Radiosity Algorithms Running in Sub-quadratic Time *

László Szirmay-Kalos and Tibor Fóris
Department of Process Control, Technical University of Budapest,
Budapest, Műegyetem rkp. 11, H-1111, HUNGARY
szirmay@fsz.bme.hu

Abstract: This paper presents an iterative radiosity algorithm that is capable to determine the radiosity of the patches in $O(n \log n)$ time and using only $O(n)$ space. The basic idea is to compute the radiosity updates of all patches in a single direction parallely, requiring only one complete solution of the visibility problem per directions. The visibility problem is solved by painter's algorithm that is responsible for the $O(n \log n)$ time complexity, and thus it does not require large amount of z-buffer memory.

Keywords:
Radiosity method, iterative techniques, transillumination, complexity, painter's algorithm

Introduction

Classical radiosity algorithms including Jacobi or Gauss-Seidel iteration using the complete form factor matrix require $\Omega(n^2)$ time to compute the radiosities of the patches (n is the number of patches in the scene) since this matrix has n^2 number of elements, thus its calculation and its multiplication by a vector during iteration needs $\Omega(n^2)$ time. (The notation that a *function f is in $\Omega(g)$* means that there exist positive numbers C and N for which inequality $f(n) \geq C \cdot g(n)$ holds for any integer $n > N$.) An alternative to complete form factor matrix techniques — the original progressive refinement — also needs $\Omega(n^2)$ time. This comes from the fact that for classical progressive radiosity, a single step of the iteration shoots the radiosity of a selected patch towards all the other patches visible from the selected one. This requires the solution of a visibility problem and updating $n-1$ patches. None of them can be done in less than linear time without pre-processing. The usually applied discrete visibility methods — z-buffer algorithm or ray tracing — run in linear time. If each patch has self emittance, then every patch must be selected at least once to shoot its radiosity, thus the number of the required iteration steps cannot be less than n, resulting in again an $\Omega(n^2)$ algorithm.

$\Omega(n^2)$ is thus the lower bound of the time complexity. It can be shown that the widely accepted algorithms of Jacobi, Gauss-Seidel iteration or progressive refinement can really solve the problem in $O(n^2)$ time. (The notation that a *function f is in $O(g)$* means that there exist positive numbers C and N for which inequality $f(n) \leq C \cdot g(n)$ holds for any integer $n > N$.) The proof of this statement has been given in [SKM95] and is based on the recognition that the error of patch radiosities for these methods can be bounded by a geometric series where the coefficient is less than 1 and is independent of the number of patches for Jacobi and Gauss-Seidel iteration, making the number of required steps also independent of the number of patches and forcing the time complexity to be determined by the time required for a single step of iteration. For progressive refinement, on the other hand, the coefficient does depend on the number of patches, decreasing the speed of convergence by a factor of at most n. Since a single step requires $O(n)$ time, the complete solution of progressive refinement has $O(n^2)$ time complexity.

*This work has been supported by the National Scientific Research Fund (OTKA), ref.No.: F 015884

This may make us believe that $O(n^2)$ time-complexity is somehow inherent in the radiosity method. However, for very complex scenes consisting of more then a million patches, this is unacceptable. This fact initiated a lot of research to find other algorithms with better complexity characteristics.

The most important directions were the hierarchical radiosity [HSA91], randomisation or Monte-Carlo radiosity [Neu95], Quasi-Monte-Carlo radiosity [Kel95] and the pre-processing of the object space before iteration to speed up the solution of the visibility problem [Ge89], [Már95], [dB92] [MSK95], [FMSK96], [NJ96]. Some of these methods have prohibitively high memory requirement, which makes them not appropriate for the radiosity solution of complex scenes. In the radiosity solution nothing that needs more than $O(n)$ space seems acceptable from practical point of view (this requirement makes the progressive refinement superior to complete form factor matrix techniques since the first needs $O(n)$ space but the second requires $O(n^2)$ memory).

The second aspect of creating fast algorithms is the possibility of hardware implementation and taking advantage of the built-in rendering hardware of advanced workstations. Although executing a part of the algorithm by a specially designed hardware does not improve the complexity measures of the algorithm, it can speed up the execution by a significant linear factor. From this point of view, hemi-cube methods combined with z-buffering can be fast on workstations equipped with hardware z-buffers, but raises memory problems if the algorithm is implemented in software. We can conclude that it is an advantage if the algorithm does not rely on a z-buffer, since this property makes it portable to workstations having no built-in z-buffer.

The third aspect of fast algorithms is parallelization, which is not considered in this paper.

This paper's main objective is to develop methods that need less than $O(n^2)$ time, still require only $O(n)$ space, do not necessitate time-consuming pre-processing and the application of complex data structures, and can exploit the built-in rendering hardware of advanced workstations.

General concepts

In order to cross the borderline of $O(n^2)$ time complexity, complete form-factor matrix techniques must be ruled out. We can try, however, to improve or alter progressive refinement since its property that a single iteration requires only $O(n)$ time and space is promising. The problem of progressive refinement is that it does not accumulate the "knowledge" used for a single iteration but starts analysing the object space from scratch in each step. Progressive refinement must be improved by saving the calculation of one shooting patch for other patches.

Progressive refinement algorithm consists of two loops. The outer loop is responsible for selecting the shooting patches while the inner loop subdivides the directional hemisphere into finite solid angles represented usually by "pixels" on the hemicube. Theoretically, there is even a third loop that runs on the surface of the selected shooting patch, but this is usually ignored in progressive refinement, saying that a single point can represent the role of the shooting surface [CG85]. This introduces errors in the calculation if the patches are not small enough. Therefore, the elimination of this rude approximation is also included in the objectives of the new algorithm.

The property that progressive refinement does not accumulate any knowledge comes from the fact that the outer loop considers shooting patches one after the other. The visibility problem generated by different shooting patches are so different that nothing can be saved for the next computational cycle. The calculation in the inner loop is easier since all the different directions can be analysed at once by the solution of a single visibility problem.

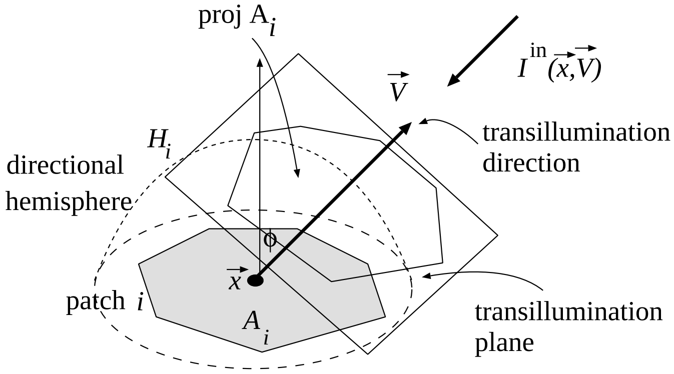
Thus the main idea is the exchange of loops, that is, the outer will be responsible for the different directions while the inner loop will consider the different patches, as suggested by the **transillumination method** [Neu95].

Transillumination method

Let us examine the power radiated onto patch A_i:

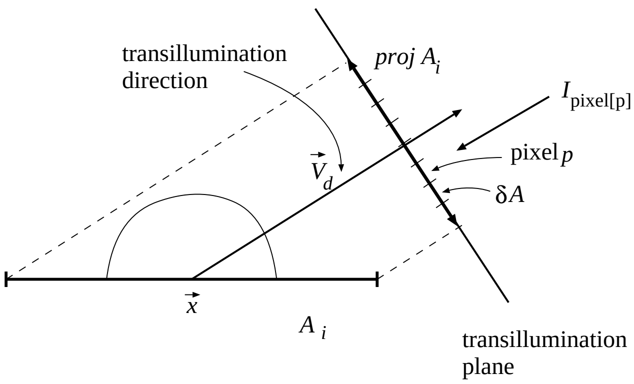
$$\Phi_i^{\rm in} = \int\limits_{\vec{V}\in H_i} \int\limits_{\vec{x}\in A_i} I^{\rm in}(\vec{x},\vec{V}) \cos\phi_{\vec{V}}\ dA_i\ d\omega_{\vec{V}} \qquad (1)$$

where H_i is the set of directions from where patch A_i can be lit (it is actually a hemisphere above the plane of patch A_i), $I^{\rm in}(\vec{x},\vec{V})$ is the intensity of the radiation at point $\vec{x}$ from direction $\vec{V}$, $\vec{x}$ and $\vec{V}$ are the running vectors on patch A_i and in H_i respectively (see figure **??**).



Having subdivided the directional hemisphere H_i of size 2π into M number of solid angles, each of them being of size $2\pi/M$, we get the following approximation sum of the previous integral:

$$\Phi_i^{\rm in} \approx \frac{2\pi}{M} \cdot \sum_{d=1}^{M} \int\limits_{\vec{x}\in A_i} I^{\rm in}(\vec{x},\vec{V}_d) \cos\phi_{\vec{V}_d}\ dA_i. \qquad (2)$$



Integral $\int\limits_{\vec{x}\in A_i} I^{\rm in}(\vec{x},\vec{V}_d) \cos\phi_{\vec{V}_d}\ dA_i$ expresses the power arriving at patch A_i from direction V_d called the **transillumination direction**. Let us place a plane perpendicular to the transillumination direction above patch A_i. This plane is called the **transillumination plane**. The same power reaching patch A_i can also be computed on this transillumination plane, just the range of integration should be changed from A_i to the projection of A_i on the transillumination plane, which is denoted by projA_i:

$$\int\limits_{\vec{x}\in A_i} I^{\rm in}(\vec{x},\vec{V}_d) \cos\phi_{\vec{V}_d}\ dA_i = \int\limits_{\vec{x}'\in {\rm proj}A_i} I^{\rm in}(\vec{x}',\vec{V}_d)\ dA_i^{\rm proj} \qquad (3)$$

since $\cos\phi_{\vec{V}_d}\ dA_i = dA_i^{\rm proj}$.

In order to numerically compute the integral on $\mathrm{proj}A_i$, the transillumination plane is discretized into finite squares of size δA, called "pixels". This discretization enables us to replace the integral of equation (2) by an approximation sum, resulting in the following formula:

$$\Phi_i^{\mathrm{in}} \approx \frac{2\pi}{M} \cdot \delta A \cdot \sum_{d=1}^{M} \sum_{p \in \mathrm{proj}A_i} I_{\mathrm{pixel}[p]}. \tag{4}$$

Let us turn our attention to the power reflected by patch A_i, denoted by Φ_i^{out}. If patch A_i is diffuse (which is a general restriction of radiosity methods), then the total reflected power is the total incoming power multiplied by the diffuse coefficient ϱ_i plus the self-emission $(E_i \cdot A_i)$, thus we get:

$$\Phi_i^{\mathrm{out}} = E_i \cdot A_i + \varrho_i \cdot \Phi_i^{\mathrm{in}} \approx E_i \cdot A_i + \varrho_i \cdot \frac{2\pi}{M} \cdot \delta A \cdot \sum_{d=1}^{M} \sum_{p \in \mathrm{proj}A_i} I_{\mathrm{pixel}[p]}. \tag{5}$$

By definition, the radiosity of patch i is the reflected power of a unit surface area, that is:

$$B_i = \frac{\Phi_i^{\mathrm{out}}}{A_i} \approx E_i + \varrho_i \cdot \frac{2\pi}{M} \cdot \frac{\delta A}{A_i} \cdot \sum_{d=1}^{M} \sum_{p \in \mathrm{proj}A_i} I_{\mathrm{pixel}[p]}. \tag{6}$$

The source of intensity $I_{\mathrm{pixel}[p]}$ is also a patch that is also assumed to be diffuse and to have radiosity $B_{\mathrm{pixel}[p]}$. As known, for diffuse radiance $B = I\pi$ [SKe95], thus equation (6) can also be written in the following form:

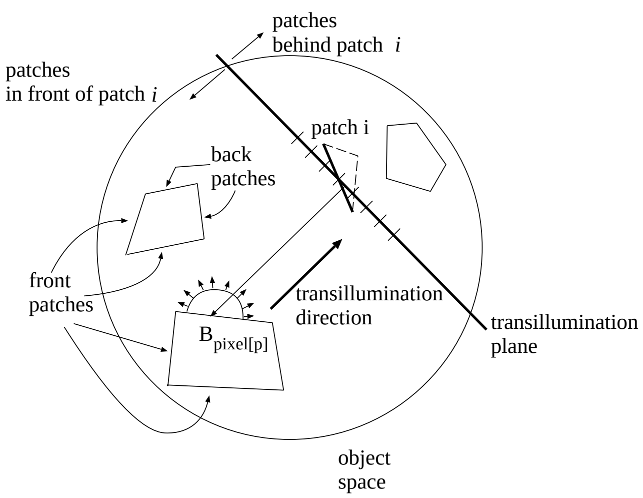
$$B_i \approx E_i + \varrho_i \cdot \frac{2}{M} \cdot \frac{\delta A}{A_i} \cdot \sum_{d=1}^{M} \sum_{p \in \mathrm{proj}A_i} B_{\mathrm{pixel}[p]}. \tag{7}$$

This is an alternative form of the radiosity equation. As the classical radiosity equation, the alternative form can also be used as a basis of an iterational solution:

$$B_i^{\mathrm{new}} \approx E_i + \varrho_i \cdot \frac{2}{M} \cdot \frac{\delta A}{A_i} \cdot \sum_{d=1}^{M} \sum_{p \in \mathrm{proj}A_i} B_{\mathrm{pixel}[p]}. \tag{8}$$

The effective evaluation of this formula is the key to find an algorithm of desired time complexity.

Let us fix the transillumination direction and evaluate the contribution of the scene of all patches to patch i from the given direction d.



According to the orientation relative to the transillumination direction, patches can be classified as *front facing* and *back facing*. Obviously, back facing patches cannot receive energy from a transillumination direction, thus only front facing patches must be examined.

If patch i is considered as front facing from the transillumination direction, then the contribution from direction d is

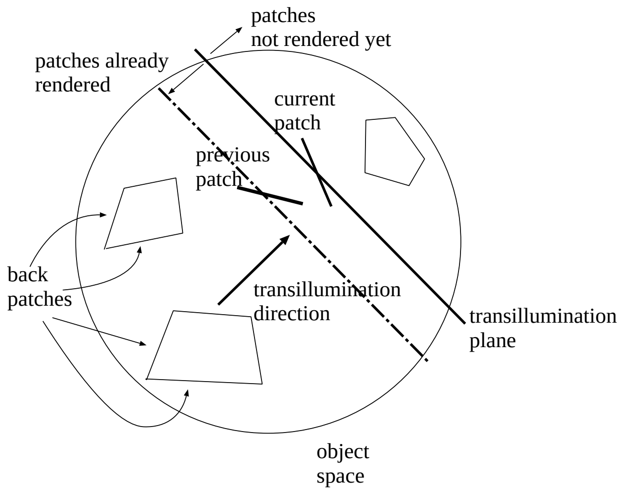
$$B_i^{\mathrm{new}}(d) \approx \varrho_i \cdot \frac{2 \cdot \delta A}{M} \cdot \sum_{p \in \mathrm{proj} A_i} B_{\mathrm{pixel}[p]}, \tag{9}$$

where $B_{\mathrm{pixel}[p]}$ is the radiosity of the patch visible in this pixel and facing towards patch i. If no patches are visible in a pixel, then $B_{\mathrm{pixel}[p]}$ is zero or is determined by external illumination (the effect of skylight illumination can be simulated in this way [Neu95]). Finally, if the patch visible in the pixel is not facing towards patch i (this pathological case cannot happen if the scene consists of solids), then $B_{\mathrm{pixel}[p]}$ is zero.

Summarising, for each patch, the visible patches must be determined through a discretized window. This is a visibility problem, and the result is an "image" of the patches, assuming the eye to be in the centre of patch i, the window to be on the transillumination plane and the colour of a patch j that is facing to patch i to be j and 0 otherwise. (Another alternative would be the direct storage of radiosities in the image buffer. It would allow linear radiosity approximations during iteration by applying simple Gouraud shading like techniques. The limited depth of the image buffer, however, might impose numerical problems.) Note that the patches that are facing to patch i are those back facing patches in the given transillumination direction, which are in front of patch i.

To compute the new radiosity of patch i, the contribution of the scene can be calculated by computing an "image" of the scene then producing the sum from the "pixels" of this image according to equation (9).

When creating an "image", the selected patch should be the eye (or more precisely, the eye should be at some point of this patch) and the viewing direction should be the current transillumination direction. Obviously, only those patches can contribute to the "image" that are in front of patch i in the transillumination direction.



Let us assume that the patches are sorted in the transillumination direction. Such sorting is not always trivial because patches may overlap in this direction and cyclic overlapping might also occur. However in the solution of the radiosity problem, the patches are usually small, thus simply sorting them by one of their arbitrary point introduces only a negligible error. Sorting a data set is known to have $O(n \log n)$ time complexity. However, if this error cannot be tolerated, then exact methods can also be used that subdivide the patches to allow precise sorting, including the painter's algorithm [NNS72], depth order algorithm presented in [dB92] and the application of Binary Space Partitioning (BSP) trees [PY90]. Painter's algorithm also runs in $O(n \log n)$ time in the average case, but its worst case complexity is $O(n^2)$ and may result in $O(n^2)$ subdivided patches. The depth sort algorithm presented in [dB92], on the other hand, solves the depth sorting problem in $O(n^{5/3})$ time even in the worst case if there is no cyclic overlapping. Another promising alternative is to pre-process the object space into a BSP tree. Such a BSP tree provides the sorting of the patches in a view-independent way, thus the same structure can be used for any transillumination direction. Since the construction of the BSP tree may also require the subdivision of patches, the number of resulting patches in the tree is between $\Omega(n)$ and $O(n^2)$.

Let us assume that the required sorting has been generated by any of the mentioned methods. Now, the most important question of this approach can be asked: what is the difference between two "images" generated from two subsequent patches in the order of the transillumination direction? The answer is quite promising: the later contains one patch more, namely the previous patch. This means that if the "images" needed to compute equation (9) are generated in the order of the patches in the transillumination direction, then the results generated for a patch can be reused for the next patch. Stepping from patch i to the subsequent patch j requires the rendering of patch i to complete the "image" and the evaluation of equation (9). Thus the extra computation needed for a single patch in a single transillumination direction is independent of the number of patches in the scene.

In order to consider all transillumination directions, $B_i^{\text{new}}(d)$s of different directions d must be summed.

This is summarised in the following algorithm for the iterative evaluation of equation (8):

1) **for** each patch i **do** $B_i = E_i$;
2) **loop**
3)   **for** each patch $B_i^{\text{new}} = E_i$
4)   **for** each direction d
5)   Sort patches in direction d
6)   **for** each patch i classify patch to *front* and *back*
7)   Clear image
8)   **for** each patch i in sorted order **do**
9)   **if** patch i is front facing **then**
10) $B_i^{\text{new}} += \varrho_i \cdot \frac{2}{M} \cdot \frac{\delta A}{A_i} \sum_{\text{pixel} \in A_i} B_{\text{image[pixel]}}$
11)   Render patch i with colour 0
12)   **else** Render patch i with colour i
13)   **endfor**
14)   **endfor**
15)   **for** each patch i **do** $B_i = B_i^{\text{new}}$
16) **endloop**

Complexity evaluation

Let us examine the time complexity of the above algorithm. Denoting the number of patches by n, execution of the first line requires $O(n)$ time. Line 5 can be realised in $O(n \log n)$ time, since it is inherently a sorting problem if the small error of depth overlapping can be tolerated. Line 6 examines the patches one by one, which requires linear time. A single cycle of **for** loop starting at line 8 contains only instructions that work with a single patch and an "image", thus the time required for a single cycle is independent of the number of patches. Since the **for** loop is executed n times, the time complexity of the **for** loop of line 8 is $O(n)$.

This means that a single cycle of the **for** loop starting at line 4 consists of an $O(n \log n)$ phase and an $O(n)$ phase, thus it is in $O(n \log n)$. Since the number of directions is constant and is independent of the number of patches, the complete **for** loop of line 4 is $O(n \log n)$. Line 15 is obviously linear in time.

Consequently the algorithm consists of an initialisation phase of $O(n)$ and an iteration phase starting at line 2, where a single step of the iteration requires $O(n \log n)$ time. In order to find the complexity measure of the complete algorithm, the number of iteration steps must be computed.

In the following section we prove that the number of steps that decreases the error below a predefined limit does not increase with the number of patches, that is, the number of iterational loops is $O(1)$.

Convergence

Assume that in iteration step k the error of radiosity of patch i is $\Delta B_i(k)$. Thus if the correct radiosity estimate were $\hat{B}_i$, then the available radiosity in step k would be:

$$B_i(k) = \hat{B}_i - \Delta B_i(k). \tag{10}$$

Substituting this into equation (8), the error in the next step can be computed:

$$\Delta B_i(k+1) = \varrho_i \cdot \frac{2}{M} \cdot \frac{\delta A}{A_i} \cdot \sum_{d=1}^{M} \sum_{p \in \mathrm{proj} A_i} \Delta B_{\mathrm{pixel}[p]}. \tag{11}$$

Let $\Delta B(k)$ be the maximum of absolute values of error in the radiosity estimates of all patches in step k. Using $\Delta B(k)$, the following upper bound can be derived for $\Delta B_i(k+1)$:

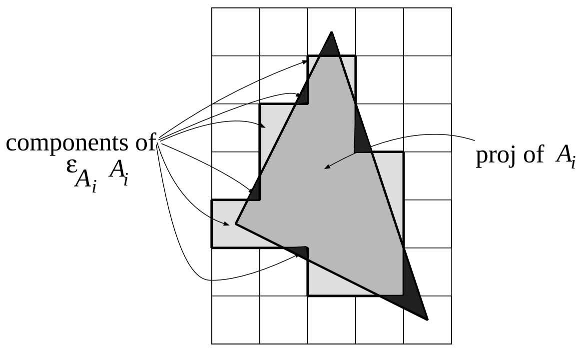
$$|\Delta B_i(k+1)| \leq \Delta B(k) \cdot \varrho_i \cdot \frac{2}{M} \cdot \frac{1}{A_i} \cdot \sum_{d=1}^{M} \sum_{p \in \mathrm{proj} A_i} \delta A. \tag{12}$$

Note that

$$\sum_{p \in \mathrm{proj} A_i} \delta A \approx A_i \cdot \cos \phi_i(d), \tag{13}$$

where $\phi_i(d)$ is the angle between the transillumination direction d and the normal of patch i. To replace this approximation by an exact formula, the relative error term ϵ_{A_i} of the discrete approximation of the projected area is introduced:

$$\sum_{p \in \mathrm{proj} A_i} \delta A = A_i \cdot (\cos \phi_i(d) + \epsilon_{A_i}). \tag{14}$$



This means that

$$\frac{2}{M} \cdot \frac{1}{A_i} \sum_{d=1}^{M} \sum_{p \in \mathrm{proj} A_i} \delta A = \frac{2}{M} \sum_{d=1}^{M} (\cos\phi_i(d) + \epsilon_{A_i}). \tag{15}$$

Note furthermore that

$$\frac{2}{M} \sum_{d=1}^{M} \cdot \cos\phi_i(d) = \frac{1}{\pi} \cdot \frac{2\pi}{M} \sum_{d=1}^{M} \cdot \cos\phi_i(d) \approx \frac{1}{\pi} \int_{2\pi} \cos\phi_i(d) d\omega_d = 1. \tag{16}$$

Using the previous trick, having introduced error term ϵ_{d_i} of the directional integration, the approximation can be replaced by the following equation:

$$\frac{2}{M} \sum_{d=1}^{M} \cdot \cos\phi_i(d) = \frac{1}{\pi} \cdot \int_{2\pi} \cos\phi_i(d) d\omega_d \cdot (1 + \epsilon_{d_i}) = 1 + \epsilon_{d_i}. \tag{17}$$

Using these equalities, inequality (12) can also be written in the following form:

$$|\Delta B_i(k+1)| \leq \Delta B(k) \cdot \varrho_i \cdot (1 + |\epsilon_{d_i}| + 2|\epsilon_{A_i}|). \tag{18}$$

Since $\Delta B(k+1) = \max |\Delta B_i(k+1)|$, we get:

$$\Delta B(k+1) \leq \Delta B(k) \cdot \max_i \varrho_i \cdot (1 + |\epsilon_{d_i}| + 2|\epsilon_{A_i}|) \tag{19}$$

which means that the maximal error of patch radiosities is limited by a geometric series of coefficient $\max_i \varrho_i \cdot (1 + \max_i |\epsilon_{d_i}| + 2\max_i |\epsilon_{A_i}|)$. For physically correct models $\max_i \varrho_i$ is always less than 1. In order to make the bounding series convergent, the error term should satisfy:

$$\max_i |\epsilon_{d_i}| + 2\max_i |\epsilon_{A_i}| \leq \frac{1}{\max_i \varrho_i} - 1. \tag{20}$$

This imposes a constraint on the accuracy of discrete approximation of the directional and surface integrations. Not suprisingly, the diffuse interreflections are smaller, the constraints are less severe. If this error constraint is satisfied, the coefficient of the geometric series is less than 1, providing convergence for any arrangements. Note furthermore that the coefficient is independent of the number of patches and their geometrical arrangements, thus the number of iterational steps required to generate estimates with some predefined precision is constant.

The time complexity of the algorithm is therefore proportional to the time needed for the execution of a single loop, resulting in $O(n \log n)$ time complexity.

The space complexity of the algorithm is also quite appealing. To store the B_i and B_i^{new} radiosity values, we need $O(n)$ space (more precisely $2 \cdot n \cdot \mathrm{size\_of}(\mathit{double})$ space). The extra memory used by sorting is also in $O(n)$ (it is $n \cdot \mathrm{size\_of}(\mathit{pointer})$). A single image buffer of constant size is needed to solve the discrete visibility problem in different transillumination directions. The size of this buffer is independent of the number of patches and is determined by the size of the object space and the above accuracy requirements. Summarising, the space complexity is $O(n)$.

Alternatives for the discussed sorting step

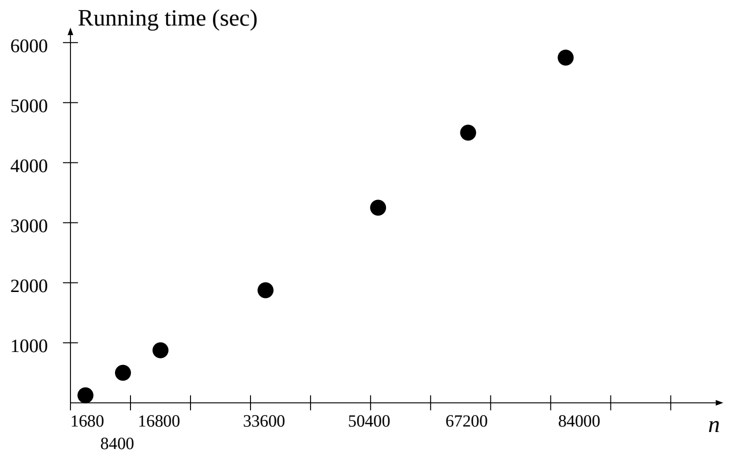
In the discussed method, sorting in line 4 is responsible for the $O(n \log n)$ time complexity. However, this can be moved to a pre-processing phase, which would generate the sorted list of patches in all directions. During iteration these lists can be used. After pre-processing characterised by $O(n \log n)$ time complexity, the iterational solution of the radiosity problem would need only $O(n)$ time. The price of a linear time algorithm is quite high in terms of memory required. In the naive implementation an ordered list of size n must be stored for each transillumination direction requiring $n \cdot M \cdot \text{size\_of}(pointer)$ additional memory. It is still in $O(n)$, but n is multiplied by a large number (M).

If the error introduced by the depth overlapping cannot be tolerated, then exact methods must be used such as the mentioned painter's algorithm and depth sorting technique. The application of painter's algorithm would make this approach have $O(n^2)$ worst-case but still $O(n \log n)$ average case complexity. The use of depth sorting technique, on the other hand, would result in an algorithm of $O(n^{5/3})$ time complexity for object scenes having no cyclic overlapping.

The application of BSP trees would necessitate a pre-processing step of $O(n \log n)$ average case and $O(n^2)$ worst-case complexity. Then, during the iteration, the sorted order could be available in linear time of the number of resulting patches, which is $O(n)$ in average and $O(n^2)$ in the worst case.

Simulation results

That variation of the presented method, which applies the exact painter's algorithm, has been implemented and tested with object scenes consisting of large number of randomly placed spheres. The running time measurements are summarised in figure **??**, which indicates the sub-quadratic average case complexity.



Conclusions and further research directions

This paper presents an iterative radiosity algorithm that can achieve sub-quadratic time complexity while using only $O(n)$ storage. Depending on the applied strategy in sorting, these complexity measures may be valid only in the average case while the worst-complexity is in $O(n^2)$. Still, the variations of this approach are better than the known, deterministic and non-hierarchical radiosity algorithms that use quadratic time both in the average and in the worst case.

This method, similarly to other methods, applies discretization, for which the required accuracy has been determined. The computed bound, however, can be very high if the object space contains highly reflective objects. The question arises whether or not these limits can be eased by jittering the transillumination plane and the selected directions.

A related question comes from the fact that the presented iterational formula (equation 8) is only an approximation of the original equation, thus its solution might differ from the solution of the original integral equation. Again, instead of uniform sampling, Monte-Carlo or Quasi-Monte-Carlo quadrature can be used to approximate the real solution of the original equation.

Both problems forces us to consider the combination of the presented method with some randomisation.

References

[CG85] Michael Cohen and Donald Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Proceedings of SIGGRAPH '85*, pages 31–40, 1985.

[dB92] Mark de Berg. *Efficient Algorithms for Ray Shooting and Hidden Surface Removal.* PhD thesis, Rijksuniversiteit te Utrecht, The Nederlands, 1992.

[FMSK96] Tibor Fóris, Gábor Márton, and László Szirmay-Kalos. Ray-shooting in logarithmic time. In *Winter School of Computer Graphics '96*, pages 84–90, Plzen, Czech Republic, 12–16 February 1996. University of West Bohemia.

[Ge89] A.S. Glassner (editor). *An Introduction to Ray Tracing.* Academic Press, London, 1989.

[HSA91] P. Hanrahan, D. Salzman, and L. Aupperle. Rapid hierachical radiosity algorithm. *Computer Graphics (SIGGRAPH proceedings '91)*, 1991.

[Kel95] A. Keller. A quasi-monte carlo algorithm for the global illumination in the radiosity setting. In H. Niederreiter and P. Shiue, editors, *Monte-Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 239–251. Springer, 1995.

[Már95] Gábor Márton. Acceleration of ray tracing via voronoi-diagrams. In Alan W. Paeth, editor, *Graphics Gems V*, pages 268–284. Academic Press, Boston, 1995.

[MSK95] Gábor Márton and László Szirmay-Kalos. On average-case complexity of ray tracing algorithms. In *Winter School of Computer Graphics '95*, pages 187–196, Plzen, Czech Republic, 14–18 February 1995. University of West Bohemia.

[Neu95] L. Neumann. Monte carlo radiosity. *Computing*, 55:23–42, 1995.

[NJ96] K. Nechvile and Sochor J. Form factor evaluation with regional bsp trees. In *Winter School of Computer Graphics '96*, pages 285–293, Plzen, Czech Republic, 12–16 February 1996. University of West Bohemia.

[NNS72] M.E. Newell, R.G. Newell, and T.L. Sancha. A new approach to the shaded picture problem. In *Proceedings of the ACM National Conference*, page 443, 1972.

[PY90] M. S. Paterson and F. F. Yao. Efficient binary partitions for hidden-surface removal and solid modeling. *Discrete and Computational Geometry*, 5:485–503, 1990.

[SKe95] L. Szirmay-Kalos (editor). *Theory of Three Dimensional Computer Graphics.* Akadémia Kiadó, Budapest, 1995.

[SKM95] László Szirmay-Kalos and Gábor Márton. On convergence and complexity of radiosity algorithms. In *Winter School of Computer Graphics '95*, pages 313–322, Plzen, Czech Republic, 14–18 February 1995. University of West Bohemia.