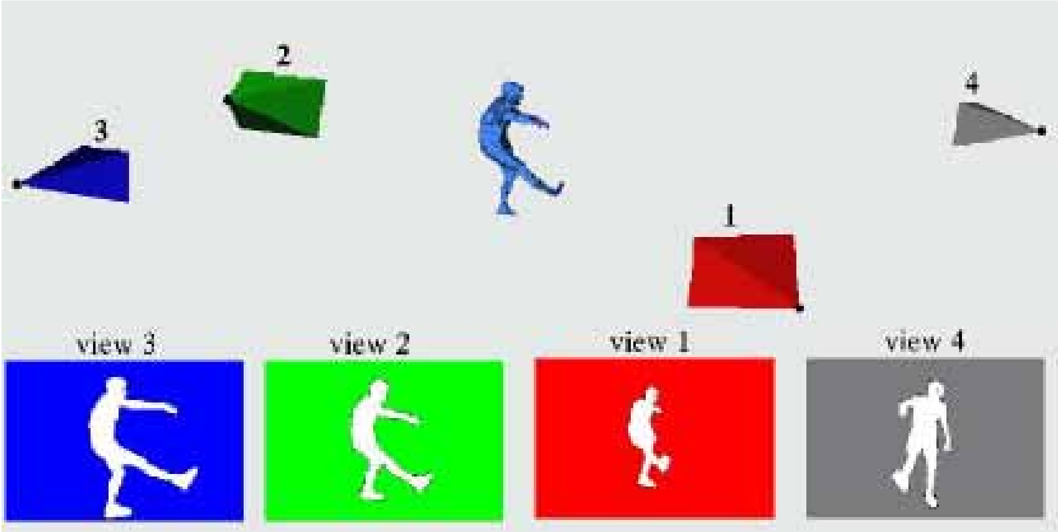


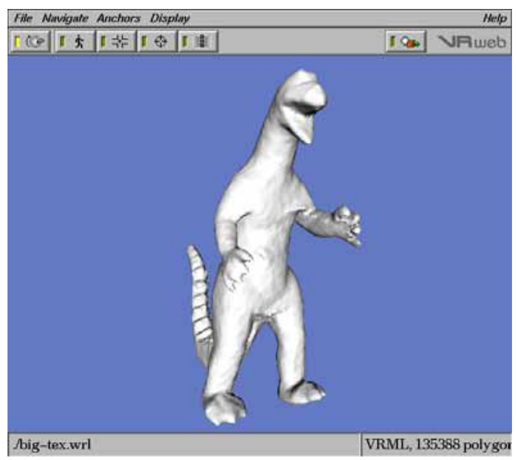
Shape from Silhouettes I

Guido Gerig

CS 6320, Spring 2015

Credits: Marc Pollefeys, UNC Chapel Hill, some of the figures and slides are also adapted from 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

Big Picture

- Multi-camera environments
- Dynamic scene

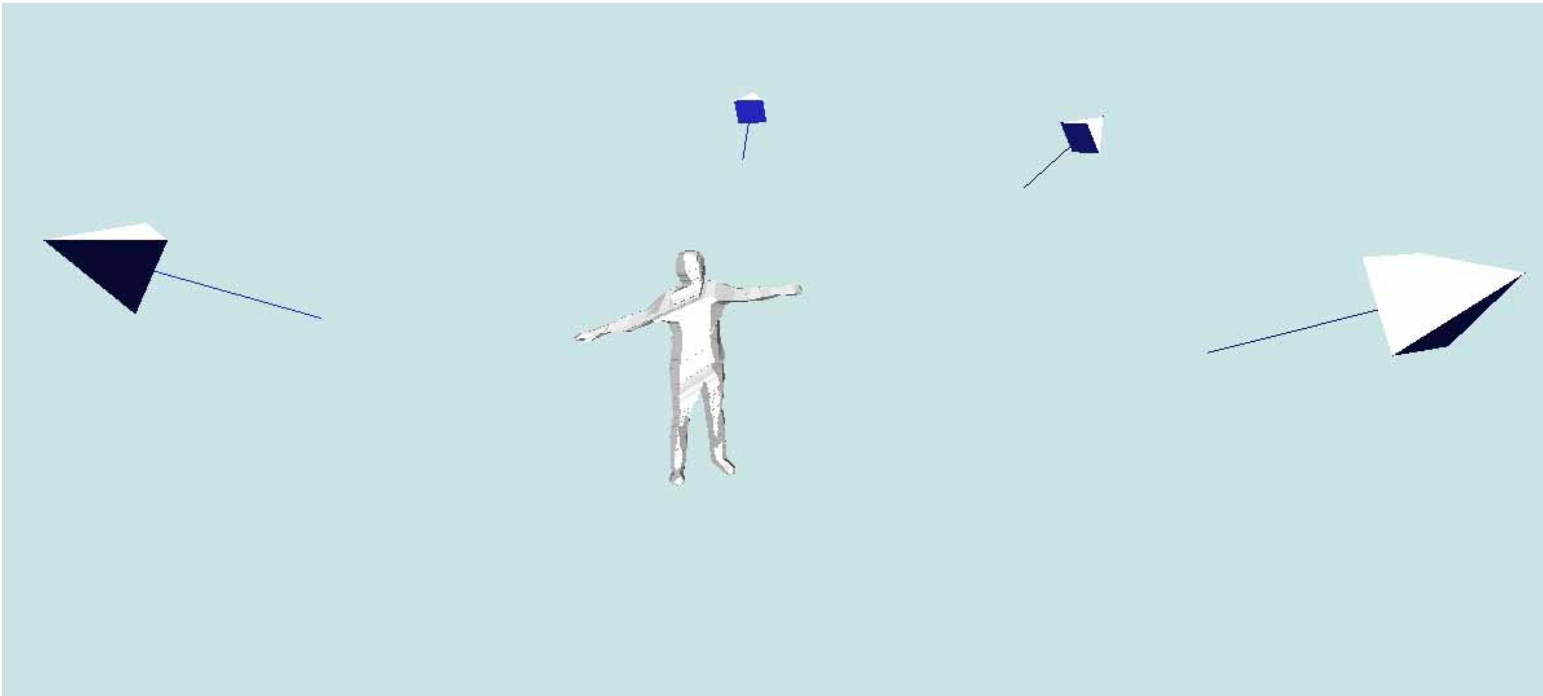
Outdoor data capturing with 9 video cameras behind the Ackland Museum, UNC-Chapel Hill, 2006/8/24. Pictured by Jae Hak Kim.

- N cameras observe the scene and produce N video streams
- What can we do with this data?

Motivation: Movies

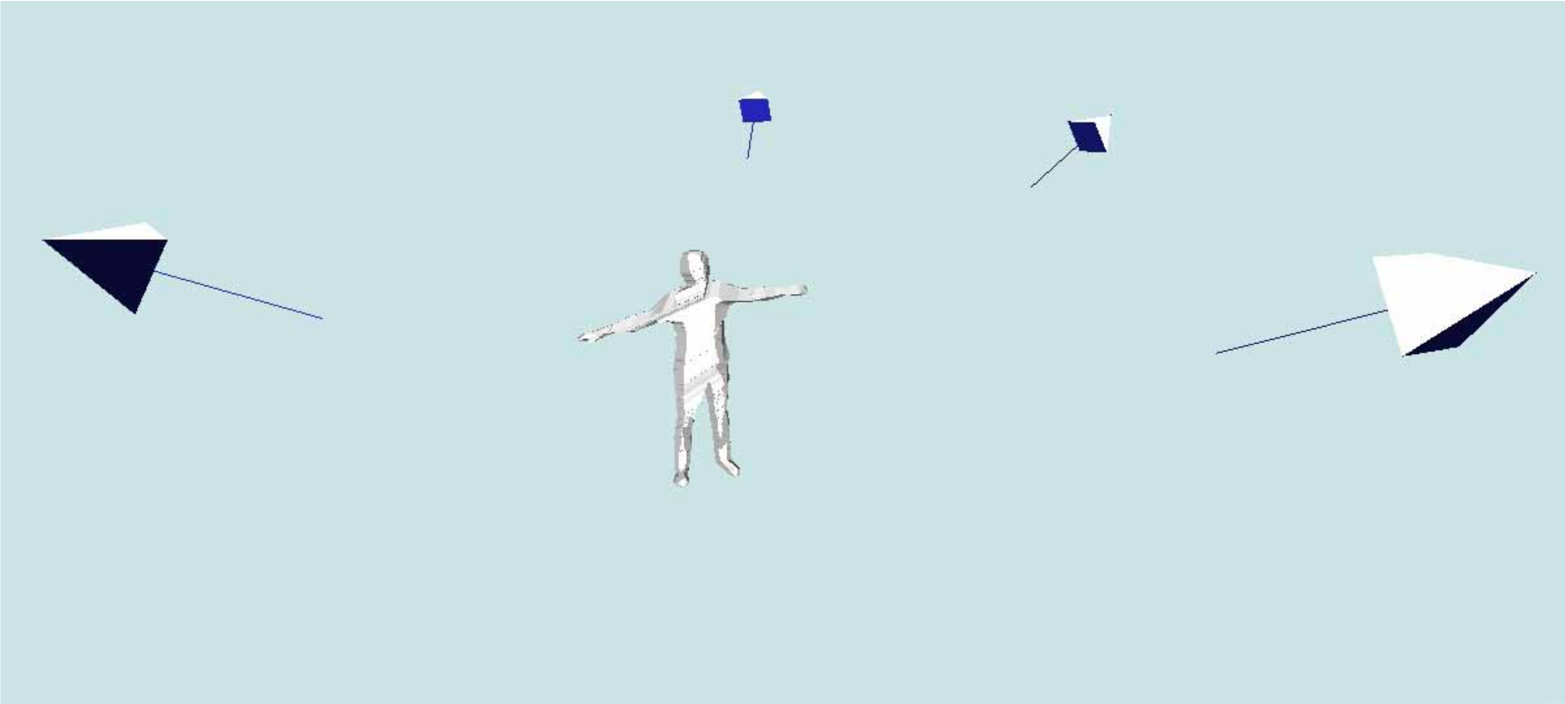
Sinha Sudipta, UNC PhD 2008

Motivation: 3D from Movies



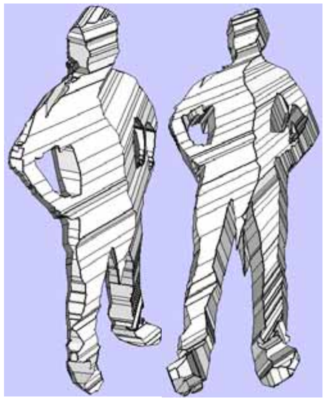
Sinha Sudipta, UNC PhD 2008

Motivation: 3D from Movies: Replay from arbitrary viewpoints

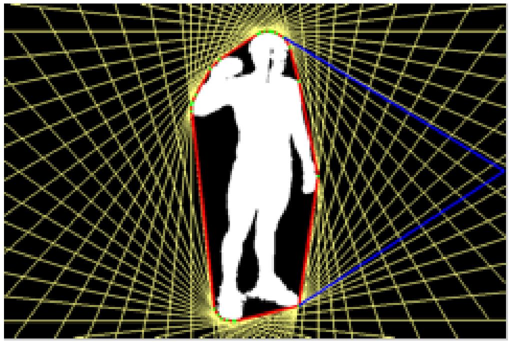


Sinha Sudipta, UNC PhD 2008

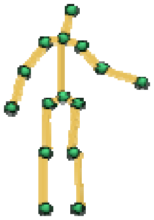
What can we do with this data?



- Reconstruct scene objects:
  - shape from silhouettes
  - photo-consistency



- Calibrate cameras
  - recover epipolar geometry

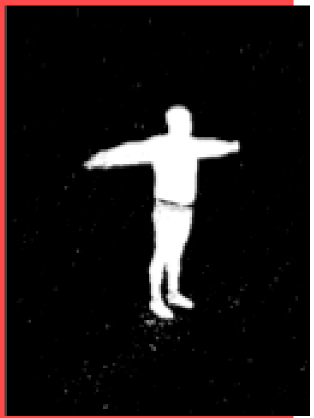


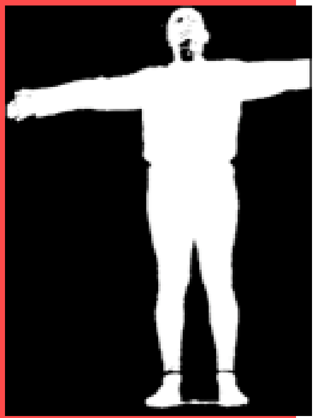
- Fit specific models (articulated models)

Outline

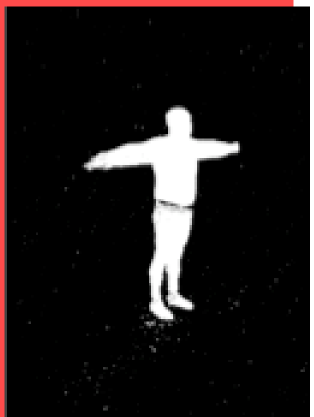
- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes
  - calibrate from silhouettes
- Perspectives and interesting ideas

Silhouettes of objects of interest





- Silhouettes are the regions where objects of interest project in images
- Silhouettes can generally be obtained using low level information (fast)
- They give information about the global shape of scene objects



How to extract silhouettes?

- Sometimes done manually (for offline applications, ground truth and verifications)
- Region based-extraction (automatic)
  - silhouette extraction is a 2-region image segmentation problem, w/ specific solutions:
    - chroma keying (blue, green background)
    - background subtraction (pre-observed static or dynamic background)

(refer to segmentation course)

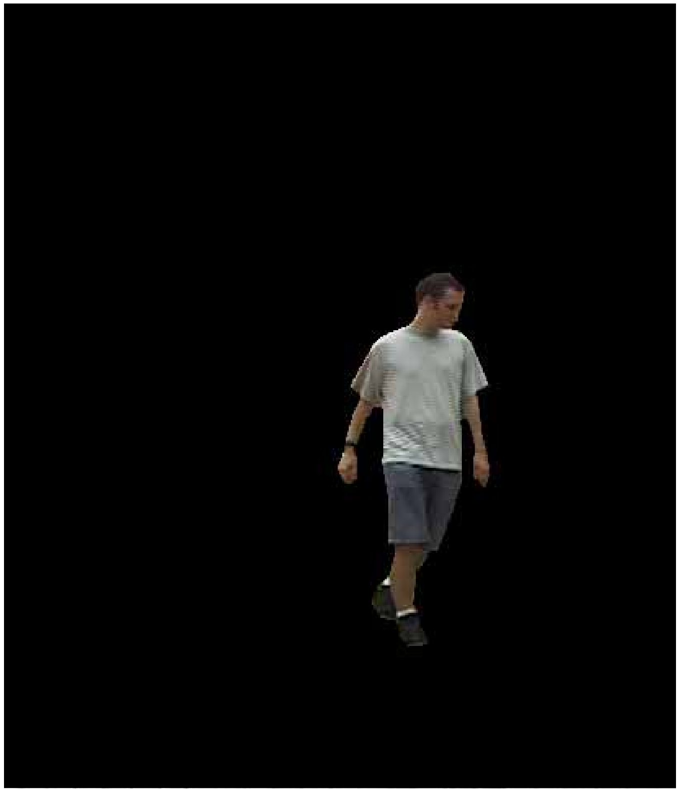
How to extract silhouettes?

- Contour-based extraction
- focus on silhouette *outline* instead of region itself
  - snakes, active contours: fitting of a curve to high gradients in image, local optimization

Yilmaz&Shah ACCV04

How to extract silhouettes? (cont.)

- Background subtraction

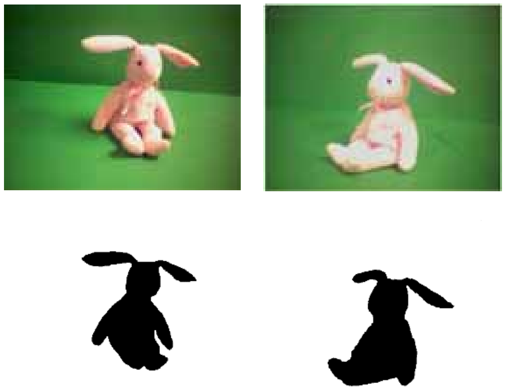


- Simple thresholding
- Train an appearance model for each pixel, from a set of background images
  - RGB 3D-Gaussian model
  - HSV model
  - GMM model
  - Non-parametric model (histogram/kernel density function)
- Apply the pixel color to the model, then classify it to be foreground/background

- We will talk about this in more detail later

Why use a Visual Hull?

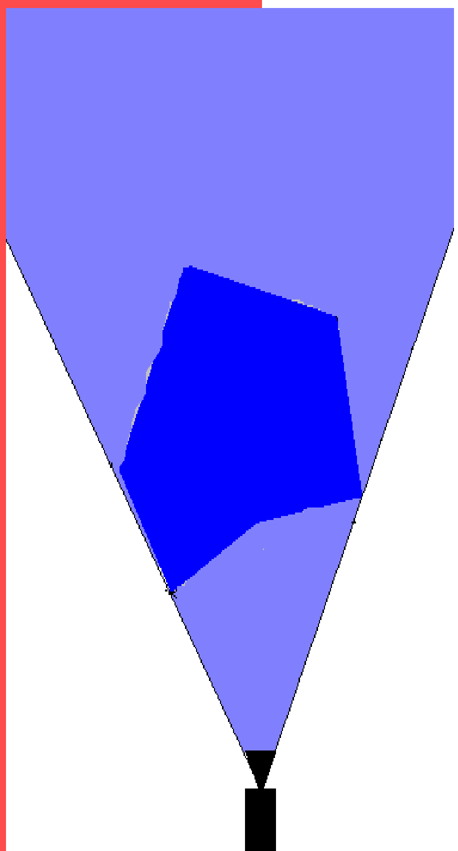
- Good shape representation
- Can be computed efficiently
- No photo-consistency required
- As bootstrap of many fancy refinement ...



Outline

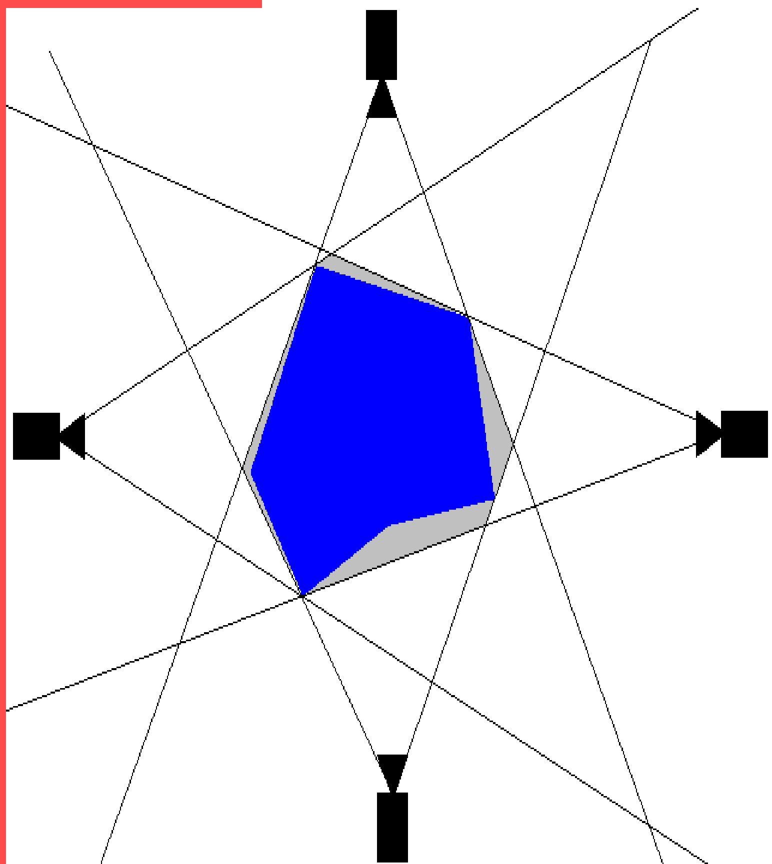
- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes
  - calibrate from silhouettes
- Perspectives and cool ideas

What is shape from silhouette?



- The silhouette, or occluding contour of an object in an image contains some information about the 3D shape of the object.
- Given a single silhouette image of an object, we know that the 3D object lies inside the volume generated by back-projecting the silhouette area using the camera parameters.

What is shape from silhouette?

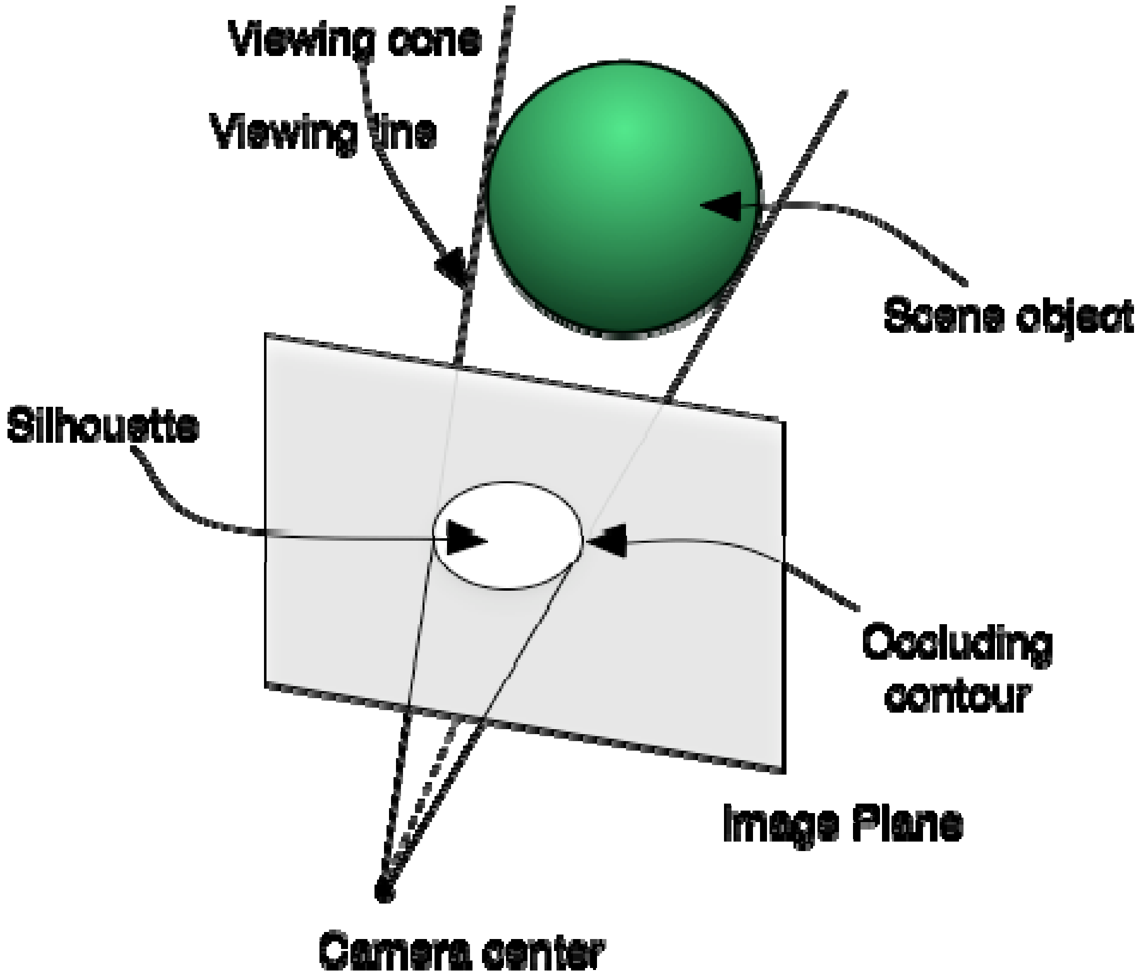


- With multiple views of the same object, we can intersect the *generalized cones* generated by each image, to build a volume which is guaranteed to contain the object.
- The limiting smallest volume obtainable in this way is known as the *visual hull* of the object.

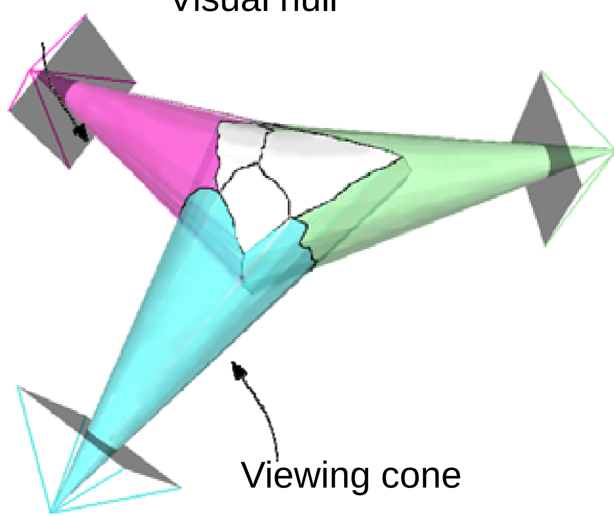
Literature

- Theory
  - Laurentini ’94, Petitjean ’98, Laurentini ’99
- Solid cone intersection:
  - Baumgart ’74 (polyhedra), Szeliski ’93 (octrees)
- Image-based visual hulls
  - Matusik et al. ’00, Matusik et al. ’01
- Advanced modeling
  - Sullivan & Ponce ’98, Cross & Zisserman ’00, Matusik et al. ’02
- Applications
  - Leibe et al. ’00, Lok ’01, Shlyakhter et al. ’01

One-view silhouette geometry



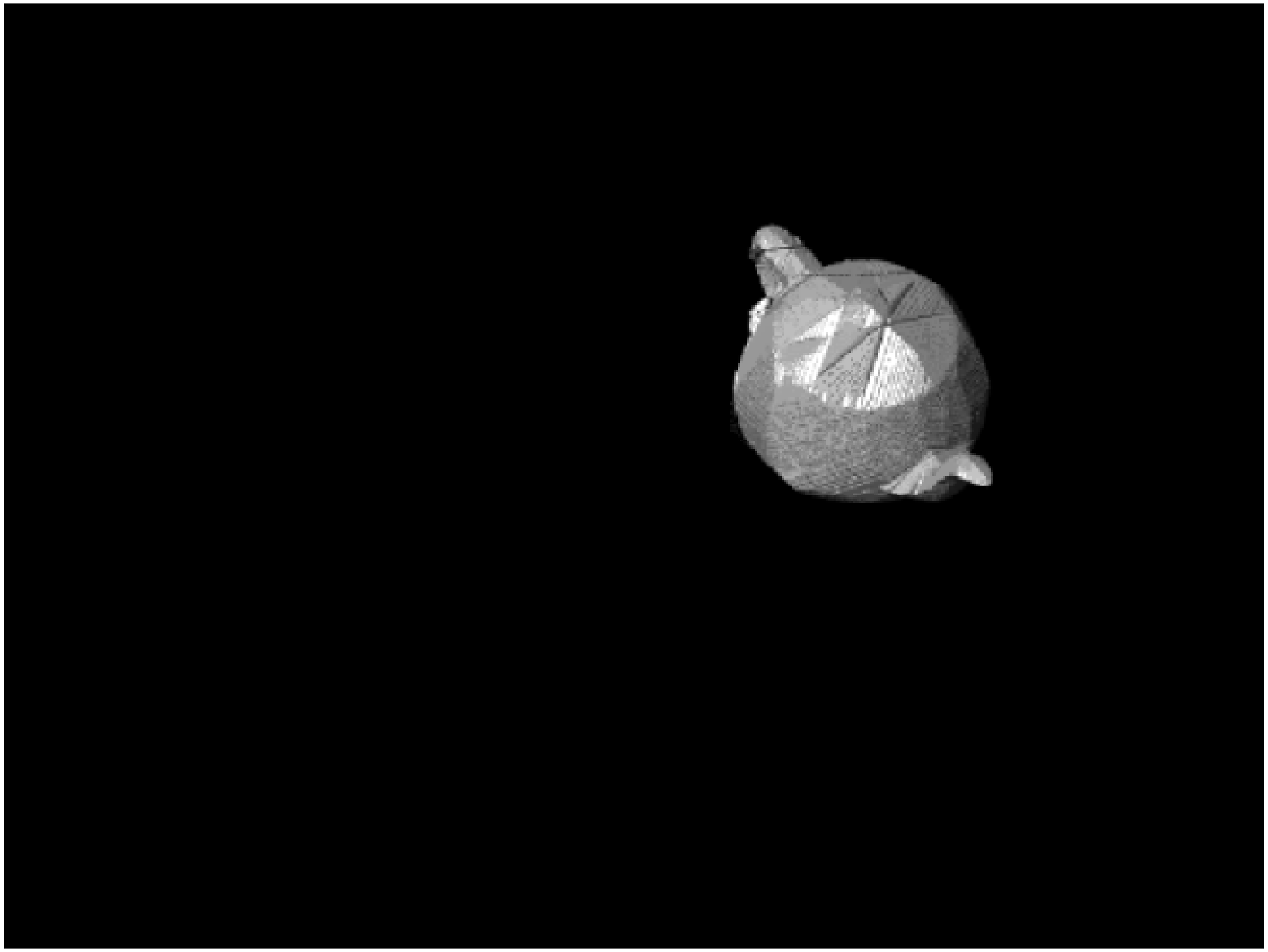
Multi-view silhouette geometry: the Visual Hull



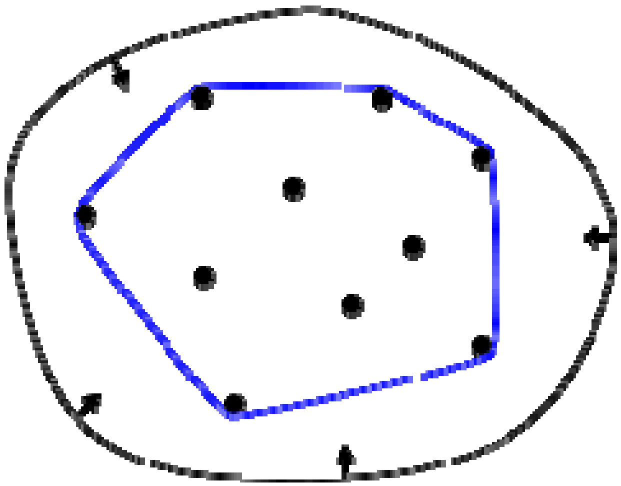
- **Maximal volume consistent with silhouettes** [Laurentini94] [Baumgart74]
- Can be seen as the intersection of viewing cones

- Properties:
  - Containment property: contains real scene objects
  - Converges towards the shape of scene objects minus concavities as N increases
  - Projective structure: simple management of visibility problems

Visual Hull: A 3D Example



Convex Hull: Computational Geometry Problem



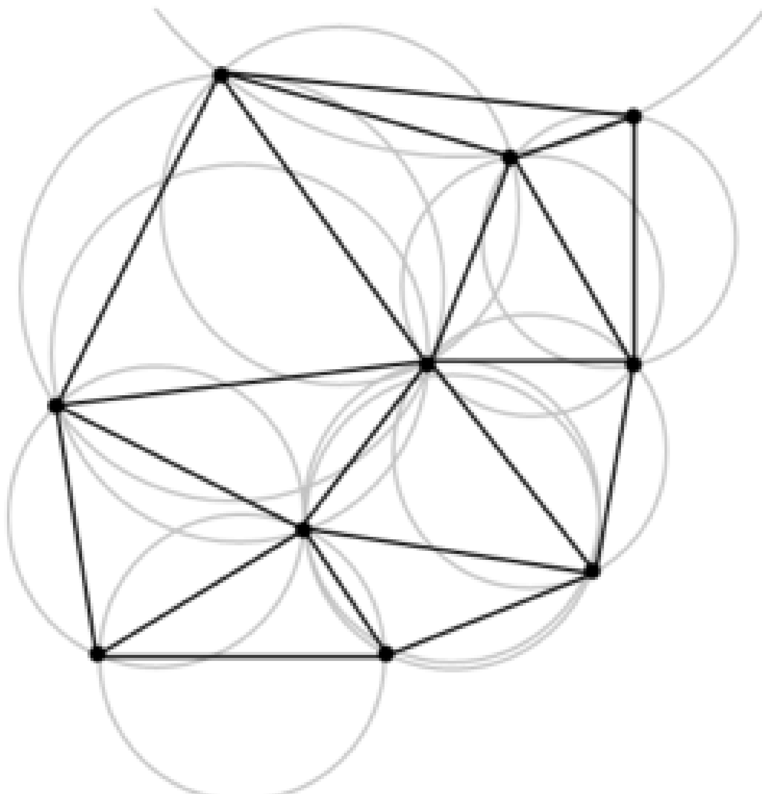
Convex hull: Elastic band analogy: Concave parts of object not part of hull.

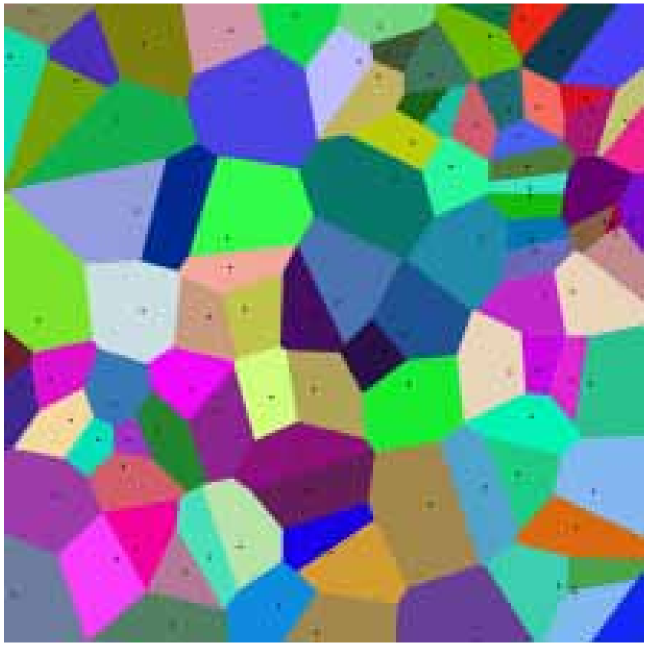
In mathematics, the **convex hull** or **convex envelope** for a set of points *X* in a real vector space V is the minimal convex set containing *X*.

In computational geometry, it is common to use the term "convex hull" for the *boundary* of the minimal convex set containing a given non-empty finite set of points in the plane. Unless the points are collinear, the convex hull in this sense is a simple closed polygonal chain.

Convex Hull: Computational Geometry Problem

Hint: Calculate the convex hull based on the Delauney triangulation and its dual, the Voronoi diagram.

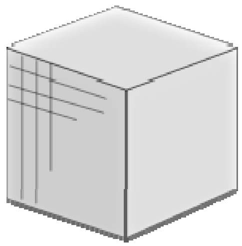




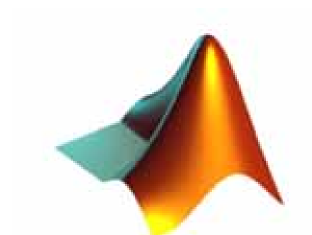
Outline

- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes
  - calibrate from silhouettes
- Perspectives and cool ideas

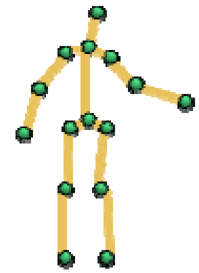
What representation for scene objects?



Voxel grid
Volumetric approaches



Surface
Surface approaches



A priori knowledge
ex: articulated model

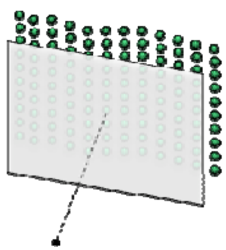
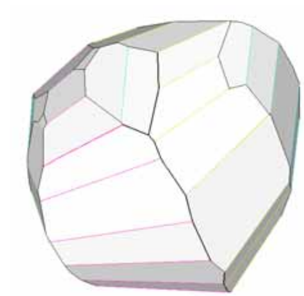


Image-based approaches



Polyhedron mesh

General idea and assumptions

- 2 main families of approaches for VH:
  - focus on visual hull as volume: locate portions of space that don't project in silhouettes (carving)
    - use 2D silhouette *regions* in images
  - focus on visual hull as surface: locate the boundary surface of the visual hull
    - use 2D silhouette *contours* in images
- General assumptions:
  - very good silhouettes are extracted
  - views are calibrated
    - parameters and positions are known

Computational complexity

- Intersection of many volumes can be slow
- Simple polyhedron-polyhedron intersection algorithms are inefficient

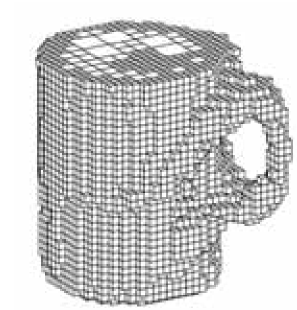
- To improve performance, most methods:
  - Quantize volumes

  and/or

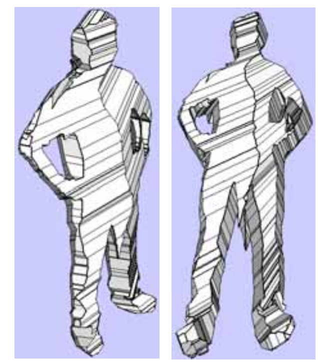
  - Perform Intersection computations in 2D not 3D

Algorithms

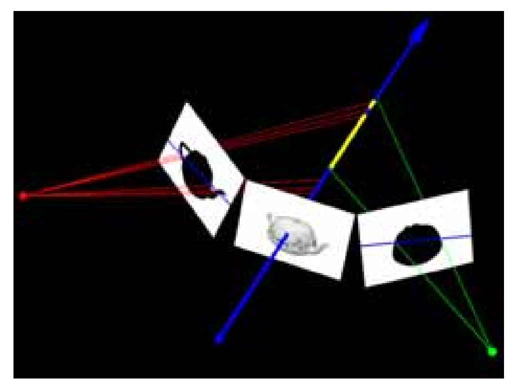
- Standard voxel based method
  Marching Intersections



- Exact polyhedral methods



- Image-based visual hulls

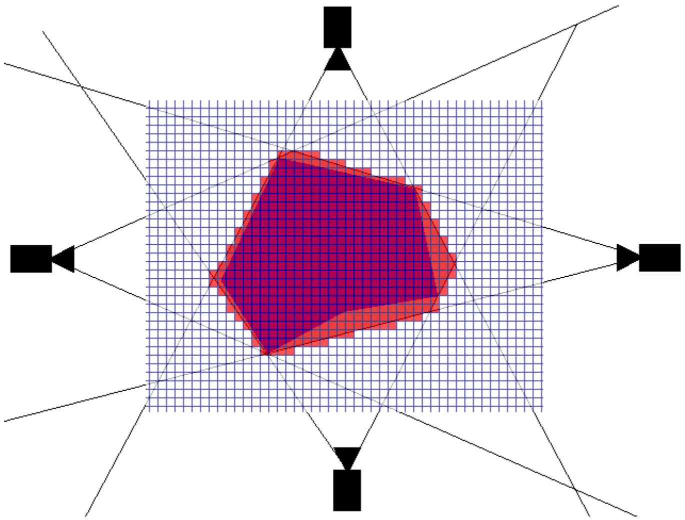


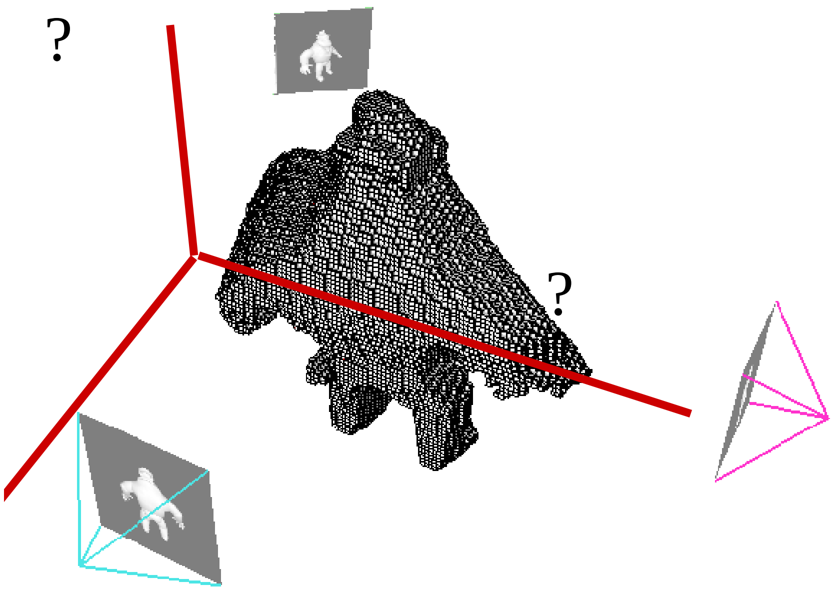
Voxel based

- First the object space is split up into a 3D grid of voxels.
- Each voxel is intersected with each silhouette volume.
- Only voxels that lie **inside all silhouette** volumes remain part of the final shape.

Visual hull as voxel grid

- Identify *3D region* using voxel carving
  - does a given voxel project inside all silhouettes?

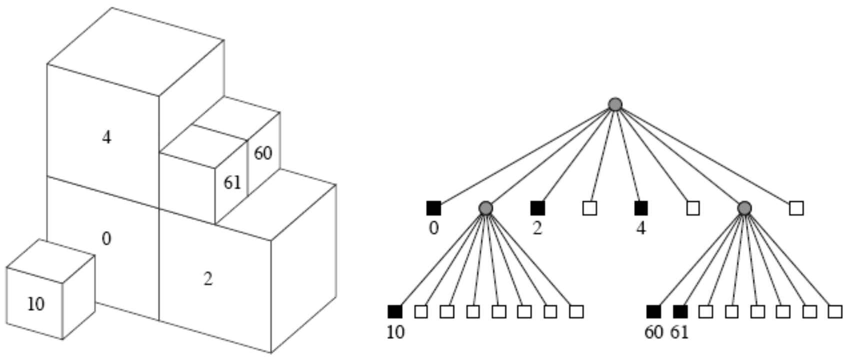


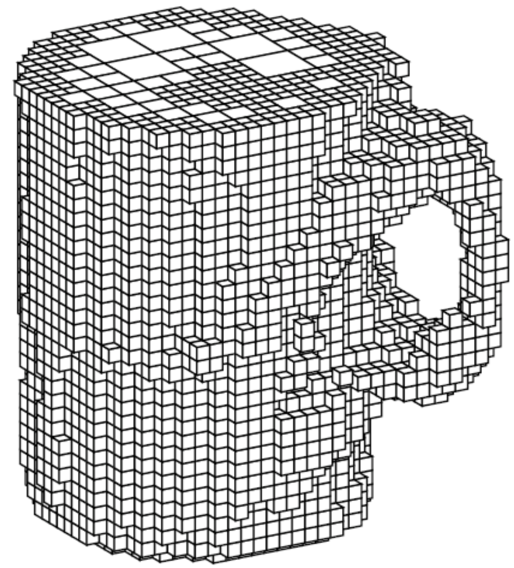


- pros: simplicity
- cons: bad precision/computation time tradeoff

Classical voxel grid improvement: octrees

- Same principle, but refinement through space subdivision





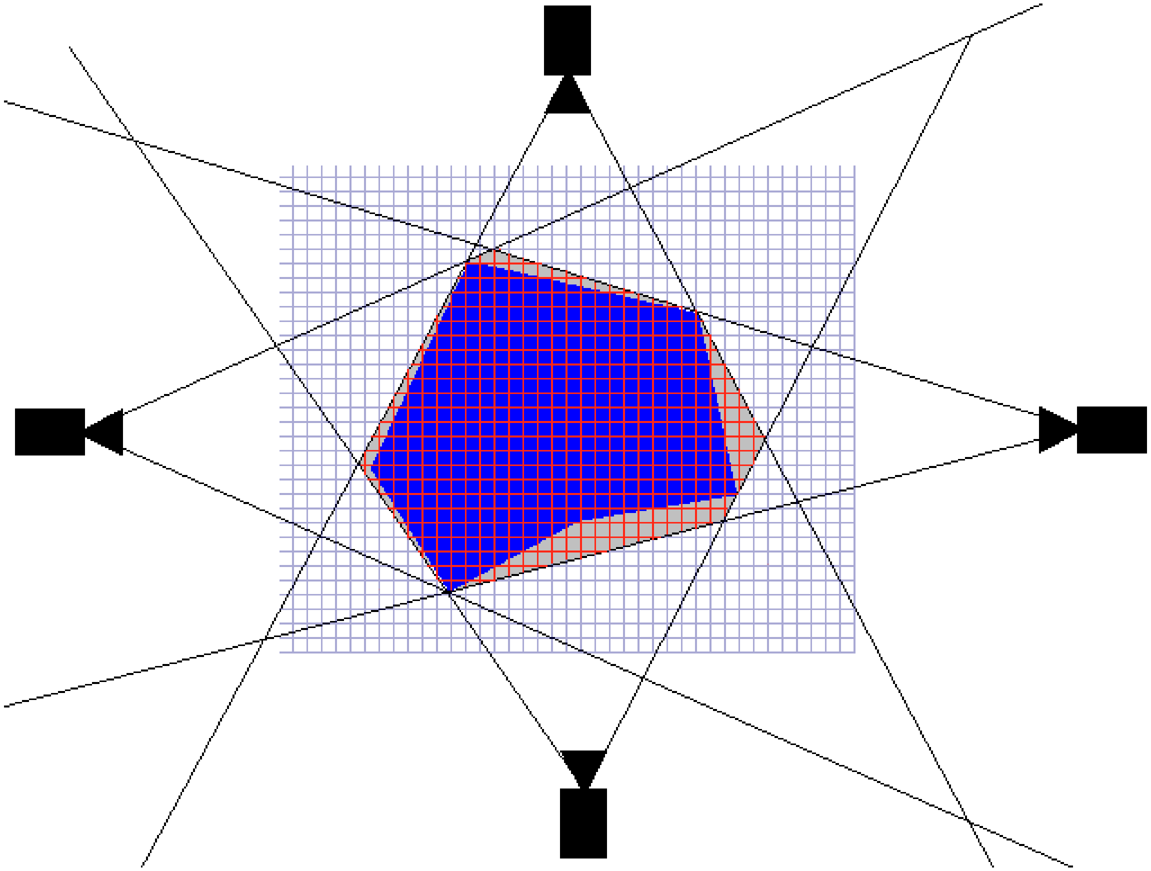
[Szeliski TR 90’]

Marching intersections
Tarini et al., 2002

- The object space is again split up into a 3D grid.
- The grid used is made of 3 sets of rays, rather than voxels.
- Rays are aligned with the 3 axes, and store points of entry/exit into the volume
- Each silhouette cone can be converted to the marching intersections data structure.
- Then merging them is reduced to 1D intersections along each ray.

M. Tarini et al, *Marching intersections, An efficient Approach to Shape from Silhouette*

Marching intersections - example



Marching intersections - example

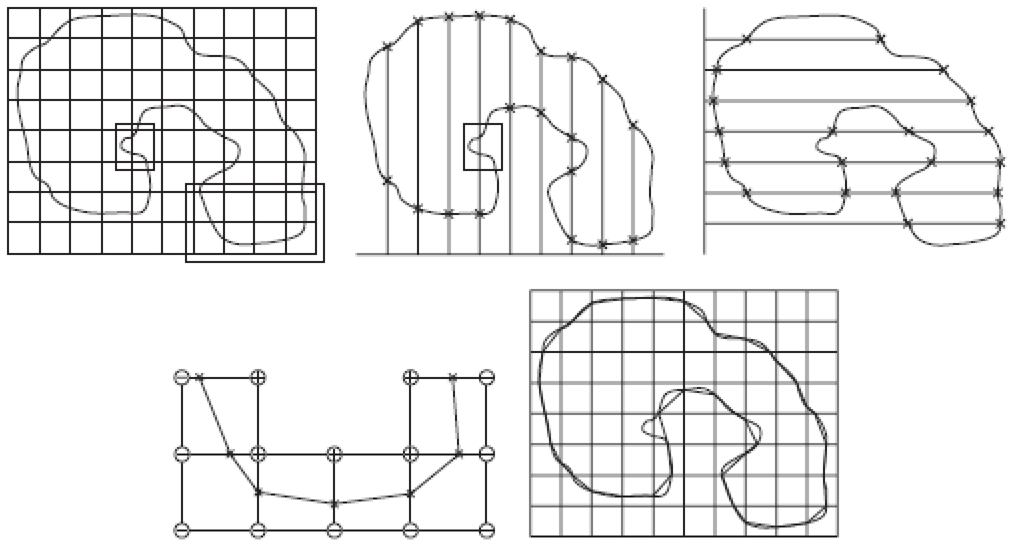
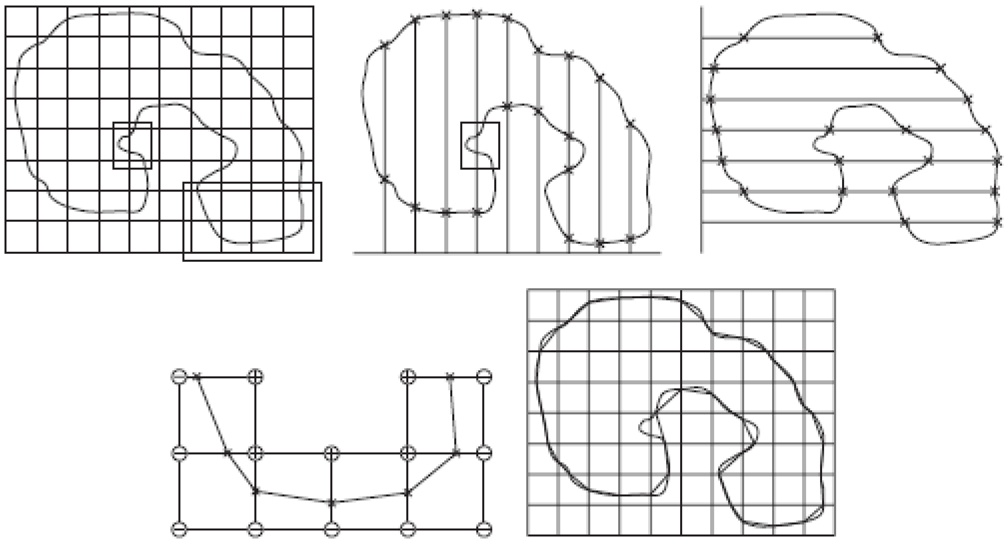


Figure 1: The *MI* data structure and conversion algorithm in a 2D example: (top-left) the curve to be processed and the MI data structures collecting the intersections between the input curve and the vertical (top-center) and horizontal (top-right) lines of the user selected grid; (bottom-left) based on the horizontal and vertical intersections, an MC look up table entry code is located for each not empty (virtual) cell; (bottom-right) the reconstructed curve. The vertical intersections inside the small dotted box are collected and then deleted from the algorithm because belonging to the same virtual cell. This removal is a prerequisite for the correct operation of the algorithm and it leads to the removal of high frequency details.

Marching intersections -Concept



M. Tarini et al, *Marching intersections, An efficient Approach to Shape from Silhouette*

- Given a curve
- Select reference grid
- Intersections between curve and horizontal and vertical lines: MI
- Create look-up-table for each non-empty box

Marching intersections - Silhouettes

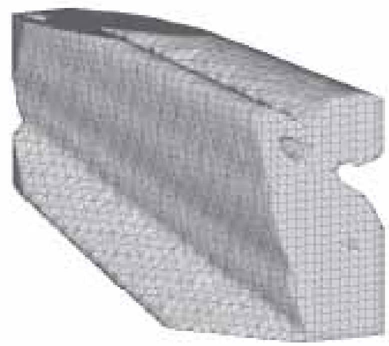


Figure 5: Surface of a conoid extracted from a MI structure obtained with algorithm of Fig. 4.

```
for each ray-set (X,Y,Z)
 for each ray (i,j) in it
  compute the projection on I of ray(i,j)
  scan-convert the corresponding 2D line:
  for each intersection k with silhouette found
    remove perspective distortion, obtaining k'
    add k' to ray(i,j) of the current ray-set
```

Figure 4: A pseudo code for the conversion into a MI structure of a truncated conoid implicitly defined by a 2D silhouette on an image **I** and by a camera position.

- Convert conoid structures to MI datastructure
- Intersection tested in 2D image: purely 2D operation
- Intersection of conoids: AND operations on MI datastructures

Marching intersections - Silhouettes

Figure 6: One object (top left) and its silhouette with 2D lines traced over it to find intersections along rays in the X, Y and Z ray-set of the MI, respectively. The number of lines has been reduced for illustration purposes: in typical cases they would cover most of the area of the image.

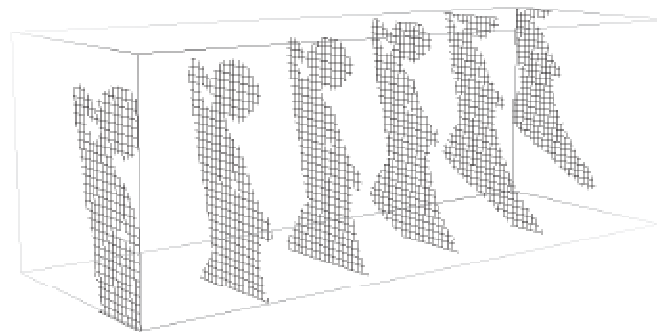


Figure 7: Some slices of the conoid visible in Fig. 5, composed by intervals defined by some of the MI ray along X and Y. Notice that each slice is the replication of another one at a different scale. Cropping is applied on the sides of the volume to make the front and back faces.

Final step: Convert MI datastructure representing all intersections to triangular mesh

Marching intersections - Silhouettes

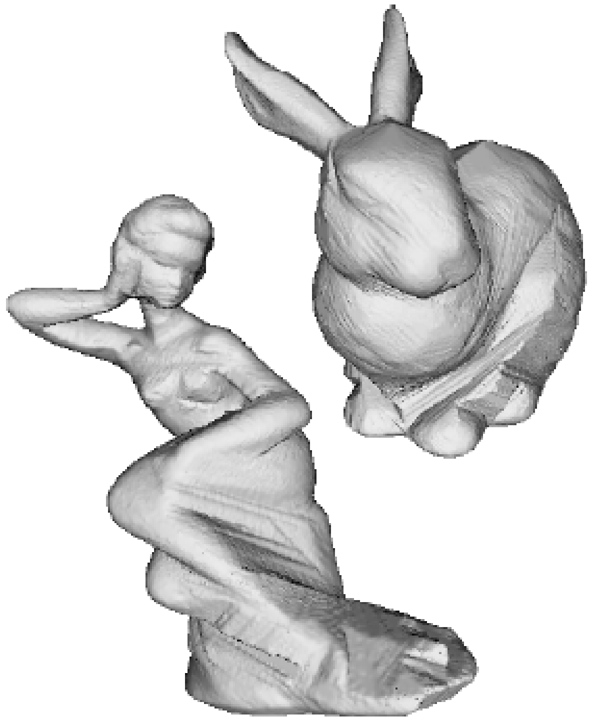


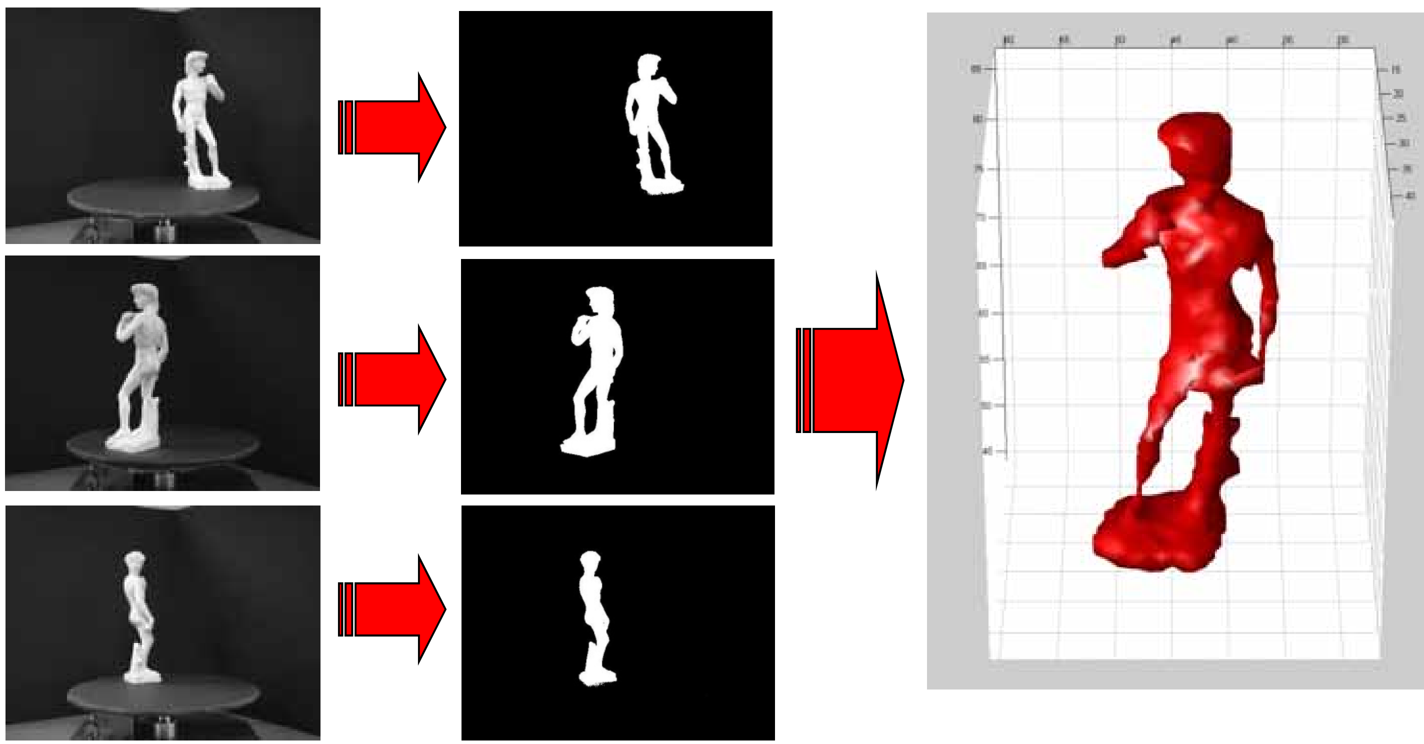
Figure 9: Examples of reconstruction results: the Lady, obtained by silhouettes like the one in Fig. 6, and the familiar Stanford bunny. In both cases a 256x256x256 MI structure has been carved by 128 (synthetic) 1000x1000 silhouettes images taken from a (virtual) camera rotating around it (like when a rotating plate is used). The two meshes are composed by 257K and 451K faces respectively.

Size	Hits	Time0	Time1
Lady silhouettes			
64	12.3%	0.692	0.070
128	44.0%	1.805	0.310
256	80.2%	2.643	1.542
512	94.6%	3.190	8.021
Bunny silhouettes			
64	15.3%	1.350	0.091
128	52.8%	3.079	0.511
256	85.1%	4.049	2.604
512	96.0%	4.895	13.189

Figure 8: Some results of our application running on a normal PC (Athlon 1.4 GHz 512 MB). *Size* is the grid size, *Hits* is the percentage of cache hits, that is, how many out of the $3Size^2$ rays composing the structure were found in the cache described in Sec. 4.2 and were not scan-converted. *Time0* is the time in seconds required to convert a silhouette into a MI structure for the conoid, and to intersect it with the previous MI structure. *Time1* is the time in seconds to extract the final mesh from the MI structure. Total time to compute the models shown in Fig. 9 is ((*Time0* $\times$ number of silhouettes) + *Time1*), which is 340 sec. for the Lady and 577 sec. for the Bunny.

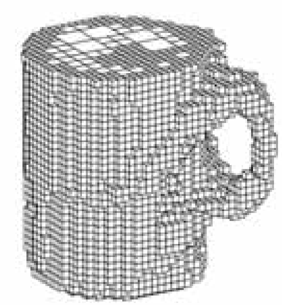
Example: Student Project

- Compute visual hull with silhouette images from multiple calibrated cameras
- Compute Silhouette Image
- Volumetric visual hull computation
- Display the result

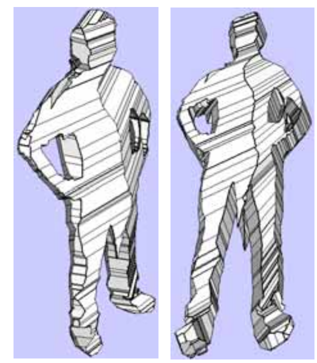


Algorithms

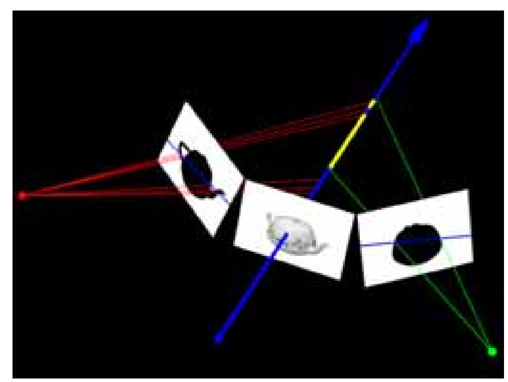
- Standard voxel based method



- Exact polyhedral methods



- Image-based visual hulls



Exact Polyhedral Methods Wojciech Matusik et al.

- First, silhouette images are converted to polygons. (convex or non-convex, with holes allowed)
- Each edge is back projected to form a 3d polygon.
- Then each polygon is projected onto each image, and intersected with each silhouette in 2D.
- The resulting polygons are assembled to form the polyhedral visual hull

Wojciech Matusik, *An Efficient Visual Hull Computation Algorithm*

Exact Polyhedral Methods

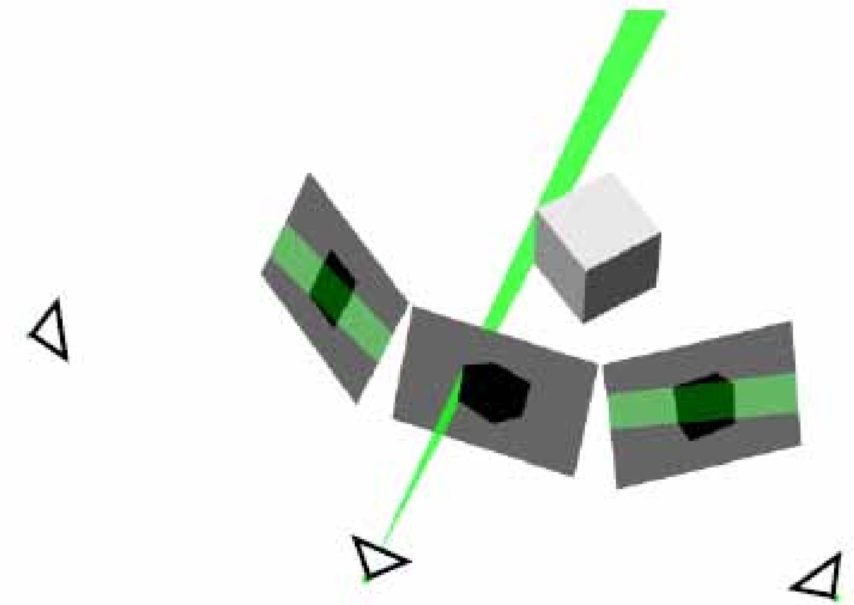
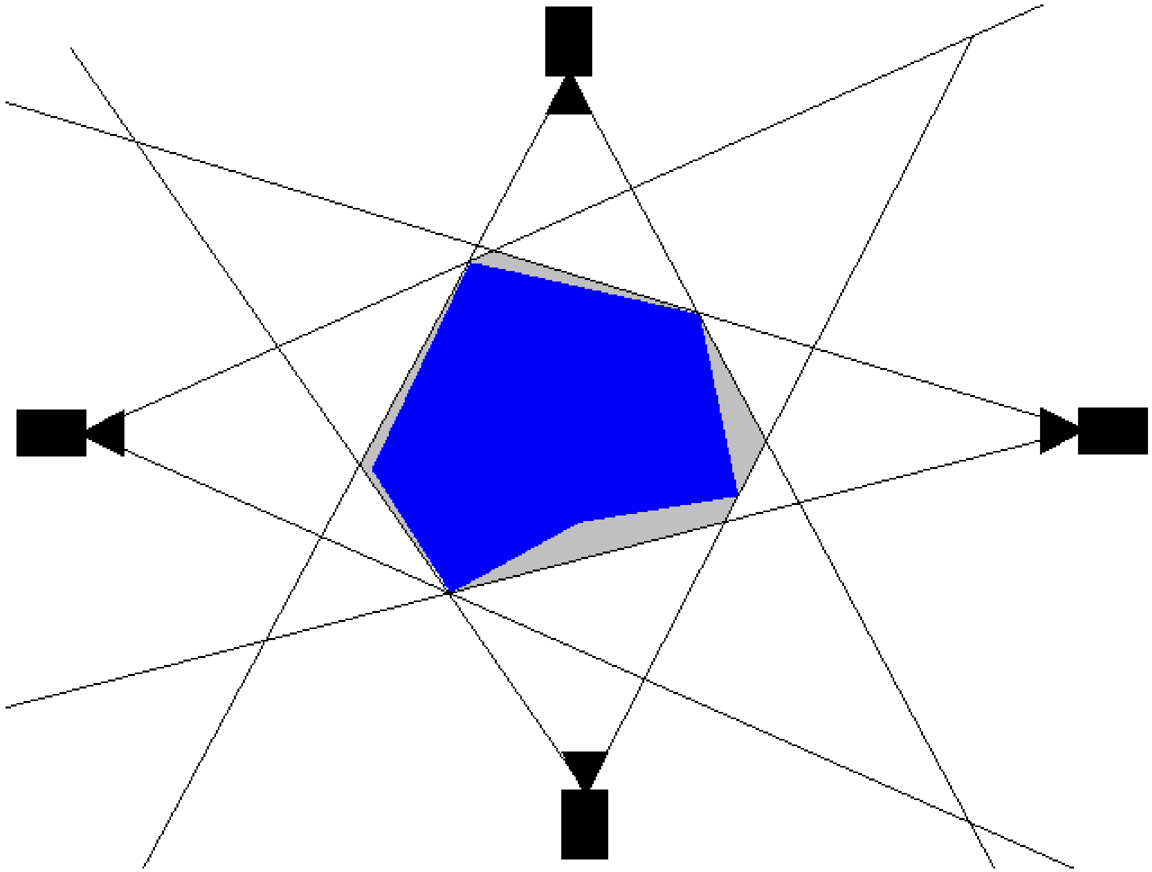


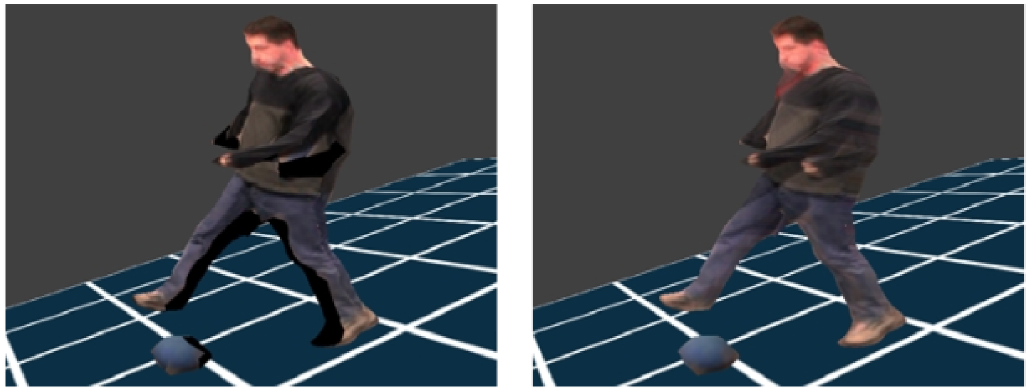
Fig. 1. A single silhouette cone face is shown, defined by the edge in the center silhouette. Its projection in two other silhouettes is also shown.

Wojciech Matusik, *An Efficient Visual Hull Computation Algorithm*

Exact Polyhedral - example

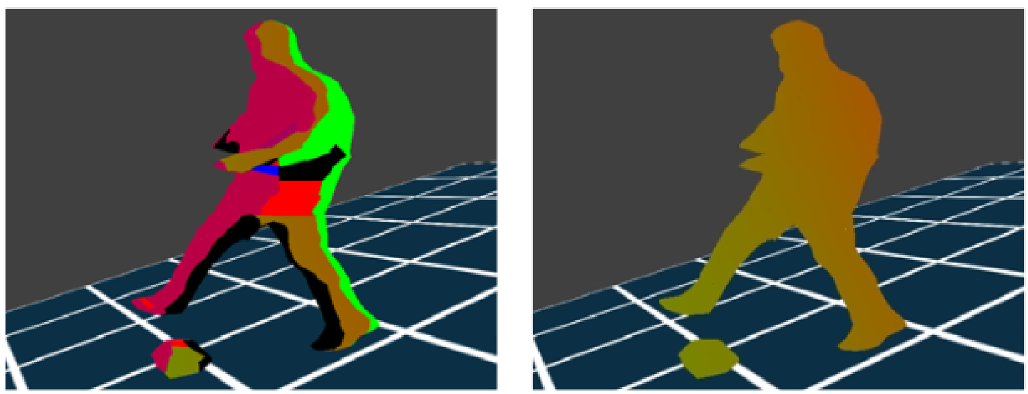


Wojciech Matusik et al.



(a) (b)

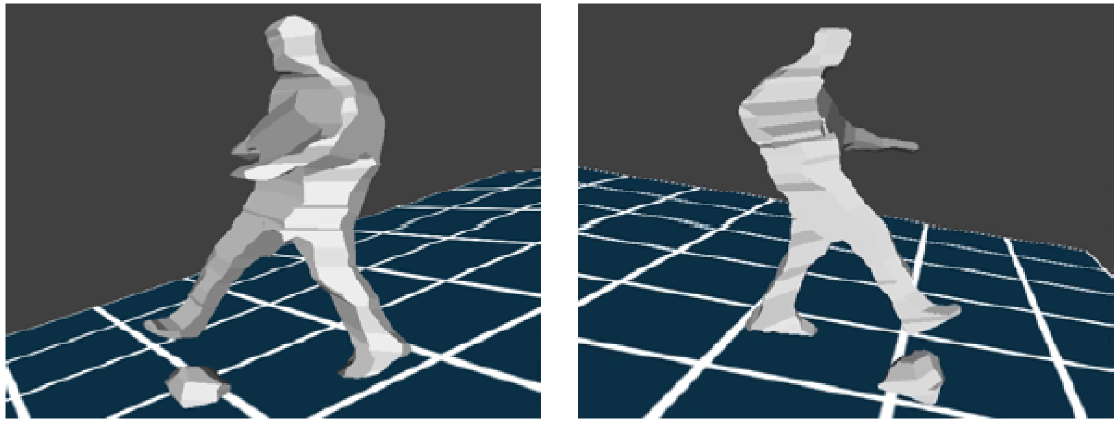
Fig. 7. Two view-dependently textured views of the same visual hull model. The left rendering uses conservative visibility computed in real-time by our algorithm. The right view ignores visibility and blends the textures more smoothly but with potentially more errors.



(a) (b)

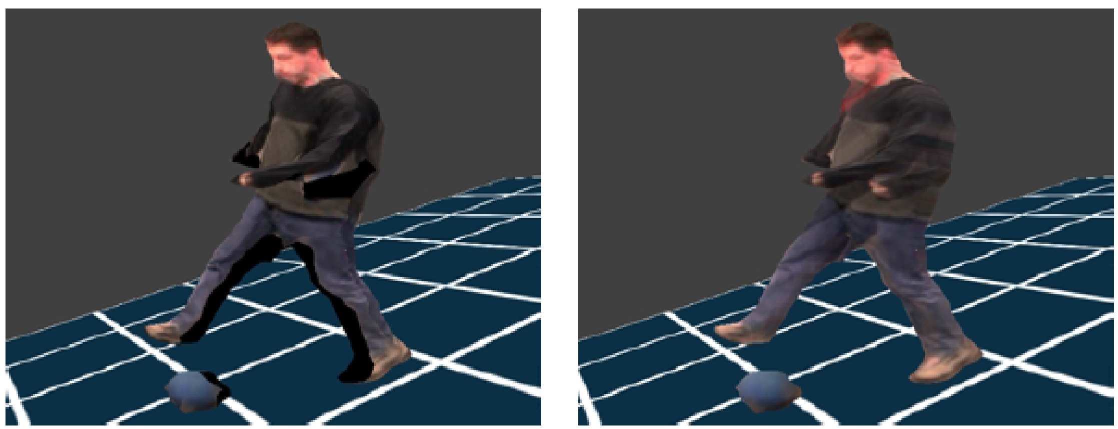
Fig. 8. Two visualizations of the camera blending field. The colors red, green, blue, and yellow correspond to the four cameras in our system. The blended colors demonstrate how each pixel is blended from each input image using both (a) visibility and (b) no visibility.

Wojciech Matusik et al.



(a) (b)

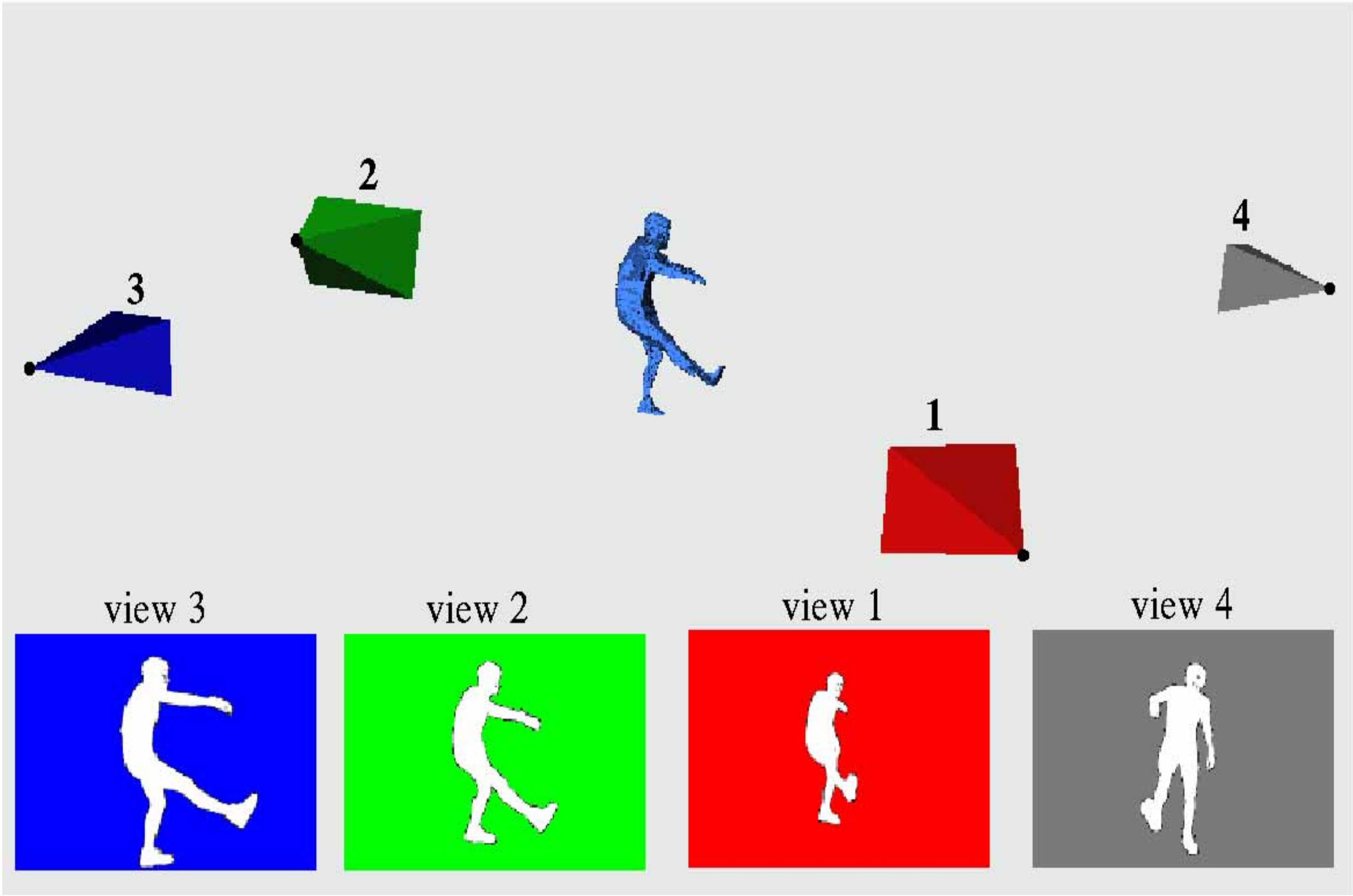
Fig. 6. Two flat-shaded views of a polyhedral visual hull.



(a) (b)

Fig. 7. Two view-dependently textured views of the same visual hull model. The left rendering uses conservative visibility computed in real-time by our algorithm. The right view ignores visibility and blends the textures more smoothly but with potentially more errors.

Metric Cameras and Visual-Hull Reconstruction from 4 views



Final calibration quality comparable to explicit calibration procedure

IBVH Results

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

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

IBVH Results

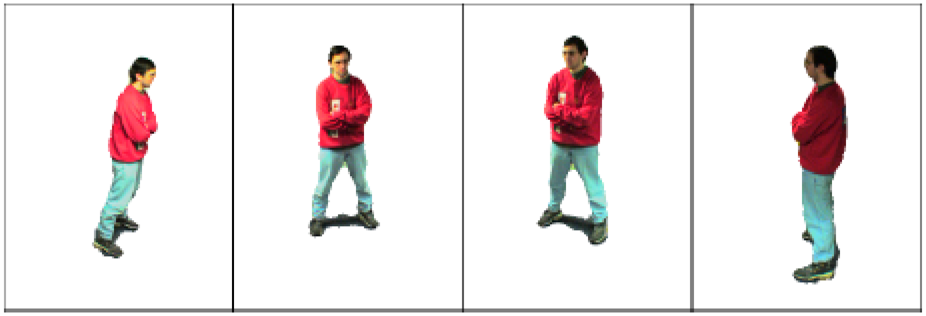


Figure 6 – Four segmented reference images from our system.

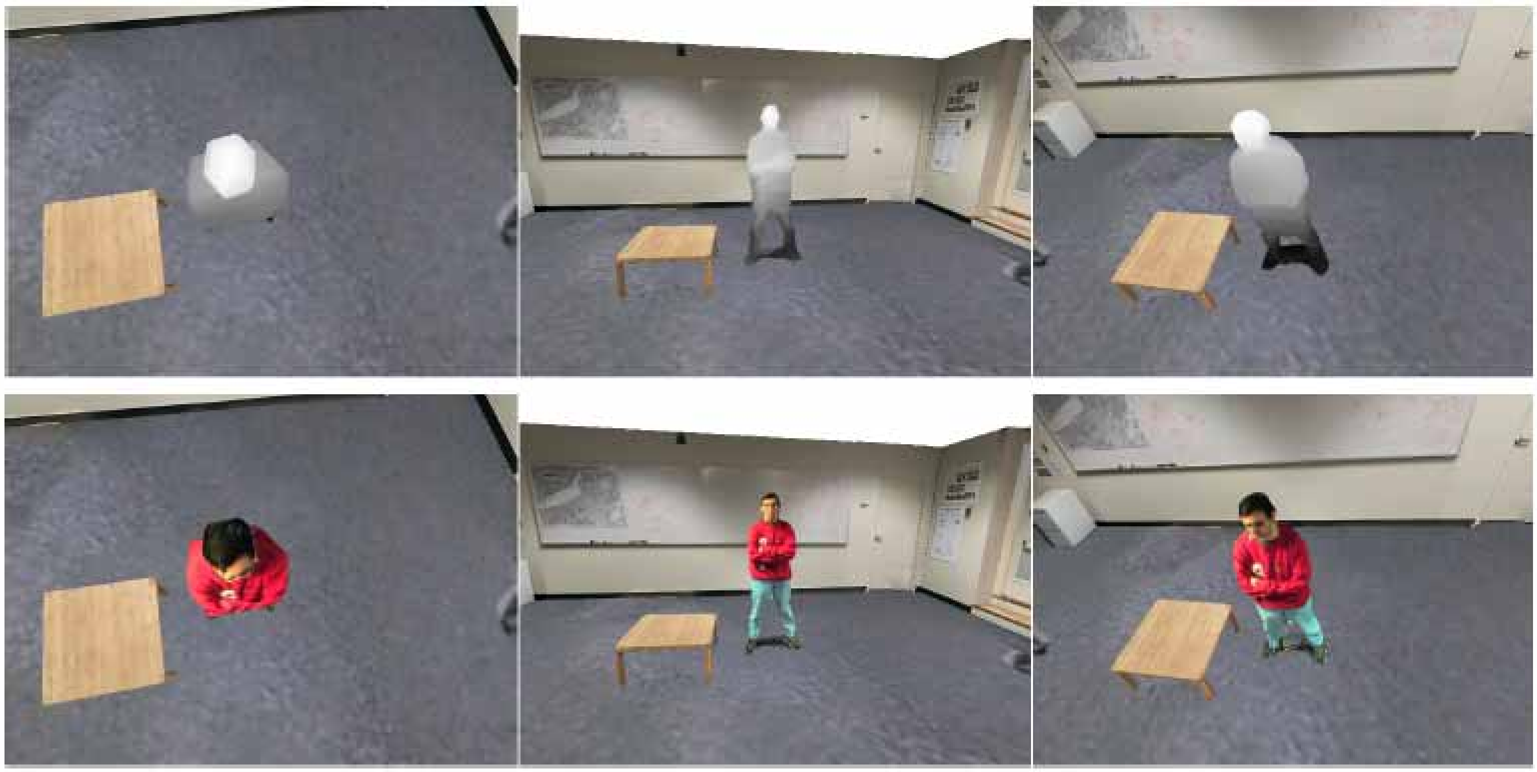
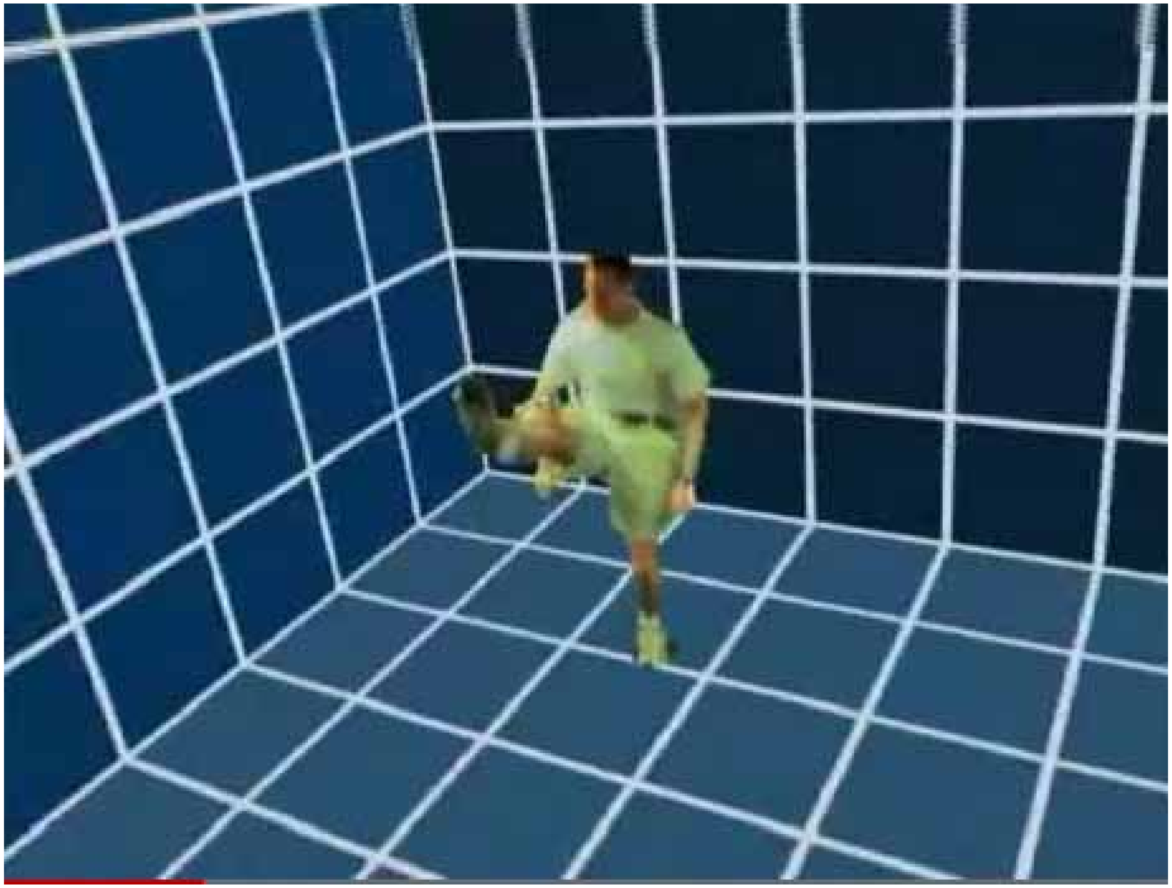


Figure 8 - Example IBVH images. The upper images show depth maps of the computed visual hulls. The lower images show shaded renderings from the same viewpoint. The hull segment connecting the two legs results from a segmentation error caused by a shadow.

Image Based Visual Hulls

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



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

Video Shading

Movie

http://www.google.com/url?q=http://www.vidoemo.com/yvideo.php%3Fi%3DTHc5YUZhcWuRpSG9iYW8%26image-based-visual-hulls%3D&ei=Zc8ES8wggaKyA73OrcIK&sa=X&oi=video_result&resnum=5&ct=thumbnail&ved=0CCQQuAIwBA&usg=AFQjCNETWA_Eqgy8_10mJUmc540zpx8T6A