FMM-based Illumination Maps For Point Models

Second Progress Report

Submitted in partial fulfillment of the requirements
for the degree of

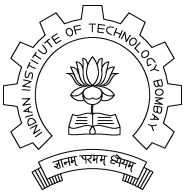
Ph.D.

by

Rhushabh Goradia
Roll No: 04405002

under the guidance of

Prof. Sharat Chandran



Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai
August 22, 2006

[0]Time : 10:45-12:00pm
[1]Date : 23.08.2006
[2]Venue : Conference Room, EE Department
[3]RPC : Prof. Sharat Chandran, Prof. Merchant, Prof. Shevare

Acknowledgments

I would like to thank Prof. Sharat Chandran for devoting his time and efforts to provide me with vital directions to investigate and study the problem.

I would also like to thank Prof. Amitava Datta (University of Western Australia) and all the members of ViGIL for their valuable support.

Rhushabh Goradia

Contents

List of Figures

Abstract

Point-based methods have gained significant interest due to their simplicity. The lack of connectivity touted as a plus, however, creates difficulties in generating global illumination effects. We are interested in looking at inter-reflections in complex scenes consisting of several models, the data for which are available as hard to segment aggregated point-based models.

In this report we use the Fast Multipole Method (FMM) which has a natural point based basis, and the light transport kernel for inter-reflection to compute a description – illumination maps – of the diffuse illumination. These illumination maps may be subsequently rendered using methods in the literature such as the one in [38]. We present a hierarchical visibility determination module suitable for point based models.

Chapter 1

Introduction

1.1 Introduction

The pixel indeed has assumed mystical proportions in a world where computer assisted graphical techniques have made it nearly impossible to distinguish between the real and the synthetic. Digital imagery now underlies almost every form of computer based entertainment besides serving as an indispensable tool for fields as diverse as scientific visualization, architectural design, and as one of its initial killer applications, combat training. The most striking effects of the progress in computer graphics can be found in the filmed and interactive entertainment industries (Figure 1.1).

Figure 1.1. Impact of photorealistic computer graphics on filmed and interactive entertainment. Left: A still from the animated motion picture 'Final Fantasy : The Spirits Within'. Right: A screenshot from the award-winning first person shooter game 'Doom III'

The process of visualizing a virtual three dimensional world is usually broken down into three stages:

- **Modeling.** A geometrical specification of the scene to be visualized must be provided. The surfaces in the scene are usually approximated by sets of simple surface primitives such as triangles, cones, spheres, cylinders, NURBS surfaces, **points** etc.
- **Lighting.** This stage involves ascribing *light scattering properties* to the surfaces/surface-samples composing the scene (e.g. the surface may be purely reflective like a mirror or glossy like steel). Finally, a description of the *light sources* of the scene must be provided - those surfaces that spontaneously emit light.

- **Rendering.** The crux of the 3D modeling pipeline, the rendering stage accepts the three dimensional scene specification from above and renders a two dimensional image of the same as seen through a camera. The algorithm that handles the simulation of the light transport process on the available data is called the *rendering algorithm*. The rendering algorithm depends on the type of primitive to be rendered. For rendering points various rendering algorithms like QSplat, Surfel Renderer etc are available.

Photorealistic computer graphics attempts to match as closely as possible the rendering of a virtual scene with an actual photograph of the scene had it existed in the real world. Of the several techniques that are used to achieve this goal,*physically-based* approaches (i.e. those that attempt to simulate the actual physical process of illumination) provide the most striking results. The emphasis of this report is on a very specific form of the problem known as *global illumination* which happens to be a photorealistic, physically-based approach central to computer graphics. This report is about capturing interreflection effects of a set of objects when the input is available as point samples. We use the technique of the Fast Multipole Method which also starts with points as primitives. Figure 1.2 shows some of the application domains where this method can be applied.

Figure 1.2. Grottoes, such as the ones from China and India form a treasure for mankind. If data from the ceiling and the statues are available as point samples, can we capture the interreflections?

1.1.1 Point Based Modelling and Rendering

Figure 1.3. Example of Point Models

In recent years, point-based methods have gained significant interest. In particular their simplicity and total independence of topology and connectivity make them an immensely powerful and easy-to-use tool for both modelling and rendering. For example, points are a natural representation for most data acquired via measuring devices such as range scanners [25], and directly rendering them without the need for cleanup and tessellation makes for a huge advantage.
Second, the independence of connectivity and topology allow for applying all kinds of operations to the points without having to worry about preserving topology or connectivity

[30, 27, 31]. In particular, filtering operations are much simpler to apply to point sets than to triangular models. This allows for efficiently reducing aliasing through multi-resolution techniques [31, 32, 39], which is particularly useful for the currently observable trend towards more and more complex models: As soon as triangles get smaller than individual pixels, the rationale behind using triangles vanishes, and points seem to be the more useful primitives. Figure 1.3 shows some example point based models.

1.1.2 Global Illumination

Global illumination algorithms are those which, when determining the light falling on a surface, take into account not only the light which has taken a path directly from a light source (direct illumination), but also light which has undergone reflection from other surfaces in the world (indirect illumination).

Figure 1.4 gives you some examples images showing the effects of *Global illumination*. It is a simulation of the physical process of light transport. It has been traditionally solved for models with some *surface representation*. The lack of any sort of connectivity information in point-based modeling (PBM) systems now *hurt* photo-realistic rendering. This becomes especially true when it is not possible to correctly segment points obtained from an aggregation of objects (see Figure 1.2) to stitch together a surface.
There have been efforts trying to solve this problem [38], [3, 34], [1, 27] , [32]. Our view is that these methods would work *even better* if fast pre-computation of diffuse illumination could be performed. Fast Multipole Method (FMM) provides an answer.

1.1.3 Fast computation with Fast Multipole Method

Computational science and engineering is replete with problems which require the evaluation of pairwise interactions in a large collection of particles. Direct evaluation of such interactions results in $O(N^2)$ complexity which places practical limits on the size of problems which can be considered. Techniques that attempt to overcome this limitation are labeled N-body methods. The N-body method is at the core of many computational problems, but simulations of celestial mechanics and coulombic interactions have motivated much of the research into these. Numerous efforts have aimed at reducing the computational complexity of the N-body method, particle-in-cell, particle-particle/particle-mesh being notable among these. The first numerically-defensible algorithm [10] that succeeded in reducing the N-body complexity to $O(N)$ was the Greengard-Rokhlin Fast Multipole Method (FMM) [17].
The algorithm derives its name from its original application. Initially developed for the fast evaluation of potential fields generated by a large number of sources (e.g. the gravitational and electrostatic potential fields governed by the Laplace equation), this method has been generalized for application to systems described by the Helmholtz and Maxwell equations, and to name a few, currently finds acceptance in chemistry[5], fluid dynamics[16], image processing[12], and fast summation of radial-basis functions [6]. For its wide applicability and impact on scientific computing, the FMM has been listed as one of the top ten numerical algorithms invented in the 20th century[10].
The FMM, in a broad sense, enables the product of restricted dense matrices with a vector to be evaluated in $O(N)$ or $O(N \log N)$ operations, when direct multiplication requires $O(N^2)$ operations.

1.2 Overview of the Report

Having got a brief overview of the keyterms, let us review the approach in detail in the subsequent chapters. The rest of the report is organized as follows. We present our idea in Chapter 2 along with our contributions. We then discuss our ideas and directions of future work in Chapter 3. We conclude the report with our concluding remarks in Chapter 4.

We have detailed our literature survey in the Appendix. In Appendix A, we present an

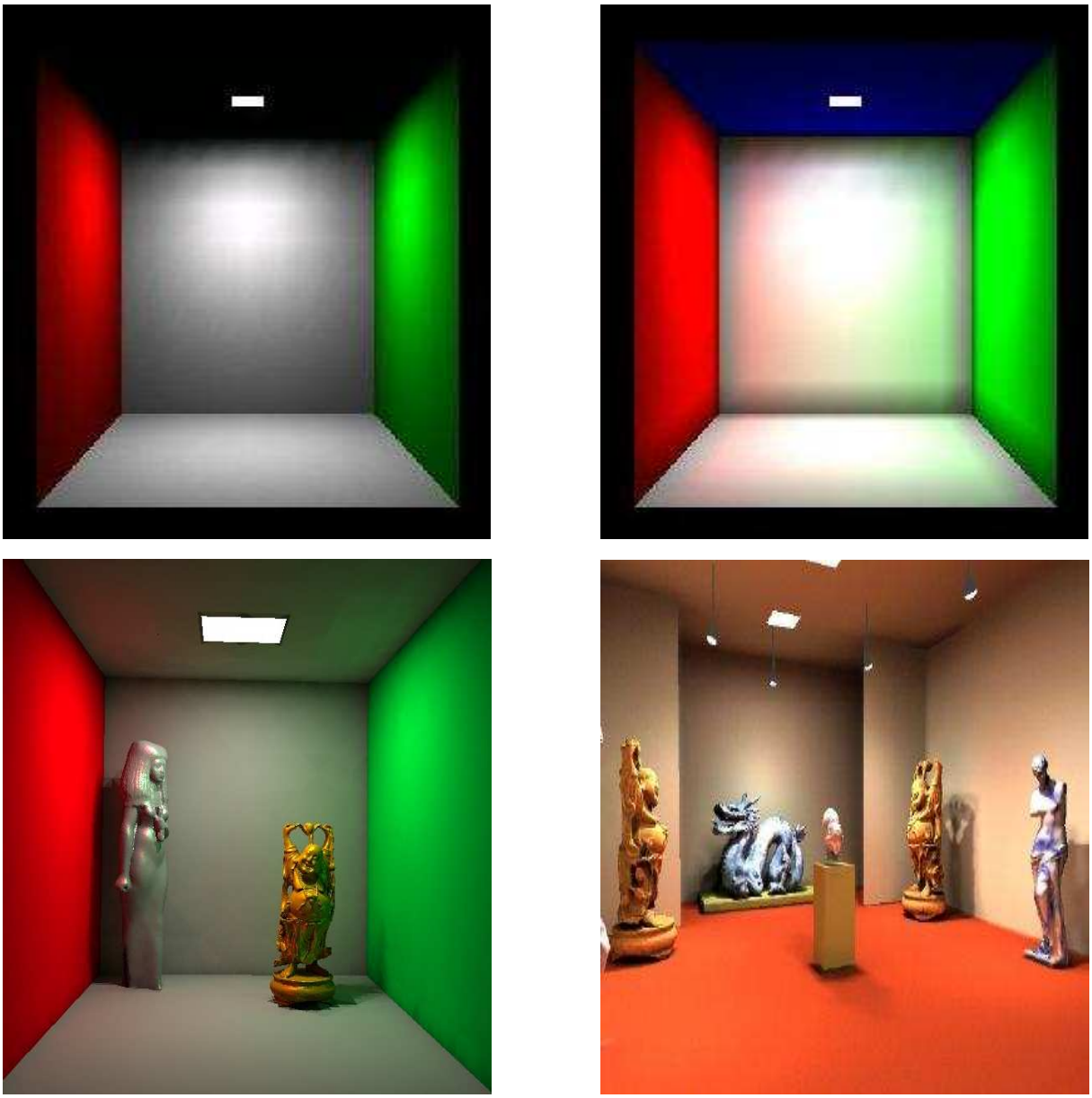


Figure 1.4. Global Illumination. Top Left[23]: The 'Cornell Box' scene. This image shows local illumination. All surfaces are illuminated solely by the square light source on the ceiling. The ceiling itself does not receive any illumination. Top Right [23]: The Cornell Box scene under a full global illumination solution. Notice that the ceiling is now lit and the white walls have color bleeding on to them. Bottom Left: A global illumination solution with reflections and shadows. Bottom Right: (from [9]) "a major goal of realistic image synthesis is to create an image that is perceptually indistinguishable from an actual scene".

overview of techniques for simulating light transport with an emphasis on the radiosity method. We also mention other techniques and previous work in this field. In Appendix B, we present the theoretical foundations of the Fast Multipole Algorithm. We present the requirements subject to which the FMM can be applied to a particular domain and discuss the actual algorithm and its complexity. Finally, in Appendix C, we present the mathematical apparatus required to apply the FMM to radiosity. Five theorems with respect to the core radiosity equation are proved in this context.

Chapter 2

Work done

In this chapter, I will describe the details of the paper [13] we wrote.

2.1 Introduction

This project is about capturing interreflection effects of a set of objects when the input is available as point samples. We use the technique of the Fast Multipole Method which also starts with points as primitives.

Global illumination – the simulation of the physical process of light transport – is a memory intensive and compute intensive operation. This problem has been considered for several years with interesting methods like statistical photon tracing, directional radiance maps, and wavelets based hierarchical radiosity. Traditionally all these methods *assume a surface* representation for the propagation of indirect lighting. Surfaces are either explicitly given as triangles, or implicitly computable. The problem becomes more interesting when the input is in terms of **Point Models**, which do not have any connectivity information.

Points as primitives have come to increasingly challenge polygons for complex models; as soon as triangles get smaller than individual pixels, the raison d'etre of traditional rendering can be questioned. Simultaneously, modern 3D digital photography and 3D scanning systems [25] acquire both geometry and appearance of complex, real-world objects in terms of (humongous) points. More important, however, is the considerable freedom points enjoy. The independence of connectivity and topology enable filtering operations, for instance, without having to worry about preserving topology or connectivity [30, 27, 31].

The lack of any sort of connectivity information in point-based modeling (PBM) systems now *hurt* photo-realistic rendering. This becomes especially true when it is not possible to correctly segment points obtained from an aggregation of objects (see Figure 1.2) to stitch together a surface. Recent work [38] suggests one way to handle this problem — ray tracing. However, as both rays and points are singular primitives, this requires one to trace thick rays [3, 34]. Alternatively, points are seen as covering a finite area by expanding them to ellipses [32], or filtering them with an implicit function [1, 27].

Our view is that these methods would work *even better* if fast pre-computation of diffuse illumination could be performed, much the way photon tracing is done for triangulated models before rendering.

There are numerous problems which require the evaluation of pairwise interactions in a large collection of particles. Direct evaluation of such interactions results in $O(N^2)$ complexity which places practical limits on the size of problems which can be considered. The first numerically-defensible algorithm [10] that succeeded in reducing the N-body complexity to $O(N)$ was the Greengard-Rokhlin Fast Multipole Method (FMM) [17].

Since Global Illumination for point models also requires the evaluation of pairwise interactions in a large collection of particles, FMM naturally suits as a solution for a fast pre-computation algorithm required for diffuse illumination.

*Visibility calculation between point pairs is **essential** as a point recieves energy from other point only if it is **visible** to that point.* But its easier said than done. Its complicated in our case as our input data set is a point based model with *no connectivity* information. Thus, we do not have knowledge of any intervening surfaces occluding a pair of points. Theoretically, it is therefore impossible to determine exact visibility between a pair of points. We, thus, restrict ourselves to **approximate visibility** with a value between 0 & 1.

2.1.1 Statement of the Problem

After getting a brief overview of the topics, let us now define the problem we pose in this report.

Problem Statement: To compute **illumination maps** for **inter-reflections** in complex scenes represented as **point-models** using **Fast Multipole Method**. The system must handle **occlusions** present in the scene as well.

There are three major things to look out for:

- How FMM solves the radiosity equation to provide us with a **fast way** to get illumination maps
- How we compute point-point visibility
- How we incorporate the visibility algorithm in the FMM way to solve radiosity

2.1.2 Contributions

Can the point-based framework of the FMM (albeit without visibility) be coupled with the input point models to store the diffuse illumination? This report answers this question in the affirmative. We store the precomputation in a data structure called *Illumination Maps* which are conceptually like photon maps except that we do not employ statistical photon tracing. The challenges we solve in the process are

- Earlier [24] presented the mathematical apparatus required to apply the *linear-time adaptive* FMM algorithm to diffuse objects given as triangles. Five mathematical results with respect to the core interreflection kernel under full visibility are now available. We extend this to blend the point based nature of FMM with input available as PBMs instead of triangles. For storing illumination maps, this is sufficient. For more complete rendering (purely based on the FMM technique) we require the BRDF to be available as a low rank matrix. This coupled with a directional discretization of radiance [37] should be employed for pure FMM-based rendering of non-diffuse objects.
- The visibility function is highly discontinuous and, like the BRDF, does not easily lend itself to an analytical FMM formulation. Thus the nature of this computation is $\Omega(n^2)$ for n primitives, which depends on the geometry of the scene. We present a new visibility algorithm (Section 2.2.4) for PBMs. The key features are twofold. First, we have a basic point-to-point approximate visibility function that might be useful in its own right. Second, we have a hierarchical version of aggregated point clouds.

2.2 Our Approach

We will have an insight on each of the above contributions in subsequent sections. We start with the basic background of FMM followed by our mathematical results for factorizing the interreflection kernel. Subsection 2.2.2.3 provides the crucial extension needed when points are given as input. Our visibility requires us to provide a stripped-down FMM algorithm which we give in subsection 2.2.3. We follow this with our algorithm for ignoring occluded surfaces

which consists of two parts. First, our basic "primitive" visibility algorithm for point to point visibility is given in subsection 2.2.4.1. Next, we extend this primitive in the FMM context to build a hierarchical visibility algorithm.

2.2.1 FMM for Global Illumination

The Fast Multipole Method [17] is concerned with evaluating the effect of a "set of sources" $\mathbb{Y}$, on a set of "evaluation points" $\mathbb{X}$. More formally, given

$$\mathbb{X} = \{x_1, x_2, \ldots, x_M\}, \quad x_i \in \mathbb{R}^3, \quad i = 1, \ldots, M, \tag{2.1}$$

$$\mathbb{Y} = \{y_1, y_2, \ldots, y_N\}, \quad y_j \in \mathbb{R}^3, \quad j = 1, \ldots, N \tag{2.2}$$

we wish to evaluate the sum

$$f(x_i) = \sum_{j=1}^{N} \phi(x_i, y_j), \quad i = 1, \ldots, M \tag{2.3}$$

The function ϕ which describes the interaction between two particles is called the "kernel" of the system. The function f essentially sums up the contribution from each of the sources y_j. Assuming that the evaluation of the kernel ϕ can be done in constant time, evaluation of f at each of the N evaluation points requires N operations. The total complexity of this operation will therefore be $O(NM)$. The FMM attempts to reduce this seemingly irreducible complexity to $O(N \log N + M)$ or even $O(N + M)$. The two main insights that make this possible are:

- Factorization of the kernel
- The observation that many application domains do not require that the function f be calculated at very high accuracy.

2.2.2 Interreflection in the FMM context

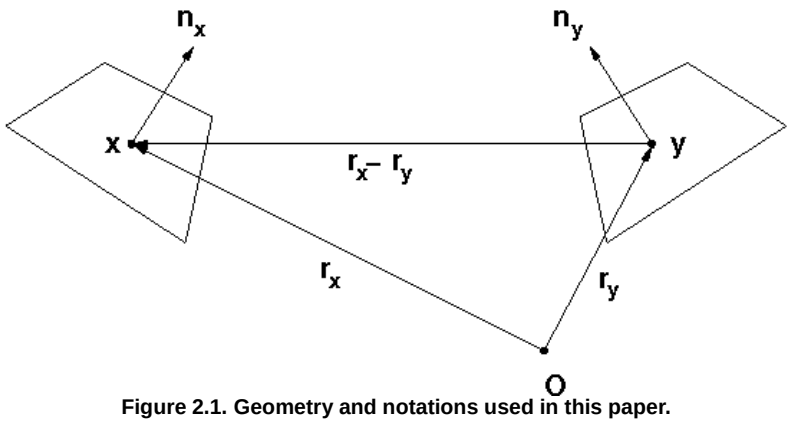


Figure 2.1. Geometry and notations used in this paper.

Figure 2.1 shows how a point x receives irradiance from a small area around y. The nature of this interaction is quadratic for all points as in Equation 2.3. Further, the kernel of the geometric interaction (assuming full visibility) can be written as:

$$K(x) = \int_{A_y} \frac{[\vec{ny}.(\vec{rx} - \vec{ry})][\vec{nx}.(\vec{ry} - \vec{rx})]}{\pi |\vec{ry} - \vec{rx}|^4} dA_y \tag{2.4}$$

Notice that the interaction written in this form is coupled in nature. The theory of the FMM, in general, requires factorization and translation theorems for the type of kernel under consideration. Simply stating, these results are based on the position and orientation of the source and receivers. These results are given in brief in Section 2.2.2.1 and the proofs appear in [24, 23]. The nature of light transport is even more complicated than this, but Equation 2.4 is sufficient to capture the diffuse illumination maps.

2.2.2.1 Multipole Expansion

As per [23], if we denote the spherical coordinates of $r\vec{x}$ by (r_x, θ_x, ϕ_x), then it makes use of [20] to write (for $r_y < r_x$),

$$\frac{1}{|r\vec{y} - r\vec{x}|^4} = \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} \pi_n^j \left\{ \frac{1}{r_x^{n+4}} Y_{n-2j}^m(\theta_x, \phi_x) \right\} \left\{ r_y^n \overline{Y_{n-2j}^m(\theta_y, \phi_y)} \right\} \tag{2.5}$$

where

$$e_n^j = 4\frac{(n-j+1)!(j+1/2)!}{(n-j+1/2)!j!}$$

and Y_n^m are the normalized spherical harmonics.

Figure 2.2. By associating a constant number of coefficients at center O, we can calculate the irradiance received by x from a number of differential emitters. The value of the coefficients depends upon the location of these emitters, and the recipient has to be sufficiently far.

Substituting (2.5) in (2.4) and rearranging terms, we get the *multipole expansion* in Equation 2.8 as

$$I(x) = \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} e_n^j R_{nj}^m(x) \otimes M_{nj}^m(A_y) \tag{2.6}$$

$$R_{nj}^m(x) = \frac{\rho(x)}{r_x^{n+4}} Y_{n-2j}^m(\theta_x, \phi_x) \mathbf{RM}(x) \tag{2.7}$$

$$M_{nj}^m(A_y) = \int_{A_y} r_y^n \overline{Y_{n-2j}^m(\theta_y, \phi_y)} \mathbf{SM}(y) B(y) dA_y \tag{2.8}$$

Here, **RM**(x) and **SM**(y) stands for the receiver and the source matrices respectively. The intuition for this step is shown in Figure 2.2. For practical implementation, the summation to infinity is truncated to some p terms. We have theoretically and experimentally verified [24] that the error incurred is very small.

Since the FMM algorithm is hierarchical, we need a way to collect irradiance, as shown in Figure 2.3.

2.2.2.2 Local Expansion

Equation 2.6 may be viewed as an irradiance gather process "outside" the sources. We need a similar expression on how irradiance collected at a center is distributed to receivers. For $r_x < r_y$, we derive [23] the so-called *local expansions* in terms of the coefficients L_{nj}^m. Our intuition behind this formulation is explained in Figure 2.4.

Similar to the multipole coefficients, the *local coefficients* L_{nj}^m are also additive, and can be translated to a different coordinate system. We illustrate this in Figure 2.5.

Finally, a very important result is ilustrated in Figure 2.6.

Figure 2.3. Multipole coefficients are additive and can be translated to a different coordinate system. This enables a hierarchical approach by considering the effect of several clusters. For each cluster $C_1, C_2, C_3, \ldots C_k$, the *multipole coefficients* $M_{nj}^m(A_y)$ are first accumulated at the respective origins and then "translated" to get the cumulative effect of the entire set of clusters.

Figure 2.4. The irradiance stored at a virtual point O in the form of a constant number of coefficients can be dissemenated to different receivers. This is valid only if the receiver points are "close by."

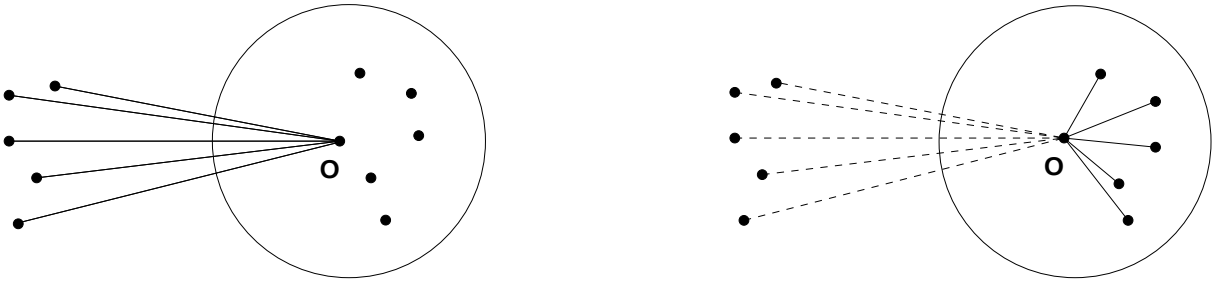


Figure 2.5. Local coefficients are additive. On the left, we first collect the cumulative local coefficient of several clusters from the local coefficients of each cluster and accumulate it in the center O. We then disseminate it to the recipients.

2.2.2.3 Assigning Weights to Points

The equations in the previous section assume that we are in a position to integrate over a surface area. In our earlier work, we had assumed triangles as input and we performed Gaussian quadrature to calculate the integral exactly. For PBMs, we do not have any surface information; we therefore approximate this integration. Weights are assigned to each point and signify the contribution of the point to the reconstruction of the surface. This is a local property based on the normal available at points. As the number of points increase, the integration is computed more accurately.

In summary, we can define the multipole coefficients (and similarly local coefficients) for a point y as

$$M_{nj}^m(y) = w(y) r_y^n \overline{Y_{n-2j}^m(\theta_y, \phi_y)} \mathbf{S} M(y) \tag{2.9}$$

We thus replace the interaction between surfaces and points (in Equation 2.8) as between

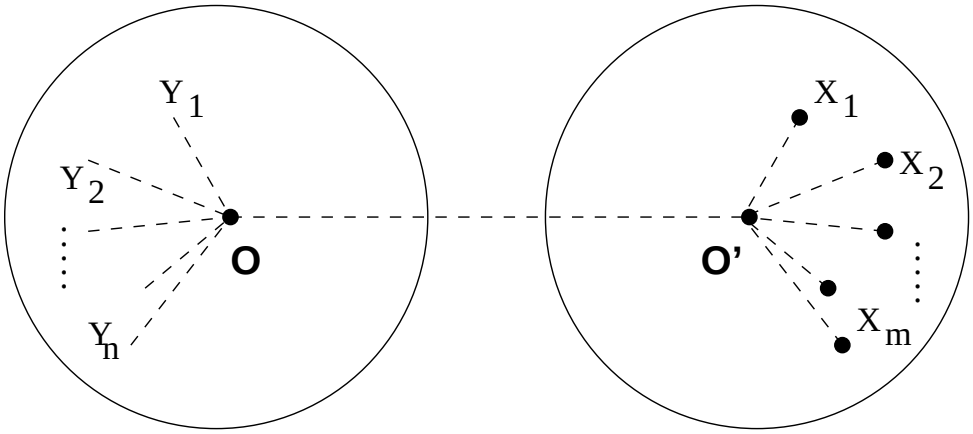


Figure 2.6. A crucial part of FMM is the conversion of multipole coefficients at a given center into local coefficients at another center. The multipole to local translation converts the multipole coefficients of a set of N source points into local coefficients for a set of M receiver points.

points only. This interaction is termed as a *particle interaction*.

2.2.3 The FMM Algorithm

A brief version of the algorithm is given here for the sake of completeness.

1. **Setup**: We start with the given input point model. All points are arranged in an adaptive octree such that no leaf node contains more than $s = O(1)$ points. With each node, we associate two set of disjoint nodes:
   - *near neighbors* of a node b are nodes that share a common boundary point of the node. Points in these nodes do not satisfy the distance constraint in (Equation 2.6).
   - *interaction list* of a node b are the children of the near neighbors of the parent of b — children who are *not* near neighbors of b itself. When occlusion is present in the scene, the interaction list is modified as in Section 2.2.4.
2. **Upward Pass**: For each leaf node in the octree, we calculate the multipole coefficients of all points contained in the node about its center. Then, for each level (starting from the penultimate level) we calculate the multipole coefficients of each node at that level by translating and accumulating the multipole coefficients of its children.
3. **Downward Pass**: For each level (starting from the second), the local coefficients at each node b are calculated by converting the multipole coefficients of boxes in the interaction list of b into local coefficients about b's center using the multipole to local translation algorithm (Figure 2.6). Additionally, the local expansion coefficients obtained from the individual points contained in the local interaction list are aggregated.
4. **Evaluation**: For each leaf b in the octree, for each evaluation point $x \in b$, the local expansion about the center of b is evaluated at x.

We iterate over these steps till sufficient convergence is reached. The evaluation points are the same points that represent the input point model.

2.2.4 Visibility in Point Models

Visibility is not considered in the original FMM algorithm. For our purposes it is complicated in that occlusion is a point to point based phenomenon and not a node to node phenomenon

where the bulk of the computation occur. In this section we first give a point to point visibility algorithm. Later we incorporate it in the FMM context.

2.2.4.1 Point–Point Visibility

Since our input data set is a point based model with *no connectivity information*, we do not have knowledge of any intervening surfaces occluding a pair of points. Theoretically, it is therefore impossible to determine exact visibility between a pair of points. Thus, we restrict ourselves to *approximate visibility* with a value between 0 and 1. Consider two points p and q (as in Figure 2.2.4.1 in the input scene on which we run a number of tests to efficiently produce $O(1)$ possible occluders.

First we apply the culling filter to straightway eliminate backfacing surfaces.

$$n_p \cdot \overline{pq} > 0 \quad \text{and} \quad n_q \cdot \overline{qp} > 0$$

where n_p and n_q are normals at point p and q respectively.

If the above condition is satisfied, we then determine the possible occluder set X (—X—=k). This is a set of points in the point cloud which are close to $\overline{pq}$ and thus can possible affect the visibility. These points lie in a cylinder around $\overline{pq}$. In Figure 2.2.4.1, for example, x_3 is dropped. This set is further pruned by considering the tangent plane at each potential occluder. If the tangent plane does not intersect $\overline{pq}$ the occluder is dropped (for example, x_1 in Figure 2.2.4.1). A final pruning happens by measuring the *distance along the tangent* to $\overline{pq}$. We pick the smallest $O(1)$ occluders (equal to 3 in our implementation) using this distance metric. We compute a visibility fraction based on this distance. This results in Algorithm `point_visible`.

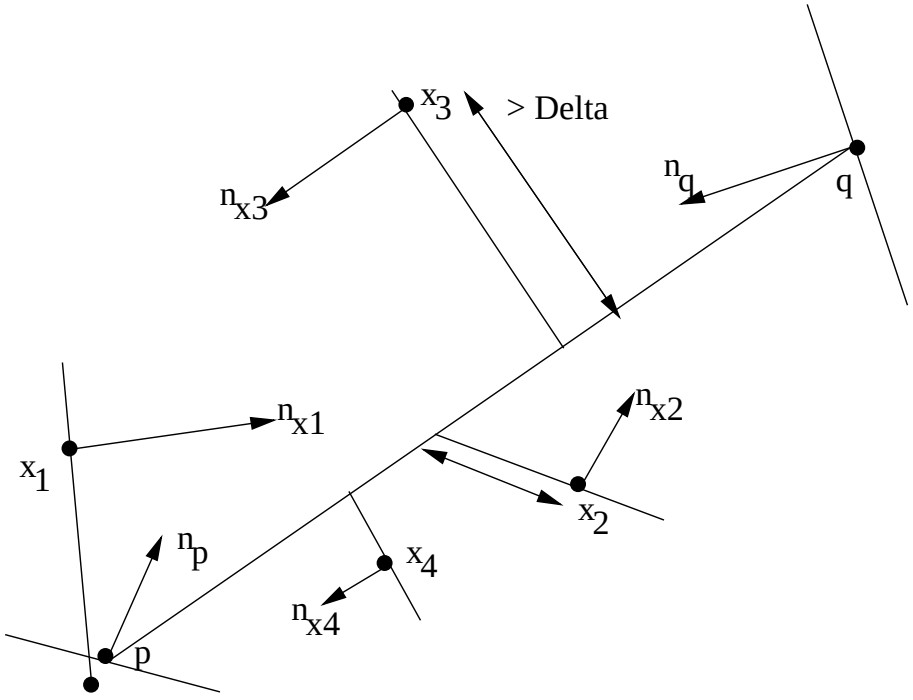


Figure 2.7. Only x_2 and x_4 will be considered as occluders. We reject x_1 as the intersection point of the tangent plane lies outside the line segment $\overline{pq}$. x_3 has earlier been rejected because it is more than a distance Δ from the line segment $\overline{pq}$.

```
Procedure point_visible(Point p, Point q)
Define threshold t_1, visible_{p,q} = 1
if FacingEachOther(p,q) then
  Find k closest points in region Δ around pq
  Prune based on tangent plane
  for i = 0 to 2 do
    contributeVis_i = visibility_look_up(distance_i)
    visible_{p,q} = visible_{p,q} * contributeVis_i
  end for
  if (visible_{p,q}) > t_1 then
    return(visible)
  end if
end if
```

2.2.4.2 Hierarchical Visibility

In this section, we incorporate the visibility into the FMM algorithm (Section 2.2.3). Recall that the object space composed of points was divided into an adaptive octree. Note that each point receives energy from every other point either directly, or through the points in the interaction list of the ancestor of the leaf it belongs to. The key idea is to modify the interaction list much the way the adaptive version of the FMM works.

If the points in a node c in the interaction list of node b are completely visible from *every* point in b, then the *visibility state* of the pair (b,c) is said to be *valid*. If, on the other hand, no point in c is visible from any point in b, the visibility state of the pair (b,c) is said to be *invalid*. The node c is dropped from the interaction list since no exchange of energy is permissible. Finally, when the visibility state is *partial*, we *postpone* the interaction. In the sequel, we ensure that the postponed interaction happens at the lowest possible depth (the root is at depth 0) for maximum efficiency. This is done by extending the notion of point–point visibility to the node level as follows.

2.2.4.3 Point–Leaf Visibility

In this section, we determine the visibility between a leaf node C and a point p. We start by making point to point visibility calculations between point p and every point $p_i \in C$. This results in Algorithm `point_Leaf_visibility`.

```
Procedure point_Leaf_visibility(Point p, Leaf L)
Declare threshold t_2, Visi_point_L = 0
for each point p_i ∈ L do
  state = point_visible(p, p_i)
  if equals(state,visible) then
    Visi_point_L = Visi_point_L + 1
    if Visi_point_L > threshold t_2 then
      return(visible)
    end if
  end if
end for
return(invisible)
```

2.2.4.4 Leaf–Leaf Visibility

Similarly we determine visibility between two leaf nodes C and L. For every point $p_i \in L$, we start by calculating *Point–Leaf Visibility* between point p_i and C. This results in Algo-

rithm `Leaf_Leaf_visibility`.

```
Procedure Leaf_Leaf_visibility(Leaf L, Leaf C)
Declare threshold t_3, Visi_point_L = 0
for each point p_i ∈ C do
   state = point_cell_visible(p_a, Leaf L)
   if equals(state,visible) then
      Visi_point_L = Visi_point_L + 1
   end if
end for
if Visi_point_L > threshold t_3 then
   return(visible)
end if
return(invisible)
```

2.2.4.5 Node–Node Visibility

In this section, we determine the visibility between nodes A and B of the octree. We start by computing visibility of all $b \in Leafnodes(B)$ to all $a \in Leafnodes(A)$. If all are visible, the status is valid. If none are visible, the state is invalid. Otherwise, we have partial visibility. In this scenario, we repeat the procedure *Node–Node Visibility* for all the child nodes of A and B. Note that there is no case of partial visibility between leaf nodes. The algorithm is summarized below.

```
Procedure Node_Node_visibility(Node A, Node B)
Declare vis_cnt = 0
for each a ∈ leafcell(A) do
   for each b ∈ leafcell(B) do
      state = Leaf_Leaf_visible(a, b)
      if equals(state,visible) then
         vis_cnt = vis_cnt + 1
      end if
   end for
end for
if equals(vis_cnt,LeafNode(A).size*LeafNode(B).size) then
   return(valid)
else if equals(vis_cnt,0) then
   return(invalid)
else
   return(partial)
end if
```

2.2.4.6 Computing Interaction Lists

We now are in a position to compute the interaction list as in Algorithm `Octree_Visibility`. The complexity of this algorithm is around $O(N^2 logN)$.

```
Procedure Octree_Visibility(Node A)
for each node B ∈ interactionlist(A) do
  if notLeaf(A) then
    state=Node_Node_Visibility(A,B)
  else if Leaf(A) then
    state=Leaf_Leaf_Visibility(A,B)
  end if
  if equals(state,valid) then
    Retain B in interactionlist(A)
  else if equals(state,partial) then
    for each a ∈ children(A) do
      for each b ∈ children(B) do
        interactionlist(a).add(b)
      end for
    end for
  else if equals(state,invalid) then
    interactionlist(A).remove(B)
  end if
end for
for each R ∈ child(A) do
  Octree_Visibility(R)
end for
```

2.2.4.7 Visibility Cone Optimization

The algorithm in the previous subsection works well when there are unoccluded environments where every node is visible to every other node. This is because no postponement will happen. In highly occluded environments, there is a possibility of duplication of the leaf-leaf visibility. One way of avoiding duplication is to construct conservative *visibility cones* [18] which have the property that any object in the visibility cone is *invisible* (the converse is not true).
We adapt this idea to build a visibility cone for every leaf, and recursively build visibility cones for all nodes.
Once such cones are available, we modify Algorithm `Octree_Visibility`. The `for` loop in the first step of the algorithm receives a pruned set of nodes. Specifically, suppose we have a node B in the interaction list of A. We first check whether B lies within the visibility cone of A by a *cheap visibility cone intersection test*. If yes, we immediately know that the state is invalid. On the other hand, if B lies partially inside, or completely outside the visibility cone of A, we cannot say what the state is. When pruning happens at low depths, substantial savings result since the visibility cone test is cheap.

2.3 Sample Results

In this section, we first provide evidence of the correctness of our visibility algorithm, showing the relevant results. Later we show how much closer are we to the desired radiosity output.

2.3.1 Correctness of visibility

Figure 2.8 and Figure 2.9 shows results which ascertain the correctness of the visibility algorithm implemented. The yellow cells in the figure indicate the cells visible to the point colored in cyan. Correct visibility ensures that the energy transfer correctly takes place between nodes in FMM.

The rendering was done taking OpenGl points as primitives. Note that rendering is not important here. The emphasis here is to prove the correctness of the visibility algorithm used and

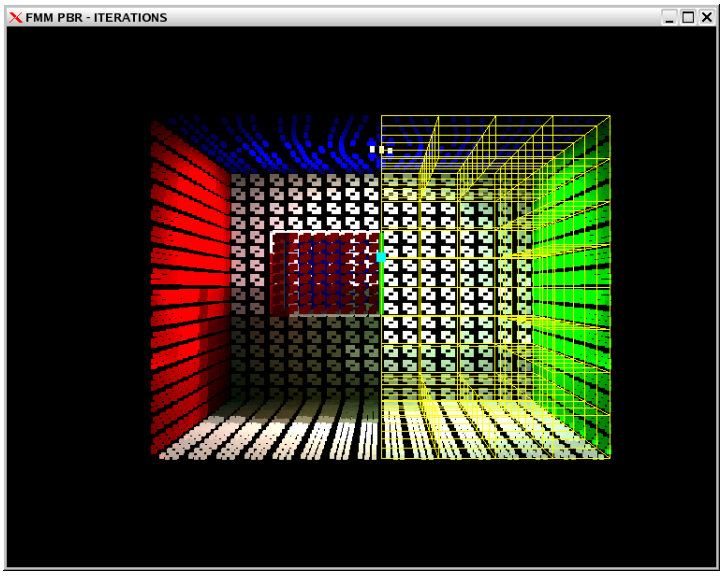


Figure 2.8. Figure showing correctness of the visibility algorithm

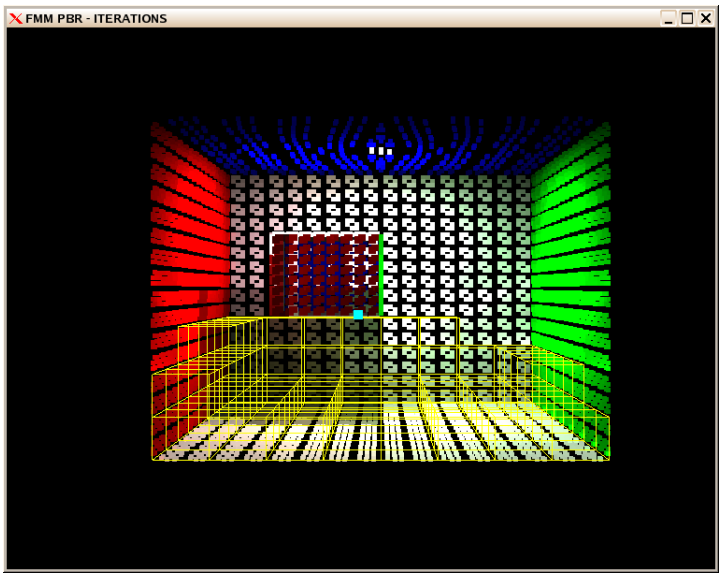


Figure 2.9. Figure showing correctness of the visibility algorithm

not the rendering.

Another thing to note here are the **color bleeding** effects (on the back white wall) and the **soft shadows** (of the box) we get, which also gives us the feel of *Global Illumination* taking place.

Rendering was done on a 2.4 GHz Pentium IV machine with 1GB RAM, *without* any GPU in place.

2.3.2 Surfel Rendered output

Figure 2.10 shows results of our FMM algorithm against a output generated with radiosity rendering equation. We see certain artifacts appearing on the walls in the output we have generated, which we are trying to resolve.

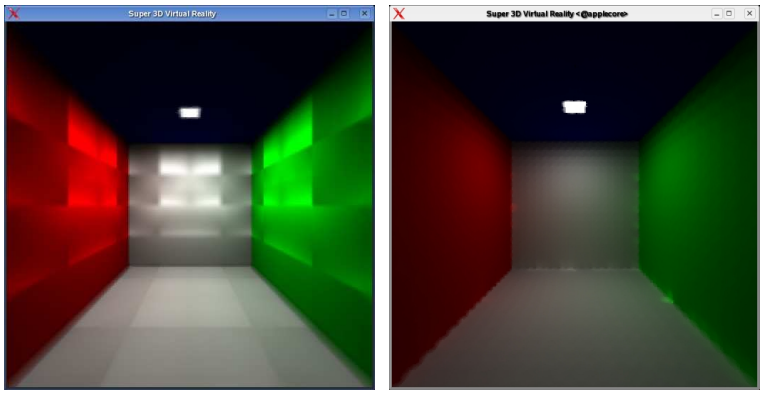


Figure 2.10. Figure showing FMM generated output againest a output generated with directly solving the radiosity equation

One of the major challenge of Point Based Rendering (PBR) algorithms is to achieve a continuous interpolation between discrete point samples that are irregularly distributed on a surface. A couple of PBR algorithms are available in the literature like QSplat, Surface splatting etc. We used surface splatting renderer as a final module in our system. We needed to make some change so that our outputs are compatible with the input format required by the renderer. Also, we had to eliminate the lightening calculations made by the renderer so that we can insert our calculated *illumination maps* into it.

Rendering was done on a 2.4 GHz Pentium IV machine with 1GB RAM, *without* any GPU in place.

2.3.3 Computational Advantage

The graph [23] above shows the computational advantage we get over the brute force algorithm. The break even point for the fast multipole method vs. the brute force method is very high (about $N = 10000$). However, for problem sizes much higher (which normally is the case for point models where the size go to billions of points) , the FMM exhibits a linear increase in time taken whereas the brute force time will blow up.

2.4 Final remarks

In summary, i will list out the important points we have gone through this report.
Problem Statement: To compute **illumination maps** for **inter-reflections** in complex scenes represented as **point-models** using **Fast Multipole Method**. The system must handle **occlusions** present in the scene as well

- Our algorithm is designed to work for point models
- We use FMM to solve the radiosity rendering equation for Global Illumination
- FMM provides us with a linear time approach to solve the rendering equation in almost linear time
- We have algorithms defined to handle Point-Point Visibility
- We have extended the visibility algorithm to fit into the FMM context
- We have modified the Point-based Renderer to suit our requirement

N	s	FMM Time(s)	Brute Force Time(s)
110528	900	30.568	130.412
36600	300	15.975	51.201
12632	200	7.302	9.261
1658	100	1.112	0.575

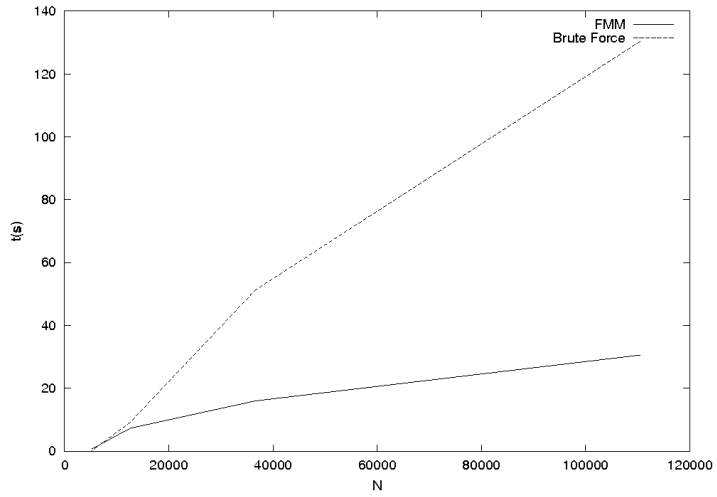


Figure 2.11. Results for a single iteration with different values of N and a chosen value of s for the cornell box scene.

Chapter 3

Future Work

In this chapter, I brief, in order of interest, some of the ideas that I visualize as interesting problems to venture in my dissertation.

3.1 Ideas

1. The immediate goal is to take measures to remove the artifacts from the rendered image. I have debugged this problem to its source, which lies in the basic mathematics on which this whole system was built. I look forward to have a deep insight into the math involved and solve it so as to get visually pleasing results.

2. I also intend to come up with a good *Hybrid Rendering System*, which can work even if both point as well as triangle models are given as inputs. There has to be a trade off as to when to render triangles and when to render points.

3. I also plan to optimizing the visibility code in terms of speed, retaining/improving the quality at same time. Right now, I almost have a $O(N^2 logN)$ algorithm. We need to improve the timing keeping in mind that the quality of the rendered image is not affected.

4. I also plan to extend the visibility algorithm to fit onto a parallel architechture framework. Our visibility algorithm is amazingly parallel in nature and we can hope to get better timings once this is implemented.

5. I would also like to parallelize FMM so as to get the best of the FMM's fast rendering speed

6. I would also like to take up some related problems to point based modelling like *Point Model Segmentation*, which I feel will be much beneficial in helping out solve other problems in point based modelling. The aim here is to segment the input scene consisting of various point based models into different objects in the 3D space. Once done, we can:

   - construct surfaces on individual objects.
   - run all the algorithms available for triangulated models on it.
   - improve visibility in terms of timing and quality (it won't be *approximate* any more.
   - Helps out in developing the Hybrid Renderer
   - Helps in collision detection, deformable models. etc.

7. Collision Detection, Deformable Point Models, Level of Detail control in point models are also very interesting problems I would like to give some thought to.

Chapter 4

Conclusion

We have discussed the basic concepts of Global Illumination and Point Models in Chapter 1. In Chapter 2, we state the problem definition (Section 2.1.1) followed by our contributions in Section 2.1.2. In Section 2.2 and Section 2.2, we provide the details of our approach. In Section 2.3, we provide our results. Finally in Chapter 3, we discuss some of the directions/problems I plan to work for my dissertation.

In Appendix A, I present an overview of techniques for simulating light transport with an emphasis on the radiosity method. In Appendix B, I present the theoretical foundations of the Fast Multipole Algorithm. Finally, in Appendix C, we present the mathematical apparatus required to apply the FMM to radiosity.

We saw that the FMM method is elegant because it trades off error with quality in a disciplined quantitative way. We have made the kernel of the energy balance in the rendering equation conformant to the FMM by deriving the near and far field expansions. The illumination problem over surfaces is reduced to a solution over points enabling point based rendering. We have also given a new visibility algorithm for point based models. Both these steps can be viewed as a 'preprocessing' step for photo-realistic global illumination of complex point-based models.

APPENDIX

Appendix A

Light and Global Illumination

A.1 Introduction

What exactly is Light? The struggle of the intellect to decipher Nature's secrets has resulted in a plethora of theories. Around 500 BC, Empedocles, the greek philosopher, came to the conclusion that Aphrodite made the human eye out of the four elements (fire, air, earth and water) and that "she lit the fire in the eye which shone out from the eye making sight possible". Newton, Maxwell, Einstein et al contributed to the acceptance of the dual nature of light - as electromagnetic radiation or photons propagating through space. Superstring theorists now hold that light is nothing but a vibration of the fifth dimension in a ten dimensional hyperspace. Our interest in light is not to exactly describe its behavior but rather to accurately simulate visual phenomenon created by the interplay of light and reflecting surfaces. For our purposes, an extremely simplified form of the theory of light as an electromagnetic radiation suffices. To this extent, we present some important physical quantities and the light transfer equation on which much of computer graphics has developed. This chapter also presents an overview of previous work in the field of global illumination.

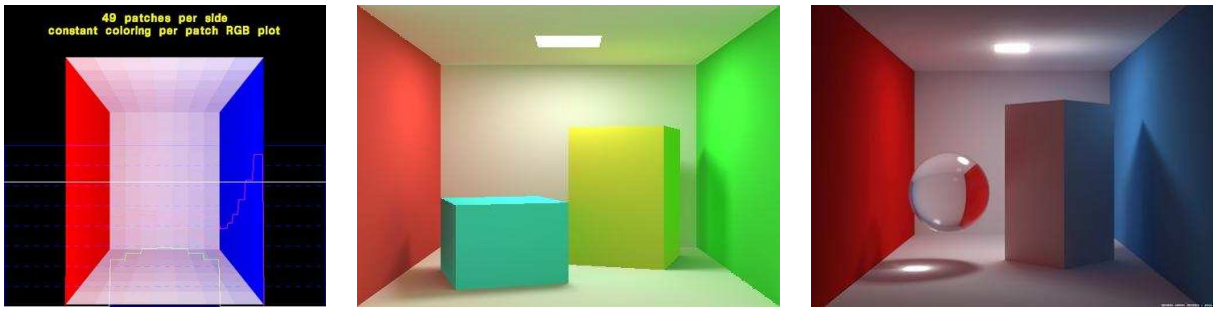


Figure A.1. Progress in the field of global illumination. From Left: The original simulated cornell box by the finite element method[14], cornell box with shadows, cornell box with non diffuse elements[21]

A.2 Radiometry and Photometry

Radiometry is the science of measuring radiant energy transfers. These transfers can be characterized by a set of physical quantities described below. The radiometric quantities presented here are considered to be independent of time, polarization, and wavelength. *Photometry* uses elements from perceptual psychology to predict subjective impressions caused by the physical process of illumination.

A.2.1 Radiance

The most fundamental quantity to describe radiant energy transfer is *radiance*. Radiance $L(x, \vec{\omega})$ is defined as the amount of energy traveling at some point x in a specified direction $\vec{\omega}$, per unit area perpendicular to the direction of travel, per unit solid angle. Since radiance is defined "per unit solid angle", it does not attenuate with distance. Most light receivers, including the human eye and photographic cameras, are directly sensitive to radiance and therefore the knowledge of radiances leaving all surfaces in a scene is sufficient to render a picture.

A.2.2 Radiosity

We are primarily interested in the simulation of diffuse illumination, i.e. surfaces that reflect light equally in all directions. In this case, the direction of radiation has no importance and the physical quantity of *radiosity* proves more useful. *Radiosity* $B(x)$ at a surface point x is defined as the total power radiated from that point over all directions. Radiosity is a useful quantity since it characterizes the total radiation leaving a surface locally around a given point. A corresponding quantity *irradiance* represents the total incident radiation at a given point.

A.2.3 Color

Both radiance and radiosity are measures with respect to a specific wavelength, and are thus independent of the human visual system. The physical basis of color is the variation of light intensity with wavelength. The color of a given radiation is completely described by its *spectrum*, i.e., the relative intensity of the radiation at all visible wavelengths. However, to characterize the color perceived by a human observer, it is not necessary to specify the spectrum at all wavelengths. Research in human vision shows that color can be represented in a three dimensional space due to the presence of three different types of color-sensitive receptor cells on the retina. The majority of applications in computer graphics utilize relative strengths at the red, green, and blue wavelengths to characterize color. This technique is not perceptually accurate and more complex schemes do exist[35], however, all these techniques work by calculating the intensity at a few particular wavelengths, independent of all other wavelengths.

A.3 The Rendering Equation

Ignoring participating media (such as smoke), and concentrating solely on the interaction of light with scene surfaces, the global illumination problem can be captured in the following *rendering equation*[22].

$$L(x, \vec{\omega}) = L_e(x, \vec{\omega}) + \int_{\Omega} \rho(x, \vec{\omega} \rightarrow \vec{\omega}_i) L_i(x, \vec{\omega}_i) \cos\theta_x d\omega_i \tag{A.1}$$

- $L(x, \vec{\omega})$ is the total radiance leaving x in the direction $\vec{\omega}$
- $L_e(x, \vec{\omega})$ is the radiance directly emitted from x in the direction $\vec{\omega}$
- $\rho(x, \vec{\omega} \rightarrow \vec{\omega}_i)$ is the fraction of radiance incident from direction $\vec{\omega}_i$ that is reradiated in direction $\vec{\omega}$
- $L_i(x, \vec{\omega}_i)$ is the radiance incident on x from the direction $\vec{\omega}_i$
- θ_x is the angle between the surface normal at x and $\vec{\omega}_i$
- Ω is the hemisphere lying above the tangent place of the surface at x

Stated plainly, the equation states that the radiance emitted from a surface point x in the direction $\vec{\omega}$ is equal to the radiance the surface emits in that direction by itself in addition to the integral over the hemisphere of the incoming radiance that is reflected in the same direction $\vec{\omega}$.

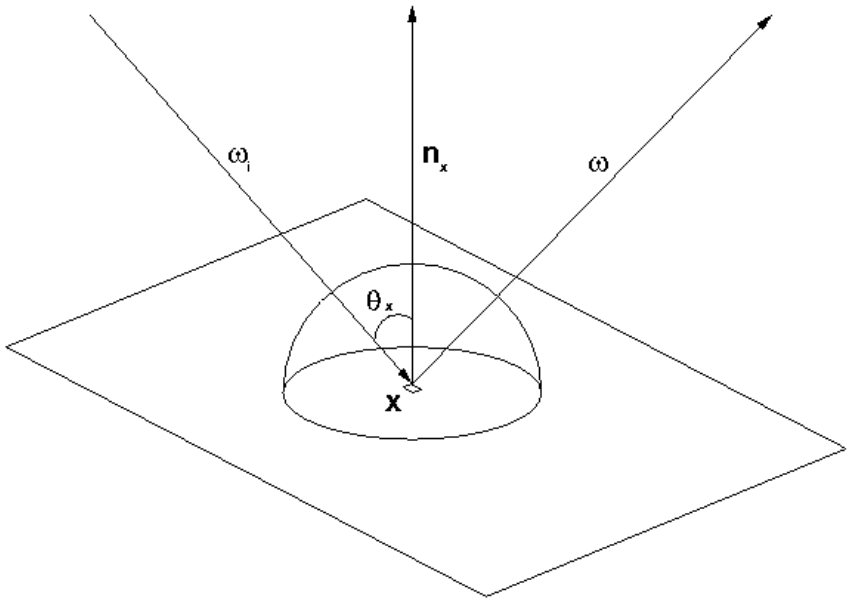


Figure A.2. The rendering equation models the energy balance at point x by expressing the outgoing radiance in terms of the various sources of incoming radiant energy - emitted and reflected. In the diagram, w_i varies over the hemisphere of directions above point x.

The rendering equation is actually a *Fredholm equation of the second kind* which has the general form

$$f(x) = g(x) + \int_a^b K(x,y)f(y)dy \tag{A.2}$$

where $f(x)$ is the unknown function, $g(x)$ is a known function, and $K(x,y)$ is called the *kernel* of the integral operator. Except for the most trivial of cases, it cannot be solved analytically.

A.3.1 The Diffuse Assumption

If we assume that all surfaces reflect light diffusely, i.e., incoming radiation is dissipated equally in all directions, then the radiance at a point loses its directional dependence, i.e. $L(x,\vec{\omega}) \equiv L(x)$ and we have

$$L(x) = L_e(x) + \rho(x) \int_\Omega L_i(x,\vec{\omega}_i) \cos\theta_x d\omega_i \tag{A.3}$$

Using $d\omega = ((\cos\theta)/r^2)dA$, and $B(x) = \pi L(x)$, we can rewrite (A.3) as the *radiosity equation*:

$$B(x) = E(x) + \rho(x) \int_{y \in S} K(x,y)V(x,y)B(y)dy \tag{A.4}$$

where

$$K(x,y) = \frac{\cos\theta_x \cos\theta_y}{\pi|x-y|^2} \tag{A.5}$$

and

- $B(x)$ is the total radiosity at point x
- $E(x)$ is the emittance at point x
- $\rho(x)$ is the reflectance at point x
- $V(x,y)$ is the visibility between points x and y
- $K(x,y)$ is the kernel of the integral operator between points x and y

A.3.2 Discrete Formulation

Suppose the surfaces of the environment are subdivided into a collection of N disjoint patches P_j $j = \{1 \dots N\}$ with surface areas A_j $j = \{1 \dots N\}$. The integral over all surfaces in (A.4) is simply broken into N pieces, each corresponding to a discrete patch:

$$B(x) = E(x) + \rho(x) \sum_{j=1}^{N} \int_{y \in P_j} K(x,y)V(x,y)B(y)dy \tag{A.6}$$

Due to the assumption of constant radiosity over a patch, at each point y in P_j, $B(y) = B_j$, and the radiosity can be moved outside the integral:

$$B(x) = E(x) + \rho(x) \sum_{j=1}^{N} B_j \int_{y \in P_j} K(x,y)V(x,y)dy \tag{A.7}$$

The incoming radiosity and outgoing emittance of a patch are averaged over the surface:

$$B_i = \frac{1}{A_i} \int_{x \in P_i} B(x)dx \qquad\qquad E_i = \frac{1}{A_i} \int_{x \in P_i} E(x)dx \tag{A.8}$$

Since reflectance is also assumed constant over a patch, we denote $\rho(x) = \rho_i$ and we obtain

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j \frac{1}{A_i} \int_{x \in P_i} \int_{y \in P_j} K(x,y)V(x,y)dydx \tag{A.9}$$

More concisely,

$$B_i = E_i + \rho_i \sum_{j=1}^{N} F_{ij} B_j \tag{A.10}$$

where

$$F_{ij} = \frac{1}{A_i} \int_{x \in P_i} \int_{y \in P_j} \frac{\cos\theta_x \cos\theta_y}{\pi r^2} V(y,x)dydx \tag{A.11}$$

is called the *form factor* between patches P_i and P_j.

A.3.3 Solution to the Radiosity Equation

Since the radiosity equation (A.10) is written for a single patch, there exists a system of N linear equations with N unknowns (the radiosities of each of the N patches). Assuming all the coefficients of these equations are known, any linear equation solver can be used to extract the radiosity values from the system.

The N instances of (A.10), obtained by considering all possible values for i, can be grouped to form the following matrix equation:

$$\begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix} + \begin{pmatrix} \rho_1 F_{11} & \rho_1 F_{12} & \dots & \rho_1 F_{1n} \\ \rho_2 F_{21} & \rho_2 F_{22} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \rho_n F_{n1} & \dots & \dots & \rho_n F_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} \tag{A.12}$$

or equivalently

$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & \dots & \dots & 1-\rho_n F_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix} \tag{A.13}$$

Obtaining an exact solution to the matrix equation is equivalent to inverting the form factor matrix. Inversion of the matrix has complexity $O(N^3)$. Iterative methods such as Jacobi, Gauss-Seidel or Southwell relaxation can be used to bring down the complexity to $O(kN^2)$ where k is a constant.

Appendix B

The Fast Multipole Method

B.1 The Fast Multipole Method

The Fast Multipole Method [15, 17, 4] is concerned with evaluating the effect of a "set of sources" $\mathbb{X}$, on a set of "evaluation points" $\mathbb{Y}$. More formally, given

$$\mathbb{X} = \{x_1, x_2, \ldots, x_N\}, \quad x_i \in \mathbb{R}^3, \quad i = 1, \ldots, N, \tag{B.1}$$

$$\mathbb{Y} = \{y_1, y_2, \ldots, x_M\}, \quad y_j \in \mathbb{R}^3, \quad j = 1, \ldots, M \tag{B.2}$$

we wish to evaluate the sum

$$f(y_j) = \sum_{i=1}^{N} \phi(x_i, y_j), \quad j = 1, \ldots, M \tag{B.3}$$

The function ϕ which describes the interaction between two particles is called the "kernel" of the system (e.g. for electrostatic potential, kernel $\phi(x, y) = |x-y|^{-1}$). The function f essentially sums up the contribution from each of the sources x_i.

Assuming that the evaluation of the kernel ϕ can be done in constant time, evaluation of f at each of the M evaluation points requires N operations. The total complexity of this operation will therefore be $O(NM)$. The FMM attempts to reduce this seemingly irreducible complexity to $O(N + M)$ or even $O(N \log N + M)$ by making a key observation that most application domains do not require that the function f be calculated at high accuracy. In other words, a specified acceptable accuracy of computation ϵ is available to us. This observation, although simple, is absolutely crucial for our purposes as once it is admitted that the result of a calculation is needed only to a certain accuracy, approximations can be used.

B.2 Kernel Properties

The Fast Multipole Method cannot be applied to an arbitrary kernel. The kernel must have the following properties:

- **Multipole Expansion**. The kernel must be "degenerate", i.e. can be evaluated as a series expansion about an arbitrary spatial point $x_c \neq x$. The expansion is expected to be valid outside a sphere centered at x_c with radius $|x - x_c|$:

  $$\phi(x, y) = \mathbf{M}(x, x_c) \circ \mathbf{S}(y, x_c), \quad |y - x_c| \geq |x - x_c| \tag{B.4}$$

  $\mathbf{M}$ and $\mathbf{S}$ are tensor objects in p-dimensional space representing the series coefficients and basis functions respectively. The $\circ$ denotes a contraction operation between these objects, distributive with addition (e.g. dot product of vectors of length p):

  $$(u_{i_1}\mathbf{M}_{i_1} + u_{i_2}\mathbf{M}_{i_2}) \circ \mathbf{S} = u_{i_1}\mathbf{M}_{i_1} \circ \mathbf{S} + u_{i_2}\mathbf{M}_{i_2} \circ \mathbf{S} \tag{B.5}$$

- **Local Expansion**. The kernel must have a complementary expansion valid inside a sphere centered at y_c with radius $|x - y_c|$:

$$\phi(x, y) = \mathbf{L}(y, y_c) \circ \mathbf{R}(x, y_c), \quad |y - y_c| \leq |x - y_c| \tag{B.6}$$

As before, $\mathbf{M}$ and $\mathbf{R}$ are tensor objects in p-dimensional space representing the series coefficients and basis functions respectively. Figure 1 depicts the domain of validity for these expansion pictorially

Figure B.1. Constraints on the kernel for application of FMM. Left: Multipole Expansion, Right: Local Expansion. D indicates the domain of validity of the corresponding expansions

The usefulness of the multipole expansion can be seen by substituting (B.4) in (B.3):

$$f(y_j) = \left(\sum_{i=1}^{N} \mathbf{M}(x_i, x_c) \right) \circ \mathbf{S}(y_j, x_c), \quad j = 1, \ldots, M \tag{B.7}$$

Observe that the quantity in the brackets remains constant for each of the M evaluations of f. This quantity can be calculated once for all M points. Assuming that the contraction operator takes constant time, the $O(NM)$ complexity is reduced to $O(N + M)$. However, note that the use of (B.4) places constraints on the location of the source and evaluation points.

B.3 Properties of the expansion coefficients

Consider the multipole expansion. For the same source particle, we can have separate basis functions for multipole expansions about different centers. However, in their common domain of validity, they will evaluate to the same value. The conversion of coefficients of an expansion about a particular center to the coefficients of expansion about another center is called a translation. The following translations must be supported by the basis functions derived above.

- **Multipole-Multipole**. Consider the multipole expansion (B.4) about the point x_c valid for evaluation at $y \in |y - x_c| \geq |x - x_c|$. The multipole expansion about x_c can be translated to a multipole expansion about another point x_c' by the multipole translation operator $(\mathbf{S}|\mathbf{S})$:

$$\mathbf{M}(x, x_c') = (\mathbf{S}|\mathbf{S})(x_c', x_c)[\mathbf{M}(x, x_c)] \tag{B.8}$$

- **Multipole-Local**. Consider the multipole expansion (B.4) about the point x_c valid for evaluation at $y \in |y - x_c| \geq |x - x_c|$. The multipole expansion about x_c can be translated to a local expansion (B.6) about another point y_c valid for evaluation at $y \in |y - x_c| \geq |x - x_c| \cup |y - y_c| \leq |y_c - x_c|$ by the multipole translation operator $(\mathbf{S}|\mathbf{R})$:

$$\mathbf{L}(y, y_c) = (\mathbf{S}|\mathbf{R})(y_c, x_c)[\mathbf{M}(x, x_c)] \tag{B.9}$$

- **Local-Local**. Consider the local expansion (B.6) about the point y_c valid for evaluation at $y \in |y - y_c| \leq |x - y_c|$. The local expansion about y_c can be translated to a local expansion about another point $y_c' \in |y - y_c| \leq |x - y_c|$ by the local translation operator $(\mathbf{R}|\mathbf{R})$:

$$\mathbf{L}(y, y_c') = (\mathbf{R}|\mathbf{R})(y_c', y_c)[\mathbf{L}(y, y_c)] \tag{B.10}$$

B.4 Hierarchical Spatial Domains

The data structure on which the fast multipole method is based is the three dimensional octree (extensions to arbitrary dimensions also exist[19]). To simplify matters, we make the following removable restrictions:

- the set of sources $\mathbb{X}$ is the same as the set of evaluation points $\mathbb{Y}$ and
- these points are distributed uniformly in space.

To construct the octree, we start with the bounding box of the system - the cube that encloses all the particles. This box is assigned to level 0. The level 0 box can be divided into eight smaller boxes of equal size by dividing each side in half. All boxes of this size are assigned to level 1. This procedure is repeated to produce a sequence of boxes at level 2, level 3, and so on. At level l of an octree, we have 8^l boxes. The procedure is terminated when there are no more than a constant number of particles s in each box. Let this condition be reached at level l_{max}. Then $8^{l_{max}} s > N$ or $8^{l_{max}} = O(N)$. All boxes at level l_{max} are called the leaves of the tree.
With each box b throughout the hierarchy, we associate the following operations:

- $P(b)$ denotes the parent of the box b.
- $C(b)$ denotes the boxes obtained by subdivision of b. Note that $|C(b)| = 8$.
- $N(b)$ denotes the *near neighbors* of the box b. Near neighbors of a box are boxes at the same level that share a common boundary point. Note that $b \in N(b)$. Additionally, the maximum number of near neighbors a box can have in an octree is $\max(|N(b)|) = 27$.
- $F(b)$ denotes boxes at the same level as b which are not in $N(b)$
- $I(b)$ denotes the *interaction list* of b and is defined as $C(N(P(b)))\backslash N(b)$, i.e., children of the near neighbors of the parent which are not near neighbors of b itself. Note that the maximum size of the interaction list is $\max(|N(b)|).|C(b)| - \max(|N(b)|) = 189$.

Additionally, we use the notation b_c to refer to the center of the box b.

Theorem B.4.1 *For a given box b, if a particle $x \in \mathbb{X}$ then*

$$x \in N(b) + I(b) + I(P(b)) + I(P(P(b))) + \dots$$

Proof: We perform induction over the number of levels. The statement can be easily verified for $l = 1$ since at this level, the near neighbors of any box constitute the bounding box of the system. Suppose this statement is also true for a box b at level l ,i.e.,

$$x \in \mathbb{X} \Rightarrow x \in N(b) + I(b) + I(P(b)) + I(P(P(b))) + \dots \tag{B.11}$$

Consider the box c at level $l+1$ which is a child of b. Thus, $b = P(c)$. Substituting this in (B.11), we get

$$x \in \mathbb{X} \Rightarrow x \in N(P(c)) + I(P(c)) + I(P(P(c))) + I(P(P(P(c)))) + \ldots \tag{B.12}$$

By definition, we have that $x \in b \Rightarrow x \in C(b)$. Therefore,

$$x \in N(P(c)) \Rightarrow x \in C(N(P(c))) \tag{B.13}$$

Furthermore, by definition of $I(b)$,

$$x \in C(N(P(c))) \Rightarrow x \in N(c) + I(c) \tag{B.14}$$

From (B.12),(B.13), and (B.14), we have

$$x \in \mathbb{X} \Rightarrow x \in N(c) + I(c) + I(P(c)) + I(P(P(c))) + I(P(P(P(c)))) + \ldots \tag{B.15}$$

Hence proved.

Lemma B.4.1 *Suppose b is a box with center b_c containing particles $x_i \quad i = \{1 \ldots n\}$. Consider a point y such that $|y - b_c| > |x_i - b_c| \quad i = 1 \ldots n$. Then from (B.4)*

$$\sum_{i=1\ldots n} \phi(x, y) = \left(\sum_{i=1\ldots n} \mathbf{M}(x_i, b_c) \right) \circ \mathbf{S}(y, x_c) = \mathbf{M}(b) \circ \mathbf{S}(y, x_c) \tag{B.16}$$

$\mathbf{M}(b)$ denotes the multipole expansion coefficients of all particles in b about its center.

Lemma B.4.2 *Suppose b is a box in the hierarchy at level l.*

$$\mathbf{M}(b) = \sum_{d \in C(b)} (\mathbf{S}|\mathbf{S})(b_c, d_c)\mathbf{M}(d) \tag{B.17}$$

Theorem B.4.2 *Suppose b is a leaf box containing a particle y. Let the center of the box $P^i(b)$ (where P^i denotes i applications of the function P) be denoted by c_i. Then,*

$$\sum_{x \in \mathbb{X}} \phi(x, y) = \sum_{x \in N(b)} \phi(x, y) + \prod_{i=0}^{l_{max}-2} \mathbf{D}_i 0 \tag{B.18}$$

Where the operator

$$\mathbf{D}_i = \left(\sum_{d \in I(P^i(b))} (\mathbf{S}|\mathbf{R})(c_i, d_c)\mathbf{M}(d) \right) + (\mathbf{R}|\mathbf{R})(c_i, c_i + 1) \tag{B.19}$$

Proof: By the previous theorem, we have

$$\sum_{x \in \mathbb{X}} \phi(x, y) = \sum_{x \in N(b)} \phi(x, y) + \sum_{i=0}^{l_{max}-2} \sum_{d \in I(P^i(b))} \sum_{x \in d} \phi(x, y) \tag{B.20}$$

Now, by definitions of multipole and local expansion,

$$\begin{aligned}
\sum_{d \in I(P^i(b))} \sum_{x \in d} \phi(x, y) &= \sum_{d \in I(P^i(b))} \mathbf{M}(d) \circ \mathbf{S}(y, d_c) \\
&= \left(\sum_{d \in I(P^i(b))} (\mathbf{S}|\mathbf{R})(c_i, d_c)\mathbf{M}(d) \right) \circ \mathbf{R}(y, d_c) \\
&= \left[\left(\prod_{j=0}^{i} (\mathbf{R}|\mathbf{R})(c_j, c_j + 1) \right) \left(\sum_{d \in I(P^i(b))} (\mathbf{S}|\mathbf{R})(c_i, d_c)\mathbf{M}(d) \right) \right] \circ \mathbf{R}(y, c_0)
\end{aligned} \tag{B.21}$$

Substituting (B.21) in (B.20) and performing certain algebraic manipulations, we get (B.18). We now present the non-adaptive version of the fast multipole method which is essentially a restatement of the theorem above.

B.5 The Non-Adaptive FMM

- **Step 1**. For each leaf b in the octree, calculate the multipole expansion coefficients $\mathbf{M}(b)$ of all particles contained in the box about the center of the box using Lemma(B.4.1). Since we essentially evaluate the multipole coefficients of all the N particles about different centers, this step costs $O(N)$.

- **Step 2**. For each level $l = l_{max} - 1, ...2$, for each box b at that level, calculate the multipole expansion coefficients $\mathbf{M}(b)$ due to all particles in that box by translating and aggregating the multipole expansion coefficients of all its children using Lemma(B.4.2). At each level, there are 8^l boxes. For each box, we make 8 translations. Total cost is therefore

$$\sum_{l=2}^{l_{max}} 8^l.8 = O(8^{l_{max}}) = O(N)$$

- **Step 3**. For each level $l = 2...l_{max}$, for each box b at that level, for each box $c \in I(b)$, convert and aggregate the multipole expansion coefficients $M(c)$ of that box into local expansion coefficients about b_c. Also translate and add the local expansion coefficients of $P(b)$. This procedure is simply the downward operator $\mathbf{D}_i$ of Theorem B.4.2. For each of the 8^l boxes at level l, we make a maximum of $\max(|I(b)|) = 189$ multipole to local translations and an additional local to local translation. Total cost of this step is therefore

$$\sum_{l=2}^{l_{max}} 8^l.(189 + 1) = O(8^{l_{max}}) = O(N)$$

- **Step 4**. For each leaf b in the octree, for each particle y in the box, evaluate the local expansion at the center of the box at that particle. Account for near neighbor interactions $(x \in N(b)$ directly by evaluations of ϕ. For each of the N particles at the last level, there is one local evaluation and $max(|N(b)|)s = 27s$ near neighbor evaluations giving a total cost of

$$N + 27s = O(N)$$

Each step in this algorithm is completed in $O(N)$ (given that all translations and evaluations are done in $O(1)$).

B.6 The Adaptive FMM

We had assumed at the beginning that the set of particles were distributed uniformly in space and therefore, all boxes at a given level contain the same number of particles. In many practical cases however, this is simply not true and heavily non-uniform distributions of particles may cause a lot of unnecessary computation. To efficiently account for non-uniform particle distributions, the fast multipole method was modified to the adaptive fast multipole method[7, 8].

The adaptive fast multipole method proceeds by subdividing only those boxes which have greater than s particles. With each box, four lists are associated:

- **Near neighbors**. This is same as before except for the fact that in the adaptive case, near neighbors may not be at the same level. Near neighbors at the same level are called colleagues.

- **Interaction list**. This is exactly as before: children of colleagues of the parent which are not colleagues of the box itself.

- **Local interaction list**. Leaves in the descendants of the near neighbors of the box which are not near neighbors of the box itself make up the local interaction list. These are boxes which are too close to interact via a multipole-local translation but the local expansion coefficients due to individual particles in these leaves can be evaluated and summed at the box.

- **Multipole interaction list**(with leaves only). This is the inverse of the local interactions list. These boxes are close enough to prevent their multipole expansion to being converted into a local expansion. However, there multipole expansion can be evaluated at each particle in the box.

The modifications to the algorithm are in

- **Step 3** In addition to converting the multipole expansion coefficients of all boxes in the interaction list into local expansion coefficients at the box center, the local expansion coefficients obtained from the individual particles contained in the local interaction list are aggregated.

- **Step 4** In addition to evaluating the near neighbors and local expansion coefficients at each particle, also evaluate the multipole expansion coefficients of all boxes in the multipole interaction list.

The efficiency of the adaptive FMM is hard to analyze and is a topic of ongoing research [19].

Appendix C

Properties of the Radiosity Kernel

The math appearing here has been solved by [23]. The application of the FMM to radiosity requires the radiosity kernel to hold certain properties. As seen in the previous chapter, the kernel must have a far-field and local expansion. In addition, the coefficients of expansion must have certain translation properties. In this chapter, we present the mathematical apparatus required for the successful application of FMM to radiosity.

C.1 Multipole Expansion

Consider the interaction of two surface points x and y. As we saw in Chapter 2, the radiosity at syrface point x due to surface point y is given by

$$B(x) - E(x) = \rho(x)\frac{\cos\theta_x \cos\theta_y}{\pi r^2} B(y) \tag{C.1}$$

Rewriting in vector form,

$$B(x) - E(x) = \rho(x)\frac{[\vec{n}_y.(\vec{r}_x - \vec{r}_y)][\vec{n}_x.(\vec{r}_y - \vec{r}_x)]}{\pi|\vec{r}_y - \vec{r}_x|^4} B(y) \tag{C.2}$$

For a surface point x, $\vec{r}_x$ and $\vec{n}_x$ denote the three dimensional coordinate and normal respectively.

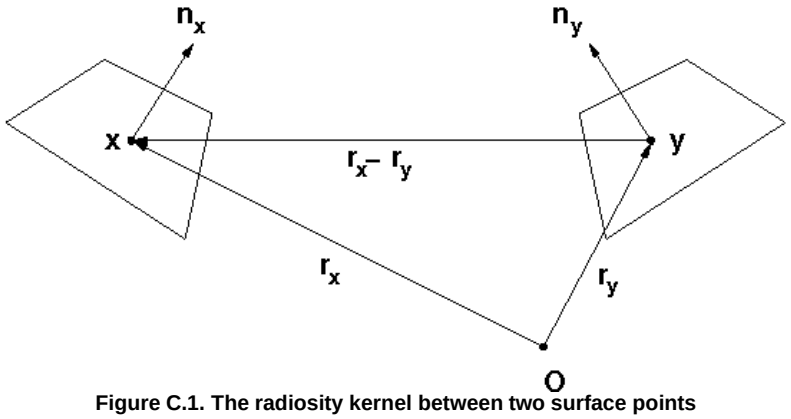


Figure C.1. The radiosity kernel between two surface points

Representing vectors as 3x1 matrices,

$$\vec{r} = (x, y, z) \equiv \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{r} \tag{C.3}$$

$$\vec{r}_1.\vec{r}_2 = \mathbf{r}_1^t\mathbf{r}_2 = \mathbf{r}_2^t\mathbf{r}_1 \tag{C.4}$$

we can expand the expression in the numerator as follows:

$$\begin{aligned}
[\vec{n}_y.(\vec{r}_x - \vec{r}_y)][\vec{n}_x.(\vec{r}_y - \vec{r}_x)] &= [\vec{r}_x.\vec{n}_y - \vec{r}_y.\vec{n}_y][\vec{r}_y.\vec{n}_x - \vec{r}_x.\vec{n}_x] \\
&= [\mathbf{r}_x^t\mathbf{n}_y - \mathbf{r}_y^t\mathbf{n}_y][\mathbf{r}_y^t\mathbf{n}_x - \mathbf{r}_x^t\mathbf{n}_x] \\
&= \mathbf{r}_x^t\mathbf{n}_y\mathbf{r}_y^t\mathbf{n}_x - \mathbf{r}_x^t\mathbf{n}_x\mathbf{r}_x^t\mathbf{n}_y - \mathbf{r}_y^t\mathbf{n}_y\mathbf{r}_y^t\mathbf{n}_x + \mathbf{r}_y^t\mathbf{n}_y\mathbf{r}_x^t\mathbf{n}_x
\end{aligned}$$

For the sake of notational convenience, we define the *receiver matrices* RM, the *source matrices* SM, and an operator $\otimes$ as:

$$\mathbf{SM}(y) = \begin{bmatrix} \mathbf{n}_y\mathbf{r}_y^t \\ \mathbf{n}_y \\ \mathbf{r}_y^t\mathbf{n}_y\mathbf{r}_y^t \\ \mathbf{r}_y^t\mathbf{n}_y \end{bmatrix} \qquad \mathbf{RM}(x) = \begin{bmatrix} \mathbf{r}_x^t \\ \mathbf{n}_x \\ \mathbf{r}_x^t\mathbf{n}_x\mathbf{r}_x^t \\ \mathbf{r}_x^t\mathbf{n}_x \\ \mathbf{n}_x\mathbf{r}_x^t \end{bmatrix}$$

$$\mathbf{RM}(x) \otimes \mathbf{SM}(y) = \mathbf{r}_x^t(\mathbf{n}_y\mathbf{r}_y^t)\mathbf{n}_x - \mathbf{r}_x^t\mathbf{n}_x\mathbf{r}_x^t(\mathbf{n}_y) - (\mathbf{r}_y^t\mathbf{n}_y\mathbf{r}_y^t)\mathbf{n}_x + (\mathbf{r}_y^t\mathbf{n}_y)\mathbf{r}_x^t\mathbf{n}_x \tag{C.5}$$

Notice that,

$$\sum_{y=y_1}^{y_k} \mathbf{RM}(x) \otimes \mathbf{SM}(y) = \mathbf{RM}(x) \otimes \sum_{y=y_1}^{y_k} \mathbf{SM}(y) \tag{C.6}$$

For $r_y < r_x$, from Hausner [20]

$$\frac{1}{|\vec{r}_y - \vec{r}_x|^4} = \sum_{n=0}^{\infty}\sum_{j=0}^{[n/2]}\sum_{m=-n+2j}^{n-2j} \pi e_n^j \left\{\frac{1}{r_x^{n+4}} Y_{n-2j}^m(\theta_x, \phi_x)\right\}\left\{r_y^n \overline{Y_{n-2j}^m(\theta_y, \phi_y)}\right\} \tag{C.7}$$

where Y_n^m are the $spherical harmonics$ and

$$e_n^j = 4\frac{(n-j+1)!(j+1/2)!}{(n-j+1/2)!j!}$$

Substituting (C.5), (C.6), and (C.7) in (C.2) and rearranging terms, we get

$$B(x) - E(x) = \sum_{n=0}^{\infty}\sum_{j=0}^{[n/2]}\sum_{m=-n+2j}^{n-2j} e_n^j R_{nj}^m(x) \otimes M_{nj}^m(O) \tag{C.8}$$

$$R_{nj}^m(x) = \frac{\rho(x)}{r_x^{n+4}} Y_{n-2j}^m(\theta_x, \phi_x)\mathbf{RM}(x) \tag{C.9}$$

$$M_{nj}^m(O) = B(y) r_y^n \overline{Y_{n-2j}^m(\theta_y, \phi_y)}\mathbf{SM}(y) \tag{C.10}$$

C.2 Local Expansion

Analogous to the multipole expansion is the local expansion. For $r_y > r_x$, from (C.7)

$$\frac{1}{|\vec{r}_y - \vec{r}_x|^4} = \sum_{n=0}^{\infty}\sum_{j=0}^{[n/2]}\sum_{m=-n+2j}^{n-2j} \pi e_n^j \left\{\frac{1}{r_y^{n+4}} Y_{n-2j}^m(\theta_y, \phi_y)\right\}\left\{r_x^n \overline{Y_{n-2j}^m(\theta_x, \phi_x)}\right\} \tag{C.11}$$

Subsequently, we get

$$B(x) - E(x) = \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} e_n^j E_{nj}^m(x) \otimes L_{nj}^m(O) \tag{C.12}$$

$$E_{nj}^m(x) = \rho(x) r_x^n \overline{Y_{n-2j}^m(\theta_x, \phi_x)} \mathbf{RM}(x) \tag{C.13}$$

$$L_{nj}^m(O) = B(y) \frac{1}{r_y^{n+4}} Y_{n-2j}^m(\theta_y, \phi_y) \mathbf{SM}(y) \tag{C.14}$$

C.3 Translation of Multipole Expansion

From Sack [33], if $(r, \theta, \phi) = (r_2, \theta_2, \phi_2) + (r_1, \theta_1, \phi_1)$ where $r_1 < r_2$, then

$$r^N Y_L^M(\theta, \phi) = \sum_{l_1=0}^{\infty} \sum_{m_1=-l_1}^{l_1} \sum_{l_2=l_{min}}^{l_{max}} \sum_{s=0}^{\infty} J(l_1, m_1, l_2, s, N, M, L) r_1^{l_1+2s} Y_{l_1}^{m_1}(\theta_1, \phi_1) r_2^{N-l_1-2s} Y_{l_2}^{M-m_1}(\theta_2, \phi_2) \tag{C.15}$$

where,

$$\begin{aligned}
J(l_1, m_1, l_2, s, N, M, L) &= 2\pi(-1)^{\lambda} (LM|l_1 m_1|l_2(M-m_1)) \frac{(\frac{-n}{2})_{\lambda} (\frac{n+3}{2})_L (\lambda - \frac{n}{2})_s (-\frac{n+1}{2} - \lambda_1)_s}{(\frac{1}{2})_{l_1+1} (\frac{n+3}{2})_{\lambda_1} (l_1 + \frac{3}{2})_s s!} \\
n &= N - L \\
\lambda &= \frac{-L + l_1 + l_2}{2} \\
\lambda_1 &= \frac{L - l_1 + l_2}{2} \\
(LM|l_1 m_1|l_2 m_2) &= (-1)^M \sqrt{\frac{(2L+1)(2l_1+1)(2l_2+1)}{4\pi}} \begin{pmatrix} L & l_1 & l_2 \\ -M & m_1 & m_2 \end{pmatrix} \begin{pmatrix} L & l_1 & l_2 \\ 0 & 0 & 0 \end{pmatrix}
\end{aligned}$$

$(\alpha)_n$ is the *pocchammer symbol* defined as

$$(\alpha)_n = \frac{\Gamma(\alpha + n)}{\Gamma(n)}$$

$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$ are the *Wigner* $3j$ symbols defined as

$$\left[\frac{(j_1 + j_2 - j_3)!(j_1 - j_2 + j_3)!(-j_1 + j_2 + j_3)!}{(j_1 + j_2 + j_3 + 1)!} \right]^{1/2}$$
$$[(j_1 + m_1)!(j_1 - m_1)!(j_2 + m_2)!(j_2 - m_2)!(j_3 + m_3)!(j_3 - m_3)!]^{1/2}$$
$$\sum_i \frac{(-1)^{i+j_1-j_2-m_3}}{i!(j_1 + j_2 - j_3 - i)!(j_1 - m_1 - i)!(j_2 + m_2 - i)!(j_3 - j_2 + m_1 + i)!(j_3 - j_1 - m_2 + i)!}$$

and the limits of l_2 are such that they satisfy

$$\begin{aligned}
|l_1 - l_2| \leq \quad & L \quad \leq l_1 + l_2 \\
L - l_1 - l_2 \quad & : \quad even
\end{aligned}$$

If we substitute $l_1 + 2s = k$ in (C.15) and collect powers of r_1^k, we get

$$r^N Y_L^M(\theta, \phi) = \sum_{k=0}^{\infty} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} \sum_{l_2=l_{min}}^{l_{max}} J(k-2s, m_1, l_2, s, N, M, L) r_1^k Y_{k-2s}^{m_1}(\theta_1, \phi_1) r_2^{N-k} Y_{l_2}^{M-m_1}(\theta_2, \phi_2) \tag{C.16}$$

where the summation over l_2 proceeds in steps of two and,

$$\begin{aligned} l_{max} &= L + k - 2s \\ l'_{min} &= max(|L - k + 2s|, |M - m_1|) \\ l_{min} &= \begin{cases} l'_{min} & L - k + 2s - l'_{min} \quad even \\ l'_{min} + 1 & L - k + 2s - l'_{min} \quad odd \end{cases} \end{aligned}$$

Now consider points r_1, r_2, O and O' where

$$\begin{aligned} |O'r_1| &< |O'r_2| \\ |Or_1| &< |Or_2| \quad and \\ |OO'| &< |O'r_2| \end{aligned}$$

From (C.7), we have

$$\frac{1}{|\vec{r}_1 - \vec{r}_2|^4} = \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} \pi e_n^j \left\{ |O'r_2|^{-n-4} Y_{n-2j}^m(O'r_2) \right\} \left\{ |O'r_1|^n \overline{Y_{n-2j}^m(O'r_1)} \right\} \quad (C.17)$$

$$= \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} \pi e_n^j \left\{ |Or_2|^{-n-4} Y_{n-2j}^m(Or_2) \right\} \left\{ |Or_1|^n \overline{Y_{n-2j}^m(Or_1)} \right\} \quad (C.18)$$

Since $Or_2 = O'r_2 + OO'$, substituting Or_2 in (C.18),

$$\begin{aligned} = & \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} \pi e_n^j \left\{ |Or_1|^n \overline{Y_{n-2j}^m(Or_1)} \right\} \times \\ & \left\{ \sum_{k=0}^{\infty} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} \sum_{l_2=l_{min}}^{l_{max}} J(k-2s, m_1, l_2, s, -n-4, m, n-2j) \times \right. \\ & \left. |OO'|^k Y_{k-2s}^{m_1}(OO') |O'r_2|^{-n-4-k} Y_{l_2}^{m-m_1}(O'r_2) \right\} \end{aligned}$$

now, if we set

$$\begin{aligned} l_2 &= n' - 2j' \\ n' &= n + k \\ m' &= m - m_1 \end{aligned}$$

and eliminate n, m and l_2, we get

$$\begin{aligned} = & \sum_{n'=0}^{\infty} \sum_{j'=0}^{[n'/2]} \sum_{m'=-n'+2j'}^{n'-2j'} \pi |O'r_2|^{-n'-4} Y_{n'-2j'}^{m'}(O'r_2) \times \\ & \sum_{k=0}^{n'} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} \sum_{j=j_{min}}^{j_{max}} e_{n'-k}^j J(k-2s, m_1, n'-2j', s, -n'+k-4, m'+m_1, n'-k-2j) \times \\ & |OO'|^k Y_{k-2s}^{m_1}(OO') |Or_1|^{n'-k} \overline{Y_{n'-k-2j}^{m_1+m'}(Or_1)} \end{aligned} \quad (C.19)$$

where,

$$\begin{aligned} j_{min} &= max(-k + s + j', 0) \\ j_{max} &= min(j' - s, n' - k + s - j') \end{aligned}$$

Finally, comparing (C.17) and (C.19), for *any* three points O, O' and r_1,

$$|O'r_1|^n\overline{Y^m_{n-2j}(O'r_1)} = \sum_{k=0}^{n}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{j'=j_{min}}^{j_{max}}\frac{e^j_{n-k}}{e^j_n}J(\ldots)|OO'|^kY^{m_1}_{k-2s}(OO')|Or_1|^{n-k}\overline{Y^{m_1+m}_{n-k-2j'}(Or_1)} \tag{C.20}$$

From (C.10),

$$M^m_{nj}(O') = \sum_{y=y_1}^{y_k} B(y)|O'y|^n\overline{Y^m_{n-2j}(O'y)}SM(O'y)$$

where

$$\begin{aligned}
\mathbf{SM}(O'y) &= \begin{bmatrix} \mathbf{n}_y(\mathbf{O'y})^t \\ \mathbf{n}_y \\ (\mathbf{O'y})^t\mathbf{n}_y(\mathbf{O'y})^t \\ (\mathbf{O'y})^t\mathbf{n}_y \end{bmatrix} \\
\text{Since}\quad (\mathbf{O'y})^t &= (\mathbf{Oy})^t - (\mathbf{OO'}) \\
\mathbf{SM}(O'y) &= \begin{bmatrix} \mathbf{n}_y(\mathbf{Oy})^t - \mathbf{n}_y(\mathbf{OO'})^t \\ \mathbf{n}_y \\ (\mathbf{Oy})^t\mathbf{n}_y(\mathbf{Oy})^t - (\mathbf{Oy})^t\mathbf{n}_y(\mathbf{OO'})^t - (\mathbf{OO'}^t\mathbf{n}_y(\mathbf{Oy})^t + \mathbf{n}^t_y(\mathbf{OO'})(\mathbf{OO'}^t \\ (\mathbf{Oy})^t\mathbf{n}_y - (\mathbf{OO'})^t\mathbf{n}_y \end{bmatrix}
\end{aligned}$$

Define *translation matrices* $\mathbf{TM}$ and denote

$$\begin{aligned}
\mathbf{TM}(OO') &= \begin{bmatrix} (\mathbf{OO'}) \\ (\mathbf{OO'})(\mathbf{OO'})^t \end{bmatrix} \\
\mathbf{SM}(O'y) &= \mathbf{TM}(OO') \otimes \mathbf{SM}(Oy)
\end{aligned} \tag{C.21}$$

Notice that,

$$\sum_{y=y_1}^{y_k}\mathbf{TM}(OO')\otimes\mathbf{SM}(Oy) = \mathbf{TM}(OO')\otimes\sum_{y=y_1}^{y_k}\mathbf{SM}(Oy) \tag{C.22}$$

From (C.10),(C.20), (C.21), and (C.22), we have

$$\begin{aligned}
M^m_{nj}(O') &= \sum_{y=y_1}^{y_k}B(y)\sum_{k=0}^{n}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{j'=j_{min}}^{j_{max}}\frac{e^j_{n-k}}{e^j_n}J(\ldots)\times \\
&\quad |OO'|^kY^{m_1}_{k-2s}(OO')|Oy|^{n-k}\overline{Y^{m_1+m}_{n-k-2j'}(Oy)}\mathbf{TM}(OO')\otimes\mathbf{SM}(Oy) \\
&= \sum_{k=0}^{n}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{j'=j_{min}}^{j_{max}}\frac{e^j_{n-k}}{e^j_n}J(\ldots)\times \\
&\quad |OO'|^kY^{m_1}_{k-2s}(OO')\mathbf{TM}(OO')\otimes \\
&\quad \left\{\sum_{y=y_1}^{y_k}B(y)|Oy|^{n-k}\overline{Y^{m_1+m}_{n-k-2j'}(Oy)}\mathbf{SM}(Oy)\right\} \\
&= \sum_{ksm_1j'}^{n}\frac{e^j_{n-k}}{e^j_n}J(\ldots)|OO'|^kY^{m_1}_{k-2s}(OO')\mathbf{TM}(OO')\otimes M^{m_1+m}_{n-k,j'}(O)
\end{aligned} \tag{C.23}$$

C.4 Multipole to Local Translation

Consider (C.9),

$$R_{nj}^{m}(Ox) = \rho(x)|Ox|^{-n-4}Y_{n-2j}^{m}(Ox)\mathbf{RM}(Ox)$$

$$\mathbf{RM}(Ox) = \begin{bmatrix} (\mathbf{Ox})^t \\ \mathbf{n}_x \\ (\mathbf{Ox})^t\mathbf{n}_x(\mathbf{Ox})^t \\ (\mathbf{Ox})^t\mathbf{n}_x \\ \mathbf{n}_x(\mathbf{Ox})^t \end{bmatrix}$$

Since $(\mathbf{Ox}) = (\mathbf{xox}) + (\mathbf{Oxo})$

$$\mathbf{RM}(Ox) = \begin{bmatrix} (\mathbf{xox})^t + (\mathbf{Oxo})^t \\ \mathbf{n}_x \\ (\mathbf{xox})^t\mathbf{n}_x(\mathbf{xox})^t + (\mathbf{xox})^t\mathbf{n}_x(\mathbf{Oxo})^t + (\mathbf{Oxo})^t\mathbf{n}_x(\mathbf{xox})^t + \mathbf{n}_x(\mathbf{Oxo})(\mathbf{Oxo})^t \\ (\mathbf{xox})^t\mathbf{n}_x + (\mathbf{Oxo})^t\mathbf{n}_x \\ \mathbf{n}_x(\mathbf{xox})^t + \mathbf{n}_x(\mathbf{Oxo})^t \end{bmatrix}$$

Denote using translation matrices $\mathbf{TM}$,

$$\mathbf{RM}(Ox) = \mathbf{RM}(x_0x) \otimes \mathbf{TM}(Ox_0) \tag{C.24}$$

From (C.15),

$$\begin{aligned} |Ox|^{-n-4}Y_{n-2j}^{m}(Ox) &= \sum_{k=0}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{l_2=l_{min}}^{l_{max}} J(k-2s, m_1, l_2, s, -n-4, m, n-2j) \times \\ &\quad |x_0x|^k Y_{k-2s}^{m_1}(x_0x)|Ox_0|^{-n-4-k}Y_{l_2}^{m-m_1}(Ox_0) \end{aligned} \tag{C.25}$$

Substituting (C.24) and (C.25) into (C.9),

$$\begin{aligned} R_{nj}^{m}(Ox) &= \rho(x)\sum_{k=0}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{l_2=l_{min}}^{l_{max}} J(\ldots)|x_0x|^k Y_{k-2s}^{m_1}(x_0x) \times \\ &\quad |Ox_0|^{-n-4-k}Y_{l_2}^{m-m_1}(Ox_0)\mathbf{RM}(x_0x) \otimes \mathbf{TM}(Ox_0) \\ &= \sum_{k=0}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{l_2=l_{min}}^{l_{max}} J(\ldots)E_{ksm_1}(x_0x) \otimes F_{ksm_1l_2}^{njm}(Ox_0) \end{aligned} \tag{C.26}$$

$$E_{ksm_1}(x_0x) = \rho(x)|x_0x|^k Y_{k-2s}^{m_1}(x_0x)\mathbf{RM}(x_0x) \tag{C.27}$$

$$F_{ksm_1l_2}^{njm}(Ox_0) = |Ox_0|^{-n-4-k}Y_{l_2}^{m-m_1}(Ox_0)\mathbf{TM}(Ox_0) \tag{C.28}$$

Substituting (C.26) in (C.8),

$$\begin{aligned} B(x) - E(x) &= \sum_{n=0}^{\infty}\sum_{j=0}^{[n/2]}\sum_{m=-n+2j}^{n-2j} e_n^j R_{nj}^{m}(\vec{Ox}) \otimes M_{nj}^{m}(O) \\ &= \sum_{n=0}^{\infty}\sum_{j=0}^{[n/2]}\sum_{m=-n+2j}^{n-2j} e_n^j \left\{ \sum_{k=0}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{l_2=l_{min}}^{l_{max}} J(\ldots)E_{ksm_1}(x_0x) \otimes F_{ksm_1l_2}^{njm}(Ox_0) \right\} \\ &\quad \otimes M_{nj}^{m}(O) \\ &= \sum_{k=0}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s} E_{ksm_1}(x_0x) \otimes L_{ksm_1}(Ox_0) \end{aligned} \tag{C.29}$$

$$L_{ksm_1}(Ox_0) = \sum_{n=0}^{\infty}\sum_{j=0}^{[n/2]}\sum_{m=-n+2j}^{n-2j}\sum_{l_2=l_{min}}^{l_{max}} e_n^j J(\ldots)F_{ksm_1l_2}^{njm}(Ox_0) \otimes M_{nj}^{m}(O) \tag{C.30}$$

C.5 Local Translation

Using the following properties of spherical harmonics,

$$\begin{aligned}\overline{Y_n^m} &= (-1)^m Y_n^{-m} \\ Y_n^m(-\vec{r}) &= (-1)^n Y_m^m(\vec{r})\end{aligned}$$

From (C.20), given $x_0x = x_0x_1 + x_1x$,

$$\begin{aligned}|x_0x|^k Y_{k-2s}^{m_1}(x_0x) &= \sum_{k'=0}^{k}\sum_{s'=0}^{k'/2}\sum_{m_1'=-k'+2s'}^{k'-2s'}\sum_{j'=j_{min}}^{j_{max}}(-1)^{k+m_1}\frac{e_{k-k'}^{j'}}{e_k^s}\times \\ &\quad J(k'-2s', m_1', k-2s, s', -k+k'-4, -m_1+m_1', k-k'-2j') \\ &\quad |x_1x|^{k'}Y_{k'-2s'}^{m_1'}(x_1x)|x_1x_0|^{k-k'}\overline{Y_{k-k'-2j'}^{-m_1'+m_1}(x_1x_0)} \qquad \text{(C.31)}\end{aligned}$$

From (C.24),

$$\mathbf{RM}(x_0x) = \mathbf{RM}(x_1x)\otimes\mathbf{TM}(x_0x_1) \qquad \text{(C.32)}$$

Substituting (C.31) and (C.32) in (C.27),

$$\begin{aligned}E_{ksm_1}(x_0x) &= \rho(x)\sum_{k'=0}^{k}\sum_{s'=0}^{k'/2}\sum_{m_1'=-k'+2s'}^{k'-2s'}\sum_{j'=j_{min}}^{j_{max}}(-1)^{k+m_1}\frac{e_{k-k'}^{j'}}{e_k^s}\times \\ &\quad J(k'-2s', m_1', k-2s, s', -k+k'-4, -m_1+m_1', k-k'-2j') \\ &\quad |x_1x|^{k'}Y_{k'-2s'}^{m_1'}(x_1x)|x_1x_0|^{k-k'}\overline{Y_{k-k'-2j'}^{-m_1'+m_1}(x_1x_0)} \\ &\quad \mathbf{RM}(x_1x)\otimes\mathbf{TM}(x_0x_1) \\ &= \sum_{k'=0}^{k}\sum_{s'=0}^{k'/2}\sum_{m_1'=-k'+2s'}^{k'-2s'}\sum_{j'=j_{min}}^{j_{max}}(-1)^{k+m_1}\frac{e_{k-k'}^{j'}}{e_k^s}J(\ldots) \\ &\quad E_{k's'm_1'}(x_1x)\otimes|x_1x_0|^{k-k'}\overline{Y_{k-k'-2j'}^{-m_1'+m_1}(x_1x_0)}\mathbf{TM}(x_0x_1) \qquad \text{(C.33)}\end{aligned}$$

Substituting (C.33) in (C.8),

$$\begin{aligned}B(x)-E(x) &= \sum_{k=0}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}E_{ksm_1}(x_0x)\otimes L_{ksm_1}(Ox_0) \\ &= \sum_{k=0}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\Bigg\{\sum_{k'=0}^{k}\sum_{s'=0}^{k'/2}\sum_{m_1'=-k'+2s'}^{k'-2s'}\sum_{j'=j_{min}}^{j_{max}}(-1)^{k+m_1}\frac{e_{k-k'}^{j'}}{e_k^s}J(\ldots) \\ &\quad E_{k's'm_1'}(x_1x)\otimes|x_1x_0|^{k-k'}\overline{Y_{k-k'-2j'}^{-m_1'+m_1}(x_1x_0)}\mathbf{TM}(x_0x_1)\Bigg\}\otimes L_{ksm_1}(Ox_0) \\ &= \sum_{k'=0}^{\infty}\sum_{s'=0}^{k'/2}\sum_{m_1'=-k'+2s'}^{k'-2s'}E_{k's'm_1'}(x_1x)\otimes L_{k's'm_1'}(x_0x_1) \\ L_{k's'm_1'}(x_0x_1) &= \sum_{k=k'}^{\infty}\sum_{s=0}^{k/2}\sum_{m_1=-k+2s}^{k-2s}\sum_{j'=j_{min}}^{j_{max}}(-1)^{k+m_1}\frac{e_{k-k'}^{j'}}{e_k^s}J(\ldots)|x_1x_0|^{k-k'}\overline{Y_{k-k'-2j'}^{-m_1'+m_1}(x_1x_0)} \\ &\quad \mathbf{TM}(x_0x_1)\otimes L_{ksm_1}(Ox_0) \qquad \text{(C.34)}\end{aligned}$$

Bibliography

[1] Anders Adamson and Marc Alexa. Ray tracing point set surfaces. In *Proceedings of Shape Modeling International 2003*, 2003.

[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[3] John Amanatides. Ray tracing with cones. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 129–135, 1984.

[4] R. Beatson and L. Greengard. A short course on fast multipole methods.

[5] J. A. Board, J. W. Causey, J. F. Leathrum, A. Windemuth, and K. Schulten. Accelerated molecular dynamics simulation with the parallel fast multipole method. *Chemistry Physics Letters*, 198:89–94, 1992.

[6] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *Proceedings of ACM SIGGRAPH*, pages 67–76, August 2001.

[7] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal of Scientific and Statistical Computing*, 9:669–686, July 1988.

[8] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155:468–498, 1999.

[9] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Radiosity for point sampled geometry. In *Pacific Graphics*, 2004.

[10] J. Dongarra and F. Sullivan. The top ten algorithms. *Computing in Science and Engineering*, 2:22–23, 2000.

[11] Philip Dutre, Parag Tole, and Donald P. Greenberg. Approximate visibility for illumination computation using point clouds. Technical report, Cornell University, 2000.

[12] A. Elgammal, R. Duraiswami, and L. Davis. Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. *IEEE Transactions on PAMI*, 2003.

[13] Rhushabh Goradia, Anish Chandak, Biswarup Choudary, and Sharat Chandran. Fmm-based illumination maps for point models. In *Was submitted to Symposium on Point Based Graphics (pbg06)*, 2006 (Not Accepted).

[14] C. M. Goral, K. E. Torrance, and D. P. Greenberg. Modeling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(3):213–222, July 1984.

[15] L. Greengard. *The rapid evaluation of potential fields in particle systems*. MIT Press, Cambridge, Massachusetts, 1988.

[16] L. Greengard, M. C. Kropinski, and A. Mayo. Integral equation methods for stokes flow and isotropic elasticity. *Journal of Computational Physics*, 125:403–414, 1996.

[17] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[18] J. P. Grossman and William J. Dally. Point sample rendering. In *Rendering Techniques*, pages 181–192, 1998.

[19] N. A. Gumerov, R. Duraiswami, and E. A. Borikov. Data structures, optimal choice of parameters, and complexity results for generalized multilevel fast multipole methods in d dimensions. Technical report, Perceptual Interfaces and Reality Laboratory, Institute for Advanced Computer Studies, University of Maryland, College Park, 2003.

[20] A. Haunsner. Multipole expansion of the light vector. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):12–22, Jan-Mar 1997.

[21] H. W. Jensen. Global illumination using photon maps. *Eurographics Rendering Workshop 1996*, pages 21–30, June 1996.

[22] James T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150. ACM Press, 1986.

[23] A. Karapurkar and S. Chandran. Removed for purposes of review. Master's thesis, Indian Institute of Technology, Bombay, 2003.

[24] A. Karapurkar, N. Goel, and S. Chandran. Removed for purposes of review. *Indian Conference on Computer Vision, Graphics, and Image Processing*, pages 119–125, 2004.

[25] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[26] Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical report, University of North Carolina at Chapel Hill, 1985.

[27] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.

[28] Mark Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, ETH Zurich, 2003.

[29] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society.

[30] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[31] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 335–342. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[32] Szymon Rusinkiewicz and Marc Levoy. QSplat: A multiresolution point rendering system for large meshes. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 343–352. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[33] R. A. Sack. Addition theorems for functions of spherical harmonics. *Journal of Mathematical Physics*, 5(2):245–251, Feb 1964.

[34] G. Schaufler and H. Jensen. Ray tracing point sampled geometry. In *Eurographics Rendering Workshop Proceedings*, pages 319–328, 2000.

[35] F. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, 1994.

[36] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics*, 25(4):61–68, 1991.

[37] Ingo Wald. High-Quality Global Illumination Walkthroughs using Discretized Incident Radiance Maps. *Technical Report, SCI Institute, University of Utah, No UUSCI-2005-010 (submitted for publication)*, 2005.

[38] Ingo Wald and Hans-Peter Seidel. Interactive Ray Tracing of Point Based Models. In *Proceedings of 2005 Symposium on Point Based Graphics*, 2005.

[39] M. Wand and W. Strasser. Multi-resolution point sample ray-tracing. In *Graphics Interface Conference Proceedings*, 2003.

[40] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3d: An interactive system for point-based surface editing, 2002.

[41] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, New York, NY, USA, 2001. ACM Press.