실시간 영상 공간 해칭

GPU 기반 실시간 픽셀 단위 영상공간 해칭

Real-time Image-space Hatching

김용진, Yongjin Kim*, 이승용, Seungyong Lee**

요약 ˇ ˇ 해칭은 물체의 곡률방향을 따라 평행한 선 스트로크를 그려 모양과 음영을 표현하는 효과적인 예술적 기법이다. 본 논문에서는 주어진 스트로크 방향으로 선 스트로크를 그리는 실시간 픽셀단위 영상공간 방법을 제시한다. 본 해칭 방법은 화면 위에서 직접 적용되기 때문에 복잡한 구조를 갖는 3 차원 장면을 효율적으로 실시간에 그려낼 수 있다. 우리는 본 방법을 GPU 의 픽셀 쉐이더를 이용하여 구현한다.

Abstract ˇ ˇ Hatching is an effective artistic tool for conveying shape and shading by placing parallel line strokes on drawing objects. We present a simple and effective per-pixel image-space hatching method to draw line strokes using given stroke directions. Our hatching method directly runs on the screen and it can efficiently render highly complex scenes in hatching styles. We implement the algorithm using a pixel shader in a modern GPU.

핵심어: non-photorealistic rendering, line-art illustration, hatching, real-time rendering

본 연구는 문화체육관광부 및 정보통신연구진흥원의 IT 원천기술개발사업의 일환으로 수행하였음. [2008-F-031-01, 영상 및 비디오 콘텐츠를 위한 계산사진학 기술 개발]

*주저자 : 포항공과대학교 컴퓨터공학과 박사과정; e-mail: sldk@postech.ac.kr

**공동/교신저자 : 포항공과대학교 컴퓨터공학과 교수; e-mail: leesy@postech.ac.kr

1. 서론

해칭(hatching)은 복잡한 물체들의 모양, 음영을 효과적으로 표현할 수 있는 예술적 기법이다. 펜화 드로잉(drawing), 연필 드로잉 등이 미술에서 찾아볼 수 있는 대표적인 해칭의 예이며, 평행한 선 스트로크(line stroke)들을 곡률 방향을 따라 그려 물체를 표현하게 된다. 그 동안 비사실적 렌더링 연구에서 3 차원 물체를 해칭으로 표현하기 위한 다양한 기법들이 개발되어 왔으며, 최근 실시간으로 해칭을 하는 기법들도 제시되었다 [1,2,3].

Praun [1] 등은 3 차원 메쉬(mesh)를 부분적으로 겹치는 여러 장의 차트(chart)로 나눈 뒤 각각을 파라미터화(parameterization) 하여, 파라미터화된 공간에 선 스트로크 텍스쳐(texture)를 매핑(mapping)하는 방법을 제시하였다. Webb [2] 등은 [1]의 방법에서 좀 더 세밀한 밝기의 표현을 얻기 위한 방법을 제시하였다. 이들 방법은 좋은 결과를 보여주지만, 렌더링 전 메쉬의 복잡한 파라미터화가 선행되어야 한다는 단점이 있다.

Lee [3] 등은 연필 렌더링을 위하여 파라미터화가 필요없는 해칭 방법을 제시하였다. Lee 등은 이 방법을 이용해 복잡한 선행과정 없이 양질의 연필화 렌더링결과를 얻었다. 그러나 이 방법은 3 차원 메쉬의 삼각형 단위로 선 스트로크 텍스쳐를 매핑하는 과정에서 삼각형의 경계부분에서 선 스트로크가 끊어질 수 있다는 한계점이 있다.

본 논문에서는 기존의 방법들과 비등한 수준의 결과를 얻으면서 복잡한 파라미터화가 필요하지 않고, 영상공간(image-space)에서 끊김없는 선 스트로크를 얻을 수 있는 실시간 영상공간 해칭 알고리즘을 제시한다. 이 알고리즘은 각 픽셀의 밝기와 선 스트로크의 방향이 주어졌을 때, GPU 의 픽셀 쉐이더(pixel shader) 한 단계(pass)만을 이용하여 간단하게 구현될 수 있다. 본 알고리즘은 해칭을 위해 3 차원 메쉬의 정보를 이용하지 않고, 영상공간에서 직접 수행되기 때문에, 계산적 복잡도가 3 차원 메쉬의 복잡도에 의존하지 않으며, 복잡한 3 차원 장면들을 손쉽고 빠르게 해칭 스타일로 렌더링 할 수 있다.

2. 실시간 영상공간 해칭 과정

3 차원 물체를 실시간에 해칭으로 표현하기 위해서는 물체의 음영(shading)과 물체의 곡률방향(principal curvature direction)을 계산하고, 이 정보들과 2 차원 선 스트로크 텍스쳐를 이용하여 해칭 알고리즘을 적용하게 된다. 물체의 음영을 계산하기 위해 본 논문에서는 퐁 쉐이딩(Phong shading)과 분산 그림자 맵(variance shadow mapping)을 사용하였고 [4], 곡률 계산을 위해서는 최근 제시된 포컬 곡면(focal surface)방법을 이용하였다 [5]. 물체의 음영과 곡률 계산은 반드시 이러한 방법으로 수행될 필요는 없으며, 알려진 다양한 다른 방들로 대체될 수 있다.

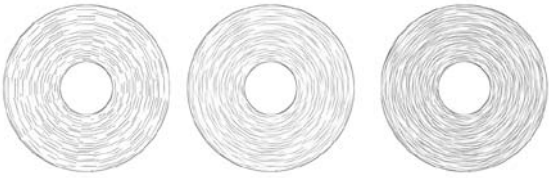

**그림 1. 영상공간 해칭 각 단계 별 결과.
(좌) 잘라 붙이기, (중) 스트로크 텍스쳐 블렌딩,
(우) 오프셋 파티션을 이용한 겹쳐 그리기**

한편, 선 스트로크 텍스쳐를 만드는 것은 [1]에서 토널 아트 맵(tonal-art map)이라는 기법이 처음 소개되었으며, [2]에서는 좀 더 정확한 음영표현을 위해 이를 확장하였다. [3]에서는 [2]의 텍스쳐 구조에서 밉맵(mipmap)을 제외한 구조를 사용하였다. 본 논문에서는 [3]에서 사용된 스트로크 텍스쳐 구조를 사용한다. 이 텍스쳐 구조는 여러 장(slice)의 스트로크 텍스쳐가 쌓아 올려진 3 차원 볼륨(volume) 형태로 되어 있으며, 각 장은 밝기 순으로 정렬되어 있다.

본 논문에서 제시하는 영상공간 해칭 알고리즘은 주어진 픽셀의 밝기와 곡률방향이 있을 때, 위와 같은 구조의 선 스트로크 텍스쳐를 적절히 참조하는 텍스쳐 좌표(texture coordinate)를 픽셀 단위로 결정하여 실시간에 선 스트로크 텍스쳐를 화면에 그려낸다. 본 알고리즘은 부드럽게 이어지는 선 스트로크를 생성하는 과정에서 인접한 픽셀들을 참조하지 않고 각 픽셀에서 독립적으로 텍스쳐 좌표를 결정한다. 이러한 특성 때문에 본 알고리즘은 픽셀 쉐이더에서 매우 효과적으로 구현 될 수 있다.

3. 실시간 영상공간 해칭 알고리즘

3.1 선 스트로크 텍스쳐 잘라 붙이기

본 영상공간 해칭 알고리즘의 첫 번째 단계는 영상을 비슷한 스트로크 방향을 갖는 픽셀들로 분할(partitioning)하여 각각의 영역에 그 영역에 대표되는 방향을 따라 스트로크 텍스쳐를 잘라 붙이는 것(cut-and-paste)이다.

$$(u,v)=(\cos\theta^{Q}\cdot x-\sin\theta^{Q}\cdot y,\sin\theta^{Q}\cdot x+\cos\theta^{Q}\cdot y) \quad (1)$$

우선 각 픽셀마다 주어진 선 스트로크의 방향, 즉 곡률방향에 해당하는 각도 $\theta \in [0, \pi)$ 를 12~24 개 사이의 정해진 개수로 양자화(quantization) 한다. 이렇게 하면 영상을 비슷한 곡률방향을 갖는 픽셀들로 분할한 것과 같은 효과를 얻을 수 있다. 이렇게 양자화된 각도 θ^{Q} 를 이용해 픽셀의 위치좌표 (x, y)를 회전시키면, 같은 양자화된 각도를 갖는 각각의 파티션은 마치 그 파티션 자체가 회전된 것과 같은 효과를 갖게 된다. 이렇게 회전시킨 좌표를 선 스트로크 텍스쳐의 참조 좌표 (u, v)로 이용하게 되면 (수식 1), 선 스트로크 텍스쳐가 마치 해당하는 영역에 잘라 붙여진 것과 같은 효과를 갖게 되며, 선 스트로크의 방향은 양자화된 곡률 방향을 따르게 된다.

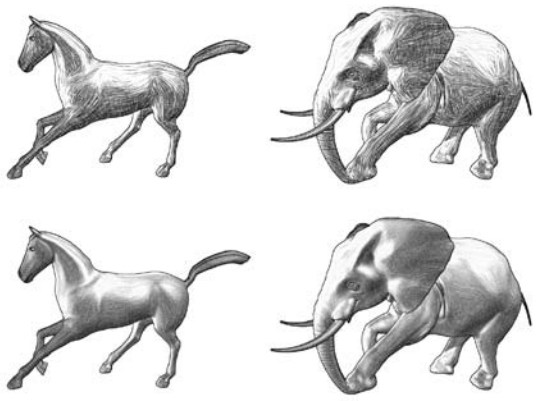

그림 2. 개별 3 차원 물체 해칭 결과 (51fps), (상) 펜 스트로크를 이용한 해칭, (하) 연필 스트로크를 이용한 해칭

각 픽셀에서 위와 같이 텍스쳐 좌표를 결정할 때, 인접한 픽셀들을 참조하지 않고 독립적으로 결정하는 것을 주목할 필요가 있다. 이러한 특성 때문에 위 알고리즘은 픽셀쉐이더에서 매우 쉽고 효율적으로 구현될 수 있다. 그럼에도 불구하고 영상의 분할된 각 영역에는 그 영역을 가장 잘 대표하는 선 스트로크 방향을 따라 선 스트로크가 자연스럽게 연결되어 매핑된 것을 볼 수 있다 (그림 1(좌)).

3.2 스트로크 텍스쳐 블렌딩

위의 방법을 사용하면 분할된 각 영역에 픽셀단위로 선 스트로크를 효과적으로 매핑할 수 있지만, 각 파티션의 경계(boundary)부분에서 스트로크가 끊어지는 문제가 있다. 이를 해결하기 위해 우리는 각 파티션에 대하여 인접한 파티션과 블렌딩(blending)을 적용한다.

우선 각 픽셀의 좌표 (x, y)를 그 픽셀의 양자화된 각도와 인접한 두 양자화된 각도를 이용해 세 개의 텍스쳐 좌표 (u_k,v_k)를 얻는다.

$$(u_k, v_k) = (\cos\theta_k \cdot x - \sin\theta_k \cdot y, \sin\theta_k \cdot x + \cos\theta_k \cdot y), \quad (2)$$

k=−1,0,1 이고, $\theta_k = \theta^Q + \Delta\theta$ 이다. 세 개의 텍스쳐 좌표를 이용해 얻은 선 스트로크 텍스쳐 값을 $f(u_k,v_k)$라고 하였을 때, 우리는 세개의 텍스쳐 값에 가중치를 적용하여 식 3 과 같이 블렌딩 한다.

$$I^{blended} = \sum_{k=-1}^{1} w_k f(u_k, v_k),$$

$$w_k = \begin{cases} 1 - \dfrac{|\theta - \theta_k|}{\Delta\theta}, & \text{when } |\theta - \theta_k| < \Delta\theta \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

그림 3. 3 차원 장면 해칭 결과. 영상공간 해칭은 3 차원 장면을 영상공간에서 직접 다루어 간단히 해칭 결과를 얻을 수 있다. (45fps)

위와 같은 과정은 2.1 에서 수행했던 결과에서 인접한 파티션으로 스트로크가 부드럽게 연장되는 효과를 얻게 하며, 그 결과로 파티션의 경계가 눈에 띄지 않게 된다 (그림 1(중)).

3.3 오프셋 파티션을 이용한 겹쳐 그리기

그러나 이렇게 했을 경우, 각각의 스트로크는 부드럽게 이어지지만, 인접한 파티션 간 양자화된 각도는 부드럽게 변하지 않으므로, 기존의 부드러운 곡률 방향의 변화가 직선 스트로크들로 끊겨 표현되는 문제점이 있다. 이 문제를 해결하기 위해 우리는 사람이 부드럽게 굽어진 스트로크를 그리기 위해 여러 개의 스트로크를 겹쳐 그리는 것에 착안하였다. 이를 적용하기 위해 우리는 양자화 하는 단위의 반에 해당하는 오프셋(offset)을 주고 곡률방향을 새로 양자화 한 뒤, 3.1 과 3.2 에서 수행한 과정을 똑같이 수행하여, 원래의 결과와 반씩 섞어준다. 그림 1(우)는 이와 같이 수행한 최종적인 결과를 보여준다.

4. 결과

본 논문에서는 실시간 영상공간 해칭 시스템을 DirectX 10 의 shader model 4 를 사용하여 구현하였으며, 2.4Ghz Intel Core2 Duo 6600 CPU 와 NVidia Geforce 8800 ULTRA 그래픽 카드를 이용하여 테스트 하였다.

그림 2 는 본 논문에서 제시하는 영상공간 해칭을 개별 3 차원 물체에 적용한 결과를 보여준다. 각각의 이미지는 640x480 해상도에서 초당 약 51 프레임의 속도로 렌더링 되었다. 본 논문에서 제시된 방법은 매우 간단하고 구현하기 쉬운 방법임에도 불구하고, 물체의 표면에서 선 스트로크가 끊기지 않으며, 기존에 제시된 다른 방법 [1,2,3] 과 비등한 수준의 결과를 보여준다.

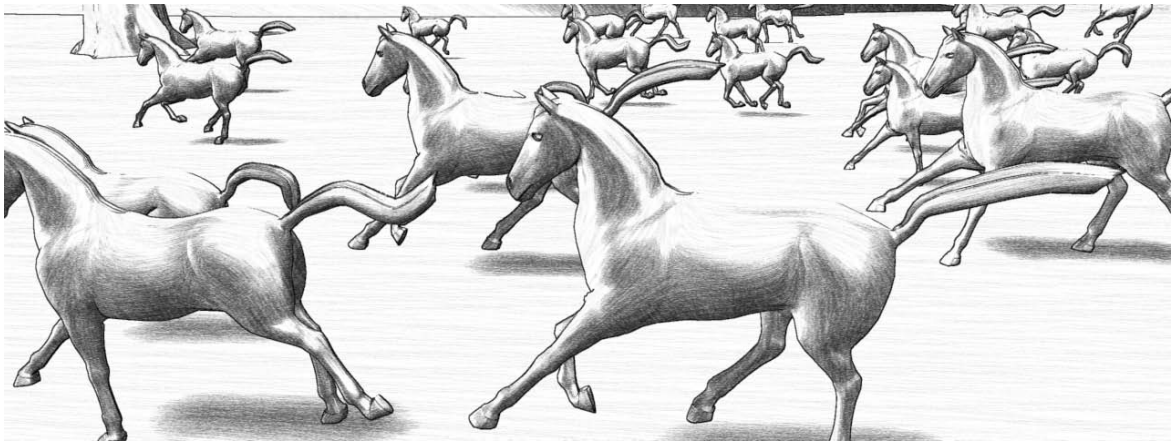

그림 4. 복잡한 3 차원 장면 해칭 결과. 영상공간 해칭은 다수의 삼각형을 갖고 있는 복잡한 3 차원 장면을 쉽게 실시간에 렌더링 할 수 있다. (21 fps)

그림 3 은 3 차원 장면을 본 알고리즘을 이용해 해칭한 결과를 보여준다. 이 장면은 약 35,000 개의 삼각형으로 이루어져 있으며, 640x480 의 해상도에서 초당 약 45 프레임으로 렌더링 되었다. 기존의 방법 [1,2]는 여러 개의 물체로 이루어진 3 차원 장면을 렌더링 하기 위해서 각각의 물체를 개별적으로 파라미터화 하는 과정을 거쳐야 하지만, 본 알고리즘은 이와 같은 복잡한 과정을 거칠 필요 없이 픽셀쉐이더 한 단계만을 이용해 이와 같은 결과를 얻을 수 있다.

그림 4 는 다수의 삼각형으로 이루어진 복잡한 3 차원 장면을 본 알고리즘을 이용해 렌더링한 결과를 보여준다. 이 장면은 약 1,100,000 개의 삼각형으로 이루어져 있으며, 1280x480 해상도에서 약 21fps 로 렌더링 되었다.

5. 토의 및 향후 연구

본 논문에서는 실시간 픽셀단위 영상공간 해칭방법을 제시하였으며, 이 방법은 기존에 제시된 방법들과 비등한 결과를 보여준다. 본 논문에서 제시된 방법은 매우 간단하여 쉽게 픽셀 쉐이더로 구현될 수 있다. 또한 본 방법은 영상공간 알고리즘으로 계산의 복잡도가 3 차원 입력 메쉬의 복잡도에 의존하지 않기 때문에, 복잡한 3 차원 장면을 쉽게 다룰 수 있다. 이러한 장점으로 본 논문의 방법은 게임 등 다양한 컴퓨터 그래픽스 응용분야에서 3 차원 컨텐츠를 스타일화 하여 렌더링(stylistic rendering)하는 데 쉽게 응용될 수 있을 것이다.

본 알고리즘에서 픽셀이 갖는 선 스트로크 방향을 양자화하여 영상공간에서 파티션을 얻을 때, 곡률의 변화가 큰 부분은 파티션이 매우 잘게 나누어 질 수 있다. 그 결과로 이러한 부분에서는 긴 곡선의 스트로크 대신 짧은 여러 개의 스트로크들이 그려지는 한계점이 있을 수 있다.

본 알고리즘을 실시간 네비게이션이나 동적 환경에 적용하는 경우, 스트로크들은 시간적으로 일관된 움직임(temporal coherency)을 보여주지만, 다른 영상공간 비사실적 렌더링 연구와 마찬가지로 샤워문 효과(shower-door effect)라는 알려진 한계점을 보일 수 있다 [3]. 샤워문 효과는 영상에 매핑된 선 스트로크들이 물체의 움직임과 일관되게 움직이지 않을 수 있는 경우를 말하며, 이것을 해결하는 것은 영상공간 비사실적 렌더링 연구의 도전적인 향후 연구가 될 것이다.

기존의 3 차원 메쉬구조를 사용하는 해칭방법[1,2,3]들과 달리 본 알고리즘은 픽셀단위의 밝기와 곡률방향만을 가지고 각 픽셀에서 직접 텍스쳐좌표를 결정하여 선 스트로크를 그린다. 이와 같은 특징은 본 알고리즘이 3 차원 물체의 렌더링 뿐 아니라 2 차원 플로우(flow)를 가시화(visualization)하거나 대화식 드로잉 시스템(interactive drawing system)으로 응용될 수 있는 가능성을 보여준다. 본 알고리즘을 위와 같은 시스템으로 확장하는 것은 중요한 향후 연구가 될 것이다.

참고문헌

[1] E. Praun, H. Hoppe, A. Finkelstein, "Real-Time Hatching", ACM SIGGRAPH 2001, pp. 579-584, 2001.

[2] M. Webb, E. Praun, A. Finkelstein, H. Hoppe, "Fine Tone Control in Hardware Hatching", Proc. International Symposium on Non-Photorealistic Animation and Rendering 2002, pp. 53-58, 2002.

[3] H. Lee, S. Kwon, S. Lee, "Real-Time Pencil Rendering", Proc. International Symposium on Non-Photorealistic Animation and Rendering 2006, pp. 37-45, 2006

[4] W. DONNELLY, A. LAURITZEN, "Variance shadow maps", Proc. Symposium on Interactive 3D Graphics and Games 2006, 161-165, 2006

[5] J. YU, X. YIN, X. GU, L. MCMILLAN, S. GORTLER, "Focal surfaces of discrete geometry", Proc. Eurographics Symposium on Geometry Processing 2007, 23-32, 2007