균일한 렌더링 부하를 위한 영역기반의 기여도 컬링

이범종° 윤종현 박종승
인천대학교 컴퓨터공학과
{leeyangg°, jhyoon, jong}@incheon.ac.kr

Region-Based Contribution Culling for Uniform Rendering Workload

Bum-Jong Lee°, Jong-Hyun Yoon, Jong-Seung Park
Department of Computer Science and Engineering, University of Incheon

요 약

가상공간의 시각적 렌더링에 있어서 카메라의 위치와 방향에 따라서 현재 보여지고 있는 장면의 복잡도가 달라지게 된다. 단순한 장면과 복잡한 장면이 혼합되어있는 경우에 렌더링의 프레임율이 크게 달라지게 되어서 사용자의 가상공간의 항해에 대한 카메라 이동 시에 끊김 현상이 느껴지게 된다. 본 논문에서는 장면이 복잡도에 크게 의존되지 않으면서 지속적으로 안정된 프레임율을 보장할 수 있는 기여도 컬링 기법을 제시한다. 컬링 기술은 복잡한 장면의 빠른 렌더링을 위해 필수적으로 사용되어왔다. 자연스럽고 빠른 렌더링을 위해 본 논문에서는 이미지 공간에서 모든 객체에 대한 바운딩 사각형의 넓이를 빠르게 계산한다. 영상 영역에서의 계산된 넓이가 작은 객체들을 프레임율을 만족시키도록 렌더링 파이프라인에서 제외시킨다. 전체적으로 기여도가 높은 객체들은 최대한 포함시키기 때문에 렌더링의 질을 보장한다. 복잡한 환경에서의 기여도가 작은 많은 객체들을 배제시킴으로써 속도를 향상시킨다. 실제 대도시의 일정 영역에 제안한 기법을 적용하여 본 결과, 복잡한 장면들이 질적인 저하 없이 균일하고 빠른 렌더링을 보장한다는 것을 보여주었다.

1. 서론

복잡한 환경에 대한 장면을 렌더링할 때 기하 데이터의 복잡성 때문에 렌더링의 시간이나 렌더링의 질 중 한쪽에 초점을 맞춰야 하는 문제가 있다. 기존의 연구들은 복잡한 환경의 실시간 렌더링이 가능한 높은 질에 대해 초점을 맞춰왔다. 객체들을 많이 사용하여 렌더링을 할 경우 렌더링의 질은 높아질 수 있으나 속도는 떨어진다.

일부 기술들은 복잡한 환경을 빠르고 좋은 질로 렌더링할 수 있는 방법에 대해 제안했다. 컬링은 장면에서 보이지 않는 객체를 렌더링 하지 않음으로써 렌더링 속도를 향상시킬 수 있는 개념이다. 컬링의 대표적인 방법은 후면 컬링 (Backface Culling), 시각 절두체 컬링 (View Frustum Culling), 포탈 컬링 (Portal Culling), 차폐 컬링(Occlusion Culling) 등으로 나눌 수 있다[1]. 후면 컬링은 카메라가 바라보는 방향과 객체의 각 표면의 노말로 계산될 수 있고 두 방향이 같으면 컬링되는 것이다. 시각 절두체 컬링은 뷰 볼륨 외의 객체들을 선별하는 것으로 뷰 볼륨 외부의 객체들은 모두 컬링하고 뷰 볼륨 내부에 있는 객체들만 렌더링한다. 포탈 컬링은 공간을 포탈이나 셀로 나누고 뷰 영역에 포함되지 않는 포탈이나 셀에 포함되어 있는 객체를 컬링하는 기술이다. 차폐 컬링은 어떤 객체에 의해 가려짐으로써 보이지 않게 되는 객체를 렌더링하지 않는 기술이다. 차폐 컬링은 또한 픽셀당 검사보다 큰 모양에서 보이는 객체인지를 분석하는 알고리즘이다[2].

복잡한 환경의 렌더링 속도를 향상시키기 위한 또 다른 방법은 LOD(Levels of Detail) 기법을 사용하는 것이다. LOD는 카메라와 객체의 거리를 기반으로 멀리 있는 객체의 경우 정밀도를 감소시키는 방법이다. 정밀도를 감소시켜도 멀리 있기 때문에 장면의 질은 떨어지지 않는다. 같은 개념으로 기여도 컬링 (Contribution culling)은 렌더링될 이미지에서 모델의 기여도에 따라 컬링하는 것을 말한다. LOD는 정적 LOD와 동적 LOD가 있다. 정적 LOD에서 객체의 정밀도는 정해져 있고 카메라와의 거리에 따라서 레벨별로 바꿔치기 해가며 렌더링한다. 연산이 간단하기 때문에 속도가 빠르다는 장점이 있고 여러 레벨의 객체를 갖고 있기 때문에 메모리 낭비가 있고 거리에 따라서 객체의 단계가 급격하게 변하기 때문에 튐현상이 발생한다. 동적 LOD는 카메라와 물체의 거리에 따라서 실시간으로 객체의 정밀도를 변화시키는 기법으로 거리에 따라 자연스럽게 상세단계가 이뤄지기 때문에 튐현상이 발생되지 않고 메모리 낭비도 없다.

본 논문에서는 복잡한 환경의 빠른 렌더링을 위해 기여도 컬링 방법에 대해 설명한다. 이미지 공간에서 객체의 바운딩 볼륨의 넓이가 매우 작은 것들은 장면에서 기여도가 떨어진다. 기여도가 낮은 객체를 컬링함으로써 장면의 질을 보장하고 렌더링 속도를 향상시킬 수 있는 방법을 제안한다.

2장에서는 복잡한 장면의 렌더링 방법에 대해 설명한다. 3장에서는 제안된 기여도 컬링 방법에 대해 설명한다. 4장에서는 복잡한 환경에 대한 실험결과를 기술한다. 마지막으로 결론 및 향후 연구과제가 5장에서 설명된다.

2. 복잡한 장면에 대한 빠른 렌더링

실시간으로 복잡한 환경을 렌더링하기 위한 연구가 활발히 진행되어왔다. 한 장면에서 보이는 객체와 보이지 않는 객체를 계산하는 방법들에 대해 많은 연구들이 진행되어왔다. 도시환경에 대한 선 공간 (line space)에 대한 분할을 사용한

가시검사가 제안되었다[3]. 매우 큰 지형자료의 빠른 렌더링 또한 연구되었다. Gudukbay[4]은 지형 자료의 깊이 정보를 보존하는 거리기반 기준을 소개하였다. Yin[5]은 군집 환경에서 큰 지형들에 대한 효율적인 렌더링 시스템을 연구하였다. Tecchia[6]은 충돌 검사가 간소화된 복잡한 도시환경의 렌더링 방법을 연구하였다. 분산된 가시컬링 기술은 2단계 가시계산을 사용하여 크고 복잡한 장면을 렌더링하는 것을 제안하였다[7]. Cohen-Or[8]는 뷰 공간을 보이는 객체의 세트를 포함하는 셀들로 분할하는 방법을 제안했다. 그 다음에 시각 절두체 컬링에 대한 기술은 소개되었다[9]. 시각 절두체는 6개 바운딩 평면에 의해 정의될 수 있다. 객체와 6개 평면들의 교차검사를 수행함으로써 컬링되는 객체인지 아닌지를 결정할 수 있다. 한 점이 시각 절두체의 모든 평면의 안쪽에 있으면 그 점은 시각 절두체 안에 존재한다. 시각 절두체 컬링에 대한 몇 가지 최적화 방법이 연구되었었다. 8분 공간 검사는 교차 검사의 수를 감소시키기 위해 시각 절두체를 각 축에 따라 반으로 분할하여 8개의 공간을 생성한다. 그 다음 바운딩 볼륨이 바운딩 볼륨과 가장 가까운 세 평면 안쪽에 있으면 그 객체는 시각 절두체의 모든 평면 안쪽에 있는 것이다. 또 다른 최적화 개념은 한 노드의 바운딩 볼륨이 완전히 시각 절두체의 평면 중 한 평면의 안쪽에 있다면 노드의 자식들의 바운딩 볼륨들 또한 완벽하게 그 평면의 안쪽에 있게 된다. 그러므로 그 평면에 대한 검사는 그 노드의 하위 트리에서 완전히 제외시킬 수 있으므로 검사의 수를 감소시킨다.

시간의 연관성과 카메라 이동 연관성은 교차 검사를 감소시킬 수 있는 또 다른 최적화 방법이다. 시간 연관성은 어떤 바운딩 볼륨이 이전 프레임에서 시각 절두체 평면 중 한 평면의 바깥쪽에 있었다면 그 바운딩 볼륨은 이번 프레임에서도 같은 평면의 바깥쪽에 있게 된다는 것이다. 카메라 이동과 회전 연관성은 공간에서 이동 및 회전을 할 때 단지 한 축 둘레로 회전하거나 이동하는 것은 공통적으로 카메라에 적용된다는 것이다. 이러한 경우에 이전 프레임에서 평면 바깥에 있던 많은 객체들은 이번 프레임에서도 바깥에 남는다. 그림 1은 시각 절두체 컬링, 후면 컬링, 차폐 컬링에 대한 예를 그림으로 보여준다. 객체의 일부 중 점선으로 표시되어 있는 부분들은 컬링되어 렌더링되지 않는 면이고 실선으로 되어 있는 부분들은 렌더링되는 면을 나타낸다.

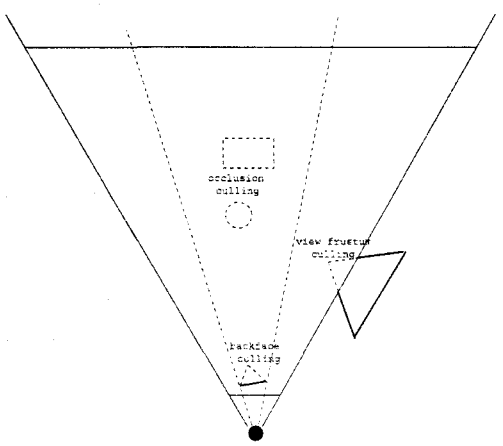

그림 1. 객체들에 대한 시각 절두체와 바운딩 볼륨 계층구조의 대응관계

시각 절두체 컬링은 보이지 않는 객체들을 제거함으로써 렌더링 속도를 향상시킨다. 본 논문에서는 시각 절두체 컬링을 수행한 후, 화면 공간으로 투영된 객체의 바운딩 사각형의 넓이를 기반으로 한 기여도 컬링을 수행한다. 거대한 수의 객체들을 포함하는 복잡한 환경을 렌더링할 때 렌더링 속도는 느려지지 않는다. 보이는 객체의 수에 의존하는 것을 피하기 위해 본 논문에서는 화면 공간에서 작은 객체들은 그 객체가 카메라 위치에 가까이 있을지라도 컬링한다.

3. 영역기반의 기여도 컬링

객체에 대한 다각형의 넓이를 3D 공간에서 계산하게 되면 넓이가 매우 작지만 카메라에 매우 가까이 있어서 실제 화면에서는 크게 보이는 객체를 제외시킬 수도 있고 실제 넓이는 크지만 카메라로부터 매우 멀리 떨어져 있어서 실제 화면에서는 작게 보여 제외시켜도 되는 객체를 포함 시키는 현상이 발생할 수 있다. 본 논문에서는 다각형의 넓이를 2D 공간에서 계산하는 방법으로 접근한다. 제안된 방법의 전체 과정이 그림 2에 나타난다.

단계 1: 시각 절두체 계산
- 현재 카메라 위치에 대한 시각 절두체를 계산

단계 2: 시각 절두체 외부의 객체들 판별
- 시각 절두체 외부에 존재하는 객체들은 이후 계산에서 제외함.
- 내부에 존재하는 객체들만 Step3부터 적용

단계 3: 모든 객체의 정점을 화면공간으로 투영
- 모든 객체의 정점들을 화면공간으로 투영

단계 4: 투영된 객체의 바운딩 사각형의 넓이를 계산
- 투영된 각 객체들을 감싸는 최소의 바운딩 사각형의 넓이를 계산

단계 5: 객체를 렌더링
- 임계치 이상의 넓이의 객체들을 후면 컬링, 차폐 컬링을 사용하여 렌더링

그림 2. 제안된 방법의 전체 과정

그림 2의 단계 1은 2장에서 설명한 시각 절두체를 계산하는 과정이다. 시각 절두체 내의 객체들을 선별하기 위해 계산을 한다. 단계 2는 시각 절두체 내에 있는 객체들을 선별하고 시각 절두체 내에 있는 객체들은 단계 3부터 적용하여 렌더링하고 시각 절두체 외부에 있는 객체들은 렌더링 파이프라인에서 완전히 제외시킨다. 단계 3에서는 시각 절두체 내의 모든 객체의 정점들을 렌더링 파이프라인의 투영행렬을 사용하여 화면 공간으로 투영한다. 단계 4에서는 투영된 객체의 최소 바운딩 사각영역의 넓이를 계산한다. 단계 5에서는 설정한 임계치와 계산된 바운딩 사각형의 넓이를 비교하여 임계치보다 큰 객체들을 후면 컬링과 차폐 컬링을 사용하여 렌더링하게 된다.

4. 실험 결과

제안된 방법의 효율성을 증명하기 위해 본 논문에서는 윈도우 플랫폼에서 C++와 DirectX 9.0을 사용하여 렌더링 엔진을 구현하였다. PC는 2.8GHz 펜티엄 4 프로세서와 512MB DDR RAM을 보유하고 64MB DRAM의 ATI RADEON 9200 GPU의 그래픽 카드를 사용하였다.

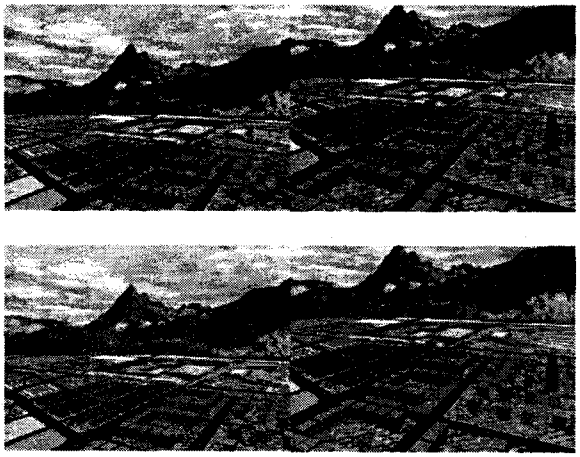

그림 3. 고정된 다각형 방법(위쪽)과 고정된 가시거리(아래쪽)을 사용하여 렌더링된 장면들

비교를 목적으로 간단한 두 가지 방법을 구현하였다. 첫 번째 간단한 방법은 일정한 수의 다각형만을 렌더링하는 방법이다. 그림 3의 첫 번째 줄은 다각형의 수를 고정시키고 복잡한 도시환경을 렌더링한 것이다. 왼쪽의 장면은 131,600개의 다각형으로 렌더링한 것이고 오른쪽의 장면은 342,160개의 다각형으로 렌더링한 것이다. 두 번째 간단한 방법은 고정된 가시거리 방법이다. 그림 3의 두 번째 줄은 가시거리를 고정시키고 도시장면을 렌더링한 것을 보여준다. 왼쪽의 장면은 650미터의 가시거리 이내의 객체들만 렌더링함으로써 151,152개의 다각형을 사용하여 렌더링되었다. 오른쪽의 이미지는 1500미터의 가시거리 이내의 객체들만 렌더링함으로써 342,160개의 다각형을 사용하여 렌더링되었다.

그림 4와 같은 시각 확대 실험과 카메라 회전 실험들에 대해 제안된 방법은 빠른 렌더링 속도에 의해 부드러운 회전이 가능했다. 그림 4(위쪽)에서 카메라는 왼쪽 위에서 오른쪽 아래의 순서로 점점 확대 되었다. 장면이 확대될수록 영역이 매우 작아서 보이지 않던 객체의 영역이 커짐에 따라 보이게 된다. 그림 4(아래쪽)에서 이미지들은 카메라의 위치를 고정시키고 각각 0, 60, 120, 180, 240, 300도 회전됐을 때 장면을 렌더링한 것이다. 제안된 방법은 시각 절두체 컬링의 방법과 비교하였을 때 질의 저하 없이 빠른 렌더링 속도를 유지하였다.

그림 5의 첫 번째 줄의 장면들은 어떤 컬링도 사용하지 않고 모든 객체를 렌더링한 것을 보여준다. 그림 5의 두 번째 줄의 장면들은 시각 절두체 컬링을 사용하여 렌더링한 것이다. 그 방법은 단지 시각 절두체 안에 있는 객체들만 렌더링하기 때문에 첫 번째 방법보다 더 빠르고 렌더링의 질은 똑같이 나타난다. 그림 5의 세 번째 줄의 장면들은 제안된 방법을 사용하여 렌더링한 것이다. 다각형의 수는 항상 시각 절두체 컬링 방법보다 적거나 같다. 하지만 렌더링의 질은 어떤 컬링도 사용하지 않은 방법만큼 좋게 나타난다. 제안된 방법은 많은 수의 매우 작은 객체들과 거의 가려진 객체들을 렌더링하지 않는다.

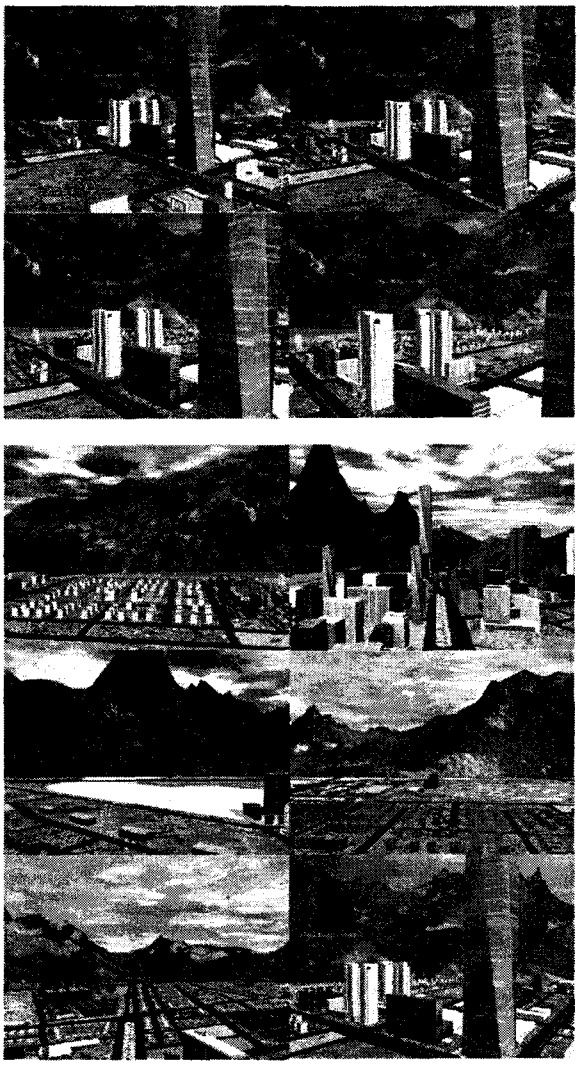

그림 4. 제안된 방법을 사용하여 렌더링된 장면 : (위쪽) 확대, (아래쪽) 회전

표 1과 표 2는 그림 5에서 보여지는 각 방법들에 대한 다각형의 수와 프레임 처리속도를 비교한 것이다. 어떤 컬링도 사용하지 않은 방법, 시각 절두체 컬링 방법, 제안된 방법은 각각 NoCulling, FrustumCulling, Proposed라고 기록하도록 한다.

표 1. 다각형 수의 비교

method	view1	view2
NoCulling	1,317,503	1,317,503
FrustumCulling	170,357	232,368
Proposed	130,237	210,560

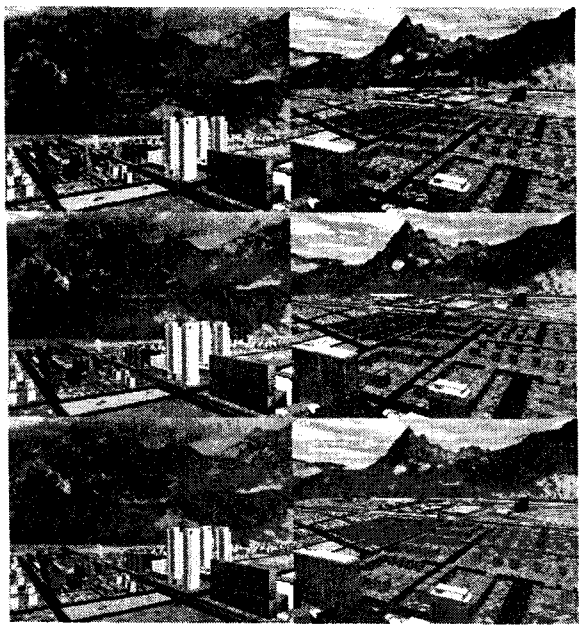

그림 5. 비교를 위한 여러 방법들에 대한 렌더링된 장면 : 컬링을 사용하지 않은 경우, 시각 절두체 컬링을 사용한 경우, 제안된 방법을 사용한 경우.

표 2. 렌더링 프레임 처리속도의 비교.

method	view1	view2
NoCulling	1	1
FrustumCulling	50	44
Proposed	53	46

그림 6(위쪽)은 카메라 의 회전 각도에 대한 다각형의 수를 나타내고 그림 6(아래쪽)은 카메라 회전 각도에 대한 프레임 처리속도를 보여준다. 프레임의 처리속도는 렌더링되는 다각형의 수에 비례하기 때문에 두 그래프의 유형이 비슷하게 나타난다. 그림 6에서처럼 제안된 방법은 다각형의 수를 약 2만 개 정도 감소시키고 렌더링 속도는 더 빨라진다. 렌더링 속도는 카메라 각도가 약 300도 일 때 0.007초이고 약 45도일 때 0.09초이다. 변화의 이유는 그림 4(아래쪽)에서 확인할 수 있듯이 중요한 객체의 수가 변하기 때문이다.

5. 결론

본 논문은 영역기반의 복잡한 환경에 대한 빠른 렌더링 알고리즘을 설명했다. 화면 투영이 매우 작은 객체들은 높은 질의 장면에 대해 기여도가 작기 때문에 렌더링 파이프라인에서 제거시켜버린다. 객체의 기여도는 객체의 화면투영의 넓이에 기반하여 계산한다. 같은 프레임 처리 속도에 대해 렌더링 장면의 질은 고전적인 상세단계 접근 방법들보다 좋다.

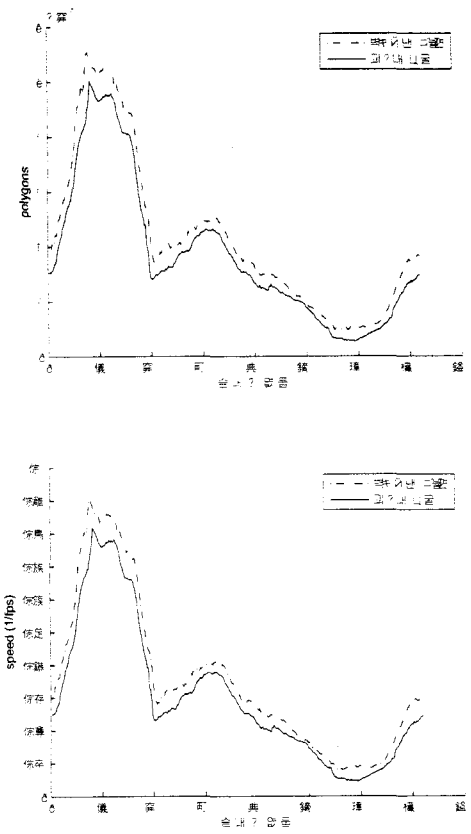

그림 6. 회전각에 대한 렌더링 성능 비교 : (위쪽) 다각형의 수, (아래쪽) 처리속도

제안된 방법은 현재 개발중인 새로운 도시지역에 적용하였다. 실험결과들은 제안된 렌더링 방법이 복잡하고 거대한 장면의 실시간 렌더링을 보장한다는 것을 보여준다.

참고문헌

[1] Akenine-Moller, T., Haines, Real-time rendering 2nd edition, 2002.

[2] Clark, J.H., Hierarchical geometric models for visible surface algorithms, Communications of the ACM, Vol. 19, pp. 547-554, 1976.

[3] Bittner, J., Wonka, P., Wimmer, M., Visibility preprocessing for urban scenes using line space subdivision, Proceedings of Pacific Graphics, pp. 276-284, 2001.

[4] Gudukbay, U., Tilmax, T., Stereoscopic View-Dependent Visualization of Terrain Height Fields, IEEE Transactions on Visualization and Computer Graphics, Vol. 8, pp. 330-345, 2002.

[5] Yin, P., Shi, J., Cluster Based Real-time Rendering System for Large Terrain Dataset, Computer Aided Design and Computer Graphics, pp. 365-370, 2005.

[6] Tecchia, F., Chrysanthou, Y., Real-Time Rendering of Densely Populated Urban Environments, Proceedings of the Eurographics Workshop on Rendering Techniques 2000, pp. 83-88, 2000.
[7] Lu, T., Chang, C., Distributed visibility culling technique for complex scene rendering, Journal of Visual Languages and Computing, Vol. 16, pp. 455-479, 2005.
[8] Cohen-Or, D., Fibich, G., Halperin, D., Zadicario, E., Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes, Computer Graphics Forum, Vol. 17, pp. 243-253, 1998.
[9] Assarsson, U., Moller, T., Optimized View Frustum Culling Algorithms for Bounding Boxes, Journal of Graphics Tools, Vol. 5, pp. 9-22, 2000.