Silhouette Extraction in Hough Space

Matt Olson and Hao Zhang

gruvi graphics + usability + visualization
School of Computer Science
Simon Fraser University

Eurographics 2006
1 Silhouette Extraction
   - Defining the Problem
   - Previous Work

2 Silhouettes in Hough Space
   - A Short Intuition of Hough Space
   - Updating Silhouettes in Hough Space
   - Finding Initial Silhouettes in Hough Space
   - Comparing Hough space to Dual space

3 Results
   - Static Silhouette Extraction Results
   - Dynamic Silhouette Update Results
   - Tree Depth Histograms

4 Conclusions
Outline

1 Silhouette Extraction
   • Defining the Problem
   • Previous Work

2 Silhouettes in Hough Space
   • A Short Intuition of Hough Space
   • Updating Silhouettes in Hough Space
   • Finding Initial Silhouettes in Hough Space
   • Comparing Hough space to Dual space

3 Results
   • Static Silhouette Extraction Results
   • Dynamic Silhouette Update Results
   • Tree Depth Histograms

4 Conclusions
Silhouettes on Meshes

- Silhouette edges separate front- and back-facing regions
- Applications in NPR, illumination, games, vision, etc.
- Silhouettes can be $O(n)$ in size [Alt et al. 2003], but usually $O(\sqrt{n})$ [Benichou and Elber 1999]
Silhouettes on Meshes

- Silhouette edges separate front- and back-facing regions
- Applications in NPR, illumination, games, vision, etc.
- Silhouettes can be $O(n)$ in size [Alt et al. 2003], but usually $O(\sqrt{n})$ [Benichou and Elber 1999]
Silhouettes on Meshes

- Silhouette edges separate front- and back-facing regions
- Applications in NPR, illumination, games, vision, etc.
- Silhouettes can be $O(n)$ in size [Alt et al. 2003], but usually $O(\sqrt{n})$ [Benichou and Elber 1999]
Silhouette is usually small compared to mesh size
- Change in silhouette over a small viewpoint movement is usually smaller still
- We may save a lot of computation with clever, output-sensitive algorithms
- [Benichou and Elber 1999] gave proven output-sensitive object-space algorithm for parallel projection
- Provable output-sensitivity in perspective projection (object-space) is still an open problem
Silhouette Extraction in Hough Space

Output-Sensitivity

- Silhouette is usually small compared to mesh size
- Change in silhouette over a small viewpoint movement is usually smaller still
- We may save a lot of computation with clever, output-sensitive algorithms
- [Benichou and Elber 1999] gave proven output-sensitive object-space algorithm for parallel projection
- Provable output-sensitivity in perspective projection (object-space) is still an open problem
Output-Sensitivity

- Silhouette is usually small compared to mesh size
- Change in silhouette over a small viewpoint movement is usually smaller still
- We may save a lot of computation with clever, output-sensitive algorithms
  - [Benichou and Elber 1999] gave proven output-sensitive object-space algorithm for parallel projection
  - Provable output-sensitivity in perspective projection (object-space) is still an open problem
Silhouette is usually small compared to mesh size
Change in silhouette over a small viewpoint movement is usually smaller still
We may save a lot of computation with clever, output-sensitive algorithms
[Benichou and Elber 1999] gave proven output-sensitive object-space algorithm for parallel projection
Provable output-sensitivity in perspective projection (object-space) is still an open problem
Output-Sensitivity

- Silhouette is usually small compared to mesh size
- Change in silhouette over a small viewpoint movement is usually smaller still
- We may save a lot of computation with clever, output-sensitive algorithms
- [Benichou and Elber 1999] gave proven output-sensitive object-space algorithm for parallel projection
- Provable output-sensitivity in perspective projection (object-space) is still an open problem
Image-space vs. Object-space

Overviews: [Gooch et al. 1999] and [Isenberg et al. 2003]

Two main approaches: image-space and object-space

- **Image-space:**
  - Find silhouettes in rendered image (e.g. depth buffer discontinuities)
  - Generally faster, GPU-friendly
  - Limited to image precision
  - Occluded silhouette edges not found

- **Object-space:**
  - Find silhouettes directly on object
  - Generally slower, not GPU-friendly
  - As precise as input mesh, machine precision allow
  - All silhouette edges found
Overviews: [Gooch et al. 1999] and [Isenberg et al. 2003]
Two main approaches: image-space and object-space

- **Image-space:**
  - Find silhouettes in rendered image (e.g. depth buffer discontinuities)
  - Generally faster, GPU-friendly
  - Limited to image precision
  - Occluded silhouette edges not found

- **Object-space:**
  - Find silhouettes directly on object
  - Generally slower, not GPU-friendly
  - As precise as input mesh, machine precision allow
  - All silhouette edges found
Image-space vs. Object-space

Overviews: [Gooch et al. 1999] and [Isenberg et al. 2003]
Two main approaches: image-space and object-space

- **Image-space:**
  - Find silhouettes in rendered image (e.g. depth buffer discontinuities)
  - Generally faster, GPU-friendly
  - Limited to image precision
  - Occluded silhouette edges not found

- **Object-space:**
  - Find silhouettes directly on object
  - Generally slower, not GPU-friendly
  - As precise as input mesh, machine precision allow
  - All silhouette edges found
Two main approaches: image-space and object-space

**Image-space:**
- Find silhouettes in rendered image (e.g. depth buffer discontinuities)
- Generally faster, GPU-friendly
- Limited to image precision
- Occluded silhouette edges not found

**Object-space:**
- Find silhouettes directly on object
- Generally slower, not GPU-friendly
- As precise as input mesh, machine precision allow
- All silhouette edges found
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Output sensitive?</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute force</td>
<td>No</td>
<td>3 tests per face</td>
</tr>
<tr>
<td>[Buchanan and Sousa 2000]</td>
<td>No</td>
<td>1 test per face</td>
</tr>
<tr>
<td>[Sander et al. 2000]</td>
<td>Empirically</td>
<td>Static only, complex</td>
</tr>
<tr>
<td>[Hertzmann and Zorin 2000]</td>
<td>Empirically</td>
<td>Static only</td>
</tr>
<tr>
<td>[Pop et al. 2001]</td>
<td>“Efficient”</td>
<td>Incremental only, origin issues</td>
</tr>
<tr>
<td>[Benichou and Elber 1999]</td>
<td>Yes</td>
<td>Orthographic only</td>
</tr>
</tbody>
</table>
Silhouette Extraction

Defining the Problem
Previous Work

Silhouettes in Hough Space

A Short Intuition of Hough Space
Updating Silhouettes in Hough Space
Finding Initial Silhouettes in Hough Space
Comparing Hough space to Dual space

Results

Static Silhouette Extraction Results
Dynamic Silhouette Update Results
Tree Depth Histograms

Conclusions
Our approach:

- Map planes to points in *Hough space* – similar to dual space, but better-behaved – and put them in an octree
- Construct initial silhouette efficiently by characterizing edges relative to the origin and fast culling of unrelated points
- Find Hough points crossed by transformed viewpoint; test only corresponding faces
- Traverse the octree efficiently using neighbour graph
Our approach:

- Map planes to points in *Hough space* – similar to dual space, but better-behaved – and put them in an octree
- Construct initial silhouette efficiently by characterizing edges relative to the origin and fast culling of unrelated points
- Find Hough points crossed by transformed viewpoint; test only corresponding faces
- Traverse the octree efficiently using neighbour graph
Our approach:

- Map planes to points in *Hough space* – similar to dual space, but better-behaved – and put them in an octree.
- Construct initial silhouette efficiently by characterizing edges relative to the origin and fast culling of unrelated points.
- Find Hough points crossed by transformed viewpoint; test only corresponding faces.
- Traverse the octree efficiently using neighbour graph.
Our approach:

- Map planes to points in *Hough space* – similar to dual space, but better-behaved – and put them in an octree
- Construct initial silhouette efficiently by characterizing edges relative to the origin and fast culling of unrelated points
- Find Hough points crossed by transformed viewpoint; test only corresponding faces
- Traverse the octree efficiently using neighbour graph
Contributions

- Hough transform offers advantages over dual transform
  - Efficient (empirically output-sensitive) calculation of both static silhouettes and changes to silhouette
  - Static silhouette extraction and incremental update on the same data structure
  - Novel neighbour graph improves efficiency further
Contributions

- Hough transform offers advantages over dual transform
- Efficient (empirically output-sensitive) calculation of both static silhouettes and changes to silhouette
  - Static silhouette extraction and incremental update on the same data structure
  - Novel neighbour graph improves efficiency further
Contributions

- Hough transform offers advantages over dual transform
- Efficient (empirically output-sensitive) calculation of both static silhouettes and changes to silhouette
- Static silhouette extraction and incremental update on the same data structure
- Novel neighbour graph improves efficiency further
Contributions

- Hough transform offers advantages over dual transform
- Efficient (empirically output-sensitive) calculation of both static silhouettes and changes to silhouette
- Static silhouette extraction and incremental update on the same data structure
- Novel neighbour graph improves efficiency further
Silhouettes in Object Space

- Silhouette edges separate front- and back-facing triangles
- It’s easier to work with tris’ supporting planes
- The *Hough transform of a plane* $P$ is the point on $P$ closest to the origin
Silhouettes in Object Space

- Silhouette edges separate front- and back-facing triangles
- It’s easier to work with tris’ supporting planes
- The Hough transform of a plane $P$ is the point on $P$ closest to the origin
Silhouettes in Object Space

- Silhouette edges separate front- and back-facing triangles
- It’s easier to work with tris’ supporting planes
- The Hough transform of a plane $P$ is the point on $P$ closest to the origin
Perspective projection: facing determined relative to viewpoint (not view vector)

- The *Hough transform of a point* $p$: sphere with poles at $p$ and origin
- (If $p$ is the viewpoint, we call this the *v-sphere*)
- Every point on the v-sphere is the Hough transform of a plane passing through the viewpoint
Viewpoints in Hough Space

- Perspective projection: facing determined relative to viewpoint (not view vector)
- The *Hough transform of a point* $p$: sphere with poles at $p$ and origin
- (If $p$ is the viewpoint, we call this the *v-sphere*)
- Every point on the v-sphere is the Hough transform of a plane passing through the viewpoint
Perspective projection: facing determined relative to viewpoint (not view vector)

The *Hough transform of a point* $p$: sphere with poles at $p$ and origin

(If $p$ is the viewpoint, we call this the *v-sphere*)

Every point on the v-sphere is the Hough transform of a plane passing through the viewpoint
The Hough transform does not preserve plane orientation.

We can’t tell whether a plane is front- or back-facing from its Hough point.

But when the v-sphere crosses a plane’s Hough point, we know it has changed facing.

We want to know which faces the v-sphere crosses.
The Hough transform does not preserve plane orientation.

We can’t tell whether a plane is front- or back-facing from its Hough point.

But when the v-sphere crosses a plane’s Hough point, we know it has changed facing.

We want to know which faces the v-sphere crosses.
Assume the viewpoint moves smoothly

Suppose we have the silhouette at frame $t$

Then we can find the silhouette at $t + 1$ by finding only the edges that have changed

The only edges that change lie on faces that change facing

Same basic approach as [Pop et al. 2001]
Assume the viewpoint moves smoothly
Suppose we have the silhouette at frame $t$
Then we can find the silhouette at $t + 1$ by finding only the edges that have changed
The only edges that change lie on faces that change facing
Same basic approach as [Pop et al. 2001]
Assume the viewpoint moves smoothly.
Suppose we have the silhouette at frame $t$
Then we can find the silhouette at $t+1$ by finding only the edges that have changed.
The only edges that change lie on faces that change facing.
Same basic approach as [Pop et al. 2001]
Assume the viewpoint moves smoothly
Suppose we have the silhouette at frame \( t \)
Then we can find the silhouette at \( t + 1 \) by finding only the edges that have changed
The only edges that change lie on faces that change facing
Same basic approach as [Pop et al. 2001]
Assume the viewpoint moves smoothly
Suppose we have the silhouette at frame $t$
Then we can find the silhouette at $t + 1$ by finding only the edges that have changed
The only edges that change lie on faces that change facing
Same basic approach as [Pop et al. 2001]
Recall that faces change facing when the v-sphere crosses their Hough points.

- We call the space between two v-spheres the *active region*.
- Find all points in this region and check their edges.
- Use octree to accelerate search.
Recall that faces change facing when the v-sphere crosses their Hough points.

We call the space between two v-spheres the *active region*.

Find all points in this region and check their edges.

Use octree to accelerate search.
Check Every Point Between Two V-Spheres

- Recall that faces change facing when the v-sphere crosses their Hough points.
- We call the space between two v-spheres the *active region*.
- Find all points in this region and check their edges.
- Use octree to accelerate search.
We already know where to start looking in the octree (from last frame’s results)

Why bother starting from the root all the time?

Use *neighbour graph* to walk along tree near leaves

Neighbour graph: links nodes to adjacent nodes of equal or lesser depth (see paper for details)
We already know where to start looking in the octree (from last frame’s results)

Why bother starting from the root all the time?

- Use *neighbour graph* to walk along tree near leaves
- Neighbour graph: links nodes to adjacent nodes of equal or lesser depth (see paper for details)
The Neighbour Graph

- We already know where to start looking in the octree (from last frame’s results)
- Why bother starting from the root all the time?
- Use *neighbour graph* to walk along tree near leaves
- Neighbour graph: links nodes to adjacent nodes of equal or lesser depth (see paper for details)
We already know where to start looking in the octree (from last frame’s results)

Why bother starting from the root all the time?

Use *neighbour graph* to walk along tree near leaves

Neighbour graph: links nodes to adjacent nodes of equal or lesser depth (see paper for details)
Begin with nodes intersecting last frame’s v-sphere

- Breadth-first search along neighbour graph, checking faces in nodes
- Until this frame’s v-sphere reached
Traversing The Active Region

- Begin with nodes intersecting last frame’s v-sphere
- Breadth-first search along neighbour graph, checking faces in nodes
- Until this frame’s v-sphere reached
Traversing The Active Region

- Begin with nodes intersecting last frame’s v-sphere
- Breadth-first search along neighbour graph, checking faces in nodes
- Until this frame’s v-sphere reached
Lines and Edges in Hough Space

- Line $L$ maps to circle through origin and projection of origin onto $L$.
- Geometrically, every point on the circle is the Hough point of a plane going through $L$.
- An edge transforms to an arc on its line’s circle.
Lines and Edges in Hough Space

- Line $L$ maps to circle through origin and projection of origin onto $L$.
- Geometrically, every point on the circle is the Hough point of a plane going through $L$.
- An edge transforms to an arc on its line’s circle.
Lines and Edges in Hough Space

- Line $L$ maps to circle through origin and projection of origin onto $L$.
- Geometrically, every point on the circle is the Hough point of a plane going through $L$.
- An edge transforms to an arc on its line’s circle.
To find a mesh’s silhouette from an arbitrary viewpoint, we:

- Check edges silhouetted from the origin
- Check edges crossing the v-sphere
- (Use an augmented octree to accelerate)
- Both edge sets are about $O(\sqrt{n})$ in size
To find a mesh’s silhouette from an arbitrary viewpoint, we:

- Check edges silhouetted from the origin
- Check edges crossing the v-sphere
- (Use an augmented octree to accelerate)
- Both edge sets are about $O(\sqrt{n})$ in size
To find a mesh’s silhouette from an arbitrary viewpoint, we:

- Check edges silhouetted from the origin
- Check edges crossing the v-sphere
- (Use an augmented octree to accelerate)
- Both edge sets are about $O(\sqrt{n})$ in size
In general, a silhouette edge might not cross the v-sphere. But we know that it must if the edge is not on the silhouette from the origin. (See paper for proof) We can precalculate silhouette edges from origin, and use an octree to speed checking the rest.
In general, a silhouette edge might not cross the v-sphere.

But we know that it must if the edge is not on the silhouette from the origin.

(See paper for proof)

We can precalculate silhouette edges from origin, and use an octree to speed checking the rest.
In general, a silhouette edge might not cross the v-sphere.
But we know that it must if the edge is not on the silhouette from the origin.
(See paper for proof)
We can precalculate silhouette edges from origin, and use an octree to speed checking the rest.
We build an octree around the Hough points of the input mesh.

- Each octree cell has a spatial extent
- A minimal bounding box for every point in its subtree
- And another bounding box containing the points of adjacent faces
We build an octree around the Hough points of the input mesh.

- Each octree cell has a spatial extent
- A minimal bounding box for every point in its subtree
- And another bounding box containing the points of adjacent faces
The Augmented Octree

We build an octree around the Hough points of the input mesh.

- Each octree cell has a spatial extent
- A minimal bounding box for every point in its subtree
- And another bounding box containing the points of adjacent faces
### Hough space vs. Dual space

**Dual transform is well-known; why bother switching?**

**Dual space:**
- Planes through origin map to points at infinity
- Dual points cluster around origin
- Points, lines map to infinite structures
- Dual transform of arbitrary mesh can extend to infinity

**Hough space:**
- Planes through origin map to points at origin
- Hough points more evenly distributed
- Points, lines map to finite structures
- Hough transform of arbitrary mesh contained in bounding sphere
Hough space vs. Dual space

Dual transform is well-known; why bother switching?

**Dual space:**
- Planes through origin map to points at infinity
- Dual points cluster around origin
- Points, lines map to infinite structures
- Dual transform of arbitrary mesh can extend to infinity

**Hough space:**
- Planes through origin map to points at origin
- Hough points more evenly distributed
- Points, lines map to finite structures
- Hough transform of arbitrary mesh contained in bounding sphere
Dual transform is well-known; why bother switching?

**Dual space:**
- Planes through origin map to points at infinity
- Dual points cluster around origin
- Points, lines map to infinite structures
- Dual transform of arbitrary mesh can extend to infinity

**Hough space:**
- Planes through origin map to points at origin
- Hough points more evenly distributed
- Points, lines map to finite structures
- Hough transform of arbitrary mesh contained in bounding sphere
Dual transform is well-known; why bother switching?

**Dual space:**
- Planes through origin map to points at infinity
- Dual points cluster around origin
- Points, lines map to infinite structures
- Dual transform of arbitrary mesh can extend to infinity

**Hough space:**
- Planes through origin map to points at origin
- Hough points more evenly distributed
- Points, lines map to finite structures
- Hough transform of arbitrary mesh contained in bounding sphere
## Dual transform is well-known; why bother switching?

### Dual space:
- Planes through origin map to points at infinity
- Dual points cluster around origin
- Points, lines map to infinite structures
- Dual transform of arbitrary mesh can extend to infinity

### Hough space:
- Planes through origin map to points at origin
- Hough points more evenly distributed
- Points, lines map to finite structures
- Hough transform of arbitrary mesh contained in bounding sphere
Comparison of Point Distributions

Dual and Hough transforms of the hand mesh:

Dual-space:

Hough-space:
1. Silhouette Extraction
   - Defining the Problem
   - Previous Work

2. Silhouettes in Hough Space
   - A Short Intuition of Hough Space
   - Updating Silhouettes in Hough Space
   - Finding Initial Silhouettes in Hough Space
   - Comparing Hough space to Dual space

3. Results
   - Static Silhouette Extraction Results
   - Dynamic Silhouette Update Results
   - Tree Depth Histograms

4. Conclusions
We tested six models in the paper
Only two are shown here, for clarity
Static Extraction: Bunny

x $10^4$

Bunny

- Red: Mesh model size
- Blue: No. of bounding box checks

Average silhouette size

500 1000 1500 2000 2500 3000 3500 4000

$7 \times 10^4$
Silhouette Extraction
Silhouettes in Hough Space
Results
Conclusions

Static Silhouette Extraction Results
Dynamic Silhouette Update Results
Tree Depth Histograms

Static Extraction: Dragon

![Graph showing mesh model size and number of bounding box checks vs average silhouette size]

- **Red**: Mesh model size
- **Blue**: No. of bounding box checks

Matt Olson and Hao Zhang
Silhouette Extraction in Hough Space
Silhouette Extraction

Silhouettes in Hough Space

Results

Conclusions

Static Silhouette Extraction Results

Dynamic Silhouette Update Results

Tree Depth Histograms

Silhouette Update: Bunny

Bunny

Bounding box tests

Angles in a 360-degree rotating view

Hough (nbr)

Hough (full)

Dual
Silhouette Extraction in Hough Space

Silhouette Update: Dragon

- **Silhouette Update: Dragon 29**
  - Matt Olson and Hao Zhang

**Figure:**

- **Graph:**
  - **X-axis:** Angles in a 360-degree rotating view
  - **Y-axis:** Bounding box tests
  - **Legend:**
    - Hough (nbr)
    - Hough (full)
    - Dual

**Title:** Dragon

**Subtitle:**

- Static Silhouette Extraction Results
- Dynamic Silhouette Update Results
- Tree Depth Histograms
- Conclusions
Silhouette Extraction in Hough Space

Static Silhouette Extraction Results
Dynamic Silhouette Update Results
Tree Depth Histograms

Tree Depths: Dragon

Dragon

Leaf nodes per bucket vs. Leaf depth buckets for Dragon.

- **Hough**
- **Dual**

Tree Depths: Dragon

Matt Olson and Hao Zhang

Silhouette Extraction in Hough Space
1 Silhouette Extraction
   - Defining the Problem
   - Previous Work

2 Silhouettes in Hough Space
   - A Short Intuition of Hough Space
   - Updating Silhouettes in Hough Space
   - Finding Initial Silhouettes in Hough Space
   - Comparing Hough space to Dual space

3 Results
   - Static Silhouette Extraction Results
   - Dynamic Silhouette Update Results
   - Tree Depth Histograms

4 Conclusions
The 3D Hough transform is efficient and powerful.

- We can use the Hough transform of a mesh to quickly extract and update its perspective silhouette.
- Traversing the tree with the neighbour graph lets us exploit frame coherence for a noticeable performance improvement.
The 3D Hough transform is efficient and powerful

- We can use the Hough transform of a mesh to quickly extract and update its perspective silhouette
- Traversing the tree with the neighbour graph lets us exploit frame coherence for a noticeable performance improvement
Conclusions

- The 3D Hough transform is efficient and powerful
- We can use the Hough transform of a mesh to quickly extract and update its perspective silhouette
- Traversing the tree with the neighbour graph lets us exploit frame coherence for a noticeable performance improvement
Limitations

- Building octree is a required preprocessing step
  - Not very expensive, but not necessarily real-time
  - Static meshes only; deformable and dynamically generated meshes are hard
- Can’t easily distinguish points at origin
  - All planes through the origin map to the same point
- Whole octree needed in memory
  - Processing extremely large meshes becomes difficult
Limitations

- Building octree is a required preprocessing step
  - Not very expensive, but not necessarily real-time
  - Static meshes only; deformable and dynamically generated meshes are hard
- Can’t easily distinguish points at origin
  - All planes through the origin map to the same point
- Whole octree needed in memory
  - Processing extremely large meshes becomes difficult
Limitations

- Building octree is a required preprocessing step
  - Not very expensive, but not necessarily real-time
  - Static meshes only; deformable and dynamically generated meshes are hard
- Can’t easily distinguish points at origin
  - All planes through the origin map to the same point
- Whole octree needed in memory
  - Processing extremely large meshes becomes difficult
Future Work

- Hough space demands more study
- Improve tree balance
  - Tree balance clearly affects performance
  - Specialized data structures may give better balance
  - Mesh origin affects its Hough transform – select origins to give better-balanced trees
- Extend this approach to deal with animated or deforming meshes
Future Work

- Hough space demands more study
- Improve tree balance
  - Tree balance clearly affects performance
  - Specialized data structures may give better balance
  - Mesh origin affects its Hough transform – select origins to give better-balanced trees
- Extend this approach to deal with animated or deforming meshes
Future Work

- Hough space demands more study
- Improve tree balance
  - Tree balance clearly affects performance
  - Specialized data structures may give better balance
  - Mesh origin affects its Hough transform – select origins to give better-balanced trees
- Extend this approach to deal with animated or deforming meshes
Future Work

- Hough space demands more study
- Improve tree balance
  - Tree balance clearly affects performance
  - Specialized data structures may give better balance
  - Mesh origin affects its Hough transform – select origins to give better-balanced trees
- Extend this approach to deal with animated or deforming meshes
Future Work

- Hough space demands more study
- Improve tree balance
  - Tree balance clearly affects performance
  - Specialized data structures may give better balance
  - Mesh origin affects its Hough transform – select origins to give better-balanced trees
- Extend this approach to deal with animated or deforming meshes
Future Work

- Hough space demands more study
- Improve tree balance
  - Tree balance clearly affects performance
  - Specialized data structures may give better balance
  - Mesh origin affects its Hough transform – select origins to give better-balanced trees
- Extend this approach to deal with animated or deforming meshes
## Object-Space Algorithms, Revised

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Output sensitive?</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute force</td>
<td>No</td>
<td>3 tests per face</td>
</tr>
<tr>
<td>[Buchanan and Sousa 2000]</td>
<td>No</td>
<td>1 test per face</td>
</tr>
<tr>
<td>[Sander et al. 2000]</td>
<td>Empirically</td>
<td>Static only, complex</td>
</tr>
<tr>
<td>[Hertzmann and Zorin 2000]</td>
<td>Empirically</td>
<td>Static only</td>
</tr>
<tr>
<td>[Pop et al. 2001]</td>
<td>“Efficient”</td>
<td>Incremental only, origin issues</td>
</tr>
<tr>
<td>[Benichou and Elber 1999]</td>
<td>Yes</td>
<td>Orthographic only</td>
</tr>
<tr>
<td>[Olson and Zhang 2006]</td>
<td>Empirically</td>
<td>Static and incremental</td>
</tr>
</tbody>
</table>
Questions?