Implementation of the Radiosity Algorithm for Large Scale Scenes

A.S. Shcherbakov[1,2], V.A. Frolov[1,3]
alex.shcherbakov@graphics.cs.msu.ru|vladimir.frolov@graphics.cs.msu.ru
[1]Lomonosov Moscow State University, Moscow, Russia;
[2]Gaijin Entertainment, Moscow, Russia;
[3]Keldysh Institute of Applied Mathematics, Moscow, Russia

We propose an upgrade for the Radiosity algorithm that allows to efficiently apply radiosity for large scale scenes. This is achieved by considering only the patches located close to the observer. For each frame we update local form-factor matrix with a little set of patches, effectively reusing information from the previous frame in this way. Our method is completely expressed via matrix-vector operations, thus, it's GPU implementation is natural and straightforward. We achieve high occupancy for both CPU and GPU versions of the algorithm by thanks to we use special matrix of several reflections for which update operation effectively combine computations with memory operations.

Keywords: *Global Illumination, Radiosity, Large Scenes, GPU*

1. Introduction

The methods of global real-time global illumination in general can be divided into two classes: the methods that uses preprocessing/precomputation and those who does not uses it (i. e. calculate everything in dynamic). The main advantage of methods that do not use pre-processing is the fully dynamic content. However, the methods of the first group due to the precomputation and pre-integration usually are significantly better in terms of speed/quality ratio.

Our work is devoted to the problem of building methods of the "intermediate class" that combines the advantages of both classes.

2. Related works

2.1 Dynamic methods

LPV [9], VCT [4] and Volumetric Irradiance Map [20] discretizes the space around the observer and model the movement of light along the voxel grid. Main disadvantage of LPV and VCT is the high computational cost and high memory requirements. Volumetric Irradiance Map [20] is uses for half-static scenes and the secondary lighting is applied to dynamic objects, but these objects do not contribute to secondary lighting of static objects. Thus, Volumetric Irradiance Map [20] is one of those "intermediate class" examples.

Instant Radiosity [11], Reflective Shadow Map [6] and analogues creates secondary light sources, calculating the secondary illumination in the same way as the primary. The most significant disadvantage of RSM-based methods is their low precision. As the number of secondary sources increases, the accuracy increases, but the calculation time also increases significantly [2, 15, 21].

2.2 PRT Methods

Precomputed Light Transport (PRT) methods al-low to move the most complex calculations from the rendering stage to the precomputation stage.

For example, Radiance regression function [17] is based on neural networks. It is restricted by static geometry and materials. The second disadvantage of method is that it works only for point light sources.

Spherical harmonics based methods [8, 20] uses idea of "relighting", treating the dynamic light source as a linear combination of pre-processed sources. These methods suffers from light leaking which was noted in [20]. In general these methods are well-suited for outdoor but not indoor scenes.

Despite huge number of papers in real-time global illumination, radiosity [3, 19] to this day remains one of the best method by means of speed/quality ra-tion. It is used by Enlighten and Unity for exam-ple [1, 5, 12, 14] and lighting engineering programs [22, 23].

Radiosity, like the previously considered methods, uses scene discretization (but use patches instead of voxels). The pre-calculation is done by calculating the matrix of form-factors. The form-factor is a value indicating how much of the lighting is transferred from one patch to another. The calculation of the global illumination is reduced to solving the linear equation system or several matrix-vector multiplications. By expanding the calculations at the preprocessing stage, it is possible to reduce the calculation of the global illumination method to a single matrix-vector multiplication [18].

However Radiosity still have $O(N^2)$ algorithmic complexity where N is the number of scene patches. This is the main disadvantage of the algorithm, since such computational complexity is unacceptable for large scenes (with an increase in the number of sites N). Our work is aimed specifically at solving this problem. The existing approaches of hierarchical radiosity and analogues [7, 13] are light-dependent (due to they split scene to patches according to the cur-

rent lighting setup) and does not exploit its the main advantage — precomputation of form-factors.

3. Proposed approach

For a better explanation of our methods, we will begin with a few simple ideas that will be refined further.

Our **first idea** is that we can consider only M clos-est patches to the observer (where $M << N$). We as-sume lighting from other patches equals to some con-stant or evaluate it on a rougher approximation (that is, we may build cascades of discrete representations of the scene in the same way as the Cascaded LPV [10] does). This idea leads us to the fact that we can make a *submatrix* of form-factors, on which we will perform basic calculations. We will call it *local matrix* (Fig. 1, left).

However, the direct implementation of this method will not be efficient because the matrix of the form-factor for the entire scene (even if special methods are used to effectively store sparse matrices) does not fit into the computer's RAM. Therefore, reading even M^2 of random elements (shown in green in Fig. 1, left) from DRAM or external memory cannot be implemented efficiently.

Therefore, **second idea** is that it is necessary to reuse information from the previous frames as much as possible, updating only patches that were not loaded on previous frames (Fig. 1, right). This allows us to reduce the number of form-factors that are loaded into the local matrix each frame to only a few readings (or approximately M if the observer moves quickly, which, nevertheless, is significantly better than M^2).

However, reading even a few elements from the global matrix at random addresses is still inefficient for modern computing systems. During the update of the local matrix of form-factors, computing system (both CPU and GPU) will be practically idle. At the same time, after updating, when calculating the radi-ance on the local matrix of form-factors, they will be loaded and these calculations may become a bottle-neck since you need to perform at least 3 multiplica-tions of the local matrix by a vector (to account for three light bounces).

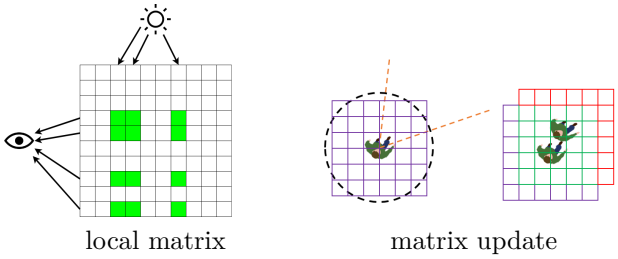


Fig. 1. Local matrix is a submatrix of global form-factor matrix. It is composed of green-labelled cells (left). Changing sites in the surroundings of the observer when moving (right).

Therefore, **third idea** is to combine updating the local matrix with the multiple reflection matrix from [18]. This will allow organizing a pipeline in which calculations are immediately performed for each form-factor obtained from the global matrix with form-factors from the local matrix. That is, we will store and update the local *matrix of several reflections*, rather than the usual local matrix of form-factors. In this case, the subsequent calculation of the global illumination is reduced to a *single multiplication of the matrix by the vector*, which, given the small size of the local matrix, is very cheap.

Finally, **the fourth idea** is that with a small movement of the observer, it is possible to update the local matrix less often than to calculate the lighting on it (for example, once in 2-5 frames). Therefore, it makes sense to reduce the cost of the stage of calculating the lighting using a local matrix of several reflections.

3.1 Baseline

The matrix of form-factors of several reflections [18] is based on the idea of the composition of form-factors. We further describe the principle of composition on simple examples.

Consider three patches with indices i, j, k. Then F_{ij} and F_{jk} — form-factors of energy transfer from the j-th path to the i-th patch and from the k-th patch to the j-th patch respectively. $Colors_j$ — color of j patch. Then value

$$F_{ij} \cdot Colors_j \cdot F_{jk}$$

shows how much of the world will go from the k-th patch to the i-th patch when the multi-reflection from the j-th patch.

After going through all patches as intermediate in the reflection, we get the complete second reflection of light from the patch number k to the patch number i:

$$F^2_{ik} \quad \sum_{j=0}^{N} F_{ij} \cdot Colors_j \cdot F_{jk},$$

where N — number of patches in local matrix.

Thus, at the preprocessing stage, it is possible to calculate reflections of an arbitrary order. The multiple reflection matrix contains a specified number of reflections.

3.2 Local matrix of multiple-reflection

In order to extend this approach to a local ma-trix in the surrounding of an observer, two ways are possible:

1. Calculation of the matrix of several reflections for the entire scene and the use of its parts.
2. "Online" matrix calculation for the observer's area using the usual matrix of form-factors (Fig. 1, right).

The first option is not applicable for large scenes since it requires to store $3N^2$ of real numbers. The proposed method implements the second option, since the matrix of form-factors is sparse and can be effi-ciently stored/accessed.

To update the matrix of several reflections, two operations are required: (1) adding a site to the matrix and (2) removing the site from it. Delete operation is implemented by zeroing the row and column of the matrix of form-factors corresponding to the deleted patch.

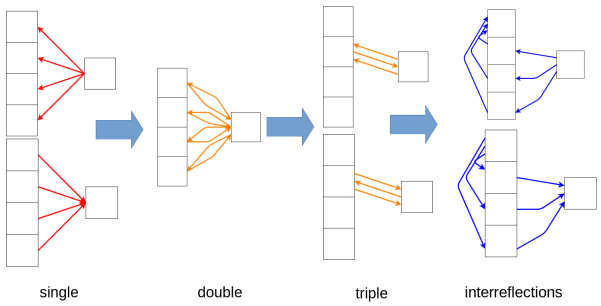


Fig. 2. Scheme of updating form-factors for local matrix of several reflections.

To add a new patch, we must consider several types of reflections. First, we need to include the usual form-factors f_{column} and f_{row} (Fig. 2, "single"). Next, triple reflections are taken into account, in which light is reflected twice from a new patch. To do this, double reflection is first calculated (Fig. 2, "double"):

$$double\_reflection = f_{column} \circ Colors \cdot f_{row} \quad (1)$$

By using double reflection *double_reflection* triple reflections can be calculated further (Fig. 2, "triple"):

$$\begin{aligned} g_{col} &= f_{column} \cdot Color \cdot double\_reflection \\ g_{row} &= f_{row} \cdot Color \cdot double\_reflection \end{aligned} \quad (2)$$

Finally, we add information on form-factors that takes into account the light reflections between patches already participating in the matrix (Fig. 2, "interreflections"):

$$\begin{aligned} g'_{col} &= F_{local} \cdot g_{col} \\ g'_{row} &= g_{row} \cdot F_{local} \end{aligned} \quad (3)$$

Thus, the form-factors for the new patch are equal to

$$\begin{aligned} g'_{col} + g_{col} + f_{column}, \\ g'_{row} + g_{row} + f_{row}. \end{aligned} \quad (4)$$

Now, in addition to adding the form-factors of the new patch, it is required to update the form-factors of the rest of the matrix. For this we use previously calculated form-factors for new patch (4):

$$\begin{aligned} F_{local} \leftarrow F_{local} + (g'_{col} + g_{col} + f_{column}) \cdot \\ \cdot (g'_{row} + g_{row} + f_{row}) \end{aligned} \quad (5)$$

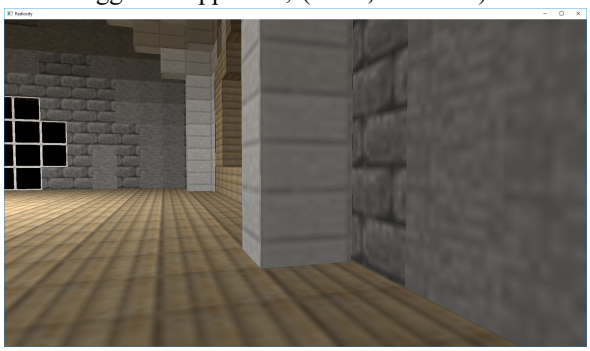
Thus, we can add and remove patches from the local matrix. The size of the matrix is limited by a constant specified by the user.

3.3 GPU implementation

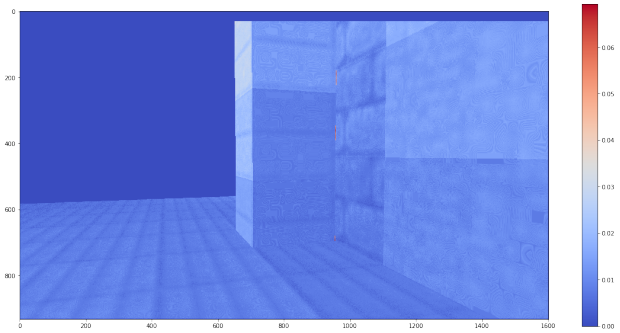
To implement the proposed method we create 6 compute shaders to update the matrix, one to delete the patch data and one shader to update the lighting. The following shaders are used for updating:

1. Calculating *double_reflection* (1) using parallel reduction on GPU.
2. Calculates triple reflections g_{col}, g_{row} (2) by multiplying *double_reflection* with form-factors.
3. Multiplying g_{col} by a matrix of form-factors to get g'_{col} (3).
4. Multiplying the form-factor matrix by g_{row} to get g'_{row} (3).
5. Adding to the matrix of information about reflections in view of the new patch (5).
6. Adding new site form-factors to the matrix.

Suggested approach; (**2 ms, 128 MB**)



Multiple reflection matrix from [18]; (**300 ms, 30GB**)



Difference image. In average is is less than 0.05.

Fig. 3. Comparison of Radiosity for the full matrix and the proposed method. Since the full matrix 63125 × 63125 does not fit in the memory of the GPU, the time was estimated.

The most computational-cost are steps 3-5. How-ever, steps 3 and 4 are just matrix-vector multiplica-tions, and the step 5 implements the addition of two matrices. Both of these these operations are to be easily implemented on GPU.

3.4 Details

The matrix of form-factors can be stored as a tex-ture in the format RGBA16F. At the same time, the matrix size is limited to 4096 pixels on some GPU due to the specific hardware limits (mobile GPUs).

4. Results

The proposed method was compared with the com-mon Radiosity on local form-factor matrix and signif-icantly outperforms it (Fig. 4).

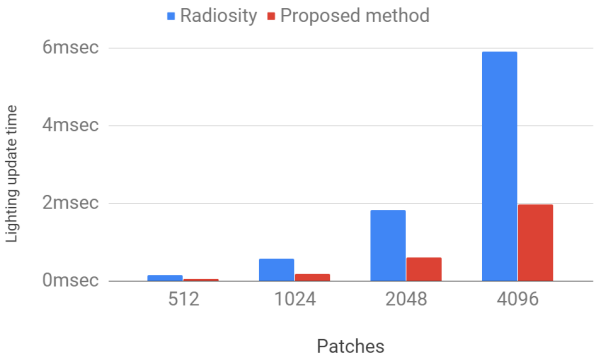


Fig. 4. Comparison of the proposed method for a local matrix of several reflections with an a common Radiosity for different local matrix sizes.

We test our method on the scene of 63125 patches (Fig. 3). We took a Minecraft scene in order to omit scene approximation algorithms in our work. At high speed, the proposed method gives an image close to the image obtained by the method of full-matrix Ra-diosity on the CPU.

5. Acknowledgments

This work was sponsored by RFBR 18-31-20032 grant.

6. References

[1] Akenine-Moller, T., Haines, E., Hoffman, N. *Real-time rendering. Fourth Edition.* – AK Peters/CRC Press, 2018. page 482.

[2] Budak, V.P., Zheltov, V.S., Kalakutsky, T.K. *Local es-timations of Monte Carlo method with the object spec-tral representation in the solution of global illumina-tion.* // Computer Research and Modeling, 2012, vol. 4, no. 1, pp. 75-84.

[3] Cohen, M., GreenBurg, D. *The Hemi-cube: A Radios-ity solution for complex it is hard to achieve environments* // Proceedings of SIGGRAPH 85 in Computer Graphics, 1985. 19. number 3. pp. 31-40.

[4] Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E. (2011, September). *Interactive indirect illumination using voxel cone tracing.* In Computer Graphics Forum (Vol. 30, No. 7, pp. 1921–1930). Ox-ford, UK: Blackwell Publishing.

[5] Cupisz, Kuba, and Kasper Engelstoft, *Lighting in Unity*, Game Developers Conference, Mar. 2015. http://www.gdcvault.com/play/1021765/Advanced-Visual-Effects-With-DirectX

[6] Dachsbacher, C., Stamminger, M. *Reflective shadow maps* // Proceedings of the 2005 symposium on Inter-active 3D graphics and games. — ACM, 2005. — . 203-231.

[7] Durand, F., Drettakis, G., and Puech, C. *Fast and accurate hierarchical radiosity using global visibility.* ACM Trans. Graph. 18, 2 (April 1999), 128-170. DOI=http://dx.doi.org/10.1145/318009.318012

[8] Green, R. *Spherical harmonic lighting: The gritty details* //Archives of the Game Developers Conference. – 2003. – Vol. 56. – p. 4.

[9] Kaplanyan, A. *Light propagation volumes in cryengine 3* // ACM SIGGRAPH Courses. 2009. Vol. 7. p. 2.

[10] Kaplanyan, A., Dachsbacher, C. 2010. *Cascaded light propagation volumes for real-time indirect illumina-tion.* In Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games (I3D '10). ACM, New York, NY, USA, 99-107.

[11] Keller A. *Instant radiosity.* — 1997.

[12] Magnusson, Kenny, *Lighting You Up with Battle-field 3*, Game Developers Conference, Mar. 2011. http://www.frostbite.com/2011/03/lighting-you-up-in-battlefield-3/

[13] Marries van de Hoef. *Real-Time Dynamic Radios-ity for High Quality Global Illumination*. 2013. Master Thesis. ICA-3220516. Utrecht university.

[14] Martin, Sam, and Per Einarsson, *A Real-Time Ra-diosity Architecture for Video Game*, SIGGRAPH Ad-vances in Real-Time Rendering in 3D Graphics and Games course, July 2010.

[15] Ou, J. and Pellacini, F. 2011. *LightSlice: matrix slice sampling for the many-lights problem.* ACM Trans. Graph. 30, 6, Article 179 (December 2011), 8 pages. DOI: https://doi.org/10.1145/2070781.2024213

[16] Pranckevicius, Aras, Jens Fursund, and Sam Martin, *Advanced Lighting Techniques in Unity*, Unity DevDay, Game Developers Conference, Mar. 2014. http://aras-p.info/blog/2014/05/05/shader-compilation-in-unity-4-dot-5/

[17] Ren, P., Wang, J., Gong, M., Lin, S., Tong, X., and Guo, B. 2013. *Global illumination with radiance regression functions.* ACM Trans. Graph. 32, 4, Article 130 (July 2013), 12 pages. DOI: https://doi.org/10.1145/2461912.246200

[18] Shcherbakov, A., and Frolov, V.*Accelerating radios-ity on gpus.* In WSCG'2017 Full papers proceedings (2017), 2701, Computer Science Research Notes 2701 Pilsen, Czech Republic, pp. 99–105.

[19] Sillion F. X. et al. *Radiosity and global illumination.* San Francisco : Morgan Kaufmann, 1994. Vol. 1.

[20] Yudintsev, A. *Scalable Real-Time Global Illumi-nation for Large Scenes.* GDC Talk. 2019. URL = https://www.gdcvault.com/play/1026469/Scalable-Real-Time-Global-Illumination

[21] Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Donald P. Greenberg. 2005. *Light-cuts: a scalable approach to illumination.* ACM Trans. Graph. 24, 3 (July 2005), 1098-1107. DOI: https://doi.org/10.1145/1073204.1073318

[22] DIAL *Dialux*. 2019. URL = https://www.dial.de/en/dialux/

[23] Relux Informatik AG *Relux*. 2019. URL = https://reluxnet.relux.com/en/