GPU의 병렬 처리 기능을 이용한 메쉬 평탄화 가속 방법※

이상길, 신병석
인하대학교 컴퓨터 · 정보 공학과
lsg5072@inha.edu, bsshin@inha.ac.kr

Acceleration of Mesh Denoising Using GPU Parallel Processing

Sang-Gil Lee, Byeong-Seok Shin
Dept. of Computer Science and Information Engineering, Inha University

요 약

메쉬 평탄화는 메쉬 표면의 잡음을 제거하는 것으로써 일반적으로 평탄화 필터를 적용하여 수행한다. 하지만 전체 과정이 CPU에서 수행되기 때문에 많은 실행 시간이 걸리는 문제점을 가진다. GPU는 부동소수점 연산에 특화되어 CPU에 비해 빠른 연산이 가능하기 때문에 복잡한 연산을 실시간으로 처리하는 것이 가능하다. 특히 메쉬 평탄화 과정은 메쉬의 각 정점이나 삼각형을 기반으로 같은 연산을 반복하기 때문에 GPU의 병렬 처리에 적합하다. 본 논문에서는 양방향 필터링에 GPU의 병렬 처리를 이용함으로써 메쉬 평탄화의 수행 시간을 줄이는 방법을 제안한다. 먼저 양방향 필터링을 위해 메쉬의 각 정점에 인접하는 삼각형들을 찾고 이들의 법선 벡터의 평균을 계산하여 정점들의 법선 벡터를 구한다. 양방향 필터링으로 각 정점의 새 위치를 계산하고 앞의 과정을 다시 수행하여 정점들의 새 법선 벡터를 계산한다.

ABSTRACT

Mesh denoising is a method to remove noise applying various filters. However, those methods usually spend much time since filtering is performed on CPU. Because GPU is specialized for floating point operations and faster than CPU, real-time processing for complex operations is possible. Especially mesh denoising is adequate for GPU parallel processing since it repeats the same operations for vertices or triangles. In this paper, we propose mesh denoising algorithm based on bilateral filtering using GPU parallel processing to reduce processing time. It finds neighbor triangles of each vertex for applying bilateral filter, and computes its normal vector. Then it performs bilateral filtering to estimate new vertex position and to update its normal vector.

Keyword : GPU, CUDA, Bilateral filtering, Mesh denoisng

접수일자 : 2009년 02월 27일
심사완료 : 2009년 04월 02일
※ 이 연구는 건설교통부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었습니다.
* 교신저자(Corresponding Author)
신병석, 주소 : 인천시 남구 용현동 253 인하대학교(402-751), 전화 : 032)860-7452, E-mail : bsshin@inha.ac.kr

1. 서 론

실세계의 물체를 왜곡 없이 그 형상과 표면의 색상을 사실적으로 디지털화하여 가상의 3차원 모델로 복원하는 기법은 애니메이션이나 게임 등과 같은 기존의 컴퓨터 그래픽스 분야 뿐 아니라 증강 현실(argument reality)과 혼합 현실(mixed reality), 그리고 측량 등과 같은 여러 응용 분야에서 매우 중요한 기술이다. 일반적으로 컴퓨터 그래픽스에서 모델링되는 삼각형 메쉬들은 3D 스캐너로부터 획득한 점 데이터(point set)나 CT (Computed Tomography), MRI(Magnetic Resonance Imaging) 등 단층 영상 촬영 장치에서 획득한 방대한 크기의 3차원 볼륨 데이터로부터 만들어진다. 하지만 볼륨 데이터를 메쉬로 변환할 때 입력 장치의 기계적인 특성 때문에 잡음(noise)이 발생하거나 잘못된 측정 오차 값들의 보간으로 인하여 모델링하고자 하는 물체의 특성을 변질시키는 문제가 있다. 이를 해결하기 위한 방법으로 여러 가지 필터링을 통한 메쉬 평탄화 기법이 연구되고 있다. 메쉬 표면의 평탄화는 기존의 영상처리(image processing) 분야에서 사용되었던 2차원 영상에서의 잡음 제거 방법을 메쉬로 확장하여 적용하는 방법이 널리 사용된다[1,2,3,4,5]. Tomasi가 제안한 2차원 영상의 양방향 필터링(bilateral filtering) 방법은 기존의 가우시안 필터링(Gaussian filtering)이 입력 영상의 중요한 특징까지 필터링하는 문제점을 보완한다[6]. Fleishman 등은 이를 이용한 메쉬의 표면 평탄화 방법을 제안하였다[2]. 하지만 이 방법들은 CPU를 이용하여 구현되었으며 많은 수행 시간이 걸리는 문제점이 있다.

NVIDIA™에서 개발한 CUDA(Compute Unified Device Architecture)는 병렬 처리와 부동소수점 계산에 특화된 GPU를 일반적인 목적에 이용하기 위해 제시된 GPGPU(General Purpose computation on GPUs) 기술이다[7,8]. 이를 이용하면 CPU에서 긴 수행시간이 필요한 반복적인 연산을 빠르게 실행할 수 있다. 따라서 GPU를 그래픽 연산뿐 아니라 수학, 물리 등 다양한 분야에 사용할 수 있다[9,10,11].

본 논문에서는 위와 같은 GPU 병렬 처리 시스템을 이용하기 위해 각 정점을 하나의 쓰레드(thread)화 하여 병렬 처리 시스템에 적합한 양방향 필터링 기법을 고안하였다. 전체적인 수행 과정은 다음과 같다. 우선, 각 쓰레드는 양방향 필터링을 위해 각 정점의 인접 삼각형들의 법선 벡터(normal vector)의 평균값을 이용해 할당된 정점의 법선 벡터(vertex normal vector)를 계산한다. 다음으로 메쉬의 평탄화를 위해 사용자가 정의한 특정 반지름을 가지며 각 정점을 중심으로 하는 구를 통해 구 내부에 위치한 정점들을 이웃 정점으로 정의한다. 마지막으로 각 정점들과 이웃 정점들 간 거리의 표준편차, 그리고 이웃 정점들과의 내적을 통해 위상차를 구하고, 이를 이용해 양방향 필터링으로 각 정점의 새로운 위치를 계산한다. 이 방법은 CPU에서 실행한 결과 영상과 동일한 화질을 유지하며 수행 시간은 1% 이내로 감소하였다.

논문의 구성은 다음과 같다. 2장에서는 기존의 2차원 영상 기반 양방향 필터링과 이를 이용한 메쉬 평탄화 방법들을 소개한다. 3장에서는 GPU 병렬 처리 방법과 2차원 영상에 적용되는 양방향 필터링의 원리를 설명하고, CUDA를 이용하여 메쉬의 표면을 평탄화하는 방법을 설명한다. 4장에서는 실험 결과를 기술하고, 5장에서는 결론을 맺는다.

2. 관련 연구

가우시안 필터링은 영상의 잡음을 제거하기 위해 많이 사용되는 방법이다. 하지만 이 방법은 영상의 중요한 특징(feature)까지 없애는 문제점이 있다. 따라서 메쉬 평탄화에 가우시안 필터링을 적용하면 물체의 모양을 결정하는 특징적인 형상들이 없어질 수 있다. Tomasi는 가우시안 필터의 문제를 해결하기 위해 영상의 화소 간 거리 차이와 색상 차이를 이용한 2차원 영상의 양방향 필터링(bilateral filtering) 방법을 제안하였다[6]. Taubin은 각 정점의 이웃 정점들의 법선 벡터로 라플라시안 필터링(Laplacian filtering)하여 정점들의 법

선 벡터를 구하고, 이를 이용해 최소 제곱 오차법(least squares error method)으로 정점들의 새 위치를 계산한다[1]. Fleishman은 메쉬의 각 정점에 3차원으로 확장된 양방향 필터링 알고리즘을 적용하여 표면의 잡음을 제거하는 방법을 제안하였다[2]. Jones는 메쉬의 각 정점을 그 이웃 삼각형에 투영시켜 얻은 예상점(predictor)과 해당 정점 간의 거리를 이용한 양방향 필터링을 제안하였다[3]. 이 방법은 Fleishman의 방법을 여러 번 적용한 영상과 유사한 결과를 내면서도 한 번의 필터링으로 메쉬 표면의 평탄화가 가능하다. 다만 예상점 계산 시간이 오래 걸려 수행 시간이 Fleishman의 방법을 여러 번 수행하는 것과 비슷하다. Lee는 각 정점의 법선 벡터를 양방향 필터링하고 그 결과를 이용하여 정점의 위치를 재구성하는 방법을 제안하였다[4]. 이 방법은 기존 방법들과 비슷한 수행 시간이 걸리지만 메쉬의 날카로운 모서리 등의 중요 부분을 보존하고 잡음을 더 효과적으로 줄일 수 있다. Yagou는 평균화 필터링(mean filtering)과 미디언 필터링(median filtering)을 정점의 법선 벡터에 적용한 방법을 제안하였다[5]. 하지만 이런 방법들은 모두 CPU에서 처리되며 많은 시간이 걸리는 단점이 있다.

3. GPU 병렬 처리를 이용한 메쉬 양방향 필터링

본 장에서는 2차원 양방향 필터링과 이를 3차원으로 확장시킨 방법, 그리고 이 것을 GPU를 이용하여 구현하는 방법을 설명한다. 본 논문에서는 양방향 필터링을 메쉬에 적용하는 여러 가지 방법들 중 Fleishman의 방법[2]을 이용하였다. 양방향 필터링은 입력된 2차원 영상의 각 픽셀과 그 이웃 픽셀 간의 거리 차이와 색상 차이를 이용해 필터링하여 이미지의 의미 있는 경계 부분을 보존한다[식 1].

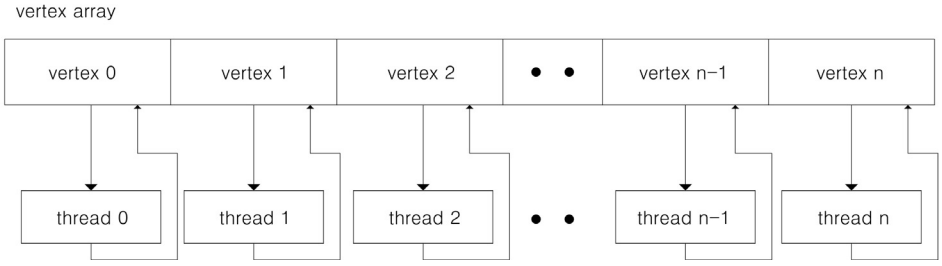
$$h(s)=\frac{\sum_{p\in\Omega} f(p-s)g(I_p-I_s)I_p}{\sum_{p\in\Omega} f(p-s)g(I_p-I_s)} \qquad [식\ 1]$$

$$f(s)=g(s)=e^{-x^2/2\sigma_s^2}$$

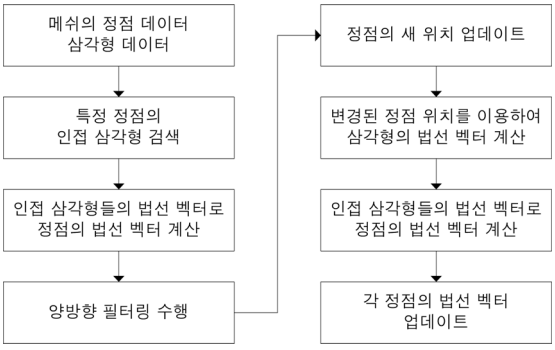
여기서 $h(s)$는 필터링된 결과 영상을 의미하고, s와 p는 영상의 해당 화소와 그 이웃 화소를 의미한다. [식 1]의 함수 f와 함수 g는 각각 s와 p 사이의 거리 차이와 색상 차이를 인자로 하는 가우시안 함수로써 f는 해당 화소와 이웃 화소 간의 거리가 클 경우, g는 해당 화소의 색상 I_s와 이웃 화소의 색상 I_p간의 차이가 클 경우 감소한다. 2차원 영상에 양방향 필터링[식 1]을 적용하면 주변 화소들 간에 색상 차가 큰 엣지 부분은 보존되고 나머지 부분은 평탄화된 결과를 얻을 수 있다.

여기서는 [식 1]의 인자들을 화소 대신에 접평면(tangent plane) 상의 정점으로 하여 계산한다. 이렇게 하면 메쉬의 특징을 보존하면서 잡음을 제거할 수 있다. 이 방법은 CUDA를 이용하여 구현했다.

한 정점의 새로운 위치는 하나의 쓰레드에서 양방향 필터링으로 계산되며, 다수의 정점들에 대하여 GPU에서 병렬 처리되기 때문에 CPU에서 필터링하는 것에 비교할 때 매우 빠르다. 각 정점의 새로운 위치는 기존 위치에서 법선 벡터 방향으로 움직이므로 정점의 법선 벡터를 계산해야 하며, 이를 위하여 각 쓰레드에서 해당 정점과 인접한 삼각형들을 찾는다. 본 논문에서 사용한 메쉬들은 위상 정보가 없기 때문에 메쉬의 삼각형 개수만큼 반복하여 해당 정점을 공유하는 이웃 삼각형들을 찾고, 각 삼각형들의 법선 벡터를 평균하여 해당 정점의 법선 벡터를 계산한다. 계산된 각 정점의 법선 벡터와 위치 정보를 이용하여 양방향 필터링을 수행한다.



[그림 1] n개의 쓰레드가 n개의 정점을 병렬 처리하는 과정



[그림 2] 제안하는 방법의 처리 절차

입력 데이터는 각 정점의 위상 정보가 없는 메쉬, 또는 정점들(point set)이다. 함수 f, g의 인자 σ_c와 σ_s는 각각 이웃 정점을 찾기 위한 구의 반지름과 이웃 정점들과의 거리의 표준편차로써, 구의 반지름은 사용자가 임의로 정한다. [그림 3]의 q_i는 $\| s - q_i \| < \lceil 2\sigma_c \rceil$ 를 만족하는 정점 s의 이웃 정점들이다. t는 해당 정점과 이웃 정점 간의 거리를 나타내며 가우시안 함수 f의 인자이다. 해당 정점의 법선 벡터와 해당 정점과 이웃 정점을 연결하는 벡터의 내적인 h는, 두 벡터의 내적이 수직에 가까울수록 0에 가까워진다는 점을 이용한 특징 보존 함수(feature-preserving weight function) g의 인자이다.

```
CalcVertexNormal(Vertices s[], VertexNormals n[])
    bi = index of this block
    ti = index of this thread within the block
    num = number of threads in each block
    index = bi*num+ti
    {nti} = NeighborTriangles(s[index])
    M = |{nti}|
    Vector sum
    for i := 1 to M
      sum += CrossProduct(nti.v0-nti.v1, nti.v0-nti.v2)
    end
    n[index] = sum/M
```

```
BilateralFiltering(Vertices s[], VertexNormals n[],
Radius r)
    bi = index of this block
    ti = index of this thread within the block
    num = number of threads in each block
    index = bi*num+ti
    {qi} = NeighborVertices(s[index], r)
    K = |{qi}|
    sum = 0
    normalizer = 0
    for j := 1 to K
      //특정 정점과 이웃 정점의 거리
      t = ||s[index]-qi||
      //두 정점의 법선 벡터의 내적
      h = DotProduct(n[index], s[index]-qi)
      f = exp(-t^2/(2σc^2))
      g = exp(-h^2/(2σs^2))
      sum += (f*g)*h
      normalizer += f*g
    end
    s[index] += n[index]*(sum/normalizer)
```

[그림 3] 본 논문에서 구현한 2개의 커널 함수. GPU에서 정점의 수만큼의 쓰레드를 생성하여 각 쓰레드에서 하나의 정점에 대한 연산을 수행한다.

[그림 3]의 CalcVertexNormal 함수는 각 정점의 법선 벡터를 계산하며, 양방향 필터링의 전후 2번 수행된다[그림 2]. [식 1]의 함수 f와 함수 g를 이용한 BilateralFiltering 함수는 해당 정점과 이웃 정점 간의 위상 차이가 크면 해당 정점의 위치를 유지하고 위상 차이가 작으면 해당 정점의 위치를 변경하여 평탄화한다. 이 함수들은 GPU에서 병렬로 수행된다.

CUDA에서 블록(block)은 쓰레드(thread)들의 묶음이며 메쉬 정점의 수에 따라 크기가 결정된다. 병렬 처리 시 정점이 누락되는 것을 막기 위하여 생성되는 쓰레드의 수는 메쉬 정점의 수와 같거나 많게 정한다. [그림 2]의 처리 절차에서 수행되는 [그림 3]의 두 함수는 병렬 처리를 위해 참조 번호(index)를 가지며, 정점들의 위치와 법선 벡터 정보를 가지고 있는 2개의 1차원 배열을 이용하여 수행한다[그림 1].

4. 실험 결과

실험은 INTEL Xeon 3GHz 에 4GB 주 메모리를 갖는 시스템에서 수행되었다. 그래픽 가속기는 NVIDIA ENGTX 280을 사용하였고, DirectX9 라이브러리와 NVIDIA™ CUDA를 사용하였다.

[표 1] 90,000개 정점 Stanford Dragon 양방향 필터링 수행 시간 비교(단위 sec)

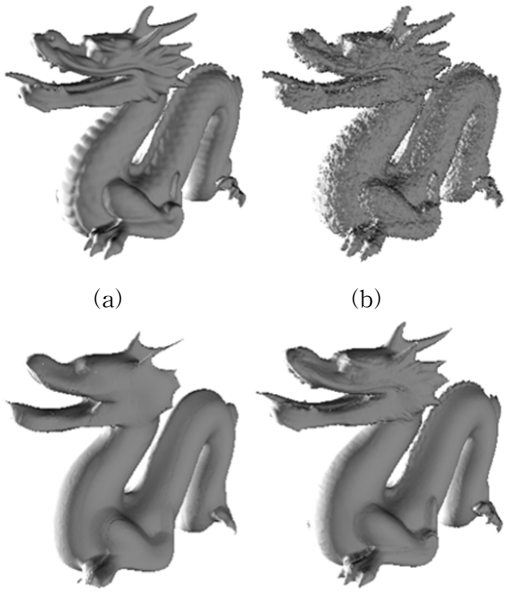
	CPU에서 수행	CUDA 적용	속도 향상 비율
각 정점의 법선 벡터 계산	30.2	0.6	51배
각 정점의 이웃 정점 검색	1342.7	0.4	3357배
양방향 필터링 수행	15.8	0.03	527배
전체 수행 시간	1388.7	1.03	1348배

CUDA를 이용하여 전체 과정을 1번 수행하였을 경우 수행 시간이 CPU보다 약 1300배, 양방향 필터링 수행 시간은 약 400배 빨라졌다[표 1]. 각 정점의 이웃 정점 검색 과정은 해당 정점과 나머지 모든 정점 간의 거리를 계산하기 때문에 CPU에서 순차적으로 수행하면 시간이 오래 걸린다. 하지만 CUDA는 연산을 병렬로 처리하며 GPU가 부동 소수점 연산에 특화되어 있기 때문에 CPU에 비하여 1000배 이상 빠르다.

[표 2] 메쉬 모델의 정점 수에 따른 양방향 필터링 1회 수행 시간 비교(단위 sec)

실험 모델	수행 시간
14k 정점 / 27k 삼각형	0.003
159k 정점 / 318k 삼각형	0.217
196k 정점 / 390k 삼각형	0.322
90k 정점 / 181k 삼각형	0.033

[표 2]는 전체 수행 과정 중 양방향 필터링 시간을 비교한 것이며, 반지름의 크기는 메쉬 크기에 맞춰 동일한 비율로 적용하였다. 정점이 많은 메쉬는 정점이 적은 메쉬보다 정점이 조밀하게 배치되므로 각 정점에 대하여 검색되는 이웃 정점의 수가 많다. 14,000개 정점을 가지는 메쉬의 경우 196,000개 정점을 가지는 메쉬에 비하여 검색되는 이웃 정점의 수가 적기 때문에 양방향 필터링 수행 시간이 빠르다. 또한 GPU는 한 번에 수행할 수 있는 활성 쓰레드(active threads)의 수가 정해 있으므로 쓰레드의 수가 많아질수록 수행 속도가 느려진다. 본 논문에서는 하나의 쓰레드에서 한 정점을 필터링하도록 구현하였기 때문에 메쉬 간 정점의 수 차이보다 수행 시간의 차이가 더 크다.

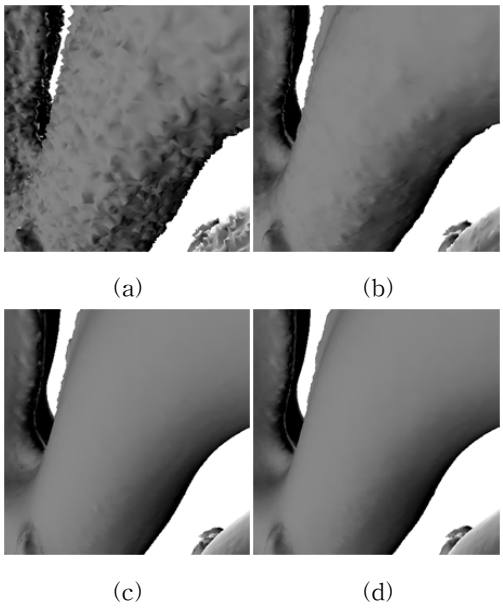


[그림 4] 90,000개 정점 Stanford Dragon 메쉬 (a) 원본 영상 (b) 원본에 잡음을 추가한 영상 (c) 평균화 필터링(mean filtering) 2회 적용한 결과 영상 (d) 양방향 필터링(bilateral filtering) 2회 적용한 결과 영상

평균화 필터링은 2차원 영상의 한 화소, 혹은 3차원 영상의 한 정점의 값을 결정할 때, 이웃 값들의 평균을 이용하여 필터링하는 방법이다. 메쉬의 표면 잡음 제거에 평균화 필터링을 적용할 경우 메쉬의 특정 부분이 사라지며 전체적인 형상이 뭉개진다. [그림 4]는 정점 90,000개의 용 메쉬에 평균화 필터링과 양방향 필터링을 적용한 결과 영상이다. 평균화 필터링의 경우 용의 뿔, 발 등의 특정 형상들이 없어지지만 양방향 필터링을 적용하면 특정 형상들이 보존되며 특징이 없는 몸통 부분이 매끈하게 평탄화된다.

이웃 정점을 찾기 위해 정의한 구의 반지름은 사용한 데이터의 크기에 따라 다르며 이에 따라 검색되는 이웃 정점의 수가 다르다. 이웃 정점의 수가 너무 적을 경우 표면의 잡음을 제거할 수 없기 때문에 본 논문에서는 메쉬 크기의 1/50에서 1/25로 정하여 테스트 하였으며 이 경우 검색된 각 정점의 이웃 정점들의 수는 평균 70개이다. [그림 5]는 반지름에 따른 결과 비교 영상이며, 반지름이 작을수록 표면의 잡음이 남아있고 반지름이 클수록 표면의 잡음이 줄어든다.

[그림 6]은 양방향 필터링 횟수에 따른 결과 영상을 나타낸다. 양방향 필터링을 1번 적용하였을 때 메쉬의 표면에 잡음이 남아있지만, 2번 적용하였을 경우 메쉬의 평평한 부분의 잡음이 완전히 제거됨을 확인할 수 있다. [그림 7]은 메쉬에 양방향 필터링을 2회 적용한 영상으로, 원본 메쉬 표면의 구멍 등의 잡음이 사라지지만 표면의 큰 굴곡과 같은 특징들이 유지됨을 보인다.



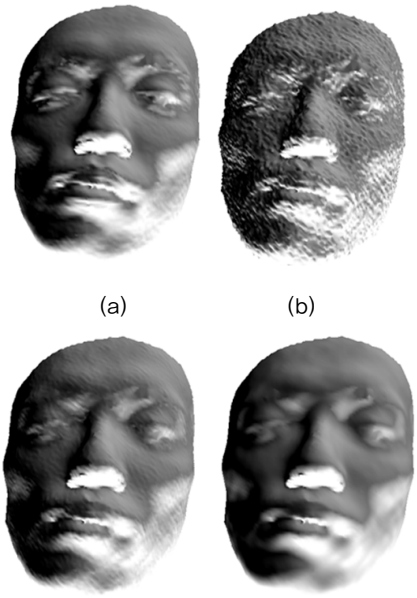
[그림 5] (a) Stanford Dragon 메쉬의 몸체 부분에 잡음 추가한 영상 (b) 메쉬의 z축 방향 최대 길이의 1/50을 반지름 σ_c에 적용한 결과 영상 (c)σ_c 에 1/35을 적용한 결과 영상 (d) σ_c에 1/25을 적용한 결과 영상

5. 결 론

본 논문에서는 CUDA의 GPU 병렬 처리 기능을 이용하여 3차원 양방향 필터링 기반의 메쉬 평탄화 방법을 구현하였다. 각 정점의 법선 벡터를

계산하였고 양방향 필터링 알고리즘을 적용하여 모델의 특징을 보존하며 잡음을 제거하였다. 부동소수점 계산과 병렬 처리에 특화된 GPU로 구현하여 기존의 CPU 구현에 비하여 월등히 빠른 수행 속도를 보였다.

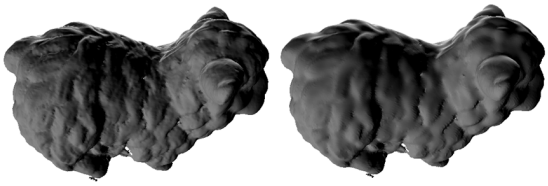
구를 이용하여 양방향 필터링을 위한 이웃 정점들을 검색하기 위해 일정 반지름을 가지는 구의 내부에 있는 정점들을 검사한다. 이 방법은 이웃 정점을 검색하는 일반적인 방법이지만, 매번 각 정점마다 이웃 정점들을 검색해야 하므로 많은 시간이 걸린다. 이 문제를 해결하기 위하여 각 쓰레드에서 정점의 위치가 아닌 삼각형의 법선 벡터를 기준으로 필터링을 수행하고, 이웃 삼각형 검색 시 구가 아닌 메쉬의 위상 정보를 이용한다면 전체 수행 시간을 상당히 줄일 수 있다.



(a) (b)

(c) (d)

[그림 6] 14,000개 정점 얼굴 메쉬
(a) 원본 영상 (b) 잡음 추가 영상
(c) 양방향 필터링 1회 적용 결과 영상
(d) 양방향 필터링 2회 적용 결과 영상



(a) (b)

[그림 7] 160,000개 정점 양 메쉬 (a) 원본 영상 (b) 양방향 필터링 2회 적용한 결과 영상

참고 문헌

[1]] G. Taubin, “Linear Anisotropic Mesh Filtering”, Tech. Rep. IBM Research Report RC2213, October 2001.

[2]] S. Fleishman, I. Drori, and D.Cohen-Or, “Bilateral Mesh Denoising”, ACM Trans Graphics, vol. 22, pp 950-953, July 2003.

[3]] T. R. Jones, F. Durand, and M.Desbrun, “Non-Iterative, Feature-Preserving Mesh Smoothing”, ACM Trans Graphics, vol. 22, pp. 943-949, July 2003.

[4]] Kai-Wah Lee, Wen-Ping Wang, "Feature-preserving Mesh Denoising via Bilateral Normal Filtering", Proceedings of the Ninth International Conference on Computer Aided Design and Computer Graphics, IEEE Computer Society, pp. 275 - 280, 2008

[5] H. Yagou, Y. Ohtake and A.Belyaev, “Mesh Smoothing via Mean and Median Filtering Applied to Face Normals”, Geometric Modeling and Processing, pp. 121-131, 2002.

[6] C. Tomasi and R. Manduchi, “Bilateral Filtering for Gray and Color Images”, Proceedings of the Sixth International Conference on Computer Vision, pp.836-846, 1998.

[7] NVIDIA, http://developer.nvidia.com/object/cuda.html

[8] NVIDIA, http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf

[9] General-Purpose Computation Using Graphics Hardware, “http://www.gpgpu.org”

[10] M. Pharr, R. Fernando, "Implementing Efficient Parallel Data Structures on GPUs",

GPU Gems 2, pp. 521-545, 2005

[11] J. D. Owens, D. Leubke, N. Govindaraju, M. Harris, J. Kruger, A. Lefohn, T. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware", Eurographics Computer Graphics Forum, vol. 26, no. 1, pp. 80-113, 2007.

이상길 (Sang-gil Lee)

2008년 2월 인하대학교 컴퓨터공학부(학사)
2008년~현재 인하대학교 정보공학부(석사)

관심분야 : GPU 병렬 처리

신병석(Byeong-Seok Shin)

1990년 2월 서울대학교 컴퓨터공학과(학사)
1992년 2월 서울대학교 컴퓨터공학과(석사)
1997년 2월 서울대학교 컴퓨터공학과(박사)
2000년~현재 인하대학교 컴퓨터공학부 부교수

관심분야 : 실시간 렌더링, 볼륨 그래픽스, 의료 영상