파라미터화된 브러쉬 함수를 이용한 효과적인 실루엣 에지 렌더링

조진화[*o] 김성수[**] 양태천[***]

[*] 경성대학교 컴퓨터 교육학과

[**] 한국전자통신연구원 컴퓨터·소프트웨어기술연구소 영상처리연구부

[***] 경성대학교 정보과학부 컴퓨터과학 전공

Effective Silhouette Edge Rendering using Parameterized Brush Functions

Jin-Hwa Cho[*o] Sung-Soo Kim[**] Tae-Cheon Yang[***]

[*] Dept. of Computer Education, Kyungsung University

[**] Dept. of Image Processing, Computer & Software Technology Laboratory, ETRI

[***] Dept. of Computer Science, Kyungsung University

jhcho@voronoi.kyungsung.ac.kr, kss62937@etri.re.kr, tcyang@star.kyungsung.ac.kr

요 약

3차원 모델을 바탕으로 실루엣 에지를 찾아 디스플레이해주는 대부분의 시스템들은 삼각 메쉬를 기반으로 한 모델 표현법을 사용하고 있다. *NPR(nonphotorealistic rendering)*에서 가장 초점을 두는것은 컴퓨터로 렌더링된 결과가 사람이 그린듯한 효과를 줄 수 있느냐에 있다. 기존에 연구된 대부분의 시스템들은 사람이 그린듯한 효과를 주기 위해 물체의 표면에 대한 텍스쳐어(*Texture*)와 어두운 정도, 그리고 스트록(*Stroke*)을 표면의 윤곽에 맞도록 그리는 많은 기법들을 소개해 왔다. 본 논문에서는 *NPR*표현의 가장 기본이 되는 실루엣 에지 추출에 초점을 두고 추출한 실루엣 에지에 대해 파라미터화된 브러쉬 함수(*Parameterized Brush Functions*)를 적용하여 다양한 스타일로 디스플레이할 수 있는 기법을 제시한다.

1. 서론

컴퓨터 그래픽스의 흐름은 크게 물리학을 기반으로한 사물이나 현상을 더욱 사실적(*photorealistic rendering*)으로 표현하는데 중점을 두는 분야와 비사실적(*nonphotorealistic rendering:NPR*)이고 사람이 그린듯하게 표현하는데 초점을 두는 두 분야로 나눌 수 있다.

비사실적표현은 사진과 같이 사실적으로 나타내는 것보다 펜이나 잉크,목탄등을 사용하여 그리는 대상을 추상화시켜 간략하게 나타내기때문에 전체적인 윤곽에 집중할 수 있게 만든다. 예를 들어, 거의 모든 의학교과서는 일러스트레이션을 이용하여 사람의 신체와 같은 복잡한 구조를 알아보기 쉽게 나타내고 기계장치의 수리 메뉴얼과 같은 책은 사진을 이용하기보다는 일러스트레이션으로 간략하고 명확하게 나타낸다. 그리고 어린이들을 위한 책에서는 좀 더 코믹하고 만화같은 느낌으로 재미를 더해주는 카툰방식을 이용하여 표현하고 있다.

최근 비사실적 표현을 위한 시스템이 많이 만들어졌는데 그 시스템들의 입력방식에 따라 크게 2가지로 나누어 진다. 하나는 3차원 모델을 바탕으로 컴퓨터로 렌더링(*Rendering*)하는 것이고, 다른 하나는 2차원 이미지를 바탕으로 사용자가 개입하여 그리거나 이미지 프로세싱(*Image processing*)기술을 이용하여 그리는 것이다. 본 논문에서는 3차원 모델을 바탕으로 실루엣 에지(*Silhouette edge*)를 추출해 다양한 브러쉬 스타일로 렌더링하는 알고리즘을 제안한다.

2. 관련 연구

실루엣 에지를 추출해내는것은 오랫동안 NPR과 기술적 일러스트레이션 에서 많이 이용되었는데 실루엣 에지를 추출해 내는 가장 큰 목적은 3차원 모델을 컴퓨터를 통해 비사실적인 그림으로 표현해 내는데 있다.

Salesin[5]이 3차원 정보를 이용하여 스트록과 여러 문양을 바탕으로 컴퓨터로 펜과 잉크 일러스트레이션을 하는 방법을 제시하였고 Markosian[2]은 인접 정보를 가지고 있는 정적 다면체 모델(*static polyhedral model*)을 실시간으로 다양한 NPR 스타일로 표현하고 수행시간을 향상시켰다. 실루엣 에지의 가시화(*visibility*)는 Appel[1]의 hidden line algorithm을 사용하여 계산하였으나 인접 정보를 가지고 있어야 하는 단점이 있다. 연결 정보는 인접한 실루엣 에지 사이를 단계별로 검색하는데 이용하여 전체적인 실루엣 커브를 찾아내는데 사용하였고 추출한 실루엣 에지를 다양한 스타일로 렌더링한다. 예를 들어 흔들거리는 효과, 손으로 그린 스타일이 있다.

Rossignac[4]의 방법은 인접 정보가 필요하지 않고 와이어 프레임 표현에서는 depth-buffer를 하얀 장면(scene)으로 먼저 렌더링 한 후 검은색의 와이어 프레임 모드로 렌더링한다. 즉, 표면이 채워져있는 보이지 않는 다각형(back-facing) 대신에 보이지 않는 다각형의 에지를 렌더링하는 방법이다. 그렇게 되면 모든 다각형들이 와이어 프레임으로 렌더링된다. 이 depth function을 'Less than or Equal'이라 한다. 보이지 않는

다각형의 에지의 칼라와 두께는 거리와 카메라의 방향, 라이트 소스 그리고 이미지의 크기에 의해 결정된다. 일반적으로 프레임 버퍼안의 직선의 선분은 상수 두께로 렌더링되므로 상수 두께의 보이는 실루엣 에지를 생성하게 된다. Raskar[3]는 Rossignac의 depth function방법을 사용하여 라인 두께를 불규칙하게 증가시켰다.

본 논문에서는 3차원 모델을 바탕으로 실루엣 에지를 찾아 다양한 스타일로 사람이 그린 듯한 효과를 줄 수 있는 렌더링 기술과 오늘날의 PC 플랫폼에서 렌더링 할 수 있는 비사실적 렌더링 알고리즘을 제시한다.

3. 실루엣 렌더링 알고리즘

3.1 알고리즘 개요

본 논문에서 제안하는 실루엣 렌더링 시스템(*Stylized Silhouette edge Renderer, SSR*)에서는 효과적인 렌더링 결과를 얻기 위해 아래와 같은 단계를 수행하게 된다.

1. 시점(view point)에서 보이는(visible) 실루엣 에지를 찾는다.
2. 위 단계에서 찾아진 실루엣 에지들을 파라미터화한다.
3. 실루엣의 탄젠트(tangent) 벡터와 법선(normal) 벡터사이의 거리를 사용하여 실루엣 브러쉬 함수를 정의한다.
4. 위 파라미터화된 브러쉬 함수를 이용하여 여러가지 스타일의 실루엣을 렌더링한다.

3.2 실루엣 에지 추출

그림 1에서 보듯이 주어진 시점 v에서 보이는 다각형(front-facing)과 보이지 않는 다각형(back-facing)을 계산한다. 주어진 이 두 다각형이 교차되는 것이 실루엣 에지가 된다.

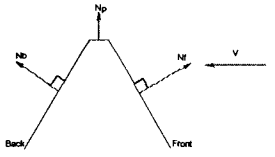


그림 1. 실루엣 에지 추출(Silhouette edge Detection)

실루엣 에지를 추출하기 위해 다음과 같은 두 가지 주요한 과정을 수행한다.

- 주어진 시점에서 보이는 다각형과 보이지 않는 뒤편(behind)의 다각형을 계산한다.
- 시점 벡터(viewing vector) V와 다각형의 법선벡터 N의 dot product $N \cdot V > 0$ 이면 보이는 다각형이고 $N \cdot V < 0$ 이면 보이지 않는 다각형이 된다. 만약 $N \cdot V = 0$ 이면 시점 방향에 수직인 경우이다.
- 주어진 두 다각형이 교차되는 것이 실루엣 에지가 된다.

본 논문에서는 수정된 Appel 알고리즘을 사용하였고, 실루엣 에지 추출을 위해 한 에지에 인접한 두 삼각형의 법선벡터 계산을 빠르게 하기 위해 *Half Edge* 자료구조를 이용하여 구현하였다.

다음은 본 논문에서 사용된 기하학적인 요소에 대한 자료구조들이다. **CHalfEdgeTable** 은 입력 메쉬에 대한 Half Edge 기반 정보를 포함한 형태이다. 이 자료 구조를 이용하여 실루엣 계산을 수행하게 된다.

```
struct CVertex {
    float x, y, z;
};
struct CHalfEdge {
    int origin;
    int twin;
};
struct CHalfEdgeTable {
    int numVertices;
    int numTriangles;
    CVertex *vertices;
    CHalfEdge *halfEdges;
};
```

브러쉬 함수를 이용하여 렌더링하기 위해서 주어진 입력 메쉬에 대해 해당 메쉬에 대한 Half Edge 정보를 생성한다. 생성을 위해서 필요한 정렬과정을 거치므로 Half Edge 정보를 생성하는 데에는 $O(n \log n)$ 시간 복잡도가 요구된다.

3.3 실루엣 에지 파라미터화

본 단계에서는 위 단계에서 수행 되어 얻어진 모든 실루엣 에지에 대해 하나씩 파라미터화하게 된다. 각 실루엣 에지의 끝점을 각각 다른 크기로 조절하여 하나의 선분(line)으로 파라미터화하게 된다. 이렇게 함으로써, 실제 사람이 스케치하는 듯한 랜덤한 효과를 얻을 수 있다.

3.4 파라미터화된 브러쉬 함수

브러쉬 함수(*Brush Function*)는 아주 기본적인 레벨(low level)에서 특정 브러쉬가 어떻게 에지를 표현할 것인가를 정의한 함수를 말한다. 전 단계에서 렌덤한 효과를 얻기 위해 파라미터화 된 각 실루엣 에지에 대해 법선 벡터 뿐만 아니라 에지의 탄젠트 벡터를 파라미터화 할 수 있다. 탄젠트와 법선 벡터는 각 정점에 대한 벡터 옵셋(offset)에 대한 브러쉬 함수와 함께 사용되어 진다. 다음 수식은 t번째 정점에서의 브러쉬 함수에 대한 수식이다.

$$q(t) = p(t) + v_x(t) \cdot p'(t) + v_y(t) \cdot n(t)$$

여기서,$q(t)$는 t에서의 브러쉬 스트로크 지점, $p(t)$는 원래의 정점의 좌표위치, $v_x(t)$와 $v_y(t)$는 각각 브러쉬 함수의 x, y 좌표값, $p'(t)$, $n(t)$는 에지의 탄젠트 값과 법선 벡터이다. 표 1은 본 논문에서 사용된 브러쉬 함수의 종류와 기능을 정리한 것이다.

브러쉬 함수	기능
Charcoal($v_c(t)$)	높은 주파수를 갖는 톱니모양 생성
Lazy($v_l(t)$)	낮은 주파수의 포물선 형태를 생성
Hand($v_h(t)$)	법선과 탄젠트에 적용되는 적은 노이즈
Rough($v_r(t)$)	법선에만 적용되는 높은 노이즈

표 1. 브러쉬 함수와 기능

표에서 설명되어진 각 브러쉬 함수는 아래와 같은 수식으로 정의되어 진다.

$$v_c(t) = t/C * dist_z$$

$$v_l(t) = t/L * dist_z$$

$$v_h(t) = (r * dist_z)/\alpha$$

$$v_r(t) = (r * dist_z)/\beta$$

여기서, C, L은 상수, α, β는 임계값이며, $r = rand()/MAX_{rand}$이고 $dist_z = Z_{max} + |Z_{min}|$이다.

Algorithm 1 RenderWithBrush

```
for each halfEdge i do
  This = CalculateDotProduct(i →origin);
  Twin = CalculateDotProduct(i →twin);
  if (Twin <=0 and This >=0) or (Twin >=0 and This <=0)
  then
    ParameterizeEdge(i);
    CalculateBrushFunction(i);
    DrawBrushEdge(i);
  end if
end for
```

알고리즘 1은 브러쉬 함수를 이용해 렌더링하는 알고리즘이다. 여기서, CalculateDotProduct 함수는 현재 시점 벡터와 해당 삼각형의 법선벡터를 dot product한 결과를 구한다. 에지에 인접한 두 삼각형의 dot product한 결과가 서로 부호가 다르거나 0이면 실루엣 렌더링을 수행하게 된다. 정점의 갯수가 n개인 메쉬에 대해서 $3n$개의 에지가 존재하므로 전체 Half Edge의 갯수는 $6n$개가 된다. 따라서, Half Edge가 구성된 이후 위 알고리즘 1의 시간 복잡도는 $O(n)$이다.

4. 실험결과

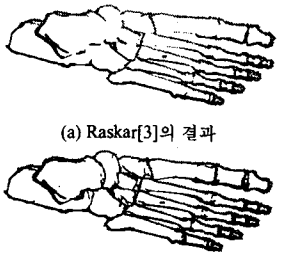
본 논문의 실험은 PentiumII-266MHz(M.M : 128MB)에서 OpenGL 라이브러리를 사용하여 C++로 구현하였다. 데이터는 Standford Graphics Lab.과 CMU에서 획득하여 실험하였다. 그림 2는 뼈 모델의 메쉬모델(a)과 Lazy 브러쉬 함수를 적용하여 렌더링한 결과를 보여 주고 있다. 실루엣 렌더링 시스템(*SSR*)은 브러쉬 함수를 이용하여 실루엣 에지를 다양한 스타일로 표현하고 스케일 조절이 가능하다.

(a) 메쉬 모델 (b) 실루엣 랜더링 결과

그림 2. 메쉬 모델과 실루엣 렌더링 결과(Lazy 브러쉬 이용)

그림 3은 목탄 효과의 실루엣 렌더링 결과를 비교하기 위해 Raskar가 제안한 기법[3]의 결과(a)와 본 논문의 Charcoal 브러쉬 함수를 이용한 결과(b)를 비교 수행하였다. 결과에서 볼 수 있듯이 본 논문에서 제안한 *SSR* 방법이 좀 더 좋은 실루엣 결과를 얻을 수 있음을 알 수 있다. 객관적인 비교를 위해 Raskar가 제안한 방법을 간단히 요약한다.

- 목탄으로 그린 그림은 선의 굵기가 일정하지가 않기 때문에 실루엣의 굵기를 다르게하여 목탄으로 그린 효과를 나타내어야 한다. Raskar는 목탄효과 렌더링을 위해 Rossignac이 제안한 'Less than or Equal'방법을 사용하였는데 이 방법은 에지의 굵기가 카메라로 부터 멀리 떨어져 있는 에지일수록 흐려져 있는 정도가 크게 되므로 굵게 표현되는 단점이 생기므로 거리의 평균값에 의해 굵기를 표현하였다.
- Rossignac의 방법으로 굵게된 에지는 실제로 다각형으로 이루진것이기 때문에 다각형의 에지들 사이에 간격이 생기는 단점을 가진다. Raskar는 다각형의 에지들 사이에 간격이 생기는 단점을 없애기 위해서 선의 굵기를 증가시켰다. 하지만 모델의 크기를 크게 하면 여전히 간격이 존재하므로 자연스러운 효과를 나타내기가 어렵다.



(a) Raskar[3]의 결과

(b) 본 논문의 결과(Charcoal 브러쉬 함수 이용)

그림 3. 뼈 모델의 목탄 효과 실루엣 렌더링 결과 비교

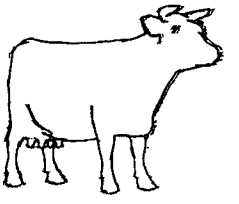


그림 4. Rough 브러쉬 함수를 이용해 렌더링한 소 모델

그림 4는 소 모델에 대해서 Rough 브러쉬 함수를 적용하여 렌더링한 결과를 보여주고 있다.

표 2는 실험에서 사용된 두 데이터에 대한 모델정보와 Half Edge수와 실루엣 에지수를 보여주고 있다.

Model	정점	삼각형	Half Edge	실루엣 에지
뼈 모델	2154	4204	12612	1736
소 모델	2904	5804	17412	1554

표 2. 데이터 통계

5. 결론 및 향후과제

본 연구는 3차원 다면체 삼각형 메쉬 모델을 이용한 비사실적 렌더링을 위해 실루엣 에지를 찾고 파라미터화된 브러쉬 함수를 적용하여 다양한 스타일로 렌더링하는 알고리즘을 제안하고 구현하였다. 향후 연구 과제로는 실루엣 에지 렌더링시 모든 에지를 다 검색하여 실루엣 에지를 찾는것이 아니라 속도 측면을 고려하여 속도를 향상시킬수 있는 방법에 관한 연구와 3차원 일러스트레이터나 카툰 표현에 활용이 이루어져야 할 것이다.

참고 문헌

[1] A. Appel. The notion of quantitative invisibility and the machine rendering of solids. In *ACM National Conference '67 Proc.*, pages 387–393, 1967.

[2] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, , and J. F. Hughes. Real-time nonphotorealistic rendering. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, Aug. 1997.

[3] R. Raskar and M. Cohen. Image precision silhouette edges. In *ACM Symposium on Interactive 3D Graphics '99*, April. 1999.

[4] J. Rossignac and M. van Emmerik. Hidden contours on a frame-buffer. In *Workshop on Computer Graphics Hardware, Eurographics ('92 Proceedings)*, pages 31–38, Sept. 1992.

[5] M. P. Salisbury, S. E. Anderson, R. Barzel, , and D. H. Salesin. Interactive pen-and-ink illustration. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 101–108, July. 1994.