그래픽스 하드웨어를 이용한 스윕 곡면의 렌더링

고대현° 윤승현† 이지은‡

삼성종합기술원, 동국대학교 멀티미디어공학과†, 조선대학교 컴퓨터공학부‡

hyuna76@hanafos.com, shyun@dongguk.edu†, jieunlee@chosun.ac.kr‡

Rendering of Sweep Surfaces using Programmable Graphics Hardware

Dae-Hyun Ko° Seung-Hyun Yoon† Jieun Lee‡

Samsung Advanced Institute of Technology, Dept. of Multimedia Eng., Dongguk Univ.†, School of Computer Eng., Chosun Univ.‡

요 약

본 논문에서는 그래픽스 하드웨어를 이용한 스윕 곡면의 효율적인 렌더링 알고리즘을 제안한다. 스윕 곡면은 스플라인 모션을 따라 움직이는 단면 곡선으로 표현된다. 이러한 표현은 행렬과 벡터의 곱으로 계산되며, 이는 프로그래밍이 가능한 그래픽스 하드웨어에 쉽게 적용될 수 있다. 스플라인 모션과 단면 곡선의 정보는 텍스쳐 메모리에 저장된다. 그래픽스 하드웨어의 정점 프로세서는 두 개의 곡면 매개변수를 2차원 정점으로 입력받아 한 번의 행렬 곱셈으로 스윕 곡면의 정점 좌표와 법선 벡터를 계산한다. 제안한 GPU 기반 스윕 곡면의 렌더링은 CPU 기반 렌더링에 비해 10배에서 40배 정도의 속도 향상을 보였다.

Abstract

We present an efficient algorithm for rendering sweep surfaces using programmable graphics hardware. A sweep surface can be represented by a cross-section curve undergoing a spline motion. This representation has a simple matrix-vector multiplication structure that can easily be adapted to programmable graphics hardware. The data for the motion and cross-section curves are stored in texture memory. The vertex processor considers a pair of surface parameters as a vertex and evaluates its coordinates and normal vector with a single matrix multiplication. Using the GPU in this way is between 10 and 40 times as fast as CPU-based rendering.

키워드: 렌더링, 자유형상 곡면, 스윕 곡면, 프로그래밍 가능한 그래픽스 하드웨어, GPU

Keywords: Rendering, Freeform surface, Sweep surface, Programmable graphics hardware, GPU

1. 서론

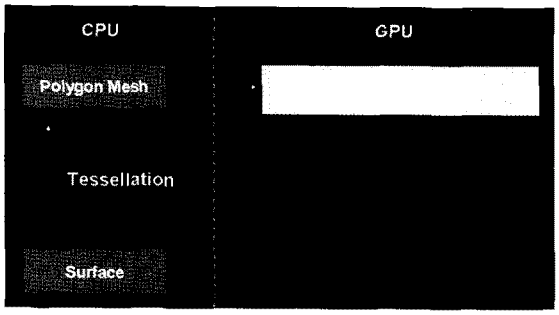
자유형상 곡면은 3차원 형상 모델을 설계하기 위해 널리 사용되고 있다. 대표적인 그래픽 소프트웨어인 MAYA와 3D Studio Max는 다양한 자유형상 설계 도구를 지원하고 있다. 최근 그래픽스 하드웨어를 이용한 넙스 곡면의 렌더링이 구현되고 있으며 [1, 2, 3, 4], 몇몇 게임 콘솔에서는 넙스 곡면이 하드웨어로 직접 렌더링되고 있다 [5, 6].

곡선이나 곡면과 같은 물체를 궤적에 따라 이동하거나 회전시켜 생성되는 스윕 곡선은 직관적이고 절차적인 모델링 기법으로, 실제로 사용되는 3차원 자유형상 모델의 상당 부분이 스윕 곡면으로 모델링되고 있다. 기계 장비의 내부 부속품은 대부분 스윕 곡면의 특수한 형태인 평면, 원기둥, 원뿔, 환원체들의 CSG(Constructive Solid Geometry) 모델이고 건축물 내외부 형상도 스위핑 기법인 선형 구축(linear extrusion)과 회전 곡면(surfaces of revolution)으로 만들어진다 [7, 8, 9, 10, 11]. 인체형상의 몸통, 팔, 다리 등이 스윕 곡면으로 모델링될 수 있다 [12, 13]. 그래픽 아티스트들에게 가장 널리 쓰이는 디자인 도구인 Generalized cylinder 역시 스윕의 특수한 형태이다.

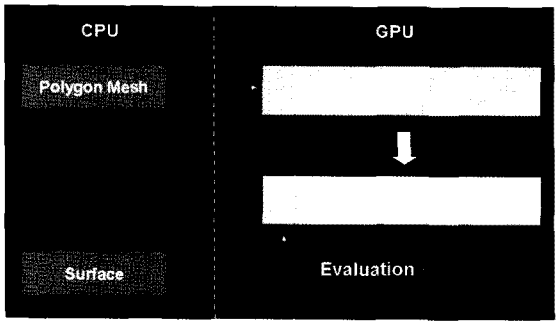
스윕 곡면을 렌더링하기 위해 많은 효율적인 알고리즘들이 개발되었다 [7, 8, 9, 14]. 본 논문에서는 지금까지 GPU 기반 곡

*투고일: 2010-04-30 / 심사일: 1차 2010-10-25 / 게재확정일: 2010-11-05

면 렌더링에서 다루어지지 않았던 스윕 곡면을 프로그래밍 가능한 그래픽스 하드웨어를 기반으로 빠르고 효율적으로 렌더링하는 방법을 제안한다. CPU에서 스윕 곡면을 테셀레이션하여 곡면상의 정점들을 계산한 후 그래픽스 파이프라인으로 전달한 것과 달리, GPU 기반의 스윕 곡면 렌더링은 GPU에서 직접 곡면상의 정점들을 계산하여 렌더링을 수행한다 (그림 1). GPU 기반의 스윕 곡면 렌더링은 계산 성능을 향상시킬 뿐만 아니라 곡면의 테셀레이션 결과를 저장할 필요가 없다는 장점이 있다.



(a) CPU를 이용한 곡면의 계산



(b) GPU를 이용한 곡면의 계산

그림 1: CPU와 GPU를 이용한 곡면의 계산

곡면을 GPU에서 계산할 경우 중요한 문제는 곡면이 가지고 있는 병렬적인 구조를 찾아서 이를 GPU 연산으로 변환하는 것이다. Jüttler와 Wagner [15]는 B-스플라인 모션에 의한 물체의 스윕핑을 소개하였다. Chang 등 [16]은 스윕 곡면을 궤적 곡선에 따라 움직이는 단면 곡선으로 표현하였다. 이와 같이 시간에 대해 매개화되는 모션의 표현은 행렬과 벡터의 단순한 곱셈 구조이므로 그래픽스 하드웨어에 쉽게 적용된다. 이러한 관찰을 토대로 본 논문은 그래픽스 하드웨어를 이용한 효율적인 스윕 곡면 렌더링 알고리즘을 제안한다.

2. 스윕 곡면

2.1 모션에 기반한 스윕 곡면

스윕 곡면 $S(u,t)$는 단면 곡선 $C(u)$를 모션 $M(t)$에 대해 스위핑하여 만들어진다 [16]:

$$\begin{aligned} S(u,t) &= M(t)C(u) \\ &= \left(\begin{array}{c|c} R(t) & T(t) \\ \hline 0\ \ 0\ \ 0 & 1 \end{array}\right) C(u). \end{aligned}$$

모션 행렬 $M(t)$는 3×3 회전 행렬 $R(t)$와 이동 벡터 $T(t)$로 이루어지며, 회전 행렬 $R(t)$는 시간에 대해 연속적인 회전변환을 나타내는 쿼터니언 곡선 $Q(t)$에 의해 설정된다. 이동 벡터 $T(t)$도 시간에 대해 연속적인 이동변환을 나타내는 곡선이다.

모션을 B-스플라인 모션으로 표현할 경우, 다음과 같이 스윕 곡면의 B-스플라인 곡면 표현을 얻을 수 있다.

$$\begin{aligned} S(u,t) &= M(t)C(u) \\ &= \left(\sum_j B_j^q(t)M_j\right)\left(\sum_i B_i^p(u)C_i\right) \\ &= \sum_j \sum_i B_j^q(t)B_i^p(u)\left(M_jC_i\right) \end{aligned}$$

여기서 $B_j^q(t)$는 차수가 q인 B-스플라인 기저함수이고 $B_i^p(u)$는 차수가 p인 B-스플라인 기저함수이다. M_j은 모션의 제어행렬, C_i는 단면 곡선의 제어점이다. 즉 스윕 곡면은 차수가 $p \times q$인 B-스플라인 곡면으로도 표현이 가능하다. 다음 절에서 스윕 곡면에 대한 두 가지 표현에 대해 필요한 연산을 분석한다.

2.2 계산 측면에서의 분석

행렬과 벡터는 GPU의 기본 자료형이며, GPU는 행렬과 벡터의 곱을 단일 연산으로 처리할 수 있다. 차수가 k인 B-스플라인 곡선의 한 점은 GPU를 이용할 때, 기저함수와 제어점에 대한 $k+1$번의 스칼라 곱셈과 k번의 벡터 덧셈으로 계산된다.

차수가 p인 B-스플라인 곡선 $C(u)$와 차수가 q인 B-스플라인 곡선 $Q(t)$와 $T(t)$를 가정하자. $C(u)$는 매개변수 u를 샘플링하여 계산되고 시간 t에 대한 회전 변환 $Q(t)$와 이동 변환 $T(t)$에 의해 기하 변환된다. u를 m번 샘플링하고, t를 n번 샘플링한다면, 연속적인 모션에 의해 표현된 스윕 곡면상의 $m \times n$개의 정점들을 계산해 내기 위해서는 총 $m(p+1)+2n(q+1)+mn$회의 곱셈과 $mp+2nq$회의 덧셈이 요구된다. 곡선 $Q(t)$와 $T(t)$와 $C(u)$가 삼차 B-스플라인 곡선이고 $m=100$이고 $n=100$이면, 총 11,200회의 곱셈과 900회의 덧셈이 필요할 것이다.

한편 스윕 곡면을 $p \times q$차의 B-스플라인 곡면으로 표현한 경우 곡면 위의 한 점을 계산하기 위해서는 총 $2(p+1)(q+1)$회의 곱셈과 $pq+p+q$회의 덧셈이 필요하다. B-스플라인 곡면을 $m=100$과 $n=100$으로 샘플링한다면, 총 320,000회의 곱셈과 150,000회의 덧셈이 필요하다. 그러므로 스윕 곡면을 샘플링

하는데는 B-스플라인 곡면 표현보다 모션에 기반한 표현이 훨씬 효율적임을 알 수 있다.

3. GPU를 이용한 스윕 곡면의 렌더링

이 절에서는 앞에서 살펴 본 스윕 곡면을 그래픽스 하드웨어에서 계산하여 렌더링 성능을 향상시키는 방법을 설명한다. 그림 2는 스윕 곡면을 GPU를 이용하여 렌더링하는 과정의 개요도이다.

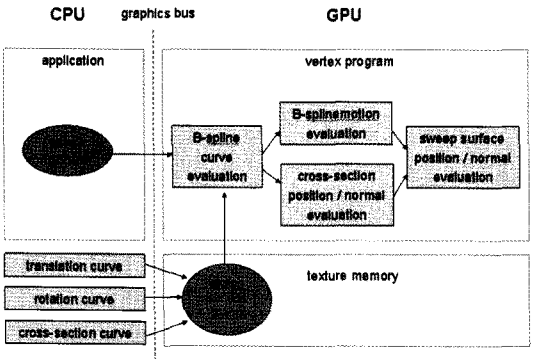


그림 2: GPU를 이용한 스윕 곡면의 렌더링 시스템의 개요도

스윕 곡면을 렌더링하기 위해 프로그래밍이 가능한 정점 프로세서를 이용하였다. 스윕 곡면을 정의하는 곡선 정보들은 텍스쳐 메모리에 저장되었다. 정점 프로그램은 샘플링된 곡면 매개변수 (u,t)값이 입력된 2차원 정점들이 전달되면서 활성화된다. 스윕 곡면 위의 점들은 u에 의해 결정된 단면 곡선상의 점들을 모션 $M(t)$에 의해 회전 및 이동하여 계산된다. 스윕 곡면의 계산은 시점 변환과 음영처리에 선행되므로, 계산된 3차원 정점들은 이웃한 정점들과 삼각형 프리미티브들로 조합되어 그래픽스 파이프라인의 다음 단계인 래스터화 과정으로 전달된다.

3.1 스윕 곡면의 정보를 저장하는 텍스쳐 데이터

스윕 곡면은 이동 곡선과 회전 곡선, 단면 곡선에 의해 정의된다. 이러한 곡선 정보들은 그림 3에 보인 것과 같이 2차원 텍스쳐 메모리에 적재된다.

텍스쳐의 각 행은 하나의 곡선 정보를 담고 있다: 각 행의 첫 번째 픽셀은 곡선의 차수와 나트(knot)의 갯수, 제어점의 갯수를 R, G, B 요소에 각각 저장하고 있다. 두번째 픽셀부터 나트벡터와 제어점들이 순서대로 저장된다.

3.2 정점 위치 및 법선 벡터

주어진 스윕 곡면의 매개변수 (u',t')에 대해, 정점 프로그램은 이동 곡선에 대해 $T(t')$와 회전 곡선에 대해 $Q(t')$을 계산하고,

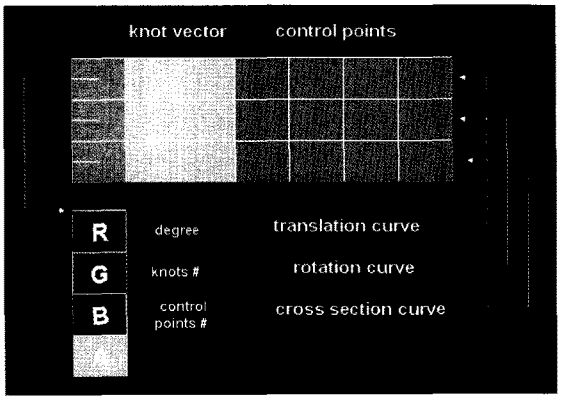


그림 3: 스윕 곡면 렌더링을 위한 테스쳐 메모리 매핑

단면 곡선에 대해 정점 위치 $C(u')$과 법선 벡터 $N(u')$을 계산한다. 스윕 곡면에 대한 정점의 좌표는 단면 곡선위의 정점 위치 $C(u')$을 $R(t')$으로 회전하고, $T(t')$으로 이동하여 계산된다. 또한 스윕 곡면에 대한 정점의 법선 벡터는 단면 곡선위의 정점의 법선 벡터 $N(u')$을 $R(t')$으로 회전하여 계산된다.

정점 좌표와 법선 벡터는 그림 4와 같이 한번의 GPU 행렬 곱셈으로 동시에 계산된다. 회전 $Q(t')$과 이동 $T(t')$에 의해 정의되는 4×4 모션 행렬 $M(t')$은 단면 곡선상의 정점 위치 $C(u')$과 법선 벡터 $N(u')$을 열로 갖는 4×2 행렬과 곱해진다. 곱셈의 결과로 얻어진 4×2 행렬은 단면 곡선의 점들이 모션에 의해 기하변환된 결과이며 스윕 곡면상의 정점 위치와 법선 벡터가 된다.

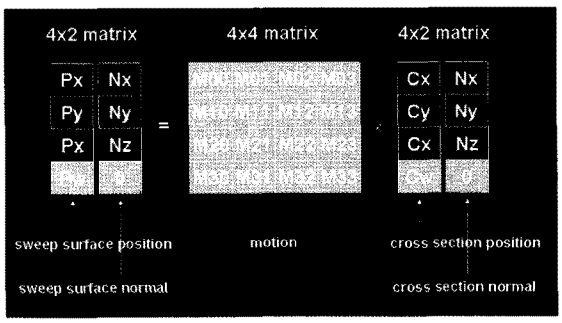


그림 4: 스윕 곡면 계산을 위한 행렬 곱셈

4. 실험 결과

제안하는 GPU 기반 스윕 곡면의 렌더링은 OpenGL과 NVIDIA Cg 쉐이딩 언어로 구현되었으며, 2.4GHz의 CPU와 NVIDIA GeForce 8800 Ultra GPU를 탑재한 펜디엄4 PC에서 실험되었다.

그림 5는 서로 다른 회전 곡선에 의해 비틀림 효과를 가한 스

웝 곡면의 렌더링 결과를 보여 준다. GPU를 이용한 경우 초당 2800 ~ 3100 프레임을 렌더링 하였으며, CPU 기반 렌더링의 경우 평균적으로 105 프레임을 랜더링 하였다. 즉 26 ~ 29배의 속도 향상을 달성할 수 있었다.

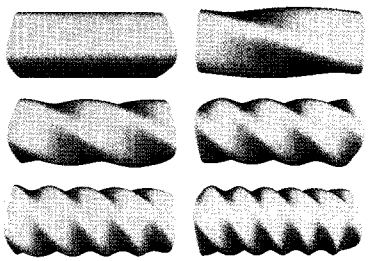
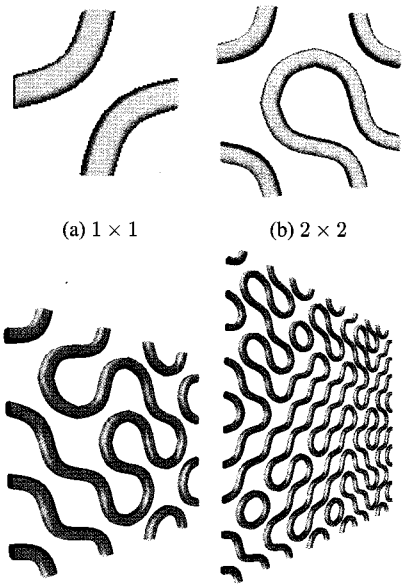


그림 5: 비틀린 스웝 곡면의 렌더링

그림 6은 파이프 곡면의 2차원 타일링 결과를 보여 준다. 그림 6(a)는 2 개의 스웝 곡면으로 만들어진 기본 블럭이며, 그림 6(b)-(d)는 기본 블럭을 반복적으로 타일링하여 만들어진 것이다. 표 1은 GPU와 CPU에서 그림 6의 모델들을 렌더링한 결과를 fps 단위로 비교하여 제시하고 있다.



(a) 1×1 (b) 2×2

(c) 4×4 (d) 10×10

그림 6: 파이프 곡면을 이용한 평면 타일링

그림 7은 파이프 곡면의 3차원 타일링 결과를 보여 준다. 그림 7(a)는 3 개의 스웝 곡면으로 만들어진 기본 블럭이며, 그림 7(b)-(g)는 기본 블럭을 반복적으로 타일링하여 만들어진 것이다. 표 2은 GPU와 CPU에서 그림 7의 모델들을 렌더링한 결과를 fps 단위로 비교하여 제시하고 있다.

타일링 횟수	CPU에서 fps	GPU에서 fps	속도 향상
$1 \times 1 \times 1$	614	5200	8.47
$2 \times 2 \times 2$	173	3380	19.54
$4 \times 4 \times 4$	45	1420	31.56
$6 \times 6 \times 6$	20	730	36.50
$8 \times 8 \times 8$	11	432	39.27

표 1: 그림 6의 CPU와 GPU를 이용한 렌더링의 성능 비교

타일링 횟수	CPU에서 fps	GPU에서 fps	속도 향상
$1 \times 1 \times 1$	455	4634	10.18
$2 \times 2 \times 2$	61	1728	28.32
$6 \times 6 \times 6$	2.28	87.60	38.42
$8 \times 8 \times 8$	0.96	36.45	37.96
$10 \times 10 \times 10$	0.48	19.12	39.80

표 2: 그림 7의 CPU와 GPU를 이용한 렌더링의 성능 비교

파이프 곡면을 위한 타일링 결과에서도 제안한 GPU 기반 렌더링은 CPU 기반 렌더링에 비해 10~40배의 속도 향상을 보였다.

5. 결론

본 논문은 GPU를 이용한 스웝 곡면의 효율적인 렌더링 알고리즘을 제안하였다. 행렬과 벡터의 곱셈으로 단순하게 표현된 스웝 곡면을 GPU의 행렬 연산으로 계산하여 렌더링 시간을 획기적으로 줄일 수 있었다.

모션에 기반하여 스웝 곡면을 계산할 때, 단면 곡선의 매개변수 방향으로 얻은 등위곡선은 단면 곡선에 동일한 모션을 적용하여 생성된다. 향후 이 점을 효과적으로 활용할 수 있도록 제안한 방법을 개선할 필요가 있다.

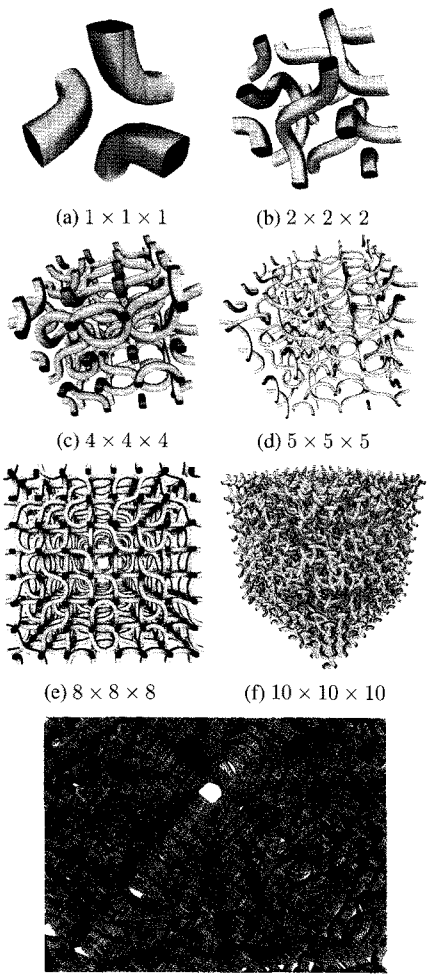
복잡한 형상은 세부 형상과 스웝 곡면에 대한 차이를 차이맵(displacement map)으로 저장할 때 스웝 곡면으로 표현 가능하다. 본 논문에서 제안된 GPU 기반 스웝 곡면의 렌더링 방법을 차이 맵도 처리할 수 있도록 확장할 계획이다.

감사의 글

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2010-0015293, 2010-0005597)

참고 문헌

[1] M. Guthe, A. Balázs, and R. Klein, "Gpu-based trimming and tessellation of nurbs and t-spline surfaces," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1016–1023, 2005.



(a) $1 \times 1 \times 1$ (b) $2 \times 2 \times 2$

(c) $4 \times 4 \times 4$ (d) $5 \times 5 \times 5$

(e) $8 \times 8 \times 8$ (f) $10 \times 10 \times 10$

(g) $10 \times 10 \times 10$ 확대

그림 7: 파이프 곡면을 이용한 공간 타일링

[2] T. Kanai, "Fragment-based evaluation of non-uniform b-spline surfaces on gpus," *Computer-Aided Design and Applications*, vol. 4, no. 1-4, pp. 287–294, 2007.

[3] C. Loop and J. Blinn, "Real-time gpu rendering of piecewise algebraic surfaces," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 664–670, 2006.

[4] H.-F. Pabst, J. Springer, A. Schollmeyer, R. Lenhardt, C. Lessig, and B. Froehlich, "Ray casting of trimmed nurbs surfaces on the gpu," *Proc. IEEE Symposium on Interactive Raytracing*, pp. 151–160, 2006.

[5] D. Blythe, "The direct3d 10 system," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 724–734, 2006.

[6] M. Lee, "Next-generation graphics programming on xbox360," 2006. [Online]. Available: http://download.microsoft.com/download/d/3/0/d30d58cd-87a2-41d5-bb53-baf560aa2373/

[7] J. J. van Wijk, "Ray tracing objects defined by sweeping planar cubic splines," *ACM Transactions on Graphics*, vol. 3, no. 3, pp. 223–237, 1984.

[8] S. Coquillart, "A control-point-based sweeping technique," *IEEE Computer Graphics and Applications*, vol. 7, no. 11, pp. 36–45, 1987.

[9] K. I. Joy, "Visualization of swept hyperpatch solids," *Proc. CG International*, 1992.

[10] W. F. Bronsvoort and F. Kolk, "Ray tracing generalized cylinders," *ACM Transactions on Graphics*, vol. 4, no. 4, pp. 291–303, 1985.

[11] W. F. Bronsvoort, P. R. van Nieuwenhuizen, and F. H. Post, "Display of profiled sweep objects," *The Visual Computer*, vol. 5, no. 3, pp. 147–157, 1989.

[12] D.-E. Hyun, S.-H. Yoon, J.-W. Chang, J.-K. Seong, M.-S. Kim, and B. Jüttler, "Sweep-based human deformation," *The Visual Computer*, vol. 21, no. 8-10, pp. 542–550, 2005.

[13] J. Lee, S.-H. Yoon, and M.-S. Kim, "Realistic human hand deformation," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 479–489, 2006.

[14] T. Nishita and H. Johan, "A scan line algorithm for rendering curved tubular objects," *Proc. of Pacific Graphics*, pp. 92–101, 1999.

[15] B. Jüttler and M. G. Wagner, "Computer aided design with spatial rational b-spline motions," *ASME Journal of Mechanical Design*, vol. 118, pp. 193–201, 1996.

[16] T.-I. Chang, J.-H. Lee, M.-S. Kim, and S. Hong, "Direct manipulation of generalized cylinders based on b-spline motion," *The Visual Computer*, vol. 14, no. 5/6, pp. 228–239, 1998.

〈저자소개〉

이지은
- 1997년 이화여자대학교 컴퓨터공학과 공학사
- 1999년 포항공과대학교 컴퓨터공학과 공학석사
- 2007년 서울대학교 전기컴퓨터공학부 공학박사
- 1999년~2002년 LG전자기술원 정보기술연구소 연구원
- 2008년~현재 조선대학교 컴퓨터공학부 조교수
- 관심분야: 컴퓨터그래픽스, 기하모델링, 멀티미디어정보처리

고대현
- 1999년 서울대학교 컴퓨터공학부 공학사
- 2001년 서울대학교 전기컴퓨터공학부 공학석사
- 2007년 서울대학교 전기컴퓨터공학부 공학박사
- 2007년~현재 삼성전자 DMC연구소 책임연구원
- 관심분야: 컴퓨터그래픽스, 기하모델링

윤승현
- 2001년 한양대학교 자연과학대학 수학과 졸업
- 2007년 서울대학교 공과대학 컴퓨터공학과 박사
- 2007년 서울대학교 BK21 박사후 연구원
- 2007년~현재 동국대학교 영상미디어대학 멀티미디어공학과 조교수
- 관심분야: 컴퓨터그래픽스, 기하모델링