Light Clustering for Photorealistic Rendering

Norbert Bus

Department of Computer Science
ESIEE Paris

June 13, 2013

Joint work with Nabil H. Mustafa and Venceslas Biri
Rendering a scene

**Scene**
- objects (geometry, color)
- light sources

**Rendering the image**
- shoot a ray from the camera through a pixel
- determine the color of the first hit point → shoot other rays to light sources
Rendering equation

Color for a surface point [J. T. Kajiya, SIGGRAPH ’86]

\[
L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f(x, \omega_i, \omega_o) L_i(x, \omega_i) \cos \theta_i d\omega_i
\]

\[
= \int_{\Omega} f(x, \omega_i, \omega_o) L_o(hit(x, \omega_i), \omega_i) \cos \theta_i d\omega_i
\]

Approximating the solution with path tracing
- Simulate indirect illumination
- Place virtual point lights (VPLs)
- Shoot rays from original lights
With \( S \), the set of VPLs, the rendering equation

\[
L(p, \omega_o) = \sum_{i \in S} V_i(p) f(p, \omega_i, \omega_o) l_i G_i(p, \omega_i)
\]
With $S$, the set of VPLs, the rendering equation

$$L(p, \omega_0) = \sum_{i \in S} V_i(p) f(p, \omega_i, \omega_0) l_i G_i(p, \omega_i)$$

- Good solution with $\approx 100000$ VPLs
- Very expensive to calculate visibility
- How to speed it up?
- Cluster similar lights into groups and treat them as single lights
Preprocess

- Single VPLs
Preprocess

- Single VPLs
- Agglomerative clustering
  \[ \text{sim}(a, b) = I(\text{dist}(a, b) + \text{direction}(a, b)) \]
Preprocess

- Single VPLs
- Agglomerative clustering

\[
\text{sim}(a, b) = I(\text{dist}(a, b) + \text{direction}(a, b))
\]
Preprocess

- Single VPLs
- Agglomerative clustering
  \[ \text{sim}(a, b) = I(\text{dist}(a, b) + \text{direction}(a, b)) \]
Preprocess

- Single VPLs
- Agglomerative clustering
  \[ \text{sim}(a, b) = I(\text{dist}(a, b) + \text{direction}(a, b)) \]

Query for one pixel

- Descend from the root until optimal clustering (cut)
Preprocess

- Single VPLs
- Agglomerative clustering
  \[ \text{sim}(a, b) = I(\text{dist}(a, b) + \text{direction}(a, b)) \]

Query for one pixel

- Descend from the root until optimal clustering (cut)
Preprocess

- Single VPLs
- Agglomerative clustering
  \[ \text{sim}(a, b) = I(\text{dist}(a, b) + \text{direction}(a, b)) \]

Query for one pixel

- Descend from the root until optimal clustering (cut)
Preprocess

- Single VPLs
- Agglomerative clustering
  $sim(a, b) = I(dist(a, b) + direction(a, b))$

Query for one pixel

- Descend from the root until optimal clustering (cut)
Weakness

- Agglomerative clustering may be slow
- For each pixel we re-evaluate the optimal cut
- Too expensive
Weakness

- Agglomerative clustering may be slow
- For each pixel we re-evaluate the optimal cut
- Too expensive
- Common cuts [Wang et al., SIGGRAPH Asia ’11 ]
Weakness

- Agglomerative clustering may be slow
- For each pixel we re-evaluate the optimal cut
- Too expensive
- Common cuts [Wang et al., SIGGRAPH Asia ’11 ]
- Our intuition
- Structure that effectively captures all the different cuts
Well-separated pair

**Definition**

Two point sets, $P$ and $Q$ are well-separated for a fixed $\epsilon > 0$ if

$$\max(\text{diam}(P), \text{diam}(Q)) < \epsilon \cdot \text{dist}(P, Q)$$

Bounds the angles and distances which is important for lights
This enables us to bound the error of light clustering
A well-separated pair decomposition (WSPD) of a point set, $P$ is a set of pairs

$$W = \{(A_1, B_1), \ldots (A_s, B_s)\}, \quad A_i, B_i \subset P$$

such that:

1. for $\forall p, q \in P$ there exists exactly one $i$ such that $(p, q) \in (A_i, B_i)$
2. $A_i, B_i$ is well-separated for $\forall i$

Example: $W = \{(p, q) \mid p, q \in P\}$, size of $O(n^2)$
Theorem

For $\epsilon > 0$, $P \subset \mathbb{R}^d$, where $|P| = n$ there exists a WSPD of size $O(n\epsilon^{-d})$ and one can compute it in $O(n \log n + n\epsilon^{-d})$ time.
Theorem

For $\epsilon > 0$, $P \subset \mathbb{R}^d$, where $|P| = n$ there exists a WSPD of size $O(n\epsilon^{-d})$ and one can compute it in $O(n \log n + n\epsilon^{-d})$ time.

- Size of the WSPD is the number of pairs
- Build a compressed quadtree, $O(n \log n)$
- Recurse down from the root to find well separated pairs, $O(n\epsilon^{-d})$
WSPD algorithm
WSPD algorithm
WSPD algorithm

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

```
    5
   /  
  4   1
     /  
    3   3
   /  
  4   2
```
WSPD algorithm

![Diagram of WSPD algorithm with a table and a tree structure]

- The table shows a grid with numbers: 4, 2, 6, 1, 3, 5.
- The tree structure starts with a root node labeled 5, with branches leading to nodes 4 and then to other nodes.

---

*Note: The diagram includes a visual representation of the WSPD algorithm with nodes and connecting lines.*
### WSPD algorithm

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

#### Diagram

```
      5
     /   
    4    6
   / 
  2 1
```

- Node 1 is connected to nodes 2 and 6.
- Node 3 is connected to node 1.
- Node 4 is connected to node 2.
- Node 6 is connected to node 1.
- Node 5 is the root of the tree.
### WSPD algorithm

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Diagram:

- Root: 5
- Branch: 4, 1, 3
- Branch: 2, 6
WSPD algorithm
WSPD algorithm
WSPD algorithm
What does this mean?

- A point $p$ is contained in several pairs
- These pairs form a clustering of $P$ with respect to $p$
- The WSPD efficiently stores a clustering with respect to every point
What does this mean?

- A point $p$ is contained in several pairs
- These pairs form a clustering of $P$ with respect to $p$
- The WSPD efficiently stores a clustering with respect to every point
What does this mean?

- A point $p$ is contained in several pairs
- These pairs form a clustering of $P$ with respect to $p$
- The WSPD efficiently stores a clustering with respect to every point
What does this mean?

- A point $p$ is contained in several pairs
- These pairs form a clustering of $P$ with respect to $p$
- The WSPD efficiently stores a clustering with respect to every point
What does this mean?

- A point $p$ is contained in several pairs
- These pairs form a clustering of $P$ with respect to $p$
- The WSPD efficiently stores a clustering with respect to every point
Use WSPD on VPLs but three questions arise
Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
Our WSPD application

Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
  - query WSPD

- Speed?
  - theoretical: \( O(\epsilon - d \log n) \)
  - measured: 4x faster than Lightcuts

Pure spatial clustering → lighting specific WSPD

ongoing work
Our WSPD application

Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
  - query WSPD

- Speed?
Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
  - query WSPD

- Speed?
  - theoretical: $O(\epsilon^{-d} \log n)$
Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
  - query WSPD

- Speed?
  - theoretical: $O(\epsilon^{-d} \log n)$
  - measured: 4x faster than Lightcuts
Our WSPD application

Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
  - query WSPD

- Speed?
  - theoretical: $O(\epsilon^{-d} \log n)$
  - measured: 4x faster than Lightcuts

- Pure spatial clustering $\rightarrow$ lighting specific WSPD
Our WSPD application

Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
  - query WSPD

- Speed?
  - theoretical: $O(\epsilon^{-d} \log n)$
  - measured: 4x faster than Lightcuts

- Pure spatial clustering → lighting specific WSPD
  - ongoing work
Our algorithm

Preprocess, given a set of VPLs

- Create compressed quadtree on the VPLs
- Create WSPD pairs
Our algorithm

Preprocess, given a set of VPLs
- Create compressed quadtree on the VPLs
- Create WSPD pairs

Render a pixelpoint, $q$
- Query the WSPD for $(A_i, B_i)$ st. $q \in A_i$
- Return the clusters, $\{B_i\}$
Our algorithm

Preprocess, given a set of VPLs

- Create compressed quadtree on the VPLs
- Create WSPD pairs

\[ O(n \log n) \]

Render a pixelpoint, \( q \)

- Query the WSPD for \((A_i, B_i)\) st. \( q \in A_i \)
- Return the clusters, \( \{B_i\} \)
Query for the point $q$

Query the WSPD
Query the WSPD

Query for the point $q$

- Take the closest point to $q$, denote it by $p$
Query the WSPD

Query for the point $q$

- Take the closest point to $q$, denote it by $p$
- Let $\lambda = \text{dist}(p, q)$
Query the WSPD

Query for the point $q$

- Take the closest point to $q$, denote it by $p$
- Let $\lambda = \text{dist}(p, q)$
- Return the pairs, $(B_i)$, of $p$ such that $\text{dist}(p, B_i) > \frac{\lambda}{\epsilon}$
Query the WSPD

Query for the point $q$

- Take the closest point to $q$, denote it by $p$
- Let $\lambda = \text{dist}(p, q)$
- Return the pairs, $(B_i)$, of $p$ such that $\text{dist}(p, B_i) > \frac{\lambda}{\epsilon}$
- For the pairs with $\text{dist}(p, B_i) < \frac{\lambda}{\epsilon}$ check for well-separatedness and refine if necessary
Lemma I

For any $q \in \mathbb{R}^d$ and its nearest neighbour $p$, the well separated pairs of $p$ are also well separated from $q$ if $\text{dist}(p, B_i) > \text{dist}(p, q) + \epsilon$ and of those lying closer there are at most $O(1)$.
Lemma I

For any $q \in \mathbb{R}^d$ and its nearest neighbour $p$, the well separated pairs of $p$ are also well separated from $q$ if $\text{dist}(p, B_i) > \text{dist}(p, q)$ and of those lying closer there are at most $O(1)$. 
**Correctness**

**Lemma 1**

For any \( q \in \mathbb{R}^d \) and its nearest neighbour \( p \), the well separated pairs of \( p \) are also well separated from \( q \) if \( \text{dist}(p, B_i) > \frac{\text{dist}(p, q)}{\epsilon} \) and of those lying closer there are at most \( O(1) \).
Complexity - query

Complexity of the query phase

- Nearest neighbor

Refining the pairs,

$$O(1)$$

Average number of pairs is

$$O(\epsilon - d \log n)$$

Approximate nearest neighbor

For any $$p \in \mathbb{R}^d$$ its parent node in a compressed quadtree can be found in $$O(\log n)$$ time using a finger tree.

Lemma II

The expected distance is

$$O(\lambda \log \lambda)$$

where $$\lambda$$ is the distance from the nearest neighbor and Lemma I holds in this case too.
Complexity of the query phase

- Approximate nearest neighbor, $O(\log n)$ expected time

Approximate nearest neighbor

For any $p \in \mathbb{R}^d$ its parent node in a compressed quadtree can be found in $O(\log n)$ time using a finger tree.
Complexity - query

Complexity of the query phase

- Approximate nearest neighbor, $O(\log n)$ expected time

Approximate nearest neighbor

For any $p \in \mathbb{R}^d$ its parent node in a compressed quadtree can be found in $O(\log n)$ time using a finger tree.

Lemma II

The expected distance is $O(\lambda \log \lambda)$ where $\lambda$ is the distance form the nearest neighbor and Lemma I holds in this case too.
Complexity - query

Complexity of the query phase

- Approximate nearest neighbor, $O(\log n)$ expected time
- Refining the pairs, $O(1)$

**Approximate nearest neighbor**

For any $p \in \mathbb{R}^d$ its parent node in a compressed quadtree can be found in $O(\log n)$ time using a finger tree.

**Lemma II**

The expected distance is $O(\lambda \log \lambda)$ where $\lambda$ is the distance form the nearest neighbor and Lemma I holds in this case too.
Complexity - query

Complexity of the query phase

- Approximate nearest neighbor, $O(\log n)$ expected time
- Refining the pairs, $O(1)$
- Average number of pairs is $O(\epsilon^{-d} \log n)$

**Approximate nearest neighbor**

For any $p \in \mathbb{R}^d$ its parent node in a compressed quadtree can be found in $O(\log n)$ time using a finger tree.

**Lemma II**

The expected distance is $O(\lambda \log \lambda)$ where $\lambda$ is the distance form the nearest neighbor and Lemma I holds in this case too.
Results

On average we have reached about 4× speed up in the rendering phase, with comparable quality.

VPL

WSPD
Results

On average we have reached about 4x speed up in the rendering phase, with comparable quality.

16 x euclidian difference
Ongoing work

- Improve clustering quality
- Introduce *light measure*, light specific WSPD
- Model the geometry of the scene
- VPLs on surfaces → size is $O(n\epsilon^{-(d-1)})$
Summary

- Rendering scenes with VPLs
- Lightcuts method for clustering
- Novel WSPD based algorithm for clustering

Thank you!
Bibliography

S. Har-Peled
*Geometric Approximation Algorithms*
Amer Mathematical Society, ’11.

B. Walter et al
Lightcuts: a scalable approach to illumination

G. Wang et al
Efficient search of lightcuts by spatial clustering,
*SIGGRAPH Asia ’11 Sketches*, December ’11

Brian Taylor
Image on slide 2

Wikipedia
Image on slide 3,4