'88 韓國自動制御學術會議論文集 1988. 10. 21~22

복합 형상 모델링 기법의 개발

°이 강수
서울대학교 기계설계학과

이 건우
서울대학교 기계설계학과

Development of a method for modeling arbitrarily shaped body

Kang Soo Lee
Seoul National Univ.

Kunwoo Lee
Seoul National Univ.

Abstract

As an efficient way of modeling bodies of complicated shapes, the sweeping and skinning operations have been implimented. These two operations are very powerful modeling method when the body is defined by the cross sections at various locations. For the implimentation, the data structure for storing the cross sections and the resulting three dimensional body has been constructed. The resulting object is defined by the boundary representation based on the non-uniform non-periodic B-spline surface.

1. 서론

물체를 생성하는 방법에는 CGS(Constructive Geometric Solid) 모델링, 경계 모델링(Boundary modelling), 스위핑 작업(Sweeping Operation), 오일러 작업(Euler Operation), Half Space 방법[1] 등이 있으나, 이 중 자유 곡면을 가진 복잡한 형상을 모델링하기에는 스위핑 작업이 적당하다. 일반적으로 많이 사용되는 CGS 모델링으로 원하는 물체를 만들려면 여러 개의 primitive를 만든 뒤 부울리안(Boolean) 작업을 거쳐야 한다. 그러나 자유 곡면을 가진 물체의 경우 부울리안 작업을 여러 번 거치고도 원하는 형상을 모델링하기 어려운 경우가 많다. 이에 반해 스위핑 기능을 이용하면 자유 곡면을 갖는 물체를 만들 수 있고, 단면이 복잡한 물체도 보다 편리하게 만들 수 있다. 그리고 스위핑을 조금 더 확장시켜 스키닝(Skinning)을 사용하면 여러 곳에서의 단면의 형상으로 정의되는 복잡한 물체를 쉽게 모델링할 수 있다. 따라서 본 연구에서는 단면이 복잡하며 자유 곡면을 갖는 물체를 모델링하기 위한 방법으로 스위핑과 스키닝 작업을 개발을 목표로 하였다. 이러한 스위핑 기능과 스키닝 기능이 일반 CGS 모델링 기능과 결합되면 거의 모든 형상을 모델링할 수 있게 될 것이다. 이를 위해 정의된 단면 형상을 저장하기 위한 이차원 데이터 구조와 삼차원 물체를 저장하기 위한 삼차원 데이터 구조를 구축하였고, 단면 형상 및 진행 방향을 쉽게 정의할 수 있는 단면 형상 에디터를 개발하였다. 또한, 스위핑 기능과 스키닝 기능을 B-spline의 조정점의 성질을 최대로 이용하여 개발하였으며, 형성된 물체는 B-spline 곡면을 기본으로 하는 B-Rep[1]으로 표시하도록 하였다.

2. 데이터 구조의 확정

물체 형상을 취급하는 프로그램에서, 물체에 대한 정보를 저장하고 그 물체에 대한 원하는 정보를 알아낼 수 있어야 하며 이러한 정보가 각 프로그램간에 전달될 수 있어야 한다. 본 연구에서는 단면 형상 및 그 진행 path를 정의하기 위한 이차원 데이터 구조와 모델링 시스템에 의해 형성된 삼차원 물체를 정의 및 저장하기 위한 삼차원 데이터 구조를 구축하였다. 삼차원 데이터 구조는 edge에 대한 정보를 충분히 저장하는 winged-edge 데이터 구조 [2, 3]를 사용 하였다. 이 데이터 구조는 edge에 대한 정보를 충분히 저장 함으로써, edge를 매개로 body까지의 데이터를 모두 찾을 수 있는 특징이 있다. 이차원 데이터 구조는 위의 winged-edge 데이터 구조에서 이차원 개념으로 볼 때 필요없는 것을 없애고 사용하였다.

2.1 이차원 데이터 구조

이 데이터 구조는 winged-edge 데이터 구조에서 조금 변형된 것이다. Skin_Work_Set에서는 상부 구조에서 하부 구조로 데이터를 탐색할 필요는 있으나, 그 역 과정은 필요하지 않다. 이러한 불필요한 데이터에 해당하는 기억 장소를 지우면, 아래와 같은 구조가 된다.

데이터 구조는 크게 topology 데이터와 geometry 데이터를 저장할 수 있는 구조로 이루어져 있다. Topology 데이터는 형상을 이루는 profile, edge, vertex 들의 상호 연관 관계를 나타내며, geometry 데이터는 이들의 공간상의 위치를 나타낸다.

A). Topology 데이터를 저장하기 위한 데이터 구조

a. Skin_Path (3,Npath)

Skin_Path는 단면이 지나는 경로인 Path profile에 대한 정보를 저장한다.

Skin_Path (1,i) : i 번째 Skin_Path 이름이 저장되어 있는 Skin_Name 에서의 위치를 지적한다.

Skin_Path (2,i) : i 번째 Skin_Path를 구성하는 여러 개의 곡선을 저장하고 있는 Skin_Edge_Group을 지적한다.

Skin_Path (3,i) : i 번째 Skin_Path의 health를 저장한다.

b. Skin_Cross (3,Ncross)

단면을 나타내는 곡선들의 집합을 cross profile이라 하며, Skin_Cross는 이에 대한 정보를 저장한다.

Skin_Cross (1,i) : i 번째 cross profile의 이름이 저장되어 있는 Skin_Name에서의 위치를 지적한다.

Skin_Cross (2,i) : i 번째 cross profile을 구성하는 여러 개의 곡선을 저장하고 있는 Skin_Edge Group을 지적한다.

Skin_Cross (3,i) : i 번째 cross profile의 health를 저장한다.

c. Skin_Edge_Group (5,Ngroup)

Path 또는 cross profile은 여러 개의 곡선으로 이루어 지며, 이 곡선 하나 하나를 edge라 한다. 이 곡선들의 집합에 대한 정보를 저장하는 것이 Skin_Edge_Group이다.

Skin_Edge_ Group (1,i) : Path를 정의하기 위해서는 하나의 Skin_Edge_Group이면 충분하나 구멍을 갖는 cross profile을 정의하기 위해서는 여러 개의 Skin_Edge_Group이 linked list로 되어 있어 다음 Skin_Edge_Group 을 지적하도록 한다.

Skin_Edge_ Group (2,i) : i 번째 Skin_Edge_Group이 어느 path profile 또는 어느 cross profile을 정의하기 위한 것인가를 나타낸다.

Skin_Edge_ Group (3,i) : 여러개의 곡선중 임의의 하나의 곡선이 저장되어 있는 Skin_Edge에서의 위치를 지적한다. 나머지 곡선들은 Skin_Edge 안에서 linked list로 저장된다.

Skin_Edge_ Group (4,i) : i 번째 Skin_Edge_Group의 종류를 나타낸다. 즉, peripheral인가 hole인가를 나타낸다.

Skin_Edge_ Group (5,i) : i 번째 Skin_Edge_group의 health를 저장한다.

d. Skin_Edge (5,Nedge)

Path 또는 cross profile 을 구성하는 곡선에 대한 정보를 저장한다. 하나의 edge 는 시작점과 끝점에서 vertex 를 가지고 있다.

Skin_Edge (1,i) : i 번째 Skin_Edge의 시작 vertex의 번호를 저장한다.

Skin_Edge (2,i) : i 번째 Skin_Edge의 끝 vertex의 번호를 저장한다.

Skin_Edge (3,i) : 하나의 Skin_Edge_Group의 모든 Skin_Edge를 linked list로 저장하기 위해 다음 Skin_Edge를 지적한다.

Skin_Edge (4,i) : i 번째 Skin_Edge의 곡선 방정식이 저장되어 있는 Skin_Curve에서의 위치를 지적한다.

Skin_Edge (5,i) : i 번째 Skin_Edge의 health를 저장한다.

e. Skin_Vertex (2,Nvertex)

Edge 의 시작점과 끝점을 나타낸다.

Skin_Vertex (1,i) : i 번째 Skin_Vertex의 x,y,z 좌표값이 저장되어 있는 Skin_Point에서의 위치를 지적한다.

Skin_Vertex (2,i) : i 번째 Skin_Vertex의 health를 저장한다.

B). Geometry 데이터를 저장하기 위한 데이터 구조

a. Skin_Name (Nname)

모든 cross profile과 path profile의 이름을 저장한다.

b. Skin_Curve (5,Ncurve)

Skin_Curve는 모든 Skin_Edge의 곡선 방정식이 저장되는데 본 연구에서는 모든 곡선 방정식은 non-periodic non-uniform rational B-spline 으로 나타내었다. B-spline curve 를 정의하는 데는 아래와 같은 정보가 필요하며, B-spline curve 에 대해서는 [4, 5]를 참조하라.

Skin_Curve (1,i) : i 번째 Skin_Curve의 조정점의 갯수, N

Skin_Curve (2,i) : i 번째 Skin_Curve의 order, K

Skin_Curve (3,i) : i 번째 Skin_Curve가 rational B-spline 곡선인가, non-rational B-spline 곡선인가를 나타낸다.

Skin_Curve (4,i) : i 번째 Skin_Curve의 조정점 및 knot가 저장되어 있는 Skin_Point에서의 위치를 나타낸다.

Skin_Curve (5,i) : i 번째 Skin_Curve의 종류를 나타낸다.

c. Skin_Point (Npoint)

Skin_Point는 vertex의 x,y,z 좌표값, 조정점의 좌표값, 그리고 knot값을 일차원으로 저장한다.

C). WORK PROFILE DATA SET

Cross profile이 path profile상에 위치하면 cross profile이 옮겨져 새로운 profile이 생기는데, 이를 work profile 또는 instance라 하고, 다음의 기억 장소에 work profile의 집합을 기억시킨다. 이 Skin_Work_Set는 스키닝 작업을 통해 물체를 만들어 내는데 사용된다.

a. Skin_Work_Set (3,Nwork)

Skin_Work_ Set (1,i) : i 번째 Skin_Work_Set에 관련되는 path profile이 저장되어 있는 Skin_Path에서의 위치를 지적한다.

Skin_Work_ Set (2,i) : i 번째 Skin_Work_Set에 관련되는 여러개의 cross profile중 임의의 하나가 저장되어 있는 Cross_Instance에서의 위치를 지적한다. 나머지 instance들은 Cross_Instance내에 linked list로 저장된다.

Skin_Work_ Set (3,i) : i 번째 Skin_Work_Set의 health를 나타낸다.

b. Cross_Instance (2,Ninstance)

Cross_ Instance (1,i) : i 번째 instance의 원본이 저장되어 있는 Skin_Cross에서의 위치를 지적한다.

Cross_ Instance (2,i) : 하나의 Skin_Work_Set에 포함되어 있는 모든 instance를 linked list로 저장하기 위하여 다음 Cross_Instance를 지적한다.

c. Instance_Position (4,Ninstance)

Instance_ Position (1,i) : i 번째 instance의 path profile 상에서의 위치를 정의하기 위해 Skin_Path 상에서의 parameter 값을 저장한다.

Instance_ Position (2,i) : i 번째 instance의 방향을 나타내기 위해 x축을 중심으로 한 회전각을 저장한다.

Instance_ Position (3,i) : y축을 중심으로 한 회전각을 저장한다.

Instance_ Position (4,i) : z축을 중심으로 한 회전각을 저장한다.

Instance_Position의 Health는 Skin_Work_Set의 Health로 알 수 있다. Cross_Instance와 Instance_Position은 같은 순서로 기억된다. 그래서, Cross_Instance에서 Instance_Position을 지적할 필요가 없다.

Fig.1 은 스키닝에 필요한 profile을 나타낸다. (e)는 path를 나타내며, 두 개의 edge E1, E2 를 가지고 있다. (a)-(d)는 cross profile을 나타내며, 각 각 여러 개의 edge를 가지고 있다. 즉 cross

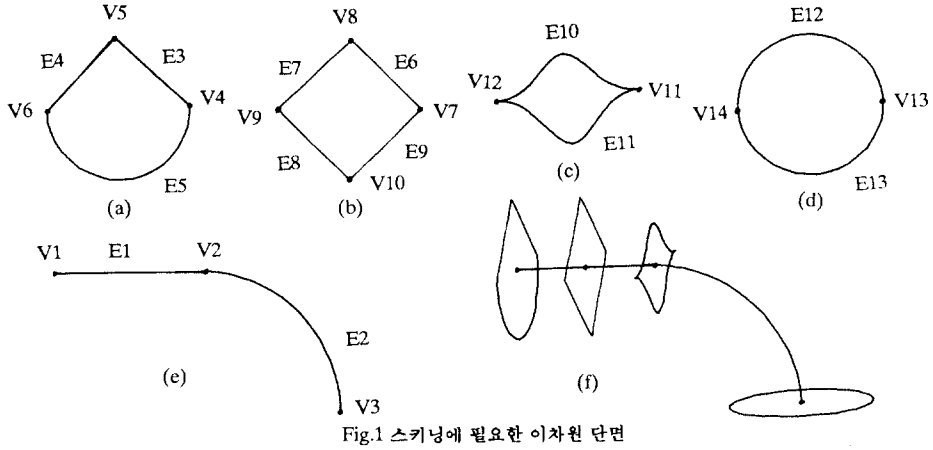

Fig.1 스키닝에 필요한 이차원 단면
(a)-(d) : Cross Profile
(e) : Path Profile
(f) : Skin_Work_Set

(a)의 Skin_Edge_Group은 E3,E4,E5로 되어 있다. (f)는 하나의 완벽한 Skin_Work_Set을 나타낸다. 이 Skin_Work_Set은 4 개의 cross를 가지고 있으며, 5장에서 설명될 스키닝 작업을 통해 원하는 단면을 가진 물체를 만드는데 사용된다.

2.2 삼차원 데이터 구조

삼차원 물체를 저장하기 위한 데이터 구조로 winged-edge 데이터 구조를 사용하였다. 이 데이터 구조는 edge에 대한 정보가 충분히 저장되어 있어, edge 정보를 매개로 다른 정보를 알 수 있다. 자세한 설명은 [2, 3]을 참고하라.

3. 단면 형상 에디터(Editor)의 개발

물체 형성 과정에서 필요한 단면을 사용자가 쉽게 정의할 수 있도록 이차원 단면 에디터를 개발하였다. 본 단면 에디터에서 다룰 수 있는 단면의 형상은 직선, 원, 원호, 스플라인으로 이루어지도록 하였다. 모든 곡선들은 B-spline 곡선 방정식 [4]으로 표시되며, 한 단면은 여러 개의 곡선 방정식으로 정의되고 그 정보는 2.1에서 설명한 이차원 데이터 구조에 저장된다. 이들 정보는 물체 형성 과정에서 사용하게 된다.

4. 스위핑 작업(Sweeping operator)의 개발

스위핑이란 주어진 path를 따라 점 또는 곡선, 면 등을 이동시켜 곡선, 면, 물체를 만드는 방법이다. 특히 단면이 일정한 물체를 만들기에 적합하며, path로는 직선과 원이 많이 사용된다.

4.1 스위핑

스위핑하는 방법으로는 [6, 7]에서 제시한 방법을 채택하였다. 이 방법은 B-spline의 가장 큰 특징인 조정점을 충분히 이용한 방법으로, 경우에 따라 만들어 지는 곡면의 식이 정확한 수학적인 식이 되기를 원할 때, 그 요구를 충족시킬 수 있다.

B-spline 은 조정점들로 정의되기 때문에, Fig.2에 표시한 것처럼 한 점을 path에 따라 offset 시켜 offset 곡선을 만드는 방법을 단면을 정의하는 모든 조정점들에 적용하면 단면을 스위핑시켜 얻어지는 물체를 만들 수 있다. 주어진 path에 offset 거리가 법선 방향으로 Dist인 offset 곡선은 path의 각 조정점 상에서 아래의 Factor만큼 offset시키면 된다.[7]

$$Factor = Dist \times (1 + \kappa \times DPN)$$

여기에서,
Dist : offset 거리
κ : control node에서의 곡률
DPN : 조정점과 control node사이의 거리

만약 Fig.3에서와 같이 cross profile상에서 법선과 ϕ의 각을 이루고 dist만큼 떨어진 점이 주어진 path를 따라 움직일 때 생기는 offset 곡선의 조정점 CP_n은[7],

$$CP_n = CP_o + Factor \times (\cos\phi\, \hat{N} + \sin\phi\, \hat{B})$$

여기에서,
CP_n : offset 곡선의 조정점
CP_o : 주어진 patch의 조정점
$\hat{N}$: 법선 방향의 단위 벡터
$\hat{B}$: Binormal 방향의 단위 벡터
Binormal 은 접선과 법선 방향의 단위 벡터의 외적(cross product) 이다.

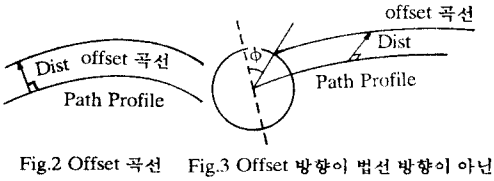

Fig.2 Offset 곡선 Fig.3 Offset 방향이 법선 방향이 아닌 경우

위의 식은 원에서 유도되었지만, 일반적인 경우에 사용해도 무리가 없다. 한 점을 offset시켜 곡선을 만드는 위의 방법에서 B-spline 곡선으로 표현된 단면을 이동시켜 곡면을 만드는 방법을 생각할 수 있다. B-spline 곡선은 조정점으로 표시되기 때문에 단면상의 조정점들이 patch를 따라 움직일 때 형성되는 offset 곡선의 조정점들을 구하면 곡면의 모든 조정점이 구하여진다. 즉, 단면을 이루는 곡선의 모든 조정점에 대해 아래 과정을 적용한다.

- Offset 각 (offset 방향과 법선 사이의 각도) 을 계산한다.
- Offset 거리를 계산한다.
- Path의 모든 조정점에서 offset된 조정점을 구한다.

이러한 방법은 path와 단면을 이루는 곡선이 B-spline으로 표시될 때 가능한 스위핑 방법이다. 여기에서 특수한 경우로 path가 직선일 때와 원일 때를 많이 사용한다. Path가 직선일

때를 translational 스위핑이라 하고, 원일 때를 rotational 스위핑이라 한다. Rotational 스위핑은 단면을 회전축을 중심으로 회전시켜 회전체를 만드는 방법이다. Fig.4는 path가 사분원이고 단면이 원일 때 스위핑시켜 만든 토러스(torus)의 일부 형상을 나타낸다.[6]

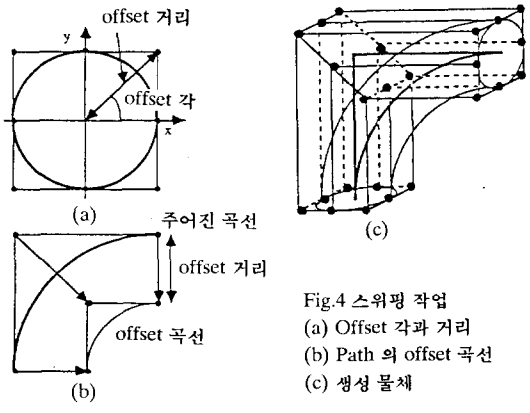

Fig.4 스위핑 작업
(a) Offset 각과 거리
(b) Path 의 offset 곡선
(c) 생성 물체

4.2 Translational 스위핑

Path 가 직선일 때 단면을 직선 방향으로 밀어내어 물체를 만드는 방법이다. 이 때에는 단면의 조정점에 path와 같은 방향으로 같은 거리만큼 더해준다.
즉, 직선으로 이루어진 path를 정의하는 2 개의 조정점 $\vec{P}_1$, $\vec{P}_2$ 로부터,

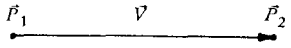

Fig.5 직선

$$\vec{V} = \vec{P}_2 - \vec{P}_1$$

라는 벡터를 정의하여, 단면을 정의하는 모든 조정점에 $\vec{V}$를 더하여 새로운 조정점을 만든다. 이 경우 원래 주어진 단면상의 조정점과 새로 얻어진 조정점은 스위핑으로 얻어진 곡면을 정의한다.

4.3 Rotational 스위핑

주어진 단면을 가진 회전체를 만드는 방법이다. 4.1에서 설명한 방법대로 path를 원으로 하면, 주어진 단면을 회전시키게 된다. 반지름이 R인 원에서는 곡률이 1/R로 일정하고, control node에서 조정점과의 거리는 "0" 또는 $(\sqrt{2}-1)\times R$로 일정하다. 원을 회전시켰을 때 토러스가 나오는 형상은 Fig.4에서 path가 완전한 원일 때 생겨남을 알 수 있다.

5. 스키닝 작업(Skinning Operator)의 개발

5.1 스키닝 방법

Fig.6에 나타난 과정을 살펴 보며 스키닝의 일반적인 과정을 살펴보자. (a)는 path를 (b),(c),(d)는 cross를 나타낸다. (b),(c),(d)에서 삼각형은 각기 다른 곡선 방정식이 교차하는 vertex를 나타낸다. (e)에서는 cross들이 2 개의 group으로 나누어진 모양이며, 이 때 나누어지는 각 group의 수는 모든 cross를 비교하여 가장 edge 수가 작은 cross의 edge 수와 같다. 즉 Fig.6 에서 각 cross의 edge 수는 (b)가 2, (c)가 3, (d)가 4 이므로, (b)를 기준으로 2 개의 group으로 나누어 졌다. 따라서 나중에 곡면이 형성될 때 위, 아래 2 개의 곡면이 만들어지게 된다. (f)는

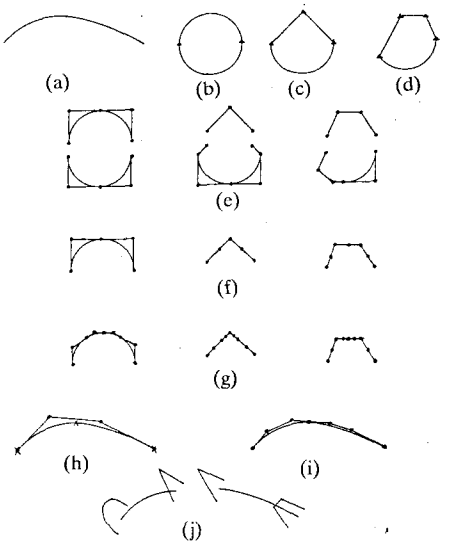

Fig.6 스키닝 과정
(a) Path Profile
(b),(c),(d) Cross Profile
(e) 나누어진 group
(f) 차수를 맞춘 뒤 하나의 복합 곡선으로 만든 모습
(g) 조정점의 갯수를 맞춘 모습
(h) Path의 조정점과 cross가 놓일 위치
(i) Path의 분리
(j) Path 위에 놓인 cross 모습

(e)에서 윗 부분 6 개의 곡선 중 최고 차수를 찾아 (e)에 포함된 모든 곡선을 최고 차수로 차수를 올린 뒤 각 각 하나의 복합 곡선으로 만든 것이고, (g)는 (f)에서 세 개의 복합 곡선의 knot 벡터를 비교해서로 없는 knot를 첨가시켜 세 복합 곡선의 조정점의 갯수가 같아지게 한 것이다. 이렇게 하면 같은 group에 포함된 모든 복합 곡선의 차수와 조정점의 갯수가 같게 되고 다음의 두 가지 방법으로 곡면의 식을 구할 수 있다. 첫번째 방법은 4장에서 설명한 스위핑의 방법을 이용한 것이고, 두번째 방법은 각 복합 곡선의 해당 조정점들을 스플라인으로 연결하는 방법이다.

먼저 스위핑에서 사용된 방법을 이용해 곡면의 식을 만드는 방법을 설명하겠다.여기서 스위핑과 다른 점은 스위핑에서는 offset하고자 하는 점에서 offset 각과 거리가 일정하나 스키닝에서는 다르다는 점이다. 스키닝에서는 offset 각과 거리는 양 끝에 위치한 cross에서 각 각 offset 각과 거리를 구한 다음 path 의 해당 조정점의 chord length의 비로 선형 보간하여 구한다. Fig.6의 (h)에 x로 표시된 위치는 cross 가 놓여질 위치를 나타내고, 이 점에 path 의 조정점이 위치하도록 이 점에 해당하는 parameter를 path의 order (차수+1) 만큼 첨가한다. 그러면 (i)처럼 path의 조정점들이 위치하게 되고, path는 양 끝에 cross를 가진 2 개의 곡선으로 분리된 모양을 나타낸다. Path의 각 곡선마다 곡면을 이루는 조정점을 모두 구할 수 있고, path의 곡선마다 구한 곡면의 조정점을 모두 합치면 구하고자 하는 전체 곡면을 정의하는 조정점이 얻어진다.

두번째 방법으로는 cross들을 path 위에 위치시킨 다음, cross들의 조정점을 지나는 스플라인을 만드는 방법이다. 여기에서는 homogeneous 좌표에 주의해야 하며, 역시 선형 보간하여 구한다.

위에서 언급된 곡선을 분리시키는 방법, 복합 곡선으로 만드는 방법, 조정점의 갯수를 같게하는 방법, 곡면의 식을 구하는 방법들을 설명하겠다.

5.2 곡선을 분리시키는 방법

B-spline으로 나타내어진 곡선은 보엠 알고리즘(Boehm Algorithm)으로 그 곡선의 차수만큼 knot를 첨가하면 두 개의 곡선으로 분리된다.[8]

5.3 복합 곡선으로 만드는 방법

스키닝에서 주어지는 cross를 이루는 곡선의 수도 서로 다르며 각 곡선의 차수도 서로 다르다. 그런데 하나의 곡면 식을 만들기 위해서는 포함된 cross 들이 하나의 곡선으로 되어야 하며, 그 차수도 같아야 한다. 여기에서는 서로 연결된 (1차 연속성이 만족되는) 여러 개의 B-spline 곡선을 하나의 복합 곡선 (composite curve)으로 만드는 과정을 설명하겠다. 복합 곡선으로 되려면 아래의 과정을 거쳐야 한다.

- 곡선들 가운데 가장 큰 차수를 찾아 모든 곡선을 이 차수에 일치시킨다.
- 서로 연결된 곳에서 중복되어 있는 조정점을 하나씩 없애며 복합 곡선을 만든다.

B-spline curve는 곡선을 조정점을 사용하여 기하학적으로 나타내는 방법이므로 곡선의 모양을 바꾸지 않고 차수를 올릴 수 있다. 차수를 올리는 과정은 [9]에 나타나 있고, 여기서는 차수가 모두 같게된 곡선들을 하나의 복합 곡선을 만드는 것만 설명하겠다. Non-periodic B-spline은 곡선의 시작점과 끝점에 조정점이 놓이게 된다. 그러므로, 두 곡선이 만날 때 만나는 위치에서 조정점이 중복되게 된다. 복합 곡선을 만들 때 복합 곡선의 조정점은 중복되는 조정점을 하나 빼고 두 곡선의 조정점을 합한 것이다. 복합 곡선의 knot 벡터는 앞 곡선의 마지막 knot 를 하나 빼고, 뒤 곡선의 knot 벡터의 처음 부분에서 (차수 + 1) 개의 knot를 뺀 뒤, 앞 곡선의 마지막 knot를 더하여 합한 것이다. 아래의 보기는 이를 식으로 나타낸 것이다.

(보기) : Non-periodic B-spline으로 표현되고 차수가 같은 두 개의 곡선을 하나의 복합 곡선으로 만드는 방법

입력 곡선 :
곡선 1 : 차수 --- K-1
조정점의 갯수 --- N1
조정점 --- $\{P1_i\}$, i=1,...,N1
Knot 벡터 --- $\{T1_i\}$, i=1,...,N1+K
곡선 2 : 차수 --- K-1
조정점의 갯수 --- N2
조정점 --- $\{P2_i\}$, i=1,...,N2
Knot 벡터 --- $\{T2_i\}$, i=1,...,N2+K
조건 : $P1_{N1} = P2_1$

복합 곡선 :
곡선 : 차수 --- K-1
조정점의 갯수 --- N = N1 + N2 - 1
조정점 --- $\{P_i\}$, i=1,...,N
$P_i = P1_i$, i = 1,N1
$P_i = P2_{i-N1+1}$, i = N1+1,...,N
Knot 벡터 --- $\{T_i\}$, i=1,...,N+K
$T_i = T1_i$, i = 1,...,N1+K-1
$T_i = T1_{N1+1} + T2_{i-N1+1}$, i = N1+K,..,N+K

5.4 복합 곡선의 조정점의 갯수를 모두 같게하는 방법

B-spline에서 knot 벡터의 knot 갯수는 (조정점의 갯수 + 차수 + 1) 이다. 그러므로, 복합 곡선의 조정점의 갯수가 다르다면 서로 다른 knot를 가지고 있다. 서로의 knot를 비교하여 없는 knot를 첨가시켜 준다. B-spline 곡선에서 knot를 하나 첨가하면 조정점의 갯수도 하나 증가된다. 이렇게 하여 knot 벡터의 knot를 모두 같게하면 복합 곡선의 조정점의 갯수는 같아지게 된다. Knot를 첨가시키는 방법은 [8]에 설명된 보엠 알고리즘 (Boehm Algorithm)을 사용한다.

5.5 곡면의 식을 구하는 방법

곡면의 식을 구하는 방법으로는 조정점을 offset시키는 방법과 조정점을 지나는 스플라인을 만드는 두 가지 방법이 있다.

A. Path 의 조정점에서 offset시켜 곡면의 식을 구하는 방법
이 방법은 3.1에서 설명한 스위핑 방법을 확장시킨 것으로 다음 두 과정을 통해 곡면의 식을 구한다.

- Path의 시작점과 끝점이 아닌 위치에 cross가 있을 때, 그 위치를 중심으로 path를 분리시킨다.
- Path의 각 구간마다 곡면의 식을 구해 합하면 전체 식이 된다.

Path의 양 끝점이 아닌 위치에 cross가 놓여 있다면 그 위치에 해당하는 parameter를 구해 path를 분리시킨다. Path를 분리시키는 방법으로는 위에서 설명한 knot를 첨가시키는 방법으로 보엠 알고리즘을 사용한다. 이렇게 path를 분리시키는 작업이 끝나면 path의 분리된 곡선마다 양 끝에 cross를 가지게 되어 4.1의 방법을 이용할 수 있다. 다른 점은 양 끝 cross에 위치한 해당 조정점이 서로 다른 offset 각과 거리를 가지고 있어 path의 중간 조정점에서 사용할 offset 각과 거리를 구하여야 한다는 점이다. 즉 cross의 한 조정점을 path를 따라 offset 시킬 때, path의 조정점마다 다른 offset 각과 거리를 가진다는 것이다. 이 offset 각과 거리는 control polygon의 길이를 기준으로 하여 양 cross의 offset 거리와 각을 선형 보간하여 구한다. 이렇게 하면 path의 분리된 각 구간에서 곡면의 식이 나오고, 이를 합하면 전체 곡면의 식이 된다.

B. Cross의 조정점을 지나는 스플라인으로 곡면의 식을 구하는 방법
Path위에 여러 개의 cross가 놓여 있을 때 사용하며, cross들을 path 위에 위치시킨 다음 cross의 조정점을 지나는 스플라인을 만드는 방법이다.[10] 스플라인은 non-rational B-spline으로 만들어 지기 때문에 만들어 지는 조정점의 homogeneous 좌표에 영향을 주지 않으나, cross가 일반적으로 rational이기 때문에 이를 고려해야 한다. 이를 위해 만들어진 스플라인에서 각 knot 마다 3 개가 되도록 (스플라인의 차수가 3 이므로) knot를 첨가하면, 입력으로 주었던 점에 스플라인의 조정점이 위치하게 되고 입력점 사이에 두개의 조정점이 생기게 된다. 그러면 원래의 cross의 조정점에서는 자신의 homogeneous 좌표를 가지게 되고, 그 사이에 생긴 두 점에서는 양 쪽 cross 의 homogeneous 좌표를 보간하면 된다.

5.6 Body 데이터의 생성 과정

이제까지는 path와 cross로 곡면식을 만드는 과정을 설명하였다. 곡면식을 만듦으로써 물체에 대한 모든 topology, geometry data를 제공할 준비가 되었다고 할 수 있다. 여기에서는 스키닝 작업에서 생성되는 물체를 완전히 정의하는 삼차원 데이터를 찾는 과정을 설명하겠다. 먼저 기준 cross, 즉 edge 수가 가장 적은 cross를 찾아 다른 cross를 기준 cross에 맞추는 과정에서, 각 cross의 edge와 vertex가 달라진다. Path의 edge 양 끝에 있는 cross는 path 위에

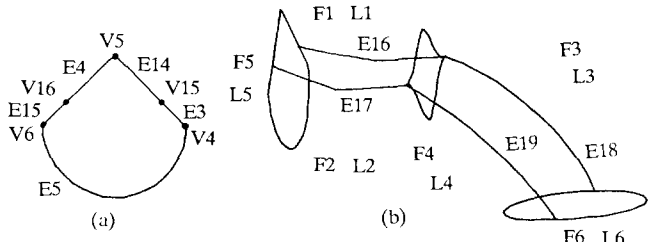

Fig.7 물체의 데이터 생성
(a) Cross의 분리
(b) 생성된 데이터

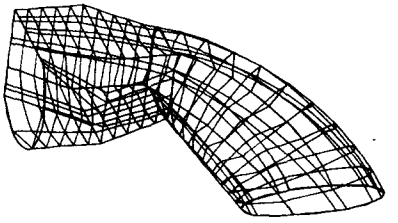

Fig.8 스키닝으로 생성된 물체

놓여진 뒤 물체의 edge, vertex 가 되며, 그 밖의 cross는 생성되는 물체의 곡면식을 형성 하는데만 관련되고, 물체의 edge, vertex 는 되지 못한다. 그리고 path 방향으로 기준 cross의 vertex 수만큼 양 끝 cross를 잇는 edge가 생기게 되고, edge 들로 구성된 폐곡선마다 face와 loop가 생기게 된다. Path의 각 edge마다 위와 같이 물체의 vertex,edge,loop,face 가 생기게 되고, path의 모든 edge에 대해 위의 작업이 끝나게 되면 물체의 양 끝에 face와 loop가 만들어 진다. 구멍이 있는 경우에도, 구멍을 이루는 Skin_Edge_Group마다 path의 edge를 따라 vertex, edge, loop, face 는 같은 방법으로 생긴다. 단, Skin_Edge_Group을 이루는 edge들의 방향을 표면 Skin_Edge_Group과 반대로 해서, 만들어지는 face의 곡면식은 법선 방향이 물체 밖으로 향하게 해야 하며, path의 양 끝면에서 hole loop는 생기나 face는 새로 생기지 않게 된다.

Fig.1을 예로 위의 과정을 따라가 보자. Fig.1에서 path의 첫번째 edge E1 에는 3 개의 cross (a), (b), (c) 가 있다. 여기에서 (c)가 (a) 보다 edge 수가 적으므로 (c)가 기준 cross가 된다. (c) 의 vertex가 이루는 각을 기준으로 (a), (b) 는 분리되게 되는데, 각을 구할 때는 cross의 vertex를 산술 평균한 지점을 중심점으로 하여, 이 점과 vertex가 이루는 각을 구하게 된다. 중심점을 원점으로 하지 않고 새로 구하는 이유는 vertex와 이루는 각을 구하는데 기준이 되는 중심점이 항상 Skin_Edge_Group 안에 있게 하기 위해서다. 대개 path는 표면을 나타내는 Skin_Edge_Group의 내부를 통과하나, 구멍을 나타내는 Skin_Edge_Group의 밖을 통과하는 경우도 있다. 이렇게 하면 (b)는 분리되지 않고, (a)는 분리되어 Fig.7에서와 같이 edge 수가 총 5 개가 된다.

5.5 에서 설명한 방법으로 path의 E1, cross (a)의 E14, E4, (b)의 E6, E7, (c)의 E10으로 곡면식을 만들 수 있다.(Fig.1과 Fig.7 참조) 이 때 path 방향으로 V15 와 V11 을 연결하는 edge E16, V16 와 V14 를 연결하는 edge E17 이 만들어 지며, 이들의 곡선은 곡면의 경계에 해당되므로 곡면식에서 바로 알 수 있다. 여기에서 edge E14, E4, E17, E10, E16 으로 이루어지는 face F1 과 loop L1 이 만들어 진다.

그리고 path의 edge E1 와 cross (a)의 E15, E5, E3, (b) 의 E8, E9, (c)의 E11 로, E15, E5, E3, E16, E11, E17 로 이루어 지는 face F2 와 loop L2 를 구성하게 된다.

같은 방법으로 path의 두번째 edge E2 와 cross (c),(d)로 path 방향 edge E18, E19 가 생기고, E10, E19, E12, E18 로 구성되는 face F3 과 loop L3, E11, E18, E13, E19 로 구성되는 face F4 와 loop L4 가 만들어 지게 된다.

이상으로 path의 edge와 관련된 face와 loop는 다 만들어졌고, 다음은 물체의 양 끝을 이루는 face와 loop가 만들어져야 한다. 이는 path의 양 끝에 위치한 cross (a), (d)를 이용해 만들며, E3, E14, E4, E15, E5 로 face F5 와 loop L5 가 만들어지며, E12, E13 으로 face F6, loop L6가 만들어진다. 이로써 물체에 대한 정보는 완성된다.

6. 결론 및 고찰

지금까지 설명한 스위핑과 스키닝 기능을 사용하면 단면이 복잡하고 자유 곡면을 갖는 물체를 다른 방법보다 쉽게 모델링할 수 있다. 그리고 규칙적인 구멍이 있는 경우에도 부울리안 작업을 거치지 않고 모델링할 수 있는 장점도 있다. 그러나 대부분의 물체는 하나의 path로 모델링 할 수 없고 여러 개의 부분으로 만든 뒤 합쳐지는 과정을 거쳐야 한다. 즉 스위핑과 스키닝으로 만들어진 각각의 물체를 합치는 기능이 필요하며, 또 결합 부분을 매끄럽게 하기 위해서는 라운딩(Rounding) 작업도 필요하다. Fig.9는 스위핑과 스키닝을 이용해 커넥팅 로드를 모델링한 것이다. 이는 6개의 부분으로 나누어 모델링한 것인데, 하나의 물체가 되기 위해서는 합쳐지는 기능이 있어야 하며, 이는 앞으로 개발할 예정이다.

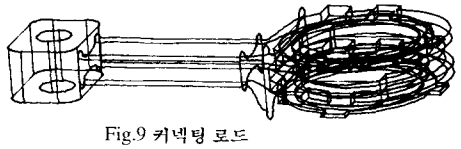

Fig.9 커넥팅 로드

참고문헌

1. M.E. Mortenson, " Geometric modeling", John Wiley & Sons, 1985
2. I.C. Braid, "A New Shape Design System", CAD Group Document No. 86 University of Cambridge Computer Laboratory, Cambridge,
3. I.C. Braid, R.C. Hillyard,"Geometric modelling in ALGOL 86", University of Cambridge Computer Laboratory, Cambridge,
4. 이 상찬, "경계를 갖는 자유 곡면의 NC 가공을 위한 공구 경로의 계산", 서울대학교 대학원 기계설계학과 석사 학위 논문
5. L. Piegl and W. Tiller, "Curve and Surface Constructions using Rational B-splines", Computer Aided Design, Vol.19, No.9, November 1987
6. S. Coquillart, "Computing offsets of B-spline Curves", Computer Aided Design, Vol.19, No.6, July 1987
7. S. Coquillart, "A Control-point-based Sweeping Technique", IEEE Computer Graphics and Applications, Vol.7, No.11, Nov. 1987
8. W. Boehm, "Inserting New Knots into B-spline Curves", Computer Aided Design, Vol.12, No.6, July 1980, pp.199-201
9. E. Cohen, T. Lyche, L.L. Schumaker, "Algorithms for degree-raising of Splines", ACM Transactions on Graphics, Vol.4, No.3, July 1985
10. B.A. Barsky and D.P. Greenberg, "Determining a Set of B-spline Control Vertices to Generate an Interpolating Surface", Computer Graphics and Image Processing, Vol.14, No.3, Nov. 1980 pp.203-206
11. W. Tiller, "Rational B-splines for Curve and Surface Representation", IEEE Computer Graphics and Applications, Vol.3, No.9, Sept. 1983, pp.61-69