韓國精密工學會 '96年度 春季學術大會論文集 pp.597~601

급속조형시스템을 위한 STL 포맷의 오류 검증에 관한 연구

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

최 홍 태*, 이 석 희**
Hong-Tae Choi*, Seok-Hee Lee**

* 부산대학교 생산기계공학과 대학원
** 부산대학교 생산기계공학과 기계기술연구소

ABSTRACT

Nowadays, the STL format, industrial standard data, which approximates three dimensional CAD model to triangular facets, is used for RP(Rapid Prototyping) system. Because most RP machine is accepted to only two dimensional line segments, but some STL translators are sometimes poorly implemented. The error verifying process is as follows. 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) Repair wrong direction and value of 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(법선벡터), overlapping error(중복오류), Delaunay triangulation(Delaunay 삼각화)

1. 서 론

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

현재 대부분의 상용 CAD 시스템이 STL 포맷을 지원하고 있으며, IGES, VDAFS, STEP, PDES 등과 같은 각종 CAD 교환 데이터를 STL 파일로 변환시키는 소프트웨어들이 개발되고 있다. 이러한 STL 변환기들은 대부분 CAD 판매자와 서드파티 그룹에서 제공하고 있지만, CAD 모델에서 변환된 STL 포맷에서 facet 데이터가 빠져 구멍이 생긴 오류와 하나의 facet에 이웃한 facet이 두개 이상 존재하는 중복 오류 등이 발생하는 문제점이 지적되어 왔다.

Brock Rooney는 보다 신뢰성 있는 STL 포맷 출력을 위하여 IGES를 STL로 변환시키는 STL 변환기를 개발하여 판매하고 있다. Rooney의 STL 변환기는 서피스모델 변환에 효과적이며 삼각형 facet이 누락되어 있거나 두 이웃한 facet 사이의 틈과 같은 오류가 있는 STL 파일도 수정할 수 있는 특징을 가지고 있다[1]. C-TAD 시스템에서는 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 제공하지만 5천불에서 1만불로 비교적 고가이다[1].

표면 정도가 좋은 급속조형물을 얻기 위한 가장 좋은 방법은 정확한 곡면 수학식으로 파트를 표현하는 것이다. 그러나 대부분의 급속조형시스템은 2차원 직선 세그먼트(line segment)만을 받아들이기 때문에 이상과 같은 STL 포맷의 문제점에도 불구하고, CAD 모델을 삼각형 facet들로 구성된 STL 파일을 산업 표준 입력파일로 사용하고 있다.

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

2. 관련연구

국외에서는 Fumiki Tanaka[2]등이 STL 파일에서 얻어진 점군 데이터를 이용하여 STL 포맷의 두 가지 오류 즉, 삼각형 패치 사이에 facet 데이터가 빠져있는 경우와 하나의 edge에 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, 법선벡터의 오류에 관한 검증은 고려하지 않았다.

M.J.Wozny[3]등은 기존의 STL 파일의 문제점을 지적하고

이를 개선하기 위한 STL 포맷의 중복된 정점을 줄이기 위해 위상정보가 부족한 STL 파일을 기초로 하여 정점, 모서리, 그리고 면의 색인 리스트(index list)를 가진 facet solid entity로 표현된 RPI 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[4]등은 기존의 CAD 데이터에서 STL 포맷으로의 변환의 결과가 만족스럽지 못하기 때문에 IGES 파일을 와이어프레임 모델의 VDAFS 포맷으로 변환하고, 이를 Facetted Representation방식의 STL 포맷으로의 변환 절차를 제시하였지만, 최종적으로 생성된 STL 파일의 신뢰성을 확인하기 위한 오류 검증에 관한 부분은 언급하지 않고 있다.

P. Vuyyuru[5]등은 기존의 STL파일은 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 후 공정이 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[6]등이 STL파일을 z축으로 일정한 간격만큼 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 레이저 빔의 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 검사해야 하는 문제점이 있다.

허정훈[7]등은 SLA를 이용한 시작작업의 효율화 방안으로 최적의 성형방향을 찾기 위한 알고리즘으로 성형 정확도 향상을 위해, 계단효과에 의해 생성된 staircase 면적을 최소화하는 목적함수를 무차원화하였으며, 성형시간 단축을 위해 가변층 두께 개념을 도입하였고, 그리고 지지대의 최소화를 위해 각 facet의 법선벡터의 z값이 일정한 범위내의 음인 facet의 투영면적과 facet의 세꼭지점이 모두 0인 facet의 면적비로 목적함수를 나타내었다. 그러나 최적의 성형 방향을 결정하기 위해 입력되는 STL 파일의 오류 검증에 관해서는 언급하지 않았다.

3. 본 론

다수의 CAD 시스템에서 지원하고 있는 STL 변환기들이 만족할만한 결과를 주지 못하기 때문에 급속조형에 필요한 파트의 최적자세 결정, 지지대 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.

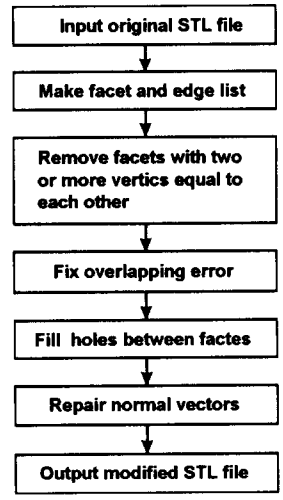

Fig. 1 Error Verifying Process of STL format

3.1 STL 포맷의 문제점

CAD 시스템에서 입체형상을 모델링하고, 그 형상 데이터를 기초로 해서 급속조형기술에 의해 시작품을 얻게된다. 대부분 CAD 시스템을 이용해서 모델화한 입체 형상은 일반적으로 자유곡면이 많다. 이러한 형상 데이터를 이용해서 수평방향으로 등간격으로 절단시킨 단면 데이터를 계산하기 위해서는 곡면을 다수의 삼각형 평면으로 근사하는 방법이 이용되고 있다. 이러한 삼각형을 이용한 데이터로써 급속조형기술에서 표준 입력 데이터로 Fig. 2와 같은 STL 파일이 널리 사용되고 있다. 그러나 CAD 모델로부터 STL 데이터로 변환할 때 문제점은 CAD 데이터로부터 얻어지는 각 곡면 단위에서 삼각형 facet을 생성할 때 자유곡면의 접합부에서 facet이 누락되어 구멍(hole)이 생기거나, 하나의 모서리(edge)를 두 개 이상의 facet이 공유하여 중복(overlapping)되는 현상이 발생한다. 이런 경우의 STL 파일에서는 정점(vertex)과 삼각형 facet의 연결을 나타내는 정확한 위상정보를 가지지 못하기 때문에 단면 정보의 도출에서 문제가 발생한다.

Fig. 3은 완벽한 STL 포맷의 삼각형 facet을 보여주고 있으며, Fig. 4에서는 STL 포맷의 중요한 오류인 구멍오류와 중복오류가 생긴 삼각형 facet을 나타내고 있다.

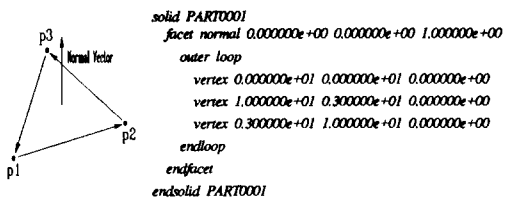

Fig. 2 Structure of STL file

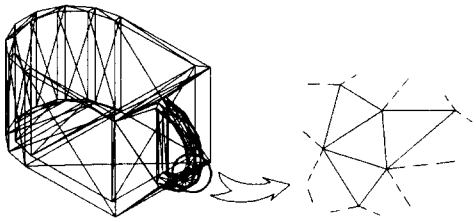

Fig. 3 Correct Facets

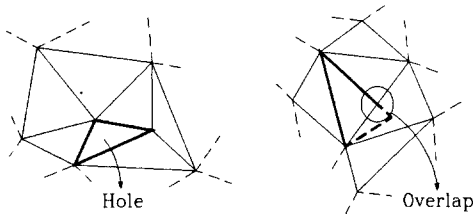

Fig. 4 Incorrect Facets

3.2 삼각형 facet의 edge, 중복 및 구멍오류 검증

오류가 없는 완전한 STL 파일이 되기 위해서는 다음과 같은 조건을 만족하여야 하며, facet의 오류 검증 알고리즘에 반영한다. 이를 위해서 입력 STL 파일을 Fig. 5와 6에서와 같은 구조체 리스트로 구성하였다.

(1) 체적을 가진 CAD 모델을 닫혀진 삼각형 facet으로 근사시키기 때문에 매쉬안의 모든 facet은 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.

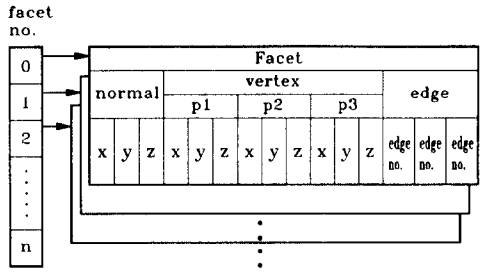

Fig. 5 List of Facet Structure

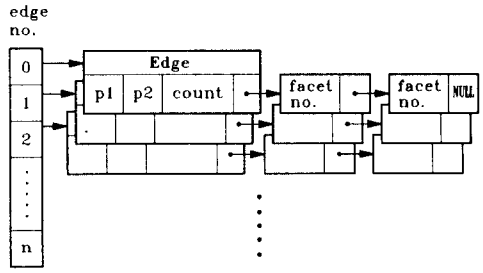

Fig. 6 List of Edge Structure

삼각형 facet의 edge 오류가 존재하는 경우

Fig. 5의 Facet 구조체 멤버인 vertex의 p1, p2, p3중 두 점 이상이 같은 경우, 해당 Facet과 Edge 구조체를 리스트에서 삭제한다.

각 facet의 세 edge를 공유하는 이웃한 facet이 존재하지 않을 경우, Fig. 6의 Edge 구조체 멤버인 count(edge를 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 리스트에서 삭제한다.

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 고친다.

삼각형 facet 사이에 구멍이 존재하는 경우

Fig. 7에서와 같은 다수의 삼각형 facet이 빠진 다각형 구멍을 채우는 알고리즘은 다음과 같은 단계를 거친다.

[step1] 구멍의 경계 데이터(p1~p6)를 구한다.

[step2] 경계 데이터를 xy, xz, yz 평면중 투영면을 결정한다.

- p1~p6의 x값이 모두 같으면 yz 평면에 투영.
- p1~p6의 y값이 모두 같으면 xz 평면에 투영.

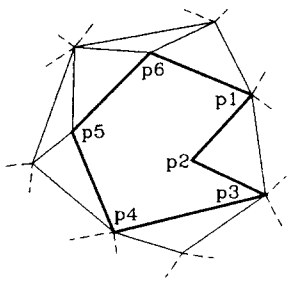

Fig. 7 Example of Hole Error

- 그 외는 모두 xy 평면에 투영.

[step3] Delaunay 삼각형 분할을 시작한다.

- 경계 데이터 점중 세 점을 선택한다.
- 선택된 세 점으로부터 외접원의 중심 즉, 외심을 구한다.
- 외접원의 반지름을 Ro로 둔다.
- 외심과 경계데이터의 점(p1~p6)과의 거리, R[i] (i=0,1,2 ..., n)중 최소값을 R_{min}으로 둔다.
- Ro < R_{min}을 만족하면 [step4]로 가고 아니면, [step3]을 반복한다.

[step4] 선택된 세 점이 이루는 삼각형이 구멍의 경계 데이터 내부에 존재하는가를 검사한다.

- 선택된 세 점으로부터 내접원의 중심 즉, 내심을 구한다.
- 구한 내심이 경계 데이터 내부에 있으면 [step5]로 가고 아니면, [step3]으로 간다.

[step5] 선택된 세 점의 배열과 법선벡터를 결정한다.

- 선택된 세 점의 2가지 배열(p1→p2→p3, p1→p3→p2)에 대한 법선벡터 n_1^1, n_1^2를 3.3에서와 같은 방법으로 구한다.
- 이웃 삼각형 facet의 법선벡터 n_2와 n_1^1, n_1^2의 내적에 의한 사이각 θ_1, θ_2를 구한다.
- 사이각 θ_1, θ_2중 작은쪽의 법선벡터와 세정점을 facet 리스트에 더한다.

3.3 삼각형 facet의 법선벡터 방향 및 크기 검증

3점을 지나는 평면 방정식

한 직선 위에 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.

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

따라서, (1)식을 (2)식과 같은 법선벡터 a, b, c로 나타낸 일반 평면방정식 형태로 전개하면 식(3), (4), (5)와 같이 법선벡터 a, b, c를 주어진 세 점 P_0, P_1, P_2의 x, y, z 좌표값으로 구할 수 있다.

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

$$a = \{(y_1-y_0)(z_2-z_0) - (z_1-z_0)(y_2-y_0)\} \quad \cdots\cdots (3)$$
$$b = \{(z_1-z_0)(x_2-x_0) - (x_1-x_0)(z_2-z_0)\} \quad \cdots\cdots (4)$$
$$c = \{(x_1-x_0)(y_2-y_0) - (y_1-y_0)(x_2-x_0)\} \quad \cdots\cdots (5)$$

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.

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

$$factor = 1.0 / length \quad \cdots\cdots\cdots\cdots (7)$$

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

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, p3를 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, Intel사의 3차원 렌더링 라이브러리인 3DR을 사용하여 가시화 프로그램을 작성하였다.

Fig. 8에서는 구멍오류와 중복오류가 있는 STL 파일을 렌더링 하여 오류 부분을 직접 확인할 수 있다. Fig. 9는 오류를 검사하고 수정한 STL 파일을 가시화한 것이다.

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 정점 중 두 개 이상의 정점이 같은 경우, 각 facet의 세 모서리를 공유하는 이웃한 facet이 존재하지 않을 경우, 삼각형 메쉬 사이에 facet 데이터가 빠져 구멍이 생긴 경우, facet의 법선벡터 방향과 크기가 잘못된 경우를 고려하였다.

본 연구의 주요 특색은 다음과 같다.

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

(2) Edge 구조체 리스트에서 edge를 공유하는 삼각형 facet 수에 의해 간단히 중복오류를 검색하여 수정할 수 있다.

(3) Delaunay 삼각형 분할법을 이용하여 삼각형 facet 사이의 구멍을 채움으로써 구멍오류를 고칠 수 있다. Delaunay 삼각형 분할법은 정삼각형에 가까운 삼각형을 생성하기 때문에 이후의 슬라이싱 공정에서 직선 세그먼트의 길이를 일정하게 할 수 있는 이점이 있다.

(4) 삼각형 facet의 법선벡터 방향과 크기의 오류를 검증함으로써 이후의 지지대 구조 생성의 신뢰성 있는 데이터를 제공할 수 있다.

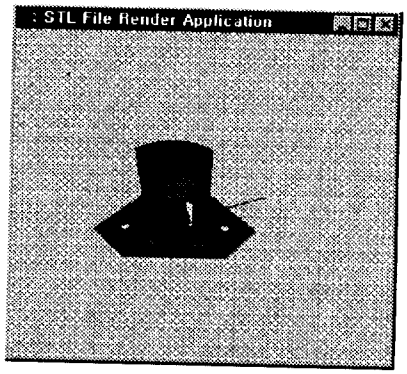

Fig. 8 STL format including error

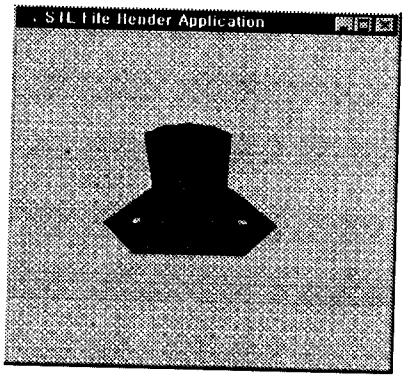

Fig. 9 STL format after verification

참고문헌

[1] Rapid Prototyping Report, pp.4～6, January 1992.

[2] 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 日本光造形システムシンポジウム, pp.39～45, 1994.

[3] M.J.Wozny, "DATA DRIVEN SOLID FREEFORM FABRICATION", IFIP Transactions B-3: Human Aspects in Computer Integrated Manufacturing, pp.71～82, 1992.

[4] A.Dolenc, I.Mäkelä, R.Hovtun, "Better Software for Rapid Prototyping with INSTANTCAM", IFIP Transactions B-3: Human Aspects in Computer Integrated Manufacturing, pp.449～456, 1992.

[5] P. Vuyyuru, C.F.Kirschman, G.Fadel, A.Bagchi, C.C.Jara-Almonte, "A NURBS-Based Aproach for Rapid Product Realization", Proceeding of the Fifth international Conference on Rapid Prototyping, Dayton, Ohio, pp.229～238, 1995.

[6] 김준안, 홍삼열, 백인환, "광조형법에 있어서 OFFSET정보 생성 알고리즘개발에 관한 연구", 대한산업공학회 추계학술대회 발표논문집, pp.77～81, 1995.

[7] 허정훈, 이건우, "SLA를 이용한 신속 시작작업에서 최적 성형방향의 결정", 한국정밀공학회지, 제13권 제4호, pp.163～173, 1996.

[8] Pro/ENGINEER Interface Guide for Release 11.0, Parametric Technology Corporation.

[9] 3DR Version 2.1 Rasterizing Engine Programming Manual, Intel Corp., 1996.

[10] 3DR Version 2.1 Graphics Pipeline Programming Manual, Intel Corp., 1996.