GPU 기반의 병렬화된 점진적 지터드 샘플링*

고지은[01] 정효진[1] 이성길[2]
[1]성균관대학교 전기전자컴퓨터공학부
[2]성균관대학교 소프트웨어학과
jieunko@skku.edu, junghj0625@skku.edu, sungkil@skku.edu

GPU-Based Parallel Progressive Jittered Sampling

Jieun Ko[01] Hyojin Jung[1] Sungkil Lee[2]
[1]School of Electronical engineering and computer science, Sungkyunkwan University
[2]College of Software, Sungkyunkwan University

요 약

컴퓨터 그래픽스에서 blue-noise 샘플링 기법은 다양하게 활용되고 있다. 일반적으로 CPU 기반의 샘플링 기법에서는 샘플의 개수가 증가할수록 선형으로 생성시간이 증가한다. 이를 GPU 기반 병렬화를 통해 성능을 향상시킬 수 있다. 이때 병렬화 시, 샘플링을 점진적으로 수행하며 샘플 셋의 랜덤하게 균일한 분포를 유지하는 progressive한 알고리즘은 그 순차적 의존성이 있어 이를 해결할 필요가 있다. 본 논문은 progressive함과 동시에 GPU 기반의 병렬처리를 통해 효율성을 높인 Parallel Progressive Jittered 알고리즘을 제안한다. 본 논문은 이전 단계에서 4개의 샘플을 읽은 후, 각 샘플 당 영역을 사분할하여 단계의 샘플 위치들을 Progressive Jittered 샘플링의 패턴에 따라 병렬적으로 계산한다. 본 알고리즘은 $\log_4 n$의 시간복잡도로 샘플링 개수가 증가할수록 기존의 기법보다 더 효율적임을 보였다.

1. 서 론

그래픽스에서 blue-noise 샘플링 기법은 다양하게 활용되고 있다. 일반적으로 쓰이는 샘플링 연구 중 Poisson disk [1] 샘플링은 같은 샘플 개수 대비 고품질의 이미지를 생산한다. 이 후 Poisson disk 샘플링의 연산량이 많기 때문에 샘플 생성 속도를 높인 Jittered [2, 3] 샘플링 기법이 제안되었다. 이 기법은 grid 구조로 분할되어 순서대로 샘플링을 하는 특성상 단계별로 샘플을 생성하는 과정에서, 모든 이전 단계의 샘플들이 blue-noise 상태의 분포가 되어 있음을 보장하는 progressive sample sequence의 특징을 만족하지 못한다. 그러므로 샘플링을 점진적으로 수행하는 Progressive Jittered (PJ) [4]와 같은 기법이 제안되었다. 이와 같은 CPU 기반 샘플링 알고리즘들은 샘플의 수에 비례해 연산시간이 선형으로 증가한다. 이를 GPU 기반 병렬화를 통해 성능을 향상시킬 수 있다. 하지만, progressive 샘플링의 경우 이전 샘플을 읽어 샘플을 생성하는 순차적인 의존성이 발생하기 때문에 이는 병렬화가 어렵다. 본 연구는 GPU 기반으로 병렬화된 Parallel Progressive Jittered 샘플링 알고리즘을 제안한다. 이전 단계의 4개의 샘플을 읽은 후, 다음 레벨 샘플들을 동시에 그리는 방식을 통해 각 구역의 progressive 패턴을 유지하면서 GPU에서 병렬화를 수행한다. 이 방식은 n번의 샘플링을 수행하는 기존 방법에 비해 하나의 구역에 할당된 1개의 샘플이 구역이 분할되며 4개의 샘플을 생성하므로 $\log_4 n$의 연산만으로 샘플링이 가능하다. 본 논문[1]의 실험결과를 통해 샘플의 수가 증가할수록 제안된 알고리즘의 성능이 효율적임을 보인다.

2. 관련 연구

샘플이 특정 거리 이상을 유지하면서 위치의 랜덤성을 가지는 Poisson disk [1] 샘플링 기법은 적은 샘플링 양으로도 충분히 좋은 이미지 품질을 보여준다. 그러나 Poisson disk 샘플링은 목표 영역이 샘플로 채워질 때까지, CPU를 이용한 순차적인 알고리즘을 통해 brute-force한 계산을 반복한다.

*이 논문은 미래창조과학부의 재원으로 한국연구재단의 <실감교류인체감응솔루션> 글로벌프론티어사업(2018M3A6A3058649), 과학기술인문융합연구사업(2017M3C1B6070980)의 연구비 지원을 받아 수행된 연구임

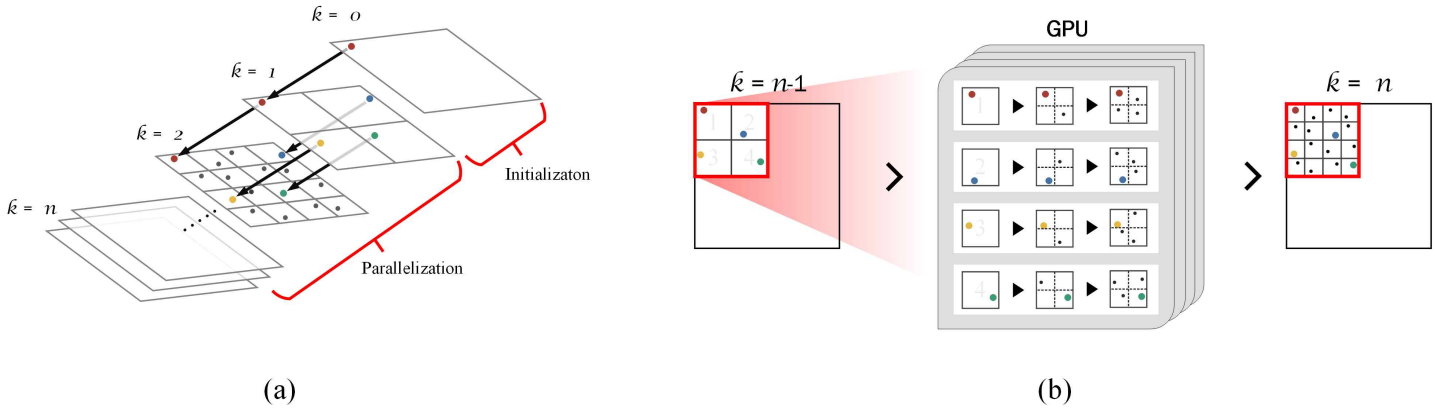

그림 1. (a) 본 알고리즘의 개략도. 병렬화 단계에서 읽을 샘플 4개를 만들어주는 초기화와 이후 샘플링을 위한 병렬화 단계로 이루어진다. 각 영역(같은 색)의 샘플이 이후 단계로 복사된다. (b) 본 알고리즘 샘플링의 병렬화 과정. 각 색의 점은 각 사분 면의 이전 레벨의 샘플을 의미한다.

Jittered [2, 3] 샘플링은 전체 공간을 일정한 간격의 하위 구역으로 나누어서 한 구역에 하나의 샘플을 랜덤으로 위치시킨다. 반면에, 이 기법은 샘플링 수행 시, 각 단계에서는 blue-noise 분포가 유지되질 못한다. 이에 샘플링을 progressive하게 수행하는 Progressive Jittered [4] 샘플링과 같은 기법이 제안되었다.

이에 Progressive Jittered 샘플링 기법은 단계가 진행됨에 따라 이전 단계의 샘플셋을 가져온 후, 각 샘플을 기준으로 4구역으로 분할 후 새로운 샘플들을 추가하여 샘플셋의 progressive성을 유지한다.

Parallel Poisson disk [5] 샘플링에서는 샘플링 구역을 grid 단위로 분할해 충분한 거리를 주어 phase group을 만들어 같은 phase group 단위로 동시에 샘플을 그린다.

CPU 기반의 샘플링 기법에서는 샘플의 개수가 증가할수록 선형으로 생성시간이 증가하기에, 본 연구는 progressive한 샘플들을 병렬화를 통해 다량으로 생성할 수 있는 Parallel Progressive Jittered 샘플링 알고리즘을 제안한다. 이 병렬화 과정에서 Progressive Jittered 샘플링의 알고리즘적 의존관계를 분리해야 하는 문제가 있으며 이를 해결하는 방법에 대해 논의한다.

3. 알고리즘

본 연구의 알고리즘은 난수표를 사용하여 샘플링시 랜덤성을 부여하여 좌표를 계산하도록 한다. 초기화 과정을 통해 처음 읽을 수 있는 4개의 샘플을 만든 후 반복적인 병렬화 과정을 거쳐 샘플 셋을 생성한다.

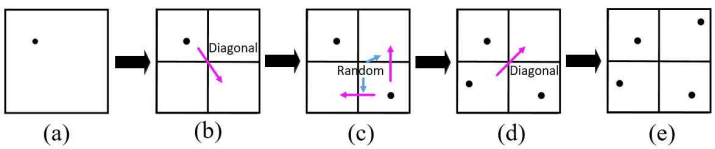

그림 2. 한 구역 단위 내의 PJ [4]의 샘플링 패턴

3.1. 난수표의 생성

본 연구에서는 샘플의 랜덤한 배치를 위해 난수를 사용해 샘플의 위치를 계산하기 때문에, 본 알고리즘에서는 CPU에서 난수를 생성한 후 난수표를 생성하여 텍스처에 저장하여 GPU에 업로드한다.

3.2. 초기화

본 알고리즘은 샘플의 위치 계산을 할 시, 이전 단계의 구역의 샘플을 4개씩 읽어와 다음 점을 병렬적으로 계산한다. 그림 1의 (a)에서 초기화와 그 이후 진행되는 병렬화 단계를 보여준다. 병렬화 과정에서 4개의 점 단위로 다음 레벨의 샘플들을 연산을 할 수 있도록 초기화 과정에서 첫 4개의 점을 생성한다.

3.3 병렬화

PJ [4]의 경우 이전 샘플을 사용해 다음 샘플을 생성하는 이전 결과에 의존성이 있는 계산을 한다. 이는 병렬화가 힘든 요인이 된다. 본 알고리즘은 이 알고리즘 간의 순차적인 의존관계를 해결하기 위해 그림 1의 (b)와 같이 이전 단계에서 계산된 샘플을 4개 단위로 읽는다. 이 4개의 샘플들은 각기 새로운 샘플 4개를 병렬적으로 생성한다. 이 때 하나의 구역에 대한 추가 생성과정은 그림 2와 같다. 먼저, (a)와 같이 이전 레벨의 샘플을 읽어와 이 샘플을 기준으로 사분면을 나눈다. 그 후 (b)와 같이 대각선 방향의 구역에 샘플을 추가한다. 그 후 (c) 단계에서 남은 두 개의 사분면 중 하나를 난수표를 사용하여 선택하여 샘플을 생성한다. 마지막으로, (d) 순서에서 그 샘플의 대각선 방향의 사분면에 샘플을 생성한다. 이를 통해 (e)와 같이 총 4개의 샘플이 생긴다. 이 과정들을 통해 병렬화 연산과 함께 4개의 분할 구역이 가지

고 있는 progressive함을 유지하였다.

4. 결과 및 토론

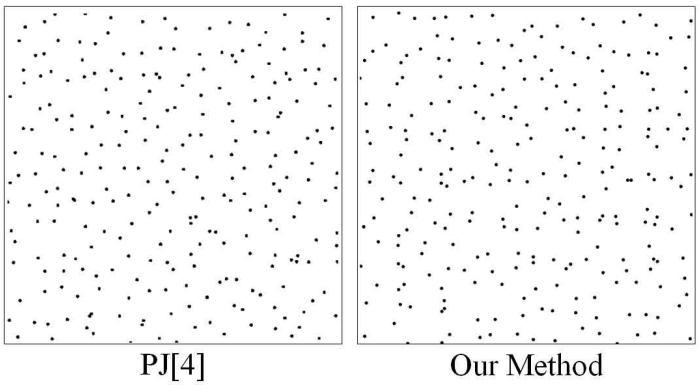

그림 3. PJ [4]와 본 연구 제안기법의 샘플 이미지 비교

본 알고리즘의 테스트는 Intel Core i7-5960X, NVIDIA GeForce GTX 980Ti의 환경에서 진행되었다. 그림 3에는 PJ [4] 샘플링과 본 연구에서 제안한 샘플링 기법으로 256개의 샘플을 생성한 결과를 나타내었다. 이를 통해, 본 기법 적용 시 샘플의 분포가 PJ와 유사한 결과를 나타냄을 알 수 있다.

표 1은 샘플의 개수에 따른 PJ와 제안 알고리즘의 수행시간을 보여준다. 샘플의 개수를 16개에서부터 약 4M 개까지 증가시키며, 수행시간을 측정하였다. 이를 통해 소수의 샘플링을 수행 시, GPU를 준비하는 데 드는 비용으로 인해 CPU 기반 알고리즘의 성능이 높음을 보였으나, 샘플 수가 증가함에 따라 본 논문에서 제안한 알고리즘이 $\log_4 n$의 연산만으로 샘플링이 가능하여 유리함을 볼 수 있다.

그림 4는 본 논문에서 제안하는 방법과 PJ의 성능 비교를 보인다. 이를 통해 PJ와 비교하여 본 알고리즘의 경우 샘플 수의 증가에 따른 연산시간 증가폭이 적으므로, 다량의 샘플링 시 제안 알고리즘을 사용하는 것이 나은 것을 보였다. 본 연구가 기존 연구보다 성능향상과 함께 유사한 품질의 샘플링의 결과를 확보할 수 있는 것을 확인하였다.

5. 결론

본 연구는 CPU 기반으로 제안된 progressive한 샘플링 알고리즘을 단위 구역으로 분할하여 GPU 기반으로 병렬화한 Parallel Progressive Jittered 알고리즘을 제안하였다. 본 논문이 제시한 기법은 기존 샘플링과 비교하여 샘플의 개수가 증가할수록 기존 알고리즘과의 성능의 차이가 벌어지는 것을 볼 수 있었으며, 기존 알고리즘과 유사한 품질을 보였다.

표 1. 샘플링 성능 측정 결과

샘플 수	PJ[4]	제안 알고리즘
16	0.003ms	0.54ms
64	0.007ms	1.036ms
256	0.021ms	1.537ms
1024	0.076ms	2.035ms
4096	0.283ms	2.536ms
16K	1.158ms	3.03ms
65.5K	4.309ms	3.534ms
262K	10.048ms	4.039ms
1M	31.922ms	4.533ms
4.19M	148.718ms	5.025ms

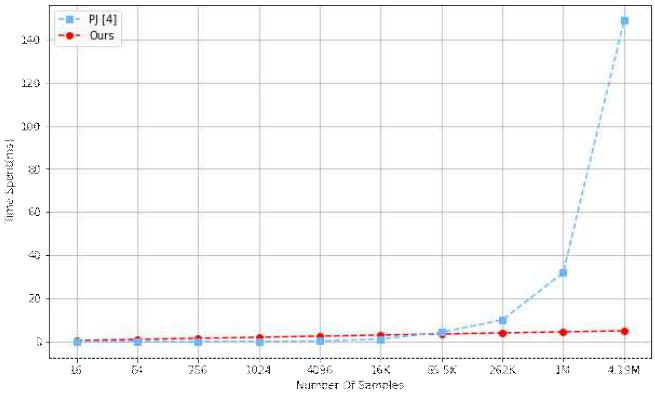

그림 4. PJ [4]와 본 연구 제안 기법의 성능 비교

참 고 문 헌

[1] Robert L. Cook, “Stochastic sampling in Computer Graphics,” in ACM Transactions on Graphics, vol. 5, 1986.

[2] James T. Kajiya, “The Rendering Equation,” in Proceedings of the SIGGRAPH, vol. 20, 1986.

[3] Mark A. Z. Dippe et al.,. “Antialiasing Through Stochastic Sampling,” in Proceedings of the SIGGRAPH, vol. 19, 1985.

[4] P. Christensen et al., “Progressive Multi-Jittered Sample Sequence, “ in Proceedings of the Eurographics, vol. 37, 2018.

[5] Wei. L. “Parallel Poisson Disk Sampling” in ACM Transactions on Graphics, vol. 27, 2008.