Eurographics Symposium on Rendering (2005)
Kavita Bala, Philip Dutré (Editors)

Out-of-Core Photon-Mapping for Large Buildings

D. Fradin and D. Meneveaux and S. Horna

SIC Laboratory, University of Poitiers, France †

Abstract
This paper describes a new scheme for computing out-of-core global illumination in complex indoor scenes using a photon-mapping approach. Our method makes use of a cells-and-portals representation of the environment for preserving memory coherence and storing rays or photons. We have successfully applied our method to various buildings, composed of up to one billion triangles. As shown in the results, our method requires only a few hundred megabytes of memory for tracing more than 1.6 billion photons in large buildings.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation

1. Introduction

Rendering massively complex environments remains a difficult challenge to overcome. Though recent research have provided methods capable of rendering several hundred million triangles at interactive frame rates [ACW*99, WDS04], global illumination techniques still remain difficult, due to computing time and memory management.

For buildings, large occluders, such as walls, highly reduce the number of objects seen from a given viewpoint or region. Global illumination techniques can benefit from this information, as shown in previous approaches for the radiosity method [ARFPB90, TFFH94]. Usually, the scene is first subdivided using a binary space partitioning (BSP) decomposition [ARFPB90, TFFH94]. The resulting cells are separated by portals used to estimate a visibility graph and potentially visible sets (PVS). While PVS drastically reduce the initial number of polygons required for the computation, accurate estimation remains time-consuming and difficult to implement. Even with PVS, the information necessary to perform global illumination in a given region can easily exceed the available memory for series of potentially visible rooms containing a high number of polygons.

Christensen et al. [CB04] demonstrated that it is possible to deal efficiently with global illumination in general scenes made up of several million geometric primitives. Their method distributes the photons into *brick maps*, enabling efficient caching and memory coherence.

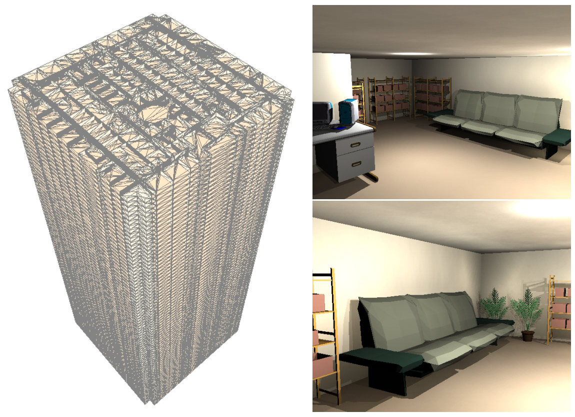
Figure 1: *(left) 80 unfurnished floors of our Tower_100 building made up of 1.07 billion triangles; (right) Inside view with global illumination and furniture.*

In this paper, our goal is to provide a method for out-of-core photon mapping with very large buildings, which are composed of several million polygons with thousands light sources and rooms. Our most complex scene contains one billion unique, non instantiated triangles. We do not make any assumption about the number of polygons contained in each room. We take advantage of a cells and portals data structure for computing view-independent global illumination. Cells and portals are essentially used for favoring data locality.

† {fradin,daniel}@sic.univ-poitiers.fr and sebhorna@free.fr

As proposed by Christensen et al. [CB04] our method creates groups of photons according to the scene subdivision, fitting with building topology. However, we keep a classical photon-map (instead of brick maps) in each cell. Portals are used to facilitate photons propagation.

Energy transfers are performed room after room: photons are cast from light sources and reflected through the currently processed room. When a photon reaches a portal, it is momentarily stored and propagated later when the corresponding adjacent room (cell) is processed. Utilizing this method, only one cell needs to be stored in memory at the same time.

Our contributions include:

- The use of cells and portals without PVS estimation for a memory efficient photon mapping in large buildings
- A memory-coherent ray propagation technique dedicated to buildings
- A memory management scheme for propagating and storing photons

This paper is organized as follows. In Section 2, we present work most related to our concerns. Section 3 provides an overview of our system while our approach to global illumination, memory management and rendering is described in Sections 4, 5 and 6. We provide results in Section 7 and discuss the presented method in Section 8.

2. Related Work

Computing global illumination for large scenes is a difficult task for two main reasons. First, the whole scene, including geometry, photometry, radiometry, and accelerating structure, does not fit in memory. Second, the computing time is high for an accurate global illumination solution.

Several authors proposed to subdivide the scene into cells either with a BSP method [ARFPB90, TFFH94] or with building-dedicated rules [MBMD98]. For each cell, portals are determined and used for creating a visibility graph indicating the set of potentially visible cells, commonly known as the *Potentially Visible Set* or *PVS*. Accurately constructing this graph is not obvious and various authors have addressed such visibility issues [TS91, CT97, COCSD03]. With PVS, the whole radiosity process can be decomposed into subproblems so that only a few cells can be stored in memory during the computations [TFFH94, MBM98, MBSB03].

Some parallel approaches dedicated to densely occluded environments have also been proposed for both radiosity [Fun96, FY97, MB99, Gar01] and Monte-Carlo-based global illumination [WBS03]. The latter approach selects the contributing light sources using an importance sampling scheme [KW00]. Using a cluster of computers, this method provides interactive frame rates. Renambot et al. used virtual walls as interfaces [RAPP97] for distributed global illumination.

Several authors have addressed the problem of ray tracing and/or photon-mapping for complex scenes. For example, Wald et al. have proposed several methods for visualizing general complex scenes [WBWS01, WDS04]. Recently, Christensen proposed an efficient method for photon-mapping in complex scenes, using a hierarchical data structure called *brick map* [CB04]. A brick map is associated with parts of a scene, including geometry, normals and irradiance. In both cases, one key idea is to exploit geometry coherence and caching.

Our aim is to compute view-independant photon maps [Jen96, Jen01] only once, and use it to produce images from any viewpoint in the scene. We do not want to make the assumption that the user position or path is known in advance. Our method relies on partitioning a scene into cells and portals based upon scene topology. When using photon-mapping, only a small number of photons travel between cells through portals. No PVS needs to be estimated a priori. Instead, portals are used to indicate which photons leave a cell and thus enter another cell as in [RAPP97] with virtual walls. Therefore, only one cell has be stored in memory rather than the entire PVS. Images are computed using portals and photon maps located in each room.

3. Work Overview

Our scene data structure contains a list of cells and portals. Each cell corresponds to a room in the building, while each portal represents a link between two adjacent cells. Such a data structure can be generated by several algorithms [ARFPB90, TFFH94, MBMD98]. In practice, we create the list of cells and portals via a modeler, designed for creating large buildings.

In each room, photons are first cast from light sources then propagated within the environment. Photon impacts are stored in several photon tables (one for each room) and Russian Roulette is used to decide whether photons are reflected. When a photon hits a portal, it is stopped and stored in a specific data structure. Its path is prolongated when the corresponding adjacent room is processed.

Once all the photons have been propagated and stored, the photon table corresponding to each room is re-organized as a kd-tree [Jen01]. Compared to the total number of photons stored in the entire building, each room only contains few photons. Therefore, the depth of each kd-tree is low and photon-map queries are very efficient during the rendering phase.

4. Photon Mapping

While processing a room, most existing radiosity methods require that both the PVS and room geometry are loaded into memory. Therefore, only a small subset of the scene geometry will be taken into account. Our photon-mapping method further localizes computations in rooms. A photon cast in a given room hits surfaces and reflects within the room. Photons reaching a portal are stored in a list associated with the adjacent room for later propagation.

4.1. Cells and Portals Representation

As stated in [PKGH97], memory coherence is a key point for accelerating ray tracing. All the geometric primitives should not be checked every time a ray is traced. Instead, groups of rays can be traced within smaller regions. When a ray gets out of one region, it is stored and propagated later, when the corresponding region is processed. In [PKGH97], regions are defined by a uniform grid. With buildings, we use large occluders (such as walls, defining cells corresponding to rooms) for preserving local information and exploiting geometry coherence: a ray can leave a region only through few "small" portals.

For each cell, the associated geometry, photometry and radiometry parameters are stored in distinct files. Some triangles, specifically dedicated to portals, contain the identifier of the corresponding adjacent room. For conservation of memory, the overall adjacency graph is never explicitly stored. Instead, the information is only accessed through portals.

We used the photon data structure proposed by Jensen [Jen01]. The following data structure represents the list of photons entering a given room.

```
InPhotons
{
   Photon inPh[MAX_TMP_PHOTONS] ;
   int totalNumber ;
   int memoryNumber ;
};

// all incoming photons (shared)
static InPhotons TP[MAX_ROOMS];
```

The array *inPh* corresponds to the $totalNumber$ of photons entering a room through its portals. *memoryNumber* indicates how many photons are stored in memory. $(totalNumber - memoryNumber)$ gives the number of photons placed on the disk. The data structure associated with a room is the following:

```
Room
{
   // Current room: photons to be propagated
   Photon allPhotonsIn[MAX_PHOTONS_PER_ROOM];

   // Photon table for the current room
   Photon impacts[PHOTONS_PER_PHOTONMAP];
   int nbOfImpacts;

   // corresponding files
   FILE *incoming, *impacts;

   // Triangle vertices and photometry
   float *vertices;   // 3*3*nbTriangles
   Properties *props; // triangle properties
   int nbTriangles;

   // Uniform grid associated with the room
   UniformGrid *grid;
}
```

During photon tracing, the photons entering a room are loaded from the disk and placed in the array $allPhotonsIn$. When a photon hits a portal, it is placed in the array TP at the position corresponding to the adjacent room. As explained above, this table contains only a few thousands entering photons for each room. When a row is full, the corresponding photons are moved onto the disk.

All the photon impacts are stored in the table $impacts$ which corresponds to the photon map. Both TP and $impacts$ are partially stored on the disk. The table $vertices$ represents the list of $nbTriangles$ triangles while $props$ indicates reflectance and other properties for each triangle. One uniform grid ($grid$) is used in each room for accelerating ray tracing. Actually, a single grid is allocated in memory and updated every time a new room is loaded.

Since the number of rooms can be high, allocating TP for all the rooms requires a large amount of memory. For each processed room, the only necessary information concerns the rooms linked via a portal. Therefore, we set a maximum number of rooms in TP and employ a LRU replacement scheme.

4.2. Photon Propagation with Cells and Portals

Photons emitted by a light source in a given room hit various surfaces corresponding to furniture, walls, or portals. Portal triangles are integrated into the uniform grid as any other triangle. However, a photon reaching a portal is not reflected. Instead, it is stored in TP, in the row corresponding to the adjacent room R_i (Figure 2).

Photon propagation is performed as long as photons remain in the current room. Every time a room R_j is processed, the table TP is updated according to the current room portals: for each room R_k adjacent to R_j, if the entry in TP does not already exist, the photons corresponding to the least recently used entry E_l are saved onto the disk and E_l is re-

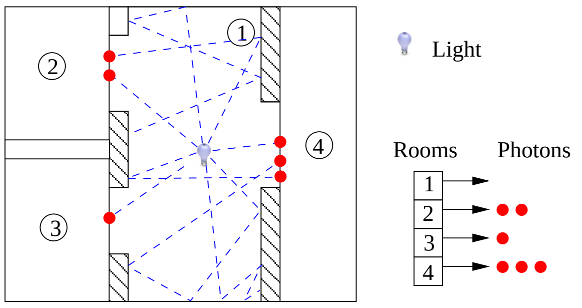

Figure 2: *Photons emitted from a light source located in R_1. The right table illustrates TP_1, TP_2, TP_3 and TP_4.*

associated with R_k. Note that photons of E_l never need to be loaded back into memory since they only need to be used later for the kd-tree construction.

The number of photons hitting a portal depends on geometry factors related to the building structure and light sources positions. Therefore, the size given to each list of TP is difficult to estimate. In our implementation, this size , $MAX\_TMP\_PHOTONS$, is identical for all the rooms and determined empirically.

During the algorithm, when an entry of TP is full, the corresponding photons are moved onto the disk, appended to a file corresponding to R_i. The block of $MAX\_TMP\_PHOTONS$ photons is written in one go.

Figure 3 shows photon impacts for two adjacent rooms separated by a portal. Only one room contains a light source and both rooms receive light flux.

Finally, once all the photons have been propagated (Russian roulette), photon tables are reorganized as kd-trees (photon-maps) in every room [Jen01].

5. Ray Casting and Data Management

Both our rendering process and photon propagation method are based on ray casting for triangular models. Non-triangular polygons are triangulated for more efficiency, as stated in [PKGH97, WBWS01]. We use an accelerating *spatial subdivision* structure for each room. In our case, we rely on a uniform grid.

We make the assumption that every room can be completely loaded in memory. Our method deals with rooms having up to 500K triangles). Thus we choose to store only one room in memory during the computation.

5.1. System architecture

Our system architecture is described in Figure 4, and the algorithm works as follows. While photons remain to be cast in the scene, (i) the scheduler selects a room R_i and loads it in memory as well as the list of photons to be cast and

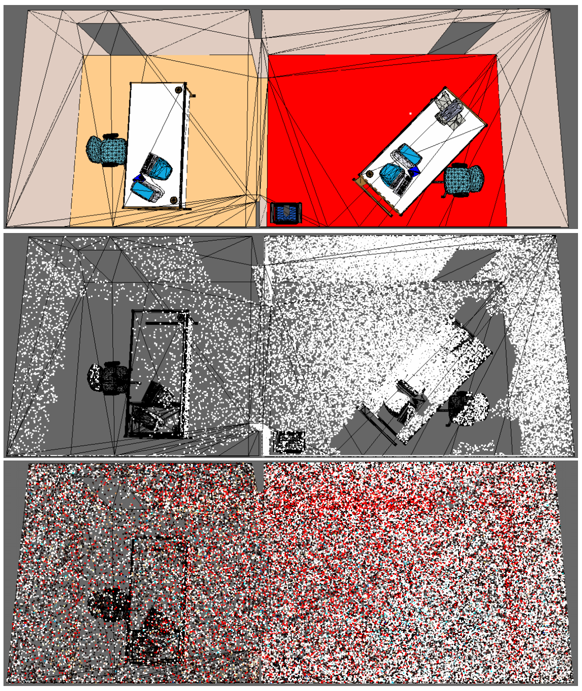

Figure 3: *Two adjacent rooms with one portal; the only light source is placed at the right. The second image shows photons corresponding to direct lighting, showing the influence of portals. The third image shows all indirect lighting photons.*

the uniform grid information; (ii) the table TP is updated according to the portals of R_i; (iii) all the photons are cast, reflected and stored either on the local table of photon or in TP; (iv) the list of local photons is then saved onto the disk, appended to the list of photons file corresponding to R_i.

A room's data is compressed on the disk in separate files so that loading one room requires reading one block of information. Note that in a given room, the number of casted photons depends on both the number of photons sent from the current light sources and the number of photons entering the room through portals. However, the number of photons entering through portals is small in comparison. On the average, 5% of the total photons come from portals in our test cases.

5.2. Scheduler Ordering Strategy

Our aim is to reduce disk accesses and cast as many photons as possible every time a room is loaded into memory. Statistically, the more photons casted in one room will result in more photons propagated in adjacent rooms. If only a few photons are cast in a room, the impact on the overall convergence of the algorithm is insignificant Therefore, our scheduler selects the room having the highest number of photons to be processed.

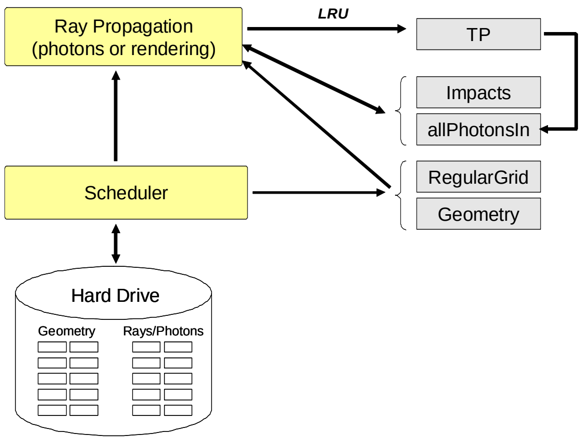

Figure 4: *Program architecture; the scheduler selects one room and loads it into memory as well as the rays/photons information. Geometry and uniform grids information are compressed for saving disk space while rays/photons information is stored as is because of files updates during the algorithm.*

5.3. Data Compression

For complex scenes, compression is necessary. Our most complex building, Tower_100, requires more than 400GB including photons, compressed geometry, and grid data.

We use 4 types of files: (i) room geometry, (ii) uniform grids, (iii) list of entering photons, (iv) photon maps.

The room geometry is represented as a list of triangles with photometric and radiometric properties. Each room is stored in a compressed file as well as its associated uniform grid. Every-time the scheduler requires a room, the corresponding data and grid files are uncompressed and read into memory.

Each room is also associated with two other files: one for photon impacts and one for entering photons. These files are not compressed since they are continuously modified during the lighting simulation process.

6. Rendering

The system architecture for the rendering program is the same as the one for photon propagation. Rays reaching portals are stored in the files corresponding to the adjacent rooms for further propagation. As for global illumination computations, the whole building does not remain in memory with all the photons. Only one room is stored in memory and all the rays waiting for being traced are processed. Except for corridors, only a few rooms are needed for one single viewpoint. The total number of rooms loaded is based upon the point-to-region visibility, which is typically much smaller than the PVS, i.e. region-to-region visibility.

Indirect lighting is estimated according to the method proposed in [Jen01] with a constant kernel. This leads to discontinuities at the portals boundary, but this problem can be reduced using density estimation. For direct lighting, shadow rays are cast toward light sources.

One of the most difficult challenges for rendering large buildings concerns the high number of light sources. In the case of our Tower_100 building (see Section 7), 60K light sources have to be taken into account. It is unreasonable to send 60K secondary rays every time a primary ray hits a triangle.

Similar to PVS, it is possible to estimate the set of visible light sources for each room. From the light source's point-of-view, this corresponds to point-to-region visibility. This method allows us to compute the set of rooms potentially lit directly by a given light source.

As a first estimation, point-to-region visibility can be computed during photon propagation. Every time a primary photon hits a portal, we can store the visibility information of the light source with the adjacent room. This technique allows us to compute light source-to-region without overhead. However, for a given light source, most photons do not hit portals. Therefore, the visibility estimation is not very accurate.

It would also be possible to use portals and the Plücker representation for estimating light sources-to-region visibility [TH93]. We have chosen a simpler method, based on a stochastic estimation [ARFPB90], even though a few distant light sources can be missed. For all light sources in each room, portals are randomly sampled. Rays are cast from the light source through the samples on the portal to determine visibility. Every time a ray hits a portal, the list of visible light sources of the corresponding room is updated. This process is very fast since the adjacent room furniture is not used.

This method drastically reduces the number of light sources taken into account for each viewpoint. Moreover, for the images we have computed, we could not visually detect any artifact due to the possibly missing light sources.

7. Implementation and Results

We used a Dual Intel Xeon 2GHz processors with 2GB of RAM and a 1TB RAID/disk. We applied our method to several buildings, including one with characteristics close to [CB04]. Some of them have only few very furnished rooms, some others have a lot of rooms, duplicated floors, randomly placed objects, etc.

7.1. Scenes Description

The following buildings have been designed with our modeler. Our test scenes are illustrated in Figure 5 and database information is given in Table 1 and 2.

Building	# Polygons	# rooms	# lights	# floors
L-Shape	336 500	27	24	2
Z-Building	1 074 000	22	15	1
Octagon	5 250 000	232	155	3
Tower_100	1.074 billion	17.8K	61K	100

Table 1: *Building properties for 4 test scenes.*

Building	Avg tri. in room	Max tri. in room	Disk space uncompr.	Disk space compr.
L-Shape	12 020	68 400	37.9MB	3.55MB
Z-Building	48 816	217 429	127MB	10.4MB
Octagon	22 621	451 606	624MB	54.8MB
Tower_100	60 134	172 772	110 GB	8.5 GB

Table 2: *Database information for our 4 test scenes with (i) average number of triangles in the rooms, (ii) max number of triangle in the rooms (iii) uncompressed database size (iv) compressed database size.*

For the L-Shape and Octagon buildings, furniture have been manually placed while for the Z-Building and Tower_100 we used an automatic process. The Tower_100 building contains roughly 10 million triangles per floor.

The Octagon building corresponds to the worst case for our method since it contains a few rooms with a high number of triangles (the most furnished room contains more than 400K triangles). Note that the provided database sizes do not include photon maps.

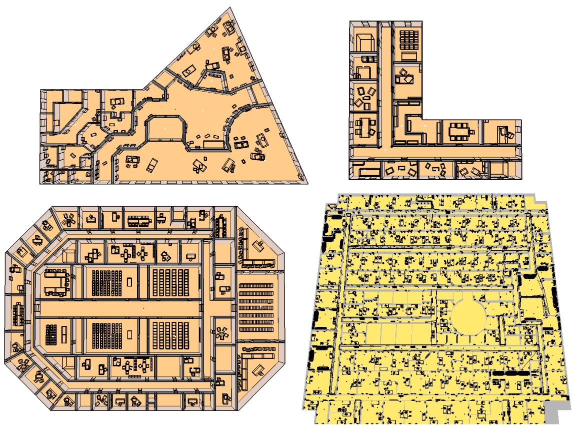

Figure 5: *Top-left: Z-Building building, 1 074 000 input triangles for one floor. Top-right: L-Shape building, 336 500 input triangles for two floors. Bottom-left: Octagon building, 5 250 000 for three floor. Bottom-right: 1.07 billion triangles for one hundred floors.*

7.2. Global illumination

A fixed number of photons is cast from each light source. For the Tower_100 building, rooms contain several light sources. 20K photons have been sent for each of them, so that about 100K photons be stored in each room (at least) for correct irradiance sampling. For the other scenes, there is at most one light source in each room and we sent 50K photons from each of them. The computing time we obtained is provided in table 3.

Building	# Photons (million)	Phot-prop. time	Phot-Map time
L-Shape	4.6	1'19"	21"
Z-Building	3	33"	1"
Z-Building_021	63	18'07"	5'24"
Z-Building_101	303	1h49'	26'02"
Octagon	30	4'04"	2'14"
Tower_024	51	30'39	4'16"
Tower_040	898	4h07'	1h34
Tower_100	1655	10h11'	3h56

Table 3: *Computing time for several buildings. # Photons corresponds to the number of photons impacts stored in the environment. Phot-prop time corresponds to the time needed for propagating photons and Phot-map time correspond to the time needed to construct all the kd-trees.*

In table 3, Z-Building_021 and Z-Building_101 correspond to building containing 20 (resp. 100) additional floors, copied from Z-Building. Tower_24 and Tower_040 correspond to the first 24 (resp. 40) floors of Tower_100. Tower_024, contains 257.385 million triangles and 2200 photons have been cast from each source for a total of 51K photons, as in the scene presented in [CB04].

For all these scenes, only 380MB of memory were required (for the most complex room): 96MB for geometry description, 109MB for photon maps and 175MB for other program data structure (BRDF, uniform grid and so on).

During the tests we made, computing time has clearly been affected by disk reads and writes. For the octagon scene, when *MAX_TMP_PHOTONS* is below 10 000, disk access frequency highly reduce the algorithm performance.

For our Tower_100 scene, a room is loaded 5.44 times in average, only one room has never been loaded (it has no portal and no light source), and one room has been loaded up to 41 times (a corridor with a high number of portals). Photons have been reflected a minimum of 1 time, a maximum of 59 times and 3.78 times in average.

Computing time is difficult to compare with the algorithm proposed by Christensen et al. since processors and disks are different. Moreover, the test scenes configuration is particular. However, according to the brute number of triangles and photons for Tower_024, our photon-tracing method provides similar results in terms of computing time.

The test buildings we presented here also correspond to the worst cases of our algorithm since we wanted to show that our method also allows to deal with precisely designed furniture in the rooms. For rooms with less furniture, it would be possible to cache several rooms and drastically increase the algorithm performances.

7.3. Rendering

Images of L-Building and Z-Building are presented in Figure 6. Computing time is generally about 15 minutes according to objects and rooms visible, light sources, and anti-aliasing parameters. For example, the top image presented in Figure 1 was computed in 9 minutes and 41 seconds.

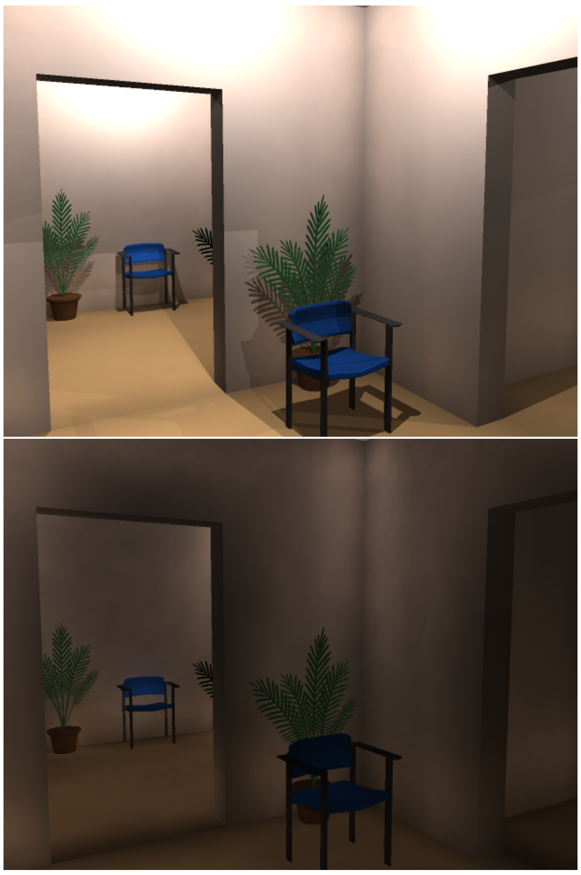

Figure 6: *(Top) Inside view of the Z-Building; (bottom) same image, indirect illumination only (photon-maps).*

8. Limitations

The presented method deals with large buildings with detailed furniture. In the case rooms contain fewer triangles, a cache system could easily be implemented. We believe this would be an appropriate solution for buildings since only few rays reach portals, at the opposite of a uniform grid structure. This would further reduce computing time.

On the images provided in this paper, the boundary between portals can be visible. This is only due to the search of photons in the kd-tree since only the triangles belonging to the current room are taken into account. This problem is commonly known and can be corrected either with a density estimation technique (e.g. [WHSG97]) or with the use of the photon-maps located in the neighbor rooms.

9. Conclusion and Future Work

This paper presents a new global illumination scheme dedicated to complex indoor scenes, based on a photon mapping approach. It combines the advantages of several methods such as virtual walls used in parallel rendering [RAPP97], memory coherent ray tracing [PKGH97] without caching and per-region grouping photons [CB04]. We have applied our algorithm to a wide variety of complex buildings. Our method can deal with scenes composed of more than one billion triangles. Computing time could be further reduced in the case several (smaller) rooms can be stored in memory with a caching system.

Aknowledgements

We wish to thank Pascal Gautron and MarK Colbert for providing helpful comments and suggestions.

References

[ACW*99] Aliaga D. G., Cohen J., Wilson A., Baker E., Zhang H., Erikson C., Hoff K. E., Hudson T., Sturzlinger W., Bastos R., Whitton M. C., Jr. F. P. B., Manocha D.: MMR: an interactive massive model rendering system using geometric and image-based acceleration. In *ACM Symposium on Interactive 3D Graphics* (1999), pp. 199–206.

[ARFPB90] Airey J. M., Rohlf J. H., F. P. Brooks J.: Towards image realism with interactive update rates in complex virtual building environments. In *ACM Symposium on Interactive 3D Graphics* (1990), pp. 41–50.

[CB04] Christensen P. H., Batali D.: An irradiance atlas for global illumination in complex production scenes". In *Proceedings of Eurographics workshop on Rendering* (June 2004), pp. 133–141.

[COCSD03] Cohen-Or D., Chrysanthou Y., Silva C. T., Durand F.: A survey of visibility for walk-through applications. *IEEE Trans. Vis. Comput. Graph. 9*, 3 (2003), 412–431.

[CT97] Coorg S. R., Teller S. J.: Real-time occlusion culling for models with large occluders. In *ACM Symposium on Interactive 3D Graphics* (1997), pp. 83–90, 189.

[Fun96] FUNKHOUSER T. A.: Coarse-grained parallelism for hierarchical radiosity using group iterative methods. In *Computer Graphics (ACM SIGGRAPH'96 Proceedings)* (Aug. 1996), pp. 343–352.

[FY97] FENG C. C., YANG S.-N.: A parallel hierarchical radiosity algorithm for complex scenes. In *PRS'97: Proceedings of the IEEE symposium on Parallel rendering* (1997), ACM Press, pp. 71–78.

[Gar01] GARMANN R.: Spatial partitioning for parallel hierarchical radiosity on distributed memory architectures. In *Proceedings of the Third Result and Review Workshop on High Performance Computing in Science and Engineering 2000* (Berlin, Germany, 2001), Springer-Verlag, pp. 478–493.

[Jen96] JENSEN H.: Global illumination using photon maps. In *Proceedings of Eurographics workshop on Rendering* (New York, NY, 1996), Springer-Verlag/Wien, pp. 21–30.

[Jen01] JENSEN H.: *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001.

[KW00] KELLER A., WALD I.: Efficient Importance Sampling Techniques for the Photon Map. In *Vision Modelling and Visualization 2000* (Saarbruecken, Germany, November 2000), pp. 271–279.

[MB99] MENEVEAUX D., BOUATOUCH K.: Synchronisation and load balancing for parallel hierarchical radiosity of complex scenes on a heterogeneous computer network. *Computer Graphics Forum 18*, 4 (Dec. 1999).

[MBM98] MENEVEAUX D., BOUATOUCH K., MAISEL E.: Memory management schemes for radiosity computation in complex enviroments. In *Computer Graphics International* (1998).

[MBMD98] MENEVEAUX D., BOUATOUCH K., MAISEL E., DELMONT R.: A new partitioning method for architectural environments. *Journal of Visualization and Computer Animation 9*, 4 (1998), 195–213.

[MBSB03] MENEVEAUX D., BOUATOUCH K., SUBRENAT G., BLASI P.: Efficient clustering and visibility calculation for global illumination. In *Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa* (2003), ACM Press, pp. 87–94.

[PKGH97] PHARR M., KOLB C., GERSHBEIN R., HANRAHAN P.: Rendering complex scenes with memory-coherent ray tracing. In *Computer Graphics (ACM SIGGRAPH'97 Proceedings)* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 101–108.

[RAPP97] RENAMBOT L., ARNALDI B., PRIOL T., PUEYO X.: Towards efficient parallel radiosity for dsm-based parallel computers using virtual interfaces. In *PRS '97: Proceedings of the IEEE symposium on Parallel rendering* (1997), ACM Press, pp. 79–86.

[TFFH94] TELLER S., FOWLER C., FUNKHOUSER T., HANRAHAN P.: Partitioning and ordering large radiosity computations. In *Computer Graphics (ACM SIGGRAPH'94 Proceedings)* (1994), pp. 443–450.

[TH93] TELLER S., HANRAHAN P.: Global visibility algorithms for illumination computations. In *Computer Graphics (ACM SIGGRAPH'93 Proceedings)* (1993), pp. 239–246.

[TS91] TELLER S. J., SEQUIN C. H.: Visibility preprocessing for interactive walkthroughs. *Computer Graphics (ACM SIGGRAPH'91 Proceedings) 25*, 4 (July 1991), 61–69.

[WBS03] WALD I., BENTHIN C., SLUSALLEK P.: Interactive global illumination in complex and highly occluded environments. In *Proceedings of Eurographics workshop on Rendering* (2003).

[WBWS01] WALD I., BENTHIN C., WAGNER M., SLUSALLEK P.: Interactive rendering with coherent ray tracing. In *Computer Graphics Forum (Proceedings of Eurographics)* (2001), Chalmers A., Rhyne T.-M., (Eds.), Blackwell Publishers, Oxford.

[WDS04] WALD I., DIETRICH A., SLUSALLEK P.: An interactive out-of-core rendering framework for visualizing massively complex models. In *Proceedings of Eurographics symposium on Rendering* (June 2004), pp. 81–92.

[WHSG97] WALTER B., HUBBARD P. M., SHIRLEY P., GREENBERG D. P.: Global illumination using local linear density estimation. *ACM Transactions on Graphics 16*, 3 (1997), 217–259.

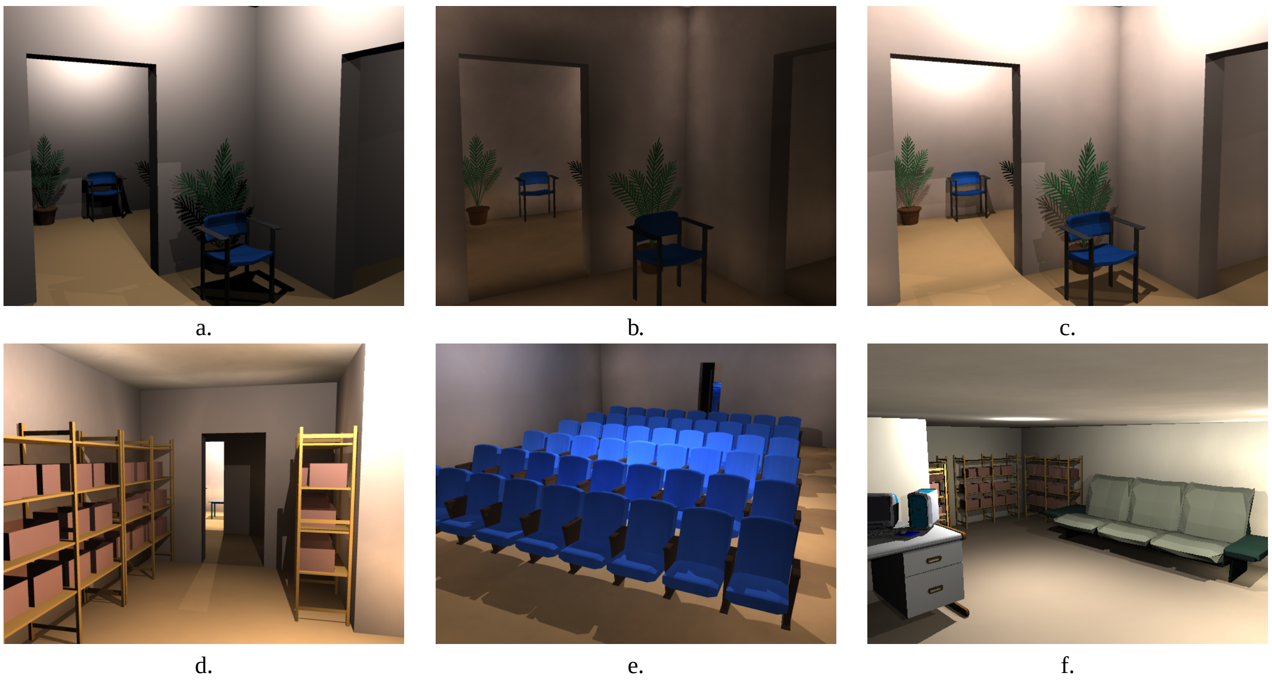

Figure 7: *a: Ray tracing Image, direct illumination only. - b: indirect illumination only (photon map). - c: final image with both direct and indirect illumination. - d,e,f: images from octagon and Tower_31 buildings.*