Shape from Silhouettes II

Guido Gerig
CS 6643, Spring 2017

(slides modified from Marc Pollefeys
UNC Chapel Hill, some of the figures and slides are
adapted from M. Pollefeys, J.S. Franco, J. Matusik’s
presentations, and referenced papers)
Shape from silhouettes

Automatic 3D Model Construction for Turn-Table Sequences, A.W. Fitzgibbon, G. Cross, and A. Zisserman, SMILE 1998

Slides from Lazebnik, Matusik Yerex and others
Outline

• Silhouettes
  – basic concepts
  – extract silhouettes
  – fundamentals about using silhouettes
  – reconstruct shapes from silhouettes
  – use uncertain silhouettes
  – calibrate from silhouettes

• Perspectives and interesting ideas
Algorithms

- Standard voxel based method
- Exact polyhedral methods
- Image-based visual hulls
IBVH: Image based visual hulls

- VH without constructing auxiliary geometric/volumetric representation
- All steps of rendering are computed in 2D “image space” coordinates (eliminates resampling/quantization artifacts)
- Reference images used as textures for shading the VH
- Image-based objects in real-time

Wojciech Matusik, *Image Based Visual Hulls*
Image based visual hulls

- This algorithm will only produce renderings of a visual hull from any view.
- Every pixel in the desired output image is back-projected to form a 3D ray.
- Each of those rays is intersected with each of the input silhouettes in the same way as the rays in the marching intersections method.
- Then a pixel in the output image is inside the new rendering of the visual hull if its ray has any segments left in it that are intersecting the visual hull. The depth of these pixels is known from the depth of the nearest entry point on the ray.

Wojciech Matusik, *Image Based Visual Hulls*
Image based - example
Image-Based Computation

1. Project 3D viewing ray (blue) into reference image
2. Intersection of projected ray with 2D silhouettes → list of intervals along ray interior to cone’s cross section
3. Intervals lifted back to 3D, intersected with results of other reference images

Figure 2 – Computing the IBVH involves three steps. First, the desired ray is projected onto a reference image. Next, the intervals where the projected ray crosses the silhouette are determined. Finally, these intervals are lifted back onto the desired ray where they can be intersected with intervals from other reference images.
Naïve Algorithm

\[
\text{IBVHIssect (intervalImage } \& \ d, \text{ refImList } R) \{
\text{for each referenceImage } r \text{ in } R \\
\text{computeSilhouetteEdges } (r) \\
\text{for each pixel } p \text{ in desiredImage } d \text{ do} \\
p.\text{intervals} = [0..\inf] \\
\text{for each referenceImage } r \text{ in } R \\
\text{for each scanline } s \text{ in } d \\
\text{for each pixel } p \text{ in } s \\
\text{ray3D } \text{ry3} = \text{compute3Dray}(p,d.\text{camInfo}) \\
\text{lineSegment2D } l2 = \text{project3Dray}(\text{ry3},r.\text{camInfo}) \\
\text{intervals int2D} = \text{calcIntervals}(l2,r.\text{silEdges}) \\
\text{intervals int3D} = \text{liftIntervals}(\text{int2D},r.\text{camInfo},\text{ry3}) \\
p.\text{intervals} = p.\text{intervals ISECT int3D} \\
\}
\]

Efficiency for desired view:

\[n: \#\text{pixels per scanline}\]
\[O(n^2): \#\text{pixels in image}\]
\[k: \#\text{images}\]
\[l: \text{average } \#\text{times ray intersects silhouette}\]
\[O(ln): \#\text{silhouette edges}\]
\[\text{(search along epip lines)}\]

\[O(lkn^3): \text{Efficiency}\]
Image-Based Computation

Remember Epipolar Geometry:

- Epipolar planes: Set of planes that share line OO’
- Epipolar planes project to lines in each image: epipolar lines
- All epipolar lines intersect in common point: epipole
**Image-Based Computation**

**Improved Algorithm:**
Take advantage of incremental computations enabled by epipolar geometry relating reference and desired images

**Efficiency:** $O(lkn^2)$

- Traverse scan line of desired view
- Each ray project to line through epipole in reference image
- Set of rays in desired image: pencil of epipolar lines in reference image
- Monotonic change of slope
- Compute silhouette intersections for whole scan line incrementally (trick: presorted silhouette edges relative to epipole)

**Complexity:** $O(lkn^2)$

---

Figure 3 – The pixels of a scanline in the desired image trace out a pencil of line segments in the reference image. An ordered traversal of the scanline will sweep out these segments such that their slope about the epipole varies monotonically.
Observation

- Incremental computation along scanlines
Shading Algorithm

- A view-dependent strategy
Shading Algorithm

A view-dependent strategy

- Use reference images as textures
- Rank reference-image textures as “best” to “worst” based on angle between desired viewing ray and rays of each reference image
- Consider visibility (avoid textures of points that are blocked): Take advantage of epipolar geometry for incremental procedure
Visibility Algorithm
Visibility in 2D

Reference view

Desired view
Visibility in 2D

Reference view

Desired view

Front-most Points
Visibility in 2D
Visibility in 2D

Coverage Mask

Reference view

Desired view
Visibility in 2D

Coverage Mask

Reference view

Desired view

Visible
Visibility in 2D

Coverage Mask

Reference view

Desired view

Visible
Visibility in 2D

Coverage Mask

Reference view

Not Visible

Desired view
Visibility in 2D

Coverage Mask

Reference view

Desired view
Visibility in 2D

Coverage Mask

Reference view

Desired view

Visible
Visibility in 2D

Coverage Mask

Reference view

Desired view
Visibility in 2D

Coverage Mask

Reference view

Not Visible

Desired view
System

- Camera Client
- Camera Client
- Camera Client
- Camera Client

Server (4x 500 Mhz)
System

Compressed video

Server (4x 500 Mhz)

Camera Client

Camera Client

Camera Client

Camera Client
System

Camera Client  Camera Client  Camera Client  Camera Client

Server (4x 500 Mhz)  Intersection
System

Camera Client
Camera Client
Camera Client
Camera Client

Server (4x 500 Mhz)

Shading
IBVH Results

- Approximately constant computation per pixel per camera
- Parallelizes
- Consistent with input silhouettes
IBVH Results

Figure 6 – Four segmented reference images from our system.

Figure 8 - Example IBVH images. The upper images show depth maps of the computed visual hulls. The lower images show shaded renderings from the same viewpoint. The hull segment connecting the two legs results from a segmentation error caused by a shadow.
Image Based Visual Hulls

http://www.youtube.com/watch?v=Lw9aFaHobao

See also: http://www.youtube.com/watch?v=UdmBW4kDcok
Outline

• Silhouettes
  – basic concepts
  – extract silhouettes
  – fundamentals about using silhouettes
  – reconstruct shapes from silhouettes
  – use uncertain silhouettes
  – calibrate from silhouettes

• Perspectives and cool ideas
Next Lecture

- Visual Hulls using uncalibrated camera views (Forbes et al.)
- Wrap-up