효과적인 Fur 렌더링을 위한 적응적 시스템

혼합 렌더링을 이용한 빠른 Fur 렌더링 방법

An Adaptive System for Effective Fur rendering

김혜선, Hyesun Kim*, 반윤지, Yunji Ban**, 이충환, Chunghwan Lee***,
남승우, Seungwoo Nam****, 최진성, Jinsung Choi*****, 오준규, Junkyu Oh******

요약 ˘ ˘털 렌더링은 사실적인 렌더링을 위해 많은 수의 털 데이터를 처리해야 하는 어려움이 있다. 대량의 털 데이터를 렌더링함에 있어서 가장 어려운 점은 렌더링 시간이 많이 걸린다는 점이다. 기존의 털 렌더링 방법은 털을 원통형의 실린더로 간주하고 2D 형태의 리본으로 변환하고 삼각화하여 렌더링하는 방법이다. 하지만 이 방법은 언더 샘플링(under sampling) 문제가 있고 렌더링 시간이 오래 걸린다는 단점이 있다. 이런 단점을 개선하기 위해서 이 논문에서는 새로운 알고리즘을 제안하였다. 털을 굵기에 따라 나누고 굵은 털과 가는 털에 각각의 렌더링 방법을 사용함으로써 렌더링 속도를 개선하였다. 또한 전체 렌더링 프레임워크에 대한 제안을 함으로써 보다 효과적인 렌더링을 수행할 수 있다.

Abstract ˘ ˘Fur rendering is difficult in that there are huge numbers of objects and it takes so much time. The previous method considers fur as cylinder, transforms it into 2D ribbon, triangulates and commits rendering. But this method has problem like under sampling and takes rendering time so long. To resolve these shortcuts we proposed new algorithm. We divide fur into thick and thin fur and we applied adaptive rendering methods for each type of fur. Also we can perform an effective rendering according to the proposed rendering framework.

핵심어: fur rendering, adaptive rendering, ribbon rendering, alpha rendering, deep render buffer: 렌더링, 털 렌더링, 리본, 알파 렌더링, 딥 렌더 버퍼

- 본 연구는 정보통신부 및 정보통신연구진흥원의 IT 신성장동력핵심기술개발 사업의 일환으로 수행하였음. [2006-S-044-01, 기능 확장형 초고속 렌더러]

*김혜선 : 한국전자통신연구원 선임연구원 e-mail: hsukim@etri.re.kr

**반윤지 : 한국전자통신연구원 연구원 e-mail: banyj@etri.re.kr

***이충환 : 한국전자통신연구원 선임연구원; e-mail: brian@etri.re.kr

****남승우 : 한국전자통신연구원 선임연구원; e-mail: swnam@etri.re.kr

*****최진성 : 한국전자통신연구원 선임연구원; e-mail: jin1025@etri.re.kr

******오준규 : 레이소프트 ; e-mail: ohjunkyu@gmail.com

1. 서론

실제 세상에서, 온 몸이 털로 덮여 있는 동물들은 셀 수 없이 많은 수의 가느다란 털들이 털 밑의 피부가 안보일 정도로 촘촘하게 박혀있다. 이를 3D 가상 세계로 옮긴다면, 털이 달린 가상의 동물들 또한 온 몸이 털들로 빽빽하게 뒤덮여야 한다. 그러나 이러한 털 데이터는 털의 개수뿐만이 아니라 처리 방법에 있어서도 쉽게 처리할 수 있는 것이 아니다. 특히 털을 표현하기 위해 필요한 엄청난 렌더링 시간은 사용자들이 털 표현을 기피하게 하는 데 큰 역할을 했다.

털 표현의 경우 실제와 비슷하게 표현하기 위해서는 수백만에서 수천만 개 이상의 털을 렌더링할 수 있는 기술이 필요하다. 품질이 우수한 가상의 캐릭터를 만들기 위해서는 점점 더 많은 수의 털을 필요로 하게 된다. 이런 식으로 털의 개수가 많아지면서 렌더링하는 데 걸리는 시간이 더 많이 필요할 수밖에 없다. 실제로 렌더링 시간의 한계로 털을 필요한 만큼 사용하지 못하는 경우도 많은 만큼 털의 렌더링 시간은 콘텐츠 제작에 있어서 결정적인 문제가 되고 있다.

이전에도 털의 렌더링 방식에 대한 여러 연구가 이루어졌지만 개체가 많은 털의 특성상 렌더링 시간을 획기적으로 단축하기에는 문제가 있었다.

이 논문에서는 기존의 방식을 보완하여 털을 좀 더 빨리 렌더링할 수 있는 새로운 알고리즘을 제안한다. 또한 실제 구현을 위한 전체 렌더링 프레임워크와 렌더링 과정에서 필요한 데이터 저장 방식을 설명하고자 한다.

2. 연구동향

털 렌더링에 있어서 중요한 이슈는 크게 5 가지로 나뉘어 진다. 털의 형상을 표현하기 위한 방법과 색깔값을 결정하기 위한 쉐이딩 방법, 그림자 음영을 표현하기 위한 쉐도우 표현법이 있다. 그리고, 이러한 전체적인 렌더링 과정을 가속하기 위한 방법이 연구되고 있다[1, 2]. 최근에는 좀 더 사실적인 털 표현을 위해 전역 조명 기법을 가미하는 방법들이 연구되기 시작했다.

2.1. 털 표현 방법

털의 형상을 표현하기 위한 방법은 다시 명시적 방법(explicit method)과 암시적 방법(implicit method)으로 나뉘어 지는데, 털 지오메트리 모델을 어떻게 표현하느냐에 따라 결정된다. 삼각형이나 라인 기반의 렌더러에서는 명시적 방법을 사용하며, 볼륨 기반의 렌더러에서는 암시적 방법을 사용한다.

명시적 표현법에서는 보통 한 가닥의 털을 곡선형의 실린더(curved cylinder)로 표현한다. 하지만 각각의 털 가닥들을 모두 곡선형의 실린더로 표현하기 위해서는 많은 양의 지오메트리 데이터가 필요하므로, 털을 효율적으로 표현하기 위해 많은 연구가 진행되어 왔다. Watanabe 는 털을 삼각 기둥 모양의 프리즘 형태로 표현했으며[3], Neulander 는 항상 카메라 쪽을 바라보는 리본 형태의 평면으로 털을 더욱 더 간소화하여 표현하였다[4]. 위 두 가지 방법이 삼각형 기반의 표현 방법이라면, LeBlanc 은 털 한 가닥이 한 픽셀보다 얇은 오브젝트라는 가정 하에 라인 기반의 알파 렌더링 표현 방법을 제안하였다[5]. 그 후, Nishita 는 원통형의 물체를 표현하기에 좀 더 효율적으로 차별화된 스캔라인 알고리즘을 제안하기도 하였다[6].

털을 볼륨 모델 형태로 표현하는 암시적 표현법은 Kajiya 가 제안한 3D texel 렌더링 방법이 있다[7]. 이 방법에서는 털 데이터를 밀도를 가지는 3D 볼륨 형태의 텍스쳐로 표현하고 렌더링한다. Yang 이 소개한 클러스터 헤어 모델(Cluster Hair Model) 또한 암시적 표현 방법의 좋은 예이다[8].

2.2. 털 쉐이딩

앞서 말한 Kajiya 의 texel 렌더링 시스템에서는 또 한가지 중요한 제안을 하고 있는데, 바로 털 쉐이딩을 위한 반사 모델이다[7]. Kajiya 는 정확한 노말(normal) 정보를 가지고 있지 않는 털의 쉐이딩 값을 구하기 위해, 털 커브 라인의 진행 방향에 직교하는 평면에 털 커브 라인을 투영시킨 벡터를 이용하여 쉐이딩 값을 결정하였다. 이 방법은 상당히 오래 전에 제안되었음에도 불구하고 현재까지 상용 털 렌더링 시스템에서 가장 많이 사용되고 있다. 이에 Marschner 는 Kajiya 의 방법을 좀 더 개선하여 물리적으로 좀 더 정확한 반사 모델을 제안하였다. 실제 털을 현미경으로 관찰했을 때, 털들이 얇은 단층 구조로 겹쳐 있다는 점에 착안한 방법이다[9]. 최근에는 좀 더 사실적인 털 표현을 위해 다중 산란(multiple scattering)을 적용한 반사 모델들이 계속해서 연구되고 있다[10, 11].

2.3. 그림자 표현

사실적인 렌더링을 위해 그림자 표현은 없어서는 안 될 중요한 기법이다. 전통적으로 스캔라인 렌더링 방법에서는 그림자 맵(depth map)을 사용해 왔지만 털과 같이 가느다란 원통형의 투명체를 표현하기 위해 이를 그대로 사용하기에는 부적절하다. 픽사 애니메이션 스튜디오의 Lokovic 은 그림자 값을 가시성 함수(visibility function)로 표현하는 방법을 제안했는데[12], 현재 렌더맨 계열의 렌더러들은 이 방법을 사용하여 그림자를 표현하고 있다. 픽셀들 각각의 가시성 함수를 사용하는 Lokovic 의 방법과는 달리 김태영이 제안한 방법은 깊이에 따라 여러 개의 그림자 맵을 생성하여 그 사이 깊이의 그림자 값을 보간해서 좀 더 부드러운 그림자를 렌더링할 수 있도록 하였다[13]. 이후에 Yuksel 은 김태영의 방법을 좀 더 보완하여 적은 개수의 그림자 맵을 가지고도 높은 퀄리티의 그림자 표현을 할 수 있도록 하였다[14].

2.4. 가속기법

털 렌더링에 있어서 가장 어려운 점은 렌더링 시간과의 싸움이다. 털이 입혀진 물체를 표현하기 위해서는 수백만에서 수천만 개의 털을 렌더링해야 하므로 고사양의 하드웨어 시스템과 오랜 시간 기다릴 수 있는 사용자의 인내심이 필요하다. 이에 털 렌더링을 좀 더 빠르게 할 수 있는 가속 기법들이 연구되어 왔는데, 크게 두 가지 방법으로 나뉜다. 털 데이터를 그럴듯하게 보이도록 하는 페이크(fake) 기법과 GPU 를 활용하는 기법이다. 때로는 이 두 가지 기법이 혼용되어 실시간으로 어느 정도의 털 렌더링 퀄리티를 내기도 한다. Goldman 은 간단한 털 모양의 패턴만으로도 그럴듯한 털 표현을 하였고[15], 이는 디즈니사의 '101 마리 달마시안' 영화에 사용되었다. Lengyel 은 텍스쳐 기법을 사용하여 임의의 객체에 실시간으로 털을 렌더링 하는 방법은 소개하였지만[16], 털의 렌더링 퀄리티보다는 실시간으로 렌더링 했다는 점에 더 의의를 줄 수 있다. 실시간 털 렌더링 방법은 NVidia 사에서 적극적으로 개발 중인데, 2004 년 실시간으로 긴 머리털이 바다 속에서 움직이는 것을 직접 시연하기도 하였다[17].

2.5. 전역 조명 기법

전역 조명 처리 작업은 털이 아닌 일반적인 객체를 표현하는 데에도 많은 시간이 걸린다. 그래서, 털과 같이 렌더링 타임이 많이 걸리는 객체에 전역 조명 효과를 적용하는 연구는 최근 들어 조금씩 연구가 시작되는 단계이다. Yuksel 은 털 표현에 전역 조명 효과를 사용하는 연구 결과를 2005 년에 이어 2006 년에 소개하였다[18, 19]. 특히, Yuksel 이 제안한 가시성 계산 방법(visibility calculation method)에 사용하는 이글루(Igloo) 모양의 맵은 간접 조명을 표현하는 데 유용하다[18].

본 논문에서 제안하는 시스템은 털 렌더링의 여러 가지 이슈들 중에서 명시적인 방법으로 털을 표현하는 렌더링 시스템이다. 쉐이딩은 Kajiya 가 제안한 반사 모델을[7] 이용하였으며, Lokovic 의 딥 쉐도우(deep shadow) 기법[12]으로 그림자를 표현하였다.

3. 본론

이번 장에서는 이 논문에서 제안한 내용을 자세히 알아본다. 먼저 기존의 일반적인 털 렌더링 방법들에 대해서 설명하고, 기존의 알고리즘을 개선한 새로운 알고리즘을 소개한다. 다음에는 전체적인 털 렌더링 프레임워크를 설명하고 이 시스템에서 사용한 데이터 저장 방식인 딥 렌더 버퍼(deep render buffer)에 대해 자세히 설명한다.

3.1 Basic

먼저 털을 표현할 때 가장 일반적으로 사용되었던 두 가지 알고리즘을 소개하고자 한다.

가장 일반적인 털 표현 방법은 Neulander 가 소개한 리본 모양으로 삼각화하는 방법이다[4]. 각각의 털들을 두께를 가지는 3D 커브 데이터로 모델링하고 카메라가 보는 방향에서 수직인 리본 형태로 변환한다. 리본 형태의 평면으로 삼각화된 털 데이터들은 일반 오브젝트와 같은 방식으로 렌더링이 가능하다. 실제 대부분의 상용 털 렌더링 시스템들이 이 방법을 사용하고 있다. 리본 삼각화 방식은 가장 일반적으로 사용되고 있지만 털의 굵기가 아주 가늘어서 한 픽셀보다 작을 경우 심각한 언더 샘플링(under sampling) 문제가 발생한다. 즉, 샘플링을 충분히 많이 하지 않는다면 픽셀 내의 털을 제대로 샘플링해 내지 못해서, 전체적인 퀄리티를 떨어뜨리는 결과를 초래하게 된다. 게다가 애니메이션 시에는 해당 픽셀에 털이 갑자기 샘플링 되기도 해서 반짝거리는 플리커링(flickering) 현상이 발생한다. 이런 문제를 해결하기 위해서는 샘플링을 충분히 많이 해야 하는데 샘플링 횟수에 따른 부담은 곧바로 렌더링 시간이 늘어나는 문제로 이어진다.

또 하나의 방법은 LeBlanc 가 제안한 알파 렌더링 방식으로 한 픽셀 안에서 털이 그려질 면적의 비율을 해당 픽셀의 알파값(투명값)으로 계산하는 방식이다[5]. 이 방식은 샘플링이 필요하지 않으므로 언더 샘플링으로 인한 플리커링 현상을 막을 수 있다. 가느다란 털이라도 놓치지 않고 정확하게 표현할 수 있기 때문이다. 또한 샘플링으로 인해 필요한 렌더링 시간을 단축시킬 수 있다는 장점이 있다. 하지만 이 방법은 털이 한 픽셀의 크기보다 가늘 경우에만 사용 가능하다. 한 픽셀 이상의 굵은 털에 적용할 경우 해당 픽셀들의 알파값을 계산하는 경우의 수가 더욱 더 복잡해져서 일반화할 수 없는 단점이 있다.

3.2 Main algorithm

여기서 제안하고자 하는 방법은 3.1 장에서 설명한 기존의 두 가지 방법의 장점만을 택하여 털 렌더링 시스템의 속도와 퀄리티, 두 가지 면에서 개선을 이루어낸 방법이다.

이 알고리즘의 기본 아이디어는 한 픽셀의 크기를 기준으로 굵은 털과 가는 털로 나누어서 처리하는 것이다. 즉, 한 픽셀의 크기보다 굵은 털과 픽셀 크기보다 가는 털로 구분한다. 굵은 털은 기존의 리본 삼각화 방법과 같이 렌더링하고 가는 털은 알파 렌더링 방법처럼 각 픽셀에서 털이 그려져야 하는 면적의 비율을 계산해서 렌더링한다.

굵은 털은 리본 삼각화 방식과 같이 삼각화하여 한 픽셀을 NxN 개의 서브 픽셀로 샘플링하고, 각각 서브 픽셀 정보를 합산한 값이 해당 픽셀의 색깔 값과 투명도 값이 된다. 가는 털의 경우 픽셀 위로 털이 지나가는 영역의 면적 비율을 계산해서 비율에 따른 색깔 값과 투명 값을 그 픽셀에 적용한다. 이 방법은 기존의 일반적인 리본

삼각화 방법에서 가는 털만 골라내어 알파 렌더링 방법을 적용함으로써 더 효과적으로 렌더링을 할 수 있다. 가는 털 렌더링 방법에는 샘플링이 필요 없으므로 렌더링 속도가 향상될 수 있고, 언더 샘플링(under sampling)으로 인해 생기는 오류, 즉 깜빡임 현상을 방지할 수 있다. 따라서 기존의 리본 삼각화 방법에서 굵기가 아주 가는 경우 생길 수 있는 문제점을 자연스럽게 해결할 수 있다.

그러나, 굵은 털을 처리하는 리본 삼각화 방식은 샘플링을 해야 하는 반면, 가는 털을 처리하는 알파 렌더링 방식은 샘플링 없이 직접 렌더링해야 하므로, 두 방식을 한번에 같이 쓰기 위해서는 새로운 시스템 프레임워크(frame work)이 필요하다. 다음 장은 두 가지 렌더링 방법을 혼합해서 쓸 수 있는 프레임워크에 대한 설명이다.

3.3 Framework

털 렌더링 프레임워크는 그림 1 과 같다. 털을 제외한 일반적인 오브젝트들은 털을 렌더링하기 전에 처리하고 털 렌더링은 후처리 방식으로 처리한다.

털은 따로 처리하는데 먼저 굵은 털과 가는 털을 체크하는 부분이 필요하다. 이 두 가지로 나누는 기준은 픽셀 하나의 크기이다. 각각의 털 처리는 3.2 장에서 설명한 알고리즘을 적용한다.

그림 1 을 보면 일반적으로 삼각화 과정을 통해 털이 아닌 일반 오브젝트들과 함께 처리되었던 굵은 털이 후처리 과정에서 처리됨을 확인할 수 있다. 투명값을 가지고 있는 털은 래스터(raster)과정 후에 버퍼상에서 픽셀 단위로 각각 깊이에 따라 정렬되어야 한다. 이때 알파 렌더링 처리 방법으로 렌더링되는 가는 털과 함께 정렬처리 되어야 하므로, 가는 털과 함께 후처리 과정에서 처리 된다. 반면 가는 털은 샘플링을 하지 않는 알파 렌더링 방법을 써야 하므로 일반 오브젝트들과 같이 렌더링할 수 없어서 후처리 과정에서 처리한다.

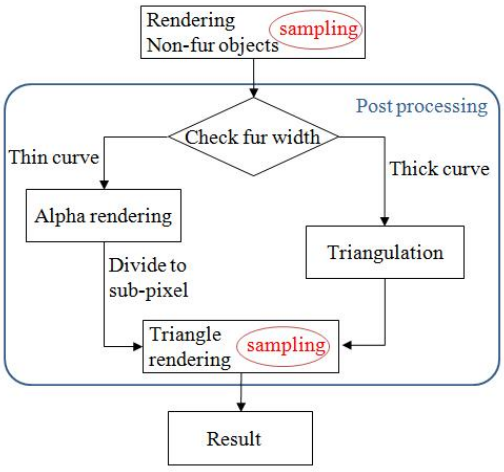

그림 1. 털 렌더링 프레임워크

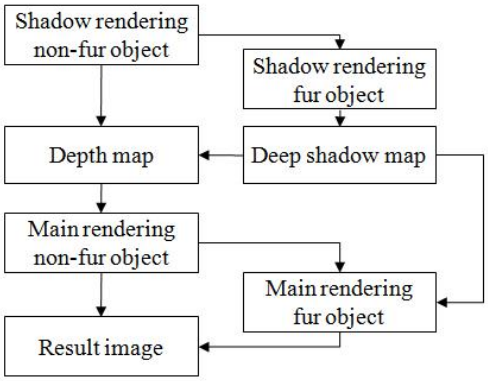

그림 2. 전체 프레임워크

또한 그림 1 을 보면, 가는 털을 먼저 렌더링한 후에, 굵은 털을 렌더링하고 있음이 나타나있다. 이는 알파 렌더링 방법은 한 픽셀을 단위로 털의 면적을 계산하는 반면 리본 삼각화 방법은 샘플링을 위해 각각의 픽셀을 더 작은 단위인 서브 픽셀 단위로 처리하기 때문이다. 한 픽셀을 단위로 가는 털을 그린 후, 각각의 픽셀을 서브 픽셀로 나눠서 굵은 털을 그리게 된다.

여기서 사용한 프레임워크의 렌더링 과정은 털 렌더링 샘플링을 일반 오브젝트와 따로 할 수 있어서 원래의 렌더링 샘플링과 상관 없이 독립적으로 털 렌더링 퀄리티를 결정할 수 있다는 장점이 있다.

그림 2 는 털 렌더링 프레임워크가 털이 아닌 일반 오브젝트 렌더링과 함께 수행될 때의 시스템 전체의 프레임워크다. 이를 자세히 설명해 보면,

1) 털을 제외한 오브젝트의 그림자 렌더링을 수행한다.
2) 1)의 그림자 렌더링 버퍼 정보를 받아와서 털 데이터의 그림자 렌더링을 수행한다.
3) 2)를 통해 생성된 딥 쉐도우 맵 정보를 1)을 통해 만들어진 쉐도우 맵에 덧입힌다.
4) 털을 제외한 오브젝트의 메인 렌더링을 수행한다.
5) 3)에서 만들어진 딥 쉐도우 맵과 4) 과정에서 만들어진 렌더링 버퍼 정보를 받아와서 털 렌더링을 수행한다.
6) 4)에서 만들어진 최종 렌더링 버퍼에 털 렌더링 결과 버퍼를 합성해서 최종 결과 이미지를 생성한다.

위의 과정과 같다.

3.4 Deep render buffer

본 시스템은 털 렌더링 중 결과값을 저장할 때 딥 렌더 버퍼(deep render buffer)를 이용한다. 딥 렌더 버퍼는 한 픽셀의 값을 깊이에 따라 여러 개 저장할 수 있는 데이터 저장 객체이다. 딥 렌더 버퍼는 픽셀의 깊이 값에 따라 차례대로 정렬해서 값들을 저장할 수 있다. 이 시스템에서는 기본으로 투명 값을 가지고, 개체수가 많은 털의 특성상 딥 렌더 버퍼를 사용하여 렌더링을 효과적으로 처리할 수 있다.

가는 털의 렌더링 처리 후 해당 픽셀의 색깔 값과 알파 값을 저장한다. 그 후에 굵은 털을 계산해서 역시 색깔 값과 알파 값을 버퍼에 추가로 저장한다. 저장할 때는 알파 값의 순서에 따라 차례대로 저장한다. 픽셀에 대한 모든 처리 과정이 끝나면 픽셀 값은 알파값에 따라 누적하여 최종 값이 자연스럽게 나타나게 된다.

4. 실험결과

이 논문에서 제안한 방법으로 털을 렌더링한 결과를 표 1 에 나타내었다. 여기서 사용한 모델 데이터는 자체적으로 제작한 다람쥐 캐릭터이다. 렌더링 시간에는 털을 포함한 모든 프리미티브의 렌더링 시간과 그림자 만드는 시간을 포함한 시간이다. 실험에 사용된 시스템은 일반적인 PC 환경이며 Quad core 2.66GHz, 메모리 2GB 를 사용하였다.

여기서 비교한 기존의 방법은 가장 일반적인 리본 삼각화 방법이고 3x3 샘플링을 하였다. 표 1 에서 볼 수 있듯이 털의 개수에 상관없이 기존의 방법보다 본 논문에서 제안한 방법의 렌더링 시간이 많이 단축됨을 알 수 있다. 기존의 방법보다 3 분의 2 정도의 시간밖에 걸리지 않았다. 모델 데이터에 가는 털이 더 많을 경우에는 더욱 시간을 단축시킬 수 있을 것이다.

표 1. 렌더링 시간 비교

털 개수	기존 방법	제안한 방법
35 만개	70 초	46 초
70 만개	143 초	103 초
200 만개	628 초	432 초

그림 3 은 결과 이미지를 나타낸 것이다. (a)는 기존의 방법으로 렌더링한 결과이고 (b)는 이 논문에서 제안한 방법으로 렌더링한 결과이다. 결과를 보면 제안한 알고리즘을 적용한 이미지가 더 부드러움을 알 수 있다. 자세한 비교를 위하여 특정 부분을 확대하였다. 기존의 방법을 살펴보면 눈 옆의 털은 가운데 몇몇 픽셀이 검은색으로 나타남을 볼 수 있다. 이는 언더 샘플링으로 털을 제대로 샘플링하지 못해서 생기는 문제이다. 이러한 문제는 애니메이션으로 만들 때 플리커링을 야기시키기도 한다. 꼬리 부분은 제안한 알고리즘이 더 부드러움을 알 수 있는데 이는 가는 털까지 빠짐 없이 렌더링했기 때문이다. (c) 는 결과 이미지를 배경과 합성한 결과이다.

5. 결론

이 논문에서는 털 렌더링 시간을 단축하기 위해 기존의 렌더링 방법을 개선한 새로운 알고리즘을 제안하였고 효과적인 렌더링 방법을 위한 프레임워크, 딥 렌더 버퍼에 대해 설명하였다.

이러한 방법을 그대로 구현하여 실험해 보고 실제로 결과를 비교해 보았다. 논문에서 제안한 방법은 첫째로 모든 털에 대하여 샘플링을 할 필요가 없기 때문에 렌더링 시간을 단축시킬 수 있었다. 둘째로는 언더 샘플링 문제와 플리커링 문제를 제거할 수 있어서 렌더링 결과의 질을 향상시킬 수 있었다.

하지만 털을 렌더링하기 위해서는 렌더링 시간뿐만 아니라 대용량의 데이터를 한번에 처리하는 방법이나 더욱 실제적인 표현을 나타내기 위한 전역 조명 처리 등 개발해야 할 기술이 많이 남아있다.

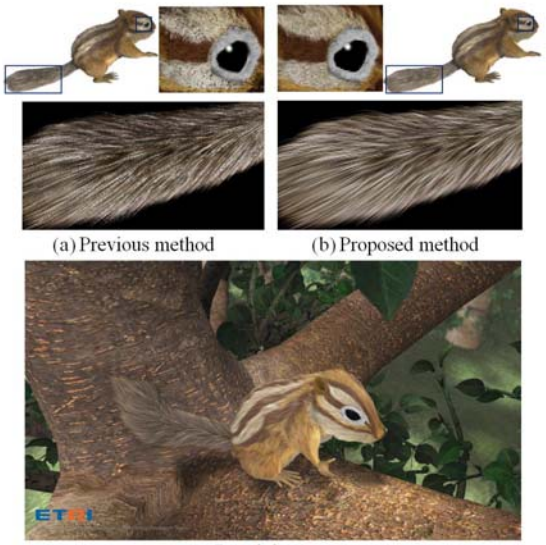

그림 3.렌더링 결과 비교 (a) 기존 방법 결과 (b) 제안한 방법 결과 (c) 배경 합성 장면

참고문헌

[1] Strand and Hair Modeling, Animation and Rendering. ACM SIGGRAPH 2007 Course 33, 2007

[2] Realistic Hair Simulation Animation and Rendering. ACM SIGGRAPH 2008 Course, 2008

[3] K.Watanabe and Y.Suenaga. A trigonal prism-based method for hair image generation. IEEE Computer Graphics and Applications. 12(1): 47-53, 1992

[4] I. Neulander and M. Van de Panne. Rendering generalized cylinders with paintstrokes. In Graphics Interfaces, page 233-242, 1998

[5] A. M. LeBlanc, R. Turner, and D.Thalmann. Rendering hair using pixel blending and shadow buffers. The Journal of Visualization and Computer Animation, 2(3):92-97, 1991.

[6] T. Nishita, H. Johan. A Rendering Method for Curved Tubular Objects by Using Scan Line Algorithm. Proceedings of the 7th Pacific Conference on Computer Graphics and Applications. Pages 92, 1999

[7] J.Kajiya and T. Kay. Rendering fur with three dimensional textures. In Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'89 conference), Computer Graphics Proceedings, Annual Conference Series, pages 271-280, New York, NY, USA, 1989. ACM Press.

[8] X. Yang, Z. Xu, J. Yang, and T. Wang. The cluster hair model. Graphics Models and Image Processing, 62(2):85-103, March 2000.

[9] S. Marschnerm H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. ACM Transactions on Graphics, 22(3):780-791, July 2003. Proceedings of ACM SIGGRAPH 2003.

[10] J. T .Moon, B. Walter and S. Marschner. Efficient Multiple Scattering in Hair Using Sperical Harmonics. ACM SIGGRAPH 2008

[11] A. Zinke, C. Yuksel, J. Keyser. Dual Scattering Approximation for Fast Multiple Scattering in Hair, ACM SIGGRAPH 2008

[12] T.Lokovic and E.Veach. Deep shadow maps. ACM SIGGRAPH 2000, pages 385-392, 2000.

[13] T-Y. Kim and U. Neumann. Opacity shadow maps. In Rendering Techniques 2001, Springer, pages 177-182, July 2001.

[14] C. Yuksel and J. Keyser. Deep opacity map. Computer Graphics Forum. Volume 27 Issue2, Pages 675-680, 2008

[15] Dan B. Goldman. Fake fur rendering. International Conference on Computer Graphics and Iteractive Techniques. Pages 127-134. 1997.

[16] J. Lengyel, E. Praun, A. Finkelstein, H. Hoppe. Real-time fur over arbitrary surfaces. Symposium on Interactive 3D Graphics. Pages 227-232, 2001

[17] C. Zeller, R. Fernando, M. Wloka, and M. Harris. Programming graphics hardware. In Eurographics-Tutorials, September 2004.

[18] C.Yuksel and E. Akleman. Rendering hair-like objects with indirect illumination. ACM SIGGRAPH 2005 Sketches, 2005.

[19] [CY06] C. Yuksel and E. Akleman. Rendering hair with global illumination", ACM SIGGRAPH 2006 Research Posters, 2006.