EUROGRAPHICS 2002 / I. Navazo Alvaro and Ph. Slusallek
(Guest Editors)

An Image-Based Radiosity Representation

T. Mertens and F. Van Reeth†

Expertise Centre for Digital Media, Limburg University Centre, Diepenbeek, Belgium

Abstract

We present a geometry-independent radiosity representation which stems from image-based methodologies. Our image-based representation is inspired by the concept of layered depth images. To construct the radiosity function we render shaded depth images and warp them into the representation. This enables us to alter Keller's instant radiosity algorithm to a view-independent variant, allowing interactive viewing of complex scenes without relighting every frame. Image warping is done on the host CPU, while graphics hardware is exploited for rendering the images. Our approach differs from the traditional finite element methods since no a priori meshing, nor an explicit form factor evaluation needs to be performed. The geometry independence ensures a reliable rendering time and memory usage, which are more dependent on image resolution rather than scene complexity. It also implies that lighting information is decoupled from scene representation, thereby avoiding the restriction to planar surfaces and issues such as discontinuity meshing.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Bitmap and framebuffer operations I.3.6 [Computer Graphics]: Graphics data structures and data types I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture—Radiosity

1. Introduction

Rendering scenes using physically accurate lighting has become an important feature for architectural walkthroughs, virtual reality, games, and other graphical applications. The radiosity function describes the diffuse interreflection in an environment, which means that light arriving at a surface is scattered in a uniform fashion. Therefore radiosity algorithms are able to provide a view-independent lighting description of a scene. It allows to precompute the lighting in a scene, and to reuse it for real-time display.

1.1. Overview

The main contribution of this paper is the introduction of an image-based radiosity representation (IBRR), based on LDIs, which allows view-independent instant radiosity.
The rest of this paper is organised as follows: first we discuss related work. In section 3 we introduce our representation. Construction and reconstruction of the radiosity function is explained in sections 4 and 5, respectively (see figure 1 for a depiction). Finally, results of our implementation are shown and further plans are mentioned, followed by our conclusion.

2. Related Work

2.1. Global Illumination and Radiosity

Global illumination describes the interaction of light with the matter in a scene. It is mathematically modeled by the rendering equation[6]. In the radiosity setting, we assume that the scene contains only diffuse reflectors and sources. Here, the rendering equation can be reduced to a simpler form, namely the radiosity equation:[4]

$$B(x) = E(x) + \rho(x) \int_S B(y) g(x,y) dA_y \qquad (1)$$

with B the radiosity function, E the self-emitted energy, ρ the reflectivity, S the surfaces in the scene and $g(x,y)$ is the geometry term. The latter indicates the fraction of energy that is transferred between two points x and y. Note that visibility needs to be accounted for in g, which is the most computation-intensive part.
Traditionally, the radiosity is computed using so-called finite element methods. Its domain S is discretised into "patches"

† {tom.mertens,frank.vanreeth}@luc.ac.be

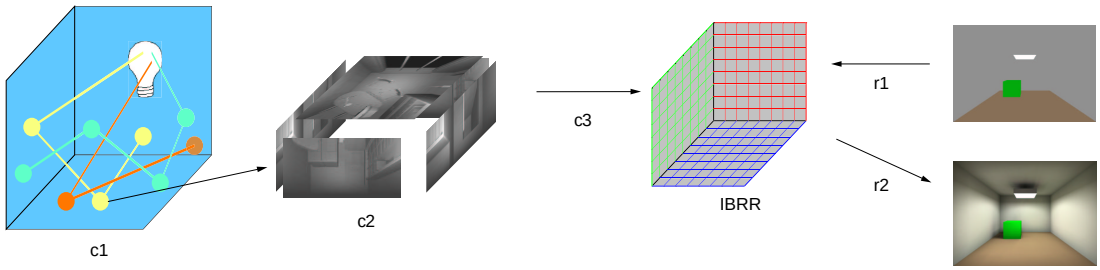

Figure 1: *Illustration of our approach. Construction:* ***(c1)*** *point light samples are generated across the scene as in instant radiosity.* ***(c2)*** *The environment is projected onto the 5 faces of a hemicube placed on the point light.* ***(c3)*** *The resulting 5 images are warped into the IBRR and the point light's contribution is added. Reconstruction:* ***(r1)*** *the scene is rendered from the camera viewpoint, and pixels are warped to the IBRR.* ***(r2)*** *Each pixel is modulated with a shading value retrieved from the IBRR.*

for which the diffuse interactions are calculated.[5] Cohen and Wallace[4] give a comprehensive overview of various finite element methods and related techniques. A lot of work has been done to make finite element methods efficient in both rendering time and quality. On the other side, implementation can be complex due to the meshing algorithms which are necessary to represent high frequencies in the radiosity function, e.g. shadow boundaries. Also, non-planar surfaces like NURBS-patches complicate things. Finite element methods are tightly coupled to the geometry, and for detailed and complex scenes they may become unpractical.
Stochastic ray tracing and Monte Carlo techniques can be employed to generate a radiosity image without discretisation, but are not (yet) suitable for interactive viewing.
Photon mapping is a very powerful global illumination technique that can generate practically every optical effect. A so-called *photon map* records the interaction of light particles shot from the light sources. After construction, stochastic ray tracing and radiance estimations based on data in the photon map are used to generate images. The map itself only contains photons; it does not contain any surface information. In that respect our method is similar to photon mapping: the representation is decoupled from geometry.

2.2. Instant Radiosity

A different approach was taken by Keller[7]: his *instant radiosity* technique combines Monte Carlo integration and traditional hardware-based illumination. The propagation of light is modeled by shooting particles from the light source in the scene. When it hits a surface, a point light is placed at that location, and the particle's path is continued until it gets absorbed. The radiosity function is constructed by rendering the scene for every point light, and adding the resulting images in the accumulation buffer. The algorithm can converge quickly to a solution: rendering the scene lit by point lights and the accumulation buffer can both be implemented using standard graphics hardware. However, this technique suffers from view dependence because the radiosity integral is re-evaluated for every frame. This implies that interactive walkthroughs are difficult, and are feasible if only a few point samples are taken, thereby sacrificing image quality.
In this paper, we address the view dependency problem by moving the accumulation from image-space to a view-independent space.

2.3. Image-Based Techniques

Image-based techniques have gained a lot of interest over the last years. In particular, image-based rendering is becoming quite popular as an alternative to classic image synthesis algorithms. Image-based rendering differs from traditional image synthesis because the scene is represented as set of reference images, usually containing depth information, which can synthetic or captured by real-life photography. New views are generated by warping one or more reference images to a destination view. An interesting aspect is that computation time is dependent on image resolution rather than scene complexity. Shade et al.[11] introduced an efficient data structure in the context of image-based rendering called a *layered depth image*. The idea is to store reference images by warping them under the same projection, implying that the pixels of the resulting image can have multiple entries. This paper builds on the concept of layered depth images to construct radiosity.
Image-based techniques have not solely been used for image synthesis. One of the most famous shadow algorithms is *shadow mapping*[12]. Here, a depth image is rendered from the light's point of view to see which objects are visible from the light. During rendering the depth information is used to decide which pixels are lit. Agrawala et al.[1] extended this approach to compute shadows with realistic penumbrae. Multiple point samples are taken over an extended light source by rendering depth views. These samples are warped and accumulated in an LDI, such that shadow attenuation is constructed. During rendering, these attenuation factors are mapped onto the camera view. Their technique is related to ours, since visibility information is stored in an image-based representation. However, in our case, we also use shading

aside from visibility. Moreover, we perform sampling over the whole scene.

3. The Image-Based Radiosity Representation

The IBRR allows to build the radiosity function by warping images. Since we essentially sample the scene with pixels, it is independent of its geometry. The two main operations on the representation are insertion and retrieval of pixels, which are used for construction and reconstruction, respectively. Our representation is based on a *layered depth cube*[8] (LDC);

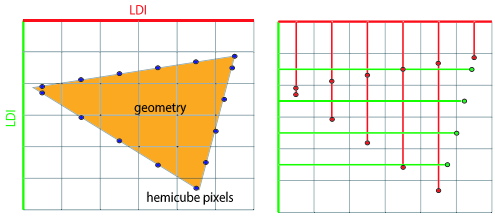

Figure 2: *2D analogy of the IBRR.* ***Left:*** *Geometry is sampled by rendering images from the hemicube sampling. Each pixel is warped to one of the 3 LDIs. The most parallel LDI is chosen according to the pixels' normals.* ***Right:*** *Data layout of the IBRR after warping.*

see figure 2. It consists of 3 parallel LDIs arranged in a cube. The warping procedure involves unprojecting a pixel based on its depth from the Z-buffer, and computing its position in an LDI. If two pixels fall together, their values are accumulated, otherwise a new layer is inserted. Pixels are stored in the LDC according to their orientation: we choose the LDI that is most parallel. This means that a per-pixel normal has to be provided. We opted for this structure for the following reasons:

- LDI-resolution will be fairly uniform over the scene, and the support of an LDI pixel over a surface will be limited. If the angle between the LDI's direction and a surface's normal would be too big, images will be resampled too coarsely.
- Insertion and retrieval of pixels need to be performed in 1 LDI only, since the appropriate one is chosen according to the incoming pixel's normal.

3.1. Robust Pixel Representation

During insertion and retrieval, we have to compare incoming pixels to pixels in the LDIs. In our representation, pixels are modeled as planes instead of depth values. An incoming pixel is compared to an LDI-pixel according to their plane equations instead of their depth values. Pixels are stored as a plane equation: a quadruple $(a,b,c,d) = (n_x, n_y, n_z, p \cdot n)$ with n the normal and p its world space position. The normal n is encoded in the RGB channel during rasterisation, while p is obtained from unprojecting the pixel. For an incoming pixel at position p' with normal n' we consider it equal to an LDI-pixel if:

$$|p' \cdot n - d| < \varepsilon \quad \text{and} \quad n' \cdot n > 1 - \varepsilon$$

for some small threshold ε. The first condition states that the

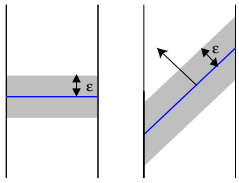

Figure 3: ***Left:*** *Pixels are stored as depth values. When an incoming pixel has to be compared to a pixel in the LDI, they are considered equal if the difference in depth is below a threshold ε. The ε-region indicates where pixels can be considered equal.* ***Right:*** *Pixels are modeled as planes instead of depth values in the LDI. Here the ε-region is fitted more tightly to the surface, since the perpendicular distance to the plane is observed.*

perpendicular distance from the LDI-pixel's plane to p' must be small enough. Figure 3 explains why this is better than just comparing depth values. The second condition states that their orientations should be roughly the same, which is essential to represent small geometry which can only be coarsely represented in the LDI.

4. Precomputation

In instant radiosity, the radiosity function is constructed by lighting the scene by numerous point light sources and accumulating the contribution of each one, thus performing point sampling across the whole scene. The essence of our approach is to generate a set of instant radiosity point lights (IRPLs), and accumulating their contributions in the IBRR, rather than image-space.
The procedure to construct the radiosity function is depicted in figure 1 and 4.

```
shoot photons
for each photon bounce {
     place light source
     render hemicube faces
     warp faces to IBRR
     accumulate
}
```

```
render camera view
for each pixel {
     warp view into IBRR
     retrieve shading r
     pixel color = ρ*r
}
```

Figure 4: *Pseudo code description.* ***Left:*** *Constructing the radiosity function.* ***Right:*** *Reconstruction.*

4.1. Hemicube Sampling

Hemicube sampling[3] was introduced in the context of finite element methods, for the purpose of computing form factors using graphics hardware. Our hemicube sampling variant is similar in a sense that we also rasterise the scene onto 5 hemicube faces. However, instead of rendering patches with unique color-IDs, we render the surfaces lit by the corresponding IRPL. By warping the 5 resulting images into the IBRR, we essentially resample the sampled geometry into a view-independent space.
Since we exploit the raw polygon processing power of graphics hardware for rendering the hemicube faces, visibility is handled efficiently even for complex scenes.
To accelerate construction, we decrease the resolution of the hemicube after the first node of a photon's path. This node represents direct lighting and needs more resolution to render shadows as sharp as possible. Indirect lighting does not contain such high frequencies and can be handled with less care.

4.2. Instant Radiosity

The radiosity function can be represented as a summation over all the IRPLs:

$$B(x) = \sum_i \rho(x) I_i g_i(x) \tag{2}$$

where $\rho(x)$ is the reflectivity at x, I_i is the intensity of the ith IRPL, and $g_i(x)$ is the geometric component of the point light model, similar to the one in equation 1:

$$g_i(x) = \frac{\cos\theta_x \cos\theta_i}{\pi r^2}$$

$g_i(x)$ is computed in hardware using per-pixel lighting. Vertex lighting could also do the job, but that would also require some degree of tesselation of the geometry to obtain smooth lighting. Since we aim at geometry independence, per-pixel lighting is essential. nVIDIA's developer site[9] provides some courseware about implementing per-pixel lighting with vertex and pixel shaders.
Because the RGB channels are taken by the color-encoded normal, the geometry term is put in the α channel. Due to the limited range of the frame buffer $g_i(x)$ is clamped to $[0,1]$, which was also the case in the original instant radiosity algorithm. The singularity caused by the inverse quadratic fall-off in $g_i(x)$ is responsible for exceeding the $[0,1]$ interval. It is also possible to compute the fall-off in software during warping to obtain more accurate results.
Now that we have the geometric component in equation 2, we only need to multiply it by I_i, which is done just before warping. In equation 2, $\rho(x)$ can be moved outside the summation. Its contribution is postponed until reconstruction, to avoid rendering an extra image for obtaining reflectivity (also see figure 4).
Note that this light model does not include a visibility term: since the IRPLs are placed at the viewpoint of that sample, we do not need render shadows. Visibility is implicitly accounted for by warping the image into the IBRR.[1]

4.3. Splatting

We employ an ad hoc solution for the splatting problem, similar to Agrawala et al.[1]. First, the splat size is estimated according to the distance and orientation of a pixel. The splatting is performed by inserting extra neighboring copies of that pixel. Overlaps from the same hemicube are skipped to avoid erroneous accumulation. This results in "stretched" pixels, as seen in texture mapping using nearest neighbor filtering. Since the contribution of a IRPL does not change abruptly, shading remains smooth. On the other hand, in some cases shadow edges can be moved inward or even erased for thin geometry due to an overestimated splat size.
This approach may seem like a very rudimentary form of splatting, but experience shows that as long as the resolutions of the LDIs and the hemicube faces are kept roughly the same, the result is satisfactory.

5. Reconstruction

Reconstruction is done by warping each pixel in the view image to the IBRR in order to retrieve a modulation value.[1] See figure 1 and 4 for a description of this procedure.
Shading is thus deferred until the camera view has been rendered. Since the scene can be rendered quickly in hardware, reconstruction is mostly dependent on the depth complexity of the LDIs, rather than scene complexity.
To mask possible aliasing artifacts, the LDI-pixels can be filtered with an averaging kernel.

6. Results and Discussion

The current implementation was developed on an Intel P4 1.3Ghz, equipped with an nVIDIA GeForce3 graphics board. At the moment it consists of straightforward, unoptimised C++ code and OpenGL calls.
The measurements in table 1 confirm our claims. It is clear that construction and recontruction time are mostly dependent on resolution rather than geometry. The same goes for memory requirements, which are expressed as the number of LDC pixels times the size of such a pixel. Moreover, even when more IRPL samples are taken, the size of the IBRR increases only slightly, analogous to the memory measurements of Agrawala et al.[1].
Figure 5 provides some visual results.

6.1. Important Issues

Every image-based technique is inherently prone to undersampling, resulting in aliasing artifacts. Images have a limited sampling rate and can cause a loss of detail of the radiosity function, especially near shadow boundaries. Even

	LDC	#IRPL	constr.	recon. fps	size Kb
A	128×128	184	17”	4.0	4346
	256×256		1’2”	3.6	12667
	512×512		3’56"	3.1	41114
B	128×128	184	51”	3.4	4845
	256×256		1’26”	3.4	11222
	512×512		3’49”	3.1	30783
C	256×256	200			17570
		371			18787
		589			18742
		751			18743

Table 1: *Measurements using a scene A consisting of 15K triangles, a scene B consisting of 97K triangles, and C consisting of 26K triangles. The resolution of the hemicube faces are kept the same as the LDC resolution.*

worse, it can imply an erroneous accumulation of the samples in the LDIs. Imagine the following situation. The frustum formed by a pixel can be occupied by more than one visible surface. This occurs in highly detailed scenes, or when an object is located far from the sampling viewpoint. Since hardware rasterisation records a single depth value per pixel, the contribution of the other surfaces is not added to the LDI, resulting in incorrectly lit spots and discontinuities. Therefore image resolution must be chosen high enough to avoid artifacts. Averaging with neighboring LDI-pixels can mask some of the errors, but not all of them.
Another problem lies with the architecture of our system. During construction and reconstruction images need to be transferred constantly from the graphics card to the CPU and back. This causes a performance bottleneck, since the available bandwidth between graphics memory and CPU memory is limited.

6.2. Further Work

In order to preserve the sampling rate of an image, a hierarchical variant of the LDI can be employed such as the LDI-tree[2] or the LDC-tree[10]. A hierarchical representation may also reduce memory cost; regions which are coarsely sampled can be represented with less LDI-pixels.
We plan to implement more efficient splatting with a gaussian kernel. This will eliminate problem at shadow boundaries. Also, when sampling with low resolution images, jagged shadow artifacts will be softened.
In the reconstruction phase, bilinear interpolation can be employed to smoothen the radiosity function. However, this is not trivial since we need to interpolate over different LDIs.
The technology used in modern off-the-shelf graphics boards advances at a very fast pace, and their programmability and functionality expand rapidly. Therefore construction and reconstruction phases may be mapped further onto hardware.
We plan to extend our representation to handle non-diffuse situations. However, this will give rise to issues concerning a more complex construction and reconstruction, and increased memory requirements.

7. Conclusion

We introduced a novel representation for the radiosity function, based on image-based methodologies. The radiosity function is constructed by sampling the scene as follows: shaded images are rendered using graphics hardware and warped into the IBRR. This allows to adapt the instant radiosity algorithm to a view-independent variant, thus making walkthroughs feasible, even for situations with a high sample count. The high polygon throughput of current off-the-shelf graphics hardware can be exploited, implying that complex scenes can be dealt with efficiently.
The resulting approach exhibits interesting properties. Its key feature is its geometry independence, and can be advantageous over pure geometry-based techniques. Quality, rendering time and memory usage are dependent on resolution of the sample images, rather than geometry. Complex techniques such as discontinuity meshing are avoided. Also, there are no restrictions on the surfaces in the scene; any type of surface that the graphics hardware renders, can be handled.

Acknowledgements

Part of this work is funded by the European Regional Development Fund and the Flemish government.
Thanks to Paul Akkermans for modeling the office scene.
We are indebted to Philippe Bekaert for his valuable comments and hints.

References

1. Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich, and Laurent Moll. Efficient Image-Based Methods for Rendering Soft Shadows. In *Computer Graphics (ACM SIGGRAPH ’00 Proceedings)*, volume 34, pages 375–384, 2000. 2, 4

2. Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. LDI tree: A Hierarchical Representation for Image-Based Rendering. In *Computer Graphics (ACM SIGGRAPH ’99 Proceedings)*, volume 33, pages 291–298, 1999. 5

3. Michael F. Cohen and Donald P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. In *Computer Graphics (ACM SIGGRAPH ’85 Proceedings)*, volume 19, pages 31–40, 1985. 4

4. Michael F. Cohen and John R. Wallace. *Radiosity*

Figure 5: ***Upper image:*** *an example scene containing roughly 100K polygons.* ***Lower image:*** *the same scene viewed from above. The complex shadowing effects on the floor are created without tesselation.*

and Realistic Image Synthesis. Academic Press Professional, Boston, MA, 1993. 1, 2

5. Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 212–222, July 1984. 2

6. James T. Kajiya. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, 1986. 1

7. Alexander Keller. Instant radiosity. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 49–56, 1997. 2

8. Dani Lischinski and Ari Rappoport. Image-Based Rendering for Non-Diffuse Synthetic Scenes. In *Rendering Techniques '98*, pages 301–314. Springer-Verlag/Wien, 1998. 3

9. nVIDIA's Developer Site. `http://developer.nvidia.com/`. 4

10. Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface Elements as Rendering Primitives. In *Computer Graphics (ACM SIGGRAPH '00 Proceedings)*, volume 34, pages 335–342, 2000. 5

11. Jonathan W. Shade, Steven J. Gortler, Li-Wei He, and Richard Szeliski. Layered Depth Images. *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, 32:231–242, 1998. 2

12. Lance Williams. Casting Curved Shadows on Curved Surfaces. In *Computer Graphics (ACM SIGGRAPH '78 Proceedings)*, volume 12, pages 270–274, 1978. 2