FMM-based Illumination Maps For Point Models

Rhushabh Goradia
Guide: Prof. Sharat Chandran

Department of Computer Science and Engineering
IIT Bombay
Annual Progress Seminar 2006

August 23, 2006
Outline

1. Introduction
   - Problem Definition

2. Fast Computation of Radiosity with FMM

3. Visibility in Point Models
   - Point-Point Visibility
   - Incorporating visibility in the FMM Hierarchy

4. Point Based Rendering

5. Results

6. Summary

7. Conclusions and Futurework
Outline

1 Introduction
   - Problem Definition

2 Fast Computation of Radiosity with FMM

3 Visibility in Point Models
   - Point-Point Visibility
   - Incorporating visibility in the FMM Hierarchy

4 Point Based Rendering

5 Results

6 Summary

7 Conclusions and Futurework
## Problem Definition

### Problem Statement
To compute **illumination maps** for complex scenes represented as **point-models**

### Some keywords to look out for:
- Point Models - Modelling and Rendering
- Global Illumination and Illumination Maps
- Fast Multipole Method
- Point-Point Visibility
Problem Definition

Problem Statement

To compute illumination maps for complex scenes represented as point-models

Some keywords to look out for:

- Point Models - Modelling and Rendering
- Global Illumination and Illumination Maps
- Fast Multipole Method
- Point-Point Visibility
Model each point as a surface sample representation [4, levoy85]
Each point has [co-ordinates, normal, reflectance, emissivity] values
Why Point Models?

Problems with traditional polygonal rendering

- lack of output sensitivity
- more storage consumption
- difficulty in consistent mesh construction from points (given by 3D scanners)
- complex multiresolution methods

10^6 triangles   10^7 triangles   10^8 triangles
What are Global Illumination Algorithms?

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).

Examples of Different GI Algorithms

- Radiosity
- Ray Tracing
- Photon Mapping

Contributes to effects like **Soft Shadows** and **Color Bleeding**
Global Illumination/ Inter-reflections

Global Illumination effects

Some example pictures showing Global Illumination
Global Illumination/ Inter-reflections

Radiosity as a GI method

Rendering Equation for Radiosity between two points

\[ B(x) = E(x) + \rho(x) \int_{A_y} \frac{[\vec{n}_y \cdot (\vec{r}_x - \vec{r}_y)] [\vec{n}_x \cdot (\vec{r}_y - \vec{r}_x)]}{\pi |\vec{r}_y - \vec{r}_x|^4} B(y) dA_y \]

- **Illumination Maps** (IM) are color values at every point in the model, due to application of Radiosity as the GI algorithm. These IM can be used for further processing.
Fast Multipole Method (FMM)

The Fast Multipole Method

[1]rokhlin is concerned with evaluating the effect of a “set of sources” $Y$, on a set of “evaluation points” $X$.

More formally, given

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

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

we wish to evaluate the sum

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

- Total complexity: $O(NM)$
The Fast Multipole Method

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

- The FMM attempts to reduce this seemingly irreducible complexity to \( O(N + M) \).
- The two main insights that make this possible are
  - **Factorization** of the kernel into source and receiver terms
  - Many application domains do not require that the function \( f \) be calculated at very high accuracy.
- FMM follows a **hierarchical structure** (*Octree*)
- Each node has an associated **Interaction Lists**
Visibility between points

Visibility Between Point Pairs

Visibility calculation between point pairs is **essential** as a point receives energy from other point only if it is **visible**.
Visibility between points

Visibility Between Point Pairs

Its complicated in our case !! Why ?

- 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 and 1.
Visibility between points

Visibility Between Point Pairs

It's complicated in our case!! Why?

- 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 and 1.
Visibility Between Point Pairs

It's complicated in our case!! Why?

- 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 and 1.
Visibility between points

Visibility Between Point Pairs

**Its complicated in our case !! Why ?**

- 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 and 1.
Visibility between points

Visibility Between Point Pairs

It's complicated in our case!! Why?

- 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 and 1.
Problem Statement Revisited

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.

Three keythings 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
Problem Statement Revisited

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.

Three key 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
Problem Statement Revisited

Application Domains
Outline

1. Introduction
   - Problem Definition

2. Fast Computation of Radiosity with FMM

3. Visibility in Point Models
   - Point-Point Visibility
   - Incorporating visibility in the FMM Hierarchy

4. Point Based Rendering

5. Results

6. Summary

7. Conclusions and Futurework
The radiosity equation revisited

Rendering Equation for Radiosity

\[ B(x) = E(x) + \rho(x) \int_{A_y} \frac{[\hat{n}_y.(\vec{r}_x - \vec{r}_y)][\hat{n}_x.(\vec{r}_y - \vec{r}_x)]}{\pi|\vec{r}_y - \vec{r}_x|^4} B(y) dA_y \]
Assigning Weights to Points

- The rendering equation assumes that we integrate over a surface area.
- In earlier work, triangles were assumed as input and Gaussian quadrature points were used 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.
- We thus replace the interaction between surfaces and points in rendering equation as between points only. This interaction is termed as a particle interaction.
Assigning Weights to Points

- The rendering equation assumes that we integrate over a surface area.
- In earlier work, triangles were assumed as input and Gaussian quadrature points were used 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.
- We thus replace the interaction between surfaces and points in rendering equation as between points only. This interaction is termed as a particle interaction.
Assigning Weights to Points

- The rendering equation assumes that we integrate over a surface area.
- In earlier work, triangles were assumed as input and Gaussian quadrature points were used 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.
  - We thus replace the interaction between surfaces and points in rendering equation as between points only. This interaction is termed as a particle interaction.
Assigning Weights to Points

- The rendering equation assumes that we integrate over a surface area.
- In earlier work, triangles were assumed as input and Gaussian quadrature points were used 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.
  - We thus replace the interaction between surfaces and points in the rendering equation as between points only. This interaction is termed as a particle interaction.
Assigning Weights to Points

- The rendering equation assumes that we integrate over a surface area.
- In earlier work, triangles were assumed as input and Gaussian quadrature points were used 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.
- We thus replace the interaction between surfaces and points in rendering equation as between points only. This interaction is termed as a particle interaction.
Assigning Weights to Points

- The rendering equation assumes that we integrate over a surface area.
- In earlier work, triangles were assumed as input and Gaussian quadrature points were used 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.

We thus replace the interaction between surfaces and points in the rendering equation as between points only. This interaction is termed as a particle interaction.
Assigning Weights to Points

- The rendering equation assumes that we integrate over a surface area.
- In earlier work, triangles were assumed as input and Gaussian quadrature points were used 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.
- We thus replace the interaction between surfaces and points in rendering equation as between points only. This interaction is termed as a **particle interaction**.
Outline

1. Introduction
   - Problem Definition

2. Fast Computation of Radiosity with FMM

3. **Visibility in Point Models**
   - Point-Point Visibility
   - Incorporating visibility in the FMM Hierarchy

4. Point Based Rendering

5. Results

6. Summary

7. Conclusions and Futurework
Visibility in Point Models

Two main parts of visibility algorithm

- Point-Point visibility
- Incorporating visibility in the FMM context
Point-Point Visibility

- Only \( x_2 \) and \( x_4 \) will be considered as occluders.
- Reject \( x_1 \) as the intersection point of the tangent plane lies outside the line segment \( \overline{pq} \).
- \( x_3 \) is rejected because it is more than a distance \( \Delta \) from the line segment \( \overline{pq} \).
Point-Point Visibility

Procedure point_visible(Point $p$, Point $q$)
Declare threshold $t_1$, $visible_{p,q} = 1$
if FacingEachOther($p, q$) then
    Find $k$ closest points in region $\Delta$ around $\overline{pq}$
    Prune based on the tangent plane
    for $i = 0$ to $2$ do
        $contributeVis_i = visibility\_look\_up(distance_i)$
        $visible_{p,q} = visible_{p,q} \ast contributeVis_i$
    end for
    if ($visible_{p,q}$) > $t_1$ then
        return($visible$)
    end if
end if
Recall that the object space composed of points was divided into an adaptive octree.

Note that each node receives energy from its interaction list.

The key idea is to modify the interaction list.

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 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 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.
Incorporating visibility in the FMM Hierarchy

Hierarchical Visibility – Physical Significance

- Recall that the object space composed of points was divided into an adaptive octree.
- Note that each node receives energy from its interaction list.
  - The key idea is to modify the interaction list.
  - 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 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 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.
Incorporating visibility in the FMM Hierarchy

Hierarchical Visibility – Physical Significance

- Recall that the object space composed of points was divided into an adaptive octree.
- Note that each node receives energy from its interaction list.
- The key idea is to **modify the interaction list**.
  - 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 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 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.
Recall that the object space composed of points was divided into an adaptive octree.

Note that each node receives energy from its interaction list.

The key idea is to **modify the interaction list**.

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 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 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.
Incorporating visibility in the FMM Hierarchy

Hierarchical Visibility – Physical Significance

- Recall that the object space composed of points was divided into an adaptive octree
- Note that each node receives energy from its interaction list
- The key idea is to modify the interaction list
- 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 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 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
Incorporating visibility in the FMM Hierarchy

Hierarchical Visibility – Physical Significance

- Recall that the object space composed of points was divided into an adaptive octree
- Note that each node receives energy from its interaction list
- The key idea is to modify the interaction list
- 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 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 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
Incorporating visibility in the FMM Hierarchy

Hierarchical Visibility – Physical Significance

- Recall that the object space composed of points was divided into an adaptive octree.
- Note that each node receives energy from its interaction list.
- The key idea is to modify the interaction list.
- 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 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 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.
Incorporating visibility in the FMM Hierarchy

Point-Leaf Visibility

Procedure pointLeafVisibility(Point p, Leaf L)
Declare threshold $t_2$, Visi_point_L = 0
for each point $p_i \in L$ 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)
Incorporating visibility in the FMM Hierarchy

Leaf-Leaf Visibility

Procedure Leaf_Leaf_visibility(Leaf L, Leaf C)
Declare threshold $t_3$, Visi_leaf_L = 0
for each point $p_i \in C$ do
    state = point_cell_visible($p_a$, Leaf L)
    if equals(state, visible) then
        Visi_leaf_L = Visi_leaf_L + 1
    end if
end for
if Visi_leaf_L > threshold $t_3$ then
    return(visible)
end if
return(invisible)
Incorporating visibility in the FMM Hierarchy

Node-Node Visibility

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

In case of partial visibility, 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
Computing Interaction Lists

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_Leg_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
    interactionlist(A).remove(B)
  else if equals(state,invalid) then
    interactionlist(A).remove(B)
  end if
end for
end for
for each R ∈ child(A) do
  Octree_Visibility(R)
end for
Point Based Rendering

Challenge of PBR
To achieve a continuous interpolation between discrete point samples that are irregularly distributed on a surface.

Algorithms available

- QSplat
- Surface splatting
- ... 

Changes we made ...
- Took care of compatibility of our output file formats and renderer’s input file formats
- Separating and removing inbuilt lightening calculations of the renderer
Point Based Rendering

Challenge of PBR
To achieve a continuous interpolation between discrete point samples that are irregularly distributed on a surface.

Algorithms available
- QSplat
- Surface splatting
- ...

Changes we made ...
- Took care of compatibility of our output file formats and renderer’s input file formats
- Separating and removing inbuilt lightening calculations of the renderer
Introduction

Fast Computation of Radiosity with FMM

Visibility in Point Models

Point Based Rendering

Results

Summary

Point Based Rendering

**Challenge of PBR**

To achieve a continuous interpolation between discrete point samples that are irregularly distributed on a surface.

**Algorithms available**

- QSplat
- Surface splatting
- ...

- Changes we made ...
  - Took care of compatibility of our output file formats and renderer’s input file formats
  - Separating and removing inbuilt lightening calculations of the renderer
Introduction

Fast Computation of Radiosity with FMM

Visibility in Point Models

Point Based Rendering

Results

Summary

Conclusions and Futurework
Surfel Rendered Output
Comparision with Radiosity Solution for Triangulated Model
Correctness of Visibility
Correctness of Visibility
Outline

1. Introduction
   - Problem Definition

2. Fast Computation of Radiosity with FMM

3. Visibility in Point Models
   - Point-Point Visibility
   - Incorporating visibility in the FMM Hierarchy

4. Point Based Rendering

5. Results

6. Summary

7. Conclusions and Futurework
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.
Conclusion

- The FMM method is elegant because it trades off error with quality in a disciplined quantitative way.
- The kernel of the energy balance in the rendering equation has been made conformant to the FMM.
- We have also given a new visibility algorithm for point-based models.
- The visibility algorithm can be viewed as a ‘preprocessing’ step for photo-realistic global illumination of complex point-based models.
Futurework

- Take measures to remove the artifacts from the rendered image
- To come up with a good rendering system
- Optimizing the visibility code in terms of speed, retaining/improving the quality at same time
- Extending visibility algorithm to fit a parallel architecture framework
- Parallelizing FMM
- Take up some related problems to point based modelling like *Point Model Segmentation*
- 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
L. Greengard and V. Rokhlin.
A fast algorithm for particle simulations.

A. Haunsner.
Multipole expansion of the light vector.

The digital michelangelo project: 3D scanning of large statues.


Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross.
Surface splatting.

THANK YOU !!!
Questions ????
The Fast Multipole Method

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

- The function \( \phi \) 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 \( \phi \) 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 \( O(NM) \).
FMM Overview

Four Key-stones of FMM

- Factorize
- Error
- Grouping
- Translation
FMM Overview

Middleman Algorithm

Standard algorithm

Total number of operations: $O(NM)$

Middleman algorithm

Total number of operations: $O(N+M)$
FMM Overview

Two Level FMM

Total number of operations: $O(N+M+KL)$

Total number of operations: $O(N+M+K+L+K_1L_1)$

© Gumerov & Duraiswami
FMM Overview

Source Data Hierarchy

Evaluation Data Hierarchy

MLFMM

© Gamess & Duraiswami

Level 2

Level 3

Level 4

Level 5

N

M
How the spatial subdivision is done?

We use octree to subdivide the 3D space into a group of nodes.
Representing vectors as 3x1 matrices,

\[ \vec{r} = (x, y, z) \equiv \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \vec{r} \]

\[ \vec{r}_1 \cdot \vec{r}_2 = r_{1t}^t r_{2t} = r_{2t}^t r_{1t} \]

we can expand the expression in the numerator as follows:

\[ [\vec{n}_y \cdot (\vec{r}_x - \vec{r}_y)][\vec{n}_x \cdot (\vec{r}_y - \vec{r}_x)] = r_{xt}^t \vec{n}_y r_{yt}^t \vec{n}_x - r_{xt}^t \vec{n}_x r_{xt}^t \vec{n}_y - r_{yt}^t \vec{n}_y r_{yt}^t \vec{n}_x + r_{yt}^t \vec{n}_y r_{xt}^t \vec{n}_x \]
The Spatial Sub-division

**Factorization of the Numerator**

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

\[
RM(x) = \begin{bmatrix}
    r^t_x \\
    n_x \\
    r^t_x n_x r^t_y \\
    n_y r^t_y
\end{bmatrix}
\]

\[
SM(y) = \begin{bmatrix}
    n_y r^t_y \\
    n_y \\
    r^t_y n_y r^t \\
    r^t_y n_y
\end{bmatrix}
\]

\[
RM(x) \otimes SM(y) = \bar{r}^t_x (\bar{n}_y \bar{r}^t_y) \bar{n}_x - \bar{r}^t_x \bar{n}_x \bar{r}^t_x (\bar{n}_y) - (\bar{r}^t_y \bar{n}_y \bar{r}^t_y) \bar{n}_x + (\bar{r}^t_y \bar{n}_y) \bar{r}^t_x \bar{n}_x
\]

Notice that,

\[
\sum_{y=y_1}^{y_k} RM(x) \otimes SM(y) = RM(x) \otimes \sum_{y=y_1}^{y_k} SM(y)
\]
The Spatial Sub-division

**Factorization of the Denominator**

If we denote the spherical coordinates of $\vec{r}_x$ by $(r_x, \theta_x, \phi_x)$, then we make use of [2]hausner to write (for $r_y < r_x$),

$$\frac{1}{|\vec{r}_y - \vec{r}_x|^4} = \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} \pi e^j_n \left\{ \frac{1}{r_{x}^{n+4}} Y_{n-2j}^m(\theta_x, \phi_x) \right\} \left\{ r_{y}^n Y_{n-2j}^m(\theta_y, \phi_y) \right\}$$

where

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

and $Y_{n}^m$ are the normalized spherical harmonics
The Multipole Expansion

- Substituting the above factorizations in the Radiosity rendering equation and rearranging terms, we get the *multipole expansion* as

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

\[
R_{nj}^m(x) = \frac{\rho(x)}{r_x^{n+4}} Y_{n-2j}^m(\theta_x, \phi_x) R M(x)
\]

\[
M_{nj}^m(A_y) = \int_{A_y} r_y^n Y_{n-2j}^m(\theta_y, \phi_y) S M(y) dA_y
\]

- For practical implementation, the summation to infinity is truncated to two terms, and the error incurred is very small.
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.
Since the FMM algorithm is hierarchical, we need a way to collect irradiance, as shown below.

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 and then “translated” to get the cumulative effect of the entire set of clusters.
The equation

\[
B(x) = \sum_{n=0}^{\infty} \sum_{j=0}^{[\frac{n}{2}]} \sum_{m=-n+2j}^{n-2j} e^{jn} R_{nj}(x) \otimes M_{nj}(Ay)
\]

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 local expansions in terms of the coefficients \( L_{nj}^m \).
The Local Expansion - Physical Significance

- The irradiance stored at a virtual point O in the form of a constant number of coefficients can be disseminated to different receivers.
- This is valid only if the receiver points are “close by”
Similar to the multipole coefficients, the *local coefficients* $L_{nj}^m$ are also additive, and can be translated to a different coordinate system.

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.
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.
The FMM Algorithm

- Arrange points in the input model in an octree
  - With each node, associate two sets of disjoint nodes
    - Near neighbors
    - Interaction list

- Upward Pass
  - Calculate multipole coefficients for the leaf cells
  - Starting from the penultimate level, for each level, calculate the multipole coefficients of each node at that level by translating and accumulating the multipole coefficients of its children
The Spatial Sub-division

The FMM Algorithm

- Arrange points in the input model in an octree
- With each node, associate two set of disjoint nodes
  - Near neighbors
  - Interaction list

Upward Pass

- Calculate Multipole co-efficients for the leaf cells
- Starting from the penultimate level, for each level, calculate the multipole coefficients of each node at that level by translating and accumulating the multipole coefficients of its children
The Spatial Sub-division

The FMM Algorithm

- Arrange points in the input model in an octree
- With each node, associate two set of disjoint nodes
  - Near neighbors
  - Interaction list
- Upward Pass
  - Calculate Multipole co-efficients for the leaf cells
  - Starting from the penultimate level, for each level, calculate the multipole coefficients of each node at that level by translating and accumulating the multipole coefficients of its children
The Spatial Sub-division

The FMM Algorithm continued ...

- **Downward Pass**
  - For each level (starting from the second), calculate local coefficients at each node $b$ 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.
  - Additionally, the local expansion coefficients obtained from the parents are aggregated.

- **Final Summation**
  - 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 $x$.
  - Add directly the effect of all points $x \in$ the near neighbors of $b$.

- Iterate over these steps till sufficient convergence is reached.
The FMM Algorithm continued ...

- **Downward Pass**
  - For each level (starting from the second), calculate local coefficients at each node $b$ 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
  - Additionally, the local expansion coefficients obtained from the parents are aggregated.

- **Final Summation**
  - 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 $x$
  - Add directly the effect of all points $x \in$ the near neighbors of $b$

- Iterate over these steps till sufficient convergence is reached
The Spatial Sub-division

The FMM Algorithm continued ...

- **Downward Pass**
  - For each level (starting from the second), calculate local coefficients at each node $b$ 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.
  - Additionally, the local expansion coefficients obtained from the parents are aggregated.

- **Final Summation**
  - 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 $x$.
  - Add directly the effect of all points $x \in$ the near neighbors of $b$.

- Iterate over these steps till sufficient convergence is reached.
Surfel Rendered Output
The Spatial Sub-division

Surfel Rendered Output
## The Spatial Sub-division

<table>
<thead>
<tr>
<th>p-Status</th>
<th>Lists-Status</th>
<th>Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>NO Change</td>
<td>1506</td>
</tr>
<tr>
<td>3</td>
<td>NO Change</td>
<td>1610</td>
</tr>
<tr>
<td>2</td>
<td>Changed</td>
<td>1886</td>
</tr>
<tr>
<td>3</td>
<td>Changed</td>
<td>2123</td>
</tr>
</tbody>
</table>