Visual Comput (2005) 21: 569–578
DOI 10.1007/s00371-005-0329-8

ORIGINAL ARTICLE

Zhong Ren
Wei Hua
Lu Chen
Hujun Bao

Intersection fields for interactive global illumination

Published online: 31 August 2005

Z. Ren (✉) · W. Hua · L. Chen · H. Bao
State Key Lab of CAD& CG, Zhejiang University, Hangzhou, China, 310027
{renzhong, huawei, chenlu, bao}@cad.zju.edu.cn

Abstract We present a novel visibility representation, Intersection Field (*i*-Field), to compute global illumination in interactive rates. The *i*-Field provides fast visibility and line-scene intersection queries. We factorize the direct illumination into local irradiance and visibility ratio. The latter is efficiently evaluated by querying the *i*-Field. The indirect illumination is simulated by photon tracing, which is also accelerated by the *i*-Field. By quickly detecting invalid portions, our approach can handle highly dynamic scenes, allowing light sources and scene geometries to be manipulated at interactive rates through rigid transformations and free deformations.

Keywords Global illumination · Shading · Hardware · Photon tracing · Global line

1 Introduction

Global illumination computation was regarded as an offline process because of the intensive computation involved. Recent researches have made interactive global illumination possible by introducing caching schemes to reuse samples, efficiently updating partially precomputed radiosity solutions, or solving the rendering equation in progressive ways, see [11] for a comprehensive survey. However, it is still a challenge to compute global illumination at interactive rates for highly dynamic scenes.

Inspired by the idea of global lines [9], we propose an interactive global illumination approach based on a new visibility representation, Intersection Field (*i*-Field), which can be intuitively regarded as an organized set of line records, each of which keeps the intersection points between a global line and the scene (including geometry objects and light sources) in order. It provides efficient visibility and line-scene intersection queries with computational complexity insensitive to the number of faces.

In our approach, the global illumination model is reformulated to make a part of computation amenable to GPU. The direct illumination is factorized into local irradiance and visibility ratio, while the indirect illumination is simulated by a photon tracing method. Since the *i*-Field implicitly encodes all light transportation paths, it is used to accelerate the global illumination computation by avoiding line-scene intersection operations. By introducing a flexible intermediate data structure and a dynamic linking scheme, our approach enables users to interactively manipulate objects in highly dynamic scenes with global illumination solution.

2 Previous work

Offline global illumination. Many methods are developed from the hierarchical radiosity algorithm [10, 35] or the photon mapping algorithm [24]. Nimeroff et al. [23] use time sequences of range images to store global illumination. Myszkowski et al. [26] and Yee et al. [21] propose perception-guided global illumination methods to improve either the quality or the efficiency. More recently, Stokes et al. [52] propose a new perceptual metric,

which is based on rendering-by-component framework, to predict the perceptual importance. Purcell et al. [49] present a GPU based photon mapping algorithm that compute global illumination for moderate scenes in a few seconds.

Interactive global illumination. Recently, some researches introduce various caching schemes to reuse calculated results. Some manage to solve the rendering equation in a progressive way. The radiosity and approximate radiosity methods [5, 33, 42, 54] precompute view-independent radiosity solutions that can be viewed interactively. Extensions for view-dependent illumination [4, 12, 29, 46, 47] still cannot provide interactive rates. The approaches based on hierarchical radiosity [14, 40] provide much better performance by introducing mechanisms for controlling frame rates and efficiently detecting modified scene parts. Drettakis and Sillion [17] use the line-space hierarchy to identify links affected by changes in the scene and redistribute the lighting energy for the current scene configuration. Based on this work, Granier and Drettakis [55] propose a unified algorithm that uses hierarchical radiosity by clustering and incremental particle tracing with a line-space hierarchy. As a means of acceleration, caching schemes are adopted by many approaches, such as post-rendering warping [53], corrective texturing [34], render cache [6, 7], Holodeck [20], Tapestry [36], shading cache [37], radiance interpolants [27, 45], irradiance atlas [38] and virtual light source [44]. The selective photon tracing algorithm [28] computes indirect lighting asynchronously using a quasi-random photon tracing similar to [1, 2], and density estimation similar to [51]. Larson and Christensen [3] improve this method to simulate photon mapping for real-time applications. Mantiuk et al. [41] develop an algorithm for plausible approximation of global illumination method in real-time using cube-map and modern graphics hardware, based on irradiance volumes [16].

In the context of parallel ray tracing, Parker et al. [48], Benthin, Wald, Slusallek and Günther et al. [8, 22, 25] use low-level optimizations and parallelism and achieve interactive ray tracing of complex scenes, albeit with simple shading models.

Global line based visibility estimation. The idea of using global lines to accelerate visibility estimation was first proposed by Bucklew [9]. In the global ray-bundle tracing method proposed by Kalos et al. [30–32], global lines are reused to solve the visibility queries. In Multi-Frame Lighting method [18, 19], the authors use global line list to simulate light transport and accelerate non-interactive global illumination computation in radiosity environments. In [13, 43], the authors propose a radiosity multipath algorithm and develop a hierarchical data structure for acceleration. In his Ph.D. thesis [15], Durand gives a detailed review about global visibility.

3 Computational Model

Under the assumption of Lambertian diffuse reflectors, the irradiance at a given point $\boldsymbol{p}$ can be determined:

$$\mathcal{E}(\boldsymbol{p}) = \sum_k \mathcal{F}(\boldsymbol{p}, S_k) + \mathcal{G}(\boldsymbol{p}) \tag{1}$$

$\mathcal{G}(\boldsymbol{p})$ is the indirect irradiance, which can be obtained by a photon tracing method. $\mathcal{F}(\boldsymbol{p}, S)$, the direct irradiance from light source S, can be approximately factorized into two terms under the premise that the distance between the point and the light source is much greater than the size of the light source:

$$\mathcal{F}(\boldsymbol{p}, S) = \int_S h(\boldsymbol{p}, dS) \cdot vis(\boldsymbol{p}, \boldsymbol{q}) dS \tag{2}$$

$$\simeq \int_S h(\boldsymbol{p}, dS) dS \cdot \int_S \frac{vis(\boldsymbol{p}, \boldsymbol{q})}{A_S} dS = \mathcal{H}(\boldsymbol{p}, S) \cdot \mathcal{V}(\boldsymbol{p}, S)$$

where h is the radiance from dS without accounting for visibility. $vis(\boldsymbol{p}, \boldsymbol{q})$ is the mutual visibility indicator of $\boldsymbol{p}$ and $\boldsymbol{q}$ (center of dS), which is 1 if they are mutually visible and 0 otherwise. $\mathcal{V}(\boldsymbol{p}, S)$, namely the visibility ratio, is the ratio of the area of the visible part of S to the area of S, A_S, when observed from $\boldsymbol{p}$.

The computation of $\mathcal{H}$ is a local operation that only involves a light source and a point, while the $\mathcal{V}$ computation is global. This factorization separates the local and global operations, and makes the $\mathcal{H}$ computation amenable to GPU. As for the $\mathcal{V}$ computation, the *i*-Field, which will be discussed in Sect. 4, can be queried. Another benefit of the factorization is that it reduces the efforts for updating global illumination solution during user interactions. We will discuss this in Sect. 6.

The criterion of the factorization can be met by dynamic subdivision of light sources. By controlling the granularity of light sources, trade-off can be made between accuracy and efficiency.

We assume each object is represented by one or more parameterized triangle meshes. The parameterization is represented by a bijective mapping Γ , which maps a 2D parameter vector $\boldsymbol{u}$ to a 3D point $\boldsymbol{p}$ on a mesh. For each mesh M and each light source S, we use two 2D textures to store the *Local Irradiance Map* (LIM) $\mathcal{H}_{M,S}(\boldsymbol{u}) = \mathcal{H}(\Gamma(\boldsymbol{u}), S)$ and the *Visibility Ratio Map* (VRM) $\mathcal{V}_{M,S}(\boldsymbol{u}) = \mathcal{V}(\Gamma(\boldsymbol{u}), S)$. And for each M, we use another texture to store the *Indirect Irradiance Map* (IIM) $\mathcal{G}_M(\boldsymbol{u}) = \mathcal{G}(\Gamma(\boldsymbol{u}))$. In a diffuse environment, the light irradiance at a point $\boldsymbol{p}$ on mesh M can be evaluated efficiently in GPU via multiple texture mapping, i.e.

$$\mathcal{E}(\boldsymbol{p}) = \left(\sum_k \mathcal{H}_{M,S_k}(\boldsymbol{u}) \cdot \mathcal{V}_{M,S_k}(\boldsymbol{u})\right) + \mathcal{G}_M(\boldsymbol{u}) \tag{3}$$

where $\boldsymbol{u} = \Gamma^{-1}(\boldsymbol{p})$.

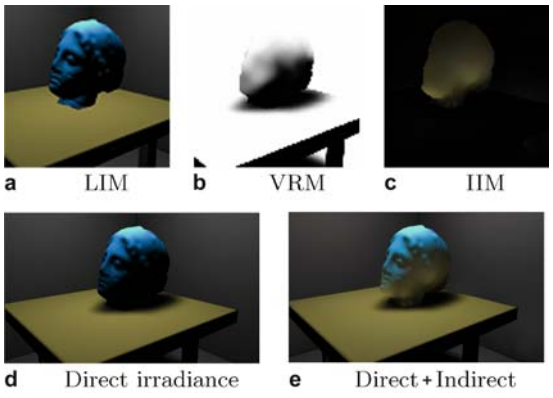


Fig. 1. Generation and synthesis of LIM, VRM and IIM

Thus, the computation of global illumination has been transformed into the generation and synthesis of the 3 kinds of maps (Fig. 1), which will be discussed in Sect. 5.

4 Intersection field

In this section, we will introduce the generation, organization and compression of the *i*-Field.

4.1 Line space parameterization

Line parameterization. A 2D line described by $y = ax + b$ can be represented by (a, b)[1]. For a 3D line, similar to [50], we find its horizontal component(a, b) by projection onto the base plane ($z = 0$), and find the vertical component(c, d) by parameterizing its projection on the plane perpendicular to base plane, determined by (a, b). Thus, any 3D line can be denoted by $l(a, b, c, d)$, or l for short. The perpendicular plane defined by (a, b) is denoted by $slice(a, b)$. The 4D parameter space is denoted by L^4. For a scene, its discrete line space is the simple discretized form of L^4 consisting of all lines passing through the scene's bounding box. Each $l(a_i, b_j, c_k, d_m)$ in the discrete line space is called a line sample. In the following, we also use L^4 to denote the discrete line space. By this parameterization we exclude all the lines that are perpendicular to the base plane. However, in L^4, we have lines that are close to these lines.

4.2 Generating intersection records

An intersection record is defined as a triple $(\boldsymbol{p}, l, T)$, indicating that line l intersects triangle T at point $\boldsymbol{p}$. To generate the *i*-Field, we must obtain the records of intersection points between all line samples and all triangles in the scene under a given discretization resolution N. We develop a fast scan-conversion algorithm to process in batches and construct the *i*-Field.

Given a triangle T, traditional 3D scan line algorithm can be used to calculate the endpoints of the intersection line segment between each $slice(a_i, b_j)$ and the triangle edges, if the intersection exists. The similar scan line algorithm can again be used in the vertical direction to calculate the intersection points between the intersection line segment and the scan lines on $slice(a_i, b_j)$. By interpolating 3D coordinates in the scan-conversion, intersection points in 3D space together with the line parameter (c_k, d_m) can be generated in batches, as in Fig. 2. The pseudo codes are listed in Fig. 3.

Given l, the intersection point $\boldsymbol{p}(x_p, y_p, z_p)$ can be represented by a intersection point parameter t, for which we simply use $\boldsymbol{p}$'s projection on the principal axis of l. Since L^4 is discretized regularly, (a_i, b_j, c_k, d_m) can be easily evaluated by index (i, j, k, m). Thus, all line records can be stored in a compact form as (t, T), omitting explicitly line parameters.

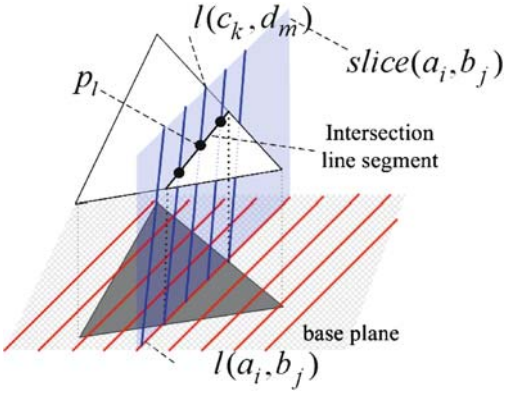


Fig. 2. Generating the intersection records of a triangle

```
For each face T in the input object
  Project T onto the base plane;
  For each a_i(i = 1, ..., N)
    Scan-convert the edges of T's projection;
    For each slice(a_i, b_j) that intersects the edges
      Calculate the intersection line segment s
        between slice(a_i, b_j) and T;
      For each c_k(k = 1, ..., N)
        Scan-convert s on slice(a_i, b_j);
        For each (c_k, d_m) that intersects s
          Output (p_l, (a_i, b_j, c_k, d_m), T), p_l = (c_k, d_m) ∩ s;
```

Fig. 3. Pseudo code for generating intersection records

[1] To make the parameterization bounded we classify 2D lines into two subspaces: $I = \{l(a, b) | l : y = ax + b, |a| \leq 1\}$ $II = \{l(a, b) | l : x = ay + b, |a| < 1\}$. For representation simplicity, we use (a, b) to denote a line, ignoring which subspace it belongs to.

4.3 Organizing intersection field

To make efficient updating possible, we introduce a simple hierarchy, in which each object has a local compressed *i*-Field. In the runtime, a global line record is obtained by dynamically linking all relevant line records in the local *i*-Fields. This allows updates to be made only for the local *i*-Fields when scene changes take place.

Local i-*Field.* To build the local *i*-Field for an object, the related intersection records are packed into line records. each of which corresponds to a line in L^4 and holds the intersection records on it, sorted in ascending order of t. Line records $\{l(a_i, b_j, c_k, d_m)|m = 0, \ldots, N\}$ are placed into a page record, which contains an array of lines in the same parameters a, b and c. Empty line records are bypassed and each line record carries an index of d. All page records are stored in a pool, indexed by a 3 dimensional table (Fig. 4). To look up a line record, the corresponding page record is firstly obtained in constant time, a linear search is then conducted to in the page record to retrieve the target line record. The computational complexity for accessing a line record is $O(N)$ in theory. However, line records are often accessed consecutively. This coherence can be exploited for acceleration. The current line record is cached as the starting point for the next search. This reduces the average access time to $(N/N_{access} + 1) \cdot t_0$, where t_0 is the retrieval cost for a record and N_{access} is the average access number per page record.

Global i-*Field.* The global *i*-Field is a logical set of the local *i*-Fields of all scene objects. A global line record is built on demand. Since all the local *i*-Fields are discretized in the same line space, a given global line index can be used to collect line records from the local *i*-Fields. The collected records are then sorted in ascending order of t and a global line record is constructed. In the rest of the paper, we refer to the global *i*-Field as *i*-Field for convenience.

Compression. Exploiting the line space coherence, we develop an enhanced lossless Run Length Encoding (RLE) to compress the line records in each page record. For the lines $\{l(a_{i0}, b_{j0}, c_{k0}, d_m)|m = \mu, \ldots, \mu + \Delta\mu, 0 < \mu, \mu + \Delta\mu < N\}$ intersecting a triangle T, the corresponding intersection point parameters t form an arithmetical progression with increment $\delta(\mu, T)$. Thus, these line records can be compressed into a compact form RLE_LR:

```
RLE_LR :=
    Δμ:  integer        // run length
    μ:   integer        // index of d
    num: integer        //num. of intersection records
    IR[1, ..., num] :   RLE_IntersectionRecord

RLE_IntersectionRecord :=
    i_T: integer        //index of the intersected triangle
    t:   float          //intersection point parameter
    δ:   float          //δ(μ, T)
```

5 Illumination computation

As discussed in Sect. 3, the illumination computation is actually the computation of LIM, VIM and IIM. Using the OpenGL Render-to-Texture extension, these textures can be generated by the graphics hardware without time-consuming buffer readback. They are then synthesized in GPU to generate the final image.

Computing Local Irradiance Map (LIM). For each mesh, a LIM is generated for each light source. By passing the information of light sources into a pixel shader, LIMs can be computed in GPU. The light source S is subdivided into some small elements $\{\Delta S_i\}$, and the LIMs can be computed by:

$$\mathcal{H}_{M,S}(\boldsymbol{u}) \simeq \sum_i h(\Gamma(\boldsymbol{u}), \boldsymbol{q}_i)\Delta S_i \tag{4}$$

where $\boldsymbol{q}_i$ is the center of ΔS_i.

Each triangle in M is rendered, taking the positions of vertices as 3D texture coordinates. The area, intensity of $\{\Delta S_i\}$ and $\{q_i\}$ are packed as an array and passed into a pixel shader, where the irradiance is computed by Eq. 4.

Computing Visibility Ratio Map (VRM). Instead of evaluating visibility ratios on a point-by-point basis, we use visibility sample splatting in the texture space to batchly estimate the visibility ratio of each point on mesh M (Fig. 5). For a point $\boldsymbol{p} = \Gamma(\boldsymbol{u})$ on mesh M, its neighborhood on

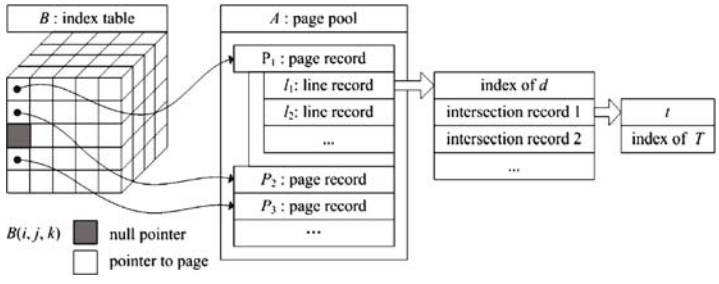


Fig. 4. The structure of a local *i*-Field without compression

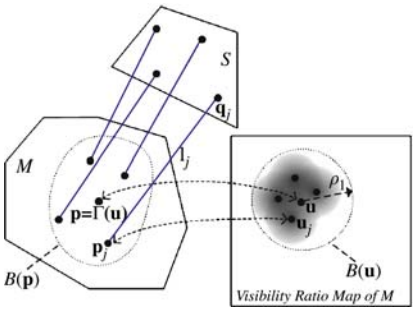


Fig. 5. Computing VRMs

M, $B(\boldsymbol{p})$, is defined by $\Gamma(B(\boldsymbol{u}, \rho_1))$, where $B(\boldsymbol{u}, \rho_1)$ is the disc neighborhood of $\boldsymbol{u}$ in the parameter space with radius $\rho_1 = \sqrt{|\boldsymbol{J}(\boldsymbol{u})|^{-1}} \cdot \rho_0$. $\boldsymbol{J}(\boldsymbol{u})$ is the Jacobian at $\boldsymbol{u}$ and ρ_0 is a constant. The intersection points pairs $\{(\boldsymbol{p}_j, \boldsymbol{q}_j) | \boldsymbol{p}_j = l_j \cap B(\boldsymbol{p}), \boldsymbol{q}_j = l_j \cap S, l_j \in L^4\}$ are used to estimate the visibility ratio:

$$\mathcal{V}_{M,S}(\boldsymbol{u}),\ S \simeq \frac{\sum_j vis(\boldsymbol{p}_j, \boldsymbol{q}_j) \cdot w(\boldsymbol{J}(\boldsymbol{u}_j) \cdot (\boldsymbol{u} - \boldsymbol{u}_j))}{\sum_j w(\boldsymbol{J}(\boldsymbol{u}_j) \cdot (\boldsymbol{u} - \boldsymbol{u}_j))} \tag{5}$$

where $\boldsymbol{p}_j = \Gamma(\boldsymbol{u}_j)$, and

$$w(\boldsymbol{x}) = \begin{cases} exp\Big(-\frac{3||\boldsymbol{x}||^2}{\rho_0^2}\Big) & ||\boldsymbol{x}|| \le \rho_0, \\ 0 & ||\boldsymbol{x}|| > \rho_0 \end{cases},$$

$$\boldsymbol{J}(\boldsymbol{u}_j) = \frac{\partial \Gamma}{\partial u}(\boldsymbol{u}_j) \tag{6}$$

Here we use the intersection points $\{\boldsymbol{q}_j\}$ on S as light source samples and the distance $||\boldsymbol{p} - \boldsymbol{p}_j|| \simeq ||\boldsymbol{J}(\boldsymbol{u}_j) \cdot (\boldsymbol{u} - \boldsymbol{u}_j)||$ to compute the weight of $vis(\boldsymbol{p}_j, \boldsymbol{q}_j)$. We use graphics hardware to generate the map, similar to the Surfel algorithm [39]. The pseudo codes in Fig. 6 output a texture, in which the red and green channels store the numerators and denominators in Eq. 5, respectively. It is an alternative representation of the VRM, for the division in Eq. 5 can be handled in the synthesis stage.

```
Gather the intersection points pairs
    {p_j = M ∩ l_j, q_j = S ∩ l_j};
Enable blending mode as src + dest;
Set texture Tex as rendering target;
For each (p_j, q_j)
  u_j = Γ^{-1}(p_j);
  c = vis(p_j, q_j);
  Set texture matrix to be J^{-1}(u_j);
  Draw a square with the Gaussian texture in flat shading
    mode with face color = (c, 1.0, 0.0), the center = u_j,
    the edge size = sqrt(|J(u)|^{-1}) · ρ_0;
```

Fig. 6. Pseudo codes for computing VRM

Computing Indirect Irradiance Map (IIM). We use photon tracing to simulate indirect irradiance. Photons are emitted from each light source in directions that are randomly selected from a discrete direction set according to a cosine distribution [24]. The discrete direction set consists of the directions of all line samples in L^4 (each line has 2 directions). The *i*-Field is used to efficiently obtain the hit points of the photons. Each photon emission direction $\boldsymbol{v}$ and point $\boldsymbol{q}$ forms a ray. The closest line sample l, is chosen as the supporting line for the photon transportation. From the intersection records in the line record, the nearest intersection point $\boldsymbol{q}'(= l \cap S)$ to $\boldsymbol{q}$ is identified. Intersection point $\boldsymbol{p}_1(= l \cap M)$ next to $\boldsymbol{q}'$ in direction $\boldsymbol{v}$ is taken to approximate the hit point at which the photon arrives (Fig. 7). A Russian roulette is used to determine whether the photon should be reflected or be absorbed according to the object's diffuse coefficient. The same operation is iteratively executed until the photon is finally absorbed or targeted to the free space.

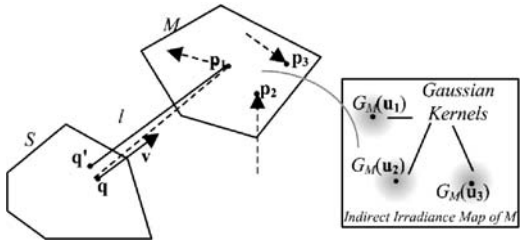


Fig. 7. Computing IIMs

After the paths of all photons are traced out, the IIM is used to collect the power of incoming photon for each mesh. The indirect irradiance at $\boldsymbol{u}_i$, namely $\mathcal{G}_M(\boldsymbol{u}_i)$, equals to the sum of all incoming photon power. We use Gaussian kernel to reconstruct the map from indirect irradiance samples $\mathcal{G}_M(\boldsymbol{u}_i)$.

6 Updates for user interactions

The key for efficient updating of global illumination solution is to make full use of the coherences and to make the update as local as possible. By decoupling the updates of LIM, VRM and IIM, temporal coherence is more easily exploited. Additionally, the *i*-Field can be queried to efficiently locate the most influenced regions with negligible overhead. This makes local updating feasible.

In the computation of VRM and IIM, we maintain a structure for each object to cache the hit points, in which the light source identifier, the hit point position, the irradiance(or visibility ratio for the VRM case) and the corresponding global line index are stored.

Scene changes are classified into six categories, for which we list what should be updated in Table 1. An all-

Table 1. User interactions and corresponding update to be conducted

	Scene change categories	LIM		VRM		IIM	*i*-Field
		mesh	light	mesh	light	mesh	obj
1	Lights property changes of *S*	visible to *S*	*S*	–	–	all	–
2	Material property changes of *M*	*M*	visible to *M*	–	–	all	–
3	Rigid transformations of *S*	visible to *S*	*S*	all	*S*	all	–
4	Free deformations of *S*	visible to *S*	*S*	all	*S*	all	*S*
5	Rigid transformations of *M*	*M*	visible to *M*	*M* and meshes in *M*'s shadow	all	all	–
6	Free deformations of *M*	*M*	visible to *M*	*M* and meshes in *M*'s shadow	all	all	*M*

Table 2. Time and space efficiency

No.	meshes	lights	triangles	*i*-Field		LIM		VRM		IIM		Synthesis	Overall
				size	num.	size	time	size	time	size	time	time	time
1	3	1	37767	14.1[2]	3	4.9	15.3	9.6	61.5	2.4	30.9	4.3	112
2	30	2	57545	72.6	60	6.7	76.4	19.4	309.5	4.3	141.3	7.8	535
3	29	3	87001	113.0	87	14.6	87.3	25.7	335.0	7.0	194.5	9.2	626

zero-flag is maintained for each VRMs. By this flag, the meshes visible to a light source or the light sources visible to a mesh can be easily found.

If a light source *S* undergoes changes, all IIMs need to be updated, but the hit points cached in each object can be reused unless they are generated from *S*. If a scene geometry object undergoes changes, the LIMs of the unchanged objects are still valid, while the VRMs of all the unchanged objects that are fully outside of the shadow volume cast by the changing object's bounding motion volume are also valid. The influenced lines can be found by scan-conversion of the bounding motion volume under the constraint of the light source's local *i*-Field. For the updating of VRMs, only the hit points that are relevant to the influenced lines need to be updated.

If free deformation takes place on an object(either a light source or a mesh), its local *i*-Field needs to be regenerated. Contrastively, for rigid transformations, the corresponding local *i*-Field is only labeled as "transformed" instead of thoroughly updated.

In the dynamic linking stage, the global line is transformed into the local frame of the "transformed" local *i*-Field to find a match. If a perfect match can't be found, the nearest line record is retrieved, with its intersection records transformed to the global frame.

The transformation overhead is minor, since only the lines involved in the illumination computation need to be transformed. However, errors might be introduced by this approach. The artifacts of the shadow of the statue in Fig. 12 are due to the transformation of the statue. It can be proven that the error is bounded by a value that is in direct ratio to the bounding box size of the transformed object and the discretization precision. We use another thread to regenerate the local *i*-Field in idle time and refine the global illumination solution when user interaction stops.

[2] Timings are given in millisecond and storage sizes in megabyte. The number of VRMs equals to that of LIMs, and the number of IIMs equals to that of meshes.

7 Results and discussions

We implement our algorithm with OpenGL 1.5 and NVIDIA Cg and carry out the following experiments on a PC with Intel 2.4 GHz CPU, 3GB main memory, and a NV6800 graphics card with 256MB texture memory.

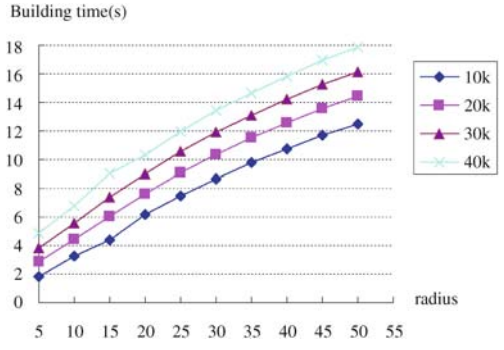


Fig. 8. Building time for the *i*-Field. The abscissa is the radius of the bounding sphere of the model. The models consist of 10k, 20k, 30k and 40k triangles, respectively

Table 3. The average frame rates for user interaction

Scene	Object	Interaction	Frame rates(fps)
1	light	rigid transformation	7.01∼10.85
	source	free deformation	6.21∼9.63
	mesh	rigid transformation	8.31∼24.82
2	light	rigid transformation	1.67∼4.53
	source	free deformation	1.46∼2.25
	mesh	rigid transformation	3.61∼8.12
3	light	rigid transformation	3.00∼5.12
	source	free deformation	1.23∼2.15
	mesh	rigid transformation	3.46∼7.94
		free deformation	1.95∼2.41

Quality. The resolution of L^4 influences the quality significantly, as shown in Fig. 9. Both the VRMs and photon paths can be determined with higher precision when resolution increases. In computating the VRMs, the Gaussian kernel radius ρ_0 is determined in a direct proportion with the discretization precision of b- and d-dimension in L^4. In this way, the resolution determines the kernel size and the number of visibility samples involved in the visibility ratio evaluation. Lower resolution results in blurrier shadow boundary (Fig. 9). In L^4, parameters a, c are relevant to the line direction and b, d are relevant to the distance. We choose different discretization resolutions for a, c and b, d, referred to as $N_{direction}$ and $N_{distance}$ respectively. $N_{direction}$ is specified as a constant. By experiments, we find 76(corresponds to 5,776 discrete directions) is a good choice. $N_{distance}$ should be chosen in direct proportion to the ratio of the size of the scene's bounding box to the size of desirable shading details. Moreover, if $N_{direction}$ and $N_{distance}$ have common divisors larger than 1, the line samples in L^4 would be focused periodically in 3D space and sometimes results in flicker in the synthesized images, as a comparison the result in Fig. 9f looks better than that in Fig. 9e. In all the tables and results presented in this section, the L^4 resolution is $76 \times 127 \times 76 \times 127$, if not otherwise stated.

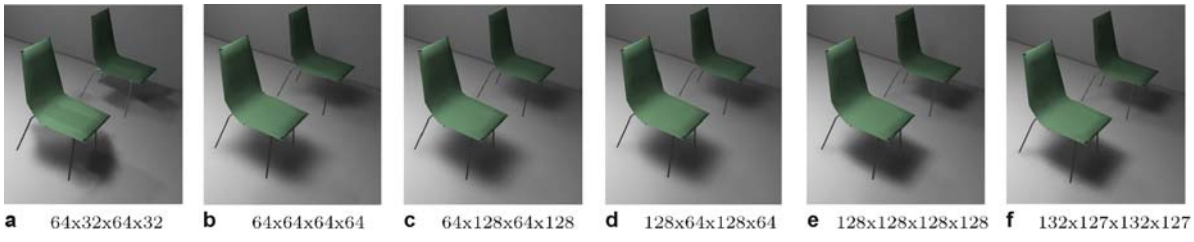


Fig. 9. The global illumination solutions are computed based on the *i*-Fields with the different resolutions

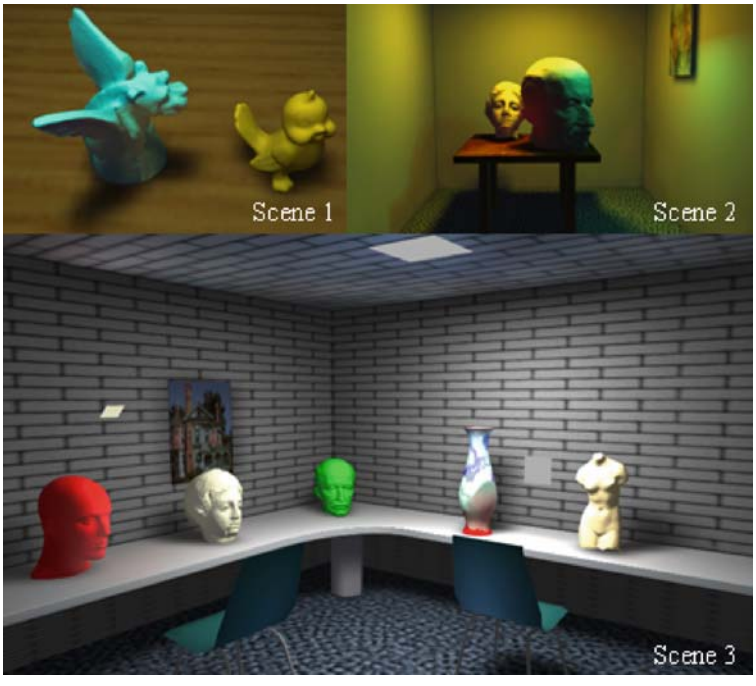


Fig. 10. Three scenes used for experiments

a Moving the light source at the roof (compare it with Fig. 10 Scene 3)

b Deforming the light source at the roof (compare it with Fig. 11(a))

Fig. 11. Transforming and deforming light source

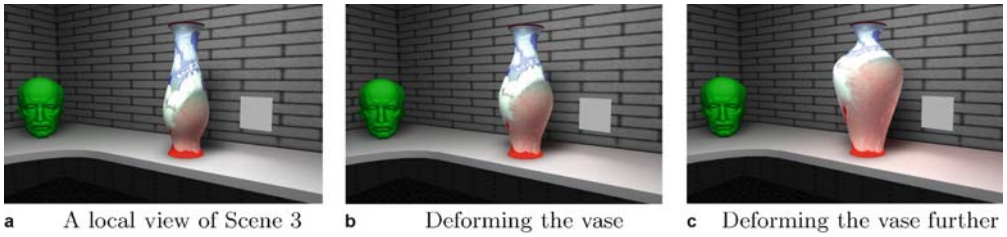
The resolutions of LIM, VRM and IIM are also determined separately. For meshes with fewer details, such as walls and floors, the LIM resolution can be rather low. Whereas, for highly detailed meshes as the statues shown in Fig. 12a, higher resolution is required. For VRM, its resolution should be chosen according to the visibility frequency on the mesh. For simplicity, we specify the resolution based on the area of the mesh. The same principle is adopted in choosing the resolution of IIM, but the resolution can be even lower.

Time and space efficiency. The time used for the scan-conversion of an object depends on the resolution of L^4 and its surface area. The time used for the remaining operations (i.e. packing, sorting and compressing) in building the local *i*-Fields depends on the number of intersection points, which is in direct proportion to the depth complexity as well as the resolution of L^4. We use a number of subdivision meshes of a statue with various size to test the overall performance of the local *i*-Field construction. From Fig. 8, we can see the time is sensitive to the surface area (or the radius of the bounding sphere), while insensitive to the number of faces. The time and space efficiency of the global illumination computation for the 3 testing scenes (Fig. 10) are listed in Table 2. Each light source is subdivided into 25 elements to calculate the LIM.Table 3 shows the frame rates during user interaction. Fig. 11a shows the change of the global illumination solution after the headlight is transformed from the original position in Fig. 10, while Fig. 11b shows the change due to headlight deformation. Fig. 12 and Fig. 13 gives some screen captures during the user interactions of changing texture, moving and deforming meshes.

Space efficiency determines the scalability of our approach, since the storage cost of the *i*-Field grows with discretization resolution in $O(N^4)$. To avoid the prohibitive storage cost, for a given *i*-Field, the "empty" lines are not

a A local view of Scene 2 **b** Changing the texture **c** Transforming the statue

Fig. 12. Texture change (takes 0.45s) and rigid transformation (takes 0.16s)



a A local view of Scene 3 **b** Deforming the vase **c** Deforming the vase further

Fig. 13. Free transformation, Updating rates are $1.9 \sim 2.4$fps

stored. The enhanced RLE compression works well for objects with larger triangles such as the walls(2 faces for each) and the tables. For the 3 test scenes, the size of the *i*-Field is 14.1MB, 72.6MB and 113.0MB, respectively (Table 2)

8 Conclusions

We present an interactive global illumination algorithm for highly dynamic environment, in which light sources and scene objects can be transformed or deformed. Tested in scenes with up to 87k triangles, our algorithm achieves interactive rates for various user interactions on a PC with 2.4GHz CPU. Our algorithm is based on three ideas:

1. Reuse the global lines for visibility query, while also reuse the intersection points.
2. Decompose the global illumination solution to make reuse more feasible.
3. Make full use of the parallel processing power of modern graphics hardware to compute in batches.

The sampling procedure is simular to that of illumination network [9]. But we sort the interseciton records by lines and organize the samples in line space, instead of sorting them by patches. This makes our approach more suitable for handling scene changes. Furthermore, *i*-Field is more friendly to compression algorithms, sinces the scene is not required to be divided into patches in carefully choosen sizes. Compared with the other approaches [13, 18, 19, 30–32, 43] which use global lines for visibility computation, our approach makes a better use of the power of the modern graphics hardware. Also, the fast scan-conversion algorithm enables us to determine the invalidated portion by scene change in a more efficient way. Compared with the hardware accelerated photon mapping method [49], our approach is not subject to the performance penalties of texture readback and copy. Moreover, we use more coherences to improve the locality of updating. Compared with the selective photon tracing approach [3, 28], our approach adapts to more kinds of interactions, including scene object deformation and light source manipulation.

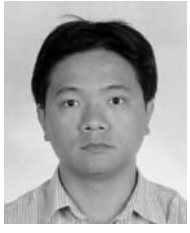
Currently, our approach is not able to catch high frequency features. We hope to integrate a ray tracer accelerated by a lower resolution line space partitioning scheme to cover these effects. Introducing adaptive techniques, both into *i*-Field and visiblity sample splatting, may significantly promote the performance and the quality. Also, the idea of GPU based sorting presented in [49] can also be used to develop a pure hardware based *i*-Field construction algorithm.

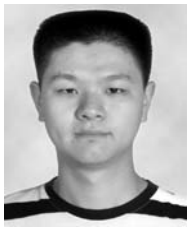
Acknowledgement This research is supported by an NSF Grant # 60203014(Geometric Signal Processing of Large Scale Scene), an NBSPC(973 program) Grant # 2002CB312102(Fundamental Theory, Algorithms of Virtual Reality and Its Implementation) and also an NSF Grant # 60021201(Science Fund of Innovation Excellence Group). We also thank Dong Xu and Xiangjun Zhao for providing the parameterized meshes.

References

1. Alexander K (1996) Quasi-Monte Carlo radiosity. EGWR: 101–100
2. Alexander K (1997) Instant radiosity. GRAPHITE: 49–56
3. Bent L, Niels J (2004) Simulating photon mapping for real-time applications. EGWR: 123–131
4. Brian E, James R (1992) David H: An importance-driven radiosity algorithm. ACM SIGGRAPH: 273–282
5. Bruce W, Gun A, et al. (1997) Fitting virtual lights for non-diffuse walkthroughs. ACM SIGGRAPH: 45–48
6. Bruce W, George D, Steven P (1999) Interactive rendering using the render cache. EGWR 10:235–246
7. Bruce W, George D, Donald G (2002) Enhancing and optimizing the render cache. EGWR: 37–42
8. Carsten B, Ingo W, Philipp S (2003) A scalable approach to interactive global illumination. Computer Graphics Forum 22(3):621–630
9. Chris, B, Donald, F (1989) Illumination networks: fast realistic rendering with general reflectance functions. ACM SIGGRAPH: 23(3):89–98
10. Cyrille D, Francois S (1999) Space-time hierarchical radiosity for high-quality animations. EGWR: 235–246
11. Cyrille D, Kirill D, Karol M (2003) State of the art in global illumination for interactive applications and high-quality animations. Computer Graphics Forum 22(1):55–77
12. David G, Francois S, Donald G (1990) Radiosity redistribution for dynamic environments. IEEE CG&A 10(4):26–34
13. Francesc C, Matue S, László N (2004) Fast multipath radiosity using hierarchical subscenes. Computer Graphics Forum 23(1):43–54
14. Frank S, Andreas P (1999) Reducing memory requirements for interactive radiosity using movement prediction. EGWR: 225–234
15. Frédo D (1999) 3D visibility analytical study and applications. PhD Thesis, Université Joseph Fourier, Grenoble, France.
16. Gene G, Peter S, et al.(1998) The Irradiance Volume. IEEE CG&A 18(2):32–43
17. George D, Francois S (1997) Interactive update of global illumination using a line-space hierarchy. ACM SIGGRAPH: 57–64
18. Gonzalo B, Mateu S (1996) The multi-frame lighting method: a Monte Carlo based solution for radiosity in dynamic environments. EGWR: 185–194
19. Gonzalo B, Pueyo X (2001) Animating radiosity environments through the multi-frame lighting method. J. Visual. Comp. Animat. 12(2):93–106
20. Gregory W, Maryann S (1999) The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments. ACM TOG 18(4):361–368
21. Hector Y, Sumanita P, Donald PG (2001) Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. ACM TOG 20(1):39–65
22. Ingo W, Philipp S, et al. (2001) Interactive rendering with coherent ray tracing, Computer Graphics Forum 20(3):153–164
23. Jeffry N, Julie D, Holly R (1996) Implementation and analysis of an image-based global illumination framework

for animated environments. IEEE Trans. on Visualization and Computer Graphics 2(4):283–298
24. Jensen H (1996) Global illumination using photon maps. EGWR: 21–30
25. Johannes G, Ingo W, Philipp S (2004) Realtime caustics using distributed photon mapping. EGWR: 109–121
26. Karol M, Takehiro T, et al. (2001) Perception-guided global illumination solution for animation rendering. ACM SIGGRAPH: 221–230
27. Kavita B, Julie D, Seth T (1999) Radiance interpolants for accelerated bounded-error ray tracing. ACM TOG 18(3):213–256
28. Kirill D, Stefan B, et al. (2002) Interactive global illumination using selective photon tracing. EGWR: 25–36
29. Larry A, Pat H (1993) A hierarchical illumination algorithm for surfaces with glossy reflection. GRAPHITE: 155–162
30. Laszlo K, Werner P(1998) Global ray-bundle tracing with hardware acceleration. EGWR: 247–258
31. Laszlo K (1999) Stochastic iteration for non-diffuse global illumination. Computer Graphics Forum 18(3):233–244
32. Laszlo K, György A, Balazs B (2003) Global illumination animation with random radiance representation. EGWR: 64–73
33. Marc S, Annete S, et al. (2000) Efficient glossy global illumination with interactive viewing. Computer Graphics Forum 19(1):13–25
34. Mark S, Jörg H, et al. (2000) Walkthroughs with corrective texturing. EGWR: 377–390
35. Martin I, Pueyo X, Tost D (2003) Frame-to-Frame coherent animation with two-Pass radiosity. IEEE Trans. on Visualization and Computer Graphics Jan.: 70–84
36. Maryann S, Carlo S (2000) Tapestry: a dynamic mesh-based display representation for interactive rendering. EGWR: 329–340
37. Parag T, Fabio P, et al. (2002) Interactive global illumination in dynamic scenes. GRAPHITE: 537–546
38. Per C, Dana B (2004) An irradiance atlas for global illumination in complex production scenes. EGWR: 132–141
39. Pfister H, Zwicker M, et al.(2000) Surfels: surface elements as rendering primitives. ACM SIGGRAPH: 335–342
40. Pueyo X, Tost D, et al. (1997) Radiosity for dynamic environments. The Journal of Visualization and Computer Animation 8(4):221–231
41. Rafal M, Sumanta P, Karol M (2002) Cube-map data structure for interactive global illumination computation in dynamic diffuse environments. ICCVG: 25–29
42. Rui B, Kenneth H, et al. (1999) Increased photorealism for interactive architectural walkthroughs. ACM Symposium on Interactive 3D Graphics: 183–190
43. Sbert M, Pueyo X (1996) Global multipath Monte Carlo algorithms for radiosity. The Visual Computer 12(2):47–57
44. Sbert M, Szécsi L, Laszlo S (2004) Real-time light animation. Computer Graphic Forum 23(3):291–299
45. Seth T, Kavita B, Julie D (1996) Conservative radiance interpolants for ray tracing. EGWR: 257–268
46. Shenchang C (1990) Incremental radiosity: an extension of progressive radiosity to an interactive image synthesis system. GRAPHITE: 135–144
47. Shenchang C, Holly R, et al. (1991) A progressive multi-pass method for global illumination. GRAPHITE: 165–174
48. Steve P, William M, Perter-Pike S (1999) Interactive ray tracing. ACM Symposium on Interactive 3D Graphics: 119–126
49. Timothy P, Craig D, et al. (2003) Photon mapping on programmable graphics hardware. SIGGRAPH/EUROGRAPHICS Workshop On Graphics Hardware: 41–50
50. Tommer, L, Olga, S, Daniel, C (2003) Ray space factorization for from-region visibility. ACM SIGGRAPH 22(3):595–604
51. Valdimir V, Karol M, et al. (2000) Using the visual differences predictor to improve performance of progressive global illumination computation. ACM TOG 19(2):122–161
52. William S, James F, et al. (2004) Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. ACM TOG 23(3):742–749
53. William M, Leonard M, Gary B (1997) Post-rendering 3D warping. Symposium on Interactive 3D Graphics: 7–16
54. Wolfgang S, Rui B (1997) Interactive rendering of globally illuminated glossy scenes. EGWR: 93–102
55. Xavier G, George D (2001) Incremental Updates for Rapid Glossy Global Illumination. Computer Graphics Forum 20(3):268–277






ZHONG REN received the BSc degree in information engineering from Zhejiang University in 2000. Now He is a PhD candidate in the State Key Laboratory of CAD&CG of Zhejiang University. His research interest is real-time rendering.

WEI HUA received the PhD degree in applied mathematics from Zhejiang University in 2002. He is now an associated professor of the State Key Laboratory of CAD&CG of Zhejiang University. His research interests include real-time simulation and rendering, virtual reality and software engineering.

LU CHEN received the BSc degree in computer science from Zhejiang University in 2005. He is now a graduate student of the State Key Laboratory of CAD&CG of Zhejiang University. His research interests is real-time simulation and rendering.

HUJUN BAO received his Bachelor and PhD in applied mathematics from Zhejiang University in 1987 and 1993. His research interests include modeling and rendering techniques for large scale of virtual environments and their applications. He is currently the director of State Key Laboratory of CAD&CG of Zhejiang University. He is also the principal investigator of the virtual reality project sponsored by Ministry of Science and Technology of China.