Photo-Realistic Rendering – Recent Trends and Developments –

Philipp Slusallek†
Computer Graphics Group
University of Erlangen, Germany

Abstract

Since the beginning of computer graphics, one of the primary goals has been to create convincingly realistic images of three-dimensional environments that would be impossible to distinguish from photographs of the real scene. The goal to create photo-realistic images has lead to the development of completely new software techniques for dealing with the inherent geometric and optical complexity of real world scenes. This report gives an overview of advanced algorithms for photo-realistic rendering and discusses recent trends and developments.

1. Introduction

Photo-realistic rendering has a strong influence on many areas of computer graphics that seems to become even larger with 3D techniques entering the consumer market. Although the advances in photo-realistic rendering are no longer as spectacular and visible to the public as with early ray-tracing or radiosity images, it is a very active field of computer graphics research with substantial new developments. This report tries to highlight some of the new developments and point out areas of future research.

Some of the important application areas of photo-realistic rendering include

3D Prototypes: Virtual mockups are used more and more to reduce the cost of manufacturing. While standard hardware generated images are sufficient for a simple geometric analysis, increased realism is required to analyze features such as the overall design, the finish of surfaces, and the visual impressions of the final product for marketing.

Lighting and Luminaire Design: The design of new luminairs is a complex and costly process that is often tightly coupled with the lighting design of architectural buildings. In particular for the latter case, it is usually not feasible to test design decisions at a model in order to avoid costly mistakes. The simulation of the distribution of light in luminairs, rooms, and buildings allows to increase the quality as well as reduce the cost and time to market.

Virtual Reality: The main goal of virtual reality is the immersion of a user in a virtual environment. This immersion depends to a large extent on the generation of adequate optical stimuli. The real-time requirements of virtual reality still impose severe limitations on the realism that can be achieved, but precomputed illumination and visual detail are an important part of convincing environments.

Computer Augmented Reality: This extended form of virtual reality combines virtual environments with real world (video) data. This poses the additional problem of simulating optical and illumination effects from real objects on objects in the virtual environment and vice-versa.

Games: Today, convincing realism is an important factor for the success of a computer game title. More and more games use techniques from realistic image synthesis such as precomputed global illumination, shadows maps, as well as advanced mirror and refraction effects to increase the illusion of reality.

3D on the Web: With techniques like VRML that allow for distributed 3D environments on the web, the demand for realistic renderings of environments in steadily increasing. Casual users are much more critical on the quality of computer presentations. Online 3D catalogues and shopping malls require much more realism than what is offered today.

Ubiquitous 3D: 3D technology is just entering the consumer market now and many new application will be certainly be developed. Many of them will benefit greatly from realistic image generation techniques.

† Email: Philipp.Slusallek@informatik.uni-erlangen.de

Software for photo-realistic rendering can be thought of consisting of three main components: *scene description*, *lighting computation*, and *image generation*. The first component deals with the problem of describing 3D environments with the necessary level of detail and physical accuracy that allows for generating realistic images. This component is also concerned with the interface that a rendering system offers for importing scene description from external modeling programs.

A key component of convincingly realistic presentations is the simulation of the rich details that are due to lighting effects, such as soft shadows and correct interreflections between objects. The proper simulation of these lighting effects can avoid the "computer" or "artificial" touch of computer generated images. Once the distribution of light in a scene has been computed, we can take a snapshot of the light field with a virtual camera and generate an image from it. This last component of photo-realistic rendering deals with problems like hidden surface removal, shading, motion-blur, and anti-aliasing.

Each of these components has its own challenges some of which will be covered in the following report. Special emphasis will be put on physically correct lighting computations. Many of the subtle lighting effects can and have been simulated manually by graphic designers and animation experts. Particularly in the film and animation industry such experts have created remarkably realistic images and film sequences often without even knowing the physics behind it. However, as 3D graphics spreads into everyday life, those experts are no longer available. This increases the demand for automatic and correct simulation of optical and lighting effects, which stimulated much of the research presented here.

The key issue in this trend is the ability to correctly simulate the physics of light, the interaction of light with matter, and the transport of light in an environment. A full and accurate simulation is only required by a few application areas, such as lighting and interior design, as well as illumination engineering. But the same techniques can also be used to compute a coarse approximation to the correct illumination. Such an automatically computed approximation will be sufficiently realistic for many applications. Adaptive controlling the accuracy of computations allows for trading quality for speed in applications. This adaptive control also requires that lighting systems can switch between rendering algorithms that offer different quality/speed trade-offs.

In recent years, photo-realistic rendering has seen a dramatic development of faster, more accurate, and robust solutions techniques. These techniques include Finite Element and Monte-Carlo solvers for integral equations, the adoption of Wavelets, multi-grid, and multi-resolution techniques. New algorithms exploit a number of advanced geometric algorithms and data structures for efficiently solving the problem of transfer and representation of light in complex environments.

The speed and the robustness of the available algorithms are now good enough for being used in industrial strength applications. This is supported by the fact that most companies offering animation systems are currently integrating advanced lighting simulation into their products (e.g. Alias|Wavefront, SoftImage). However, there is still an large gap between recent research results and the techniques available in todays commercial systems. One reason for this gap is that system integration and other practical aspects of new algorithms often receive little attention. These aspects will very likely become much more important in the future as new techniques will be used in a broader scope.

The methods and techniques that have been developed for realistic lighting computations are not restricted to generating realistic images only. These methods can easily be modified to work with other forms of radiation (like heat, radar, radio, etc.). Furthermore, many of the advanced techniques used in realistic lighting computations like efficient hierarchical methods, clustering, and variance reduction techniques can equally be applied to other engineering applications where they are less widely known.

Lets start with a brief review of the underlying physics and mathematics, before we discuss some of the new techniques for computing realistic images.

2. Basic Techniques

The fundamental quantity in physically-based rendering is *radiance*. The quantity radiance $L(y, \vec{\omega}, \nu)$ is the power radiated from a point y in a certain direction $\vec{\omega}$ per unit projected area perpendicular to that direction and per unit solid angle. The units of radiance are $[W\ m^{-2}\ sr^{-1}]$. The dependency on the frequency of the radiation is usually ignored in lighting computations and the equations are solved for each frequency band separately.

The distribution of light in a scene is completely described by the radiance distribution in space. All other quantities can be derived from it. Radiance is invariant along its direction of propagation, provided there are no absorption, emission, or scattering effects by participating media (like smoke or fog). Additionally, the amount of light received at the retina of the human eye is proportional to the radiance emitted by the viewed surface. These properties make radiance the appropriate quantity for illumination computations in a rendering system.

Light is emitted at light sources and interacts with other surfaces in the environment, being absorbed, reflected, or transmitted. The propagation of light in the absence of participating media is fully described by the radiance equa-

tion [CW93]

$$L(y,\vec{\omega}) = L_e(y,\vec{\omega}) + \int_\Omega f_{rt}(\vec{\omega}_i, y, \vec{\omega}) L_i(y,\vec{\omega}_i) |\cos\theta_y \, d\omega_i|. \tag{1}$$

It states that the radiance reflected at a point y in direction $\vec{\omega}$ is the sum of the self emitted radiance L_e and the reflected radiance. The latter is computed by integrating over the fraction of incident light from all directions that is scattered into the outgoing direction ω. The scattering characteristics of a surface is described by the *bidirectional reflection distribution function* f_{rt} (BRDF). The incident light L_i itself contains contributions from light reflected off other surfaces, thus coupling it to outgoing radiance L at all other visible point in the environment. It is this global nature of the radiance equation and the embedded problem of visibility together with complex scene arrangements that make it so difficult to compute true photo-realistic images.

We are often not interested in the full *directional* distribution of light at a point. In this case, we can eliminate the directional parameter and switch to *radiosity* for describing the light distribution. This results in the radiosity equation

$$B(y) = B_e(y) + \frac{\rho(y)}{\pi} \int_{x \in S} B(x) G(x,y) V(x,y), \tag{2}$$

where we changed the integration domain to all surfaces in the environment. This explicitly adds the visibility function V and the geometric form factor $G(x,y) = \cos\theta_x \cos\theta_y / |x-y|^2$ to the integrand [CW93].

2.1. Overview of Lighting Techniques

Computing the illumination in some environment requires the solution of the radiance or radiosity equations (1) and (2). As Fredholm equations of the second kind [Atk76], they cannot be solved analytically, except for trivial cases. Instead, numerical algorithms for computing a solution must be used, all of which are derived from two basic numerical approaches: *finite element* and *Monte-Carlo* (or point sampling) techniques.

Here, we present a the basic techniques upon we will build in later section of the report.

2.1.1. Monte-Carlo Approach

The idea of Monte-Carlo integration is, that a solution of some integral $I = \int f(x)dx$ can be approximated by taking a large number of point samples of the integrand

$$I = \int_\Omega f(x)dx = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \frac{f(\xi_i)}{p(\xi_i)}, \tag{3}$$

where the stochastic samples ξ_i are distributed according to a *probability density function* (i.e. $p(x) \geq 0, \int p(x) = 1$) [KW86]. Due to the Law of Large Numbers, the variance of the solution converges to zero for large numbers of samples. In a concrete computation the number of samples is limited, thus resulting in some remaining variance in the solution, which becomes visible as "noise" in the image (see Figure 1).

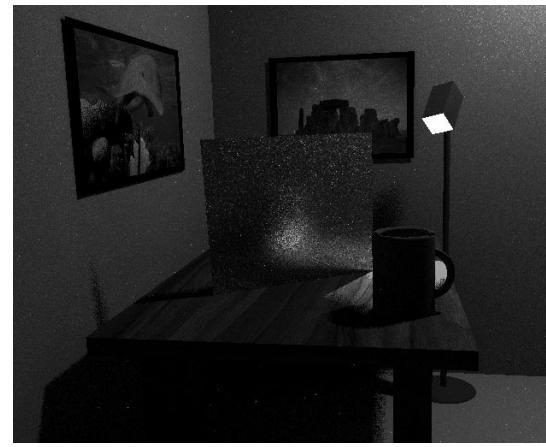


Figure 1: *Example Monte-Carlo image. One can clearly see the noise that is still present in the solution.*

In the context of image synthesis the Monte-Carlo method can be applied in several ways. Still the most common way is to integrate incoming illumination by recursively tracing rays (trees or single paths) from the camera into the environment [Kaj86, WR88]. Another technique is to start at the light source and distribute light energy in the form of particles or photons into the environment [Hec90, Col94, SWH+95, Jen96]. When the photons hit a surface information about these hits are usually stored in some data structure, which is later used to compute illumination. Note, that the data structures usually introduce some form of bias and finite element aspect into the algorithm. A third form of Monte-Carlo algorithms does not use an intermediate data structure and thus can avoid the introduction of bias by the two step process. The bidirectional technique [VG94, VG95] directly computes paths of photon from light sources through the environment to the virtual camera.

The Monte-Carlo technique is particularly well-suited for solving high-dimensional integral equations with low coherence that cannot be well approximated by smooth functions used in finite element methods. Since only point samples are used and each sample is computed independently of any other, pure Monte-Carlo algorithms are simple implement and have little to no memory requirements.

The Monte-Carlo technique requires a large number of samples to reduce the variance of the solution. The error of the solution is reduced only proportional to $1/\sqrt{N}$, which is known as the *diminishing return* of the Monte-Carlo technique, i.e. the number of samples must be quadrupled to reduce the error by one half.

The formulation of the Monte-Carlo technique is fairly simple and the same is true for a direct implementation.

However, the Monte-Carlo technique is known for its high computational cost, due to the large number of samples that have to be taken. This cost can be reduced by choosing suitable probability density functions for the specific problem – a technique known as *importance sampling*.

2.1.2. Finite Element Approach

In the finite element approach, the illumination is described as a smooth function representing outgoing radiance or radiosity over the surfaces in the scene. This technique takes advantage of the fact that reflected radiance and radiosity are usually piecewise smooth. Finite element techniques are based on a subdivision of the environment into small surface patches and compute a functional representation of the illumination on each of them.

With this approach finite element algorithms try to exploit coherence in the light distribution in the environment. Finite element techniques are in a sense at the opposite side of the solution spectrum compared to Monte-Carlo algorithms: computation at points versus surfaces, deterministic versus stochastic computations, and explicit versus implicit use of coherence. Unfortunately, approximating the light distribution by smooth functions results in biased solutions — a problem that true Monte-Carlo methods can avoid completely. However, this bias is generally not a problem for most applications.

After representing the distribution of light in the finite-dimensional function space, we can use the standard Galerkin technique to reduce the radiance or radiosity equation to a linear system describing the exchange of light between surface patches. Although many standard techniques are available for solving linear systems, only very few of them can actually be used. Compared other disciplines the linear system used for lighting computations have some unique features: The computation of the coefficients is very costly and requires knowledge about the scene as a whole. Depending on the complexity of the scene, the full matrix would become extremely large. It is thus mandatory to avoid computing or even storing coefficients if at all possible. This requirement has lead to a number of unique approaches for representing and solving the linear system that will be described in Section 3 in more detail.

2.1.3. Hybrid Approaches

There are many hybrid techniques that use a combination of finite element techniques and the stochastic Monte-Carlo approach. They usually handle certain aspects of the integral equation (1) as special cases and try to exploit some form of coherence. This often results in a better approximation to the exact solution or in faster computations for a certain classes of environments.

A common approach is to separate the indirect illumination from the illumination received directly from light sources. In a typical scene the irradiance received indirectly varies much slower and smoother than the direct illumination, where shadow boundaries are a major source of discontinuities. As a result, the indirect illumination can be approximated by fewer samples or fewer basis functions [War94, Sch94, Zat93].

Another approach that is often used, is to separate the BRDF into a diffuse component and a specular or mirror component. The smoother diffuse reflection component can then be approximated using a different technique than the highly localized mirror reflection component [WCG87, SP89, CRMT91, BT92].

3. Hierarchical Techniques for Illumination Computation

After this brief review of the available techniques for computing illumination in a virtual environment, this section concentrates on the use of hierarchical finite element techniques. The use of hierarchies has made finite element techniques practical and applicable to a large variety of virtual environments. It allows for scaling the algorithms with scene complexity, available computation time, and required accuracy.

As described in the previous section, finite element techniques offers a deterministic approach to compute the illumination in an virtual environment. The main issues are the time and memory complexity of the algorithm.

In contrast to pure Monte-Carlo techniques, which require little to no memory for storing intermediate results, the finite element algorithms must store the representation of illumination that is computed for all surfaces. For complex scenes this can become a considerable problems. Furthermore, the matrix of the linear system must be represented in some form during the solution process. Given a number of patches n in the environment an early finite element algorithm computed the light transport between any pair of them, thus resulting in quadratic complexity $O(n^2)$ in the number of patches.

The introduction of hierarchical algorithms changed the situation for finite element algorithms dramatically. Hierarchical techniques allow for automatically selecting the level of detail at which light should be transported between patches. Together with clustering techniques, described in the next section, this allows for algorithms that have only *linear* complexity $O(n)$ in the number of input surfaces.

Lets take a closer look on how this can be achieved.

3.1. The Basic Hierarchical Algorithm

Finite element algorithms require the discretisation of the environment into patches. For a given functional representation of illumination on a patch, the accuracy of the computation is determined by the discretisation of the environment.

Many small patches allow for a more detailed and accurate computation. What is required is an adaptive discretisation into smaller patches where it improves accuracy, for example along shadow boundaries or in other areas where illumination changes rapidly.

The hierarchical approach allows for exactly this kind of adaptive computation. Instead of discretizing the environment into a fine mesh of patches before starting the computation, the algorithms refines the initial coarse set of surfaces during the course of the computation wherever this becomes necessary. For recursive refinement, patches are usually represented in the form of *quadtrees* on triangles, quadrilaterals, or other parametric surfaces (see Figure 2).

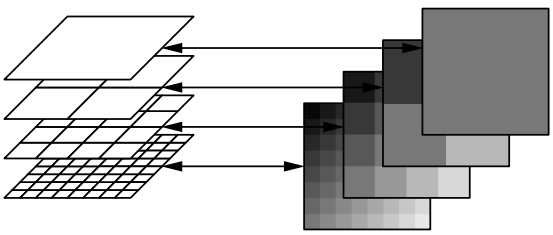


Figure 2: *Representation of illumination at different levels of a hierarchy. Patches in higher levels represent the average illumination of their children.*

When considering the transport of light between surfaces, the algorithm chooses the appropriate level of discretisation that is required for meeting a user selected error criterion. For two nearby patches that exchange a large amount of radiation, which may results in large illumination gradients on the receiver, a detailed subdivision of either the sender or the receiver or both is be required. On the other hand, if the two surfaces are far apart or exchange only very little energy, it might be sufficient to compute the exchange between large patches in upper levels of the quadtrees [HSA91] (see Figure 3).

During the computation the hierarchical algorithms automatically select the coarsest level at which the accuracy requirements can be met. The same surface may receive and send illumination at various levels of the discretisation hierarchy. It is therefore necessary to keep the different levels consistent. If energy is received at some level, this energy must also be distributed or *pushed* down to the children of the receiving patch. On the other hand, the energy must also be propagated or *pulled* to upper levels in the hierarchy. Starting at the leaf nodes in the hierarchy, parent patches then compute their illumination by summing up the energy of their children. The whole process can be described as a single *push/pull* procedure, that keeps the different representations of illumination consistent [HSA91].

3.1.1. Representing and Solving the Linear System

A matrix is certainly not an appropriate way to represent the exchange coefficients of the linear system. The complex interactions between different levels in the hierarchy can best be represented by storing the exchange coefficients explicitly as *links* between the two patches. The flexible structure of this representation allows for simple additions and deletion of the links. These dynamic changes would be difficult to implement in a matrix structure.

The solution of the linear system now proceeds by using an iterative solver for linear systems. The most commonly used iterative solvers are the Gauss-Seidel algorithm, which considers each surface in turn and *gathers* light via all links that connect to the hierarchy of patches on the receiver. Similarly, the Southwell algorithms can be used, which *shoots* energy via these links to other surfaces. However, due to the fact that the adaptively created links all transport similar amounts of energy, the original advantage of this solver is lost [CCWG88, GCS94, HSA91].

Before energy is actually transported via a link, the iterative solver must check the accuracy of the link. Due to the changes in illumination, as link may become inaccurate and may need to be refined during the course of the iterations. The refinement of links is guided by a *refiner*, that is discussed in more detail in Section 3.1.2 and Section 5.

Many more iterative solvers are known for linear systems. Most of these algorithms are difficult to adapt to the link representation of the exchange coefficients. Furthermore, the adaptive creation of links together with the computation of the correct exchange coefficient by far dominates the solution time compared to the actual transport of energy via the links. Thus, somewhat faster iterative solvers would not have much effect and would complicate the hierarchical algorithm compared to the simple Gauss-Seidel and Southwell solvers.

3.1.2. Refinement Criteria

The crucial part of any hierarchical algorithms is the *refiner* (often also called oracle). It determines if an exchange coefficient between two patches can adequately be approximated given a user selected threshold. Otherwise the patches need to be subdivided. The particular design of the refiner determines to a very large extent the accuracy, memory requirements, and running time of a hierarchical algorithm. As a consequence, it is not meaningful to compare hierarchical algorithms without specifying the particular design of the refiner and the chosen parameters.

For hierarchical radiosity computations several different refiners have been proposed. The most important refiners use the exchange coefficient itself (F-refinement) [HSA91], the resulting radiance on the sender (BF-refinement), or the reflected energy (BFA-refinement) for estimating the error that is committed by transporting light across a link. The BFA-refiner seems to work best for all kinds of environments and is used in most implementations.

These refiners are based on the observation that the error of a link can be bounded above by a constant factor times the

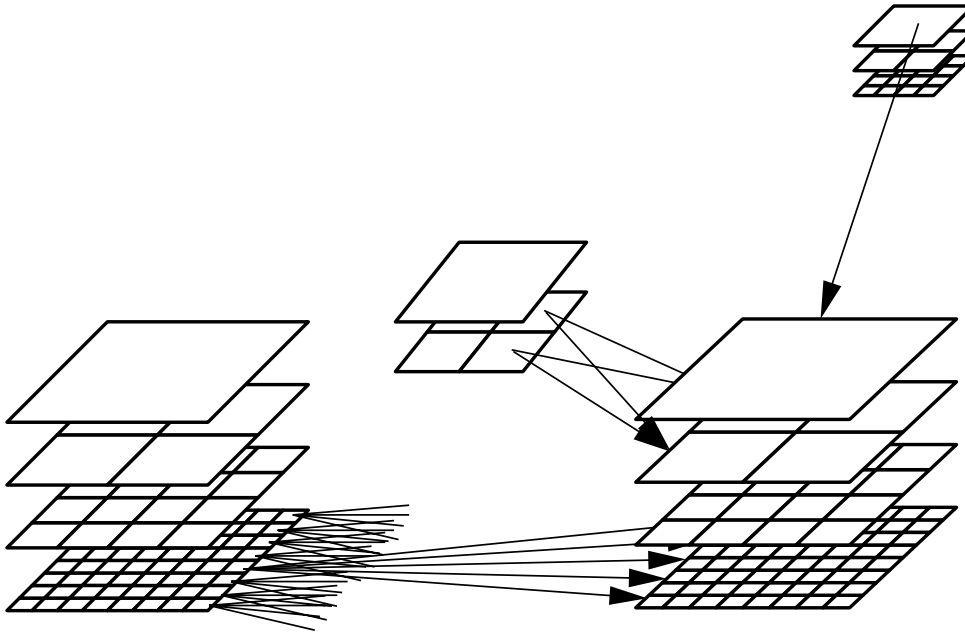


Figure 3: *Light is transported via links between various levels in two hierarchies depending on the configuration of the sending and receiving patch. A refinement procedure selects the coarsest level at which the transport can be computed within a user selected error threshold.*

exchange coefficient of the link. Weighting this error with the transported energy and the area of the receiver results in the last two refiners. The BFA-refiner is very well behaved and has the added benefit of automatically bounding the minimal size of patches, which is usually an extra parameter for the other two refiners. Several modified versions of these refiners have also been proposed (e.g. HSD94).

3.1.3. Wavelets and Higher Order Function Spaces

Most of the hierarchical algorithms use simple constant basis functions for representing illumination on a patch, requiring only a single coefficient for this representation. The Galerkin construction of finite element algorithms also works for higher order basis functions CW93. The formal mathematical basis for a hierarchical construction is the Wavelet theory that described hierarchical function spaces Dau92. Hierarchical wavelet algorithms have been used in various ways in rendering, e.g. for radiosity computations GSCH93, SGCH93, or for full radiance computations CSSD94, Sch94, PB94.

Higher order functional representations allow for using less but larger patches, because these functions have a higher approximation power. This can result in reduced subdivision and consequently in less patches and links. However, higher order functions are also more costly in terms of computing and storing exchange coefficients. For instance bi-quadratic basis functions require storing 81 exchange coefficients per link (between each pair of the nine coefficients on each patch), instead of storing only a single coefficient per link as with constant basis functions (see Figure 4).

This drastic increase in storage requirements is the main disadvantage of higher order wavelet algorithms, which becomes apparent at shadow boundaries. Even the larger approximation power of bi-quadratic or bi-cubic functions cannot adequately represent these features and still requires subdivision resulting in additional links that are much more expensive than in the constant case. In these situations the larger approximation power cannot compensate the increased memory requirements for storing links. One possibility to avoid subdivision in these situations is by approximately representing visibility separately using a *shadow mask* SSSS95, Zat93.

4. Clustering

The main problem with hierarchical algorithms as described in the previous section is that the algorithms still have quadratic complexity in the number of input surfaces. The algorithms must still consider each pair of input surfaces even if the energy transfer between them is negligible. For the case of scenes with many small surfaces instead of a few large surfaces, this *initial linking* can easily dominate the total computations time. The problem is that the benefits of linear complexity of the hierarchical algorithms are only exploited during adaptive refinement. A simple solution would be to avoid initial linking wherever possible HSD94.

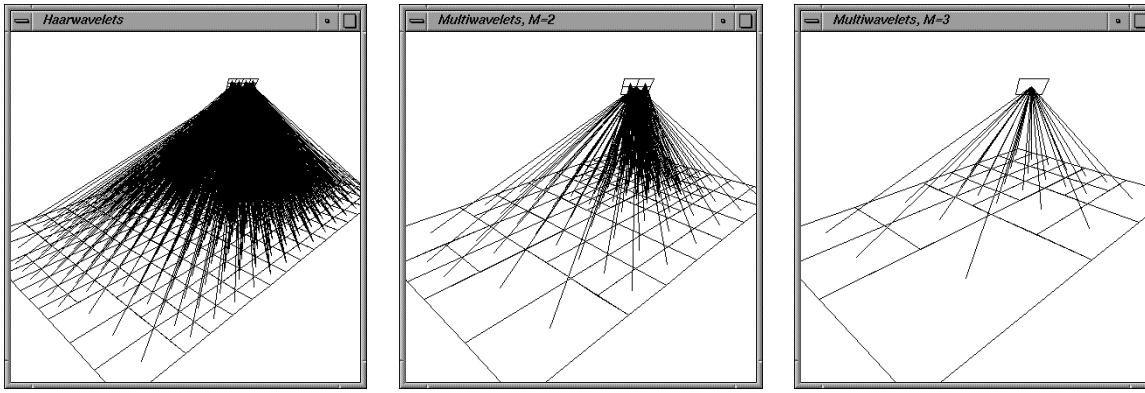


Figure 4: *Links between two patches generated by using Multi-Wavelets of various orders. Constant: 1 coefficient per link (left), Bi-linear: 16 coefficients (center), bi-quadratic: 81 coefficients (right).*

Another approach extends the hierarchical approach. In order to use the benefits of adaptive computation of hierarchical algorithms, we must extend the hierarchy that exists below the level of a surface to a full hierarchy covering the whole scene. This an be done by grouping related surfaces into *clusters* (see Figure 5). These clusters can in turn be grouped hierarchically into larger clusters until the whole scene is contained in a single *root cluster* [Sil95, SAG94].

Although "hand made" cluster hierarchies often perform slightly better [SSS97d], automatic clustering algorithms are preferred, because they are applicable to general and complex scenes. Here, hierarchical bounding volume algorithms seem to give best results [SD95, SSS97d] compared to octrees or k-d trees. The latter two often construct clusters that consist of unrelated patches that can result in large lighting error.

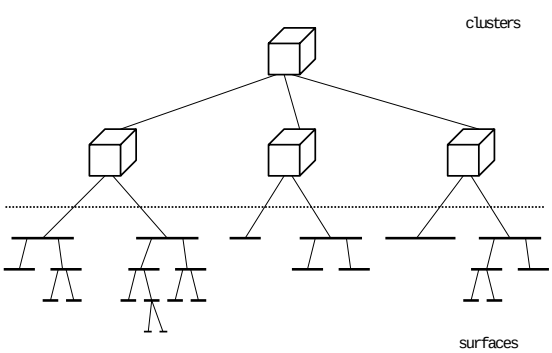


Figure 5: *Extending the hierarchical structure above the level of input patches by clustering. This creates a single hierarchy for the complete scene.*

The important feature of a cluster is that it can receive and emit energy for all of its contents. This allows to exchange light between a large group of surfaces in a single operation. An important difference to the use of planar surfaces is that a cluster may exchange energy with itself. Consequently, the processing starts by refining the first link in the scene, which is a self link from the root cluster to itself. Since the exchange of light at this level would generally introduce too much error, the refiner would replace this link, open up the cluster, and instead link its child clusters or surfaces before recursing.

This clustering strategy allows for quickly computing coarse approximations in a short amount of time. By choosing a large error threshold only a few large cluster would ever be considered. Although this would result in a coarse approximation, it may already allow a first estimate of the lighting effect in a scene (see Figure 6). By adjusting this threshold the user can continuously change the level of accuracy and speed of the computations. This is a valuable feature for many interactive applications.

Even if all surfaces in a cluster are purely diffuse, the cluster will receive and reflect light with a directionally dependent characteristics. However, in the simplest case, the light exchange of a cluster and its environment is modeled as that of a isotropic volume that emits and scatters light in all directions evenly. Although this is an extremely crude approximation, it already works quite well in most cases. In order to avoid some of the resulting artifacts, light energy received by a cluster can immediately be propagated to all surfaces in its hierarchy. This allows to use the direction of the incident light while it is still known and computing its contribution to each surface separately [SAG94]. Note, that this is different and still much more efficient than computing the form factor to each surface separately.

The cluster algorithms can also be extended for using non-isotropic interactions with its environment by storing the directional distribution of emitted energy with each cluster [SDS95].

Figure 6: *The model of a station consisting of approximately 32000 patches and 24 light sources. The lighting simulation based on clusters was computed in less than 10 minutes on a MIPS R8K CPU. Only 0.02 percent of the potential links where actually created (left). The image on the right shows the clustering hierarchy that was created.*

4.1. Visibility

Clusters cannot only be used to receive and emit energy, they are also a valuable help for visibility computations. The tree of clusters can be used for computing visibility hierarchically. If a visibility ray traverses a cluster, a special visibility refiner decides whether the cluster can be approximated by a semi-transparent volume or whether it must be opened and the ray intersected with its children. The properties of a cluster that are used for this decision and can directly be computed from its contents [SD95, Sil95].

Cluster are a natural extension of the previous hierarchical algorithms in rendering. By extending the hierarchy to the whole scene, the benefits of the hierarchical approach can optimally be exploited. Unfortunately, correct handling of clusters is more difficult due to their non-isotropic reflection characteristics. This area offers many opportunities for further research.

5. Refiners Based on Bounded Transport Computations

As already noted in Section 3.1.2, the refiner is the crucial part of any hierarchical algorithm. Optimizing the refiner can have large effects on the quality, speed, and memory requirements of a computation. Early refiners for lighting simulations used simple criteria and point sampling to estimate light transport. By accurately bounding the input parameters of the computation, it is possible to obtain tight but conservative bounds on the amount of light transport over a receiving patch. These bounds offer a new opportunity for guiding the refinement procedure for normal as well as for hierarchical algorithms [SSS97b, SSS97a].

One problem with the refiners as presented in Section 3.1.2, is that they only estimate an upper bound of some transport quantity (i.e. the form factor or radiosity). The lower bound is always assumed to be zero. This approach ignores much of the information that is available or can easily be computed for a link [SAG94]. Such information includes geometric quantities like the extent of sender and receiver, their distance, and the variation of normal vectors across each surface. The variation of normals vectors is of particular importance for the ability to treat curved surfaces efficiently.

A second problem of these refiners is the way the transport quantities are computed. These refiners generally use some form of point sampling for estimating them. They often use only a single sample for computing the exchange coefficient at the “center” of a patch, e.g. using the disc to point form factor approximation. The transport quantity may then be further modulated by estimating visibility of the link with sample rays that are sent between the two patches in question.

The sampling problem already becomes apparent for a small light source located directly in front of a large patch (see Figure 7). Although there might be a considerable transport of light to and from a small spot on the large patch, any point sampling strategy across the large patch is likely to miss this hot-spot. Another example for the sampling problem becomes apparent if we consider curved surfaces (see Figure 8). In this case, the geometric parameters (locations, distances, and normals), on which the computation of the exchange coefficients is based, vary strongly over the surface, again making any point sampling strategy highly unreliable.

A solution is to compute upper and lower bounds for any of the parameters that are used as input to the computation of the exchange coefficients. If tight conservative bound are computed for the input parameters, the resulting exchange coefficients can also be bounded by applying interval or affine arithmetic [Moo66, CS93, Sny92]. The location of points and

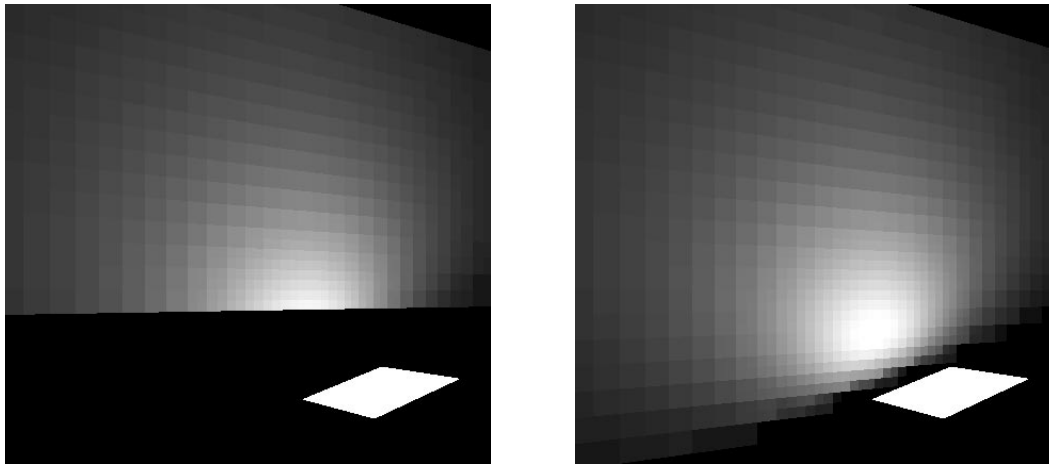


Figure 7: *Sampling problem for a small light source located directly in front large patch causing a small but bright spot: Any point sampling strategy is likely to miss the spot and omit important light contributions during subdivision (left). Correctly bounding the transport quantity allows for locating hot spots on a patch (right).*

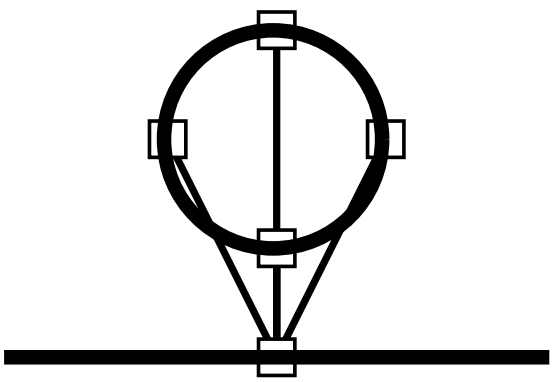


Figure 8: *Sampling problems for light transport between a planar patch and a curved surfaces (2D slice): The estimation of light transport is very unreliable, depending on the actual sampling location on the curved surface and the planar patch.*

their normals can efficiently be bound with the cone of normals technique [SA93] and bounding boxes. Based on this data, bounds on the set of connecting rays, cosines factors, distances, and solid angles can be computed. Bounds can also be computed for visibility between patches. Finally, these terms are combined for obtaining a conservatively bounded estimate of the exchange coefficient [SSS97c].

During the computation, triple values <min, estimate, max> are maintained that denote an upper and lower bound as well as an estimate of the correct value. These values can then be used by the new refinement strategy for deciding where to refine links. Links are refined, when the difference between upper and lower bound becomes to large. The new refiner also has enough information for deciding whether to refine the sender or the receiver [SSS97c]. The decision is based on their relative influence on the error. This is an important benefit over previous refiners, where no such information is available.

Figure 9 shows the effect of this new refinement strategy for the light exchange between a planar patch and a cylinder. The cylinder is refined less in the center, where most of energy is transfered, but the illumination is mostly constant. This is where the older refiners would concentrate their effort. Instead the new refiner shifts the focus to areas with higher illumination gradients as indicated by larger differences between upper and lower bounds. Thus, it refines near the silhouette edges of the cylinder where there is still a large amount of received light and where both the cosine factor at the cylinder and the distance varies considerably.

The interface to geometric objects for obtaining the required geometric information can be kept very simple. This allows a wide variety of objects to implement it [SSS97b]. As shown in [SSS97a] the same strategy can be extended to work equally well for environment with clusters. Although it is more difficult to compute bounds for clusters, the algorithms and refiners still benefit considerably for having accurate bounds available. The bounds can also be used to estimate the self illumination of non-convex curved surfaces and clusters [SSS97b].

6. Composite Lighting Simulations

As described previously, many different lighting simulation techniques have been developed for computing the distribution of light in a virtual environment. However, many of the

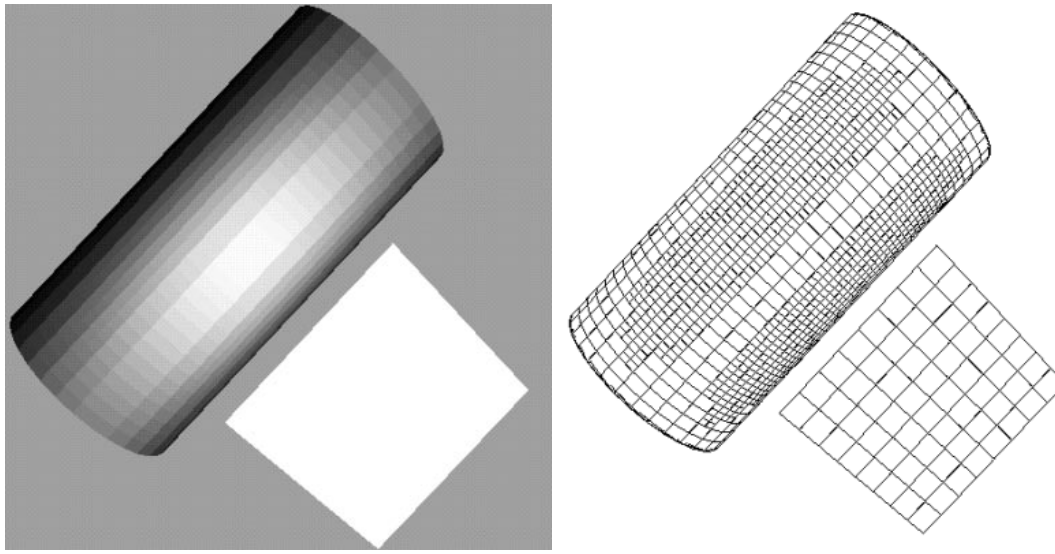


Figure 9: *A square lighting a cylinder. Note that the subdivision on the cylinder at the center coarser the in the neighborhood. This is due to the slower varying cosine term in the center.*

techniques available make a number of simplifying assumptions and thereby restrict the lighting effects they can simulate. *Local and direct Lighting* assume only direct illumination from the light sources [Whi79]. *Radiosity* [CW93] generally restricts the environment to diffusely reflecting piecewise planar surfaces, while *Monte-Carlo ray tracing techniques* are best for highly specular surfaces but have difficulties simulating diffuse illumination efficiently [Kaj86]. *Hybrid techniques* or *multi-pass methods* [WCG87, SP89, Hec90, CRMT91, Jen96] try to combine the advantages of each of these techniques to yield better algorithms. So far, they have been limited to static combinations of the above techniques.

In addition to the restricted domain of each simulation algorithm, the computation of illumination has generally been very inflexible. Only a restricted set of algorithms (each with a small set of parameters) was all that was available to a user. A configured algorithm then simulated all of the light transfer in the complete scene. No control is available for restricting the simulation to a subset of the scene or for adding special effects. Furthermore, costly algorithms that compute full radiance simulations are available but due to their high computational complexity could never be used in non-trivial scenes. Still, many scenes contain small regions that could greatly benefit if this kind of a full radiance simulation is applied locally to a limited set of surfaces, e.g. the specular reflection of some objects in a glossy surface or the caustics of another object.

The concept of Lighting Networks [SSH+96] adds the necessary flexibility to global lighting computations. Its allows to combine different lighting algorithms and to restrict them to subsets of the scene. In this respect, it generalize these multi-pass techniques and also extend them in several directions. In the Lighting Network approach each lighting algorithm is considered a *Lighting Operator* taking illumination information as input and generating new illumination information as output after having computed a part of the global lighting simulation in the scene.

In contrast to traditional monolithic lighting simulations, Lighting Networks allow for an arbitrary combination of these basic lighting operators in order to obtain a composite lighting simulation. By formalizing the representation of illumination taken as input and generated as output, different algorithms can be combined in a data-flow network of simulation algorithms. This flexible combination allows for easily obtaining special lighting effects that would be difficult to compute with a single algorithms or a fixed combination thereof.

In addition, the ability for restricting Lighting Operators to certain parts of the scene allows for tailoring the simulation to the needs of a user. It allows a user to restrict the simulation domain of particular costly algorithms to a small subset of the scene, thereby making it usable for simulating selected lighting effects even in complex scenes. This brings back the control that has been taken away from users by conventional global lighting simulations.

Each of the lighting operators in a network compute the transport of light via a subset of *light paths* through the scene. The concept of Lighting Networks can formally describe this set of light paths using an extended regular expressions notation. This formal description allows for an automatic analysis of composite lighting simulation, which can

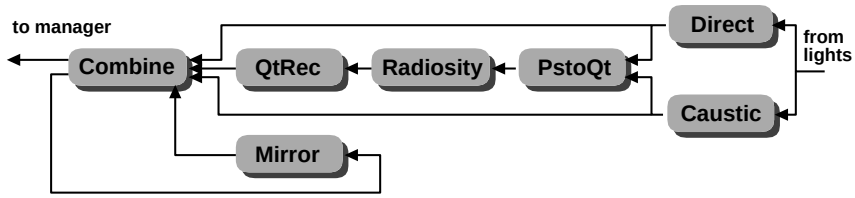


Figure 10: *Lighting Networks: A simple example.*

guarantee completeness and redundant-free computations. Together with the flexibility of the approach this description of Lighting Networks gives full control for creating physically valid simulation scenarios adapted to special needs of a user or an application.

6.1. Applications

The primary advantage of the approach is the freedom with which individual configurations of the lighting network can be created. By restricting operators and using different representations for illumination, the user can easily create custom lighting simulation models. The approach also presents an open framework for composite lighting simulations, where advanced algorithms and new solution strategies can easily be integrated.

Lighting Networks are an ideal tool in many applications that can make use of automatic lighting computations. One example is the production of animations, where the full computation of global illumination in a scene is often too costly. The realism can still be improved considerably, e.g. by restricting radiosity computations to the most important surfaces and using the automatically computed average indirect illumination of this subset of the scene as an ambient term for the rest of the environment.

Figure 10 shows a schematic view of the Lighting Network used for generating the images in Figure 11. The direct and caustic illumination (using ray tracing and Photon Maps [Jen96], respectively) from a lamp with an internal reflector is used directly and as the input for a radiosity solution. Converter lighting operators are used for converting the illumination to a quadtree representation used for input to the radiosity solver. Another converter is used for interpolating the piecewise constant output data. Yet other lighting operators account for perfect specular reflection and transmission using recursive ray tracing and combine the various contributions

While all this makes Lighting Networks a versatile tool for research and development in lighting simulation, its advantages are a best used if given as a tool to the users. Traditionally, global illumination algorithms have offered only little opportunities for configuration, tweaking, and creating special effects. Lighting Networks are a tool where the user may select where and to which extent the features of automatic lighting simulation are appropriate or necessary.

7. Software Architecture for Rendering Applications

In order to use realistic image synthesis successfully in research and development as well as in commercial products, two prerequisites have to be fulfilled. First of all, we need good, accurate, robust, and fast algorithms. Much has happened in this area and some of the important trends have already been described in this report. In addition to the algorithms themselves we need the “glue” between them: A suitable and general software architecture, that offers an environment into which these rendering algorithms can be integrated. Unfortunately, architectural issues have not received the amount of interest that they deserve and there is a need for more research and discussions about these issues.

One contribution to this discussion is the Vision rendering architecture [SS95]. It consists of a small, but flexible rendering kernel that provides a general framework for rendering algorithms and defines suitable interfaces for specific aspects of rendering, like reflection (BRDF) or emission. Algorithms that implement a specific aspect of the rendering process can then be plugged into the kernel. Such a software kernel with its interface definition not only serves as a development platform, it also acts as a theoretic framework for thinking about the rendering process and its algorithms.

Most of rendering architectures that were published in the past were restricted to a single rendering technique, e.g. [CCC87, KA88, SS91, War94]. In contrast, the Cornell “testbed for image synthesis” [TLG93] offers a large tool box of modules for implementing advanced rendering and global illumination algorithms, but it lacks a well-structured framework. Another interesting approach is Glassner’s “Spectrum” architecture [Gla93], which describes a general object-oriented framework for global illumination based on a signal processing approach. Due to this approach, it offers little support for finite element based techniques.

In the past, most rendering systems have mostly been monolithic systems with little flexibility in the fundamental rendering algorithms they offered, and even less flexibility for integrating new developments. This is true both for the research community [SS91, War94, TLG93, Kol91, POV93] and in partic-

(a) Direct illumination

(b) Caustic illumination

(c) Indirect illumination

(d) Combined full simulation

Figure 11: *Partial results computed by the Lighting Network in Figure 10. The images show direct illumination, caustic illumination, indirect illumination resulting from direct and caustic illumination, full composite lighting simulation including path tracing.*

ular for commercial rendering systems like Alias|Wavefront, SoftImage, or 3D Studio.

This situation is unsatisfying in many respects. Within a single research institute and in particular between different institutes, collaboration is often difficult due to many small, monolithic, and incompatible rendering systems, which make it hard to reuse code from one system within another. Also the transfer of technology from research to industry is impaired. It is difficult to integrate new research results into existing rendering systems, because their basic architectures are often too different. Finally, industry faces the problem that innovation is becoming increasingly difficult, because the existing rendering architectures were often designed as closed, proprietary systems. This situation has led to new developments that try to overcome these problems. One example is the new *Maya* architecture from Alias|Wavefront, which essentially follows similar ideas of an open software kernel with a flexible plug-in architecture as presented here.

The kernel of the Vision rendering architecture does not implement any rendering algorithms itself. It defines a

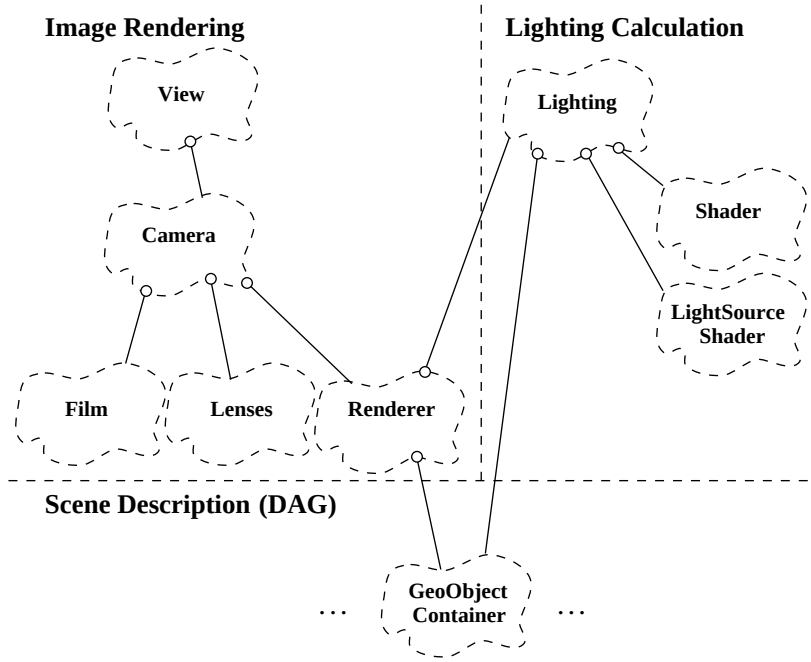


Figure 12: *The three major subsystems of the rendering process: scene description, lighting calculation, and image rendering. For each of the three areas, the subsystems that implement its functionality are depicted.*

supporting infrastructure and defines a framework for the rendering process with well-defined interfaces. This open framework can then be filled and tailored to specific needs by plugging in implementations of specific rendering algorithms.

Such a plug-in architecture acts as the glue between the particular algorithms that implement specific rendering features and the more application-oriented world that deals with the rendering process as a whole. It allows for switching between different implementations of specific rendering algorithms without any need to change other parts of the system. This is possible because of the strict interfaces in such an architecture, which hide changes in one subsystem both within the rendering system and from the application layer.

7.1. Design Goals

What are the important goals when designing an architecture for an open rendering kernel?

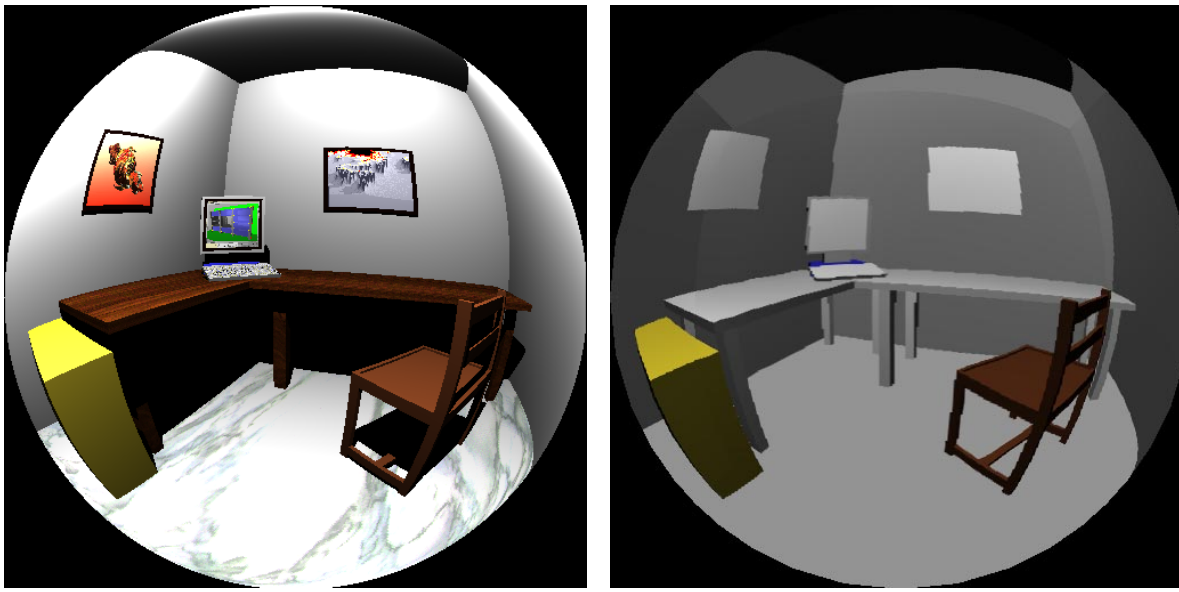
- First of all, the kernel itself should be as general and flexible as possible in order to allow for attaching a wide variety of rendering algorithms to it.
- All interfaces between subsystems of the rendering architecture must be well-defined, well documented and must support both Monte-Carlo as well as finite-element style algorithms. This allows for plugging any kind of algorithm into the kernel, if it can be reduced to (a combination of) these two fundamental techniques.
- The physical meaning and the units of quantities that are passed through the kernel and its interfaces must be standardized. However, this does not preclude the use of non-physically-based algorithms.
- All interfaces have to offer methods to access features with arbitrary accuracy such that advanced algorithms are not limited by the design of the rendering kernel.
- Finally, such an architecture not only serves as an software engineering environment for simplifying research and development. It also serves as a theoretic framework for future work in realistic image synthesis.

7.2. Structure of the Architecture

The process of rendering can be divided into three distinct major subsystems, which also roughly correspond to three different major subsystems during rendering: *scene description, lighting calculation, image rendering* (see Figure 12).

Scene description provides the basic data that rendering algorithms are concerned with: geometric shapes and rendering attributes. Shapes are described by surfaces and volumetric objects. Rendering attributes, on the other hand, directly or indirectly describe the optical properties of these geometric objects.

The *illumination calculation* must be able to compute (at



(a) Ray Tracing (b) OpenGL

Figure 13: *A 180 degree fisheye lens. The left image has been computed with ray tracing by transforming each ray separately using the ray tracing interface of the Lenses subsystem. For the right image OpenGL has been used for rendering. The viewing transformation has been approximated with 5 standard projections on a hemicube and appropriate morphing using OpenGL texture mapping.*

least an approximation to) the incident illumination at any point in a scene. This is the task of global illumination algorithms, but traditional local lighting may also be a sufficient approximation for some purposes. This would also be the correct place for implementing the Lighting Networks approach (see Section 6) within this architecture.

Finally, *image rendering* combines the scene description with the lighting calculation and computes an image of the scene. This subsystem is responsible for computing the viewing transformation, hidden surface removal, image sampling and filtering, and finally image post-processing, e.g. by tone mapping [TR93].

Each of these three major subsystems depends on the services of the preceding one. A scene description may be useful in itself (e.g. in a CAD editing application), but is a requirement for the other two steps. The lighting calculation subsystem together with the scene description could also be used for other purposes than image synthesis, for instance, for lighting simulations.

This gives a broad overview of the architecture. Each of these subsystems is then be further decomposed into smaller subsystems. For each subsystem, we therefore need to define the responsibility and the interface that defines its relationship to the rest of the system [WBW89]. The goal is to define the subsystems and their interfaces, such that the dependencies between them are reduced to a minimum in order to avoid interdependencies between subsystems and allowing independently replacing various parts of the system.

Some of the more important classes and subsystems of the Vision architecture are shown in Figure 12. For more details of this rendering architecture and a discussion of the design decisions please see the references [SS95, SS96, Slu95].

7.3. Applications

The Vision rendering architecture has been used to implement a number of different application, some of which are described in the following. The architecture is used for daily research and development by both students and researchers and it serves as the base for several industrial collaborations in various fields of computer graphics and illumination computations in particular.

All of the important algorithms for global illumination computations have been implemented for this architecture. It supports finite element algorithms like Progressive Refinement Radiosity [CCWG88], Galerkin Radiosity [Zat93], Hierarchical and Wavelet Radiosity [HSA91, GSCH93], as well as Wavelet Radiance algorithms [SH94, CSSD94]. From the class of

Monte-Carlo algorithms simple path tracing [Kaj86] and Bidirectional Estimators [VG95] have been implemented. Several hybrid techniques like Irradiance Gradients [WR88, WH92], Backward Beam-Tracing [Col94], and Photo-Maps [Jen96] are available.

The architecture has also been used to implement a RenderMan compliant renderer [SPS94]. This extension makes use of a special library to interpret RIB files or corresponding calls through the RenderMan API. These calls are used to build a suitable scene description.

Objects of special classes derived from the Shader and LightSourceShader classes define the interface between the rendering kernel and the implementation of the RenderMan ShadingLanguage. Requests to these objects are converted into calls to the appropriate RenderMan shaders and vice versa. The flexible attribute handling of the scene graph is being used for supporting the attribute features of RenderMan [SPS95].

Another example application that shows the flexibility of the architecture is the interface of the Lenses subsystem, which has the responsibility of mapping the 3D environment onto a 2D film in the virtual camera. The standard viewing transformation is usually described by a 4×4 matrix and is sufficient for many applications. However, there are other interesting viewing transformations that cannot be described by this simple technique.

In order to overcome this limitation, an extended interface to the Lenses subsystem has been designed. A client using this interface receives a list of ViewingApprox objects. Each of these objects describes a general viewing transformation for an area on the 2D film. This object contains a standard viewing transformation for a part of the image and an optional list of pairs of quadrilateral 2D grids. Together these approximations define the viewing transformation for the full field of view.

The pairs of grids are used to describe morphing of the projected images and allows for very general viewing transformations, like a 180 degree fish-eye lens, 360 degree spherical environment maps as used in OpenGL [NDW93], or even more general projections [Max83] (see Figure 13 for example images)

By offering appropriate interfaces, both flexible realistic rendering algorithms like ray tracing, and the more limited, but hardware-supported rendering techniques can be supported efficiently within the same software architecture. This flexibility of the rendering kernel is also being used to explore the capabilities of todays graphics hardware to support realistic rendering algorithms.

These few example application of the rendering architecture demonstrate the benefits that can be obtained by using a well-structured and general framework. In particular the flexibility of this architecture has proven extremely useful. It was possible to easily integrate very different rendering techniques into a common system and make them work in combination.

8. Conclusions and Open Problems

Photo-realistic rendering has come a long way in reaching the goal of generating images that are hard to distinguish from photographs of the same scene. In the process of this research many interesting new techniques have been developed that may also be applied in other engineering applications. This report gives an overview of the state-of-the-art in this field of computer graphics and describes some of the more recent developments and trends.

In particular the use of modern mathematical approaches like hierarchical descriptions and algorithms combined with the clever use of efficient data structures and programming techniques has led to new robust, accurate, and efficient algorithms that allow to simulate the physical process on which photo-realistic image generation is based.

Together with approaches that increase the practicality of these techniques like general rendering architectures and the Lighting Network approach this brings photo-realistic rendering and global illumination to new level of applicability and usability. This will hopefully result in new and better tools for the user of systems that can make use of photo-realistic rendering.

References

Atk76. Atkinson, K. E. *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind.* Society for Industrial and Applied Mathematics, Philadelphia, 1976.

BT92. Bouatouch, K. and Tellier, P. A two-pass physics-based global lighting model. In *Proceedings of Graphics Interface '92*, pages 319–328, Toronto, Ontario, May 1992.

CRMT91. Chen, S. E., Rushmeier, H. E., Miller, G., and Turner, D. A progressive multi-pass method for global illumination. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):165–174, July 1991.

CSSD94. Christensen, P. H., Stollnitz, E. J., Salesin, D., and DeRose, T. D. Wavelet radiance. In *Fifth Eurographics Workshop on Rendering*, pages 287–301, Darmstadt, June 1994.

CCWG88. Cohen, M., Chen, S. E., Wallace, J. R., and Greenberg, D. P. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):75–84, August 1988.

CW93. Cohen, M. F. and Wallace, J. R. *Radiosity and Realistic Image Synthesis*. Academic Press, 1993.

Col94. Collins, S. Adaptive splatting for specular to diffuse light transport. In Haas, S., Müller, S., Sakas, G., and Shirley, P., editors, *Fifth Eurographics Workshop on Rendering*, pages 119–135, Darmstadt, June 1994.

CS93. Comba, J. L. D. and Stolfi, J. Affine arithmetic and its applications to computer graphics. In *Anais do VII Sibgrapi*, pages 9–18, 1993. Available from http://www.dcc.unicamp.br/ stolfi/EXPORT/papers/affine-arith.

CCC87. Cook, R., Carpenter, L., and Catmull, E. The Reyes image rendering architecture. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):95–102, July 1987.

Dau92. Daubechies, I. *Ten Lectures on Wavelets*. SIAM Philadelphia, Pennsylvania, 1992.

Gla93. Glassner, A. Spectrum: An architecture for image synthesis, research, education, and practice. In Strauss, P. S., editor, *Developing Large-scale Graphics Software Toolkits*, (SIGGRAPH '93 Course Notes 3), pages 1.1–1.44. SIGGRAPH, August 1993.

GCS94. Gortler, S., Cohen, M. F., and Slusallek, P. Radiosity and relaxation methods. *IEEE Computer Graphics & Applications*, 14(6):48–58, November 1994.

GSCH93. Gortler, S. J., Schröder, P., Cohen, M., and Hanrahan, P. M. Wavelet radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27:221–230, August 1993.

HSA91. Hanrahan, P., Salzmann, D., and Aupperle, L. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):197–206, 1991.

Hec90. Heckbert, P. Adaptive radiosity textures for bidirectional ray tracing. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 145–154, August 1990.

HSD94. Holzschuch, N., Sillion, F., and Drettakis, G. An efficient progressive refinement strategy for hierarchical radiosity. In *Photorealistic Rendering Techniques (Proceedings Fifth Eurographics Workshop on Rendering)*, pages 343–357, Darmstadt, Germany, June 1994. Springer.

Jen96. Jensen, H. W. Global illumination using photon maps. In Pueyo, X. and Schröder, P., editors, *Rendering Techniques '96 (Proceedings Seventh Eurographics Workshop on Rendering)*, pages 21–30. Springer, June 1996.

Kaj86. Kajiya, J. T. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):143–150, August 1986.

KW86. Kalos, M. H. and Whitlock, P. A. *Monte Carlo Methods*. John Wiley & Sons, 1986.

KA88. Kirk, D. and Arvo, J. The ray tracing kernel. In *Proceedings of Ausgraph*, pages 75–82, July 1988.

Kol91. Kolb, C. E. *Rayshade User's Guide and Reference Manual, Version 0.1*, 1991.

Max83. Max, N. L. Computer graphics distortion for IMAX and OMNIMAX projection. In *Nicograph '83 Proceedings*, pages 137–159, December 1983.

Moo66. Moore, R. E. *Interval Analysis*. Prentice-Hall, 1966.

NDW93. Neider, J., Davis, T., and Woo, M. *OpenGL Programming Guide*. Addison Wesley, 1993.

PB94. Pattanaik, S. and Bouatouch, K. Haar wavelet: A solution to global illumination with general surface properties. In *Photorealistic Rendering Techniques (Proceedings of Fourth Eurographics Workshop on Rendering)*, pages 281–294. Eurographics, Springer, June 1994.

POV93. POV-Ray Team. *Persistence of Vision Ray Tracer (POV-Ray), Version 2.0*, 1993.

Sch94. Schröder, P. *Wavelet Algorithms for Illumination Computations*. PhD thesis, Princeton University, November 1994.

SGCH93. Schröder, P., Gortler, S., Cohen, M. F., and Hanrahan, P. Wavelet projections for radiosity. *Proceedings on the Fourth Eurographics Workshop on Rendering*, pages 95–104, June 1993.

SH94. Schröder, P. and Hanrahan, P. Wavelet methods for radiance computations. In *Photorealistic Rendering Techniques (Proceedings Fifth Eurographics Workshop on Rendering)*, pages 303–311, Darmstadt, June 1994. Springer.

SS91. Shirley, P. and Sung, K. A ray tracing framework for global illumination systems. In *Proceedings Graphics Interface '91*, pages 117–128, Calgary, June 1991.

SWH+95. Shirley, P., Wade, B., Hubbard, P. M., Zareski, D., Walter, B., and Greenberg, D. P. Global illumination via density-estimation. In Hanrahan, P. and Purgathofer, W., editors, *Rendering Techniques '95 (Proceedings Sixth Eurographics Workshop on Rendering)*, pages 219–230, Dublin, June 1995. Springer.

SA93. Shirman, L. A. and Abi-Ezzi, S. S. The cone of normals technique for fast processing of curved patches. *Computer Graphics Forum (EUROGRAPHICS '93 Proceedings)*, 12(3):261–272, September 1993.

Sil95. Sillion, F. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995.

SD95. Sillion, F. and Drettakis, G. Feature-based control of visibility error: A multi-resolution clustering algorithm for global illumination. *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 145–152, August 1995.

SDS95. Sillion, F., Drettakis, G., and Soler, C. A clustering algorithm for radiance calculation in general environments. In Hanrahan, P. and Purgathofer, W., editors, *Rendering Techniques '95 (Proceedings of Sixth Eurographics Workshop on Rendering)*, pages 196–205. Springer, August 1995.

SP89. Sillion, F. X. and Puech, C. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):335–344, July 1989.

Slu95. Slusallek, P. *Vision – An Architecture for Physically Based Rendering*. PhD thesis, University of Erlangen, IMMD IX, Computer Graphics Group, June 1995.

SPS94. Slusallek, P., Pflaum, T., and Seidel, H.-P. Implementing RenderMan - practice, problems, and enhancements. *Computer Graphics Forum (EUROGRAPHICS '94 Proceedings)*, 13(3):443–454, September 1994.

SPS95. Slusallek, P., Pflaum, T., and Seidel, H.-P. Using procedural RenderMan shaders for global illumination. In *Com-*

puter Graphics Forum (EUROGRAPHICS '95 Proceedings), pages C–311–C–324, Maastricht, August 1995.

SSSS95. Slusallek, P., Schröder, M., Stamminger, M., and Seidel, H.-P. Smart links and efficient reconstruction for wavelet radiosity. In *Rendering Techniques '95 (Proceedings Sixth Eurographics Workshop on Rendering)*, pages 240–251, Dublin, June 1995. Springer.

SS95. Slusallek, P. and Seidel, H.-P. Vision: An architecture for global illumination calculations. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):77–96, March 1995.

SS96. Slusallek, P. and Seidel, H.-P. Towards an open rendering kernel for image synthesis. In Pueyo, X. and Schröder, P., editors, *Rendering Techniques '96 (Proceedings Seventh Eurographics Workshop on Rendering)*, pages 51–60, Porto, June 1996. Springer.

SSH+96. Slusallek, P., Stamminger, M., Heidrich, W., Popp, J.-C., and Seidel, H.-P. Composite lighting with lighting networks. Technical Report TR-96-14, Universität Erlangen, IMMD 9, 1996. http://www9.informatik.uni-erlangen.de/Publications.

SAG94. Smits, B., Arvo, J., and Greenberg, D. A clustering algorithm for radiosity in complex environments. *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 435–442, July 1994.

Sny92. Snyder, J. M. Interval analysis for computer graphics. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 121–130, July 1992.

SSS97a. Stamminger, M., Slusallek, P., and Seidel, H.-P. Bounded clustering – finding good bounds on clustered light transport. Technical Report TR-97-1, Universität Erlangen, IMMD 9, 1997. http://www9.informatik.uni-erlangen.de/Publications/.

SSS97b. Stamminger, M., Slusallek, P., and Seidel, H.-P. Bounded radiosity – illumination on general surfaces and clusters. *Computer Graphics Forum (EUROGRAPHICS '97 Proceedings)*, 16(3), September 1997. to appear.

SSS97c. Stamminger, M., Slusallek, P., and Seidel, H.-P. Bounded radiosity – illumination on general surfaces and clusters. Technical Report, to appear in Proc. EUROGRAPHICS '97 TR-96-15, Universität Erlangen, IMMD 9, 1997. http://www9.informatik.uni-erlangen.de/Publications/.

SSS97d. Stamminger, M., Slusallek, P., and Seidel, H.-P. Isotropic clustering for hierarchical radiosity — implementation and experiences. In *Proceedings Fifth International Conference in Central Europe on Computer Graphics and Visualization — WSCG '97*, 1997.

TLG93. Trumbore, B., Lytle, W., and Greenberg, D. P. A testbed for image synthesis. In Strauss, P. S. and Trumbore, B., editors, *Developing Large-Scale Graphics Software Toolkits (SIGGRAPH '93 Course Notes 3)*, pages 4.7–4.17, Anaheim, August 1993.

TR93. Tumblin, J. and Rushmeier, H. E. Tone reproduction for realistic computer generated images. *IEEE Computer Graphics & Applications*, 13(6):42–48, November 1993.

VG94. Veach, E. and Guibas, L. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques (Proceedings Fifth Eurographics Workshop on Rendering)*, pages 145–167, Darmstadt, June 1994. Springer.

VG95. Veach, E. and Guibas, L. J. Optimally combining sampling techniques for Monte Carlo rendering. *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 419–428, August 1995.

WCG87. Wallace, J. R., Cohen, M. F., and Greenberg, D. P. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):311–320, July 1987.

War94. Ward, G. J. The RADIANCE lighting simulation and rendering system. *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 459–472, July 1994.

WH92. Ward, G. J. and Heckbert, P. S. Irradiance gradients. In Chalmers, A. and Paddon, D., editors, *Third Eurographics Workshop on Rendering*, pages 85–98, Bristol, May 1992.

WR88. Ward, G. J. and Rubinstein, F. A ray tracing solution for diffuse interreflection. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):85–92, August 1988.

Whi79. Whitted, T. An improved illumination model for shaded display. *Computer Graphics (Special SIGGRAPH '79 Issue)*, 13(3):1–14, August 1979.

WBW89. Wirfs-Brock, R. and Wilkerson, B. Object-oriented design: A responsibility-driven approach. In *OOPSLA 89 Conference Proceedings*, pages 71–75, New Orleans, 1989.

Zat93. Zatz, H. R. Galerkin radiosity: A higher order solution method for global illumination. *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 213–220, August 1993.