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

Photorealistic rendering

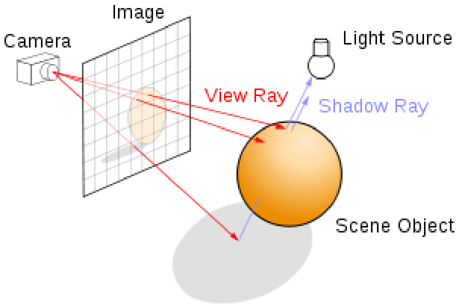
Rendering a scene

Scene

- objects (geometry, color)
- light sources

Rendering the image

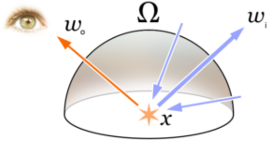
- shoot a ray from the camera through a pixel
- determine the color of the first hit point $\rightarrow$ shoot other rays to light sources



Rendering equation

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

$$
\begin{aligned}
L_o(x, \omega_o) &= L_e(x, \omega_o) + \int_\Omega f(x, \omega_i, \omega_o) L_i(x, \omega_i) \cos\theta_i \mathrm{d}\omega_i \\
&= \int_\Omega f(x, \omega_i, \omega_o) L_o(hit(x, \omega_i), \omega_i) \cos\theta_i \mathrm{d}\omega_i
\end{aligned}
$$



Approximating the solution with path tracing

VPL

- Simulate indirect illumination
- Place virtual point lights (VPLs)
- Shoot rays from original lights

VPL

- With S, the set of VPLs, the rendering equation

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

VPL

- With S, the set of VPLs, the rendering equation

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

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

Lightcuts [Walter et al., ACM Trans. Graph. '05]

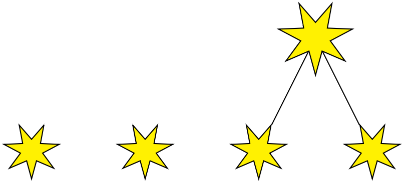
Preprocess

- Single VPLs

Lightcuts [Walter et al., ACM Trans. Graph. '05]

Preprocess

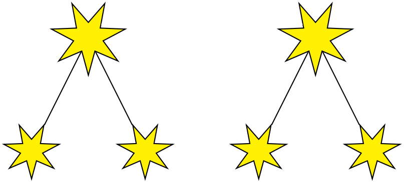
- Single VPLs
- Agglomerative clustering
  $sim(a, b) =$
  $I(dist(a, b) + direction(a, b))$



Lightcuts [Walter et al., ACM Trans. Graph. '05]

Preprocess

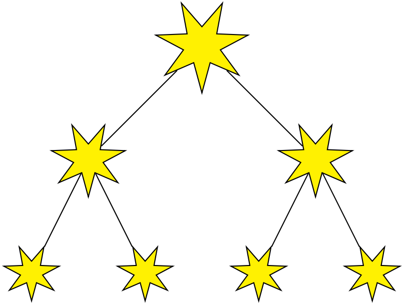
- Single VPLs
- Agglomerative clustering
  $sim(a, b) =$
  $I(dist(a, b) + direction(a, b))$



Lightcuts [Walter et al., ACM Trans. Graph. '05]

Preprocess

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



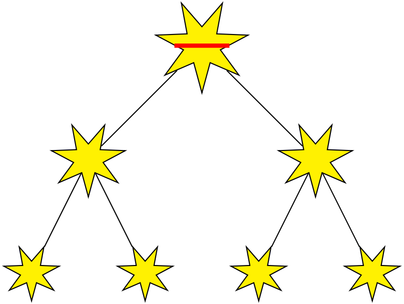
Lightcuts [Walter et al., ACM Trans. Graph. '05]

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)



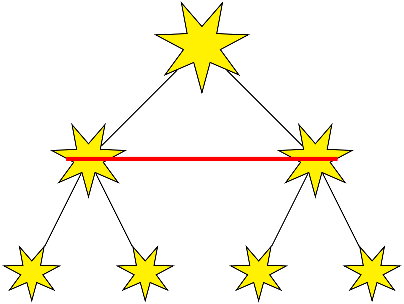
Lightcuts [Walter et al., ACM Trans. Graph. '05]

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)



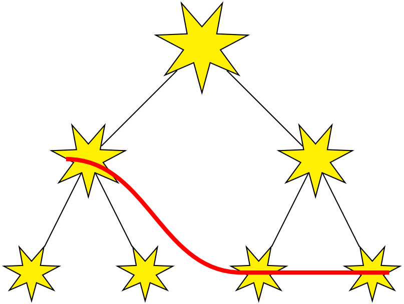
Lightcuts [Walter et al., ACM Trans. Graph. '05]

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)



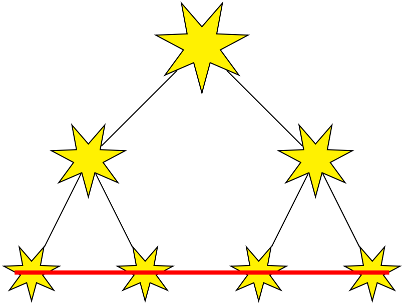
Lightcuts [Walter et al., ACM Trans. Graph. '05]

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

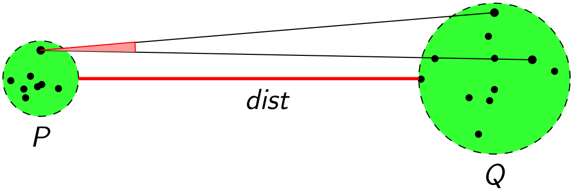
- Agglomerative clustering may be slow
- For each pixel we re-evaluate the optimal cut
- Too expensive
- Common cuts [Wang et al., SIGGRAPH Asia '11]
- Our intuition
- Structure that effectively captures all the different cuts

Well-separated pair

Definition

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

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



Bounds the angles and distances which is important for lights
This enables us to bound the error of light clustering

WSPD [Callahan et al.,JACM '95]

Definition

A well-separated pair decomposition (WSPD) of a point set, P is a set of pairs

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

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

WSPD

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.

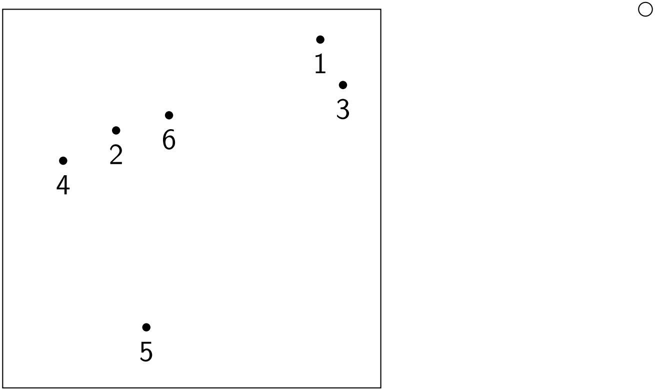
WSPD

Theorem

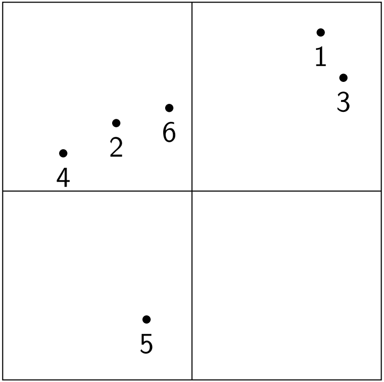
For $\epsilon > 0$, $P \subset \mathbb{R}^d$, where $|P| = n$ there exists a WSPD of size $O(n\epsilon^{-d})$ and one can compute it in $O(n \log n + n\epsilon^{-d})$ time.

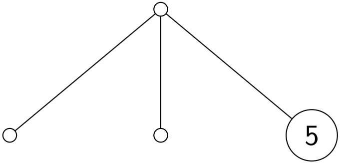
- Size of the WSPD is the number of pairs
- Build a compressed quadtree, $O(n \log n)$
- Recurse down from the root to find well separated pairs, $O(n\epsilon^{-d})$

WSPD algorithm

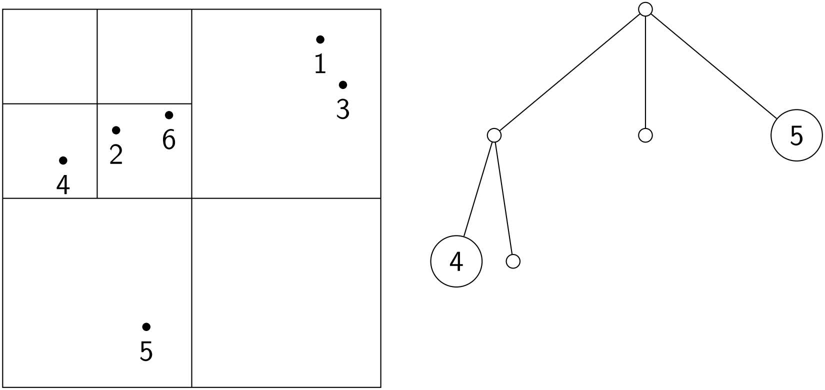


WSPD algorithm

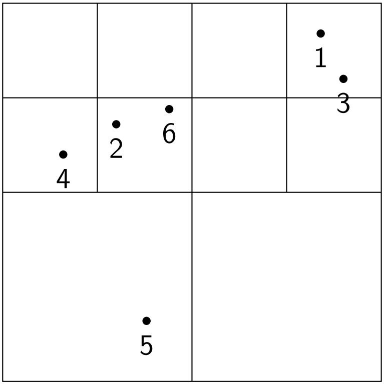


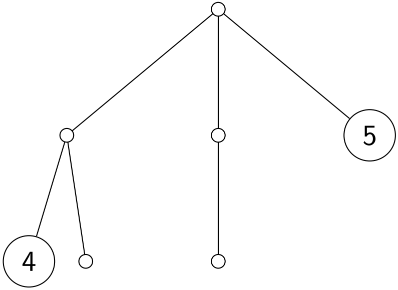


WSPD algorithm

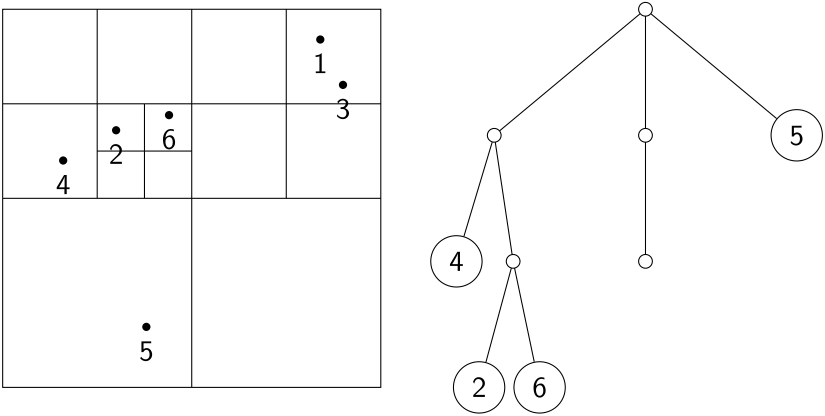


WSPD algorithm

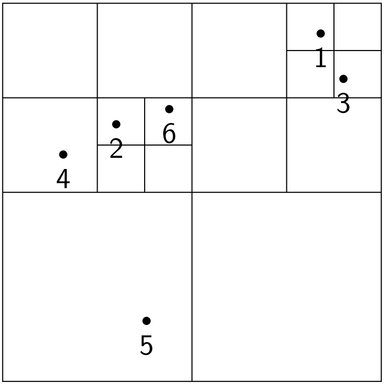


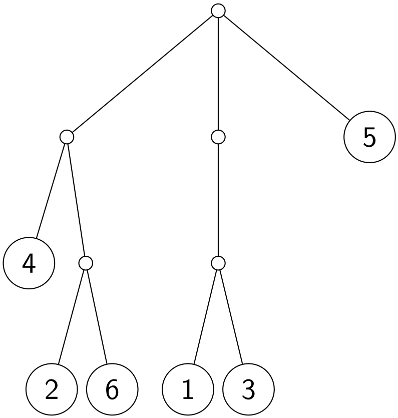


WSPD algorithm

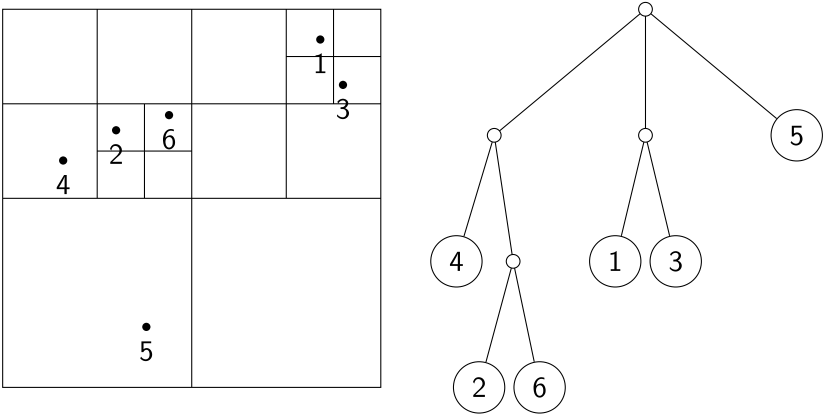


WSPD algorithm

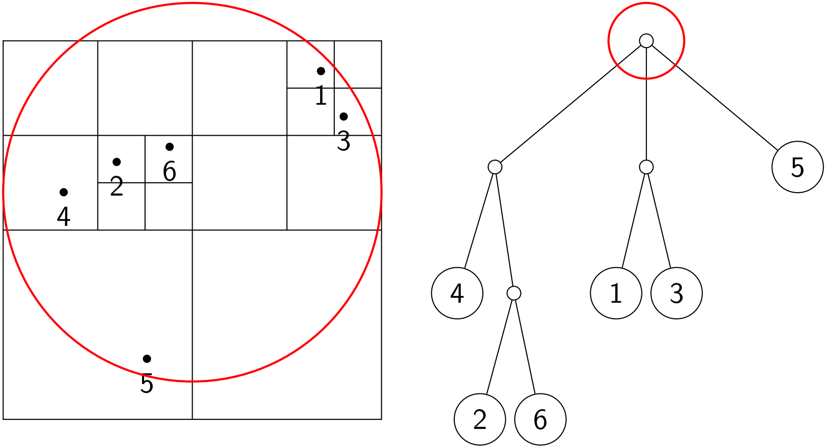




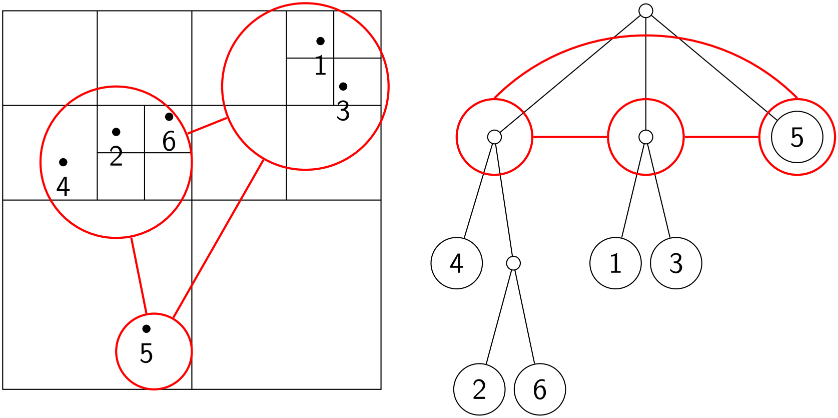
WSPD algorithm



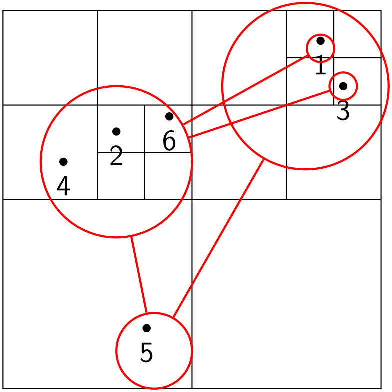
WSPD algorithm

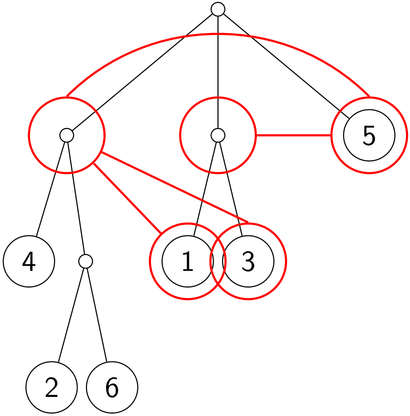


WSPD algorithm

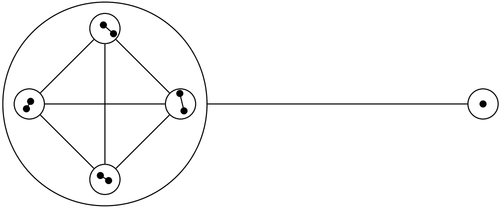


WSPD algorithm





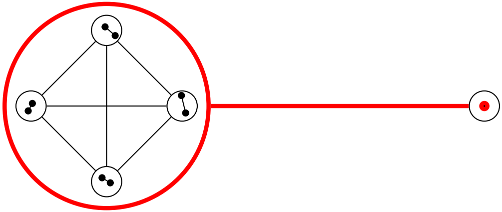
WSPD usage



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

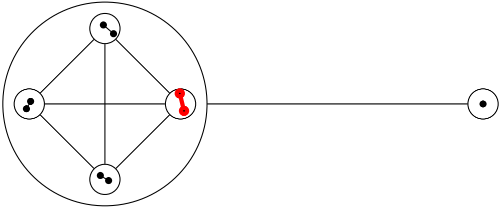
WSPD usage



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

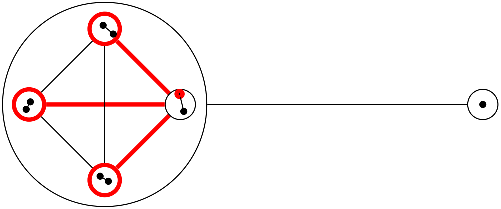
WSPD usage



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

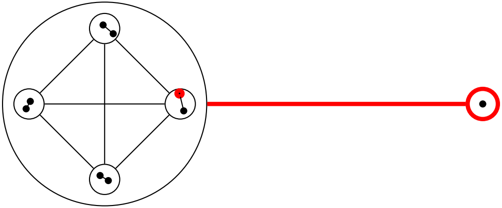
WSPD usage



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

WSPD usage



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

Our WSPD application

Use WSPD on VPLs but three questions arise

Our WSPD application

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

Our WSPD application

Use WSPD on VPLs but three questions arise

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

Our WSPD application

Use WSPD on VPLs but three questions arise

- Clustering with respect to non VPL points?
  -query WSPD
- Speed?
  -theoretical: $O(\epsilon^{-d} \log n)$

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

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 $\rightarrow$ 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

$\Big\} O(n \log n)$

Render a pixelpoint, q

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

Query the WSPD

Query for the point q

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 $\lambda = dist(p, q)$

Query the WSPD

Query for the point q

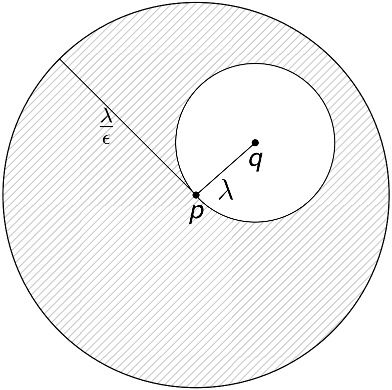
- Take the closest point to q, denote it by p
- Let $\lambda = dist(p, q)$
- Return the pairs, (B_i), of p such that $dist(p, B_i) > \frac{\lambda}{\epsilon}$

Query the WSPD

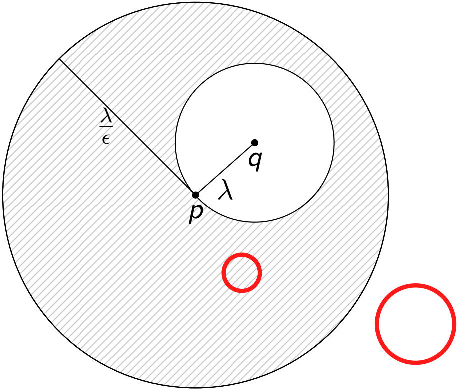
Query for the point q

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

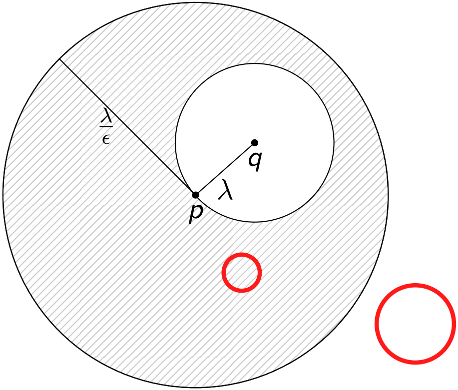
Correctness



Correctness



Correctness



Lemma I

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

Complexity - query

Complexity of the query phase

- Nearest neighbor

Complexity - query

Complexity of the query phase

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

Approximate nearest neighbor

For any $p \in \mathrm{R}^d$ 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 $p \in \mathrm{R}^d$ 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 λ 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 $p \in \mathrm{R}^d$ 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 λ 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 $p \in \mathrm{R}^d$ 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 λ is the distance form the nearest neighbor and Lemma I holds in this case too.

Results

On average we have reached about $4x$ 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 $\rightarrow$ 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
ACM Trans. Graph., 31(4):59:159:11, July '12.

G. Wang et al
Efficient search of lightcuts by spatial clustering,
*S*IGGRAPH Asia '11 Sketches, December '11

- Brian Taylor
  Image on slide 2

- Wikipedia
  Image on slide 3,4