Light Vectors for a Moving Observer

Rodolphe Crespin   Bernard Péroche

L.I.R.I.S : Lyon Research Center for Images and Intelligent Information Systems
CNRS / INSA de Lyon / Université Lyon 1 / Université Lyon 2 / Ecole Centrale de Lyon
Bâtiment Nautibus, 8 boulevard Niels Bohr
69622 Villeurbanne Cedex, FRANCE

rodolphe.crespin@liris.cnrs.fr   bernard.peroche@liris.cnrs.fr

ABSTRACT

Interactive rendering is usually made with very simple illumination models. High quality rendering is too slow to be interactive. In this paper, we extend the notion of *light vector* by computing it in 5D and we try to obtain an interactive high quality rendering with global illumination by merging the notion of *light vector* with the *render cache* approach.

Keywords

Global Illumination, Ray Tracing, Image Based Rendering, Interactive Rendering.

1 INTRODUCTION

In computer graphics, rendering is the final stage that computes pixels' color. Rendering has been greatly studied since the sixties' end; depending on the application domain, it can be classified in the categories wich follow:

- *Interactive rendering* is the most used technique and is generally based on graphic cards abilities and on a polygonal representation of the scene to allow realtime rendering. A high quality of generated images can be reached by using textures (environment mapping [BN76], bump mapping [Per85, Bli78], mapping [SA99], ...). However, for a few years, ray tracing based techniques have allowed interactive times on classical computers using databases with several millions of polygons [WBS+02].
- *Photorealistic rendering* tries to simulate light and its behavior with materials. This simulation allows effects such as refraction, shadows, interreflections, caustics, spectral phenomena, ...
  For the last fifteen years, a lot of solutions have been suggested in order to obtain this kind of picture: ray tracing [Whi80], path tracing [Kaj86], bidirectional ray tracing [VG94, LW93], radiosity [SP94], photon maps [Jen96, JC98], light vectors [ZP98], ... All these methods have a common drawback: computation is very expensive and takes a lot of time (from several minutes to several hours for one picture).
- *Non-photorealistic rendering* has appeared more recently and took an interest in the artistic expressiveness; it seeks to simulate artistic techniques such as engraving [Ost99], painting [Lit97] (impressionism [Lit97], water color [CAS+97], ...), technical drawing [TDM99], ...

These different types of rendering are not exclusive. Actually, interactive rendering is to become more and more realistic with the use of graphic cards potential, using methods like multi-texturing [SA99], graphic engines [LKM01] or pixel shaders [SA99]. Further propositions [DDM03] have been done to accelerate photorealistic rendering: For example, such techniques are caching based methods [LS99, BDT99, WDP99] or hardware optimization based methods [WBS+02].

The goal of this work is to compute photorealistic images in a nearly interactive time. In order to obtain such a result, two rather different approaches have been combined: the *render cache*'s approach and *light vectors*. However, the concept of *light vector* was initially introduced for static scenes with a fixed viewpoint. To overcome these limitations, we suggest to extend *light vectors* to $5D$, by introducing a caching method for directions in addition to the spatial caching method defined by the original concept.

Journal of WSCG, Vol.12, No.1-3., ISSN 1213-6972
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.
Copyright UNION Agency – Science Press

The structure of the paper is made up by five sections: Section 2 introduces the concepts on which our work is based. Section 3 presents our extension of *light vectors*. Some validity criteria for performing *light vectors* interpolation and for computing the priority of a pixel in the *render cache* approach are given in Section 4. Some results are described in Section 5. A conclusion and future work are discussed in Section 6.

2 PREVIOUS WORK

In this section, we will present the main concepts on which our work is based: *light vectors* and the *render cache*.

The Notion of Light Vector

2.1.1 *Introduction*

A *light vector* (LV) is a concept introduced by Zaninetti *et al.*[ZP98], based on the work of Ward[WRC88, WH92]. Its main purpose is to catch radiances as vectors in a buffer. So, instead of computing radiance at each point, the already computed *light vectors* may be interpolated (if some criteria are fulfilled) to produce a new *light vector*, from which the radiance at the given point can be deduced. Actually, it comes down to replace the complete scene at a given point with a virtual light source; thus, a *light vector* is defined by a direction $\overrightarrow{D}$, calculated with equation (2), and a magnitude M, calculated with equation (3).

Equation (1), based on Kajiya's one [Kaj86], shows that the radiance reflected at point x of the scene in a direction $\overrightarrow{\omega_r}$ depends on the incident radiances L_i according to all possible directions $\overrightarrow{\omega_i}$, on the $BRDF$ associated to directions $\overrightarrow{\omega_i}$ and $\overrightarrow{\omega_r}$ and on emitted radiance when the object is a light source.

$$\begin{aligned} L_r(x,\overrightarrow{\omega_r}) &= L_e(x,\overrightarrow{\omega_r}) \\ &+ \int_{\Omega_i} f_r(x,\overrightarrow{\omega_i} \rightarrow \overrightarrow{\omega_r}) L_i(x,\overrightarrow{\omega_i}) \cos\theta_i d\omega_i \end{aligned} \quad (1)$$

The *light vector*'s components can be defined from equation (1) by:

$$\overrightarrow{D} = \int_{\Omega_i} L_i(x,\overrightarrow{\omega_i})\overrightarrow{\omega_i} d\sigma(\overrightarrow{\omega_i}) \quad (2)$$

$$M = \frac{\int_{\Omega_i} f_r(x,\overrightarrow{\omega_i} \rightarrow \overrightarrow{\omega_r}) L_i(x,\overrightarrow{\omega_i}) \cos\theta_i d\omega_i}{f_r(x,\overrightarrow{D} \rightarrow \overrightarrow{\omega_r}) \cos(\overrightarrow{n},\overrightarrow{D})} \quad (3)$$

where $d\sigma(\overrightarrow{\omega_i})$ is the solid angle associated to the incidence direction $\overrightarrow{\omega_i}$.
We may notice that the presence of the $BRDF$ in equation (3) introduces a viewpoint dependence.
Equation (1) may be decomposed into five independent components:

$$\begin{aligned} L_r(x,\overrightarrow{\omega_r}) &= L_e(x,\overrightarrow{\omega_r}) + L_{spec}(x,\overrightarrow{\omega_r}) \\ &+ L_{dir}(x,\overrightarrow{\omega_r}) + L_{ind}(x,\overrightarrow{\omega_r}) + L_{caust}(x,\overrightarrow{\omega_r}) \end{aligned} \quad (4)$$

where $L_e(x,\overrightarrow{\omega_r})$ is the emitted radiance, $L_{spec}(x,\overrightarrow{\omega_r})$ the specular component, $L_{dir}(x,\overrightarrow{\omega_r})$ the direct one, $L_{ind}(x,\overrightarrow{\omega_r})$ the indirect one and $L_{caust}(x,\overrightarrow{\omega_r})$ the caustic component.
This decomposition allows to be specific to each component. Actually, the components have not the same behavior. For example, the indirect irradiance changes slowly whereas the direct one has more important variations, with possible discontinuities. Thus, the treatment can be optimized for each component.
If $L_e(x,\overrightarrow{\omega_r})$ exists, it is defined by the properties of the object. $L_{spec}(x,\overrightarrow{\omega_r})$ has too many variations, the interpolation will fail; its value is computed with the ray tracer.

The remaining three radiance components lead to three types of *light vectors*:

- *direct light vectors (DLV)*, for radiance coming directly from the light sources;
- *indirect light vectors (ILV)*, for radiance arriving at a point after at least one non specular reflection on an object;
- *caustic light vectors (CLV)*, for radiance coming to a non specular object after at least an interaction with a non Lambertian surface.

Figure 1 illustrates this decomposition.

Computing indirect radiance is very expensive (in general, a Monte-Carlo method is used). Thus, we chose to work on indirect radiance in order to test our method. Of course, the independence between the three components allows our method to be applied to the two other radiance components.

2.1.2 *The Algorithm*

The classical ray tracing algorithm [Whi80] is used for computing the components of Equation (4) that do not lead to a *light vector* computation.
A *light vector*'s seed has to be computed during the algorithm's initialization. As pixels are scan-line computed, a systematic bias would be introduced if the computation was done only in the scan-line order. The problem can be avoided by using a random seed.

A sketch of the algorithm for the computation of a picture with *light vectors* is shown in Figure 2.

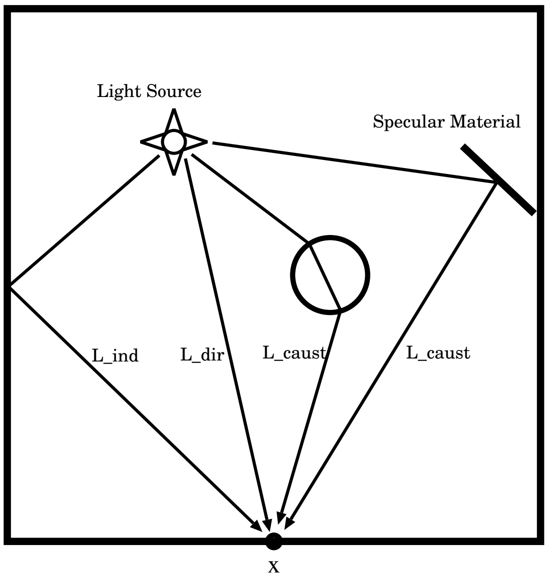

Figure 1: **Radiance Components**

```
Create the seed
For each pixel do
  Generate a primary ray
  Intersect this primary ray with the scene
  If an object is intersected then
    Compute the nearest intersection point $x$
    If there is no computed LV at point $x$ then
      Seek for LVs in the neighbourhood of $x$
      If the number of LVs is greater than a threshold
      And If criteria are fulfilled then
        Interpolate these LVs
        End If
      Else Compute a LV at point $x$
      End If
    Compute radiance at point $x$
    End For
```

Figure 2: **Image computation algorithm with LVs**

2.1.3 The Case of ILVs

If the space of directions is discretized, the indirect part of the rendering equation can be written:

$$L_{ind}(x,\overrightarrow{\omega_r}) \approx \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}[\rho_d(x,\overrightarrow{\omega_{j,k}}\rightarrow\overrightarrow{\omega_r}) L_i(x,\theta_j,\phi_k)\cos\theta_j] \quad (5)$$

where M and N are the number of azimuthal and zenithal regular splittings of the hemisphere centered at point x, θ_j and ϕ_k are the angles defining vector $\overrightarrow{\omega_{j,k}}$ and ρ_d represents the diffuse part of the BRDF.
A random ray is shot through each cell to gather coming in energy. After having recovered diffuse information for each cell, the average radiance magnitude M_0 emitted by point x can be computed, which gives the fields of the ILV.

Each *light vector* is related to a location and a viewpoint. We will study how to overcome this limitation in Section 3 by extending *light vectors* to $5D$.

During the evaluation of equation (5), an irradiance gradient ΔI, inspired by Ward's gradient[WH92], may be computed. Let $I(x)$ be the irradiance at point x.

$$I(x) = \int_{\Omega_i} L_i(x,\overrightarrow{\omega_i})\cos\theta_i d\omega_i$$

Let us discretize this irradiance equation and derive it in a neighborhood of the point where an ILV has been computed. If we denote $d = x$, y or z, then:

$$\begin{aligned}\frac{\partial I}{\partial d} &= \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}\left[\frac{\partial(L_i(\theta_j,\phi_k)\cos\theta_j)}{\partial d}\right]\\ &= \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}\left[\frac{\partial L_i(\theta_j,\phi_k)}{\partial d}\cos\theta_j\right.\\ &\qquad\left.+L_i(\theta_j,\phi_k)\frac{\partial\cos\theta_j}{\partial d}\right]\end{aligned}$$

As the term $\frac{\partial L_i(\theta_j,\phi_k)}{\partial d}$ is not computable (because of recursivity), the displacement in the neighborhood of point x is supposed to be small enough to assume that $\frac{\partial L_i(\theta_j,\phi_k)}{\partial d} = 0$. So, we obtain:

$$\frac{\partial I}{\partial d} \approx \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1} -L_i(\theta_j,\phi_k)\sin\theta_j\frac{\partial\theta_j}{\partial d} \quad (6)$$

This gradient is not mathematically exact, but it shows the perturbation degree of the region taken into account. It will be used during the computation of the criteria of an ILV, in Section 4.2.

The Render Cache

The *render cache* [WDP99] is an Image Based Rendering method [Kan99] that considers the picture as a radiance buffer.

All pictures are computed using a reprojection of the former picture, except the first one that must be fully computed. The *render cache* can work with any graphic engine that can compute individual pixels. In our work, a ray tracer has been used as in the original method. The *render cache* architecture is shown in

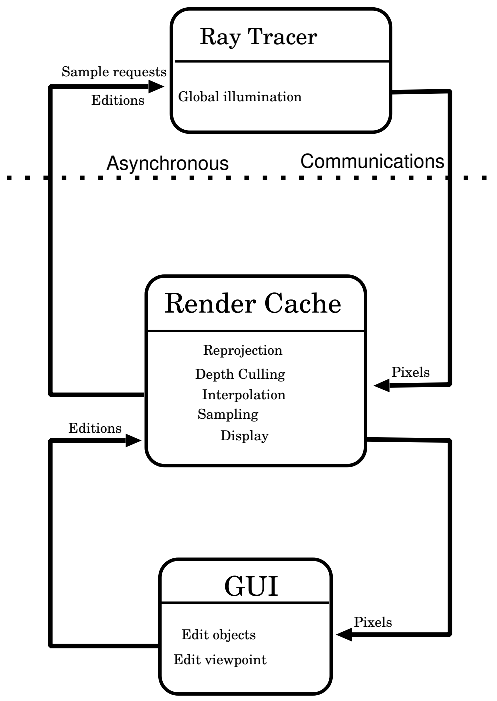

Figure 3: **Structure and interface of the render cache**

Figure 3, with asynchronous interactions between the *render cache* and a ray tracer.

Orders are sent to the interface if a change is performed. For example, if a user viewpoint modification occurs, it is sent from the interface (named GUI in Figure 3) to the *render cache* that uses the former picture to compute the new one.

First, the former picture is reprojected with the current viewpoint parameters, then a *depth culling* is performed to remove occluded or doubly mapped pixels. Simultaneously, a *priority image* is built. After a thresholding and an error diffusion, the priority image is used to choose the samples for which a request to the ray tracer will be sent. This image allows to recompute a limited number of pixels by choosing the most relevant ones. The *render cache* loops while waiting new editions; if there is none, the image is gradually fully computed. Interactivity is privileged in relation to quality, but with quality loss. Actually, the number of pixels recomputed between two displays depends on the power of the computer used; this number is usually low. The new image obtained by reprojection and calculation of a limited number of pixels can have artifacts. For further information, see Walter *et al.*'s papers [WDP99] [WDG02].

In this work, we have used the *render cache* approach in order to try to reach an interactive rate for high quality images. But instead of sending requests about individual pixels, we will use *indirect light vectors* (ILV) as a kind of hierarchical caching. If some criteria are fulfilled, then a simple vectorial interpolation will allow to compute several pixels in some area of the picture, either to a decrease of the time needed to compute the required pixels or to a recomputation of more pixels.

3 MULTI-DIRECTIONAL EXTENSION OF ILVS

In this section, we will propose a dynamic extension of *indirect light vectors*.

Motivations

Initially, a *light vector* was related to an observer location. In the dynamic case, where the viewpoint can move, an extension of the *light vector* is needed.

The idea is to make a double caching method: one for point locations and one for directions. Thus the computation of an ILV will be performed by doing a spatial interpolation and a directional one. This means that an initial set of predefined directions will be given, in addition to the initial seed of points introduced for *light vectors*.

KD-Tree$\times$Icosahedron Approach

The double caching method allows to split spatial and directional components in order to optimize storage and search.

The spatial component, which deals with point locations, is stored in a 3d-tree [Ben90].

The directional component is stored in a recursive icosahedron. This regular polyhedron is inscribed in the unit sphere and has 20 triangular faces. Each face stores the *indirect light vectors* going through the solid angle substained by the triangle. If a given threshold is reached by the number of VEIs stored on a face, then this face is replaced by four new faces, by creating the points corresponding to the middle of each edge on the sphere, as shown on Figure 4.

Results

To test this structure, a set of experiments has been performed. Search neighborhood queries have been done with several sets of directions and locations, first by fixing the number of directions, then by fixing the number of points and finally by varying both parameters.

The number of directions was tested up to 1000 and the number of points up to 100 000.

The execution time of the queries is nearly independent of the number of points; it takes between 10 and

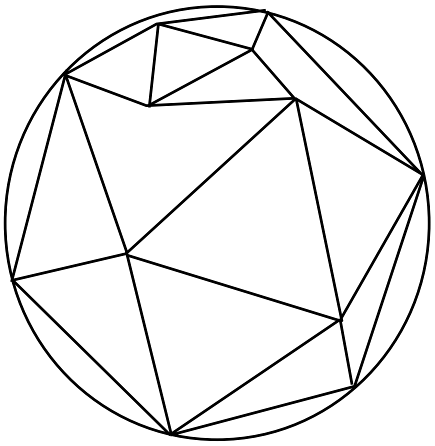

Figure 4: **Subdivision of a face of the icosahedron**

15 milliseconds per query. This phenomena can be explained by the small depth of the structure's levels.

4 VALIDITY CRITERIA

For both the *render cache* and the ILVs, a cache is used to avoid useless computations. Nevertheless, validity criteria are needed to know when the caches can be used without leading to inaccurate computations. In this section, we will first remind the heuristic criteria suggested for the *render cache* and for the ILVs. Then some criteria suitable for our new approach will be introduced, together with a new priority for a $5D$ ILV.

Criteria For The Render Cache

In Walter *et al.*'s original paper [WDP99], the main criterion used to determine the priority of a pixel is its age. Actually, the older the pixel is, the more it has the chance to be erroneous. An error diffusion algorithm [FS76] is used to propagate priorities and therefore to obtain a kind of *smoothing*.
The priority of a point missing in the cache is computed during the interpolation process and depends on the number of points re-projected in its neighborhood; the fewer points there are, the higher the priority is. Thus, reduced size zones can be focused to avoid insulated erroneous pixels, because the eye is really perceptible to high frequencies.

Criteria For ILVS

4.2.1 Spatial Criteria

To be valid at point A, an ILV computed at point B must satisfy the following conditions:

- $\|\overrightarrow{AB}\|$ is less than the average distance d_{avg_B} between B and the other objects of the scene (this is to avoid light escape by adapting the size of the validity area).
- The changed value of irradiance must be lower than a preset threshold I_B (this is to allow a certain control on the smoothing produced by the interpolation).
- The surface curvature must be lower than a fixed threshold S_{curve} (this is due to the fact that only the front surface part is used and the rear part can have a very different radiance from the front part).

If a ILV is valid, it is weighted according to the following formula:

$$w_B = \left(1 - \frac{\|\overrightarrow{AB}\|}{d_{avg_B}}\right) . \frac{1}{1 + \frac{\|\overrightarrow{AB}\|.\Delta E}{E_B}} . \frac{\cos(\overrightarrow{N_A}, \overrightarrow{N_B}) - S_{curv}}{1 - S_{curv}}$$

We may notice that the spatial weights will be normalized during interpolation.

4.2.2 Directional Criteria

While computing radiance at point x for a direction $\overrightarrow{\omega}$, the face of the recursive icosahedron associated to $\overrightarrow{\omega}$ is searched. The valid directions are chosen among those linked to this face.
An ILV computed for a direction $\overrightarrow{\sigma}$ is valid for a direction $\overrightarrow{\omega}$ if the difference between $\overrightarrow{\sigma}$ and $\overrightarrow{\omega}$ is lower than an arbitrary threshold S_{dir}. This threshold allows to define a directional weight $w_\sigma = \frac{\cos(\overrightarrow{\omega}, \overrightarrow{\sigma}) - S_{dir}}{1 - S_{dir}}$ for direction $\overrightarrow{\sigma}$. As for the spatial case, these weights will be normalized during interpolation.

4.2.3 Global Criteria

Let A be a point where a $5D$ ILV is needed for a direction $\overrightarrow{\omega}$. If a $5D$ ILV has already been computed at some point B for a direction $\overrightarrow{\sigma}$ and if both the spatial and directional criteria of the ILV are valid, then a global weight w is computed for this ILV. Its value is the product of the spatial weight associated to point B with the directional weight linked to direction $\overrightarrow{\sigma}$. Notice that $0 \leq w \leq 1$ and that the interpolation is more accurate if w is close to 1. If the number of valid ILVs is less than a given threshold, then a new ILV is computed at point A for direction $\overrightarrow{\omega}$ (and stored in the data structure). Otherwise, an ILV is computed by doing a normalized weighted interpolation of the valid ILVs found in the data structure.

Priority of a 5D ILV

We define a priority for a given pixel in the following way: during the interpolation process leading to a new ILV, a weight W is associated to this ILV by computing a normalized average of the weights of all the ILVs taken into account for the interpolation. Then, if W is not null, the priority associated to the given pixel is its priority for the *render cache* times $1/W$. This means that a low value of W increases the priority of the pixel for the ray tracer. If W is null, the priority is set to a high value to force the recomputation during the sampling process.

5 RESULTS

Our implementation has been done on a personal computer with an Athlon 1.2GHz, 512 Mo and no graphic acceleration. Images have been computed with a 512x512 resolution for the following scenes:
Scene A: Cornell Box 1. It is a room containing two parallelepipeds made of diffuse materials and a spherical light source.
Scene B: Cornell Box 2. It is the same scene as scene A, but one of the parallelepipeds is made of a transparent material.
Scene C: Sphere Flake level 1. It is a recursive union of 10 transparent spheres (taken from the NFF library[Hai87]) in a room with an area light source.
Scene D: Sphere Flake level 3. It is the same scene as scene C, but with a higher level of recursion. There are 820 spheres.

A global illumination ray tracer was used during our tests. We compare rendering times for the ray tracer alone (RT), the ray tracer with the *render cache* (RT + RC) and the ray tracer with our method (RC + 5D ILV). Only the view point is currently moving.
As it can be seen in Table 1, time savings of around 66%, 55%, 34%, 25% are obtained for the scnes A, B, C and respectively D. comparing to the use of the *render cache* alone.

There are around 21000 pixels (8% of the picture) to be recomputed by the *render cache*.

Artifacts related to the *render cache* remain present (see the black pixels in the top of the images, which correspond to pixels not seen in the former picture).

	RT	RT + RC	5DVEI + RC
Cornell Box 1			
1st picture	46”	46”	47”
next ones	46”	13”	4.4”
Cornell Box 2			
1st picture	3’20”	3’20”	3’21”
next ones	3’20”	11”	4.96”
Sphere Flake 1			
1st picture	8’6”	8’6”	8’7”
next ones	8’6”	16.2”	10.7”
Sphere Flake 2			
1st picture	20’29”	20’29”	20’31”
next ones	20’29”	20.3”	15.4”

Table 1: **Comparison of the methods**

6 CONCLUSION AND FURTHER DEVELOPMENTS

The method presented in this paper is based on the combination of the *render cache* (whose purpose is interactivity) and of *light vectors* (designed for photo-realistic rendering). The results shown in this paper are preliminary. They appear to be really promising, and we plan to make further work on this topic in order to extend and improve the method, as the objective of photo-realism in interactive time has not been reached yet.
In particular, we would intend to do more work on the following themes:

- extension of the method to caustic and direct light vectors;
- improvement of the definition of the priority used for the pixels;
- replacement of the *render cache* by the *shading cache*[TPWG02];
- presence of moving objects.

References

[BDT99] Kavita Bala, Julie Dorsey, and Seth Teller. Interactive ray traced scene editing using ray segment tree. In Dani Lischinski and Greg Ward Larson, editors, *Rendering Techniques ’99*, Eurographics, pages 31–44. Springer-Verlag Wien New York, 1999.

[Ben90] J. L. Bentley. K-d trees for semidynamic point sets. In ACM-SIGACT ACM-SIGGRAPH, editor, *Proceedings of the 6th Annual Symposium on Computational Geometry (SCG ’90)*, pages 187–197, Berkeley, CA, June 1990. ACM Press.

[Bli78] J. F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH ’78 Proceedings)*, volume 12, pages 286–292, August 1978.

[BN76] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, October 1976.

[CAS$^+$97] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. *Computer Graphics*, 31(Annual Conference Series):421–430, August 1997.

[DDM03] Cyrille Damez, Kirill Dmitriev, and Karol Myszkowski. State of the art in global illumination for interactive applications and high-quality animations. *Computer Graphics Forum*, 22(1):55–77, March 2003.

[FS76] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial grey scale. *Proc. Soc. Inf. Display*, 17:75–77, 1976.

[Hai87] Eric A. Haines. A proposal for standard graphics environments. *IEEE Computer Graphics and Applications*, 7(11):3–5, November 1987. also in SIGGRAPH '87, '88, '89 Introduction to Ray Tracing course notes, code available via FTP from princeton.edu:/pub/Graphics.

[JC98] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In Michael Cohen, editor, *Proceedings of SIGGRAPH 98*, Annual Conference Series, Addison Wesley, pages 311–320, 1998.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Workshop on Rendering*, pages 21–30, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.

[Kaj86] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986.

[Kan99] Sing Bing Kang. A survey of image-based rendering techniques. In *Videometrics VI Proceedings*, volume 3641, pages 2–16. SPIE, 1999.

[Lit97] Peter Litwinowicz. Processing images and video for an impressionist effect. *Computer Graphics*, 31(Annual Conference Series):407–414, August 1997.

[LKM01] Erik Lindholm, Mark J. Kilgard, and Henry Moreton. A user-programmable vertex engine. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, volume 35 of *Annual Conference Series*, pages 149–158. ACM Press / ACM SIGGRAPH, 2001.

[LS99] Gregory Ward Larson and Maryann Simmons. The Holodeck interactive ray cache. In ACM, editor, *SIGGRAPH 99. Proceedings of the 1999 SIGGRAPH annual conference: Conference abstracts and applications*, Computer Graphics, pages 246–246, New York, NY 10036, USA, 1999. ACM Press.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional Path Tracing. In H. P. Santo, editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, December 1993.

[Ost99] Victor Ostromoukhov. Digital facial engraving. In Alyn Rockwood, editor, *Siggraph 1999, Computer graphics Proceedings*, volume 33 of *Annual Conference Series*, pages 417–424. ACM Siggraph, Los Angeles, 1999.

[Per85] K. Perlin. An image synthesizer. In B. A. Barsky, editor, *SIGGRAPH '85 Proceedings*, volume 19, pages 287–296, July 1985.

[SA99] Mark Segal and Kurt Akeley. The opengl graphics system: A specification (revision 1.2.1), 1999.

[SP94] Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994.

[TDM99] Osama Tolba, Julie Dorsey, and Leonard McMillan. Sketching with projective 2D strokes. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Novel Output, pages 149–157, 1999.

[TPWG02] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics*, 21(3):537–546, July 2002.

[VG94] Eric Veach and Leonidas Guibas. Bidirectional Estimators for Light Transport. In *Fifth Eurographics Workshop on Rendering*, pages 147–162, Darmstadt, Germany, June 1994.

[WBS$^+$02] Ingo Wald, Carsten Benthin, Philipp Slusallek, Thomas Kollig, and Alexander Keller. Interactive global illumination using fast ray tracing. In Simon Gibson and Paul Debevec, editors, *Proceedings of the 13th Eurographics Workshop on Rendering (RENDERING TECHNIQUES-02)*, pages 15–24, Aire-la-Ville, Switzerland, June 26–28 2002. Eurographics Association.

[WDG02] Bruce Walter, George Drettakis, and Donald P. Greenberg. Enhancing and optimizing the render cache. In Simon Gibson and Paul Debevec, editors, *Proceedings of the 13th Eurographics Workshop on Rendering (RENDERING TECHNIQUES-02)*, pages 37–42, Aire-la-Ville, Switzerland, June 26–28 2002. Eurographics Association.

[WDP99] Bruce Walter, George Drettakis, and Steven Parker. Interactive rendering using render cache. In Dani Lischinski and Greg Ward Larson, editors, *Rendering Techniques '99*, Eurographics, pages 19–30. Springer-Verlag Wien New York, 1999.

[WH92] Gregory J. Ward and Paul Heckbert. Irradiance gradients. *Third Eurographics Workshop on Rendering*, pages 85–98, May 1992.

[Whi80] Turner Whitted. An improved illumination model for shaded display. In *Communications of the ACM*, volume 23, pages 349–349, 1980.

[WRC88] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A ray tracing solution for diffuse interreflection. *Computer Graphics*, 22(4):85–92, August 1988.

[ZP98] Jacques Zaninetti and Bernard Péroche. A vector model for global illumination in ray tracing. In Vaclav Skala, editor, *WSCG '98 (Sixth European Conference in Central Europe on Computer Graphics and Visualization)*, pages 448–455, Plzen, Czech Republic, 1998. University of West Bohemia.

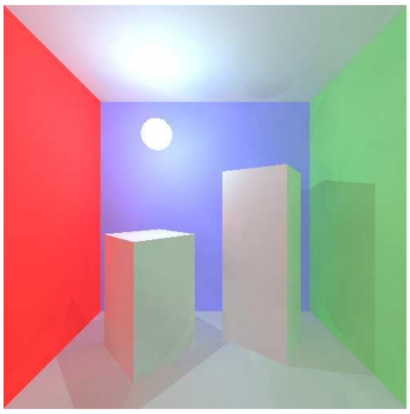
Figure 5: **Cornell Box 1 : ray tracing (RT)**

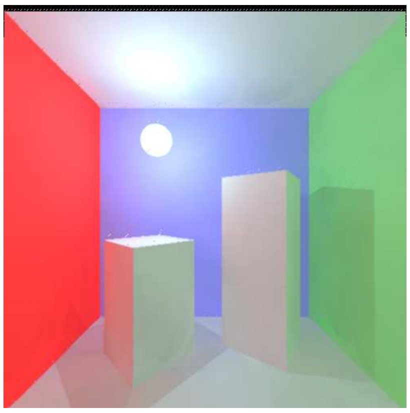
Figure 8: **Cornell Box 1 : 5D ILV + RC**

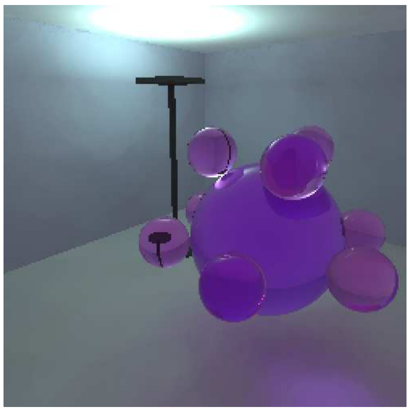
Figure 6: **Sphere Flake L1 : ray tracing (RT)**

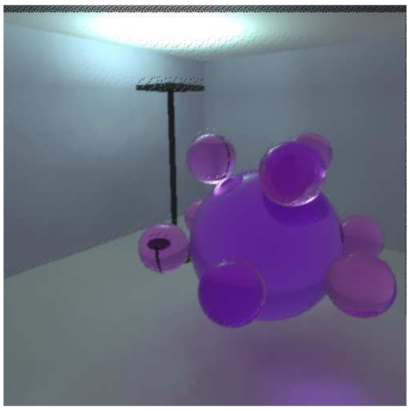
Figure 9: **Sphere Flake L1 : 5D ILV + RC**

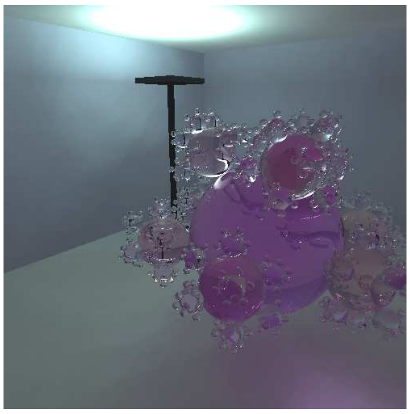
Figure 7: **Sphere Flake L3 : RT**

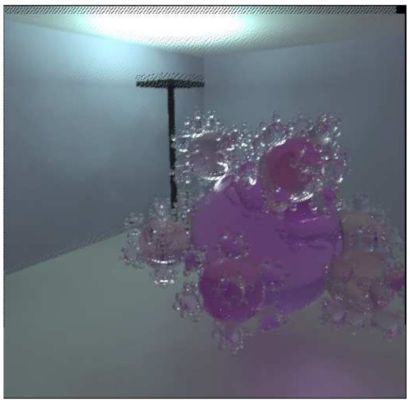
Figure 10: **Sphere Flake L3 : 5D ILV + RC**