# The Fast Multipole Method and the Radiosity Kernel

**Sharat Chandran**
http://www.cse.iitb.ac.in/∼sharat

January 8, 2006

(Joint work with Alap Karapurkar and Nitin Goel)

# Overview

- What's FMM about?

# Overview

- What's FMM about?
- Point-based rendering

# Overview

- What's FMM about?
- Point-based rendering
- Insight into the method

# Overview

- What's FMM about?
- Point-based rendering
- Insight into the method
- The algorithm and our contributions

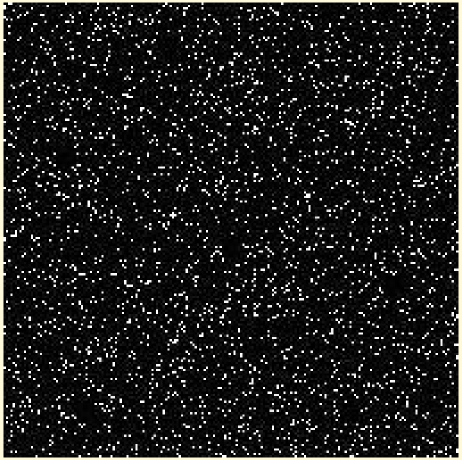
# Overview

- What's FMM about?
- Point-based rendering
- Insight into the method
- The algorithm and our contributions
- Limitations, future work and conclusion

Home Page
Title Page
Page 3 of 43
Go Back
Full Screen
Close
Quit

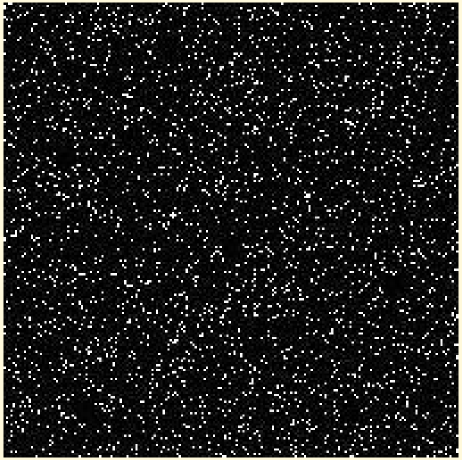
# 1. The Fast Multipole Method

- Speeds up matrix-vector product sums of certain types
  - Naive method is quadratic: $O(n^2)$
  - FMM solves the problem in linear time: $O(n)$



# 1. The Fast Multipole Method

- Speeds up matrix-vector product sums of certain types
  - Naive method is quadratic: $O(n^2)$
  - FMM solves the problem in linear time: $O(n)$



- Tricks of the trade: Math, Precision, Data Structures

The Fast Multipole Method
A Faster Solution: This . . .
Step 1: Multipole . . .
Interaction List
Step 2: Translation of . . .
Step 3: Local Expansion
Step 4: Translation of . . .
The Overall Algorithm

Home Page

Title Page

Page 3 of 43

Go Back

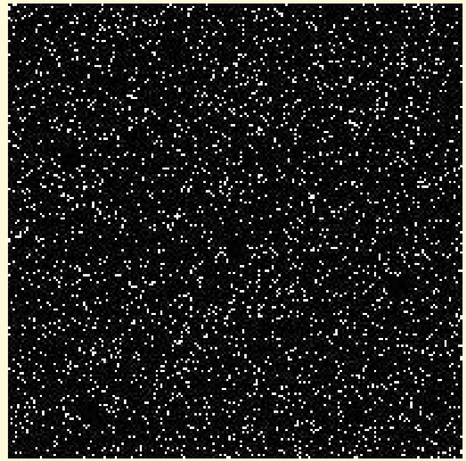
Full Screen

Close

Quit

# 1. The Fast Multipole Method

- Speeds up matrix-vector product sums of certain types
  - Naive method is quadratic: $O(n^2)$
  - FMM solves the problem in linear time: $O(n)$



- Tricks of the trade: Math, Precision, Data Structures

- Sometimes can get even better results
  - Move from $O(n^3)$ to $O(kn\log n)$ or even $O(kn)$
  - Solve “impossible” problems :-)

# Point-Based Rendering

- Performance of graphics card has increased tremendously
- Points are available as input data
- Want to go beyound flat polygons

# Point-Based Rendering

- Performance of graphics card has increased tremendously
- Points are available as input data
- Want to go beyound flat polygons

- Rendering options
  - Local illumination models . . . (surfels?)
  - Ray-tracing and photon mapping
  - Diffuse global illumination (this talk)

The Fast Multipole Method
A Faster Solution: This . . .
Step 1: Multipole . . .
Interaction List
Step 2: Translation of . . .
Step 3: Local Expansion
Step 4: Translation of . . .
The Overall Algorithm

Home Page

Title Page



Go Back

Full Screen

Close

Quit

# Point-Based Rendering

- Performance of graphics card has increased tremendously
- Points are available as input data
- Want to go beyound flat polygons

- Rendering options
  - Local illumination models . . . (surfels?)
  - Ray-tracing and photon mapping
  - Diffuse global illumination (this talk)
- Method *works* for traditional patch models too

## So what's the Problem?

- Lots of points make the problem "intractable."
  - 1GB RAM = $2^{10}2^{10}2^{10}$ bytes
  - $n^2 = 2^{30} \rightarrow n = 32K$
  - In reality, we may desire a larger number
- Space is a constraint

## So what's the Problem?

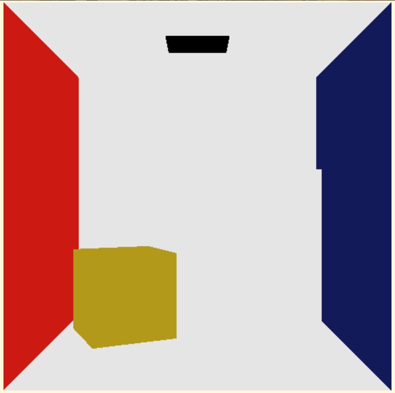
- Lots of points make the problem "intractable."
  - 1GB RAM = $2^{10}2^{10}2^{10}$ bytes
  - $n^2 = 2^{30} \rightarrow n = 32K$
  - In reality, we may desire a larger number
- Space is a constraint
- If solution involves inverting a matrix $O(n^3)$ time, that's *also* a constraint
- What if we could solve the problem in $O(n)$ space and $O(n)$ time?

# Illumination Models Are Key to Rendering

- Ray tracing produces stunning pictures but
  - Pictures appear “too good.”
  - Takes too long to produce reasonable images

# Illumination Models Are Key to Rendering

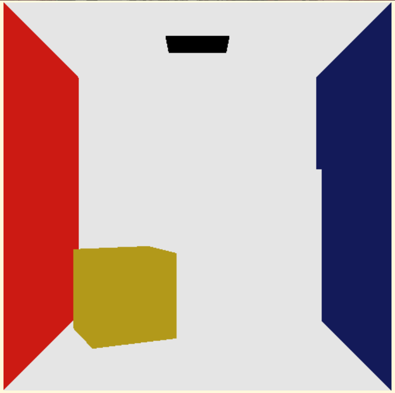
- Ray tracing produces stunning pictures but
  - Pictures appear “too good.”
  - Takes too long to produce reasonable images
- Flat shading is quick but unreal



# Illumination Models Are Key to Rendering

- Ray tracing produces stunning pictures but
  - Pictures appear "too good."
  - Takes too long to produce reasonable images


- Flat shading is quick but unreal

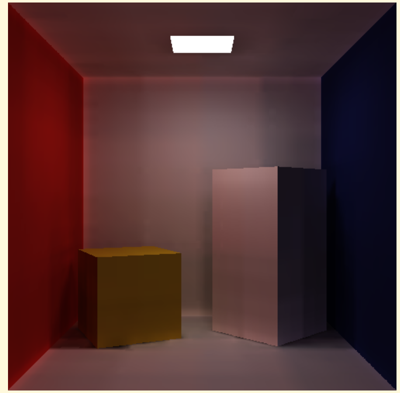


- Flat shading can be remarkably improved by discretizing environment into patches and computing intensity for each patch.

Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 7 of 43
Go Back
Full Screen
Close
Quit

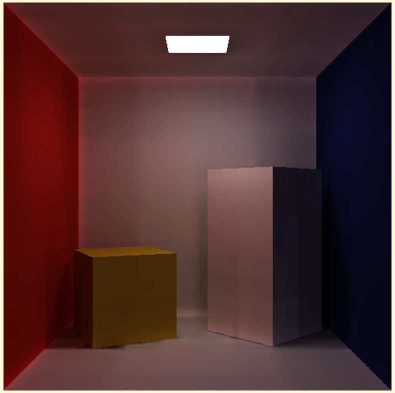
# Global Illumination In a Hurry

- Radiosity produces photorealistic pictures that can handle non specular scenes
  - Enables color bleeding effects
  - View independent representation



- Starting point is the general energy balance equation for the radiance
$$L(x,\theta_0,\phi_0) = L_e(x,\theta_0,\phi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\phi_0,\theta,\phi)L_i(x,\theta,\phi)\cos\theta\, d\omega$$

# Global Illumination In a Hurry



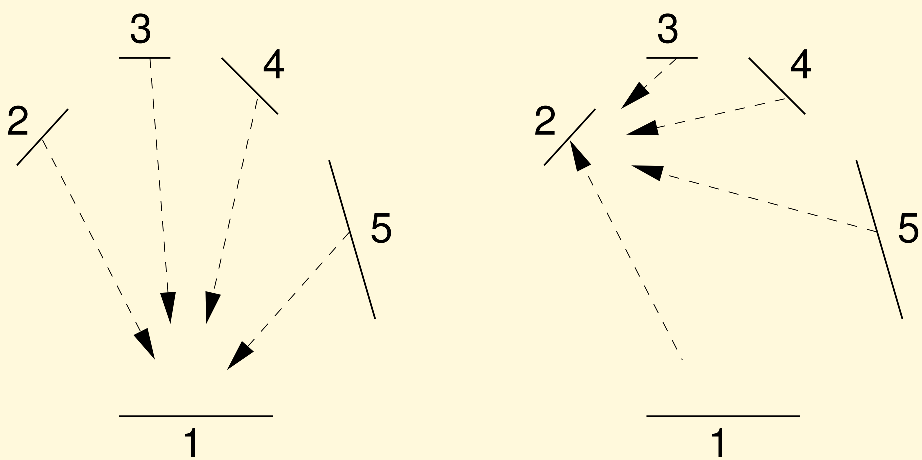
- Radiosity produces photorealistic pictures that can handle non specular scenes
  - Enables color bleeding effects
  - View independent representation
- Starting point is the general energy balance equation for the radiance
  $L(x,\theta_0,\phi_0) = L_e(x,\theta_0,\phi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\phi_0,\theta,\phi) L_i(x,\theta,\phi)\cos\theta\, d\omega$
- Point-based rendering is in some sense easier
- For diffuse surfaces, radiosity is a popular quantity to compute
- In the sequel we assume occlusion is handled separately

# Iterative Solution

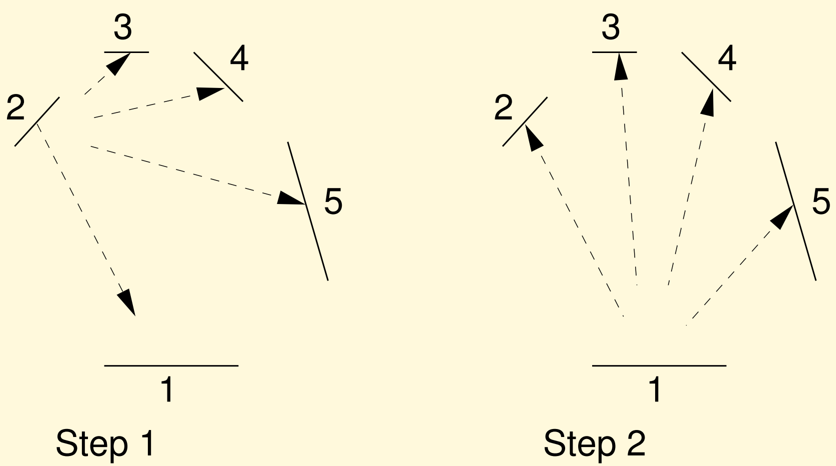
Gauss-Jordan: $B_i^{(k+1)} = E_i + \sum_{j=1}^{N} (\rho_i F_{ij}) B_j^{(k)}$ (some $i$)

$$\begin{bmatrix} X \\ B_i \\ \vdots \\ X \end{bmatrix} = \begin{bmatrix} X \\ E_i \\ \vdots \\ X \end{bmatrix} + \begin{bmatrix} X & X & X & X \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} X \\ X \\ \vdots \\ X \end{bmatrix}$$



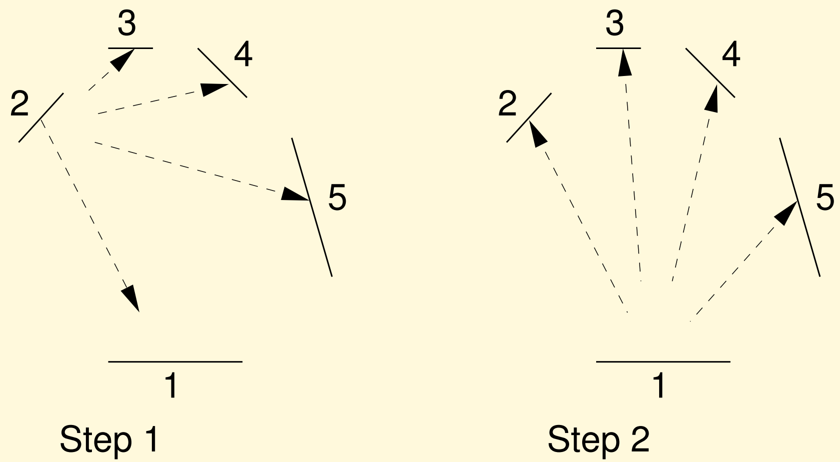

Step 1 Step 2

# Use Iterative Methods



Home Page
Title Page
Page 9 of 43
Go Back
Full Screen
Close
Quit

# Use Iterative Methods




Southwell: For all $j$ $\beta_i^{(k+1)} = \beta_i^{(k)} + \sum_{j=1}^{N} (\rho_j F_{ij}) r_i^{(k)}$

$$\begin{bmatrix} \beta_1 \\ \beta_i \\ \vdots \\ \beta_n \end{bmatrix} = \begin{bmatrix} \\ E_i \\ \vdots \\ X \end{bmatrix} + \begin{bmatrix} X \\ X \\ X \\ X \end{bmatrix} \begin{bmatrix} X \\ \vdots \end{bmatrix}$$

Results in $B_j^{(k+1)} \leftarrow B_j + \rho_j F_{ij} \frac{A_i}{A_j} \Delta B_i$

## Introduction to FMM

Suppose we have a collection of N points in 2D $x_i$ $\{i = 1, 2, \ldots N\}$ and we want to evaluate

$$f(x_j) = \sum_{i=1}^{N} \alpha_i (x_j - x_i)^2 \qquad j = 1, \ldots, N$$

Each evaluation requires $O(N)$. Since there are $N$ evaluations, straightforward method takes time $O(N^2)$

```
for j = 1 to N
  for i = 1 to N
    sum[j] += alpha[i] * (x[j]-x[i]) * (x[j]-x[i])
```

Can we do better?

# A Faster Solution: This is Not FMM

Expand $f$ as

$$
\begin{aligned}
f(x_j) &= \sum_{i=1}^{N}(\alpha_i x_j^2 + \alpha_i x_i^2 - 2\alpha_i x_j x_i) \\
&= x_j^2\left(\sum_{i=1}^{N}\alpha_i\right) + \left(\sum_{i=1}^{N}\alpha_i x_i^2\right) - 2x_j\left(\sum_{i=1}^{N}\alpha_i x_i\right)
\end{aligned}
$$

# A Faster Solution: This is Not FMM

Expand $f$ as

$$
\begin{aligned}
f(x_j) &= \sum_{i=1}^{N}(\alpha_i x_j^2 + \alpha_i x_i^2 - 2\alpha_i x_j x_i) \\
&= x_j^2\left(\sum_{i=1}^{N}\alpha_i\right) + \left(\sum_{i=1}^{N}\alpha_i x_i^2\right) - 2x_j\left(\sum_{i=1}^{N}\alpha_i x_i\right)
\end{aligned}
$$

Precompute (in $O(N)$ time)

$\beta = \left(\sum_{i=1}^{N}\alpha_i\right)$ and $\gamma = \left(\sum_{i=1}^{N}\alpha_i x_i^2\right)$ and $\delta = \left(\sum_{i=1}^{N}\alpha_i x_i\right)$

Then each evaluation is $f(x_j) = x_j^2\beta + \gamma - 2x_j\delta$ and we have the $O(N)$ code

for i = 1 to N $\beta$ += alpha[i];
for i = 1 to N $\gamma$ += alpha[i] * x[i] * x[i];
for i = 1 to N $\delta$ += alpha[i] * x[i];
for j = 1 to N sum[j] = x[j]*x[j]*$\beta$ + $\gamma$ - 2*x[j]*$\delta$

# 2. A Faster Solution: This is Not FMM

Key point : $f(x_j)$ is written as a **sum of products** using analytical manipulations

$$
\begin{aligned}
f(x_j) &= \sum_{i=1}^{N}\sum_{k=1}^{3} \alpha_i A_k(x_j) B_k(x_i) \\
&= \sum_{k=1}^{3} A_k(x_j) \underbrace{\left(\sum_{i=1}^{N} \alpha_i B_k(x_i)\right)}_{\text{precomputed}}
\end{aligned}
$$

$$A_1(x) = x^2 \quad A_2(x) = 1 \quad A_3(x) = 2x$$

$$B_1(x) = 1 \quad B_2(x) = x^2 \quad B_3(x) = x$$

Effect of the data point is independent of where it is going to be used (in management, work done is “people independent”).

# Points To Note

- Starting point is a sum of the form $f(x) = \sum_{i=1}^{N} w(y_i)K(x, y_i)$
- Faster solution because of analytical rearrangement
- But not all problems admit such a solution
- Consider $f(\omega) = \sum_{i=1}^{N} w(y_i)e^{\frac{-2\pi i\omega\sqrt{-1}}{N}}$
  - Celebrated Discrete Fourier Transform

# Points To Note

- Starting point is a sum of the form $f(x) = \sum_{i=1}^{N} w(y_i) K(x, y_i)$
- Faster solution because of analytical rearrangement
- But not all problems admit such a solution
- Consider $f(\omega) = \sum_{i=1}^{N} w(y_i) e^{\frac{-2\pi i \omega \sqrt{-1}}{N}}$
  - Celebrated Discrete Fourier Transform
  - Fast solutions are obtained due to the nature of the kernel
- FMM provides a fast solution by trading accuracy for speed but requires

# Points To Note

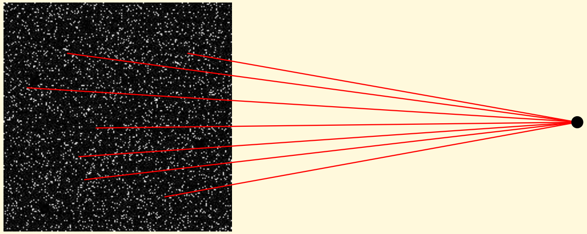
- Starting point is a sum of the form $f(x) = \sum_{i=1}^{N} w(y_i)K(x, y_i)$
- Faster solution because of analytical rearrangement
- But not all problems admit such a solution
- Consider $f(\omega) = \sum_{i=1}^{N} w(y_i)e^{\frac{-2\pi i\omega\sqrt{-1}}{N}}$
  - Celebrated Discrete Fourier Transform
  - Fast solutions are obtained due to the nature of the kernel
- FMM provides a fast solution by trading accuracy for speed but requires
  - Four types of factorization formulae
  - Analysis of convergence of analytical expression
  - Data structure issues

# Points To Note

- Starting point is a sum of the form $f(x) = \sum_{i=1}^{N} w(y_i)K(x, y_i)$
- Faster solution because of analytical rearrangement
- But not all problems admit such a solution
- Consider $f(\omega) = \sum_{i=1}^{N} w(y_i)e^{\frac{-2\pi i\omega\sqrt{-1}}{N}}$
  - Celebrated Discrete Fourier Transform
  - Fast solutions are obtained due to the nature of the kernel
- FMM provides a fast solution by trading accuracy for speed but requires
  - Four types of factorization formulae
  - Analysis of convergence of analytical expression
  - Data structure issues
- $x$ is said to be the point of evaluation of the sources $y_i$

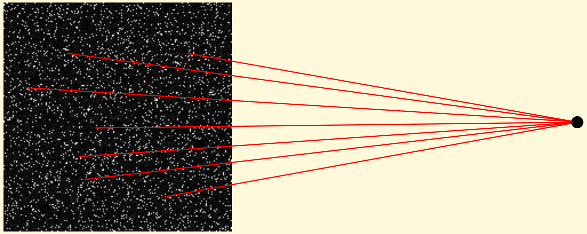
# 3. Step 1: Multipole Expansion

- Direct evaluation requires $N$ interactions at runtime

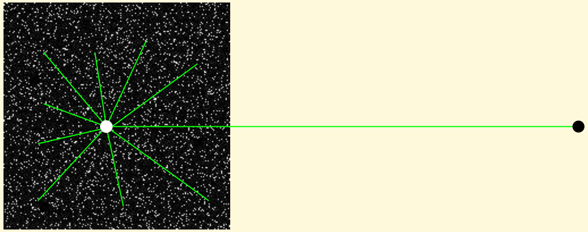


# 3. Step 1: Multipole Expansion

- Direct evaluation requires $N$ interactions at runtime



- Can we reduce this to one interaction?



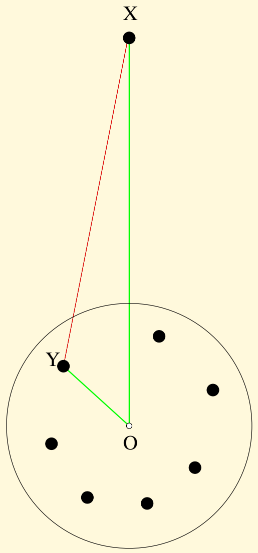
  

Home Page
Title Page
Page 15 of 43
Go Back
Full Screen
Close
Quit

# Multipole Expansion: Math

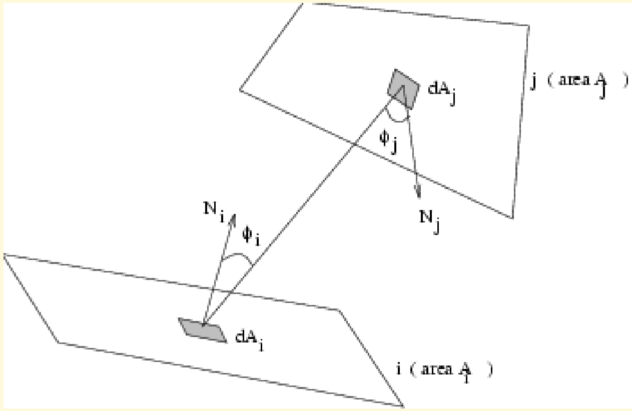
- To evaluate $f$ at $\vec{x}$ due to $\vec{y}_i\{i = 1 \ldots N\}$ $f(\vec{x}) = \sum_{i=1}^{N} w(\vec{y}_i) K(\vec{x}, \vec{y}_i)$
- If we can factorize the kernel as $K(\vec{x}, \vec{y}) = \sum_{j=1}^{p} A_j(\overrightarrow{Ox}) B_j(\overrightarrow{Oy})$
- Substituting this expansion as the kernel

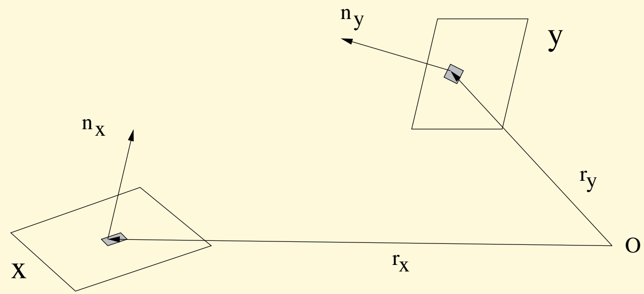
$$
\begin{aligned}
f(\vec{x}) &= \sum_{i=1}^{N} w(\vec{y}_i) \sum_{j=1}^{p} A_j(\overrightarrow{Ox}) B_j(\overrightarrow{Oy_i}) \\
&= \sum_{j=1}^{p} A_j(\overrightarrow{Ox}) \underbrace{\left( \sum_{i=1}^{N} w(y_i) B_j(\overrightarrow{Oy_i}) \right)}_{M_j(O)} \\
&= \sum_{j=1}^{p} A_j(\overrightarrow{Ox}) M_j(O)
\end{aligned}
$$




- Preprocessing cost is proportional to the number of points, but run time cost is $O(1)$

# The Radiosity Kernel Is Not Easy To Factorize




Writing the difference in energy $B(x)-E(x)$

$$= \sum_{y=y_1}^{y_k} \frac{B(y)}{A_x} \int_{A_x}\int_{A_y} \rho(x) \frac{\cos\theta_1 \cos\theta_2}{r^2} dA_x dA_y$$

$$= \sum_{y=y_1}^{y_k} \frac{B(y)}{A_x} \int_{A_x}\int_{A_y} \rho(x) \frac{[\vec{n}_y.(\vec{r}_x-\vec{r}_y)][\vec{n}_x.(\vec{r}_y-\vec{r}_x)]}{\pi|\vec{r}_y-\vec{r}_x|^4} dA_x dA_{y_i}$$

# Key Contribution: Factorization for the Radiosity Kernel

- Representing vectors as 3x1 matrices,

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

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

- Expand the expression in the numerator

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

- Define *receiver matrices* **RM** and the *source matrices* **SM**

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

$$[\vec{n}_y.(\vec{r}_x - \vec{r}_y)][\vec{n}_x.(\vec{r}_y - \vec{r}_x)] = \mathbf{RM}(\vec{r}_x) \otimes \mathbf{SM}(\vec{r}_y)$$

# Key Contribution: Factorization for the Radiosity Kernel

- For $r_y < r_x$ (Hausner, 1997)

# Key Contribution: Factorization for the Radiosity Kernel

- For $r_y < r_x$ (Hausner, 1997)

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

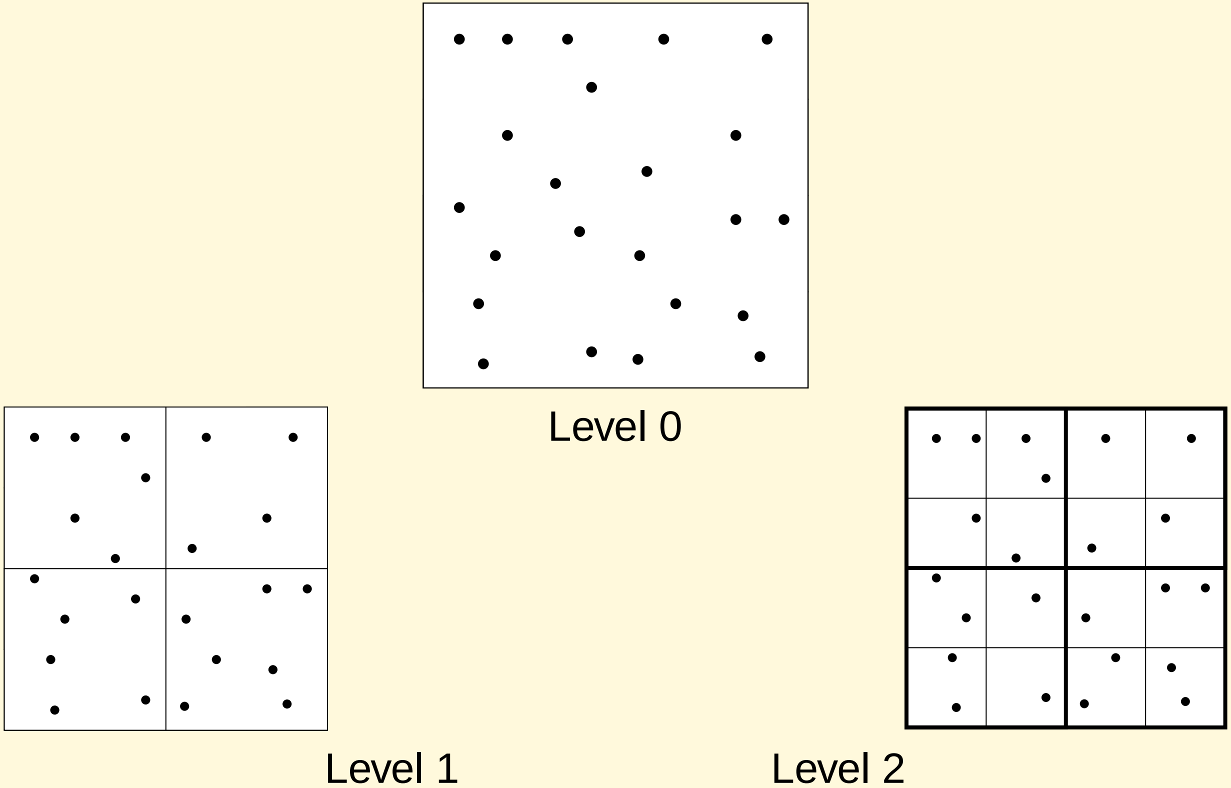
- We have the factored radiosity kernel

$$\begin{aligned} B(x) - E(x) &= \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} e_n^j R_{nj}^m(\vec{r}_x) \otimes M_{nj}^m(O) \\ R_{nj}^m(\vec{r}_x) &= \frac{1}{A_x} \int_{A_x} \frac{\rho(x)}{r_x^{n+4}} Y_{n-2j}^m(\theta_x, \phi_x) \mathbf{RM}(\vec{r}_x) dA_x \\ M_{nj}^m(O) &= \sum_{y=y_1}^{y_k} \int_{A_y} B(y) r_y^n \overline{Y_{n-2j}^m(\theta_y, \phi_y)} \mathbf{SM}(\vec{r}_y) dA_y \end{aligned}$$

- Two observations
  - $r_y < r_x$
  - $p = \infty$

# The $N \log N$ Algorithm (2 Dimensions)

- Assumption: Particles are uniformly distributed in a square
- Consider a uniform hierarchical subdivision of space



The Fast Multipole Method
A Faster Solution: This . . .
Step 1: Multipole . . .
Interaction List
Step 2: Translation of . . .
Step 3: Local Expansion
Step 4: Translation of . . .
The Overall Algorithm

Home Page

Title Page

Page 20 of 43

Go Back

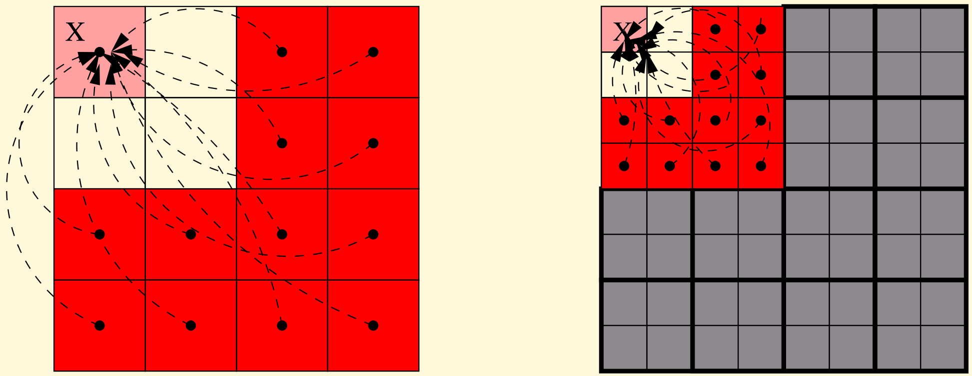
Full Screen

Close

Quit

# The $N \log N$ Algorithm (2 Dimensions)

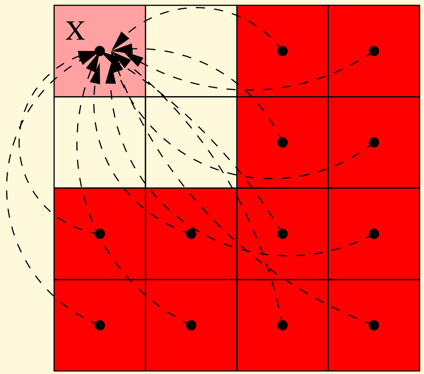
- *Nearest neighbors* share a vertex
- *Well separated Boxes* are on the same level and are not *nearest neighbors*

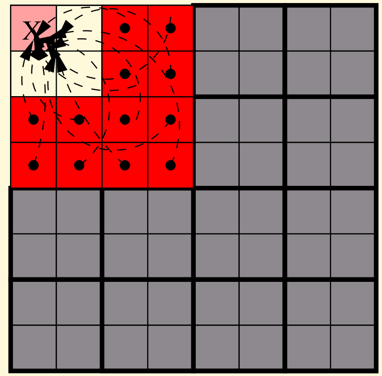



- Calculate multipole moments at the center of each box of Level 2
- Multipole Expansion cannot be evaluated at nearest neighbors
- Recursion!

# The $N \log N$ Algorithm (2 Dimensions)

- *Nearest neighbors* share a vertex
- *Well separated Boxes* are on the same level and are not *nearest neighbors*

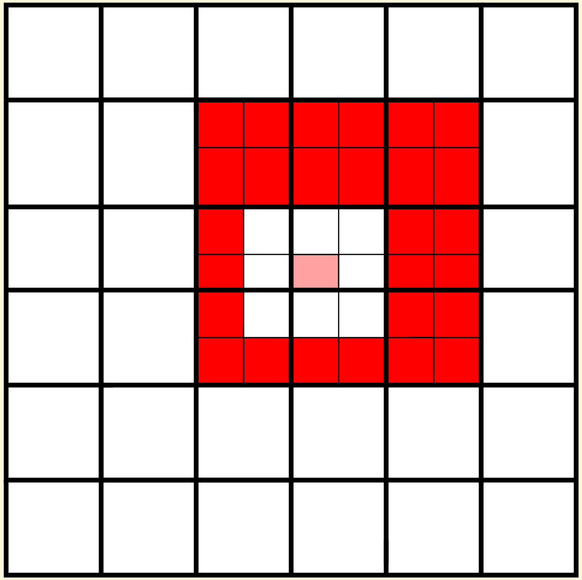



- Calculate multipole moments at the center of each box of Level 2
- Multipole Expansion cannot be evaluated at nearest neighbors
- Particles we have not yet accounted for in their interaction with box X are in children of the near neighbors of X's parent (at level 3).
- Recursion!

# 4. Interaction List

- *Interaction List* of a box i consists of children of near neighbors of i's parent which are well separated from i.
- Maximum size of the interaction list is 27



## The $N \log N$ Algorithm (2 Dimensions)

- Algorithm
  - Subdivide till each box contains not more than $M$ particles

    | | |
    |---|---|
    | Number of leaves | $N/M$ |
    | Depth $d$ of quad-tree | $\log_4(N/M)$ |
    | Total number of boxes | $\sum_{i=1}^{d} 4^i \approx 4N/3M$ |

  - At each level, calculate multipole moments at each box $Np$
  - At each level, for each particle, evaluate the multipole expansion
    of all boxes in its owners interaction list $27Np$
  - For the last level, for each particle calculate directly its interaction with all particles in its owners nearest neighbors $O(N)$
- Total Cost $O(N \log N)$!

## The $N \log N$ Algorithm (2 Dimensions)

- Algorithm
  - Subdivide till each box contains not more than $M$ particles

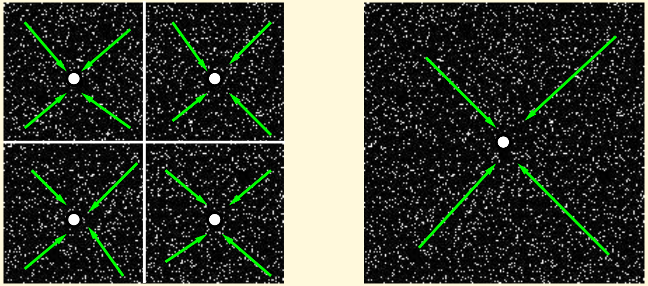
    | | |
    |---|---|
    | Number of leaves | $N/M$ |
    | Depth $d$ of quad-tree | $\log_4(N/M)$ |
    | Total number of boxes | $\sum_{i=1}^{d} 4^i \approx 4N/3M$ |

  - At each level, calculate multipole moments at each box $Np$
  - At each level, for each particle, evaluate the multipole expansion
    of all boxes in its owners interaction list $27Np$
  - For the last level, for each particle calculate directly its interaction with all particles in its owners nearest neighbors $O(N)$
- Total Cost $O(N \log N)$!
- We are accessing each particle at every level
- Can we do better?

# 5. Step 2: Translation of Multipole Expansion

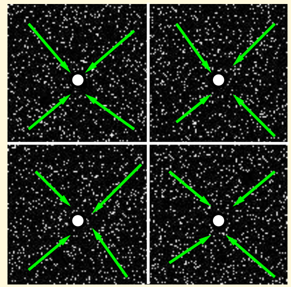
- Why recompute multipole moments for each particle at different centers?

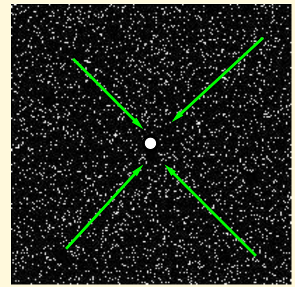


Home Page
Title Page
Page 24 of 43
Go Back
Full Screen
Close
Quit

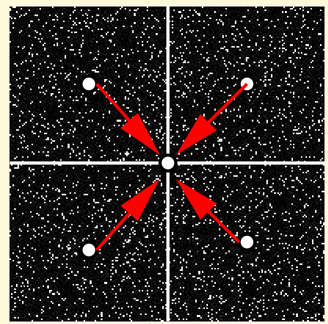
# 5. Step 2: Translation of Multipole Expansion

- Why recompute multipole moments for each particle at different centers?



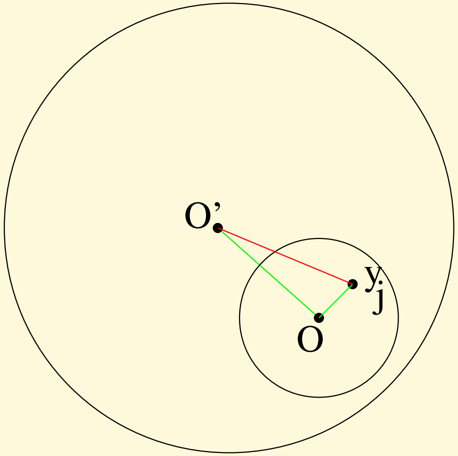


- Reuse computation, shift multipole centers



# Translation of Multipole Expansion : Math




- We have the multipole moments due to $N$ particles at $O$ as

$$M_j(O) = \sum_{i=1}^{N} w(y_i) B_j(\overrightarrow{Oy_i})$$

- To find the moments at $O'$ given the moments at $O$, if we can expand $B_j$ as

$$B_j(\overrightarrow{O'y_i}) = \sum_{k=1}^{p} B_k(\overrightarrow{Oy_i})\alpha_k^j(\overrightarrow{OO'})$$

- Then the multipole moment at $O'$ will be

$$\begin{aligned} M_j(O') &= \sum_{i=1}^{N} w(y_i) B_j(\overrightarrow{O'y_i}) \\ &= \sum_{i=1}^{N}\sum_{k=1}^{p} w(y_i) B_k(\overrightarrow{Oy_i})\alpha_k^j(\overrightarrow{OO'}) \\ &= \sum_{k=1}^{p} M_k(O)\alpha_k^j(\overrightarrow{OO'}) \end{aligned}$$

- Coefficients can be computed in $O(p^2)$ time

# Contribution: Translation Theorem for the Radiosity Kernel

$$B(x) - E(x) = \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} e_n^j R_{nj}^m(\vec{r}_x) \otimes M_{nj}^m(O')$$

$$M_{nj}^m(O') = \sum_{k=0}^{n} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} \sum_{j'=j_{min}}^{j_{max}} \frac{e_{n-k}^j}{e_n^j} J(\ldots)|OO'|^k Y_{k-2s}^{m_1}(OO') \mathbf{TM}(OO') \otimes M_{n-k,j'}^{m_1+m}(O)$$

## Contribution: Translation Theorem for the Radiosity Kernel

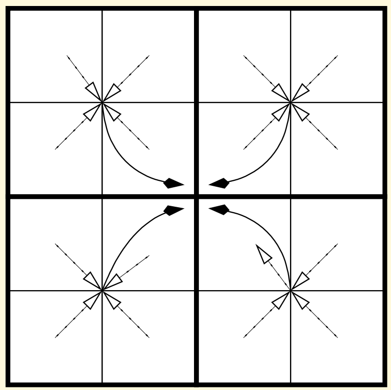
$$
\begin{aligned}
B(x) - E(x) &= \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} e_n^j R_{nj}^m(\vec{r}_x) \otimes M_{nj}^m(O') \\
M_{nj}^m(O') &= \sum_{k=0}^{n} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} \sum_{j'=j_{min}}^{j_{max}} \\
&\quad \frac{e_{n-k}^j}{e_n^j} J(\ldots)|OO'|^k Y_{k-2s}^{m_1}(OO')\mathbf{TM}(OO') \otimes M_{n-k,j'}^{m_1+m}(O)
\end{aligned}
$$

This one was rough... Used a theorem from Sack 1963

# $N \log N$ to $N$ : Translation of Multipole Expansion

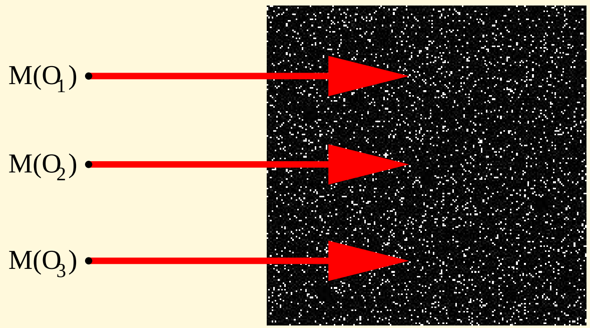
- Computing multipole moments at all levels was $O(N \log N)$
- Revised Procedure
  - Compute multipole moments at the lowest level $O(N)$
  - Shift and aggregate multipole moments upwards till Level 2 $O(\sum_{i=3}^{d} 4^i p^2) \approx O(4p^2N/3M)$



- But we are still evaluating multipole expansions for **each** particle at each level

# 6. Step 3: Local Expansion

- Multipole moments represent field **outside** a cluster in a constant number of coefficients

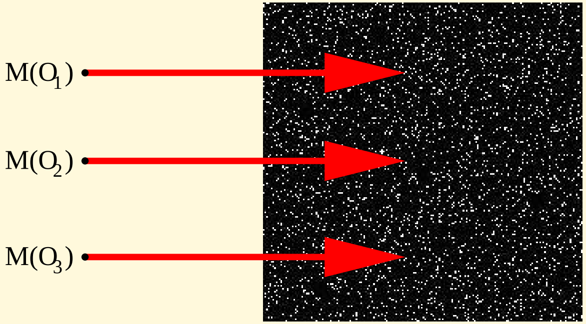




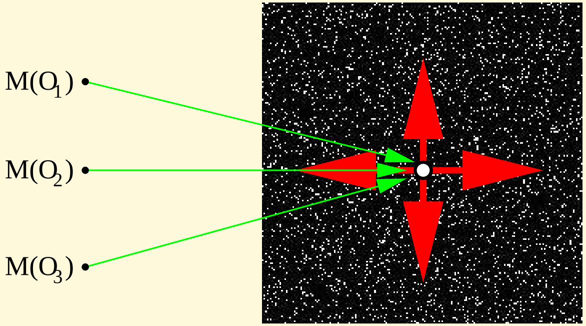

Home Page
Title Page
Page 28 of 43
Go Back
Full Screen
Close
Quit

# 6. Step 3: Local Expansion

- Multipole moments represent field **outside** a cluster in a constant number of coefficients




- Can external multipole moments be combined into a constant number of coefficients to represent the field **inside** a cluster?



# Local Expansion : Math

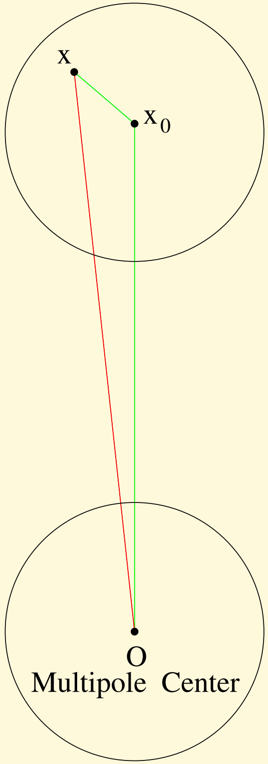
- To calculate $f$ at $\vec{x}$ we now have $f(\vec{x}) = \sum_{i=1}^{p} A_i(\overrightarrow{Ox}) M_i(O)$
- Now suppose we make $A_i$ degenerate in $x$ and $O$ about $x_0$ as

$$A_i(\overrightarrow{Ox}) = \sum_{l=1}^{p} C_l(\overrightarrow{x_0 x}) \beta_l^i(\overrightarrow{Ox_0})$$

- Substituting this expansion as $A_i$, we have

$$\begin{aligned} f(\vec{x}) &= \sum_{l=1}^{p} C_l(\overrightarrow{x_0 x}) \underbrace{\left( \sum_{i=1}^{p} \beta_l^i(\overrightarrow{Ox_0}) M_i(O) \right)}_{L_l(\vec{x}_0)} \\ &= \sum_{l=1}^{p} C_l(\overrightarrow{x_0 x}) L_l(\vec{x}_0) \end{aligned}$$

- $L_l(\vec{x}_0)$ is the $l$th local expansion coefficient about $\vec{x}_0$ and is computed in $O(p^2)$ time



## Contribution: Local Expansion for the Radiosity Kernel

If $|\vec{Ox_0}| > |\vec{x_0 x}|$

$$
\begin{aligned}
B(x) - E(x) &= \sum_{k=0}^{\infty} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} E_{ksm_1}(x_0 x) \otimes L_{ksm_1}(Ox_0) \\
E_{ksm_1}(x_0 x) &= \frac{1}{A_x} \int_{A_x} \rho(x) |x_0 x|^k Y_{k-2s}^{m_1}(x_0 x) \mathbf{RM}(x_0 x) dA_x \\
L_{ksm_1}(Ox_0) &= \sum_{n=0}^{\infty} \sum_{j=0}^{[n/2]} \sum_{m=-n+2j}^{n-2j} \sum_{l_2=l_{min}}^{l_{max}} e_n^j J(\ldots) F_{ksm_1 l_2}^{njm}(Ox_0) \otimes M_{nj}^{m}(O) \\
F_{ksm_1 l_2}^{njm}(Ox_0) &= |Ox_0|^{-n-4-k} Y_{l_2}^{m-m_1}(Ox_0) \mathbf{TM}(Ox_0)
\end{aligned}
$$

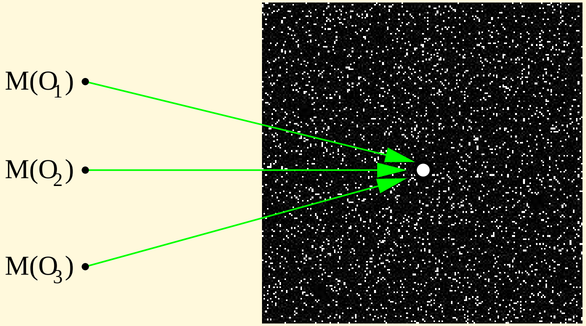
## $N \log N$ to $N$ : Local Expansion

- At each level, multipole expansions of each box in its interaction list was evaluated at a particle $27Np\log N$
- Revised Procedure
  - For each box at each level, combine multipole moments of the boxes in its interaction list into local coefficients at its center and evaluate for each particle $\sum_{i=3}^{d} 27(4^i p^2) + Np) \approx 36p^2 N/M + Np\log N$
- Why evaluate at each level?

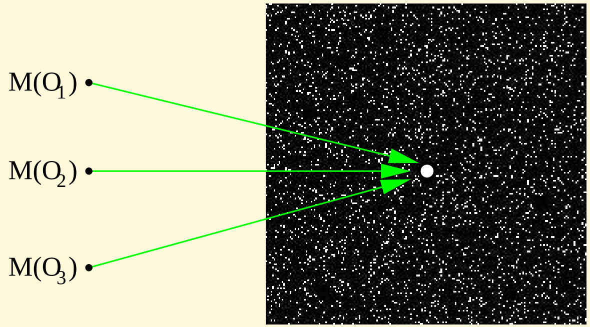
# 7. Step 4: Translation of Local Expansion

- Collect local expansion coefficients but do not evaluate

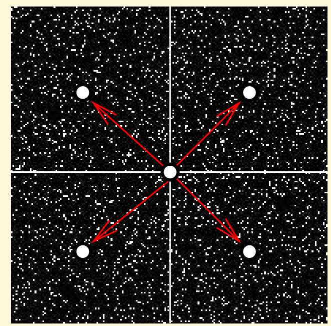


# 7. Step 4: Translation of Local Expansion

- Collect local expansion coefficients but do not evaluate



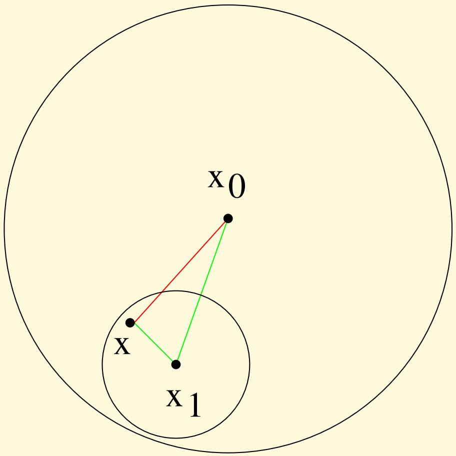

- Shift the center of the local expansion of a box to each of its children



# Translation of Local Expansion : Math

- To calculate $f$ at $\vec{x}$, we now have

$$f(\vec{x}) = \sum_{i=1}^{p} C_i(\overrightarrow{x_0 x}) L_i(\vec{x}_0)$$




- To calculate $f$ at $\vec{x}$ in terms of a shifted center $\vec{x}_1$, we expand $r_i$ as

$$C_i(\overrightarrow{x_0 x}) = \sum_{j=1}^{p} C_j(\overrightarrow{x_1 x}) \gamma_j^i(\overrightarrow{x_0 x_1})$$

- Substituting this expansion of $C_i$

$$f(\vec{x}) = \sum_{j=1}^{p} C_j(\overrightarrow{x_1 x}) \left( \sum_{i=1}^{p} \gamma_j^i(\overrightarrow{x_0 x_1}) L_i(\vec{x}_0) \right) = \sum_{j=1}^{p} C_j(\overrightarrow{x_1 x}) L_j(x_1)$$

- Thus we can evaluate $f$ in terms of the translated local expansion coefficients $L_j(x_1) = \sum_{i=1}^{p} \gamma_j^i(\overrightarrow{x_0 x_1}) L_i(\vec{x}_0)$ in $O(p^2)$ time

# Contribution: Local Expansion Translation for the Radiosity Kernel

$$B(x) - E(x) = \sum_{k=0}^{\infty} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} E_{ksm_1}(x_1 x) \otimes L_{ksm_1}(x_0 x_1)$$

$$L_{k's'm'_1}(x_0 x_1) = \sum_{k=k'}^{\infty} \sum_{s=0}^{k/2} \sum_{m_1=-k+2s}^{k-2s} \sum_{j'=j_{min}}^{j_{max}} (-1)^{k+m_1} \frac{e_{k-k'}^{j'}}{e_k^s} J(\ldots)$$

$$|x_1 x_0|^{k-k'} \overline{Y_{k-k'-2j'}^{-m'_1+m_1}}(x_1 x_0) \mathbf{TM}(x_0 x_1) \otimes L_{ksm_1}(O x_0)$$

# $N \log N$ to $N$ : Local Expansion Translation

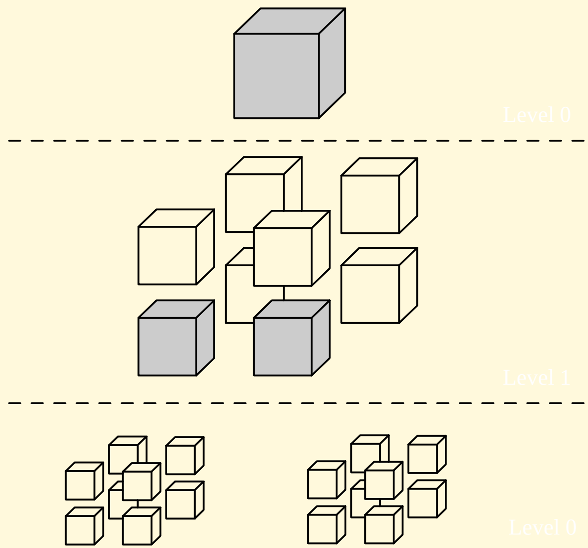
- For each box at each level, combine multipole moments of the boxes in its interaction list into local coefficients at its center and evaluate for each particle
$36p^2N/M + Np\log N$
- Revised Procedure
  - For each box at each level, combine multipole moments of the boxes in its interaction list into local coefficients and shift them to its children
  $\sum_{i=3}^{d} 27(4^i p^2) + 4^i p^2 \approx 36p^2N/M + 4p^2N/3M$
  - At the last level, evaluate the local expansion in each of the boxes at its constituent particles and compute direct interaction with near neighbors
  $9M^2(N/M) + (pM)(N/M) = 9NM + pN$
- We are in $O(N)$!

Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 36 of 43
Go Back
Full Screen
Close
Quit

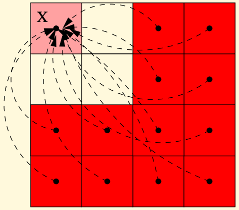
# 8. The Overall Algorithm

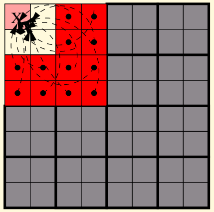
- Step 1 Construct an Octree
  Subdivide the space containing the whole system as an octree until the leaves contain not more than a constant number of bodies $k$



# The Fast Multipole Algorithm

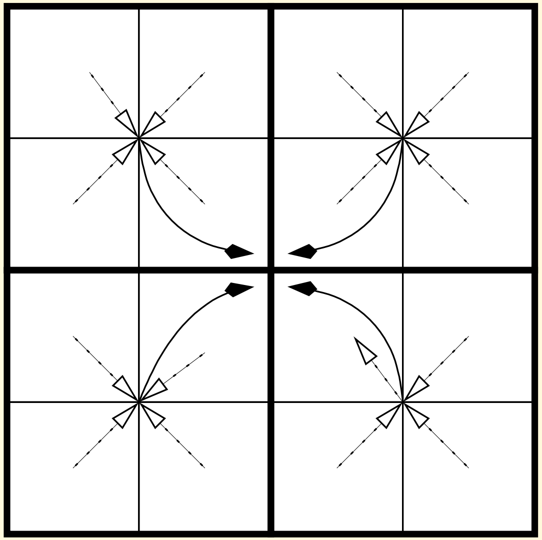
- Step 2 Compute Interaction Lists
  Two cells are *nearest neighbors* if they are at the same refinement level and are separated by not more than one cell.
  Two cells are *well separated* if they are at the same refinement level and are not nearest neighbors
  With each cell i, associate an *interaction list* consisting of children of nearest neighbors of i's parents which are well separated from cell i



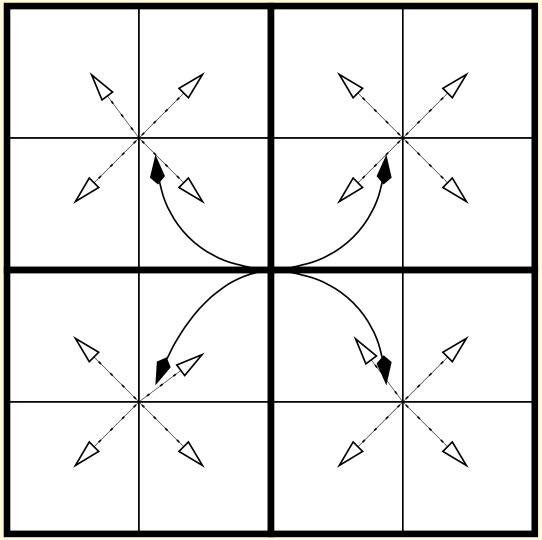
# The Fast Multipole Algorithm

- Step 3 Compute Multipole Moments (Upward)
  Compute the multipole moments $M_n^m$ for each leaf cell, at the center of the cell, due to the particles contained within the cell. For a non leaf cell, translate and aggregate the multipole moments of its children to its center. This is repeated at each level in an *upward pass* at the end of which, we have the multipole moments in each cell due to the particles contained in that cell.



# The Fast Multipole Algorithm

- Step 4 Compute Local Expansion Coefficients (Downward)
  Starting from the root cell, for each cell at level $l$, the multipole moments of all the cells in its interaction list are translated to local expansion coefficients about the center of the cell. The local expansion coefficients of the parent are then translated to the center of this cell and aggregated.

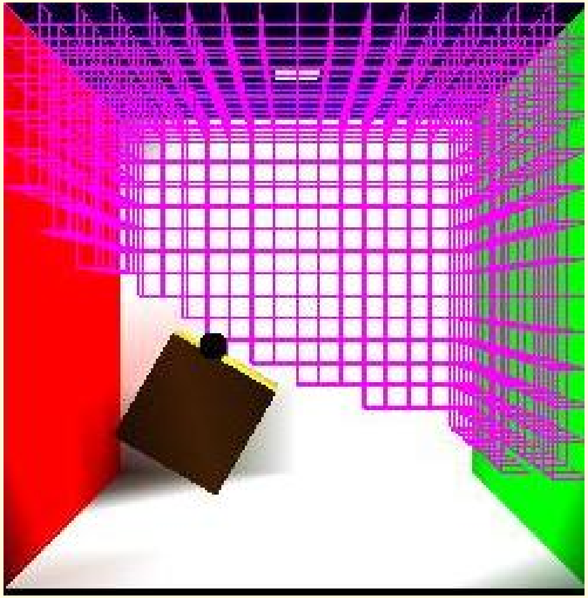


## The Fast Multipole Algorithm

- Step 5 Final Evaluation
  For each particle in the system, evaluate the local expansion using the local expansion coefficients of the cell to which it belongs. Interaction with all particles in its nearest neighbors and its parent cell are computed directly. The contribution from these sources is added to get the radiosity at that particle.

# Handling Occlusions

Visibility is a point to point phenomenon, and is not analytical



# Handling Occlusions in FMM

```
Procedure Modify(Box A) {
visible_interactionlist(A)=Null
    for each box B ∈ old_interactionlist(A){
        state=visibility(A,B)
        if equals(state,valid) then
            visible_interactionlist(A).Include(B)
        else if equals(state,partial) then
            { if(notLeaf(A))
                for each a ∈ child(A)
                for each b ∈ child(B)
                  interactionlist(a).Include(b)}}}
```

# Handling Occlusions in FMM

```
Procedure visibility(Box A, Box B) {
    visible=0;
    for each cell a ∈ leafcell(A)
      for each cell b ∈ leafcell(B){
        if FacingEachOther(a,b) then {
              result=shootAndDetect(a,b)
              if equals(result,0) then Increment(visible,1) }}
    if equals(visible,0) return(invalid)
    else if equals(visible,leafcell(A).size*leafcell(B).size)
          return(valid); else return(partial) }
```

```
Procedure Generate(Box A){
      Modify(A)
      for each a ∈ child(A)
        Generate(a)}
```

## Conclusion

- A quick introduction to FMM and Global Illumination (GI)
- Reduction of the GI problem to problems similar to FMM
- Four new theorems for the radiosity kernel
- Also supporting implementation