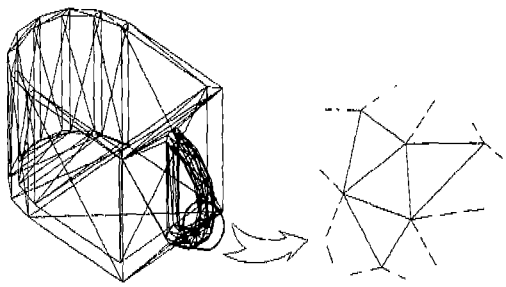

Fig. 3 Correct facets

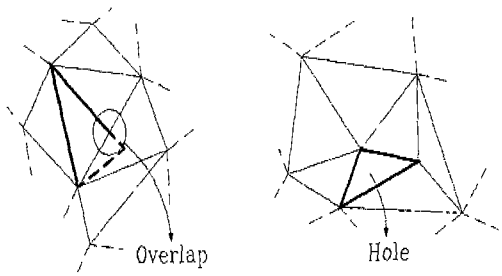

Fig. 4 Incorrect facets

3.2 Facet 및 Edge 구조체 다중 연결 리스트 구성

입력 STL 파일을 다음과 같은 단계를 거쳐 Fig. 5와 6에서와 같은 구조체를 다중 리스트로 구성한다.

(step1) STL 파일을 읽어서 최초의 삼각형 facet 데이터를 Fig. 5의 Facet 구조체의 normal, vertex에 할당한다.

(step2) 최초의 vertex값으로 Fig. 6의 Edge 구조체의 edge의 두 점을 나타내는 p1과 p2에 할당하고, 이 edge를 공유하는 삼각형의 개수를 가리키는 count에 1을 대입한 후 위의 facet no.를 리스트로 연결한다.

(step3) 이상과 같이 최초의 삼각형 facet 데이터로 세 개의 edge 구조체가 만들어지면, Fig. 5의 Faeet 구조체의 edge no.에 (step2)에서 생성된 edge 번호를 대입한다.

(step4) 두 번째 이후부터 읽어들여지는 삼각형 facet

데이터는 (step1)과 같은 방법으로 Facet 구조체를 만든다.

(step5) edge 번호가 3 즉, 4번째 Edge 구조체를 작성할 때는 현재의 edge를 구성하는 두 점과 같은 이 전의 edge가 있는가를 검사하여 있으면 해당 edge 구조체의 count에 1을 더하고, 리스트 끝에 (step5)에서의 Facet 번호를 추가하고 포인터는 NULL을 가리키도록 한다.

(step6) (step5)에서의 세 개의 edge 번호를 Facet 구조체의 edge no.에 차례로 할당한다.

(step7) (삼각형 facet 개수 - 1)만큼 (step4)에서 (step6)까지 반복

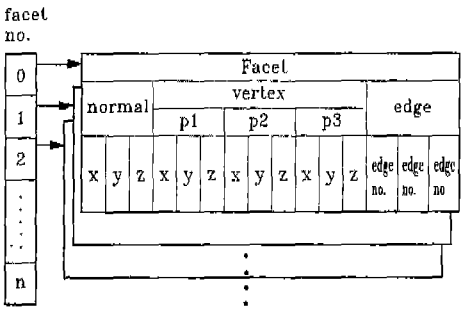

Fig. 5 List of Facet structure

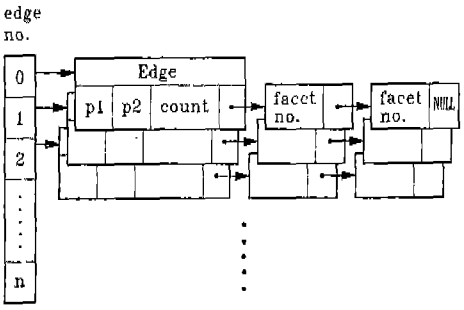

Fig. 6 List of Edge structure

3.3 삼각형 facet의 모서리, 중복 및 구멍 오류 검증

오류가 없는 완전한 STL 파일이 되기 위해서는 다음과 같은 조건을 만족하여야 하며, facet의 오류 검증 알고리즘에 반영한다. 이를 위해서 Fig. 5와 6에서와 같은 구조체 다중 리스트를 이용하여 검색한다.

(1) 체적을 가진 CAD 모델을 닫혀진 삼각형 facet으로 근사시키기 때문에 모든 삼각형 facet은 세 모서리에 정확히 3개의 이웃한 facet이 존재하여야 한다.

(2) 하나의 facet이 다른 facet과 이웃하기 위해서는 두 facet이 공유하는 모서리의 정점의 x, y, z 좌표값이 같아야 한다.

(3) 하나의 facet은 세 정점을 가진다. 따라서 두 정점이 같은 경우(line)와 세 정점이 같은 경우(point)는 Facet list에서 제거한다.

(4) 모든 facet의 정점들은 오른손 법칙을 따라 파트의 바깥쪽에서 보았을 때 반시계 방향으로 배열된다.

(5) facet의 법선벡터를 검사함으로써 facet의 안쪽과 바깥쪽을 결정한다.

STL 포맷 변환시 반올림 오차로 인해 두 이웃 삼각형이 접해야 할 모서리를 이루는 두 점 $(v1-v2)$의 좌표값이 정확히 일치하지 않는 경우가 생긴다. 이러한 문제점을 해결하기 위해서 허용공차(ε)를 사용자 입력으로 받아서 $|v1-v2| \le \varepsilon$ 인 경우만 같은 꼭지점으로 취급한다. 본 연구에서는 ε의 내정값으로 0.01㎜를 사용하였다.

3.3.1 삼각형 facet의 모서리 오류가 존재하는 경우

Fig. 5의 Facet 구조체 멤버인 vertex의 p1, p2, p3 중 x, y, z좌표값이 같은 꼭지점이 두 개 이상 존재할 경우, 해당 Facet과 Edge 구조체를 다중 리스트에서 삭제한다.

각 facet의 세 모서리에 이웃한 facet이 모두 존재하지 않을 경우, 먼저 이러한 facet을 찾기 위해 Fig. 6의 Edge 구조체 멤버인 count(모서리를 공유하고 있는 이웃 facet의 개수) 값이 1인 facet no.에 해당하는 Facet 구조체를 찾는다. 이 Facet 구조체 멤버인 edge의 나머지 두 edge no.가 가리키는 Edge 구조체 멤버중 count 값이 모두 1인 경우 이 Facet과 Edge 구조체를 다중 리스트에서 삭제한다.

3.3.2 삼각형 facet 사이에 중복이 존재하는 경우

Fig. 6의 Edge 구조체 멤버인 count 값이 3이상인 edge no.를 검색하여 해당 리스트에 연결된 facet no.를 모두 찾는다.

위와 같이 찾은 facet no.가 가리키는 Facet 구조체의 나머지 두 edge no.를 검색한 후 count값이 모두 1인 Facet과 Edge 구조체를 리스트에서 제거함으로써 삼각형 facet의 중복 오류를 수정한다.

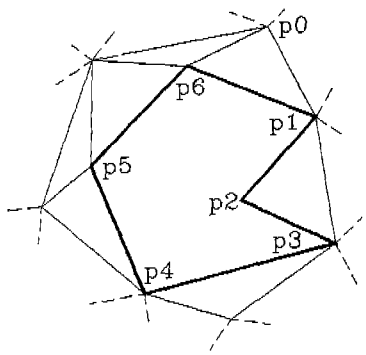

Fig. 7 An example of hole error

3.3.3 삼각형 facet 사이에 구멍이 존재하는 경우

삼각형 facet의 모서리 오류와 삼각형 facet 사이에 중복 오류를 고치고 나면 Edge 리스트의 count 값이 대부분이 2로 설정된다. 만약 위 두 단계를 거치고도 count 값이 1이 남아 있으면 구멍오류가 존재하는 경우로 판정하여 count 값이 1인 모서리만을 추출하여 구멍 다각형 데이터를 찾게 된다. Fig. 7에서와 같은 다수의 삼각형 facet이 빠진 다각형 구멍을 채우는 알고리즘은 다음과 같은 단계를 거친다.

(step1) 구멍의 다각형 데이터(p1~p6)를 구한다.

(step2) 다각형 데이터를 xy, xz, yz 평면중 투영면을 결정한다.
- p1~p6의 모든 x값이 같으면 yz평면에 투영
- p1~p6의 모든 y값이 같으면 xz평면에 투영
- 그외는 모두 xy평면에 투영

(step3) 다음과 같은 순서로 Delaunay 삼각형 분할을 시작한다.
- 다각형 데이터중 세 점을 선택한다.
- 선택된 세 점으로부터 외접원의 중심 즉, 외심을 구한다.
- 외접원의 반지름을 Ro로 둔다.
- 외심과 경계데이터의 점과의 거리, r[i] (i=0,1,2 ..., n)중 최소값을 R_{min}로 둔다.
- Ro 〈 R_{min}을 만족하면 (step4)로 가고 아니면, 다른 세 점을 선택한다.

위와 같이 주어진 세 점의 삼각형에 대해서 그 외접원의 내부에 (step1)에서 구한점(P1~P6)을 포함하지 않을 경우, Delaunay 삼각형 분할의 조건을 만족한다.

Fig. 8에서는 Fig. 7의 다각형 구멍을 예로 들어서 주어진 세 점에 대한 Delaunay 삼각형 분할 조건의 만족 여부를 보여준다.

(step4) 선택된 세 점이 이루는 삼각형이 구멍의 다각형 내부에 존재하는가를 검사한다.
- 선택된 세 점으로부터 내접원의 중심 즉, 내심을 구한다.
- 구한 내심이 다각형 내부에 있으면 (step5)로 가고 아니면, (step3)으로 간다.

구멍 오류의 다각형 내부에 내심의 존재 여부는 다음과 같은 내부, 외부 판정 알고리즘에 의해 결정할 수 있다.

1) 내심의 연장선이 다각형의 변과 만나는 경우

Fig. 9와 같이 주어진 내심에서 오른쪽으로 수평 연장선을 그릴 때 다각형의 변과 만나는 점의 개수가 홀수개이면 다각형의 내부, 짝수개이면 다각형의 외부에 내심이 존재하는 것으로 판정한다.

2) 내심의 연장선이 다각형의 꼭지점과 만나는 경우

Fig. 10과 같이 내심(C1)의 연장선이 다각형의 꼭지점을 지날 경우, 1)과 같은 방법으로는 P1이 E1, E2, E5와 만나기 때문에 교점의 개수가 3이 되어 내심 C1은 내부로 잘못 판정된다. 이를 해결하기 위해 연장선이 다각형의 꼭지점과 만나는 경우 연장선의 y좌표를 y_e, 해당 다각형 변의 두 꼭지점중 y좌표값이 작은 점의 y좌표를 y_{min}이라 하면 $y_e = y_{min}$인 변에 대해서만 개수를 세면된다. 따라서 P1은 E1에 대해서만 교점의 개수를 세기 때문에 2가 되어 외부로 판정하게 된다.

3) 내심의 연장선이 다각형의 변과 일치하는 경우

Fig. 11과 같이 연장선 안에 다각형의 변이 포함되는 경우, 2)와 같은 방법으로는 내심 C2는 변 E8와 E9와 만나는 것으로 되어서 교점의 개수가 2가 되어 외부의 점으로 잘못 판정된다. 이를 해결하기 위해 연장선 안에 다각형의 변이 포함되는 경우는 교점의 개수를 세지 않으면 된다. 따라서 내심 C1과 C2의 연장선과의 교점의 개수가 1이 되어서 다각형 내부의 점으로 판정한다.

(step5) 선택된 세 점의 배열과 법선벡터를 결정한다.

- 선택된 세 점의 2가지 배열(p1→p2→p6, p2→p1→p6)을 결정하기 위해서 이웃 삼각형 facet의 배열을 고려하여야 한다.
- 이웃 삼각형 facet의 배열이 p0→p6→p1이면 구멍 오류를 채울 삼각형 facet의 세 정점은 p2→p1→p6 순서로 배열되어야만 피비우스 띠와 같은 오류를 방지할 수 있다.
- 삼각형 facet의 세 정점의 배열 순서가 결정되면 3.4절에서의 식(1)~(8)에 의해 법선벡터를 구한다.
- 이상과 같이 구한 법선벡터와 세 정점을 Facet 구조체 리스트에 추가한다.

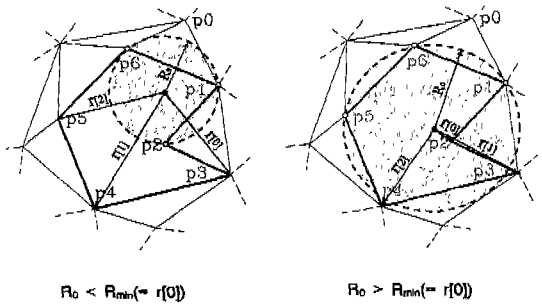

Fig. 8 The application of the Delaunay triangulation method

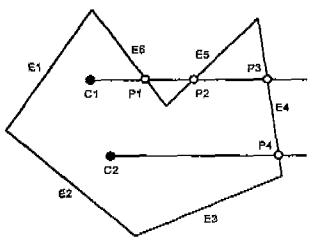

Fig. 9 Intersection of extension line and edge

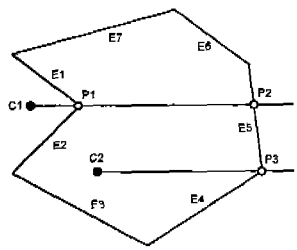

Fig. 10 Intersection of extension line and vertex

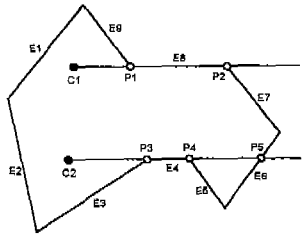

Fig. 11 Coincidence of extension line segment and edge

3.4 Facet의 법선벡터 방향 및 크기 검증

초기 입력 STL 파일에서 인접 facet의 법선벡터 크기와 방향이 각각 1과 오른손 법칙을 만족하더라도 피비우스 띠와 같이 인접 facet의 꼭지점이 배열되어 있는 문제점을 해결하기 위해 먼저 Facet과 Edge 리스트를 통해 인접 facet의 꼭지점 배열 순서를 검사한 후에 다음과 같은 과정을 거친다.

3점을 지나는 평면 방정식

한 직선위에 있지 않는 3점 $p_0(x_0,y_0,z_0)$, $p_1(x_1,y_1,z_1)$, $p_2(x_2,y_2,z_2)$를 지나는 평면 방정식은 (1)식과 같다.

$$\begin{vmatrix} x-x_0 & y-y_0 & z-z_0 \\ x_1-x_0 & y_1-y_0 & z_1-z_0 \\ x_2-x_0 & y_2-y_0 & z_2-z_0 \end{vmatrix} = 0 \qquad (1)$$

따라서, (1)식을 (2)식과 같은 법선벡터 a, b, c로 나타낸 일반 평면 방정식 형태로 전개하면 식(3), (4), (5)와 같이 법선벡터 a, b, c를 주어진 세 점의 x, y, z 좌표값으로 구할 수 있다.

$$ax + by + cz + d = 0 \qquad (2)$$

$$a = \{(y_1 - y_0)(z_2 - z_0) - (z_1 - z_0)(y_2 - y_0)\} \qquad (3)$$

$$b = \{(z_1 - z_0)(x_2 - x_0) - (x_1 - x_0)(z_2 - z_0)\} \qquad (4)$$

$$c = \{(x_1 - x_0)(y_2 - y_0) - (y_1 - y_0)(x_2 - x_0)\} \qquad (5)$$

식(3), (4), (5)에서 구한 법선벡터 성분을 식(6), (7)과 같은 과정을 거쳐 식(8)에서와 같이 크기가 1인 단위 법선벡터 성분들을 구할 수 있다.

$$length = \sqrt{a^2 + b^2 + c^2} \qquad (6)$$

$$factor = 1.0 / length \qquad (7)$$

$$\begin{Bmatrix} n_x = a \times factor \\ n_y = b \times factor \\ n_z = c \times factor \end{Bmatrix} \quad (8)$$

Fig. 5의 Facet 구조체 멤버인 normal과 vertex를 위의 식에 적용하여 삼각형 facet의 법선벡터 방향과 크기의 오류를 찾아내고 수정한다. 이를 위해 삼각형 facet 개수만큼 다음과 같은 단계를 거친다.

(step1) Facet 구조체 멤버인 vertex를 이루는 세 점인 *p1, p2, p3*를 식(1)에서 식(8)까지 과정을 통해 단위 법선벡터 n_x, n_y, n_z를 구한다.

(step2) n_x, n_y, n_z와 Facet 구조체 멤버인 normal의 x, y, z와 비교하여 같으면 법선벡터의 오류가 없는 것으로 판정하여 (step1)로 간다. 다르면 다음 단계로 간다.

(step3) Facet 구조체 멤버인 normal의 x, y, z를 식(6)에서 식(8)을 거쳐 단위 법선벡터 n_x', n_y', n_z'를 만든다. 이를 n_x, n_y, n_z와 비교하여 같으면 법선벡터의 크기 오류로 판정하여 n_x, n_y, n_z를 Facet 구조체의 normal에 대입하고 (step1)로 간다. 다르면 다음 단계로 간다.

(step4) n_x', n_y', n_z'에 -1.0을 곱한 값이 n_x, n_y, n_z과 같으면 법선벡터의 방향 오류로 판정하여 n_x, n_y, n_z를 Facet 구조체의 normal에 대입하고 (step1)로 간다. 다르면 Facet 구조체의 normal이 알 수 없는 오류로 판정하여 n_x, n_y, n_z를 Facet 구조체의 normal에 대입한다.

4. 적용 예

본 연구에서는 펜티엄 PC의 Windows 95 환경에서 Visual C++ 4.0 컴파일러를 사용하여 STL 포맷 오류 검증 알고리즘을 프로그램화하였고, 가시화 프로그램은 펜티엄 마이크로프로세서의 부동 소수점 연산을 최적화 시켜 객체를 렌더링한 상태로 회전과 이동이 PC 환경에서 실시간으로 이루어지도록 한 Intel사의 3차원 렌더링 라이브러리인 3DR을 사용하여 작성하였다.

Fig. 12에서는 총 facet수 1,320개인 솔리드 형상의 초기 STL 파일을 렌더링하여 오류 부분을 보여주고 있다. Fig. 13은 Fig. 12의 STL 포맷 오류를 검사하고, 수정한 STL 파일을 가시화한 것이다. Fig. 14에서는 총 facet수 16,554개인 Fan 형상의 초기 STL 파일의 오류 부분을 가시화 한 것이다. Fig. 15는 Fig. 14의 STL 포맷 오류를 검사하고, 수정한 STL 파일을 가시화한 것이다.

위 두 STL 포맷 파일에 대한 오류 검증 결과는 다음과 같다.

첫번째, 1,320개의 facet에 대한 오류 검증 시간은 약 3초가 소요되었으며, 1개의 중복오류와 2개의 구멍오류가 검증되었다.

두번째, 16,554개의 facet에 대한 오류 검증 시간은 약 17초가 소요되었으며, 3개의 중복오류와 2개의 구멍오류가 검증되었다.

이상의 적용예의 결과에서와 같이 오류가 있는 STL 포맷 파일을 본 시스템을 통해 빠른 시간내에 정확히 수정하는 것을 확인하였다.

Fig. 12 Example(I) of STL format including error

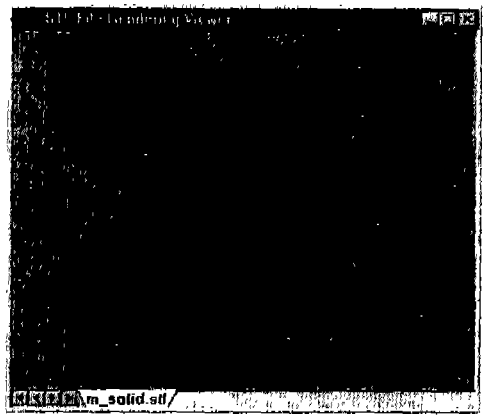

Fig. 13 Example(I) of STL format after verification

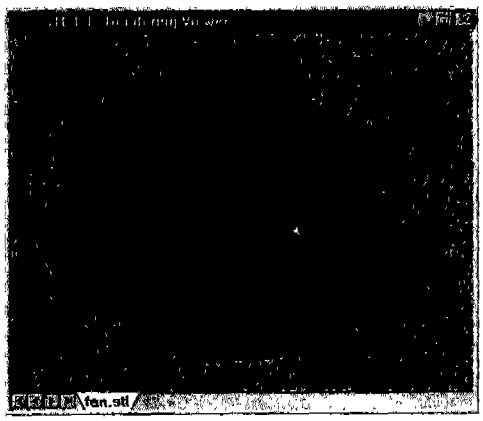

Fig. 14 Example(II) of STL format including error

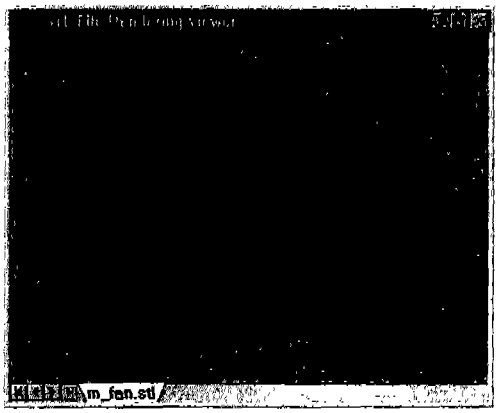

Fig. 15 Example(II) of STL format after verification

5. 결 론

본 연구에서는 일관되고 효율적인 급속조형 공정이 이루어지기 위해서 급속조형시스템의 산업 표준 데이터인 STL 포맷의 오류를 사전에 찾아내고 이를 수정하는 시스템을 개발하였다.

STL 포맷의 중요한 오류로서 하나의 facet을 이루는 세 정점중 두 개 이상의 정점이 같은 경우, 각 facet의 세 모서리를 공유하는 이웃한 facet이 존재하지 않는 경우, 삼각형 메쉬 사이에 facet 데이터가 빠져 구멍이 생긴 경우, facet의 법선벡터 방향과 크기가 잘못된 경우를 고려하였다.

본 연구의 주요 특색은 다음과 같다.

(1) 본 프로그램에서는 STL 파일의 ASCII 포맷 뿐만 아니라 ASCII 포맷 보다 파일크기가 작고 변환속도가 빠르기 때문에 많이 사용되는 Binary 포맷도 입출력할 수 있도록 하였다.

(2) Facet과 Edge 구조체 리스트를 다중으로 구성함으로써 facet의 오류를 신속하고 효율적으로 검색, 수정할 수 있다.

(3) Edge 구조체 리스트에서 edge를 공유하는 삼각형 facet 수에 의해 간단히 중복 오류를 검색하여 수정할 수 있다.

(4) Delaunay 삼각형 분할법을 이용하여 삼각형 facet 사이의 구멍을 채움으로써 구멍 오류를 수정할 수 있다. Delaunay 삼각형 분할법은 동변에 가까운 삼각형을 생성하기 때문에 이후의 슬라이스 공정에서 직선 세그먼트의 길이를 일정하게 할 수 있는 잇점이 있다.

(5) 삼각형 facet의 법선벡터 방향과 크기의 오류를 검증함으로써 이후의 지지대 구조를 생성 할 때 신뢰성 있는 STL 데이터를 제공할 수 있다.

향후 연구과제로서는, 앞으로 CAD 시스템간의 데이터 교환을 위한 국제적 표준이 될 STEP과 급속조형의 산업 표준인 STL 포맷 사이의 상호 데이터 호환에 관한 연구가 필요하다.

참 고 문 헌

1. Rapid Prototyping Report, CAD/CAM Publishing, Inc., pp.4~6, January 1992.
2. Imageware's Rapid Prototyping Module(RPM), http://boojum.iware.com/htmls/pro_desc.html.
3. DeskArtes Rapid Tools, http://www.sgi.com/Products/appsdirectory.dir/Applications/Mechanical_CAD/ApplicationNumber6181.html.
4. 田中文基, 岸浪建史, "光造形法における問題點とその解決法", 第6回 光造形システムシンポジウム, pp.39~45, 1994.
5. M.J.Wozny, "DATA DRIVEN SOLID FREEFORM FABRICATION", IFIP Transactions B-3: Human Aspects in Computer Integrated Manufacturing, pp.71~82, 1992.
6. A.Dolenc, I.Mäkelä, R.Hovtun, "Better Software for Rapid Prototyping with INSTANT-

CAM", IFIP Transactions B-3: Human Aspects in Computer Integrated Manufacturing, pp.449~456, 1992.

7. P. Vuyyuru, C.F.Kirschman, G.Fadel, A.Bagchi, C.C.Jara-Almonte, "A NURBS-Based Approach for Rapid Product Realization", Proceeding of the Fifth international Conference on Rapid Prototyping, Dayton, Ohio, pp.229~238, 1995.
8. 김준안, 홍삼열, 백인환, "광조형법에 있어서 OFF-SET정보생성 알고리즘개발에 관한 연구", 대한산업공학회 추계학술대회 발표논문집, pp.77~81, 1995.
9. Pro/ENGINEER Interface Guide for Release 11.0, Parametric Technology Corporation, 1993.
10. 3DR Version 2.1 Rasterizing Engine Programming Manual, Intel Co., 1996.
11. 3DR Version 2.1 Graphics Pipeline Programming Manual, Intel Co., 1996.