GPU 기반 실시간 멀티 바운스 주변 폐색 렌더링*

안현장[01], 최재원[2], 이성길[3]
[1]성균관대학교 독어독문학과, [2]성균관대학교 전자전기컴퓨터공학과, [3]성균관대학교 소프트웨어학과
{hyoenjang.an, jaewonchoi, sungkil}@skku.edu

GPU-Based Real-Time Multi-Bounce Ambient-Occlusion Rendering

Hyeonjang An[01], Jaewon Choi[2], Sungkil Lee[3]
[1]Department of German Language and Literature, Sungkyunkwan University.
[2]Department of Electrical and Computer Engineering, Sungkyunkwan University
[3]Department of Software, Sungkyunkwan University

요약

주변 폐색 렌더링은 간접 조명의 실시간 근사를 위해 많이 사용되나, 싱글 바운스 폐색 추정이 보편적이다. 본 논문은 보다 정확한 가시도 표현을 위해 물체 표면과 주변 기하의 폐색 관계를 멀티 바운스로 추정하는 방법을 제안한다. 멀티 바운스 기반 접근은 주변의 기하 관계를 보다 정확히 반영하여 더 사실적인 음영을 표현한다. 실시간 성능을 위한 GPU 기반 광선 추적은 기존보다 정교한 실시간 주변 폐색 렌더링을 가능하게 한다.

1. 서론

컴퓨터 그래픽스에서 주변 폐색은 난반사 물체의 간접 조명을 가시도 만으로 효과적으로 근사하는 기법이다. 이는 전역 조명을 완전히 계산하지 않고, 물체의 표면에서 반구 형태의 brute-force 방식으로 광선과 기하의 교차 검사만을 수행하기 때문에 저비용으로 간접 조명 효과를 낼 수 있다. 실시간 주변 폐색 효과 렌더링은 주로 screen space ambient occlusion (SSAO)[2] 기법으로 화면상에서 근사되었지만, 이는 G-buffer를 활용한 2차원 깊이 비교 근사로 정확성이 낮다.

본 논문은 이전보다 사실적인 음영의 표현을 위한 GPU 기반의 멀티 바운스 주변 폐색 렌더링 기법을 제안한다. 표면에서 보이는 주변 폐색을 정확하게 추적하기 위해 교차 후 광선을 추가로 바운스하여 주변 폐색 정도를 누적한다. 또한, 광선 추적법을 사용하여 발생하는 연산의 과부하는 병렬 연산이 가능한 GPU로 완화한다.

2. 관련연구

2.1 주변 폐색 렌더링

Shanmugam[1]은 동적인 장면과 복잡한 기하에도 적용 가능한 GPU 기반의 주변 폐색 근사 기법을 제안하였다. 주변 폐색 기법의 근사 알고리즘를 제시하고, GPU 기반의 깊이 버퍼와 법선 버퍼를 활용하여 이를 적용하는 방법을 고안하였다. Bavoil[2]은 실시간으로 주변 폐색을 렌더링하기 위한 SSAO 기법을 제시하였다. SSAO는 주변 폐색의 근사방법으로, G-buffer로 픽셀 주변을 샘플링하여 깊이를 구해내고 그 비교를 통해 폐색정도를 계산한다. 음영표현이 부정확하다는 단점이 있지만, 동적인 장면에서도 실시간 렌더링이 가능하다.

2.2 광선 추적 가속화

Horn[3]은 GPU에서의 KD트리 탐색 가속 방법으로, KD-restart 알고리즘을 개선한 세 가지 방법을 제시하였다. 기존의 방법은 큰 용량의 스택을 필요로 하거나, 트리의 루트 노드를 반복적으로 방문하기 때문에 추가적인 부하를 유발한다. Horn은 이 부하를 완화하기 위해 광선을 패킷화하는 방법, 서브 트리만을 방문하는 방법, 저용량 고정 스택을 이용하는 방법을 제시하였다.

3. 알고리즘

본 연구의 멀티 바운스 주변 폐색 알고리즘은 반복적으로 바운스되는 광선을 통해 주변 차폐여부를 정교하게 확인한다. 연산의 과부하는 GPU 병렬연산과 이에 적합한 KD-pushdown 탐색 알고리즘을 적용하여 해결한다.

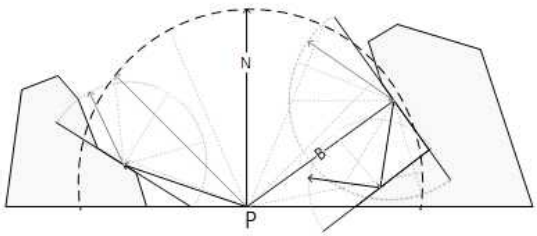

그림 1: 멀티 바운스 주변 폐색

---

* 구두 발표논문
* 학부생 주저자 논문임.
* 본 연구는 미래창조과학부의 재원으로 한국연구재단의 중견연구자지원사업(2019R1A2C2002449) 및 과학기술인문융합연구사업(2017M3C1B6070980)의 지원을 받아 수행되었음.

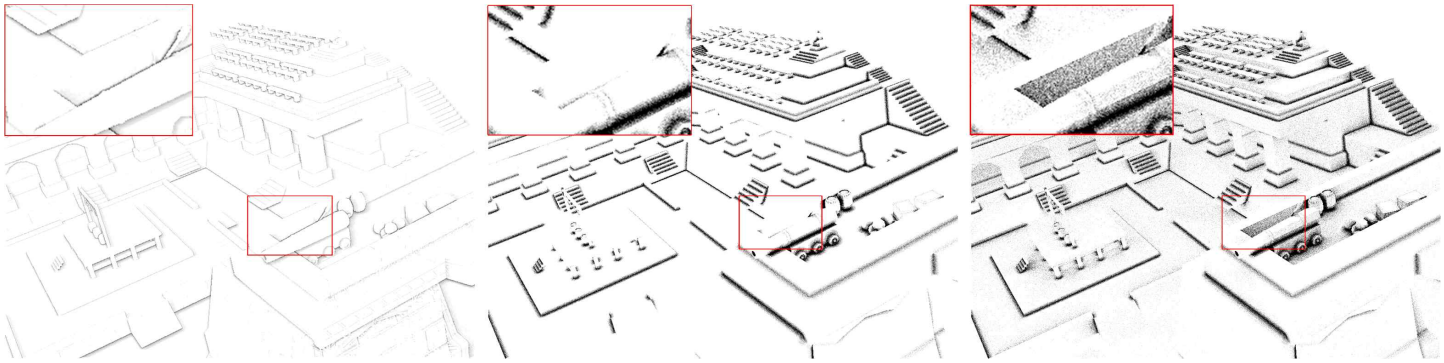

그림 2: 주변 폐색 기법에 따른 주변 폐색 맵의 예시 결과 비교. (좌) SSAO, (중간) 싱글 바운스, (우) 더블 바운스

3.1 멀티 바운스 주변 폐색

그림 1과 같이, 본 기법은 스크린 공간에서 쏜 광선이 교차하는 표면에서만 주변 폐색을 반복 계산한다. 단순히 교차를 검사하여 차폐 여부를 확인하는 것이 아니라, 교차하는 표면에서 광선을 바운스하여 다른 표면과의 차폐 여부도 함께 확인한다. 해당 알고리즘은 하단의 공식으로 정의할 수 있다.

$$B = \frac{1}{\pi}\int_{\Omega} B_i V(p,\omega_i)(n \bullet \omega_i)dw_i \qquad (1)$$

$$A(p) = 1 - \frac{1}{\pi}\int_{\Omega} B_i V(p,w_i)(n \bullet w_i)dw_i \qquad (2)$$

B는 점p에 도달한 광선$\vec{\omega}$에 대하여 반구 형태로 바운스된 광선들을 적분하는 함수이며 바운스된 광선들은 다시금 B_i로 교차한 지점에서 B와 동일한 역할을 한다. V는 가시도 함수로 점 p에서 출발한 광선 $\vec{\omega}$의 도달 지점에서 물체 표면의 교차 여부를 판단하여 0과 1을 반환한다. $(n \bullet \omega_i)$는 광선 입사각에 따른 감쇠 계수이며, 최종으로 p에서의 주변 폐색값 $A(p)$는 1에서 p에 모인 B의 반구 형태의 적분 결과를 뺀 것이 된다.

3.2 GPU 기반 광선추적법

본 기법은 픽셀에서 쏜 광선이 표면에 바운스(b)할 때마다 brute-force로 재생성된 광선의 교차를 검사한다. 이에 연산 부하는 바운스마다 재생성된 광선 수(r)의 누적 곱만큼 증가하여 총 수행 시간은 화면 해상도$\times r^b$ 가 된다. 본 연구는 GPU의 병렬 연산을 이용하여 이를 완화한다. 셰이더 상의 한정된 반복문 내에서 광선을 재생성하고, 이를 통해 주변 폐색을 검사하여 결과를 텍스처에 누적한다. 물체 표면과의 교차 연산은 물체를 GPU 상의 KD 트리 가속 구조로 구성하고 KD-pushdown 알고리즘을 적용하여 최적화한다.

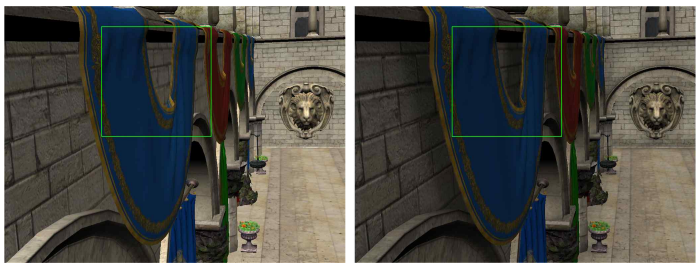

그림 3: 멀티 바운스 주변 폐색 렌더링의 적용 전(좌)과 후(우)

표 1: 렌더링 성능 측정 결과(샘플 수:4, 해상도:512×512)

모델	SSAO[2]	1 바운스	2 바운스
코넬 박스	0.048ms	1.524ms	10.431ms
유적	0.049ms	6.195ms	36.673ms
스폰자	0.169ms	21.297ms	147.508ms

4. 결과 및 토론

본 기법은 Intel Core i7-4790, NVIDIA GeForce RTX 2080 Ti에서 구현되었다. 그림 2의 1 바운스와 2 바운스가 그려낸 음영맵을 비교하면, 2 바운스의 경우 1 바운스보다 옅은 차폐와 깊은 차폐를 명확히 구분해 낸다는 사실을 확인할 수 있다. 이는 1 바운스의 경우 구할 수 있는 차폐 정보가 한정되어 있지만, 멀티 바운스의 경우 차폐 정보가 바운스마다 점진적으로 구체화되기 때문이다. 성능은 표 1에서 보듯이 1 바운스의 경우 비교적 빠르며 2 바운스부터는 연산량이 급증한다.

본 논문에서는 광선을 개별로 쏘아 brute-force로 처리하기 때문에 바운스 카운트에 따라 연산 부하가 급증한다. 추후 연구에서는 GPU 기반 광선추적법을 최적화하여 연산 부하를 효율적으로 처리하는 연구를 진행할 예정이다.

참고문헌

[1] Shanmugam, Perumaal, and Okan Arikan. "Hardware accelerated ambient occlusion techniques on GPUs." Proceedings of the 2007 symposium on Interactive 3D graphics and games. 2007.

[2] Bavoil, Louis, Miguel Sainz, and Rouslan Dimitrov. "Image-space horizon-based ambient occlusion." ACM SIGGRAPH 2008 talks. 2008. 1-1.

[3] Horn, Daniel Reiter, et al. "Interactive kd tree GPU raytracing." Proceedings of the 2007 symposium on Interactive 3D graphics and games. 2007.

[4] Díaz, José, et al. "Real-time ambient occlusion and halos with summed area tables." Computers & Graphics 34.4 (2010): 337-350. Symposium on Interactive 3D Graphics and Games. 2013.

[5] Garanzha, Kirill, and Charles Loop. "Fast ray sorting and breadth-first packet traversal for GPU ray tracing." Computer Graphics Forum. Vol. 29. No. 2. Oxford, UK: Blackwell Publishing Ltd, 2010.