Photorealistic Rendering using the PhotonMap^{TM}

Ingo Wald

August 1999

Universität Kaiserslautern
Fachbereich Informatik
AG Numerische Algorithmen
Prof. Dr. Stefan Heinrich

Aufgabenstellung und Betreuung:
Dr. Alexander Keller

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt zu haben.

Kaiserslautern, den 1. September 1999

“Comparisons with existing global illumination techniques indicate that the photon map provides an efficient environment for global illumination”
Henrik Wann Jensen

“The truth is out there”
The X-Files

“Another solution is just to always use enough photons”
Henrik Wann Jensen

“if it won’t yield to brute force - use a bigger hammer”
unknown author

First of all, I wish to thank all those who have contributed to this work, either directly or indirectly:
I wish to thank all the members of our group - Alex Keller, Johannes Timmer, Thomas Kollig, Ilja Friedel and Marc Pauly - for their help and their company, especially Thomas Kollig, with whom discussions on issues of photorealistic rendering have always been valuable, and who’s advice on Metropolis has been invaluable.
I also wish to thank Dr. Alex Keller, for allowing me to do this work, and for helping me in managing it.
Finally, I wish to thank Roger Daneker and Rene Schaetzl, for reading and correcting this work, and my girlfriend Gabriele Kaiser, for bearing me when I’ve been stressed, and for cheering me up when I’ve been disappointed.

Contents

Chapter 1

Introduction

Recently, the PhotonMap^{TM} [1] has been introduced into the field of photorealistic rendering in a series of papers ([JC95b], [Jen95], [JC95a], [Jen96a], [Jen96b], [JC98], etc.). Because of its strong impact, the goal of this work was the implementation of the photon map and its integration into an existing rendering platform. While doing this, several problems arose which were first dismissed as implementation problems, but which were soon after revealed to be intrinsic flaws of the photon map method. Since these problems have not yet been sufficiently documented, they have been profoundly investigated in this work.

The aim of this work is to analyze and to improve the original method, making it first necessary to name, quantify, classify and document its flaws. We start by briefly summarizing the basics of global illumination in section 2.1, followed by some notes on efficient implementations of the photon map algorithms in sections 2.3.1, 2.3.2 and 2.4. The integration of the photon map into an existing distribution ray tracer is presented in section 2.5, together with a brief summary of how the photon map is used in the original papers. A detailed analysis of the photon map follows in chapter 3, where both its advantages and its disadvantages are discussed. After outlining the problems, several approaches to improve the photon map are proposed and examined in chapter 4. Finally, chapter 5 presents several of the scenes used in our experiments, each with a short discussion of the results when rendering it with the photon map and its different extensions. The conclusions are drawn in chapter 6.

[1] PhotonMap^{TM} is a registered trademark of mental images, Gesellschaft f”ur Computerfilm und Maschinenintelligenz mbH & Co KG., D10623 Berlin, Germany. In the sequel, the ^{TM} sign will be omitted.

Chapter 2

The Photon Map

2.1 Basics of Global Illumination

In order to understand the photon map and its problems, we will first discuss some of the basics of photorealistic rendering, by defining the radiance equation: The basic quantity describing the perceivable illumination in a scene is the *radiance* L. For a deeper coverage of the fundamentals of global illumination, together with a complete definition and physical derivation of the term radiance, see [Kel98], [CW93] and [Kaj86]. The radiance equation was first derived in [Kaj86] and is a physically based model for describing the transport of radiance in a scene. Using the notation

S**:** the surface of the scene,

Ω**:** the hemisphere of all directions ω at surface position $x \in S$,

$L(x, \omega)$**:** the radiance leaving $x \in S$ into direction $\omega \in \Omega$,

$L_i(x, \omega)$**:** the irradiance at x, i.e. the radiance reaching $x \in S$ from direction $\omega \in \Omega$,

$L_e(x, \omega)$**:** the exitant radiance emitted from $x \in S$ into direction $\omega \in \Omega$, and

$f_r(\omega, x, \omega_i)$**:** the bidirectional scattering distribution function,

the radiance $L(x, \omega)$ is the sum of the emitted radiance $L_e(x, \omega)$ and all radiance incident in x and reflected into direction ω:

$$L(x, \omega) = L_e(x, \omega) + \int_\Omega f_r(\omega, x, \omega_i) L_i(x, \omega_i) \cos\theta_i \, d\omega_i. \tag{2.1}$$

Let $h(x, \omega)$ denote the raycasting function, i.e. $y = h(x, \omega)$ is the closest intersection between the surface of the scene and a ray starting at position $x \in S$ with direction ω. Then, restricting our problem to vacuum radiation transport[1], the irradiance can be substituted by $L_i(x, \omega) = L(h(x, \omega), -\omega)$, which yields

$$L(x, \omega) = L_e(x, \omega) + \int_\Omega f_r(\omega, x, \omega_i) L(h(x, \omega_i), -\omega_i) \cos\theta_i \, d\omega_i, \tag{2.2}$$

[1]The restriction to vacuum-transport is chosen for reasons of simplicity, since volumetric effects are not covered in this work. For extensions of the PhotonMap to participating media, see [JC98].

The radiance equation is a second kind Fredholm integral equation. With f_r as the kernel of the integral operator, it can be written in operator form as

$$L = L_e + T_{f_r} L. \tag{2.3}$$

Inserting equation 2.3 into itself yields

$$\begin{aligned} L &= L_e + T_{f_r} L = L_e + T_{f_r}(L_e + T_{f_r}(L_e + T_{f_r}(\dots))) && (2.4)\\ &= \sum_{i=0}^{\infty} T_{f_r}^i \, L_e, && (2.5) \end{aligned}$$

which is called the Neumann-series. A more profound derivation of the Neumann-series can be found in [Kel98], which shows both its correctness and its convergence under certain conditions.

The kernel f_r of the integral operator is the *bidirectional scattering distribution function* (bsdf), which describes the interaction of light with the surface. The bsdf f_r is defined as

$$f_r(\omega, x, \omega_i) := \frac{dL(x,\omega)}{L_i(x,\omega_i)d\omega_i}$$

and describes which fraction of the radiance coming from direction ω_i is scattered into direction ω. The bsdf is a probability density function and can be viewed as describing the probability of a photon with incoming direction ω_i to be scattered into direction ω.

Our implementation uses the Ward reflection model [War92], where f_r is modelled to consist of a diffuse, a specular and a glossy part

$$f_r = f_{rd} + f_{rs} + f_{rg}.$$

Using the linearity of the integral operator, T_{f_r} can be expressed as

$$T_{f_r} = T_{f_{rd}} + T_{f_{rs}} + T_{f_{rg}}. \tag{2.6}$$

Diffuse interactions (i.e. applications of $T_{f_{rd}}$) are called *regular*, while both specular and glossy interactions are called *singular*. Slightly glossy interactions may also be approximated by regular interactions, which introduces some small error, but greatly simplifies computations in scenes with lots of slightly glossy objects. As stated above, the bsdf is a probability density function, and even though our implementation only uses the Ward reflection model, the photon map is generally capable of handling any arbitrary pdf.

Using 2.6, equation 2.3 can be split into several terms

$$\begin{aligned} L &= L_e \\ &+ \underbrace{T_{f_{rd}} L}_{=: L_r} \\ &+ T_{(f_{rs}+f_{rg})} L. \end{aligned}$$

These terms will in the sequel be calculated separately: L_e is the object's emittance, given by the material definition, $T_{(f_{rs}+f_{rg})}L$ describes specular and glossy reflections, and $L_r := T_{f_{rd}}L$ is the diffusely reflected radiance. L_r can be furtherly split into

$$\begin{aligned} L_r &= T_{f_{rd}} L \\ &= T_{f_{rd}}(L_e + T_{f_r} L) \\ &= T_{f_{rd}} L_e + T_{f_{rd}} T_{f_{rd}} L + T_{f_{rd}} T_{(f_{rs}+f_{rg})} L. \end{aligned}$$

The first term $T_{f_{rd}}L_e$ is called *direct illumination*, while the second term $T_{f_{rd}}T_{f_{rd}}L$ represents *soft indirect illumination*. The last term $T_{f_{rd}}T_{(f_{rs}+f_{rg})}L$ describes *caustics*. Caustics are illumination details created when light hits a diffuse surface after at least one singular interaction. Using the light path notation, where L denotes a light source, S denotes a singular interaction (i.e. an application of $T_{(f_{rs}+f_{rg})}$), and D denotes a regular interaction, caustics are created by light paths of the form $L\{SD\}^*SD$. Therefore, photons resulting from such light paths are called *caustic photons*, all other photons are called *diffuse photons*. Note that Jensen uses a different definition by considering only direct caustics (LS^*D) (see [JC95b]), which are only a subset of our definition.

2.2 Generating the Photon Map

The photon map is a discrete density approximation ([Kel98]) of the irradiance in the scene. This discrete density is created by using a random walk ([KMS94]), by emitting photons from light sources and tracing them through the scene: Whenever a photon $p = (x_0, \omega_0, \phi)$ interacts with the surface in $x_1 = h(x_0, \omega_0)$, its new position x_1, its incoming direction ω_0, and its energy ϕ are stored in an array called the photon map data structure.

```
void PhotonMapShader::RandomWalk() {
  for each LightSource LS {
    // calculate nr of photons on source LS
    n_i = LS.Emittance() / TotalEmittance * n_Start
    for p = 1..n_i {
      (x_0, ω_0) = LS.StartPhoton();
      φ = LS.Emittance() / n_i
      singular = false;
      for (pathLen=0; pathLen<maxLen; pathLen++) {
        (x_1, obj) = h(x_0, ω_0);
        if (obj == null) break; // particle lost...
        (interaction, ω_1, f) = Interaction(x_1, ω_0, obj.bsdf(x_1));
        if (interaction is singular) {
          singular = true;
        } else {
          if (singular)
            causticMap.AddPhoton(x_1, ω_0, φ)
          else
            diffuseMap.AddPhoton(x_1, ω_0, φ)
          singular = false;
        }
        if (interaction == absorbed) break; //   particle absorbed...
        x_0 = x_1; ω_0 = ω_1; φ = f * φ;
      }
    }
  }
}
```

`Interaction(x,ω,bsdf)` corresponds to an application of T_{f_r} as in the previous section. It uses the bsdf f_r in x to calculate the interaction of the incoming photon (x, ω) with the surface. To prevent an infinite number of interactions of a photon, we use the 'Russian Roulette' approach as presented in [AK90] to choose whether the photon is absorbed or not. This method is also used to choose the type of interaction (i.e. f_{rs}, f_{rd} or f_{rg}). The out-scattering direction is generated according to the chosen interaction type. `Interaction` returns both the new direction ω_1 and $f = \frac{f_i(\omega_1, x_1, \omega_0)}{1-P(f_i)}$, where f_i is either f_{rs}, f_{rd} or f_{rg}. A new photon with random position and direction on source LS

are generated by (x, ω)=`LS.StartPhoton()`. The photons can also be generated by using Quasi-random numbers (as presented in [Kel98]).

This kind of random walk is only a pure forward simulation and therefore yields several problems, which are covered in section 3.3.7.

To be able to store large amounts of photons, a very memory-efficient storage technique is required. Therefore, a photon is stored in a compressed form similar to the one proposed in [Jen96b]:

```
class Photon {
  float         position[3];
  unsigned char theta, phi;
  rgbe          energy;
};
```

The position of the photon is directly stored as 3 floats (using doubles would nearly double memory cost by gaining few accuracy), its energy is stored compressed in Wards realpixel representation (see [War91b]), and the incoming direction ω is discretized into two characters, azimuth and declination angle:

```
theta = (unsigned char) (255.0 * acos(ω.z) / π);
phi   = (unsigned char) (256.0 * atan2(ω.y, ω.x) / 2*M_PI;
```

The direction can then be decompressed by

```
ω     = Vector(sinTheta[theta]*cosPhi[phi],
               sinTheta[theta]*sinPhi[phi],
               cosTheta[theta]);
```

where

```
cosTheta[i] = cos(i * M_PI / 255.0);
cosPhi[i]   = cos(i * 2 * M_PI / 256.0);
sinTheta[i] = sin(i * M_PI / 255.0);
sinPhi[i]   = sin(i * 2 * M_PI / 256.0);
```

are look-up tables for efficient decompression. The loss in accuracy by discretizing direction and energy was tested to be negligible. Using this representation, only 18 bytes per photon are needed. Some of the later proposed extensions (see chapter 4) require the storage of additional information, for example information about the origin x_0 of a photon.

To gain flexibility, the photon map data structure was separated from the actual photon map shader. This allows to use the abstract data structure in different applications, for example when needing multiple maps or importance maps. Therefore, the `PhotonMap` class only supplies methods for the storage of a photon (`AddPhoton`) and for the nearest-neighbour-query (`Query`). Directly after having generated all the photons, the acceleration structure for the k-nearest-neighbour query is generated by calling `PrepareForQuery`. In [Jen96b], Jensen proposed the use of a linked list of arrays of 64k photons. Compared to a single array, this approach is less efficient due to unneccessarily increasing access times. Since the number of photons is not known in advance, our implementation uses dynamic arrays [Sed98] for storing the photon maps: Dynamic arrays automatically adapt to the growing memory requirements in the random walk, and are especially useful due to the fact that the number of photons will no longer increase after the random walk is complete.

```
class PhotonMap {
   Photon *photon; // dynamic array of photons
   int N;          // current number of photons
   virtual void AddPhoton(Photon p);
   virtual void PrepareForQuery();
   virtual void Query(Point x,PhotonHeap &q);
   // search q.k nearest Photons around x, store them in q
}
```

The abstract definition and the object-oriented design allow for different implementations: Both a version based on a left-balanced kd-tree (as proposed by Keller in [Kel96] and Jensen in [Jen96b]), as well as a version based on a regular grid have been implemented. Hybrid methods seem promising, but have not yet been testet. Other implementations are available as well (see, e.g. [VBS99]).

2.3 The 3d-Tree Photon Map

2.3.1 Efficient Construction of Balanced kd-Trees.

KD-Trees ([Ben75], [BF79]) are k-dimensional binary search-trees (for our application, $k = 3$): Each subtree with its root node n in level l_n is partitioned with respect to dimension $d = l_n \bmod k$,[2] such that each node l in the left subtree of n is smaller than n, and each node r in the right subtree is larger than n (with respect to dimension d):

$$\bigwedge l \in leftsubtree(n),\ r \in rightsubtree(n)\ :\ l_d \leq n_d \leq r_d \tag{2.7}$$

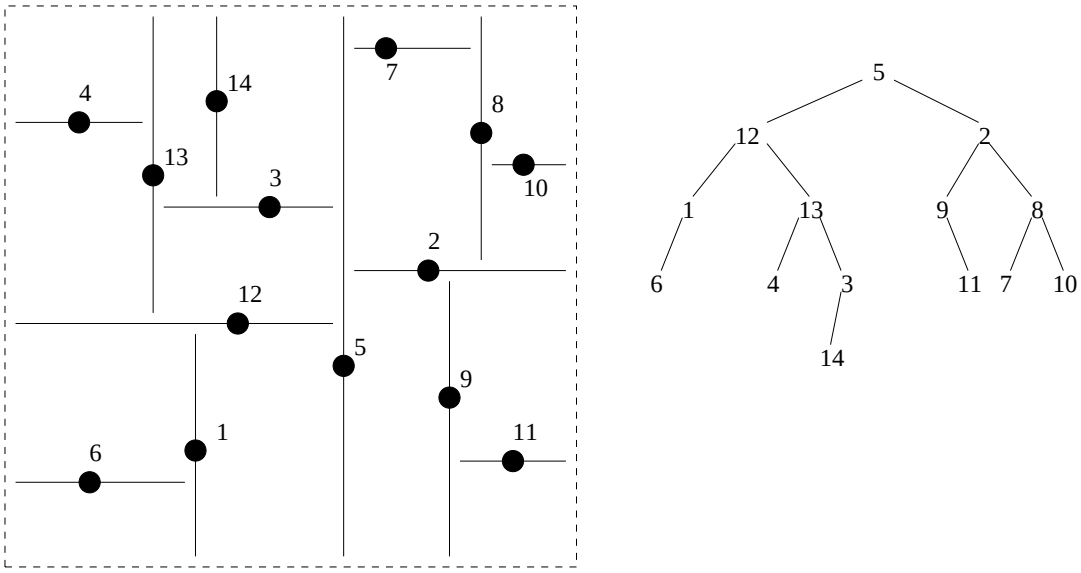


Figure 2.1: A sample kd-tree for $k = 2$ (not balanced).

In our implementation, the photons are stored in a left-balanced, complete kd-tree. Balancing the tree allows for very efficient storage and access as well as for efficient queries because of a minimized tree depth ($depth(n) = 1 + \lceil log\, N \rceil$). If a tree with N elements is complete, "all its levels are filled, except for the final one, which is filled from left to right" [Sed98]. It can then be stored efficiently in an array of N photons. Choosing the root-node's index to be 1, the methods

```
Valid(node)    := (node ≤ N)
LeftSon(node)  := (2*node);
RightSon(node) := (2*node+1);
```

allow for efficient access to the photons. *Valid(node)* returns whether *node* is still a valid index in the photon map array. This representation does not need any additional memory (such as pointers) for representing the tree.

[2] The splitting dimension may also be chosen in a different way, for example by subdividing the dimension in which the boudingbox of the subtree has the largest extent. This requires only minor modifications and may yield additional performance gains especially in settings like SCENE10, where the scenes extent is x and y dimension is much larger than in z dimension.

The algorithm for building the balanced kd-tree works by choosing the root element such that condition 2.7 holds. Then, both subtrees are processed recursively. Finding the proper root element of a subtree is done as follows: Knowing that - to ensure a balanced tree - $m-1$ elements have to be stored in the left subtree, the problem of finding the root element consists of finding the mth median, which can be determined in several ways (for example, see [VBS99]). While the idea is more or less straightforward, problems arose from the fact that the elements in a subtree are not stored sequentially (see figure 2.2).

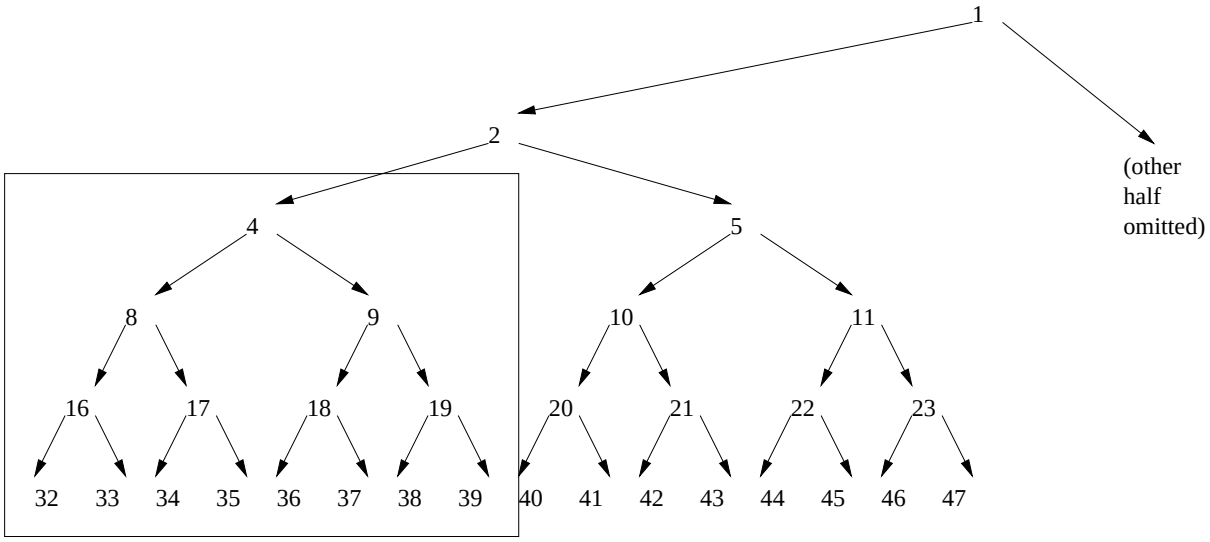


Figure 2.2: Non-sequential numbering of nodes in a subtree. The nodes in the subtree with root node 4 are 4, 8, 9, 16, 17, 18, 19, 32, ...

Therefore, another method was developed which is based on a modified QuickSort algorithm: Each subtree is first partitioned by a modified QuickSort-partitioning-step, such that condition 2.7 holds for this subtree. After that, both subtrees are built recursively. The complete *BuildTree* algorithm is presented overleaf. Its average complexity can be derived from the complexity of QuickSort (see [Sed98]) and ist therefore

$$T_{BuildTree} \in O(n \log n).$$

The algorithm works *in place* and can be considered optimal in both time complexity and memory consumption. As one can see in table 2.1, initialization time is negligible even for moderately sized photon maps. Even a tree of 10 million photons can be constructed in about 3 minutes on one MIPS R10000/250MHz processor of an SGI O2000.

photons	$T_{RandomWalk}$	$T_{RandomWalk} + T_{BuildTree}$
100k	5s	7s
200k	9s	13s
400k	17s	26s
700k	30s	48s

Table 2.1: Initialization time for *RandomWalk* and *BuildTree*. Measures have been taken with SCENE6, on a DEC alpha/333. Larger photon map resolutions could not be tested due to lack of memory.

```
BuildTree(Photon *p, int N, int root, int level)
// partition the subtree under root such that
// kd-condition holds.
{
  if (!Valid(root))
     return; // terminate recursion...
  splitDim = SplitDimension(level); // i.e. level % 3;
  Index minL(LeftSon(root));
  Index minR(RightSon(root));

  Index l = minL;
  Index r = minR;
  while(1) {
    double splitVal = p[root][splitDim];
    while (l.node <= N && p[l.node][splitDim] <= splitVal) ++l;
    while (r.node <= N && p[r.node][splitDim] >= splitVal) ++r;
    if (l.node > N && r.node > N)
      break; // kd-condition now holds for root
    if (l.node > N) {
      minR = r; l = minL; Swap(p,root,r.node); ++r;
    } else if (r.node > N) {
      minL = l; r = minR; Swap(p,root,l.node); ++l;
    } else {
      Swap(l.node, r.node); ++l; ++r;
    }
  }
  BuildTree(p, N, LeftSon(root), level+1);
  BuildTree(p, N, RightSon(root), level+1);
}

KD::BuildTree(Photon *p, int N)
{
  BuildTree(p,N,1,0)
}

class Index {
  int mask, node, level;
  Index(int root) {
    // initialize to root element
    node = mask = root; level = 0;
  }
  void operator++() {
    // step to next element in subtree
    node++;
    if ((node >> level) != mask) {
      level++;
      node = mask << level;
    }
  }
};
```

Algorithm `BuildTree` for efficiently building a left-balanced, complete kd-tree. *Index* is a class which provides a method for efficiently iterating through all nodes of a specified subtree in level-order.

2.3.2 Efficient Nearest-Neighbour-Query in a kd-Tree

The k nearest neighbours are stored in a priority queue (class *PhotonList*), which is efficiently implemented as a heap (see [Sed98]). This organization allows for direct access to the element with the greatest distance, since in a heap, MaxDistance()==dist[0]. `Process(candidate, distance)` inserts a new candidate into the queue and removes the item with the greatest distance if the list already contains k elements. `Process` may also do some basic plausibility tests before storing a photon, see section 4.1.

```
class PhotonList {
  Photon nearest[K_MAX];
  double dist[K_MAX];
  int k; // current nr of elements
  double MaxDistance();
  void Process(Photon candidate, double dist);
};
```

The Query in the kd-tree is done by recursively scanning the tree in a modified in-order:

1. determine and recursively scan the subtree which is closer to query-pos
2. if node cannot be culled, process node
3. if distant subtree can not be culled, recursively scan distant subtree

This is in fact a backtracking approach ([Mad]), which splits the problem of scanning the current subtree in several smaller problems - left and right subtree and current node - and processes these in the order of their probablity of yielding photons which are closer than the current candidates. The efficiency of the algorithm results from culling: The distant subtree does not have to be scanned if the minimal distance of a photon in the distant subtree is greater than the current maximum candidate distance. This minimum distance is approximated as the distance of the root-photon to the query-position in the current splitting dimension `splitDim`. Since this test only considers the distance in the current splitting dimension, the minimum distance is usually heavily underestimated, resulting in sub-optimal culling. However, this test was chosen for speed and simplicity. More involved tests have been examined, but the performance gained through better culling has always been offset by the higher overhead of these tests.`Query` was also implemented as an iterative version. Since a significant speedup could not be observed, the code is omitted here. Using least-cost-search ([Mad]) instead of backtracking did not yield faster query times.

```
Query(PhotonList &neighbours, Point x_q, int node, int level)
{
  if (!Valid(node))
     return; // terminate recursion...
  splitDim = SplitDimension(level); // i.e. level % 3;
  Vector dist = photon[node] - x_q;
  projDist = dist[splitDim];
  if (projDist > 0)
    { // left subtree is closer
      Query(neighbours, x_q, LeftSon(node), level+1);
      if (projDist < neighbours.MaxDistance())
        {
          if (||dist||_2 < neighbours.MaxDistance())
             neighbours.Process(node, ||dist||_2);
          Query(neighbours, x_q, RightSon(node), level+1);
        }
    } else {
      // right subtree is closer
      ...
    }
}
```

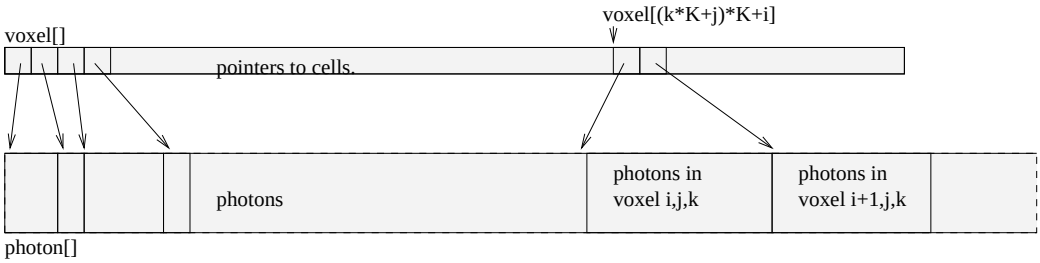
2.4 The GridPhotonMap

Since a large part of the rendering time is spent in the nearest-neighbour-query, an implementation which is faster than the kd-tree would result in a significant speedup. Therefore, additional algorithms have been developed and tested, in which the most promising approach was using a regular grid for storing the photons. The grid needs additional memory for representing the voxels: In our implementation, this is exactly one pointer and one integer (for tagging) per voxel, which is negligible when using a reasonable ratio of photons to voxels (even a ratio such low as 10:1 would yield a memory overhead of less than 5%).

A grid of resolution $K_x \times K_y \times K_z$ [3] can be stored and accessed by

```
struct Voxel {
  Photon *photon;
  int     tag;
} voxel[Kx * Ky * Kz];

Voxel(ix, iy, iz) = voxel[ix + Kx(iy + Kyiz)]
```



As compared to a kd-tree, the regular grid is not adaptive, and therefore reaches optimal performance only when photons are well distributed in the scene. While this is often the case for diffuse photons, it may be problematic with caustic photons, which tend to be clustered. Specifying a maximum search-radius yields performance-gains even in this case.

Constructing the regular grid is done 'in place' in a very straightforward way, by using QuickSort's in the different dimensions and linking pointers into the array. Since the implementation is straightforward, the code is omitted because of its length. Having $K \times K \times K$ voxels, the average complexity for building the grid is

$$T(n) = n \log n + K \frac{n}{K} log \frac{n}{K} + K^2 \frac{n}{K^2} log \frac{n}{K^2} \in O(n \log n)$$

Finding the nearest neighbours in the grid is done by processing the voxels in an order similar to a 3-dimensional seed-fill. The seed voxel V_{seed} is the voxel containing the query-point x_q. The improved performance is due to the fact that the photons in the seed voxel are already very close to the query-point and therefore very likely to be candidates. As a result, the current maximum candidate distance diminishes quickly, and results in fast culling, since only voxels closer than $MaxDist()$ have to be processed.

[3] In non-cubical scenes, each dimension of the grid can be subdivided with a different resolution to match the scene's different extents in different dimensions. SCENE10, for example, is best voxelized in a resolution of $K_x \times K_y \times K_z$, where $K_x = K_y = 10K_z$.

Query works as follows:

```
Query(x_q, neighbours)
  {
   currentTag = currentTag+1; // generate a new tag for this query
   Get Voxel V_seed which contains x_q
   Clear(neighbours);
   ProcessVoxel(V_seed, x_q, neighbours)
  }
ProcessVoxel(V, x_q, neighbours):
  {
   // first: tag current voxel...
   V.tag = currentTag;
   // second: process all photons in current voxel
   for each photon p ∈ V:
      if (||p − x_q||_2 <= neighbours.MaxDist())
         neighbours.Process(p);
   // finally: recursively process the neighbouring voxels
   for each neighbouring voxel V' to V :
      if (MinDistance(V',x_q) <= neighbours.MaxDist()
          and V'.tag != currentTag)
         ProcessVoxel(V',x_q,neighbours)
  }
```

Already visited voxels have to be tagged in order to prevent multiple processing of voxels. The implementation is quite easy, but lengthy, and so the code is omitted.

After processing the seed voxel, the order in which the following voxels are processed is almost arbitrary, since a good choice of K requires only few voxels to be processed. An implementation based on least-cost-search ([Mad], see section 'Branch and Bound') was also tested. This branch-and-bound method processes voxels in order of their distance to x_q and yields additional performance gains if many voxels have to be processed in a query, i.e. when the number of photons per voxel is small and k is high. In the general case, however, few voxels have to be scanned, and the higher overhead in finding the next voxel offsets the improved culling.

As can be seen in the following table, the GridPhotonMap yields considerable performance gains over the kd-tree, especially for large photon maps. Performance gains highly depend on the parameters N, k and K. Therefore, a method for choosing K optimally as a function of k and N would be very helpful.

N	k	3d-Tree	Grid		
			K=30	K=35	K=40
100k	10	70s	51s	51s	52s
	100	274s	271s	229s	242s
1M	10	155s	119s	108s	111s
	100	567s	305s	295s	307s

In our experiments, the following observations have been made:

1. The impact on pure query time is much larger, since rendering times contain some overhead not affected by faster query-times, i.e. ray-intersections and shading of the found photons make up about 50% of the time.
2. Initialization time is negligible with both methods.
3. Performance is parameter-dependent.
4. The more photons used, the greater the speedup.

Multilevel-methods such as using a coarse grid on the base, together with kd-trees or finer grids in overly filled voxels should further improve performance (especially when photons are unevenly distributed) while requiring only minor modifications.

2.5 The Photon Map Shader

The photon map shader is built into an existing distribution ray tracer, the McRender project (see [Kel98]) as a shader for local illumination. Shaders are methods which are called by the distribution ray tracer to calculate the diffusely reflected radiance L_r. The distribution ray tracer ([CPC84],[Coo86],[SW92]) calculates L by splitting it into several terms and calculating these separately: As stated before, L can be expressed as

$$\begin{aligned} L &= L_e \\ &+ T_{f_{rs}} L \\ &+ T_{f_{rg}} L \\ &+ T_{f_{rd}} L. \end{aligned}$$

The last term - $T_{f_{rd}} L$ - describes diffusely reflected indirect illumination, and is the only term actually calculated by the photon map shader. All other terms are handled by the distribution ray tracer: L_e is given by the objects material definition and requires no further calculations. Specular reflections $T_{f_{rs}} L$ are calculated by simply following the reflected ray and recursively calling the distribution ray tracer. Glossy reflections $T_{f_{rg}} L$ require Monte Carlo integration and can be calculated efficiently by importance sampling proportional to f_{rg}. Here too, the samples are calculated by recursively calling the distribution ray tracer. This recursive spawning of rays results in lots of shader-calls and yields high rendering times when having highly glossy scenes like MMACK or INVISIBLEDATE. Low sampling rates for glossy reflections introduce noise into the image (see chapter 5), while higher sampling rates may manifold rendering times. To reduce this effect, glossy reflections are approximated by less time-consuming specular or diffuse reflections (depending on their strength), at least after reaching a certain recursion depth.

In each recursive invocation of the distribution ray tracer, the diffusely reflected radiance $L_r = T_{f_{rd}} L$ is calculated by calling the photon map shader: Having the discrete density approximation $(x_i, \omega_i, \phi_i)_{i=0}^{N}$ of the irradiance, the reflected radiance L_r can be evaluated by using a sensor A as

$$\begin{aligned} L_r = T_{f_r} L &= \int_\Omega f_r(\omega, x, \omega') L_i(x, \omega') cos\theta \, d\omega' \\ &\approx \frac{1}{supp\, A} \int_A \int_\Omega f_r(\omega, x', \omega') \left(\sum_{i=0}^{N} \delta(x' - x_i, \omega' - \omega_i) \phi_i \right) d\omega' dx' \\ &= \frac{1}{supp\, A} \sum_{i | x_i \in A} f_r(\omega, x_i, \omega_i) \phi_i. \end{aligned}$$

Assuming that all photons have passed through x, we can approximate $f_r(\cdot, x_i, \cdot) \approx f_r(\cdot, x, \cdot)$. Using the ball which contains the k nearest neighbours to x yields the radiance estimate from the photon map as

$$L_r(x_q, \omega) \approx L_{est}(x_q, \omega) := \frac{\sum_{p \in Q(x_q, k)} f_{rd}(\omega, x_q, \omega_p) \phi_p}{\pi \, r^2},$$

where $Q(x_q, k)$ is the set of the k nearest neighbours to x_q and r is the radius of the query-ball:

$$r = max \{ \, ||x_q - p_x|| \, | \, p \in Q(x_q, k) \, \}.$$

This radiance estimate generates several artifacts, which are examined in the following chapter. As opposed to the terms handled by the distribution ray tracer, the estimate from the photon map is only a rough approximation, which - if N is fixed in advance - does *not* converge to the correct result by taking more samples.

To enable independent treatment of caustics and diffuse illumination, singular and regular photons are stored in different maps as mentioned earlier. These maps are called the *caustic map* and the *diffuse map*. The implementation is quite simple:

```
Color ShadePhotons(Intersection intersection, PhotonMap map, int k)
{
  PhotonList p;
  f_rd = intersection.Object.bsdf.f_rd;
  p.Clear(k);
  map.Query(p, intersection.x);
  return (Σ_{i=1}^{k} f_rd(intersection.ω, intersection.x, photon[i].ω) photon[i].φ) / (π · photon.MaxDist()^2)
}

Color PhotonMapShader::Shade(Intersection intersection)
{
  return ShadePhotons(intersection, diffuseMap, k_diffuse)
       + ShadePhotons(intersection, causticMap, k_caustic);
}
```

In calculating L_r, the photon map shader may also incorporate such extensions as the separate calculation of direct light or the local pass. These extensions, however, will themselves use the above methods for calculating the estimate.

In [Jen96a], Jensen utilizes the photon map to get a coarse approximation of global illumination. Based on the influence of the sample on the image, heuristics are used to decide whether the estimate from the photon map is used or whether a more complex computation is done. If the contribution of the sample is small, i.e. because it is only a single sample ray in the local pass or because it is attenuated by a small brdf, then the photon map approximation is used directly. Otherwise, if the ray is a primary ray or the point was reached by several perfect reflections, the point is considered important, and the shading is done by more accurate Monte Carlo integration, which in fact is a local pass. This notably improves image quality, but is very time-consuming and can only be applied under certain conditions. Jensen applies Ward's irradiance caching ([WH92]) on lambertian surfaces which achieves faster rendering times by interpolation.

Because of the fact that the radiance estimate is only used as an approximation in more correct Monte Carlo integration, its quality is far less important. In this case 'blurring can even be seen as an advantage' ([Jen96a]) for reducing noise. Artifacts in the estimate are partly smoothed away by the local pass and partly dominated by direct light. However, a closer look often reveals remains of these artifacts, mainly noise and blurring, especially in caustics, where the radiance estimate has to be visualized directly. Artifacts like energy-bleeding or overmodulation did not appear in the original papers because of suitably modelled scenes.

Direct illumination is calculated using shadow photons, as discussed later. This, however, is also only applicable under certain conditions. Caustics are treated separately: They are rendered directly by the photon map. Jensen admits that parameters have to be adjustet to find a tradeoff beween blurred caustic borders and noise in the caustics (see [Jen96b]: "If the parameters are badly chosen, the caustic will either become too blurred or too noisy"). To improve caustics, many more photons are used to represent caustics than are used for diffuse illumination. Jensen uses projection maps ([JC95b]) to increase the number of caustic photons by emitting photons directly from the light sources towards specular objects. This method is only capable of producing direct caustics, but performs well under this limitation.

Chapter 3

Analysis of the Photon Map

3.1 Efficiency of the Photon Map

The efficiency of the photon map is believed to depend only on the efficiency of the radiance estimate, which mainly depends on the number of photons used, and which is otherwise independent of scene complexity and geometry. This is proposed as one of the main features of the photon map, making it seemingly possible to render highly complex scenes at moderate cost.

However, though it is true that the largest portion of the rendering time is spent in the radiance estimate (querying and shading of the photons), the overall time spent on the estimates is the product of the time spent per estimate and the number of estimates evaluated. Therefore, the "efficiency of the photon map" depends on both the efficiency of the estimate *and* on the efficiency of the rest of the renderer in keeping the number of estimates small. Therefore, the overall rendering time depends on 3 factors:

- efficiency of the distribution ray tracer in keeping the number of shader calls small
- efficiency of the shader in keeping the number of estimates small
- efficiency of the estimate.

The argument of efficiency beeing independent from scene complexity only applies to the last item. The efficiency of the distribution ray tracer into which the photon map shader is built (see section 2.5) has great influence on rendering time: If the scene is highly complex, rendering time will increase because of more expensive ray intersections. Additionally, lots of specular and glossy objects will spawn lots of recursive calls to the distribution ray tracer, manifolding the number of shader calls independently from the photon map.

The second factor is the time spent in the photon map shader: Using a local pass manifolds the number of estimates. Independent calculation of direct light may also increase rendering cost under certain circumstances, mainly when having many light sources. These two items imply that realistic scenes - which are highly complex scenes with several light sources and lots of specular and glossy objects - will require high rendering times independently from the number of photons used (see discussion on MMACK in chapter 5).

Rendering cost for the photon map estimate also consists of a preprocessing part (*RandomWalk* and *BuildTree*), which can be considered to be constant, since it is not affected by the number and complexity of shader-calls. Preprocessing cost is generally small compared to rendering time: Each photon requires exactly one ray intersection in the random walk step, whereas the distribution ray tracer generates 4 million rays alone for the primary rays in a PAL-resolution image with an oversampling rate of 8 sample per pixel. However, several of the later proposed algorithms will require higher preprocessing times. When using a kd-tree or a regular grid, the time for building the acceleration structure is negligible.

The cost of the radiance estimate is small even with moderately sized photon maps (up to about 250.000 photons). The dominating cost in the estimate is the cost for finding the k nearest neighbours to x. As Jensen correcly states, increasing the number of overall photons N (while keeping k fixed) increases rendering cost but slightly. This is verified in the following table, and is due to the fact that the cost for the query is closely related to the depth of the tree (which is $1 + \lceil log\, N \rceil$): Take, for example $T(N = 100k, k = 100)$=391s and $T(N = 1M, k = 100)$=614s, where 10-folding the number of photons results in a mere 55% increase in rendering time. (Note that for each parameter pair in the table, exactly the same number of rays are shot).

(in sec)	k=10	k=100	k=1000
N=100k	72	391	1823
N=1M	156	614	4413

Contrary to this, Jensen's conclusion that the resolution of the photon map could be increased without notable increase in rendering time could not be completely verified, since increasing N requires increasing k by the same factor to yield the same image, since changing the $N : k$ ratio affects the final image (see figure 3.1 in section 3.3.1). Therefore, rendering cost will very well increase when using more photons, since influence of k on $T_{Estimate}$ is at least linear. In the above example $T(N = 100k, k = 100)$=391s would become $T(N = 1M, k = 1000)$=4413s. When using a local pass, k can be kept small when increasing N, since changing the $N : k$ ratio will then not change the final image as visibly; so Jensen is partially correct in this case.

3.2 Advantages of the Photon Map

The photon map has several advantages over many other illumination algorithms. Since most of these advantages are already sufficiently published in the original papers, they are only briefly summarized here:

- High quality: Incorporates almost any illumination features, including caustics and volumetric effects
- Flexibility: Allows for the use of arbitrary object types and arbitrary bsdfs
- Adaptive to local illumination density
- Efficiency (of the estimate) is independent of scene geometry
- Simplicity of code
- Easily understandable

The photon map - as proposed in the original papers - is generally able to render high-quality images of highly complex scenes while using only a moderate amount of resources. The photon map is even able to produce complex illumination features like caustics in a realistically looking way which are otherwise very expensive to compute. The high quality of the rendered images mainly results from the fact that the photon map is able to produce almost any illumination feature: Some features are due to the distribution ray tracer, which creates effects resulting from specular or glossy reflections. Note that lens effects such as motion blur or depth-of-field are also due to the distribution ray tracer. Diffuse illumination and caustics are rendered by the photon map shader. The photon map is also capable of rendering volumetric effects such as fog or volume caustics (see [JC98]), which are not covered in this work.

One of the main advantages over other rendering algorithms is that the photon map is able to handle arbitrary illumination models as well as arbitrary object types - objects even don't have to be parameterizable. This allows even for such complex representations as recursively, fractally, implicitly or procedurally defined objects. Additionally, the photon map does not require the scene to be tesselated (as opposed to finite element methods, mainly Radiosity methods), which enables the photon map to render complex scenes with very compact scene representations. The photon map is also generally capable of using arbitrary bsdfs, which was not tested in our implementation.

As compared to finite element methods, the discrete density automatically adapts to local changes in the illumination density. This eliminates the problem of finding a suitable tesselation, which has always been a major problem for these methods. This is especially useful for representing caustics, which have high locality.

Another advantage of the photon map is that the time spent in the estimate is independent from scene complexity: With the same number of photons, increasing scene complexity will only increase the time spent for ray-scene-intersections, while most of the time is spent in nearest-neighbour-queries and the radiance estimate, which are not influenced by scene

complexity. Note that in reality, however, increasing scene complexity would require more photons to be used to achieve comparable quality, and would also increase rendering time due to the distribution ray tracer, as documented in section 3.1. Therefore, this advantage could not be fully verified.

The simplicity of the photon map allows for a quick implementation. After the rather technical implementation of the data structures and query-algorithms, the code for the random walk and the estimate is easily created. Since distribution ray tracers are already commonly used, this part is often given in advance. This allows for the photon map to be easily integrated into any existing distribution ray tracer. Flexibility and simplicity of the photon map code also allow to use photon map algorithms and data structures in secondary applications, for example for the efficient sampling of direct light (section 4.2.3) or for getting an estimate for the importance in a scene (section 4.7).

3.3 Disadvantages of the Photon Map

While there are already several publications about the photon map in general, none of these sufficiently analyzes the flaws of the photon map method. Therefore, we will first give a detailed description of these problems, which consist of two classes: While the photon map's artifacts - which are investigated later on in this chapter - are most visibly to the viewer, the photon map also has several intrinsic problems: One of these is that the photon map highly depends on the parameters for the estimate (see section 3.3.1). Secondly, the forward simulation used in the random walk is problematic under certain circumstances, which is covered in section 3.3.7. Some of the already commonly applied extensions like direct light or local pass are also only applicable in suitable settings, see the discussion in the respective sections 4.2 and 4.3. Finally, the efficieny of the photon map highly depends on the efficiency of the distribution ray tracer, as already sufficiently discussed in section 3.1.

These problems may result in very poor performance in more realistic scenes.

3.3.1 Dependence on Parameters

The quality of the estimate highly depends on the parameters, mainly the number of overall photons N and the number of queried neighbours k. While it is clear that increasing N affects the quality of the estimate, the influence of k is less obvious. The critical parameter was found to be the ratio of $N : k$: Using smaller values for k will require fewer neighbours to be queried, resulting in a smaller query-ball and increasing low-frequency noise in the image. On the other side, increasing k produces large query-balls, taking photons from a larger volume into account and therefore results in increased blurring. So, the choice is either blurring or noise (see image 3.1).

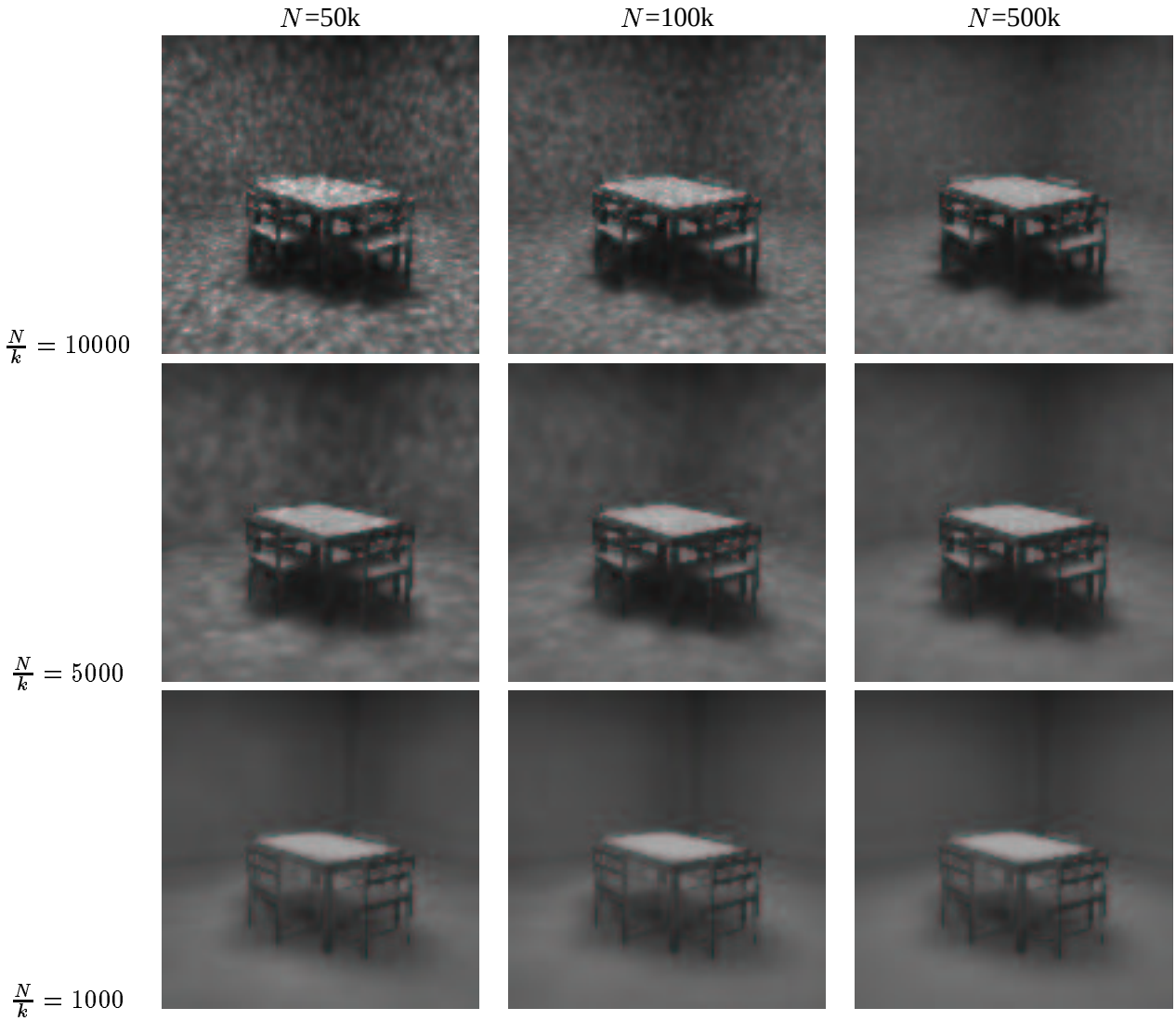


Figure 3.1: Sample images with different parameters N and k. Note the increased blurring of the shadows with decreasing $N : k$ ratio.

The 'optimal' $N : k$ ratio depends on the scene to be rendered. While $10.000 - 50.000 : 1$ was found to be a reasonable value in most of our expirements, this value is subject to change with several other scenes: In the INVISIBLEDATE scene (see chapter 5), only a small fraction of the photons reaches the room to be rendered. Using the same $N : k$ ratio as in SCENE6 would result in a huge query-ball in the rendered room.

The ratio is less critical when using a local pass, since both blurring and low-frequency noise are reduced by the local pass. Even then, smaller k's will increase Monte Carlo noise in the local pass, requiring more sample rays to be evaluated.

Larger k values will increase other artifacts like energy-bleeding, visible bands at geometry borders or overmodulation (see later in this chapter). These artifacts will negatively affect the quality of the local pass.

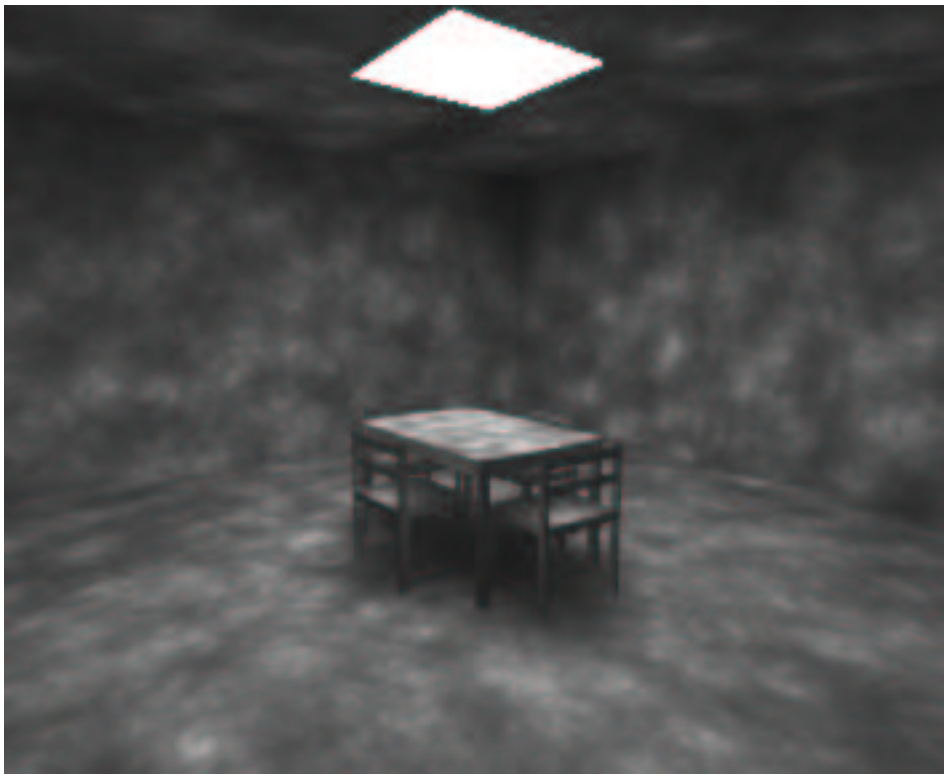
The tradeoff between noise and blurring always applies to caustics, where no local pass can be applied (see [Jen96b], 'if the parameters are badly chosen, the caustic will either become too blurred or too noisy').

3.3.2 Low-Frequency Noise

Low-frequency noise is the first-encountered and probably most perceivable artifact of the photon map. It appears when using too coarse a resolution of the photon map and too small a k for the estimate (see previous section).

When using other ray-based rendering algorithms (raytracing, BDPT, metropolis, ...), variance can be reduced by using more samples per pixel, which have to be independent of each other. This independence of samples is no longer assured when using the photon map: Similar rays result in hitting the scene in similar positions, yielding queries at similar positions. Because of the static nature of the photon map - once the discrete density is created, it will never change - queries in similar positions will produce very similar - if not the same - neighbour sets[1]. Therefore, the radiance estimate varies but slowly, leading to the low frequency of the noise.

To get independent samples, each query should result in a new candidate set. This cannot be realized because of the limited amount of memory: Rendering a PAL-resolution image - about 800 by 600 pixels - with 10 samples per pixel, one gets $800 \times 600 \times 10$ or about five million primary rays. Using a fixed $k = 100$ and assuming the photons to be optimally placed, one would need at least 500 million photons to assure independent samples for the primary rays alone. 500 million photons would need about 10 Gigabyte of memory and huge rendering times. In fact, the number of photons needed would even be much higher.

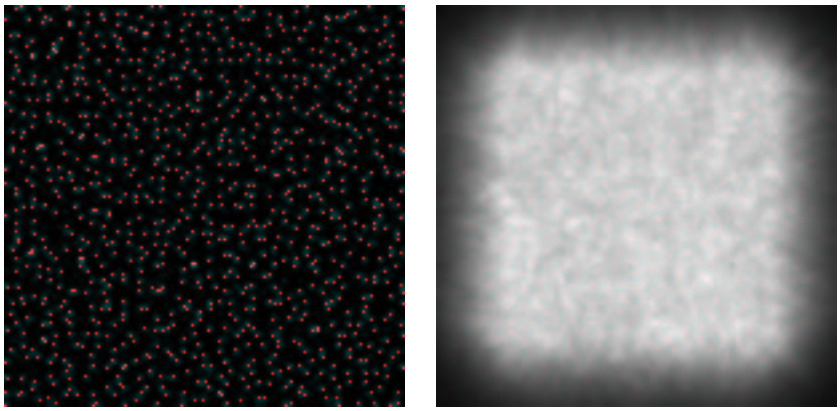


Since lower frequencies are easier perceivable to the eye than higher frequencies, this kind of noise is especially perceivable. Increasing k decreases the frequency of the noise, blurring it, but not removing it. The noise in the estimate can also be reduced by increasing the photon density, which will also increase rendering time and memory requirements (see section 3.1). Using a local pass completelys remove low-frequency noise from diffuse illumination, but also increases

[1] In fact, the volume of the scene could be tesselated into finitely many voxels in which each point has the same neighbours.

rendering time and is applicable only in certain settings. Noise can also be reduced by averaging samples from multiple independent maps, see section 4.4.

It can be shown that low frequency noise is not affected by the uniformity of the distribution of the photons. To demonstrate this, we created photons directly on a $[0,1]^2$ plane, and calculated the radiance estimate on this plane. All photons have equal energy and direction, their location was chosed according to several sampling patterns, from purely random to patterns with increased uniformity, including jittered, n-rooks-sampling and the halton sequence. One of these patterns is depicted in the left image of the following figure. The right image shows the 'rendered' image and demonstrates the noise created by the estimate. The dark bands around the border of the image are due to a wrong area estimate as discussed later. Note that this setting is closely related to the table top in SCENE6.

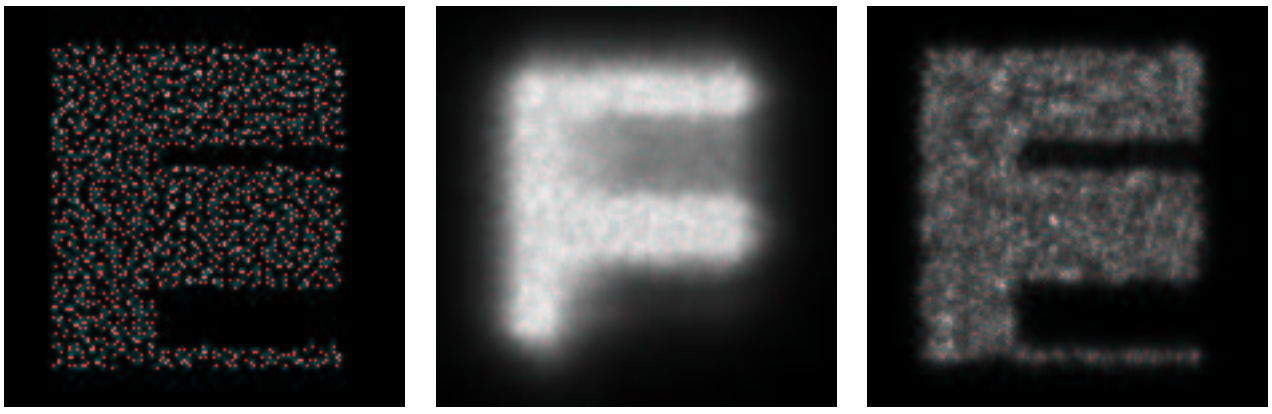


position of photons visualized radiance estimate

3.3.3 Blurring

Blurring results from the fact that an entire volume - the volume of the query ball - is used for the radiance estimate of a single point. Therefore, the higher the number of photons for the estimate, the larger the query ball and the higher the blurring effect. In consequence, blurring can be decreased by using less photons for the estimate, which in turn increases noise in the image. In [Jen96a], Jensen states that 'the blur in the photon map is actually an advantage since it reduces noise in the final gathering step...'. However, increased blurring not only requires higher rendering-times (because of increased k, see section 3.1), it also blurrs correct discontinuities like sharp caustics or shadow borders. Therefore, blurring can hardly be stated as an advantage.

As it was done in the previous section, blurring can also be demonstrated by visualizing the radiance estimate of an explicitly generated pattern. This time, the photons are not located in the unit square, but on a simple polygon:



position of photons radiance estimate (large k) radiance estimate (small k)

The left image shows the positions of the photons, from which the original figure can be easily recognized. The center image demonstrates several artifacts when using a large number of photons for the estimate: Parts of the figure disappear

because of an extremely overestimated area (see section 3.3.6). Additionally, the upper 'shadow' is extremely blurred. The image on the right shows how blurring and dark bands can be diminished by querying fewer photons, which also increases noise.

Loss of Sharp Shadows The most perceivable effect of blurring is the loss of sharp shadows, since the radiance estimate is unable to produce sharp shadow borders: Querying k photons in a shadowed area grows the query-ball until it contains k photons, however far away and however reasonable these may be (also see energy-bleeding, section 3.3.4). The energy of these k photons across the shadow border is therefore blurred into the shadowed area, resulting in a soft falloff of energy instead of a sharp shadow border (see sketch in figure 3.2, effects in figure 3.3).

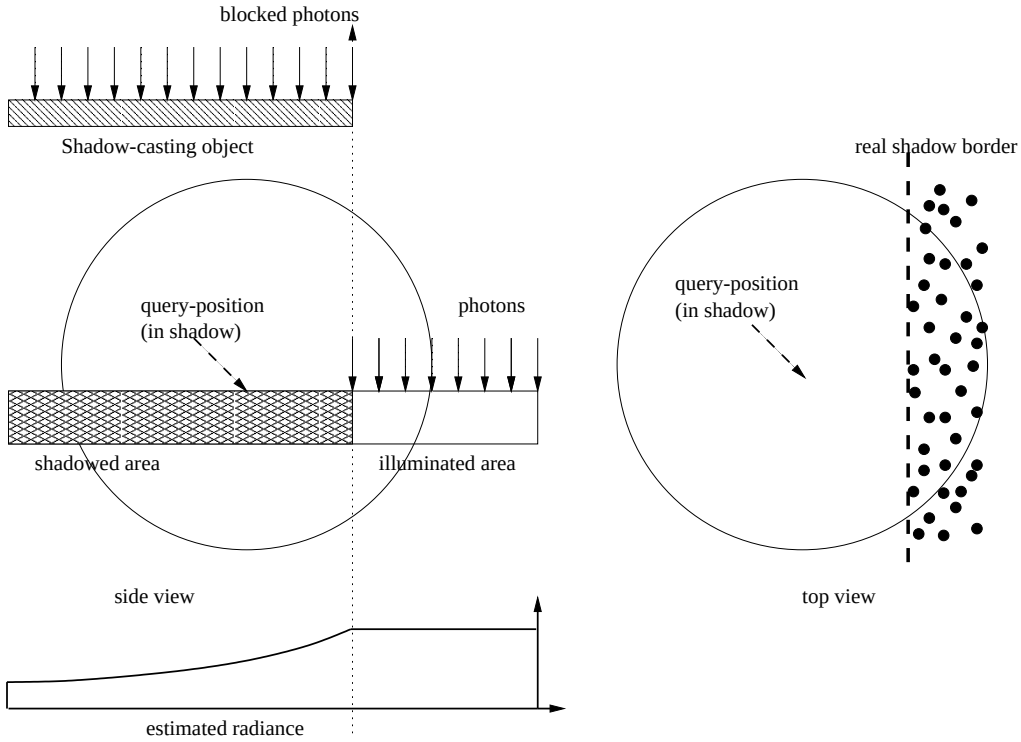


Figure 3.2: Blurring of shadow borders: The query ball is extended until it contains the required number of k photons, even if the query-position is in complete shadow.

In the extreme, this leads to a complete removal of some shadows: If the shadow-casting object is relatively small or narrow, i.e. the leg of a table or the thin lamp in INVISIBLEDATE, the shadow completely disappears when using large k's for the estimate (see figure 3.3).

For the later discussion in section 4.2, shadows have to be classified into 2 kinds: *direct shadows* and *indirect shadows*. Direct shadows are shadows resulting from direct illumination. In many scenes, these are the dominating shadows, and can often be rendered efficiently by separate calculation of direct light. Indirect shadows are shadows resulting from indirect light. Since indirect illumination is generally less directional than direct illumination, indirect shadows will appear only in special cases, for example shadows in a caustic or when having a scene illuminated only indirectly by a small, highly energetic area like the slit created by the opened door in the INVISIBLEDATE scene. Then, this area behaves almost like a light source (see [CRMT91]), casting indirect shadows.

Indirect shadows can *not* be handled by separate calculation of direct light.

Blurred Caustics Borders For caustic borders, the same argumentation holds as for shadow borders. Queries close to a caustic will extend the ball into the caustic, blurring the caustic border. Blurring of caustics is generally harder to handle than shadows: Since caustics do not result from direct light, they cannot be improved by separate calculation of direct light. Neither can they be handled by a local pass. Quality of caustics can only be improved by using more caustic photons, see section 4.6.

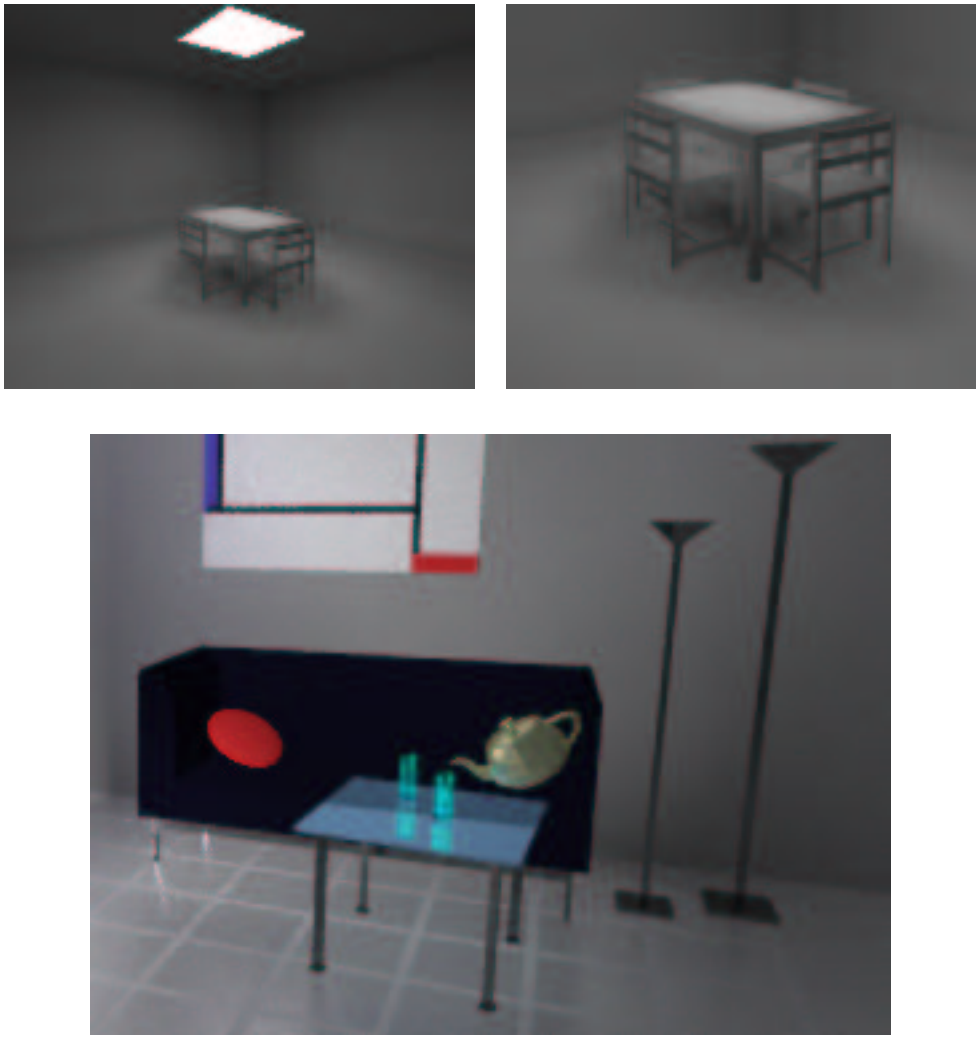


Figure 3.3: Loss of shadows: Shadows under the table and on the chairs in SCENE6 are completely smoothed away. For a description of the depicted scenes, and some more correctly rendered images, see chapter 5. The lower image shows INIVISIBLEDATE. Here too, shadows from table, sofa and lamps (as can be seen in a Metropolis-rendered master image in section 5) are missing completely. These shadows are indirect, and can not be rendered by separate calculation of direct light.

Other Effects of Blurring Blurring may also appear in other forms, i.e. by blurring the color. This did not appear in our scenes, but could be seen in several other papers, even in very simple settings. Energy bleeding is also closely related to blurring.

3.3.4 Energy-Bleeding

Energy-bleeding is an effect where energy contributes to the illumination of a location from which it is physically blocked by geometry. It is a consequence of not regarding any geometric or topologic relations in the estimate: A query at position x returns the k closest photons to x without knowing whether these photons really 'belong' to the query position in a physical sense or not (e.g. whether they are blocked by a wall as sketched in figure 3.4). This effect can be reduced by better plausibility tests, which will, however, introduce new problems (see discussion in section 4.1). Energy-bleeding is closely related to blurring, with the only difference that it is usually much stronger, and that energy will even be 'blurred' through blocking geometry, making it much more perceivable than 'normal' blurring, especially in the local pass.

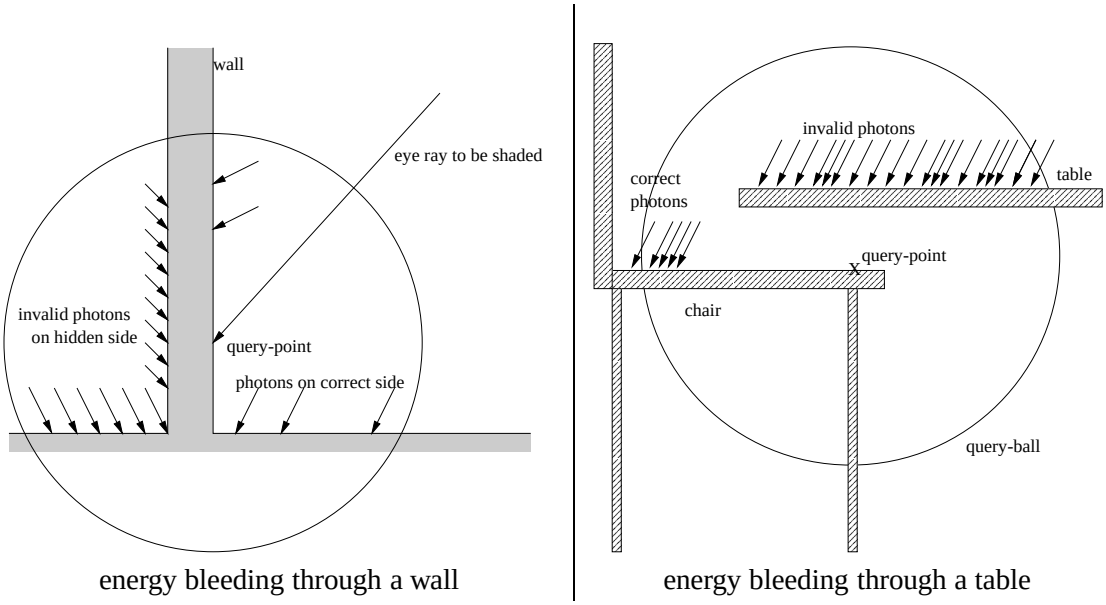


Figure 3.4: Energy bleeding: A query on a wall which is highly illuminated from the hidden side.

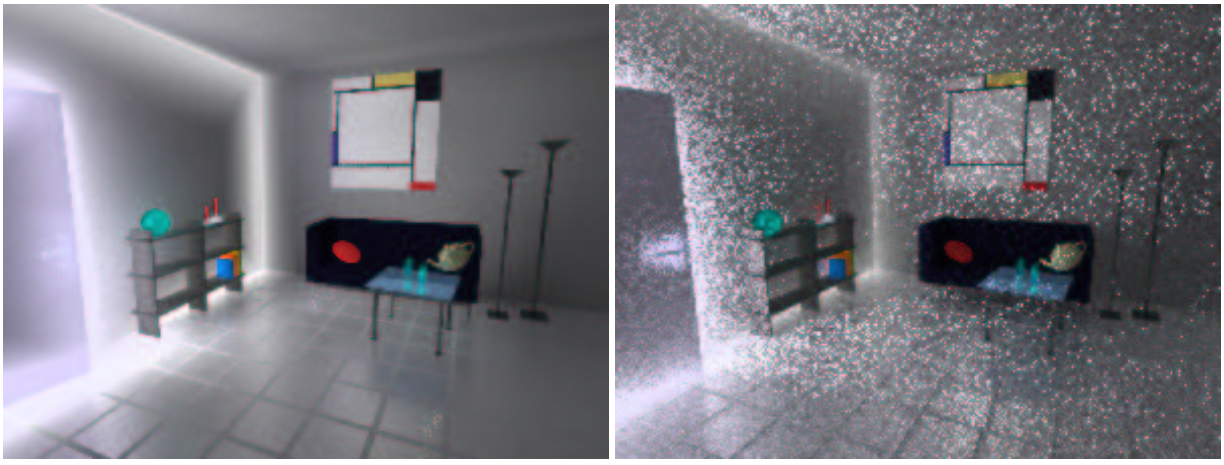


Figure 3.5: Effects of energy bleeding: Energy from the backside of the wall bleeds through around the edges of the wall (left side, smoothed by the skipping method). This artifacts even persists when using a local pass, as demonstrated on the right. Energy-bleeding is also responsible for the excessive amount of noise in the local pass, even after over 6 hours of rendering time.

In the INVISIBLEDATE scene, for example, energy bleeding produces an entirely highlit wall, if no steps are taken to prevent this. When testing surface normal and photon direction, this effect still appear at the edges of the wall, when photons on the floor and ceiling are found (see figure 3.5). Energy-bleeding is still problematic even using a local pass, as can be seen in the right image. Energy-bleeding also appears in smaller scale, for example even in such an simple setting as SCENE6: When determining the radiance estimate on the chairs with a large k, the query-ball grows until it contains some of the photons from the top of the desk. Then, energy from the top of the table bleeds through to the chairs, as sketched in figure 3.4. A method was implemented which removes energy bleeding (see section 4.1.1); its cost, however, is extremely high, and can only be justified for such severe cases as INVISIBLEDATE.

3.3.5 Overmodulation

Overmodulation is an artifact which appears as an overly illuminated area. It is due to the fact that the division by $radius^2$ results in a large radiance estimate when radius gets small. This often happens in areas where large clusters of photons are stored, e.g. because of geometry which is close to light sources, like the polygonal 'exit' letters on the exit sign in figure 3.6. Instead of limiting overmodulation to the lightsources, the blurring effect extends such overmodulated parts of the scene over larger areas.

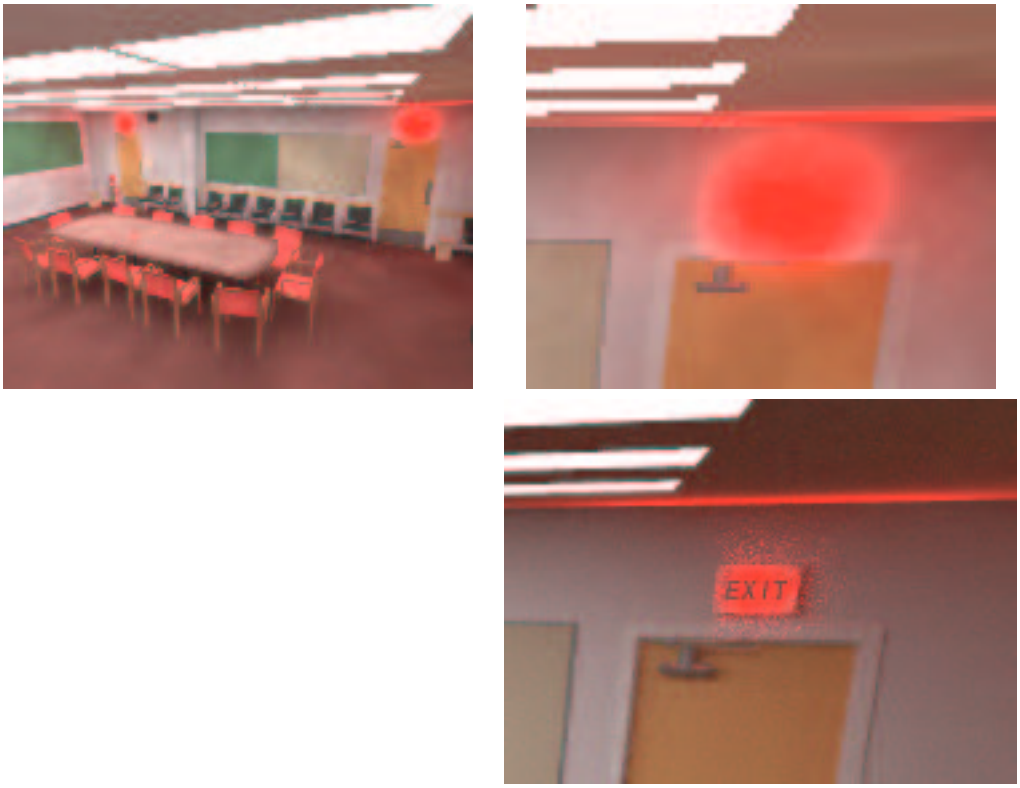


Figure 3.6: Effects of overmodulation. The high photon density at the light source leads to an entirely overmodulated area. In the lower image depicts the area as it looks when overmodulation is removed by a local pass.

3.3.6 Artifacts Resulting From a Wrong Area Estimate

Artifacts in the estimate sometimes appear as highlighted or darkened bands at geometry borders and edges, i.e. at around the borders of the table top or in the edges of the room, as can be seen in figure 3.8. These artifacts are due to a wrongly estimated area: Since the radiance estimate is in fact a density estimation problem, an estimate for the enclosed surface area is needed. This is approximated as the area of the query-ball projected onto a plane and is therefore πr^2, resulting in the previously used radiance estimate of $\frac{\sum \phi}{\pi r^2}$. SCENE6 is an especially good example for the influence of the area estimate: Here, the random walk results in all photons having about the same energy, and the change in the radiance estimate results solely from the change of query ball's radius. Because of this strong influence of the area estimate, a wrong area estimate implies a faulty radiance estimate.

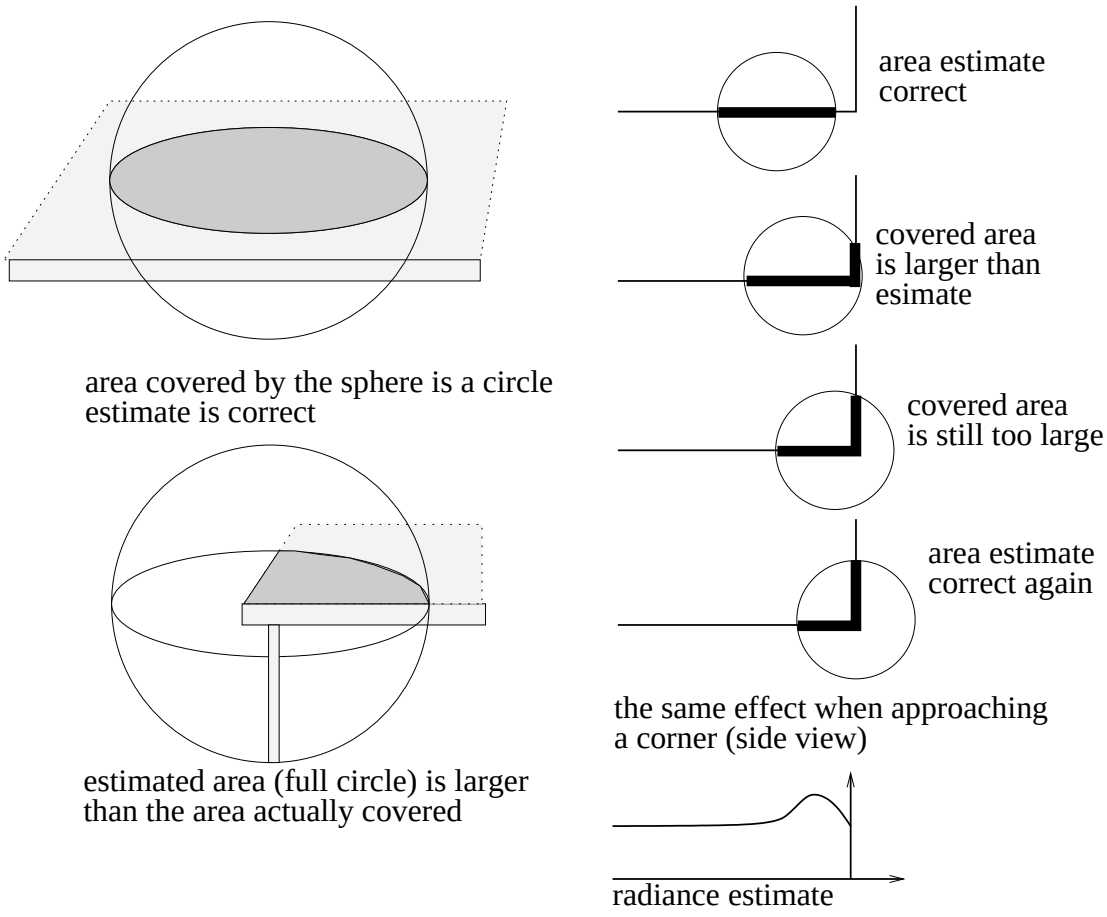


Figure 3.7: Wrong estimation of covered area at geometry borders and edges.

The area estimate is correct whereever the sphere fully covers a flat surface area, but will highly overestimate the area (and therefore underestimate radiance) at geometry borders, where the area actually covered is only a fraction of a circle (see sketch in figure 3.7). This results in a visible falloff of energy near geometry borders, as can be seen in figure 3.8. The opposite case happens if the sphere covers a surface larger than πr^2, i.e. if the sphere covers both the floor and a table leg, or both a side wall and the floor (also sketched in figure 3.7).

SCENE6

enlarged table borders

enlarged room edges

Figure 3.8: Visible bands at geometry borders and edges. The reasons for both artifacts are sketched in figure 3.7.

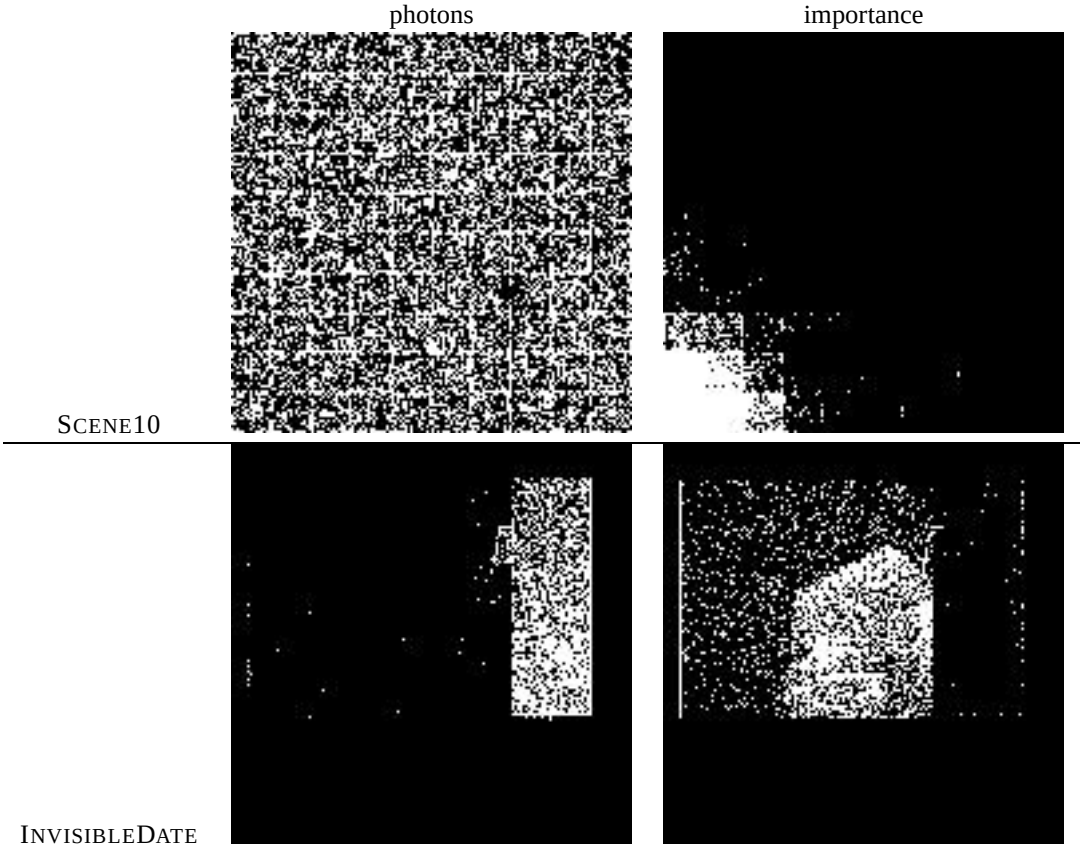
In fact, the same argumentation applies to any non-flat area: In [JC98], Jensen refers to this as “The ... estimate is valid as long as the surface is locally flat”. Since the query-ball is not infinitesimally small, the area in the query-ball very often is not flat, especially in areas with highly complex geometry such as fractally or procedurally defined objects, which are proposed as one of the main features of the photon map.

Note that the width of the bands is equal to the radius of the sphere, and can therefore be diminished by using a smaller k value if the increased noise can be tolerated.

3.3.7 Problems arising from the Forward Simulation in Generating the Photons

In the original papers, the photons are generated by random walk as shown in section 2.2. This kind of random walk is a pure forward simulation, generating the photons without taking the different *importance*[2] in different areas into account. In the INIVISIBLEDATE scene (see chapter 5), this leads to a large number of photons in the room with the lightsource (several thousand photons in the lower example), while only a small fraction of the photons (in our example, less than 50) will actually reach the room to be rendered. This means that the majority of the photons is used to approximate the radiance in the 'unimportant' neighbouring room, at very high density, while the photon density in the really important room is very low. This implies that a huge number of photons has to be used to get a reasonably high photon density in the interesting areas (see the following figure). The same applies to SCENE10: Here, the photons are created with equal density in all of the 100 rooms, while only a small fraction of these photons is actually used in the rendering stage.

The problem can be solved by using importance driven methods for generating the photons, see section 4.7



The figure depicts the unsuitable distribution of photons that arises when doing a pure forward simulation without taking importance into account. SCENE10 consists of an array of 10×10 equal, interconnected rooms, each with a light source. Here, all rooms are approximated with equal density, while samples from only few rooms are actually used in the rendering stage. The left image shows the locations of the photon - as seen from the top - while the right image depicts the importance in this scene. In INVISIBLEDATE, photon density is almost inversely related to the importance: The right room with the light source is approximated at very high density, while the left room with the viewer receives only a very small fraction of the photons.

[2] We will restrict ourselves to an intuitive definition of the term *importance*. A more profound definition can be found in [PP98].

Chapter 4

Photon Map Algorithms

One method of using the photon map is using it in a standalone version, by directly visualizing the radiance estimate. As stated in the previous chapter, this generates several artifacts, or otherwise requires huge photon map resolutions. Therefore, extensions of the photon map are needed in order to improve the quality of the image. One straightforward way for improving the standalone photon map is the separate calculation of direct light, which often makes up for more than 50 percent of the illumination in simple settings. Several algorithms for the separate calculation of direct light are covered in section 4.2. An also commonly used 'improvement' is the local pass, which is discussed in section 4.3. Even for these well-known methods - direct light and local pass - a thorough investigation is undertaken, showing both their strengths and their weaknesses. This chapter also provides some additional extensions, which are not yet publicly applied and which aim at removing the problems discussed in the previous chapter.

4.1 Testing queried Photons on their Feasibility

As stated in the previous chapter, several of the artifacts result from simply querying and shading the k nearest photons to x_q, even if these photons are not reasonable. Since the photon map does not contain any geometric or topologic information, the feasibility of a photon can not be evaluated with absolute certainty. This leaves only some heuristic plausibility tests to prevent shading of unreasonable photons.

These tests can be incorporated into either the query or in the shading step. In the first case, photons are stored only if they are presumed feasible. The second solution is to first do a query for all k-nearest neighbours, and then filtering out only the feasible ones before shading. The first method always returns k reasonable photons, while the second method may - in the extreme - discard many, if not all, of the found photons. On the other hand, the second method will perform exactly k plausiblity tests, which are much less than in the first method, where any visited photon has to be tested. Therefore, the second method is the method of choice if very expensive tests such as occlusion-testing 4.1.1 have to be performed. Less time-consuming tests should be applied directly in the query step.

Such tests, however, have to be used very carefully. They often appear plausible, but still remain only heuristics: Most of such tests are reasonable in the general case, but may utterly fail in other environments (see figures 4.2 and 4.1).

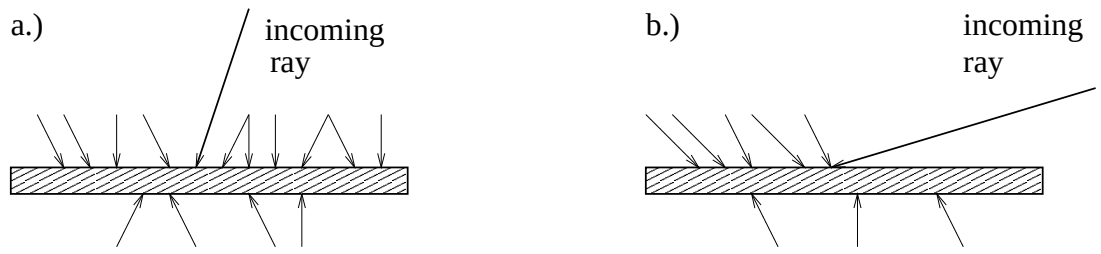


Figure 4.1: A simple plausibility test: Testing the angle $\alpha = (\omega_{ray}, \omega_{photon})$ to be below 90 degrees is simple and seems promising in case a. However, it would be a bad classification in case b. Similar cases can be constructed for other tests.

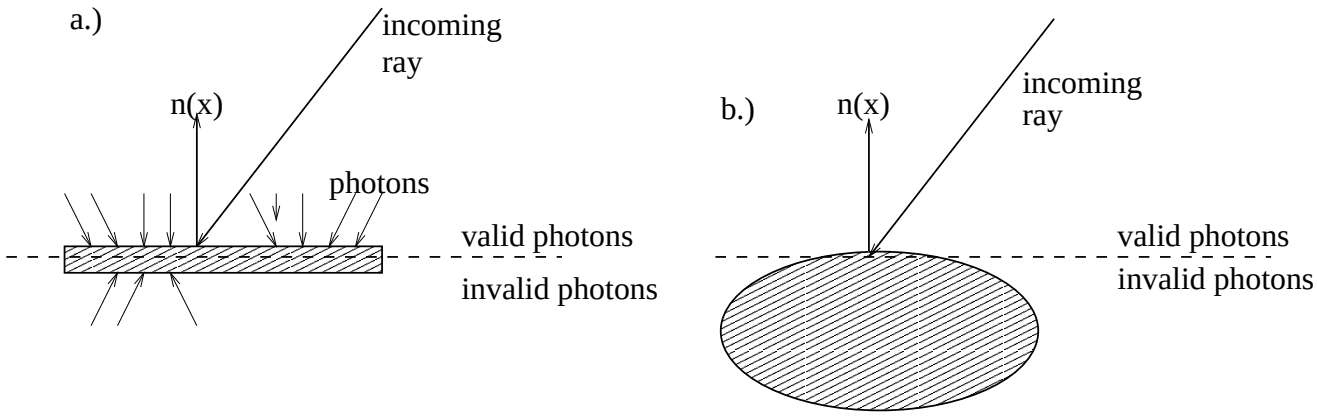


Figure 4.2: Specifying a culling plane by query position and surface normal (dashed line) yields anotherds another plausibility test. Considering photons 'behind' this culling plane invalid, illumination from the backside (as demonstrated in section 3.3.4) can be prevented on flat surfaces (as sketched in case a), but may be fatal on curved surfaces like in case b, where almost all photons on the ellipsoid are considered invalid.

In our implementation, only photons with a 'suitable' angle between photon direction and surface normal are considered feasible to reduce illumination from the backside. In INVISIBLEDATE, this test prevents photons from the opposite side of the wall from beeing found, but still results in finding photons on floor and ceiling of the neighbouring room, which often have 'correct' directions (see sketch in figure 3.4). A similar test is testing photons to be 'in front of' the query-position. This correctly culls all photons on the other side of the wall, but fails on edges or on convex objects, as sketched in figure 4.2.

Another plausibility test, especially for caustic photons, is to specify a maximum search radius and a certain threshold, and then discarding the found photons if their number is below the specified threshold (i.e. if less than 10% of the queried number could be found). This test is capable of removing the artifacts created by stray caustic photons (see discussion in section 5), therefore removing the noise created by these photons. However, it remains of very limited value in the theoretical sense, since energy is lost with this method.

As a result, it can be stated that certain plausibility tests may improve visual appearance in suitable cases. However, since these tests are mainly based on heuristics and cases of bad classifications can be easily constructed, they are of limited value. Therefore, our implementation only tests photon direction against surface normal.

4.1.1 Occlusion Testing

Section 3.3.4 and figures 3.4 and 3.2 showed how the radiance estimate produces artifacts whenever photons are found in the query which do not physically belong to the point to be shaded. This results in blurred shadow-borders and energy-bleeding through surfaces. As stated before, plausiblity tests which are based on heuristics always remain of limited value. Therefore, a more physically motivated approach was tested, by verifying whether a queried photon could physically have reached the query point: In the random walk step, each photon is tagged with the position from which it originated. In the shading step - after the query - visibility is tested between the query-point x_q and the origin (not the position !) of photon p:

$$L_{est}(x_q, \omega_q) = \frac{1}{\pi\, r^2} \sum_{p \in Q(x_q,k)} V(Origin(p), x_q)\, f_r(\omega_q, x_q, \omega_p)\, \phi_p,$$

where

$$V(x,y) = \begin{cases} 1; & \text{if x is visible from y} \\ 0; & \text{otherwise.} \end{cases}$$

This test is very expensive, since for each query and for each photon found in a query, an occluder test has to be performed. Even increasing the cost is the fact that $V(Origin(p), x_p) \equiv 1$, making it very probable that $V(Origin(p), x_q)$ is also

true, since x_q and x_p are very similar. Many methods have been developed to accelerate occluder-tests when occlusion is very likely (see, e.g. the discussion in [JC95a]). The opposite case - where occlusion is unlikely - is considerably more complex (i.e. shaft culling), making occluder tests almost always as expensive as shooting a ray, in our implementation. This justifies the method applicable only in severe cases.

Occlusion testing is very expensive, but may result in significantly increased quality (see figure 4.3), by preventing energy-bleeding and blurring of shadows. Faster occlusion-testing algorithms would reduce rendering cost. The method not only removes energy-bleeding, it also produces direct and even indirect shadows. Note that this approach is very similar to Instant Radiosity, as presented in [Kel97].

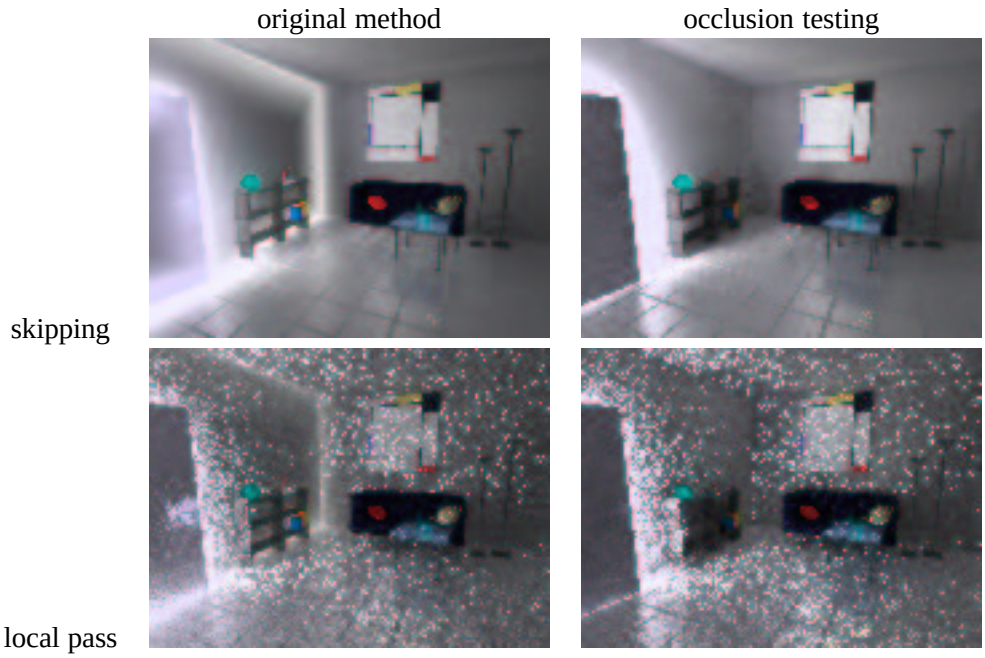


Figure 4.3: Impact of occlusion-testing on INVISIBLEDATE. For the skipping method, rendering time for INVISIBLEDATE on a DEC Alpha/333 was increased from $6\frac{1}{4}$hours to about 15 hours. The method also improves the local pass. Here, rendering time was increased from 6 to 9 hours.

4.2 Separate Calculation of Direct Illumination

As shown in section 2.1, L_r can be expressed as

$$L_r = T_{f_r} L \quad = \quad \underbrace{T_{f_r d} L_e}_{L_D} + \underbrace{T_{f_r d} T_{f_r d} L}_{L_I} + \underbrace{T_{f_r d} T_{(f_{r s} + f_{r g})} L}_{+L_C},$$

where L_D is called direct illumination, L_I is soft indirect illumination and L_C are caustics. Indirect illumination and caustics can be approximated by the photon map as before, with the only modification of only considering photons which had at least one interaction with the scene. Direct illumination

$$L_D = T_{f_r d} L_e$$

is of a relatively simple form, and can be calculated separately at high quality. In simple scenes, direct light makes up 50 to 70 percent of the illumination and is mainly responsible for shadows, which are often direct shadows. Calculating direct light separately will then visibly increase image quality by reducing low-frequency noise and enhancing direct shadows.

Another advantage of calculating direct light separarately is that direct photons need not be stored, thus noteably reducing the number of diffuse photons, leaving room for a higher density in approximating indirect illumination.

In scenes dominated by direct illumination, artifacts are reduced considerably (see figure 4.4): Except for Monte Carlo noise, artifacts in the diffuse illumination are removed completely. Since the remaining indirect illumination often is relatively weak, the artifacts in the indirect illumination are then dominated by the correct direct illumination. This makes 'weaker' artifacts (such as visible bands) almost inperceivable, especially when using textures or rendering noisy environments such as fog, pools or underwater scenes (as in [JC98]). In several cases, artifacts like low-frequency noise or overmodulation can still be seen in areas where indirect illumination dominates (see, for example, figure 4.5).

In scenes dominated by indirect illumination (e.g. INVISIBLEDATE or BATHROOM, see chapter 5), separate calculation of direct light is a waste of effort. Note that this may happen relatively easy in 'unsuitably modelled' scenes. In the BATHROOM scene, for example, though it seems to be mainly *directly* illuminated, light has to pass through the glass spheres surrounding the light bulbs before reaching the scene, making all illumination *indirect.* The same effect could be obtained by simply covering the lightsource in SCENE6 with a glass plate. This limits separate calculation of direct illumination to 'suitably modelled' scenes, and may render it futile in more realistic scenes.

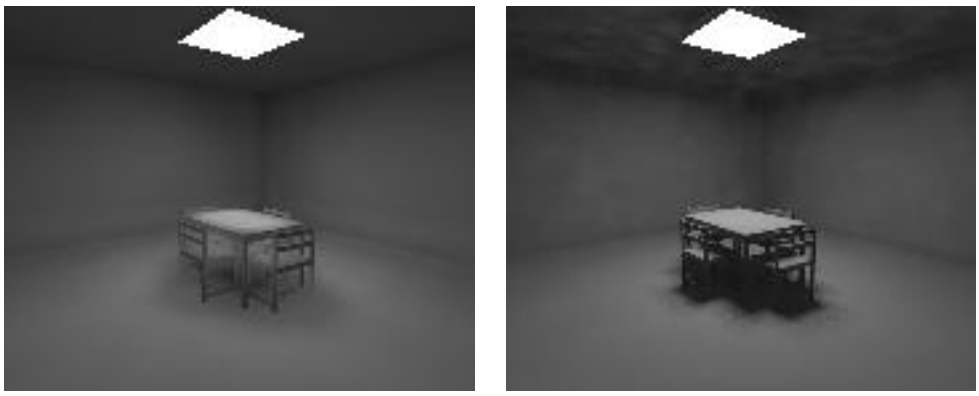


Figure 4.4: Improvements due to independent calculation of direct light. Note the sharper shadows in the right image, especially on the chairs.

Figure 4.5: This image shows a zoomed part of the ceiling, where low-frequent noise still persists.

4.2.1 Monte Carlo Integration

Since direct illumination $L_D = T_{f_{rd}} L_e$ consists mainly of the evaluation of an integral with a known integrand L_e, it can be calculated by Monte Carlo integration as proposed in [SWZ96] and [War91a]. This method is well-known and already accepted as a standard method for calculating direct light in photorealistic rendering: Let $V(x,y)$ denote the visibility function, i.e.

$$V(x,y) = \begin{cases} 1; & \text{if x is visible from y} \\ 0; & \text{otherwise,} \end{cases}$$

and let

$$G(x,y) = \frac{cos\theta_x \, cos\theta_y}{||x-y||_2^2}$$

denote the geometric term, where θ_x and θ_y are the angles between $x-y$ and the surface normals $\hat{n}_x$ in x respectively $\hat{n}_y$ in y. Then, direct illumination $L_D(x,\omega) = (T_{f_r} L_e)(x,\omega)$ can be written as

$$\begin{aligned} L_D(x,\omega) &= \int_S f_{rd}\left(\omega, x, \frac{x-y}{||x-y||_2}\right) V(x,y) G(x,y) L_e\left(y, \frac{y-x}{||y-x||_2}\right) dy \\ &= \int_{LightSources} \underbrace{f_{rd}\left(\omega, x, \frac{x-y}{||x-y||_2}\right) V(x,y) G(x,y) L_e\left(y, \frac{y-x}{||y-x||_2}\right)}_{=: L_s(x,\omega,y)} dy, \end{aligned}$$

since $L_e(x) \equiv 0$ for any point x not on a lightsource. Denoting the integrand by $L_s(x,\omega,y)$, this can be evaluated by Monte Carlo integration as

$$L_D(x,\omega) \approx \frac{1}{N} \sum_{i=1}^{N} L_s(x,\omega,y_i)$$

where the y_i are i.i.d. samples on the lightsources.

Variance can be reduced by various variance-reduction techniques as presented in [Kel98] and [SWZ96]. Stratification yields

$$L_D(x,\omega) = \sum_{L_i \in LightSources} \frac{1}{N_i} \sum_{j=1}^{N_i} L_s(x, \omega\, y_{ij}), \tag{4.1}$$

where N_i is the number of samples on source L_i and the y_{ij} are i.i.d. samples on L_i. Variance can be furtherly reduced by importance sampling, i.e. by choosing N_i proportional to the contribution of source L_i. Since this contribution is not known beforehand, the N_i are chosen to be proportional to the emittance of L_i, which often is a reasonable approximation in simple settings. Other variance reduction methods - for example Quasi Monte Carlo methods for sampling the light sources - can be applied as well.

However, as Jensen states: “This approach is very simple, but also very time consuming” [JC95a]. Monte Carlo sampling performs very well in several cases, but results in poor performance if the number of light sources is large. Each sample requires the evaluation of a *shadow ray* ($V(x,y)$), which basically requires a ray to be shot. This makes cost prohibitive when rendering large numbers of light sources. Even as several acceleration methods for shadow rays exist (see the discussion in [JC95a]), high sampling rates and lots of light sources will still require high rendering times, due to the large amount of shadow rays to be evaluated (see table 4.1). If most of the sources are hidden, most of the effort is wasted, since samples will contribute no information if $V(x,y) = 0$, and so the aim of all optimizations must be to avoid casting shadow rays that yield $V(x,y) = 0$.

Shadow rays per sample	RMS_{MC}	time (sec)
100	0.030	436
200	0.022	821
400	0.016	1507
800	0.013	2875

Table 4.1: Monte Carlo sampling (SCENE10, 8 samples per pixel). Measures are taken on a DEC Alpha/333. Note that in SCENE10, shooting 100 shadow rays generates only 1 sample per lightsource.

4.2.2 Shadow Photons

As stated before, the dominating factor in the computation of direct illumination is the cost for the evaluation of the shadow rays. In [JC95a], Jensen proposes the use of shadow photons as a means for the fast calculation of direct illumination by reducing the number of shadow rays. The method is not applied in our implementation, and is therefore only summarized here but briefly for the sake of completeness: In a preprocessing step, shadow photons are created by tracing rays from light sources through the entire scene. At the first intersection point a 'standard' direct photon is stored, just as in the original random walk. At each of the following intersection points a shadow photon is created. In this way, Jensen obtains three types of photons:

1. direct standard photons
2. indirect standard photons
3. shadow photons

This information is then used to optimize the calculation of direct illumination. In order to calculate direct light at position x, a query for the nearest neighbours to x is done. Then, the number n_s of shadow photons and the number n_d of normal direct photons among these neighbours is counted. Based upon these two numbers, three classifications of x can be obtained:

1. $n_d = 0 \rightarrow$all photons are shadow photons $\rightarrow$x is completely shadowed
2. $n_s = 0 \rightarrow$all photons are illumination photons $\rightarrow$x is completely illuminated
3. $n_s \neq 0$ and $n_d \neq 0 \rightarrow x$ is close to a shadow border.

This classification is used as an heuristic to reduce the number of shadow rays. In case 1, no shadow rays have to be shot, since the point is assumed to be completely shadowed anyway. In the second case, the visibility function $V(x, y)$ is assumed to be 1, and the contribution from the light source is added directly. Shadow rays have to be shot only in the third case. Even in this case, direct light can be approximated without shooting shadow rays by $L_D \approx \frac{n_d}{n_s+n_d}$. Jensen found out that about 90% of the shadow rays could be saved in his experiments.

However, Jensen also outlined potential drawbacks of this method, mainly the large number of shadow photons in certain settings, and the possibility of artifacts. In complex environments, each ray started on a lightsource will have a large number of intersections with the scene, each generating a new shadow photon. This leads to a large number of shadow photons, and - worst of all - a large ratio of shadow photons to direct photons. Storing about 10 to 20 times as much shadow photons as normal photons is prohibitive. Additionally, complex objects tend to create large clusters of shadow photons, increasing the probability that all k neighbouring photons are shadow photons, even if the point is illuminated. Another effect is the introduction of artifacts into the scene: Due to the probabilistic nature of the method, small shadow-casting objects may be missed by direct photons, therefore not generating shadow photons and resulting in a missing shadow. Even large photon densities might not resolve this problem. The heuristic also makes the method biased. The last problem is that the method is highly impractical for large numbers of light sources: This needs an excessive amount of shadow photons, and each light source requires independent treatment. Because of these reasons, we considered the method impracticable for realistic scenes. It is therefore not applied in our implementation.

4.2.3 Importance Sampling using Photon Map Information

In the previous sections, it was stated that the time for calculating direct light can be reduced by saving shadow rays to hidden sources. As the usage of shadow photons introduces several problems, we developed a different method - in the sequel called PhotonDirect - to save shadow rays by applying information from the photon map.

As stated in section 4.2.1, variance can be reduced by importance sampling, i.e. by choosing the number N_i of samples per source according to the importance of source S_i. The importance of source S_i to point x is equal to the amount of direct illumination received from source S_i, which is not known beforehand, but may be estimated by applying the photon map. Therefore, direct photons are tagged with its emitting lightsource[1].

To calculate direct light, a query is done among the direct photons. N_i is then chosen proportional to E_i, where E_i is the energy of the photons from source S_i among the found photons:

$$E_i = \sum_{p \in Q(x_q) \mid source(p)=i} \phi_p,$$

which is similar to the radiance estimate for direct light. This yields

$$N_i = \frac{E_i}{\sum_{i \in LS} E_i} N_{contributing},$$

where $N_{contributing}$ is the number of samples shot to the light sources from which photons were found. Direct light is then estimated as in equation 4.1. Due to the random nature of the method, some sources might be missed; therefore, at least one sample has to be shot to the sources from which no photons were found to make the algorithm unbiased. This prevents artifacts from bad classifications, as may arise when using shadow photons. Let LS_{con} (LS_{non}) be the set of all sources which did (did not) contribute photons to the nearest neighbour set. Then, direct illumination is calculated by

$$L_D \approx \sum_{L_i \in LS_{con}} \frac{1}{N_i} \sum_{j=1}^{N_i} L_s(x_{ij} \in L_i) + |LS_{non}| \frac{1}{N_{non-cont}} \sum_{j=1}^{N_{non-cont}} L_s(z_j \in L_{non}).$$

This method is useful when having a large number of lightsources, and when almost any point is illuminated by only few sources, as is the case in several realistic settings (see, for example, SCENE10). Then, the shadow rays can be concentrated to the really important lightsources. Under the assumption of having few sources illuminating any point, almost certainly all of these sources will be found in a query, resulting in all samples of the last term becoming zero. Therefore, few samples have to be shot to these non-contributing sources, saving almost all shadow rays to these sources. However, if not all contributing sources are found in the query - which is likely if a point is illuminated by many different sources - the samples of the last term tend to create noise, because they are weighted by $|LS_{non}|$, which is large[2] (see example image below).

[1] The extra memory could be avoided in some rather technical ways.

[2] The effect can be softened by mixing both MC-sampling and PhotonDirect, i.e. using the higher sample rates on the contributing sources just as in PhotonDirect, but shooting exactly one sample to each non-contributung source - then weighted by 1.

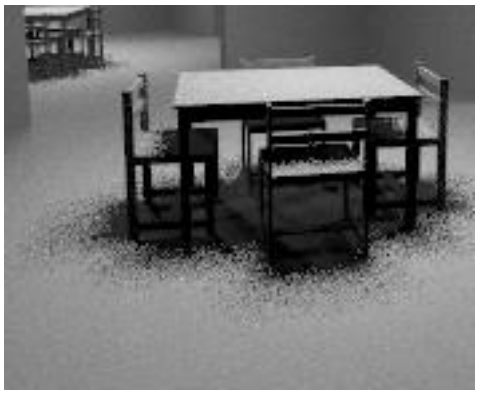
The probability of this kind of noise is correlated to the probability of not finding photons from all light-contributing sources. This probability increases with the number of sources illuminating x_q and decreases by increasing the photon map density and the number of queried photons. This yields two results:

1. Noise can be removed by using more photons
2. The method performs best when having a large number of light sources, but where most areas are only illuminated by few sources.

Instead of the time spent for the shadow rays, the limiting factor of the PhotonDirect method is the time spent in the queries - about 50 to 90 percent of the time is spent there. The resolution can be kept relatively low even for such complex scenes - the above examples have been rendered with only 100.000 photons in SCENE10, resulting in only about 1000 photons per room. The main disadvantage of the PhotonDirect method is its dependence on parameters, as is the case with all photon map methods.

The method yields several opportunities for improvements: Caching, for example, can be easily implemented by simply using the neighbourhood of the previous query if the new query point is 'close enough' to the previous one. Since the quality of the estimate does not affect correctness, caching can be applied here, and is highly effective: In several tests, caching the photons has reduced rendering time for direct illumination in SCENE10 by about a factor of 20! Further work, however - especially on the impact of caching on the quality - has not yet been undertaken. Another possible improvement is to use importance driven photon maps as presented in section 4.7, to reduce the number of photons.

Even though correctness of the method - in the sense that its expected value is correct - can be shown quite easily, proving its superiority over pure Monte Carlo sampling is considerably harder. This is therefore only 'demonstrated on the example' in the following images: Both images took about 430 seconds on a DEC alpha/333 .



Monte Carlo Sampling
100 samples

PhotonDirect
20+1 samples, 50/10.000 photons

The improvements can also be measured by the RMS-error as shown in the following table. Measures are taken with SCENE10, 8 samples per pixel, on a DEC Alpha/333. 100+1 samples means taking 100 samples from the important sources and 1 sample from unimportant sources. Rows 2 and 3 show how the same quality can be reached with different parameters, at huge differences in cost. Note that the parameters are not optimally chosen and that caching has not yet been used in these experiments.

settings (k,N)	samples	$RMS_{PhotonDirect}$	time (sec)	time to achieve same quality with Monte Carlo sampling
50/10.000	10+1	0.018	322	1155
	100+1	0.016	1212	1507
100/100.000	10+1	0.016	926	1507
	100+1	0.013	1863	2875
300/100.000	10+1	0.013	1352	2875
	20+1	0.011	1450	(not measured)
	50+1	0.009	1755	(not measured)

4.3 Local Pass Integration

As shown before, radiance can be expressed as

$$L = L_D + L_I + L_C,$$

where soft indirect illumination L_I is

$$L_I = T_{f_{rd}} T_{f_{rd}} L.$$

Noting that $T_{f_{rd}} L = L_r$, and knowing that $L_r \approx L_{est}$ as stated before, L_I can be approximated by

$$L_I \approx T_{f_{rd}} L_{est},$$

which can be evaluated by Monte Carlo integration:

$$\begin{aligned} L_I &= \int_\Omega f_{rd}(\omega, x, \omega') L_r(h(x, \omega'), -\omega') \cos\theta \, d\omega' \\ &\approx \int_\Omega f_{rd}(\omega, x, \omega') L_{est}(h(x, \omega'), -\omega') \cos\theta \, d\omega' \\ &\approx \frac{1}{N_{lp}} \sum_{i=1}^{N_{lp}} f_{rd}(\omega, x, \omega_i) L_{est}(h(x, \omega_i), -\omega_i), \end{aligned}$$

where the $\omega_i \in \Omega$ are i.i.d. samples on the hemisphere. Variance reduction techniques such as in [Kel98] can be applied. The implementation is straightforward and requires only minor changes to the photon map shader. Because of the smoothing property of the integral, applying the local pass eliminates many artifacts. Blurring and noise in the integrand are averaged away, and visible bands and overmodulation are reduced (see figure 4.6).

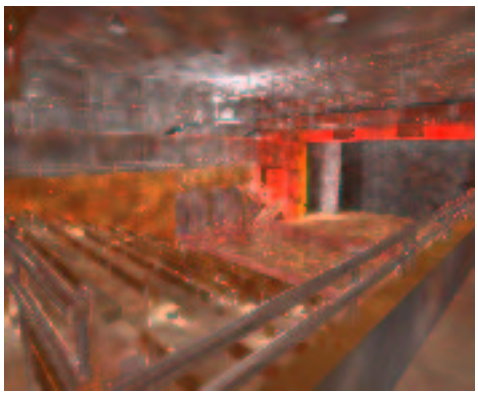
Figure 4.6: Overmodulation removed by the local pass. Note the noise around the lightsource, where more rays are likely to become overmodulated.

However, the local pass also has several drawbacks: One drawback of the local pass is that it is limited to diffuse illumination, so caustics have to be rendered directly with the estimate. As a result, artifacts still appear in caustics, i.e. caustic borders will always be smoothed out to a certain extent. This creates problems in scenes with many singular objects, for example partially reflective tiles as in INVISIBLEDATE: Large singular areas will create 'caustics' almost everywhere in the room, rendering the local pass ineffective. The same applies to BATHROOM, where all photons are classified as caustic photons after passing the glass spheres surrounding the lightsources.

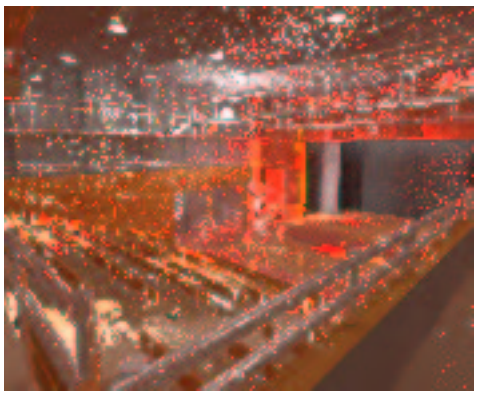
The worst drawback of the local pass is the dramatic increase in rendering time. Having N_{lp} as the oversampling rate for the local pass, instead of one evaluation of $L_{PhotonMap}$ (one query and one shading of the photons), N_{lp} such evaluations have to be computed, together with N_{lp} additional rays to be shot and N_{lp} additional queries to be performed.

This effect is increased by the observation that the local pass creates Monte Carlo noise in the image if the photon map estimate has high variance, for example because of overmodulated areas. This noise can only be diminished by using large oversampling rates N_{lp}, making the cost prohibitive. Jensen uses Wards irradiance caching [WH92] to accelerate his local pass on lambertian surfaces. Irradiance caching is not used in our implementation.

Artifacts in the estimate may sometimes persist even when using a local pass, since integrating an erroneous integrand will generally not yield the correct value for the integral. In INVISIBLEDATE, energy-bleeding around the borders of the wall (see left image in figure 4.7) is still partially visible, since in these regions, about every second sample hits an overly illuminated area. Also note the excessive amount of noise created by sample rays hitting the overmodulated door and wall corners, even after huge rendering times.



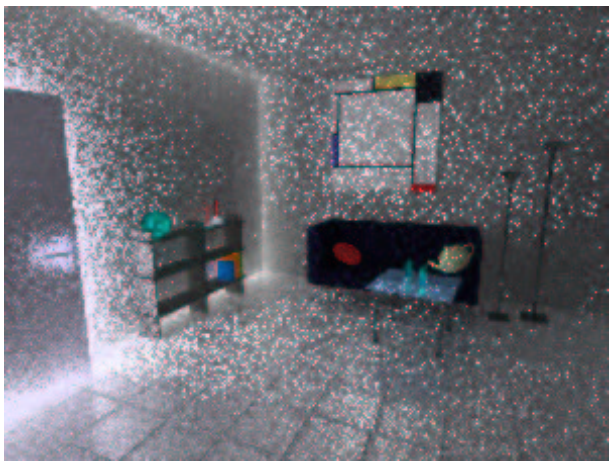
standalone photon map, 37 minutes



local pass, 20 hours

As a result, it can be stated that a local pass produces absolutely smooth, high quality images in scenes where the radiance estimate varies but slowly, but creates noise in more realistic settings. Rendering cost for removing the noise becomes prohibitive in these settings.

without local pass



with local pass

Figure 4.7: Artifacts of the local pass which are due to energy bleeding: Energy bleeding sill persists around the corners of the wall, and creates large amounts of noise.

4.4 Multiple Photon Maps

Since image quality depends on the number of photons used, it was tested how image quality is affected when organizing these photons in more than one map. Therefore, we stored the photons in M different maps, each created by its own, independent random walk. The radiance estimate then is taken as the mean of the M different estimates

$$L_{est}(x,\omega) = \frac{1}{M}\sum_{j=1}^{M} L_{est,j}(x,\omega) = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{\pi\, r_j^2}\sum_{p\in Q_j(x,k)} f_{rd}(\omega,x,\omega_p)\phi_p.$$

Experiments have shown that the achieved quality only depends on the overall number of photons used (see the following table). For example, the error of an image rendered with 1M photons equals the error of an image rendered with 10 maps of 100k photons each. Since memory cost is the same and efficiency decreases when using more maps, this method alone is of rather limited value. It only proves that using more maps yields the same quality as using more photons, which is a prerequisite for later methods.

Nr of photons	RMS-Error	time (sec)	Nr of photons	RMS-Error	time (sec)
$1 \times 10k$	0.033	40	$10k$	0.032	39
$4 \times 10k$	0.018	157	$40k$	0.019	141
$16 \times 10k$	0.012	655	$160k$	0.013	516
$64 \times 10k$	0.010	2751	$640k$	0.011	2207

An improvement can be reached by not using several smaller maps at the same time, but by using several large maps in turn, using approximately the same resources: When rendering an image, several samples are taken for the evaluation of a single pixel for the purpose of antialiasing (oversampling). As stated in section 3.3.2, these samples are not truely independent, since all of these samples are taken from the same, rigid photon map, creating low-frequency noise. If all of these samples are taken from different maps, independence is assured and low-frequency noise is reduced. This can be achieved by first generating a map of N photons for the first sample of each pixel. Then, this map is discarded, and a new map is generated for the second samples, and so on. Since each sample is taken from a map of N photons, quality and query-time for each sample is the same as with the old method. Memory cost also remains the same, because at any given time exactly one map is stored in memory. The only additional cost results from the $s-1$ additional random walks, which is both constant and relatively small compared to rendering time, at least in complex settings. With this method, the image is rendered using $s \times N$ photons effectively, which reduces low-frequency noise. Other artifacts are not affected.

```
for (i=1..s) {
   create new map P of N photons
   for each pixel x,y
      color(x,y) += 1/s Sample(x,y,P)
}
```

The method can also be implemented by averaging s single images, each rendered with N photons and oversampling 1, yielding a source for parallelization or increased interactivity through intermediate results. The impact of this method is illustrated in figure 4.8. An increase in quality can also be shown by measuring the RMS error:

photons	Single Map: 1 map, oversampling=4	Multiple Maps: 4 maps in turn, each os=1
100k	0.070	0.047
200k	0.043	0.026
1M	0.017	0.011
5M	0.010	0.007

The low frequency of the noise in the estimate not oly affects the independence of samples in the same pixel, but also the independence of samples in neighbouring pixels. Therefore, this scheme of using more maps can be carried even further by using different maps for neighbouring pixels: Now, the first map is not taken for the first sample of every pixel, but only for 1 sample in an $S \times S$ pattern of pixels:

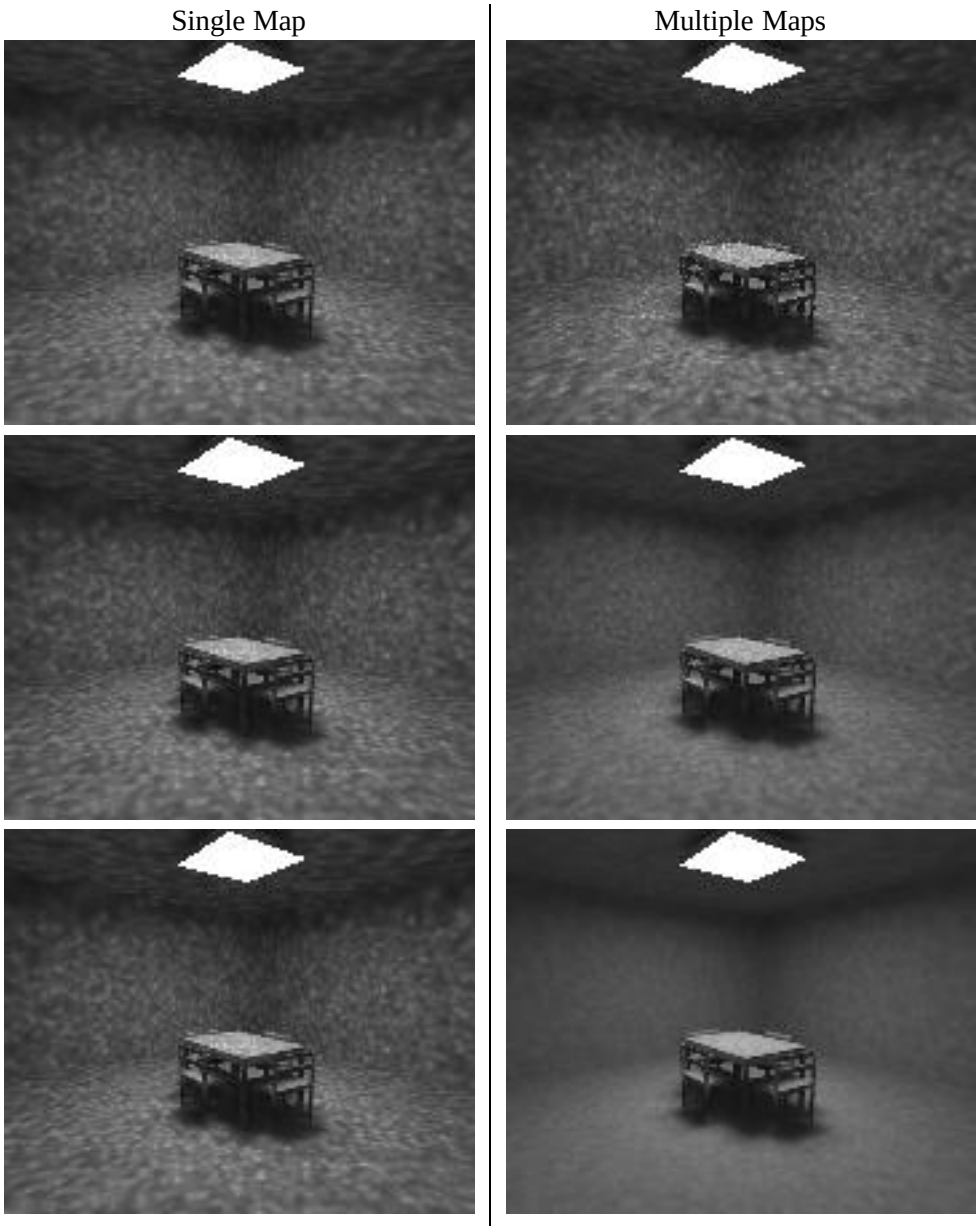


Figure 4.8: Impact of using multiple maps: The right row shows the use of 1, 4 and 16 maps and has the same memory usage as the left image with oversampling values of 1, 4 and 16 respectively. Up to some constant cost for the additional random walks, the images on the right also have the same rendering times as the corresponding images on the left.

```
for (i = 1 ... s)
  for (x0 = 0...S - 1)
    for (y0 = 0...S - 1) {
      create new map P of N photons
      for each pixel (x * S + x0 , y * S + y0)
         color(x,y) += 1/(s S^2) Sample(x,y,P)
    }
```

To prevent the $S \times S$ pattern to appear in the image, the method was randomized by using the same map for every S^2th

pixel in a randomized way. The method was called skipping, because when sampling one map, about $S^2 - 1$ out of S^2 pixels are skipped:

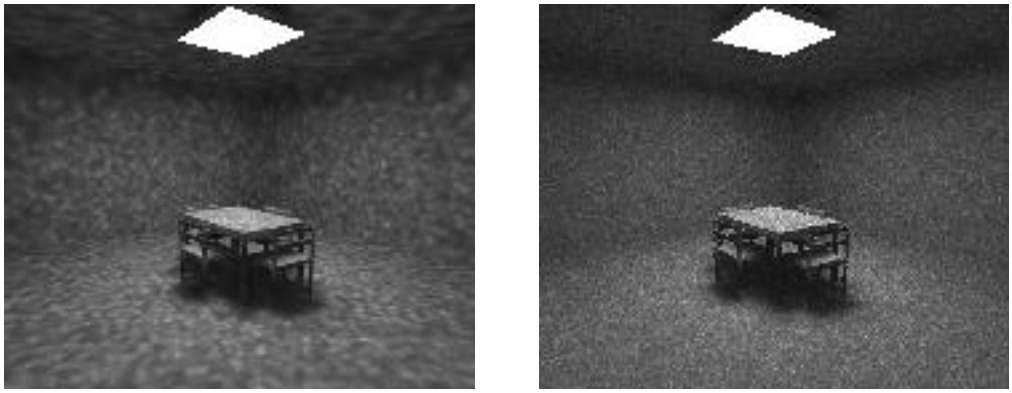
```
for (i = 1...s) {
  sigma = GetRandomPermutation(SizeX × SizeY);
  for (startindex=0..S^2 − 1) {
    create new map P of N photons
    for (index=startIndex; index < SizeX*SizeY; index+=S^2) {
    x = sigma[index] % SizeX;
    y = sigma[index] / SizeX;
    color(x,y) += 1/s Sample(x,y,P)
  }
}
```

Since only every S^2th pixel is now computed with the same map, low frequency noise is removed, creating Monte Carlo noise - which has higher frequency - in its place. Even though the following table shows that the method does not improve RMS error, figure 4.9 demonstrates the improved visual appearance, since Monte Carlo noise is much less perceivable to the human eye than low frequency noise. Monte Carlo noise can also be easier removed by using more samples.

When using a local pass, skipping only improves caustics, which are generally less critical.

photons	conventional: 1 map, os=1	skipping=4: 16 maps,each os=1
100k	0.081	0.084
200k	0.049	0.050
1M	0.021	0.022
5M	0.015	0.014

Skipping provides a powerful method for improving 'visual' image quality by removing low frequency noise from the image at almost no cost. This is especially valuable when other methods for decreasing noise - such as local pass or separate calculation of direct light - are not applicable, as for example in BATHROOM or INVISIBLEDATE. Because noise is removed, smaller k's can be used, reducing other artifacts as well. Note that skipping becomes expensive if initialization time is not negligible, for example when using out importance-driven method for generating the photons, or when using excessive splitting of caustic photons.

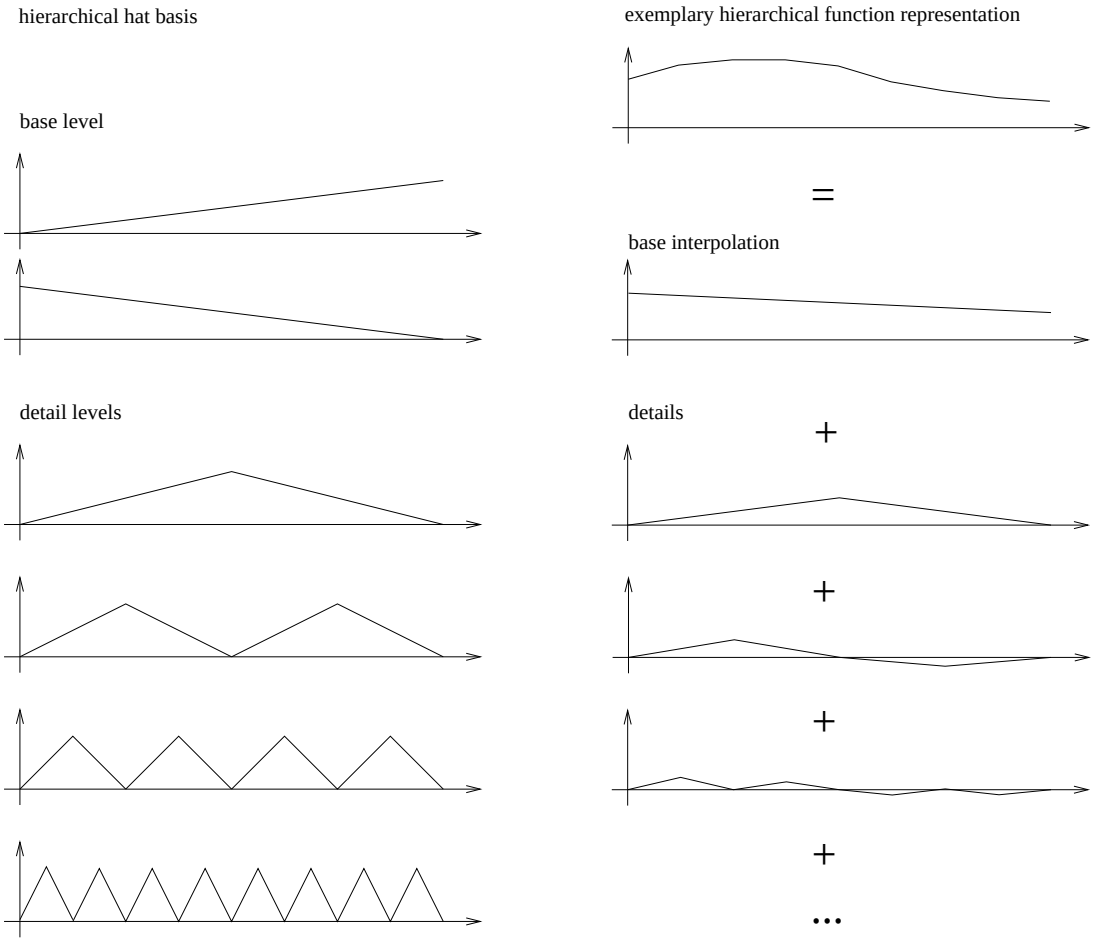


original method of 1 map of 100k photons S=4: using 16 maps of 100k photons in turn

Figure 4.9: Visual effects of skipping: Note the improved visual appearance by changing the low frequency noise of the photon map to less perceivable Monte Carlo noise. Averaging enough of these images makes the noise imperceivable.

4.5 The Hierarchical Photon Map

In [Kel99], a fundamental variance reduction technique for integro-approximation is introduced. The method is based on the method presented in [Hei98], which has been adapted to the setting of photorealistic rendering: Considering the pixel-values $(L_i)_{i=0}^{n_i}$ of a scanline as discrete samples of a continuous function $L(x)$, the method uses a sequence of $l = \lceil log\,(n_i - 1) \rceil$ interpolation operators to generate a successive hierarchical approximation of this function. For the example of the hierarchical hat basis, this process of successively adding finer details is depicted in the following figure:



The approximation can be found by calculating the coefficients of the base functions, which are properly scaled and translated hat functions. If the approximated function is smooth, higher order detail levels have decreasing contribution and decreasing variance, as can also be seen in figure. This can be exploited by using fewer samples for calculating the coefficients of the base functions in the finer levels, therefore reducing the average number of samples per pixel, which is in the sequel denoted n_{avg}. As discussed in the original paper, the interpolation results in striping effects in the image if the target function is not smooth. This can be removed by discarding the interpolation information and resampling the pixels in which discontinuities are detected. This process of resampling a pixel is called *localization*.

The low-frequency noise of the photon map is a smooth function, which implies that the coefficients of the finer levels are almost zero, resulting in good performance of the method. However, if all samples are taken from the same photon map, the method also reproduces the artifacts of the photon map, since most of the artifacts of the photon map - except for Monte Carlo noise in the distribution ray tracer and in the local pass - cannot be removed by using more samples per pixel, as already discussed in section 4.4. As stated before, low-frequency noise can be removed by using multiple maps in the evaluation of a pixel. The hierarchical photon map therefore uses the same hierarchical function approximation as above, but takes each sample in a pixel from a different map: if l is the number of levels, and n_i is the number of samples per level i, the maximum number of samples per pixel is $n_{max} = \sum_{i=1}^{l} n_i$, and therefore n_{max} different maps

are needed. Hierarchical sampling with n_{max} maps of N photons and an average number of n_{avg} samples per pixel closely corresponds to averaging n_{avg} images of 1 map of N photons, as done in section 4.4. Since the same number of samples are taken per scanline, and each sample is taken from a map of N photons, computational cost is exactly the same. However, a further reduction in low-frequency noise can be expected: The idea is that the two corner pixels used in the base interpolation are calculated with n_{max} samples, and therefore contain information from $n_{max} \gg n_{avg}$ maps. Because of the interpolation, part of this information is still used when calculating the other pixels, even if these are calculated with far less samples. This reduction of noise can be seen in the following figure: The left image was rendered by averaging 2 images, each created with a photon map of $10k$ photons, while the right image depicts the image rendered with the hierarchical photon map, which used only $n_{avg} = 1.7$ samples per pixel on the average. Localization was not employed in this experiment. Rendering cost was the same with both methods.

using 2 independent maps

hierarchical photon map with $n_{avg} = 1.7$

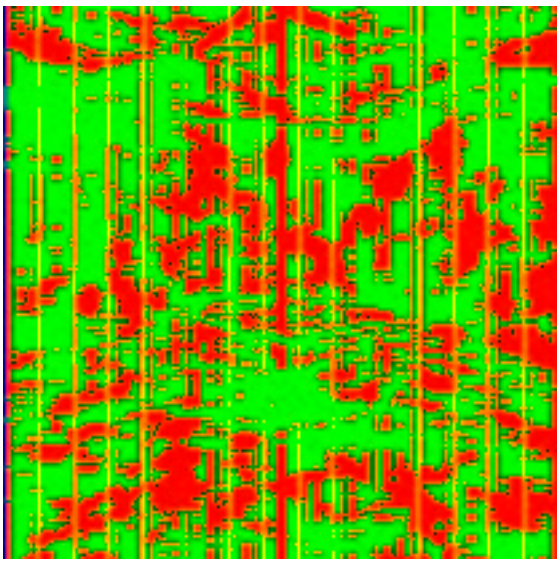
Numerical measurements are presented in the following table (measures were taken on a Dual Celeron/450, with only 10k photon per map, due to lack of memory. The images were rendered in a resolution of 513 by 513 pixels):

Hierarchical n_{avg}	n_{max}	RMS error	Multiple Maps $n = \lceil n_{avg} \rceil$	RMS error
2.80	38	0.012	3	0.017
3.75	43	0.011	4	0.016
8.39	98	0.008	9	0.011

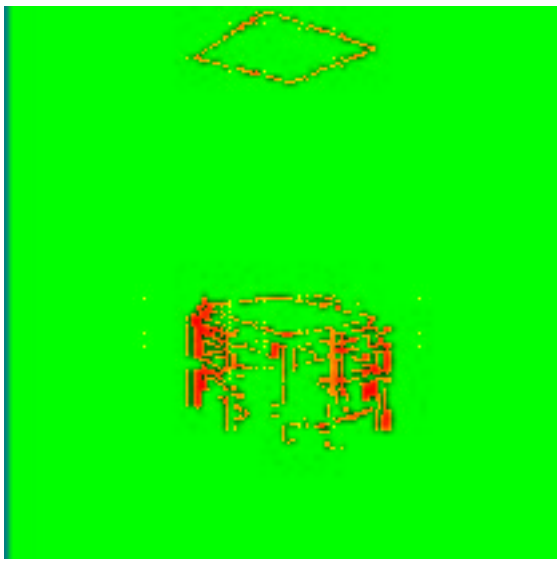
The hierarchical sampling scheme has also been implemented in a 2-dimensional way, sampling the entire image plane instead of a single scanline. Due to lack of memory, the original method was used in our examples. The new method with n_{avg} samples per pixel has the same computationas cost as using 1 equally sized photon map with n_{avg} samples per pixel.

However, further work is required to make the method applicable: Firstly, the method requires to generate many maps: To yield an average number of n_{avg} samples per pixel, $n_{max} \gg n_{avg}$ maps are needed, each requiring its own random walk with at least N ray-intersections. However, the main problem of the hierarchical photon map is its memory consumption. While both the standalone photon map and the skipping method need memory for only one map of N photons in memory at any given time, the hierarchical photon map needs *all* $n_{max} \gg n_{avg} \gg 1$ maps in memory at any time. This is due to the fact that localization requires samples from levels which have already been processed. Another disadvantage is that the method is not as easily parallelizable - it can still be parallelized in a scanline-by-scanline order, but can not be implemented by simply computing and averaging n_{avg} single images. Parallelizing the 2-dimensional version is even harder. The last problem is that localization may become problematic if the single maps are too small: In the finest levels, very few samples are taken, making it possible that the contrast between these few samples and the interpolated value from the previous levels is too high, wrongly classifying this point as a discontinuity and requiring

resampling. If this happens very often, the method is almost reduced to the original pixel-by-pixel-method, which can be demonstrated in the following figure, where the density of the photon map was only $10k$ photons, together with low sample rates and low thresholds for resampling, resulting in localization in all the pixels where the noise of the last level is strong: Red pixels indicate localization, while green pixels indicate that the variance-reduced method could used. Note the similarity of the red pixels with the low-frequency noise as depicted earlier.



bad localization due to high noise in the samples



localization as expected

As a result, it can be stated that the hierarchical photon map provides a tool to reduce low-frequency noise from the photon map. However, it currently is only applicable when the time for creating the maps is negligible, and when large amounts of memory are available. The latter problem is due to our current implementation, and can be relaxed by a suitable scheduling scheme for the order of the hierarchical computations.

4.6 Splitting of Caustic Photons

It was stated before that image quality highly depends on the resolution of the photon map. Therefore, using more photons is always desirable, but limited by the increased cost of larger resolutions. Though diffuse illumination can be improved in several ways - mostly by direct light and local pass - neither of these methods can be applied to caustics. Thus, it is desirable to have a higher density for the caustic map than for the diffuse map, as Jensen sufficiently outlined in his papers ('the solution is [...] to always use enough photons'). In our implementation, this process of increasing the number of caustic photons is called *splitting.*

In [JC95b], Jensen creates caustic photons by using projection maps: For each light source, a projection map is created, which is a (θ, ϕ)-map of the hemisphere on a point on the light source. Each element of the projection map stores a list of all objects visible in the solid angle of this element. This information is then used to directly emit photons towards specular objects. Even when not considering the inherent problems of projection maps, this method is only capable of creating *direct* caustics. Since Jensen's definition of caustics only includes such direct caustics, this is correct for his purposes, but not sufficient for our definition. For example, Jensen's method will not generate any caustics in the INVISIBLEDATE scene, where no specular objects are directly visible from the lightsources (except for the tiles in the room with the lightsource). Therefore, two other methods for splitting caustic photons have been developed: One by creating more photons by 'brute force', and one by using Metropolis sampling.

4.6.1 Splitting by Brute Force

To generate S times as much caustic photons, the brute force method works by simply starting S times as much photons, and then discarding about $S-1$ out of every S non-caustic photons. To get a correct distribution of diffuse photons, the process of discarding the excess non-caustic photons has to be randomized: Each non-diffuse photons is accepted with a fixed probability of $\frac{1}{S}$. The method requires only minor modifications to the original random walk:

```
for (startphoton = 1..N)
   ...
   if (singular)
     CausticMap.AddPhoton(x, ω, φ)
   else
     DiffuseMap.AddPhoton(x, ω, φ)
   ...
```

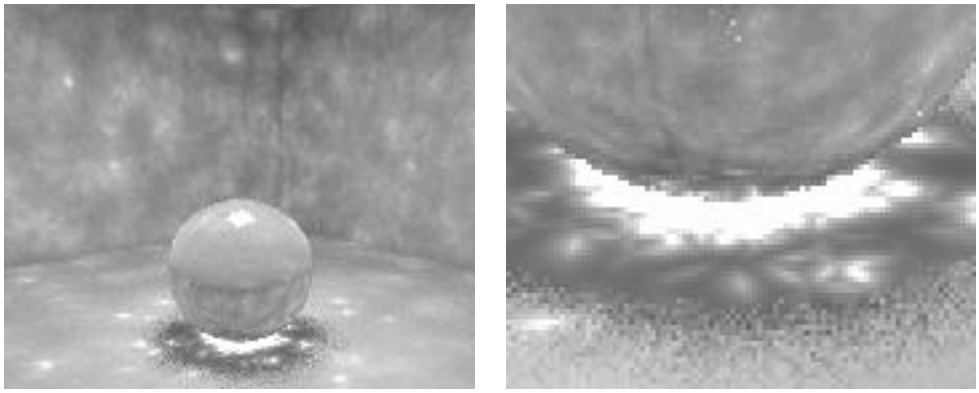
is changed to

```
for (startphoton = 1..(S × N))
   ...
   if (singular)
     CausticMap.AddPhoton(x, ω, φ)
   else
     if (ξ < 1/S) // ξ ∈ [0,1] a random value
       DiffuseMap.AddPhoton(x, ω, S ∗ φ)
     else
       ; // discard photon
```

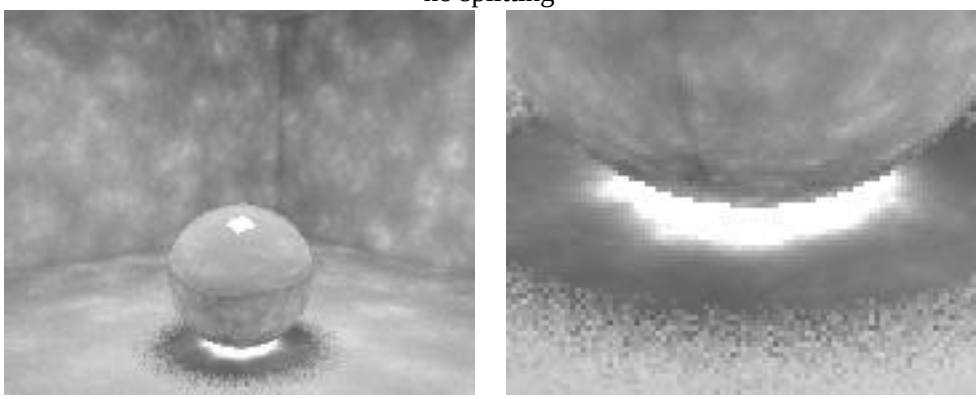
Pure rendering time (rendering time minus initialization time) is slightly increased, because S times as much caustic photons require more expensive queries and shadings for the caustics. If the number of caustic photons is considerably smaller than the number of diffuse photons, this effect is almost negligible, especially when using a local pass for diffuse illumination.

The main increase in cost results from the fact that S times as much photons have to be traced, which yields S times as much ray-scene-intersections in the random walk. This may become a dominating factor when using high splitting rates together with a huge number of photons. As an example, consider the case of a splitting rate of $S = 100$, together with $N = 1$ million photons, which results in 100 million rays beeing shot in the random walk, out of which about 99% are discarded.

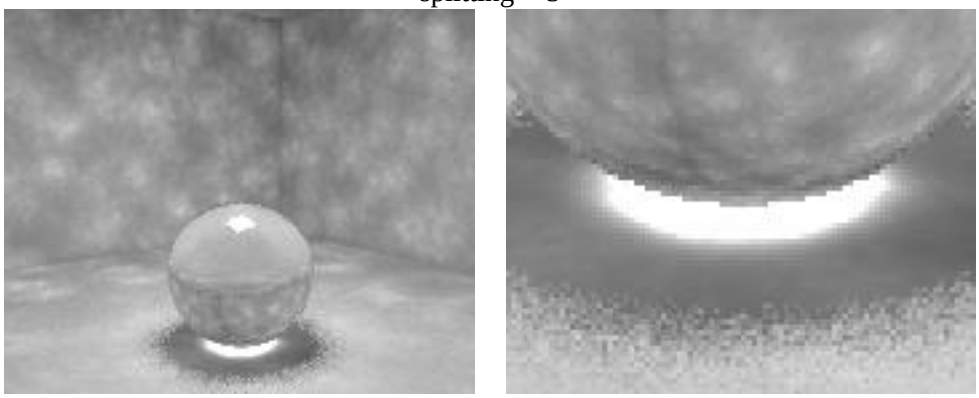
Our experiments have shown that splitting rates of 4-20 are generally sufficient. Additionally, if more photons are used, less splitting is necessary. Since initialization cost is generally small compared to rendering time, splitting improves image quality at acceptable cost. Splitting $N = 100k$ photons with $S = 10$ generates 1 million rays in the random walk, whereas a PAL-sized image with oversampling 4 yields about 2 million primary rays alone, not counted query-times, secondary rays and sample rays for local pass and direct light.



no splitting



splitting = 5



splitting = 20

Figure 4.10: Effects of splitting: Note how the caustic improves without affecting the resolution of the diffuse photons.

4.6.2 Splitting Caustic Photons using Metropolis Sampling

The Metropolis sampling algorithm aims at generating optimal samples. It was first presented for computational physics in 1953 by Metropolis, Rosenbluth, Rosenbluth, Teller and Teller and was adapted to photorealistic rendering by Veach and Guibas in [VG97].

The algorithm presented in [VG97] is a complete rendering algorithm. However, an extremely simplified version can be used for only increasing the number of caustic photons[3]. Even for the understanding of this simplified version, reading the original paper is highly suggested.

In the sequel, several definitions are needed:

- The *path* of a photon is the set of its interactions.
- A *singular chain* is a part of a path which starts at a diffuse interaction, constists of several - at least 1 - singular interactions, and terminates in either a diffuse interaction or in absorption. A path can contain more than one singular chains.
- The *characteristic* of a singular chain is defined by its types of interaction. (a singular interaction can be of different types, i.e. singular reflections or singular transmission). In this section, only perfect reflections/transmissions are called singular, glossy interactions are approximated as perfect singular interactions.

Several observations can be made:

1. Each caustic photon corresponds to exactly one singular chain.
2. Singular chains are started if and only if a diffuse photon has a singular interaction.
3. When only considering perfect reflections/transmissions as singular interactions, a singular chain is completely defined by its starting position and starting direction.

The algorithm requires several modifications of the initialization step. As before, photons are started on the light sources and traced through the scene. Whenever a diffuse photon has a singular interaction, the beginning of a singular chain is registered. This singular chain ends at the next non-singular interaction, where it is stored together with its characteristic. Now, the simplified metropolis sampling algorithm is used for the splitting of this singular chain, by trying to generate 'similar' singular chains, each of which generates a new caustic photon. The starting direction $d^{(i)}$ of the last singular chain is jittered to a new direction $d^{(i+1)}$ using an exponential distribution[4]. Other methods of jittering the starting direction are equally valid, as long as[5]

$$Prob\left(d^{(i)} \rightarrow d^{(i+1)}\right) = Prob\left(d^{(i+1)} \rightarrow d^{(i)}\right).$$

Having generated the new starting direction, it is tested whether a new singular chain can be constructed with the new starting direction, but with the same characteristic as before. If so, the new chain is accepted as the new singular chain with an initial weight 1, otherwise it is discarded, keeping the old chain and increasing its weight by 1. Accepting the chain implies accepting a new starting direction $d^{(i)}$ for jittering. Repeating the above method S times yields a set of $s \leq S$ singular chains, each with a weight and each with the same characteristic. When accepting a new singular chain, the caustic photons of the old chain can be generated, since neither the chain nor its weight will be changed in the future. The implementation of Metropolis splitting requires substantial changes to the random walk.

Metropolis splitting of caustic photons is completely different from other splitting methods and therefore deserves some additional attention. A formal proof of the correctness of the method would exceed the frame of this work. However, some observations can be made on how the method performs in practice. As compared to the 'brute force' method of splitting, Metropolis-splitting requires much less rays to be shot during the initialization phase, since additional rays have to be traced only for singular chains. However, the method also has several drawbacks: Due to the method of Metropolis

[3] In fact, the original metropolis algorithm could be used to generate all photons, rendering importance-driven methods for generating the photons unneccessary.

[4] The chosen distribution affects the efficiency of the method.

[5] This is not a limitation of Metropolis sampling, but simplifies calculations.

sampling, non-fitting samples are discarded. Therefore, some of the work may be lost, especially when parameters are badly chosen. This may lead to very poor acceptance rates, which yield both lots of discarded rays and high noise in the energy of caustic photons, which is due to Monte Carlo noise in the weights.

Another problem of the method is that the starting point of the singular chain is not jittered. Under certain circumstances - caustis are created by relatively small singular objects, and few photons are traced - only few initial singular chains are found for splitting. Since the starting points are not jittered, these starting points may act almost as secondary light sources for caustics, since all photons generated from this mutated chain will have originated from this same spot. As an example, see figure 4.11.

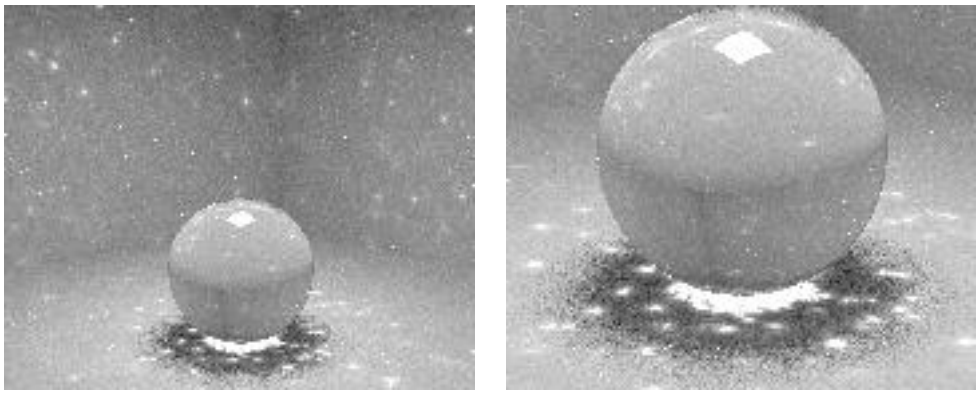


Figure 4.11: Artifacts of Metropolis splitting: The highly illuminated spots are due to singular chains starting far from the light source, acting almost as light sources because of splitting.

As a result, it can be stated that Metropolis splitting is superior to the brute force method only when high splitting values are needed, together with a large number of photons, since its overhead is much smaller. In the general case of small splitting values (about $S = 5...20$), brute force splitting yields better quality at only slightly higher cost.

4.7 Importance driven Photon Map Generation

In [PP98], a method was presented which generates the photon map according to the local importance[6] of the scene. As we have not implemented this method, only a short summary is given here. The method works in 3 passes:

1. Generate the importance map.
2. Estimate the importance of each light source by using a random walk and the importance estimate. This allows for importance sampling the light sources when generating new starting photons.
3. In the random walk, use the importance map to actively direct photons into areas of higher importance

An estimate for the importance can be taken similar to the radiance estimate. Therefore, a small number of importance-particles (called importons) is shot from the eye through the viewing-plane into the scene. These importons are then traced through the scene just like photons in the random walk. The importance then can be estimated by querying the importons.

After this preprocessing step, the importance of each light source is estimated by emitting a small number of sensor photons and tracing these through the scene, adding up the importance at the locations where these sensor photons interact with the scene. This estimate of the importance of a light source then is used for importance sampling by choosing the number of photons emitted from each lightsource according to the source's importance[7]. These photons are traced through the scene in a modified random walk, where the importance-map is used to actively direct photons into areas of high importance in the following way: Whenever a photon hits a surface, its out-scattering direction is not chosen according to the local bsdf f_r, instead it is scattered according to $f_r(\omega, x, \cdot)W(x, \omega)$ where $W(x, \omega)$ denotes the 'visual potential' (see [Kel98],[PP98]), which can be estimated by the importance map. This method for choosing the out-scattering direction of a photon is similar to the one presented in [Jen95].

Since the limited framework of this work did not allow for the complete implementation of the above algorithm, another approach was taken to experiment with the use of importance in generating the photon map: In our approach, a method was implemented which is similar to the brute force splitting algorithm presented in section 4.6.1, by replacing the fixed acceptance probability with an acceptance probability derived from the importance estimate. Firstly, an importance map is generated by emitting a small number of importons into the scene, just as in the above method. The location of these importons in the SCENE10 setting are shown in the middle image in figure 4.12. After creating the importance map, a random walk is undertaken in the same way as in section 2.2. Whenever a photon (ω, ϕ) interacts with a diffuse surface at position x, a value $0 < p(x) \leq 1$ is generated from the importance estimate. In our implementation, $p(x)$ is computed by

$$p(x) = \max\{1,\ p_{min} + (1 - p_{min})\alpha W_{est}\},$$

where W_{est} ist the importance estimate and p_{min} and α are parameters specifying a minimum acceptance probability and a value to control the influence of the estimate. The photon is then stored with probability $p(x)$ as $(x, \omega, \frac{\phi}{p(x)})$. Otherwise, it is discarded. This method results in a small density of highly energetic photons in unimportant areas and in a high density of low-energetic photons in areas with high importance, as is shown in the right image in figure 4.12.

The method is implemented by simply replacing each occurrence of

```
...
map.AddPhoton(x,omega,phi)
...
```

with

```
...
imp = ImportanceEstimate(x);
p = AcceptanceProbability(imp);
if (ξ < p)
   map.AddPhoton(x,omega,phi/p);
...
```

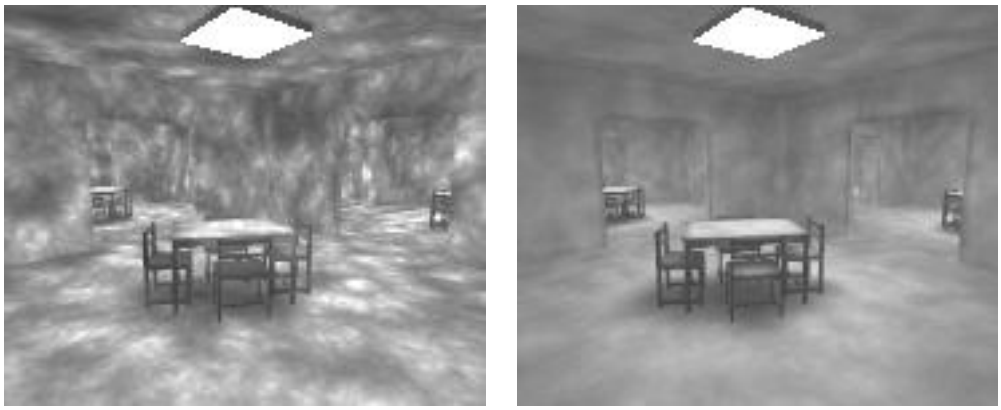
[6] For an exact definition of the term *importance* see the original paper

[7] Note that this information on the importance of a light source might also be used in different applications.

Figure 4.12: Using importance in generating the photon map: The left image shows the position of the photons in SCENE10 (10 by 10 rooms, viewed from the top) as generated with the original random walk without using importance. The center image depicts the locations of the importons, while the importance-driven map generated by this importance map is shown on the right.

As opposed to the algorithm presented in [PP98], our algorithm does not actively direct photons to areas of higher importance, it merely creates a huge number of photons and 'discards' photons in areas of lower importance with a higher probability. The algorithm is therefore very inefficient in generating the photons, however it suffices to demonstrate the value of using importance-driven photon maps. Our method is easily implemented, applicable to any given scene and produces high-quality importance driven photon maps: In INVISIBLEDATE, for example, though beeing inefficient in generating the photons, our method is less problematic than the orignal method.

The following figure shows the impact of the method on SCENE10. Both images have been rendered using 600.000 photons. Due to the high inefficiency of our method in generating the photons, the time for the random walk was about 20 times as high as with the old method. In SCENE10, this cost can be reduced by incorporating the second step of the original paper to emit more photons from important sources. This has not yet been applied in our implementation.



without using importance importance driven

In settings like SCENE10 or INVISIBLEDATE, importance driven generation of photon maps is a neccessity.

4.8 Choosing k Adaptively

When examining the photon map, it can be observed that some of the artifacts are stronger when k is small (e.g. lowering k increases low-frequency noise), while other artifacts appear mainly when k is high (energy-bleeding, bands at edges and borders, loss of shadows). These artifacts also appear in different regions of the scene. Noise can be observed only on smoothly illuminated surfaces, on which blurring is less problematic.

Since all values for k are equally valid for the estimate, it seems promising to choose k proportional to the local smoothness. However, since local smoothness is not known at rendering time, heuristics are needed for choosing k. One possible heuristic is to query a large number of k_{max} photons, and to compare the query-position to the center of gravity of the found photons. In smoothly illuminated areas, the center of gravity can be expected to be close to the query-postition. Here, blurring is not a problem, and all of the found photons can be used to prevent noise. On the other hand, at geometry or shadow borders, most of the photons will be on one side of the query-point, the distance between x_q and the center of gravity therefore beeing big. In this case, only a fraction of the k_{max} photons is used in the estimate, preventing excessive blurring of the shadow border. The thereby created noise is less perceivable in these areas. Therefore, the number k_0 of photons actually used in the estimate would be

$$k_0 = f\left(k_{max}, \frac{|| x_q - \frac{1}{k_{max}} \sum_{p \in Q(x_q, k_{max})} x_p ||}{r}\right).$$

Several cases can be constructed where this simple method fails: Small, smooth shadows would not change the center of gravity, and can therefore not be handled by this heuristic. However as the goal of this work was not the development of heuristics for 'better-looking' images, few time was spent here, and better methods may exist. Note that the method should work well at the borders of sharp, focussed caustics.

This method does only *reduce* artifacts, it does not remove the flaws which are responsible for them. However, it is easily implemented and may lead to images 'looking' more realistic, by incorporating the advantages of both reduced noise with large k's and reduced blurring with small k's. This is demonstrated in the following images:

Scene6, blurred

Scene6, noisy

Scene6, k chosen adaptively

Chapter 5

Results and Discussion

Although the advantages and disadvantages of the different photon map algorithms have already been discussed in the respective sections, this chapter discusses these algorithms from the point of view of the scene to be rendered. In the sequel, we will present several of the scenes used in our experiments, each with a short discussions on the results achieved when rendering the respective scene with photon map algorithms.

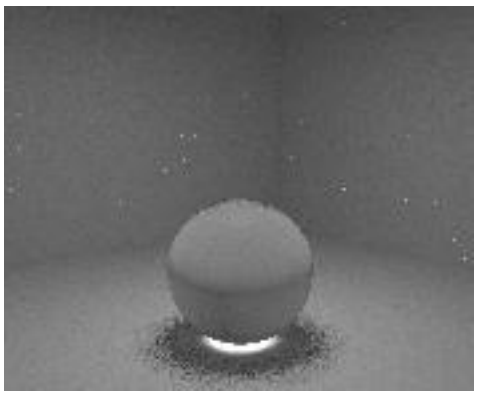
Most of the following scenes are taken from Greg Wards "Materials and Geometry Format"-package ([War95]), which provides a rich library of scenes, ranging from rather simple, ideal scenes like SCENE5 and SCENE6, to more realistic scenes like CONFERENCE, and even such highly complex scenes like MMACK. The package contains a corresponding scene for almost any problem of the photon map, including highly glossy scenes, scenes without direct light (BATHROOM), scenes with a large number of light sources (MMACK) and scenes where importance has to be considered (SCENE10). Since the MGF-Format does not incorporate textures, the artifacts of the photon map are especially visible, which is even an advantage when analyzing an algorithm.

Apart from the MGF scenes, we also used the INVISIBLEDATE scene (see [Kol99]), which - because of its complexity - provides an almost perfect testing ground for global illumination algorithms. INVISIBLEDATE was modelled by Christa Marx, who deserves special thanks for this piece of work.

I also wish to thank DIGITAL EQUIPMENT, a subsidiary of COMPAQ COMPUTERS, for providing us with one of their AlphaStations, which has been an invaluable tool in computationally expensive experiments.

SCENE5

SCENE5 is a simple scene for testing caustics. It only consists of a perfectly diffuse room with an area light source and a sphere, which is 96% transmissive and 4% specular, generating one big caustic on the floor. It provides a perfect testing bed for caustics and the splitting algorithms. Only few problems arose with this scene. One of these problems is that the 4% specularity creates stray caustic photons on wall and ceiling. The noise on the wall results from the local pass hitting the caustic with some of its sample rays.



SCENE6

SCENE6 is the equivalent to SCENE5 for diffuse illumination. It consists only of diffuse surfaces, is mainly illuminated by direct light from one lightsource, and is therefore an almost ideal scene for separate calculation of direct light and the local pass. The local pass is highly efficient in this scene, due to the fact that the illumination is smooth, resulting in all samples returning about the same value. Since the scene contains only one light source, calculating direct light by Monte Carlo integration is highly efficient, and was evaluated by only 4 shadow rays per sample in the following figures (all images have been rendered on a Dual Celeron/450, in VHS resolution and with 4 samples per pixel).

Direct light makes up for about two thirds of the illumination, and calculating it separately yields good quality even without the local pass. However, low frequency noise is then visible in areas of highly indirect illumination, e.g. on the ceiling. When using neither direct light nor local pass, SCENE6 demonstrates the dilemma of having to choose between either increased noise or increased blurring. When using the standalone photon map, huge numbers of photons have to be used to reduce both blurring and noise. The lower left image shows how noise can be reduced by the skipping method: Only 4 images were averaged for this image, which is still noisy. Skipping can not remove the dark bands around the edges of room and table, which are due to the wrong area estimate. The increased cost for the skipping method is due to the 3 additional random walks. The influence would be reduced when rendering higher image resolutions.

Note that by simply covering the lamp with a glass plate, local pass and direct light would be rendered ineffective, yielding only the quality of the image with the standalone photon map.

standalone photon map
88sec

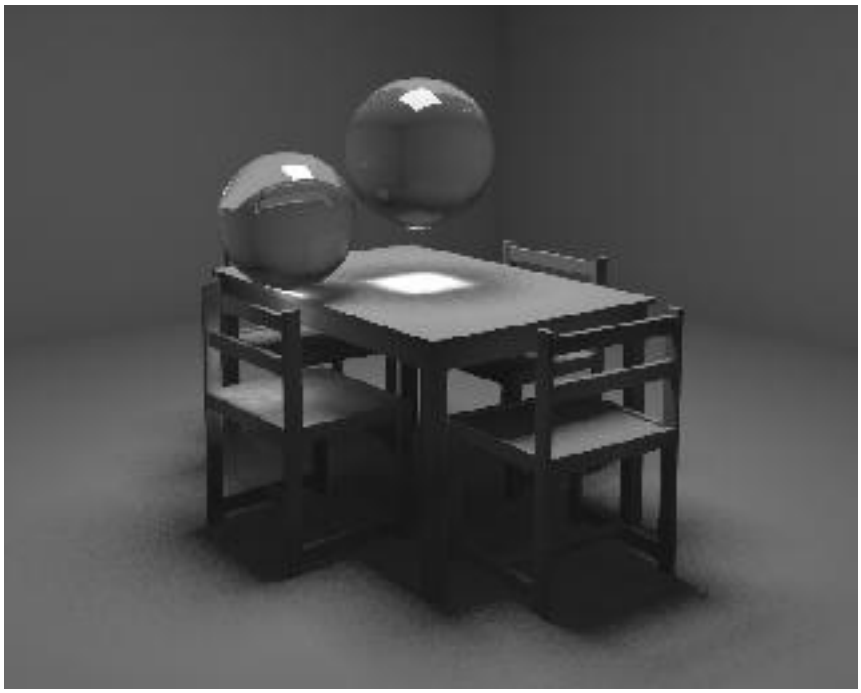
photon map with direct light
96sec

skipping: 4 single images,
9 maps per image
160sec

photon map with direct light
and local pass (4 samples)
359sec

TABLEANDSPHERES

TABLEANDSPHERES is a hybrid of SCENE5 and SCENE6. It was created by enriching SCENE6 with 2 spheres taken from SCENE5, to create 2 caustics on table and chair. The caustic on the chair is noisy, while the borders of the other caustic are blurred. This demonstrates that the dependence on parameters still holds true for caustics, even if direct light and local pass are used.



CONFERENCE

CONFERENCE is one of the more realistic scenes. Its main problems are the exit signs above the door. These tend to create overmodulation, which can only be removed by a local pass. This overmodulation also creates noise in the local pass, when sample rays become overilluminated by hitting these areas. Except for this, the local pass performs well in this setting. Several of the objects are slightly glossy, generating some caustic photons which are distributed almost evenly in the entire room. These 'stray' caustic photons create low-frequency noise which can not be removed by the local pass. Computing direct illumination is somewhat expensive due to the high number of light sources, but yields good quality, except for the ceiling, where indirect illumination dominates. PhotonDirect can not be used here, since almost all sources have the same importance.

CONFSPHERES

CONFSPHERES is a modification of the Conference room, introducing caustics by some spheres which are hovering in the room. Because of splitting, the caustics on the table could be rendered at high quality. However, note the still blurred caustic borders of the caustics on the floor. In this image, caustics photons were discarded if too few photons were found. Note how the noise at the ceiling was removed by this method. This, however, resulted in a notable loss of energy.

SCENE10

SCENE10 consists of an array of 10×10 interconnected rooms, each single room similar to SCENE6. It demonstrates the weaknesses of the forward simulation used in the random walk, because all of the 100 rooms are approximated with an equal number of photons, implying that only 1 percent of the photons is used to approximate the room with the viewer. SCENE10 is therefore an ideal scene for testing importance driven methods to generate the photon map. It also provides a good testing ground for the PhotonDirect method (see section 4.2.3), because of the large number of light sources with varying importance. The image was rendered in 6 minutes on a dual Celeron/450, using an importance-driven photon map of about 150.000 photons, together with a local pass. Direct illumination was computed with the PhotonDirect method.

BATHROOM

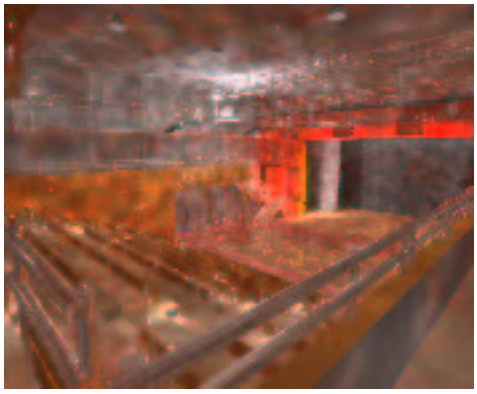
The BATHROOM scene consists of a large number of specular objects. Its main difficulty arises from the light sources, which are modelled to be surrounded by transmissive glass spheres. While this is absolutely realistic, it makes the scene very hard to be rendered with the photon map: After being emitted by a light source, each photon has to pass the surrounding glass sphere, being therefore classified as a caustic photon. Since most of the photons in this scene are therefore caustic photons - except for the photons that become diffuse at the second interaction - a local pass is highly ineffective in this scene. Separate calculation of direct light can not be applied here, since by definition only the glass spheres are directly illumintated. Since almost all photons are cautic photons in this scene, splitting simply increases the number of overall photons. This leaves the skipping method as the only improvement applicable in this setting, beeing able to diminish blurring by smaller k's and to remove low-frequency noise. Skipping can not remove the visble bands at geometry edges. However, these could be diminished by increasing the photon map density (only 60.000 photons were used in this example !).

The image have been rendered with truely glossy reflections, increasing the rendering time due to the higher sampling rates needed for the distribution ray tracer. Rendering was done with the skipping technique, by averaging 8 VHS-resolution images, each rendered with 4 independent maps. Rendering took $2\frac{1}{2}$ hours on a single Celeron/450.

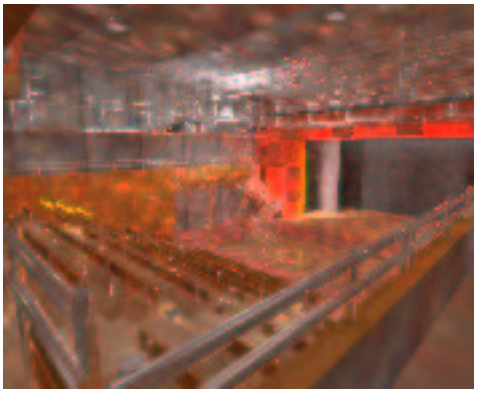
MMACK

MMACK contains a large number of light sources and highly glossy objects. The large number of lightsources makes separate computation of direct light successful, but expensive. PhotonDirect can not be applied here, because a very large number of sources have influence at almost any given point. The large number of highly glossy surfaces results in a lot of noise in both the distribution ray tracer and the local pass. They also rendered the local pass ineffective, because of the large number of caustic photons which cannot be handled by the local pass.

Images have been rendered on an DEC Alpha/333 with 200k initial photons, VHS resolution, 8 samples per pixel and caustic-splitting=4. Direct light calculation was very expensive in this scene because of the large number of light sources. Using a local pass yields huge rendering times, and cannot improve quality due to of the excessive noise. Skipping improves the visual appearance at low cost. Creating the additional photon maps almost tripled rendering time due to the increase in preprocessing time, while pure rendering time was the same as with the standalone method. Note that larger resolutions and less splitting reduces this effect.

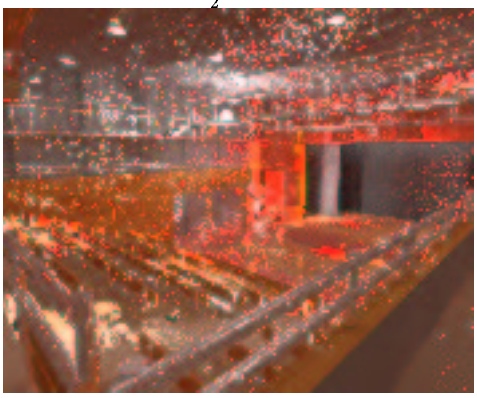


standalone photon map
36 minutes



photon map and direct light
$2\frac{1}{2}$hours

Skipping, 8 runs,
9 maps per run
$1\frac{3}{4}$hours



photon map mit direct light
und local pass (8 samples)
20 hours

INVISIBLEDATE

INVISIBLEDATE is the most problematic of our scenes, since almost all of the previously discussed problems apply to this scene:

1. Since absolutely no direct illumination reaches the room, separate calculation of direct light is futile. This also applies to the PhotonDirect method.
2. The high indirectness of the illumination results in only a small fraction of the photons reaching the room, which requires a large number of photons to create an adequate photon density in the room. Using the importance driven methods for generating the photons is highly expensive in this scene, which is due to several reasons: First, since the probability of passing through the door is very small, the majority of the photons is discarded, so large numbers of photons have to be traced to yield a reasonably high photon density in the room. Additionally, 'energy'-bleeding also applies to importance, resulting in high importance estimates on the wrong side of the wall. To prevent storing the majority of the photons on the wrong side of the wall (where they are more likely to hit the wall), occlusion testing has to be applied even in the importance estimate, making the cost for generating the photons even worse.
3. Since the wall and the door are highly illuminated from the backside, energy-bleeding occurs. Note that the high photon density on the door also produces overmodulation.
4. Even though the shadows are sharp and small, they are indirect shadows and can not be imroved by direct light. Since the radiance estimate completely removes such thin shadows, they can only be generated by a local pass or occlusion-testing.
5. The entire floor - together with numerous other objects - is highly glossy, which creates a large number of 'stray' caustic photons. This generates low-frequency noise that cannot be removed by a local pass.
6. The high amount of glossy objects also requires a lot of shader-calls due to the distribution ray tracer, since each ray hitting a glossy surface spawns several secondary rays, each generating another call to the photon map shader.
7. Compared to the rest of the room, the area around the door is highly illuminated. Since this area is relatively small, only few samples of the local pass will hit this high-energetic area, creating noise in the local pass.
8. In order to reduce the noise in the local pass, a large number of samples have to be used, furtherly increasing the number of rays and photon map-estimates.
9. The sphere should produce 2 caustics: One from reflection, the other from refraction. Both are relatively weak and extended over a large area. Because of the low probability of a photon hitting the sphere, these caustics are approximated by only few photons, especially when compared to the large amount of 'stray' caustic photons due to the glossy surfaces.
10. Caustic-splitting can not be applied, because of the large number of stray caustic photons. Splitting would also furtherly increase the cost for the random walk.

The following image is a master image rendered with Thomas Kollig's implementation of the METROPOLIS algorithm, see [Kol99]. Neither of our experiments yielded comparable quality, even after huge rendering times.

Several images rendered with the different photon map algorithms can be found in the previous sections. Our best images is presented below to allow a comparison of the images. The image was rendered with skipping and occlusion-testing, which already comes very close to the desired quality in producing a smooth image with all glossy reflections and even indirect shadows. However, a closer look reveals that several illumination details are still missing. The caustic of the glass sphere on the wall, for example, could not be sufficiently reproduced with photon map algorithms.

Chapter 6

Conclusions

The photon map undoubtedly has many advantages: In suitable settings, it creates high-quality images containing almost all the features expected from global illumination, including realistic shadows, specular effects, caustics and even volumetric effects. The photon map is capable of handling arbitrary bsdfs and almost any kind of scene representation. As opposed to many other illumination techniques, objects don't have to be parametrizable and tesselating the scene is not required, which makes the photon map a very powerful tool in rendering scenes with highly complex geometry.

However, the radiance estimate from the photon map is only a rough approximation and contains several artifacts, which requires the photon map to rely on supplementary algorithms like the separate calculation of direct light (section 4.2) and the local pass (section 4.3) to reduce these artifacts. This limits the photon map to scenes where these extensions can be used efficiently, and makes it vulnerable to settings where this is not the case: Direct light is only applicable in directly illuminated scenes, and even then is expensive to calculate if the scene contains many light sources. Under certain conditions, the high rendering cost for calculating direct illumination in scenes with many ligh sources can be relaxed by the PhotonDirect method as presented in section 4.2.3, which uses photon map information for efficiently sampling the light sources. Using a local pass is expensive, and requires the scene to be smoothly illuminated, or otherwise creates Monte Carlo noise, especially if the radiance estimate in the scene contains artifacts like energy-bleeding or overmodulation. It is also limited to diffuse illumination and therefore cannot be used to improve caustics. If the local pass is not applicable, low frequency noise can still be removed by using multiple photon maps (section 4.4), which can also be implemented efficiently in a hierarchical way (section 4.5). Caustics have to be rendered by directly visualizing the estimate, and can therefore only be improved by using more photons (section 4.6). This, however, is limited to sharp, focussed caustics. If the scene contains many singular objects, caustic photons are distributed almost evenly over the entire scene, and splitting is not practicable. Another problem of the photon map is that it depends on the efficiency of the distribution ray tracer into which it is built: Scenes with many singular and glossy objects require high rendering times independently from the photon map. Finally, the random walk used to generate the discrete density is problematic in scenes with low importance or highly indirect illumination, since most of the photons are then used to approximate areas of low importance. This makes it necessary to use importance driven methods for generating the photons as presented in section 4.7.

As holds true for almost any algorithm, examples can be generated which result in very poor performance. The problem is that some of these 'worst-case' settings for the photon map are not purely theoretical, but may easily arise in realistic scenes (see discussion in chapter 5). On the other hand, some of these scenes would also be fatal - or even impossible - for many other rendering techniques.

As a final result, it can be stated that the photon map is a very powerful tool for photorealistic image synthesis, but much work remains to be done to make it applicable in realistic settings.

Bibliography

[AK90] James R. Arvo and David B. Kirk.
Particle Transport and Image Synthesis.
Computer Graphics (ACM SIGGRAPH '90 Proceedings), 24(4):63–66, August 1990.

[Ben75] Jon Louis Bentley.
Multidimensional binary search trees used for associative searching.
Communications of the ACM, 9(18):509–517, 1975.

[BF79] J. Bentley and J. Friedman.
Data structures for range searching.
ACM Computing Surveys, 11(4):397–409, 1979.

[Coo86] Robert L. Cook.
Practical Aspects of Distributed Ray Tracing.
In *ACM SIGGRAPH '86 Developments in Ray Tracing seminar notes*. August 1986.

[CPC84] R. Cook, T. Porter, and L. Carpenter.
Distributed Ray Tracing.
In *Computer Graphics (SIGGRAPH 84 Conference Proceedings)*, pages 137–145, 1984.

[CRMT91] S. Chen, H. Rushmeier, G. Miller, and D. Turner.
A progressive Multi-Pass Method for Global Illumination.
In *Computer Graphics (SIGGRAPH 91 Conference Proceedings)*, pages 165 – 174, 1991.

[CW93] M. Cohen and J. Wallace.
Radiosity and Realistic Image Synthesis.
Academic Press Professional, Cambridge, 1993.

[Hei98] S. Heinrich.
A Multilevel Version of the Method of Dependent Tests.
In *Proc. of the 3rd St. Petersburg Workshop on Simulation*, pages 31–35. St. Petersburg University Press, 1998.

[JC95a] H. Jensen and N. Christensen.
Efficiently Rendering Shadows Using the Photon Map.
In H. Santo, editor, *Edugraphics + Compugraphics Proceedings*, pages 285–291. GRASP- Graphic Science Promotions & Publications, 1995.

[JC95b] H. Jensen and N. Christensen.

Photon Maps in Bidirectional Monte Carlo Ray Tracing of complex Objects.
Computer and Graphics, 19(2):215–224, 1995.

[JC98] H. Jensen and P. Christensen.
Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps.
In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 311–320. ACM SIGGRAPH, Addison Wesley, July 1998.

[Jen95] H. Jensen.
Importance Driven Path Tracing Using the Photon Map.
In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proc. 6th Eurographics Workshop on Rendering)*, pages 326–335. Springer, 1995.

[Jen96a] H. Jensen.
Global Illumination Using Photon Maps.
In *Rendering Techniques '96 (Proc. 7th Eurographics Workshop on Rendering)*, pages 21–30. Springer, 1996.

[Jen96b] H. Jensen.
Rendering Caustics on Non-Lambertian Surfaces.
In *Proc. Graphics Interface*, pages 116–121. Morgan Kaufmann, 1996.

[Kaj86] J. Kajiya.
The Rendering Equation.
In *Computer Graphics (SIGGRAPH 86 Conference Proceedings)*, pages 143–150, 1986.

[Kel96] A. Keller.
Quasi-Monte Carlo Methods in Computer Graphics: The Global Illumination Problem.
Lectures in App. Math., 32:455–469, 1996.

[Kel97] A. Keller.
Instant Radiosity.
In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 49–56, 1997.

[Kel98] A. Keller.
Quasi-Monte Carlo Methods for Photorealistic Image Synthesis.
Ph.D. thesis, Shaker Verlag Aachen, 1998.

[Kel99] Alexander Keller.
Hierarchical monte carlo image synthesis.
Technical report, Universität Kaiserslautern, 1999.

[KMS94] A. Kersch, W. Morokoff, and A. Schuster.
Radiative Heat Transfer with Quasi-Monte Carlo Methods.
Transport Theory and Statistical Physics, 7(23):1001–1021, 1994.

[Kol99] Thomas Kollig.
Lichtsimulation mit dem metropolis algorithmus, 1999.

[Mad] Klaus Madlener.

Skript zur Vorlesung Effiziente Algorithmen.
AG Grundlagen der Informatik, Professor Madlener, Universität Kaiserslautern.

[PP98] I. Peter and G. Pietrek.
Importance driven construction of photon maps.
Technical Report, Nr.677, Uni Dortmund, 1998.

[Sed98] Robert Sedgewick.
Algorithms in C++ (third edition).
Addison Wesley, 1998.

[SW92] P. Shirley and C. Wang.
Distribution Ray Tracing: Theory and Practice.
In *3rd Eurographics Workshop on Rendering*, pages 33 – 43, Bristol, England, May 1992.

[SWZ96] P. Shirley, C. Wang, and K. Zimmerman.
Monte Carlo Techniques for Direct Lighting Calculations.
ACM Trans. Graphics, 15(1):1–36, 1996.

[VBS99] M. Vanco, G. Brunnett, and T. Schreiber.
A hashing strategy for efficient k-nearest neighbors computation sets.
CGI '99, June 1999.

[VG97] E. Veach and L. Guibas.
Metropolis light transport.
In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 65–76. ACM SIGGRAPH, Addison Wesley, August 1997.

[War91a] G. Ward.
Adaptive Shadow Testing for Ray Tracing.
In *2nd Eurographics Workshop on Rendering*, Barcelona, Spain, 1991.

[War91b] G. Ward.
Real Pixels.
In J. Arvo, editor, *Graphics Gems II*, pages 80–83. Academic Press, 1991.

[War92] G. Ward.
Measuring and Modeling Anisotropic Reflection.
In *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, pages 265 – 272, 1992.

[War95] G. Ward.
Realistic input for realistic images, ch. the materials and geometry format.
ACM SIGGRAPH Course Notes, 1995.

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