Shape from Silhouettes
Class 13

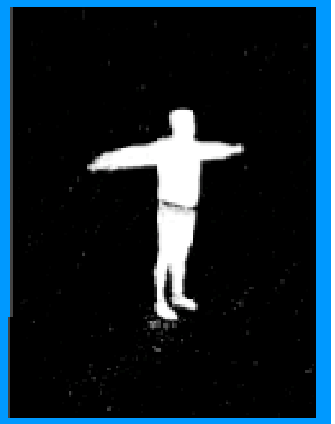
Some of the figures and slides are adapted from S. Sinha, J.S. Franco, J. Matusik's presentations, and referenced papers.

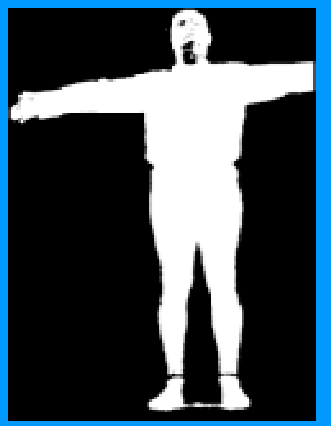
ETH

Outline

- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes

ETH

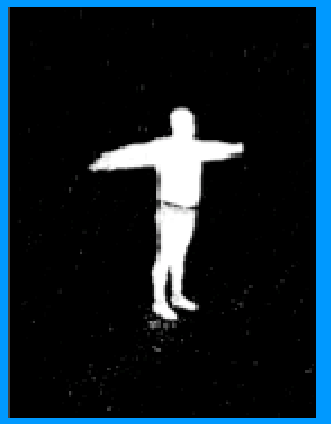




Silhouettes of objects of interest

- 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

ETH



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)

ETH

How to extract silhouettes?

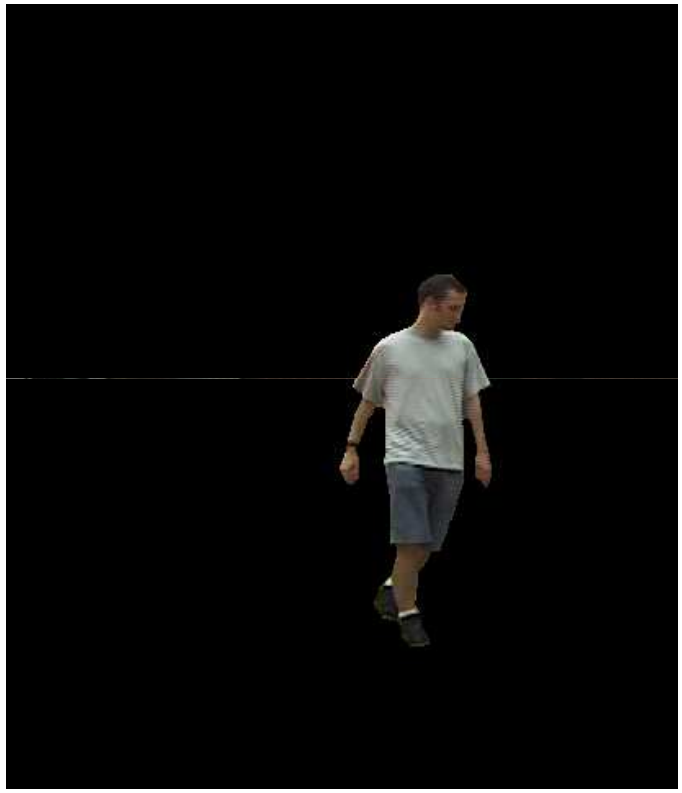
- Contour-based extraction
- focus on silhouette *outline* instead of region itself
  - snakes, active contours: fitting of a curve to high gradients in image, local optimization

Yilmaz&Shah ACCV04

ETH

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



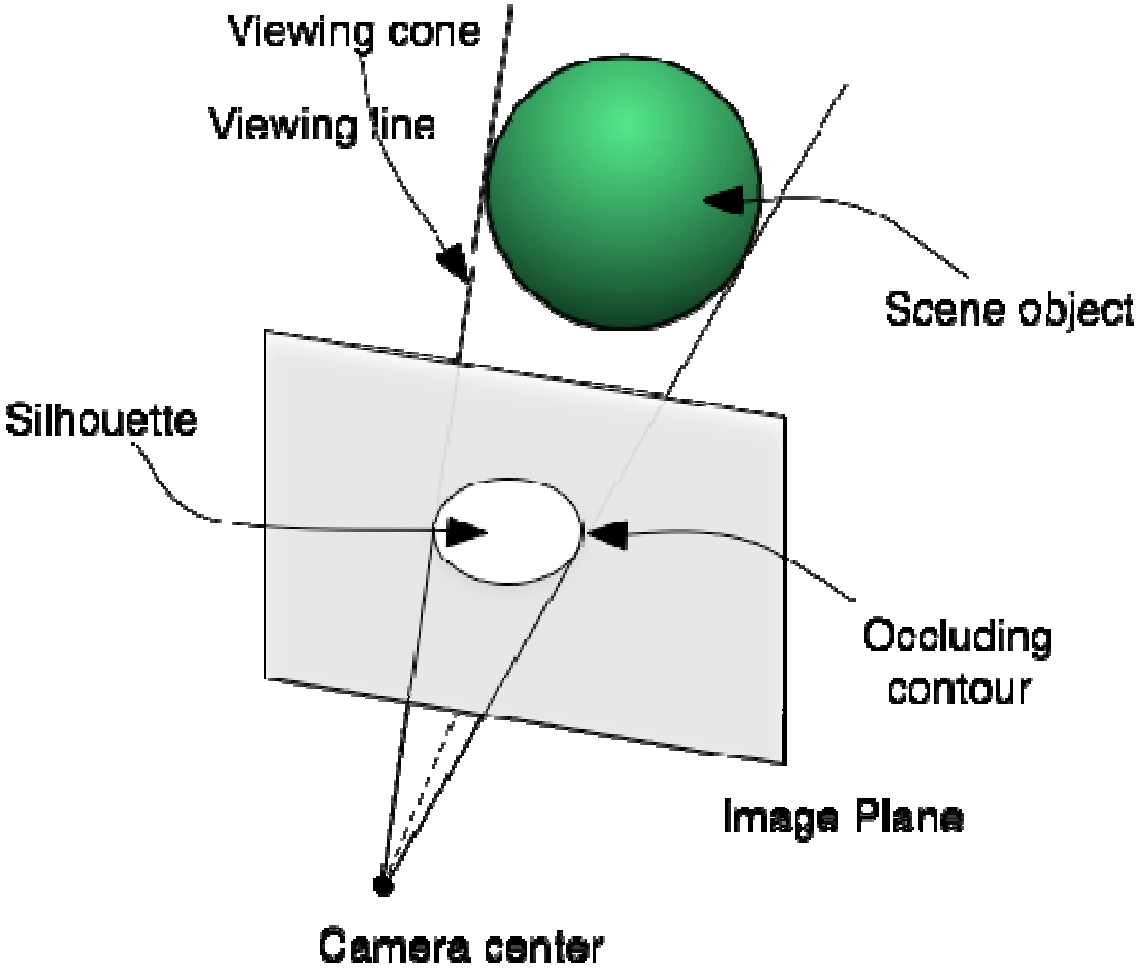
ETH

Outline

- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes

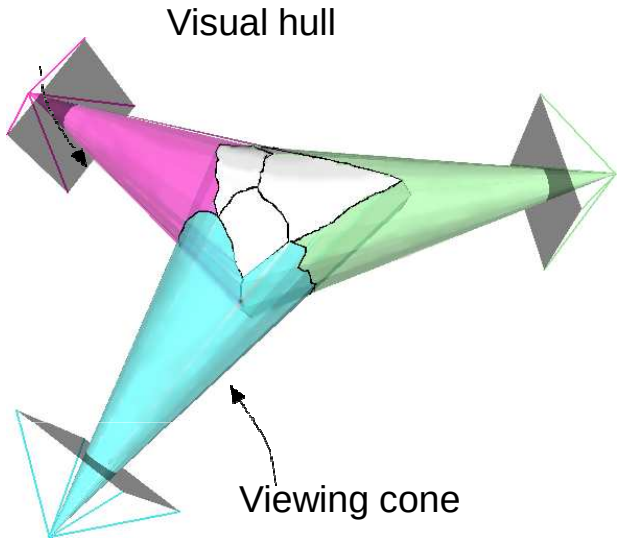
ETH

One-view silhouette geometry



ETH

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

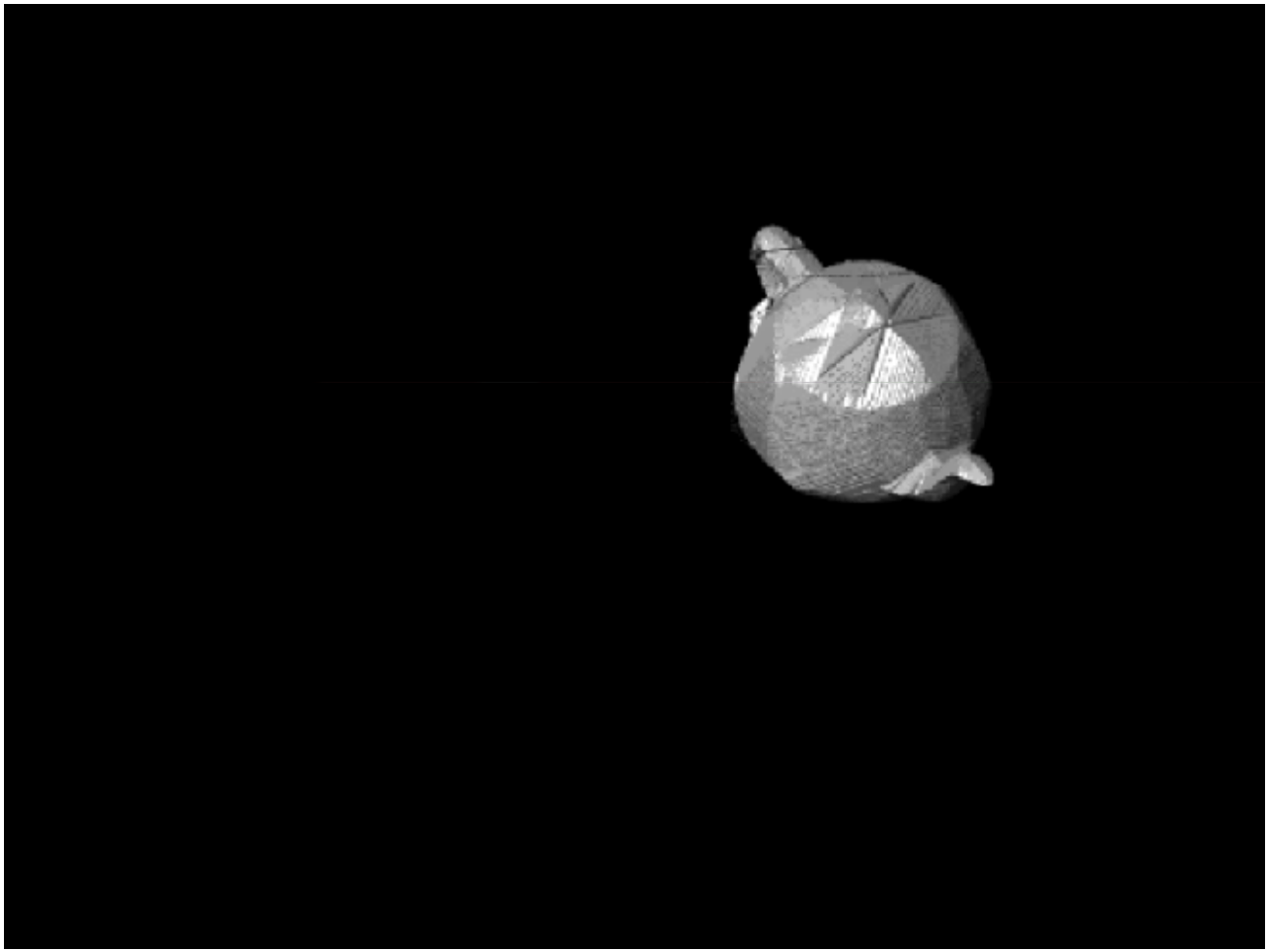
ETH

Why use a Visual Hull?

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

ETH

Visual Hull: A 3D Example

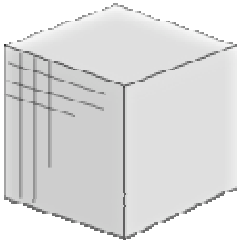


Outline

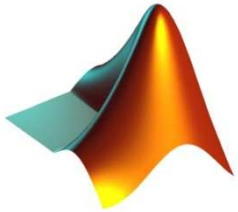
- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes

ETH

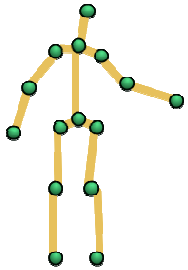
What representation for scene objects?



Voxel grid
Volumetric approaches



Surface
Surface approaches



A priori knowledge
ex: articulated model

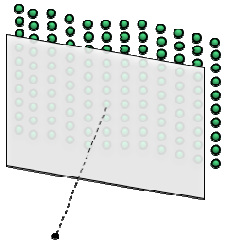
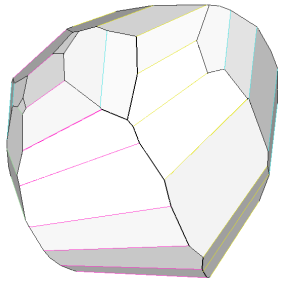


Image-based approaches



Polyhedron mesh

ETH

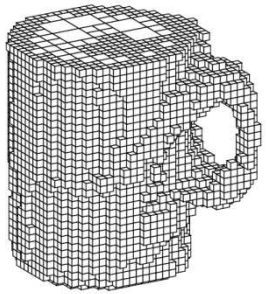
General idea and assumptions

- 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

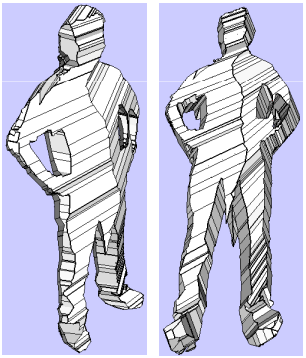
ETH

Algorithms

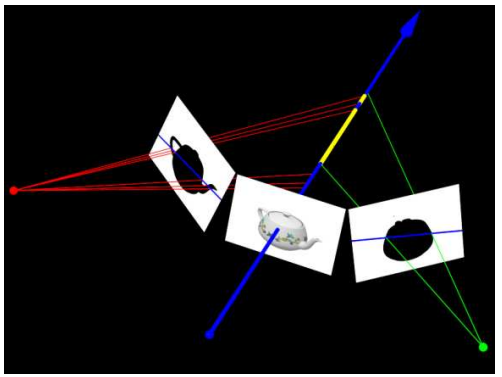
- Standard voxel based method



- Exact polyhedral methods

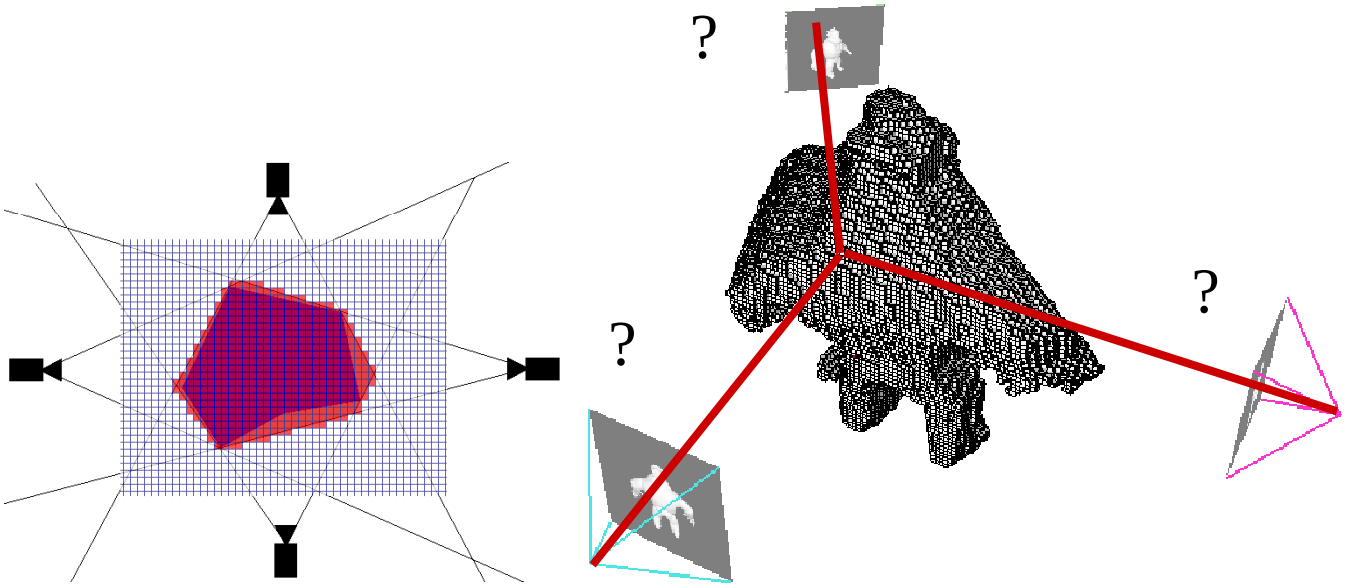


- Image-based visual hulls



Visual hull as voxel grid

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

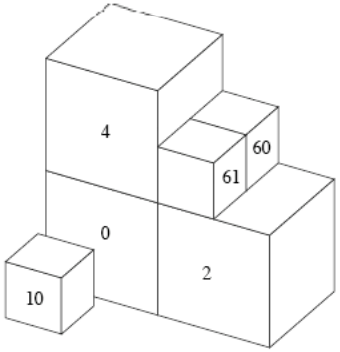


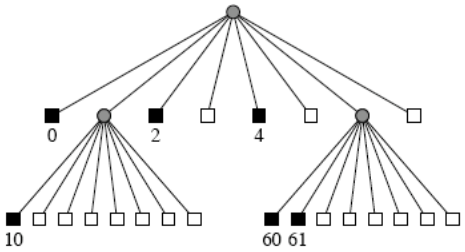
- pros: simplicity
- cons: bad precision/computation time tradeoff

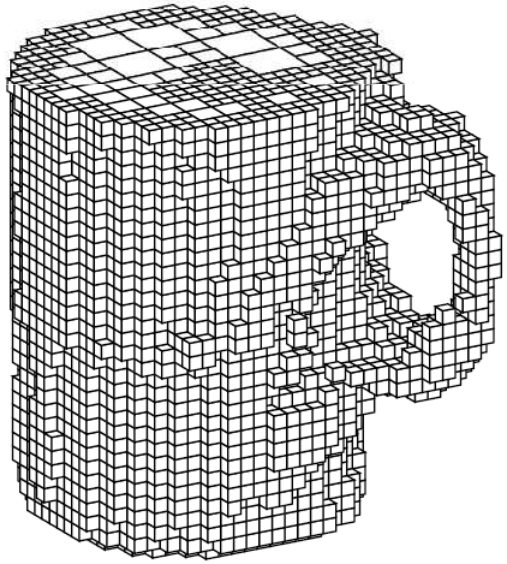
ETH

Classical voxel grid improvement: octrees

- Same principle, but refinement through space subdivision







[Szeliski TR 90']

ETH

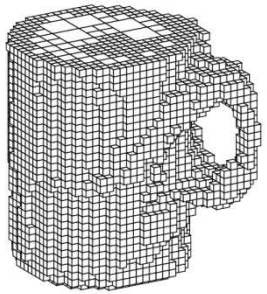
Voxel grid based algorithm

- 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.

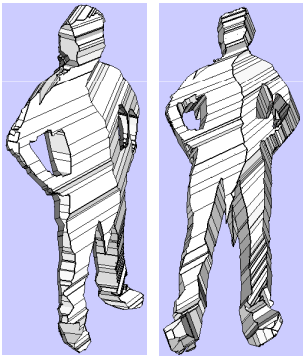
ETH

Algorithms

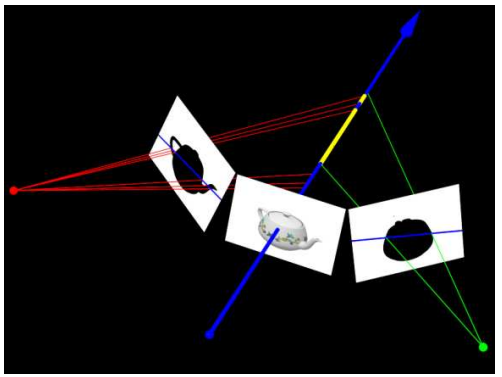
- Standard voxel based method



- Exact polyhedral methods



- Image-based visual hulls



Exact Polyhedral Methods

- 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

W. Matusik, *An Efficient Visual Hull Computation Algorithm*

ETH

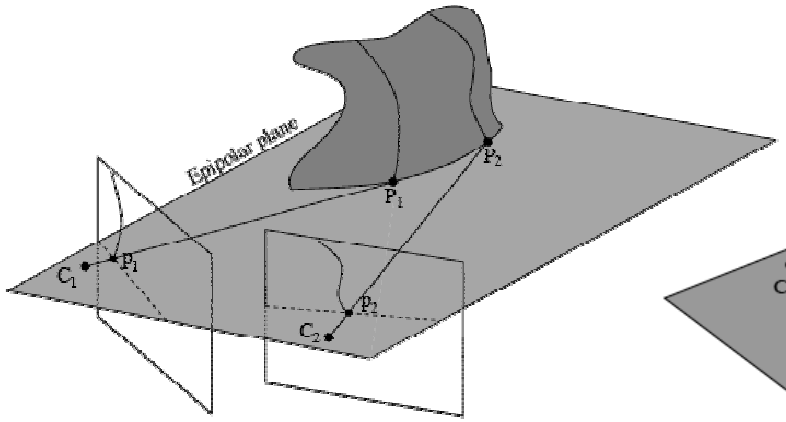
Visual Hull as surface

- Additional assumptions:
  - good silhouettes
  - either of the following is true
    - full calibration *with consistent orientation information* is known
    - cameras are weakly calibrated and the oriented epipolar geometry of the cameras is known
  - this is necessary to recover the correct outward surface normals

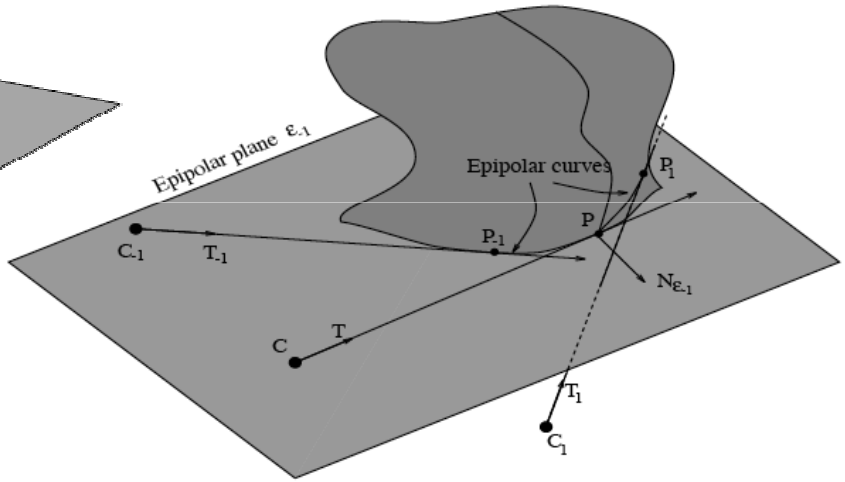
ETH

Visual Hull as smooth surface

- "shape-from-contours"



Boyer IJCV 95

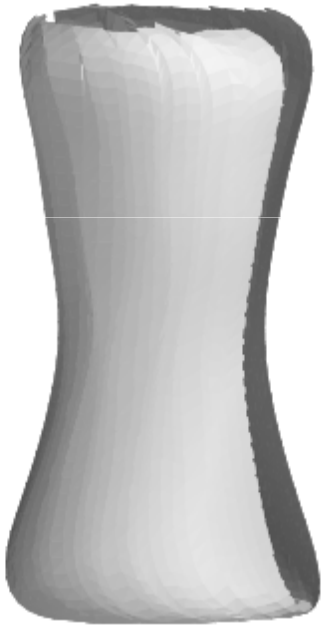


- can compute tangent and normals to smooth surfaces quite accurately from epipolar correspondances between adjacently observed occluding contours (turntable sequences)

ETH

Visual hull as smooth surface

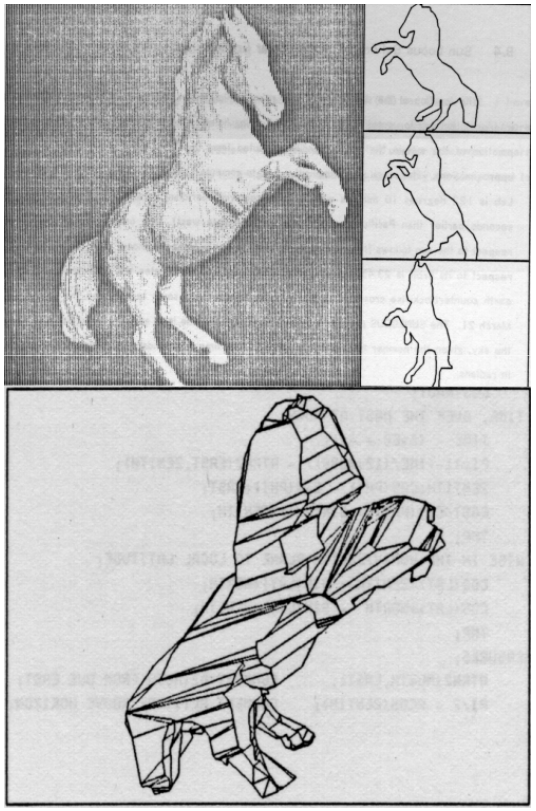
- results: problems with topology



Boyer IJCV 95

ETH

Visual Hull as Polyhedron

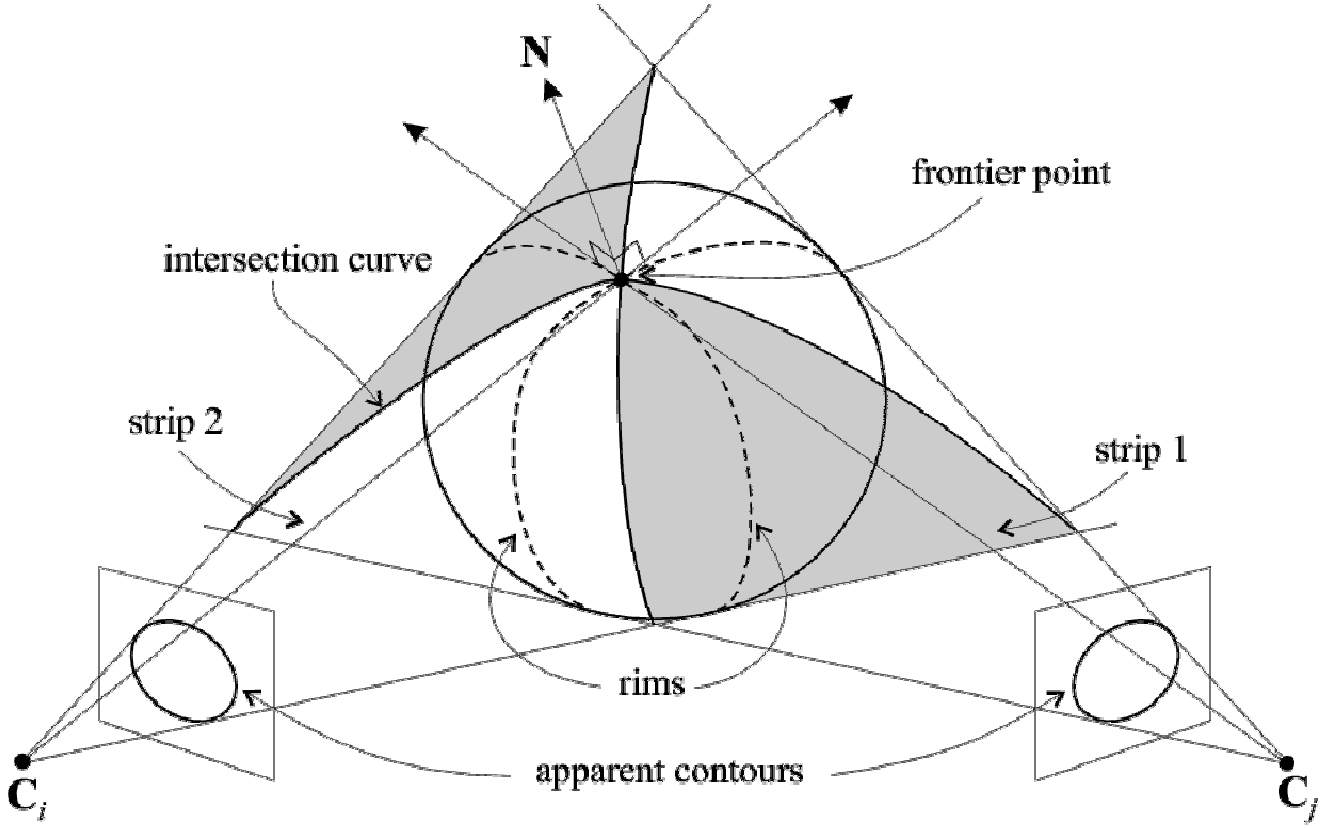


- assumes *polygonal* occluding contours $O(n^2p^2)$
- Baumgart (1974!): tests all possible intersections between primitives

ETH

Visual hull as smooth surface

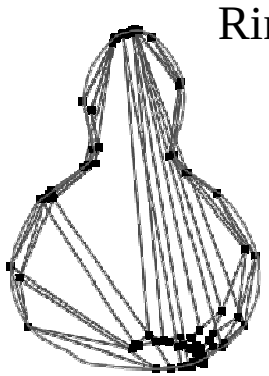
- Lazebnik 01: study of visual hull topology from smooth contours.
  Problems: numerical instability to compute frontier points



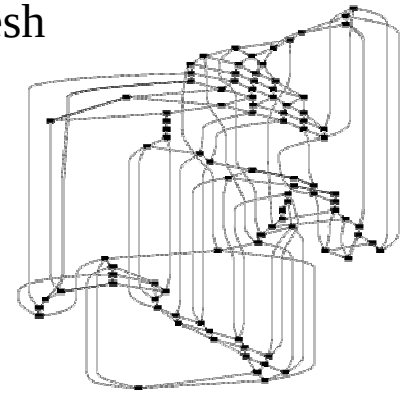
ETH

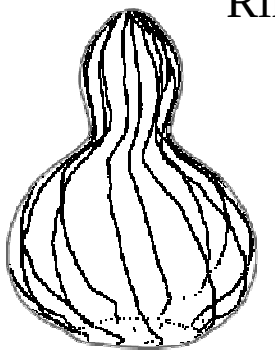
Visual hull as smooth surface

- Lazebnik 01: study of visual hull topology from smooth contours.
  Problems: numerical instability to compute frontier points

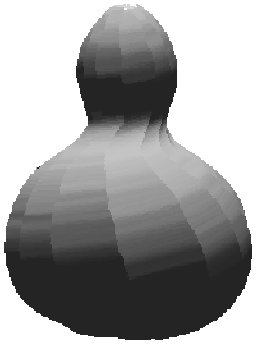


Rim mesh

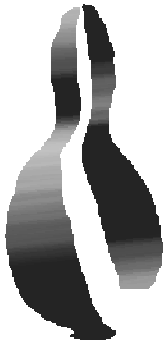




Rims

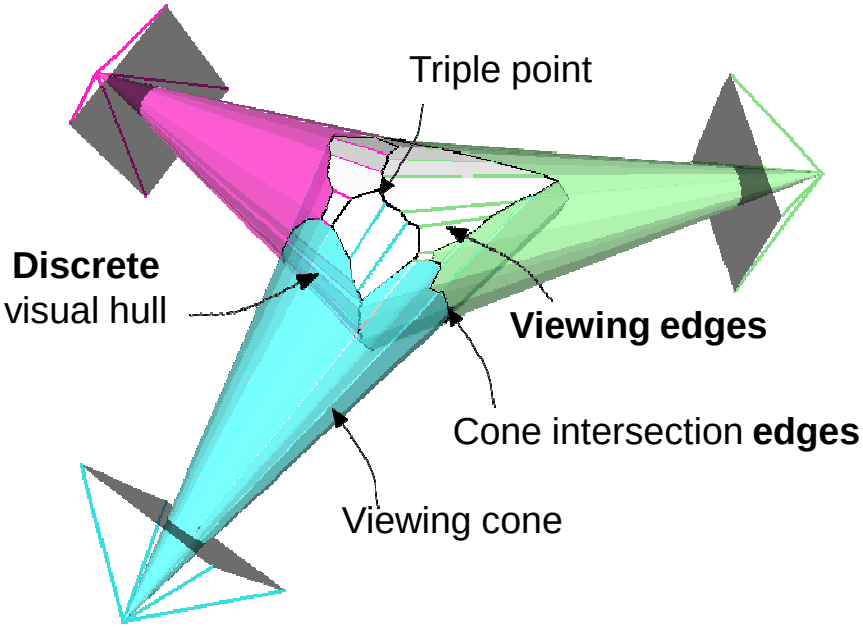


Visual hull



Strips

Exact Polyhedral Visual Hulls



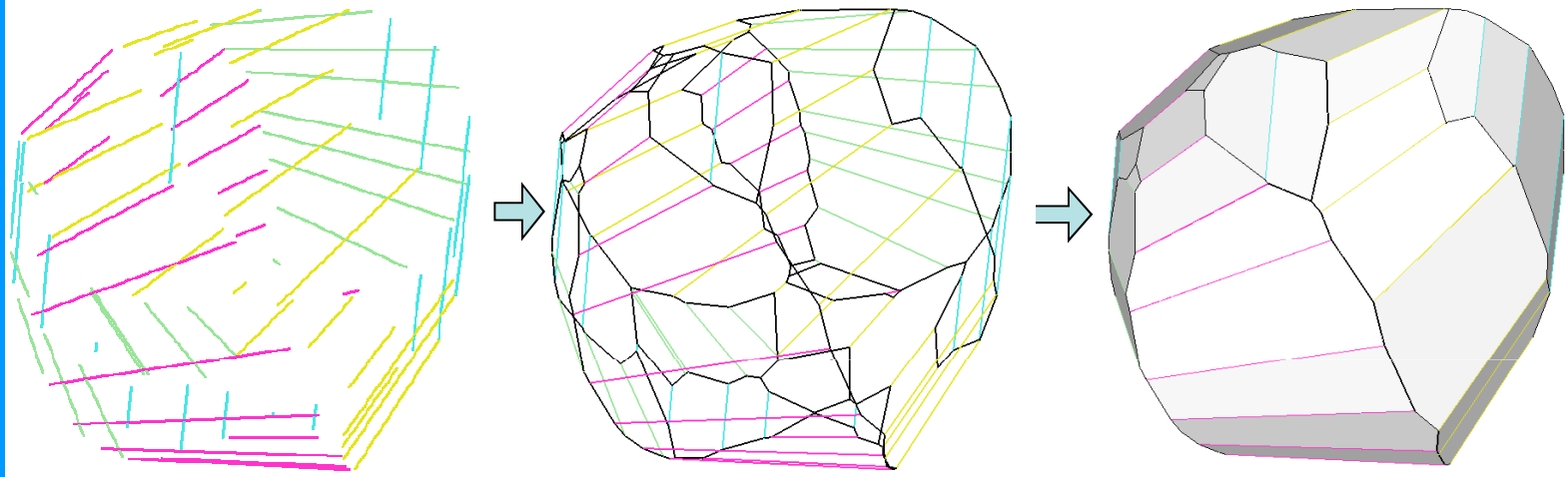
[Franco & Boyer, *BMVC*, 2003]

- Analysis of discrete visual hull structure leads to topologically correct surface reconstruction algorithm

- The contour discretization imposes the geometry of viewing cones, and the geometry of the surface
  - viewing edges
  - cone intersection edges

ETH

EPVH Algorithm



3 steps:

1. Compute viewing edges
2. Cone intersection edges and triple points
3. Faces.

$$O(n^2 \log p)$$

ETH

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

ETH

Image based visual hulls

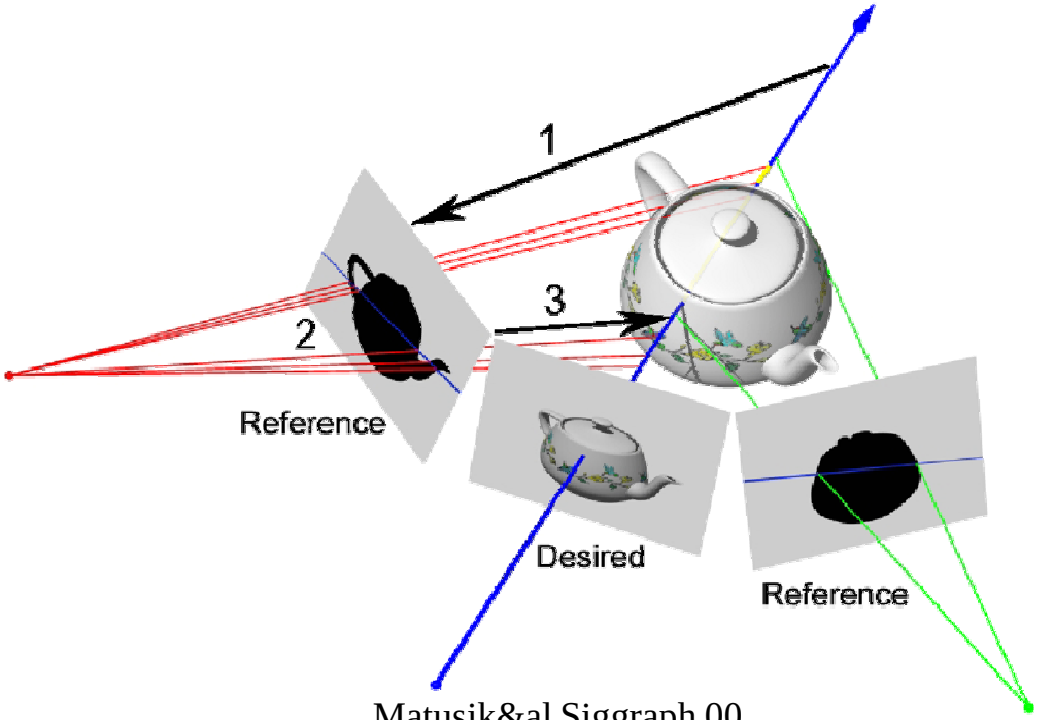
- This algorithm will only produce renderings of a visual hull from any view.
- Every pixel in the desired output image is back-projected to form a 3D ray.
- Each of those rays is intersected with each of the input silhouettes.
- Then a pixel in the output image is inside the new rendering of the visual 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

[W. Matusik, *Image Based Visual Hulls*]

ETH

Visual Hull as view dependent image-based representation

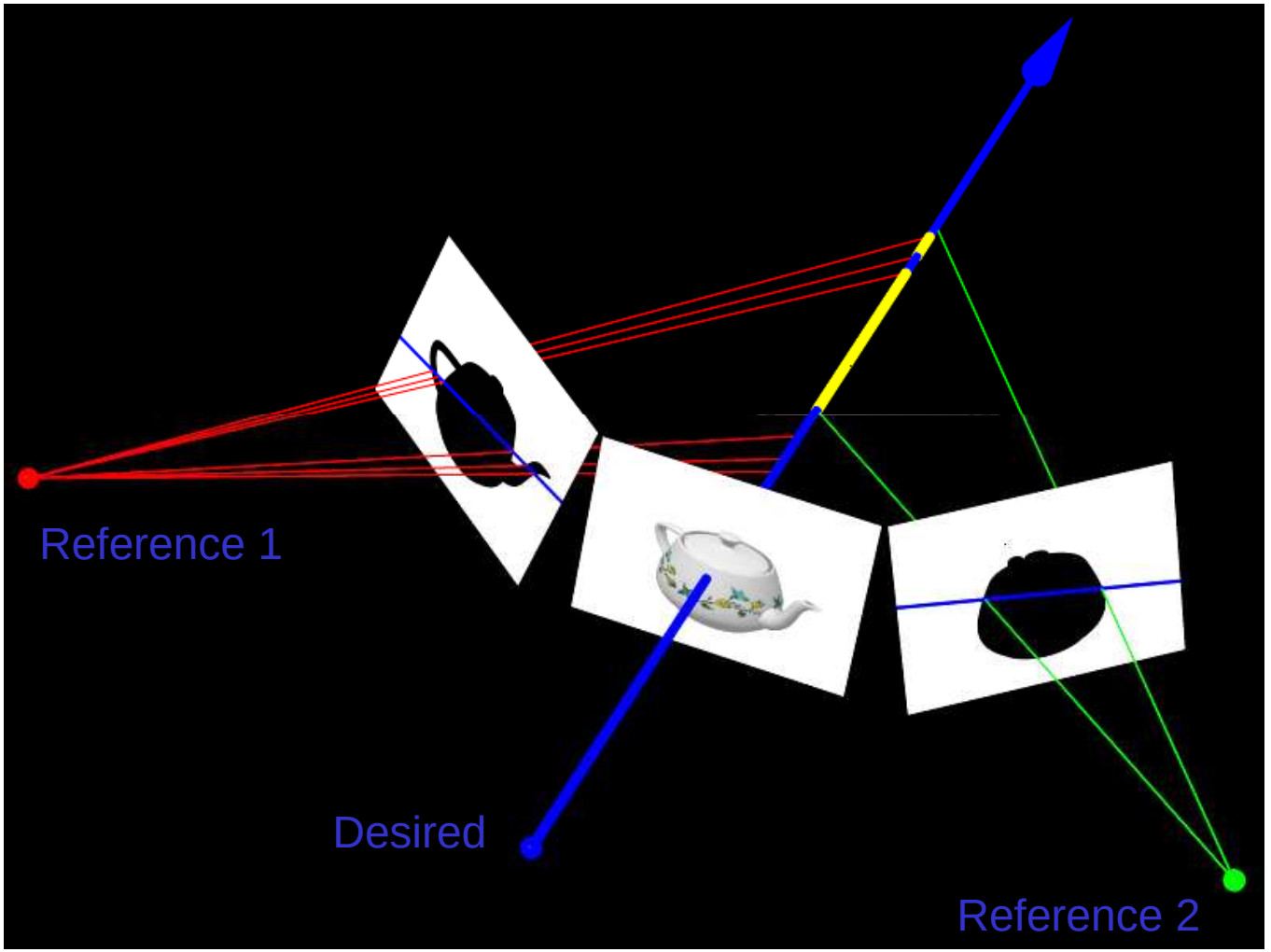
- pros: can be useful for many tasks based on pixel rendering
- cons: view-dependent representation



Matusik&al Siggraph 00

ETH

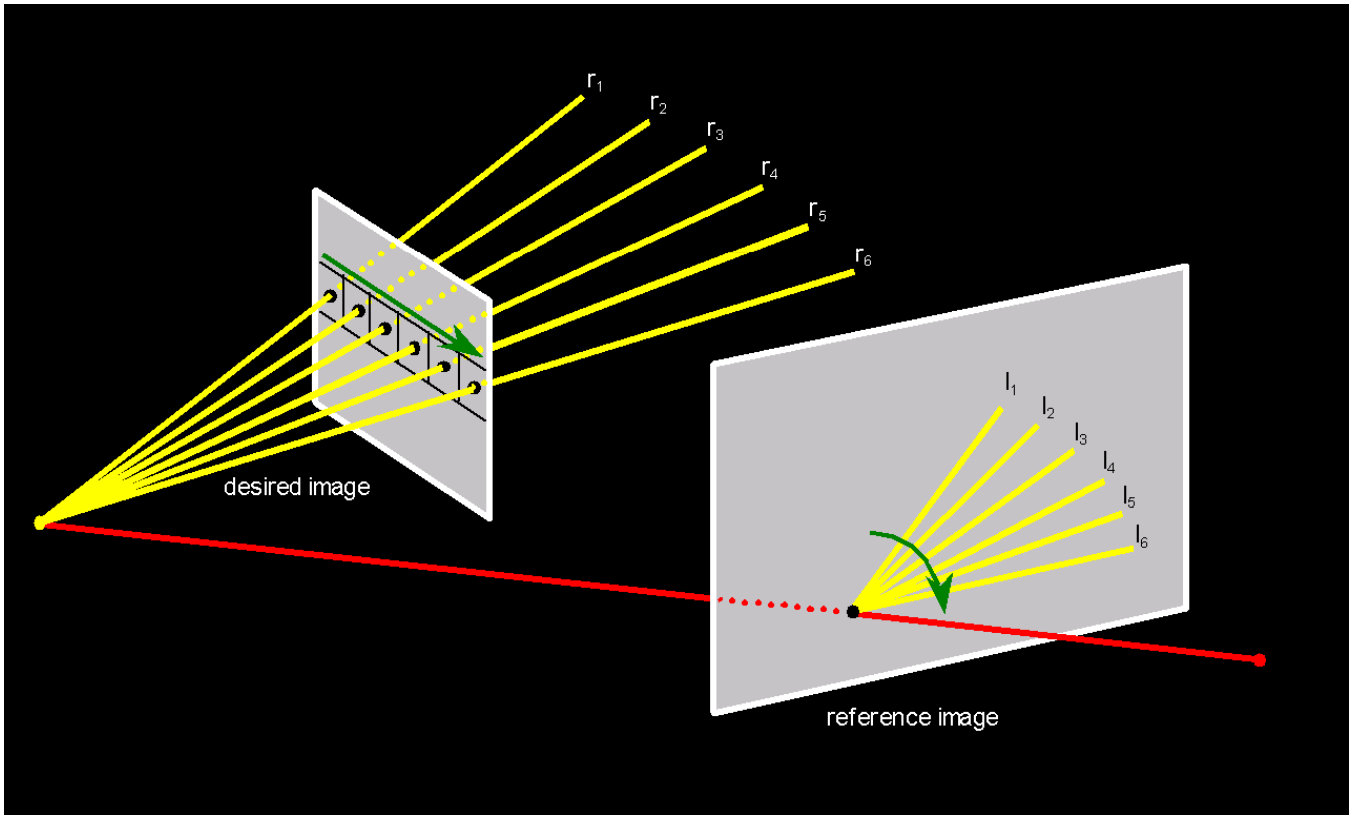
Image-Based Computation



ETH

Observation

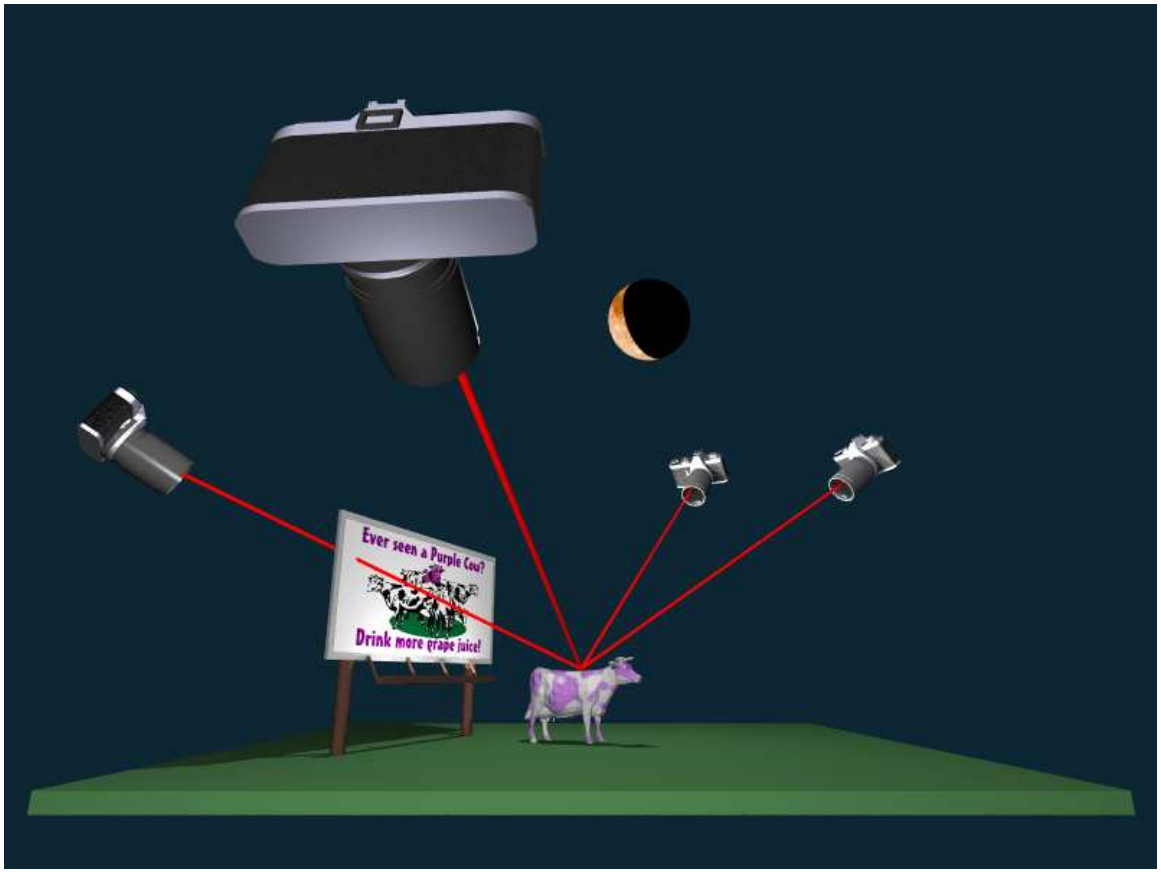
- Incremental computation along scanlines



ETH

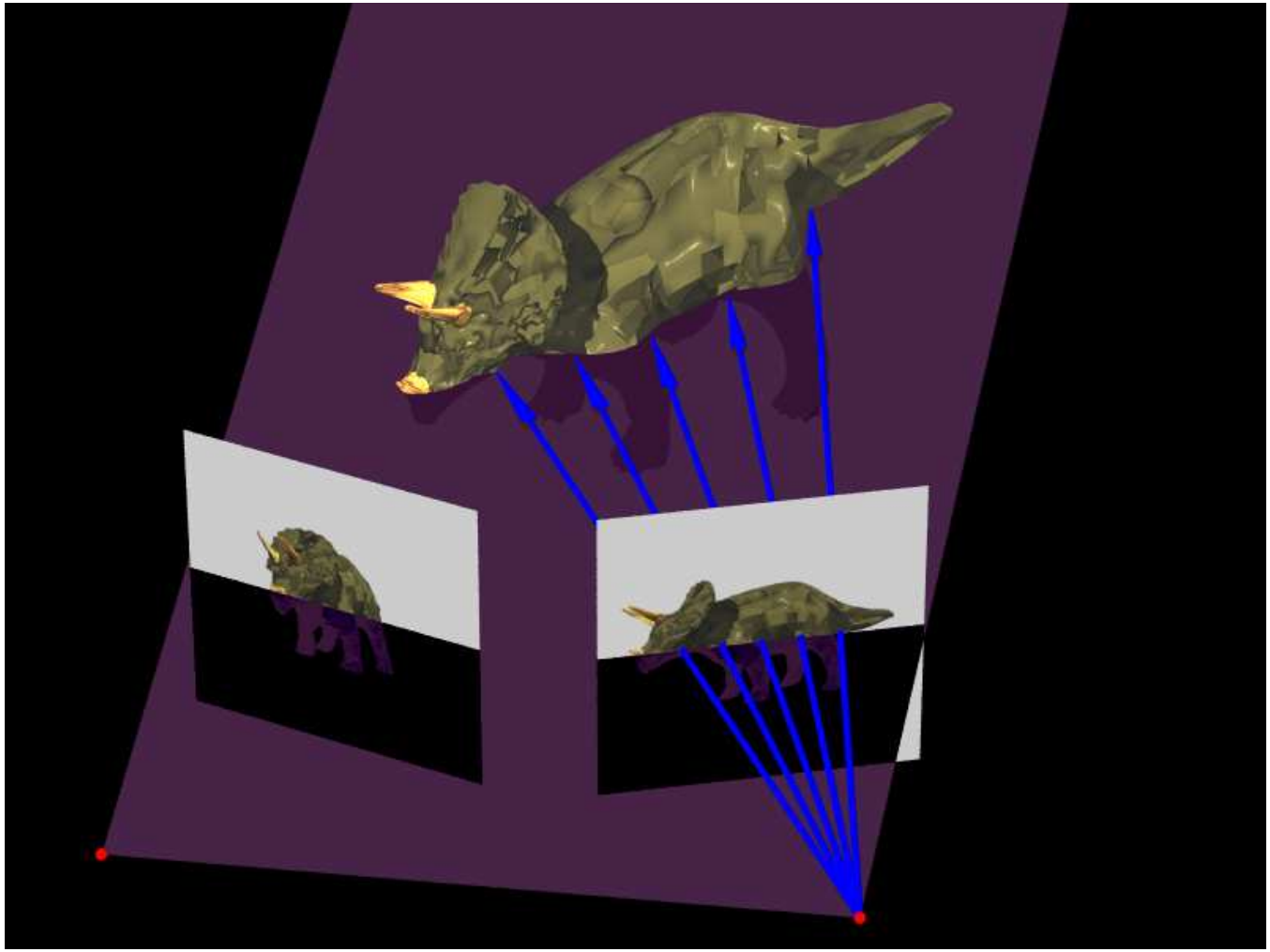
Shading Algorithm

- A view-dependent strategy



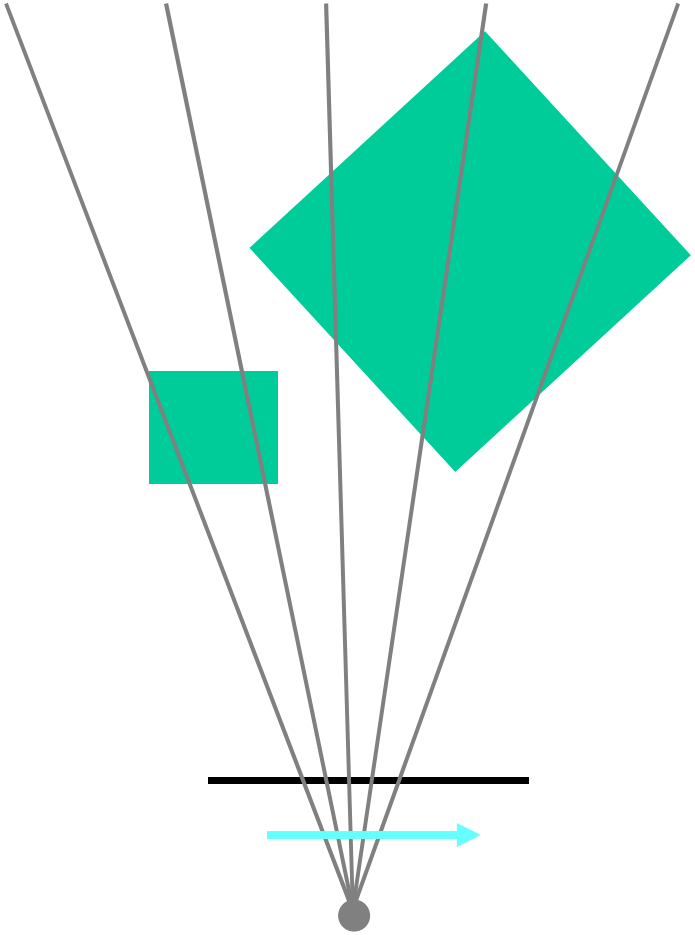
ETH

Visibility Algorithm



ETH

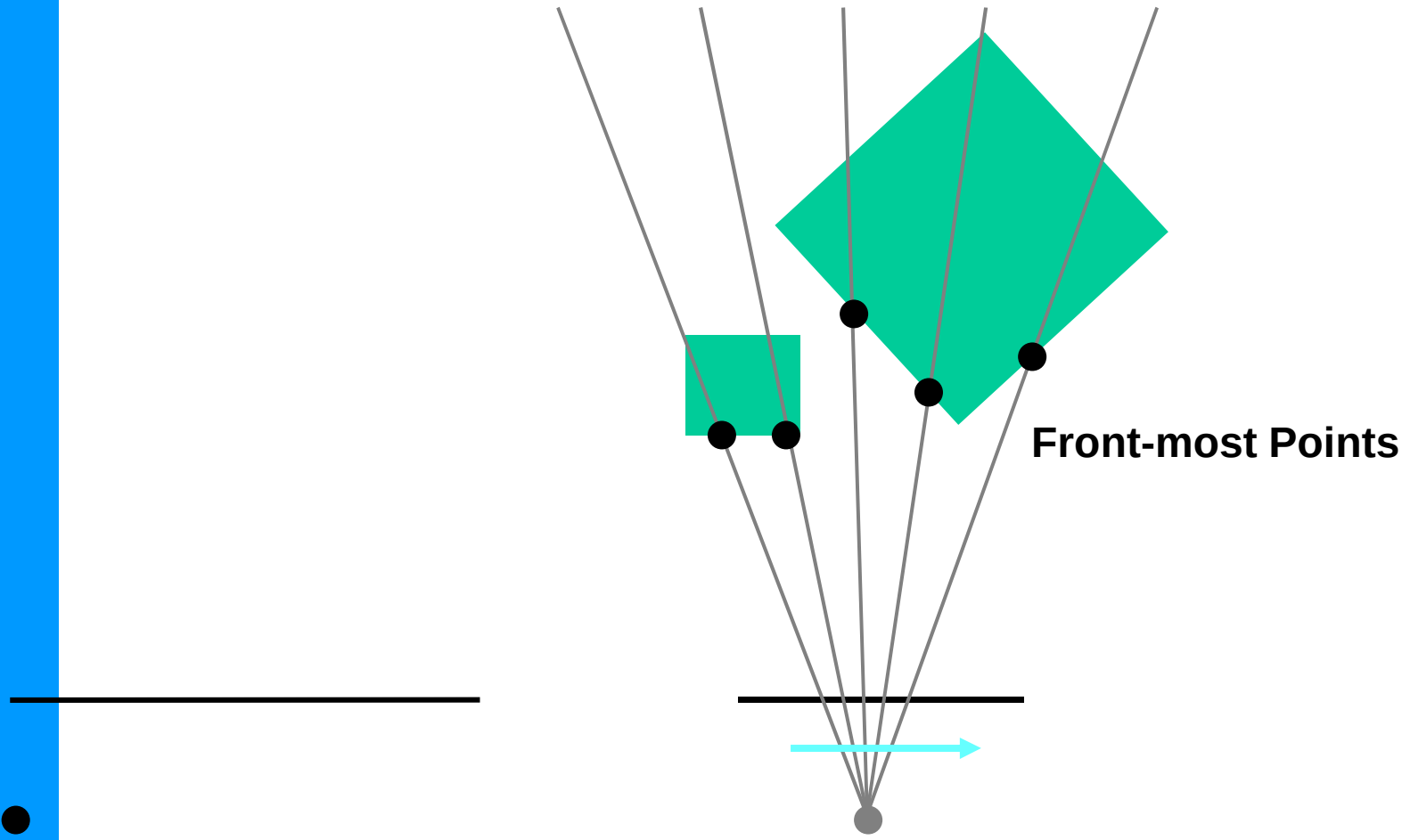
Visibility in 2D



Reference view

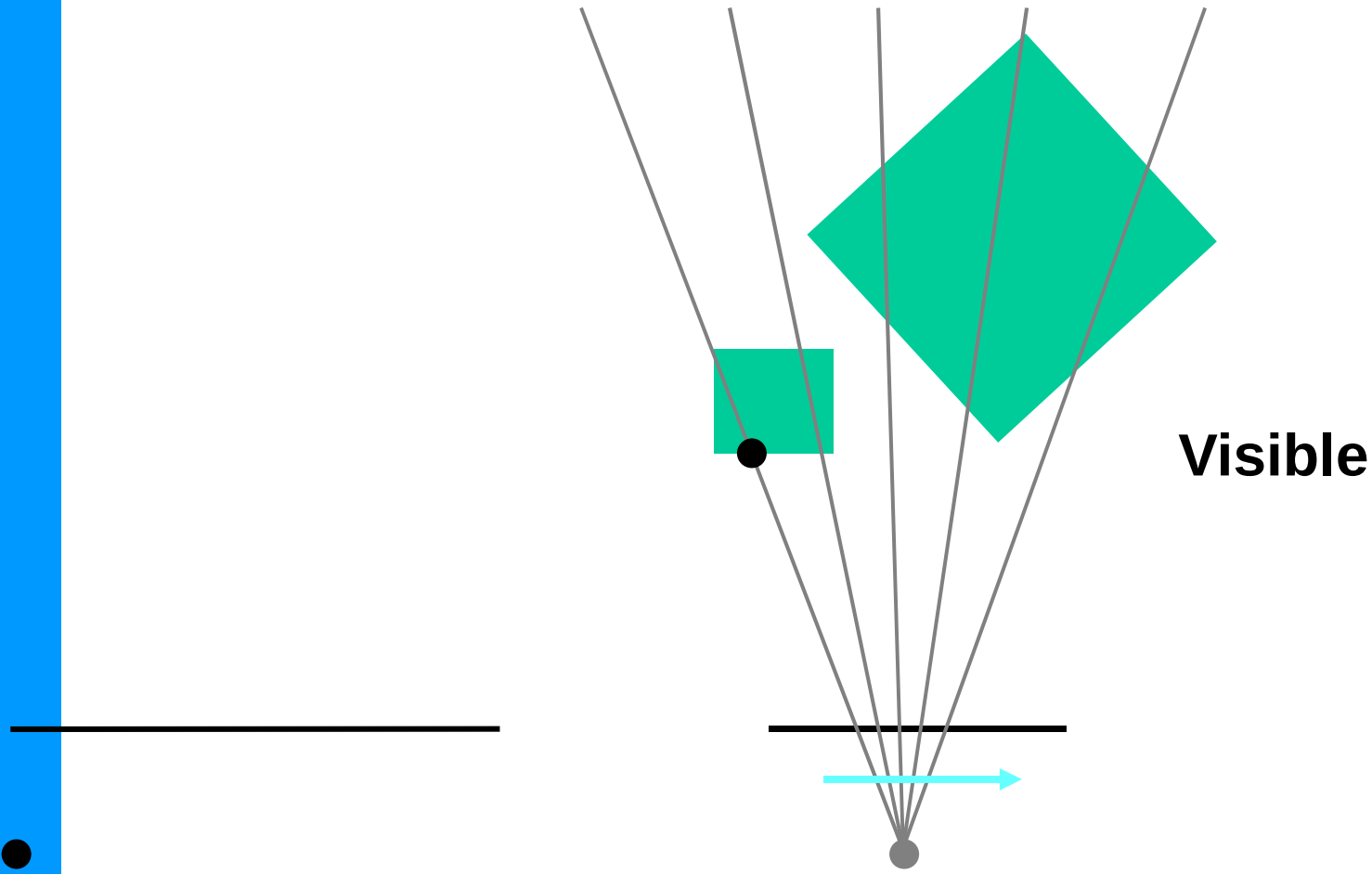
Desired view

Visibility in 2D



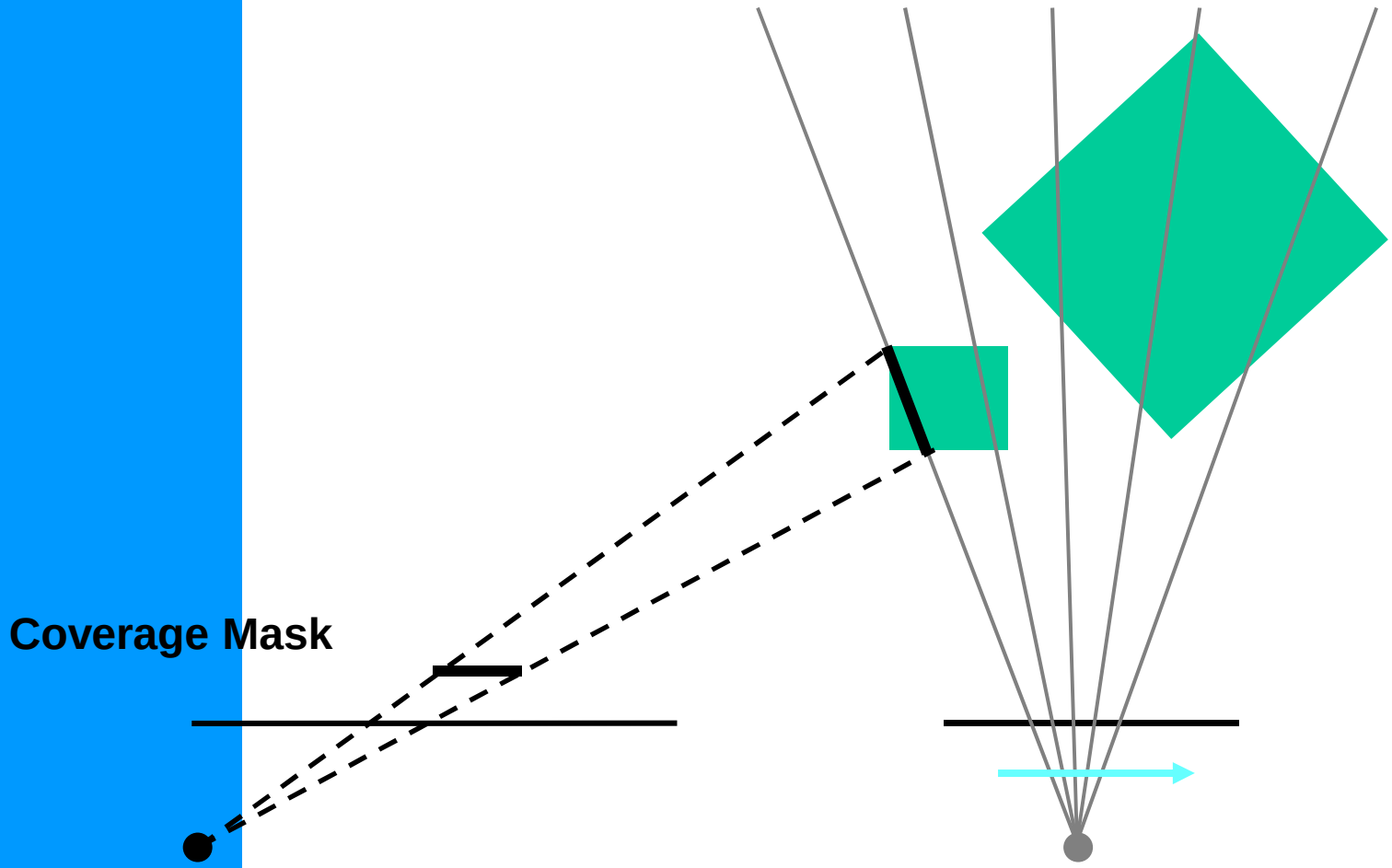
ETH

Visibility in 2D



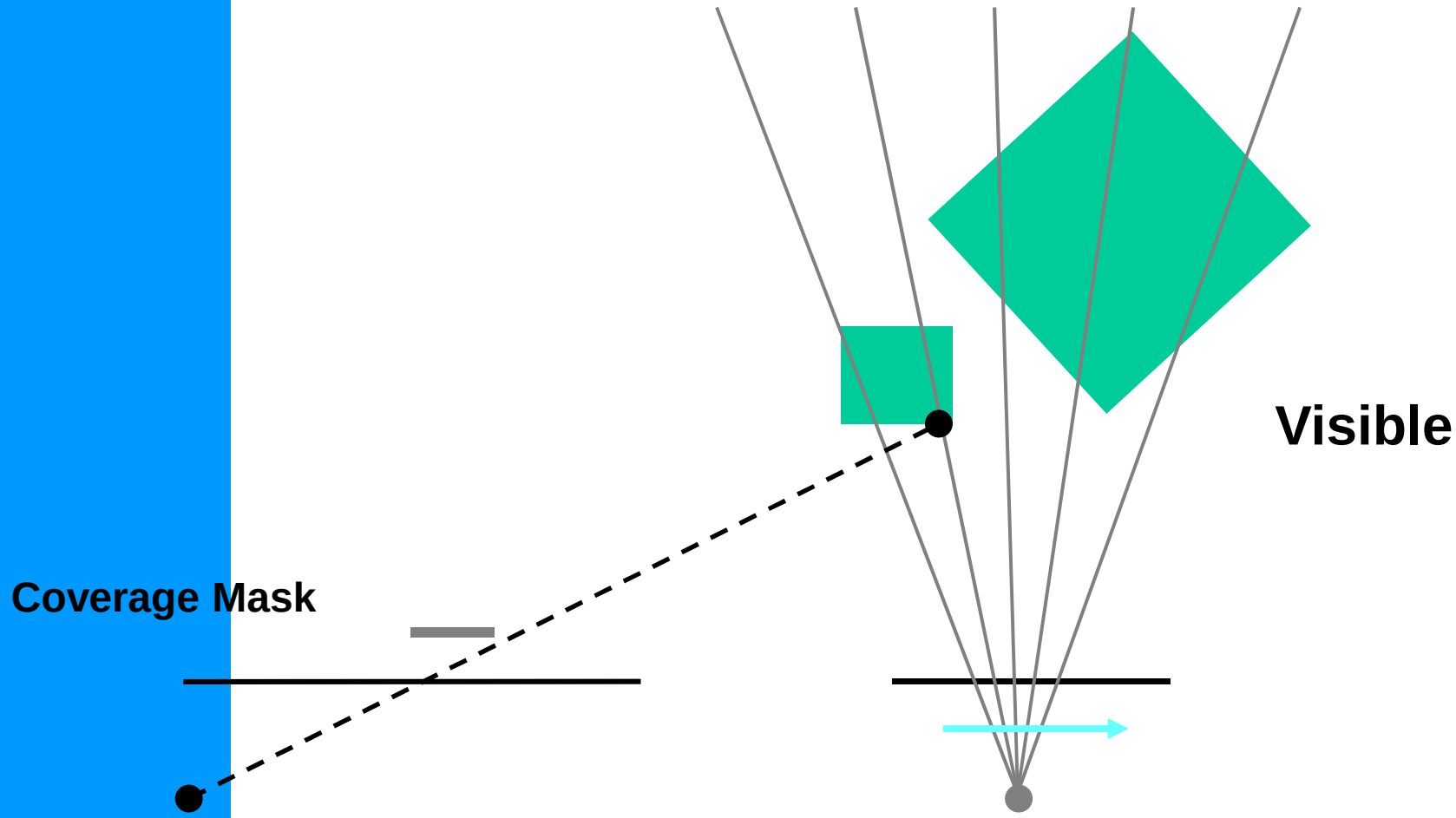
ETH

Visibility in 2D



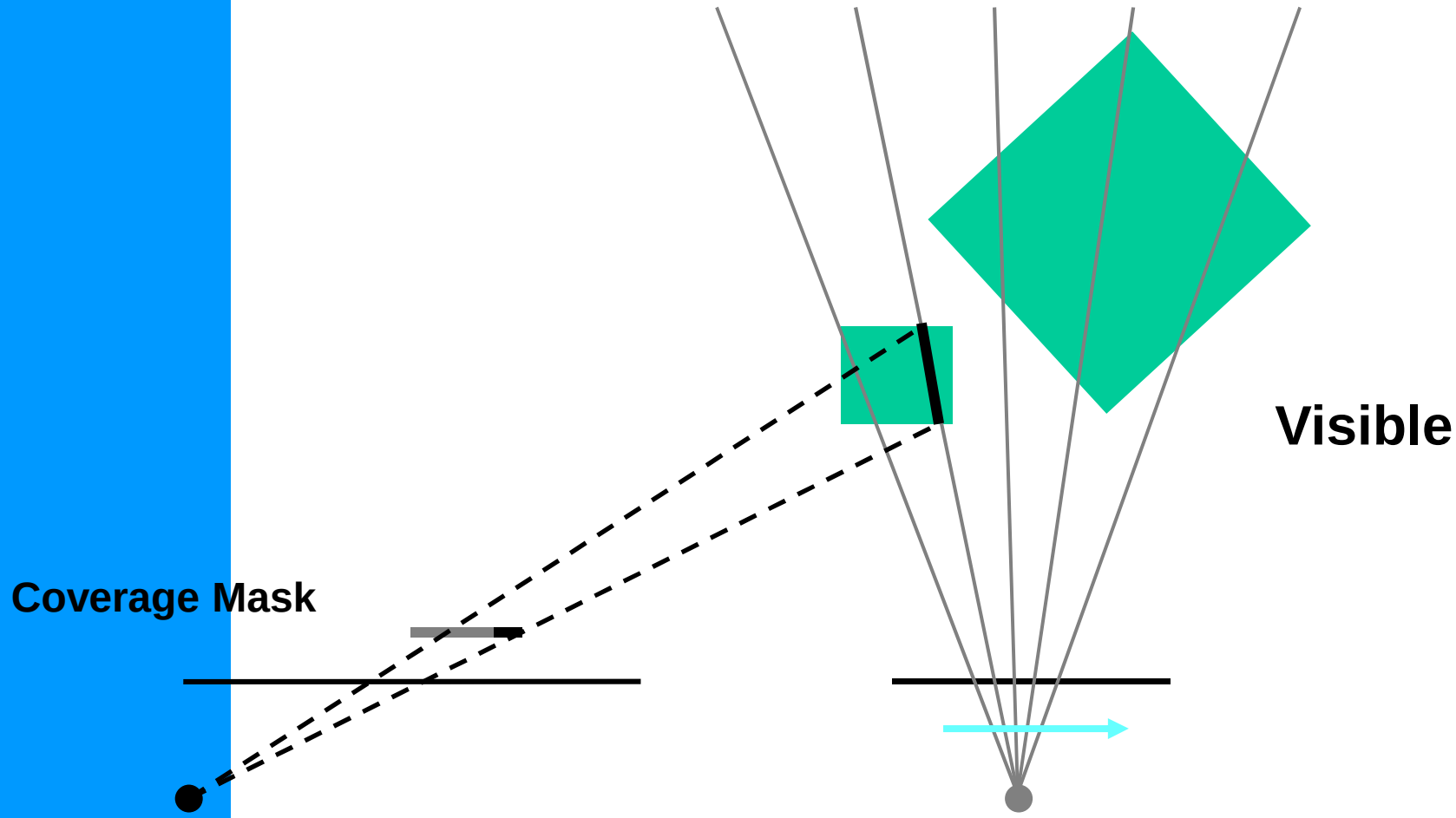
ETH

Visibility in 2D



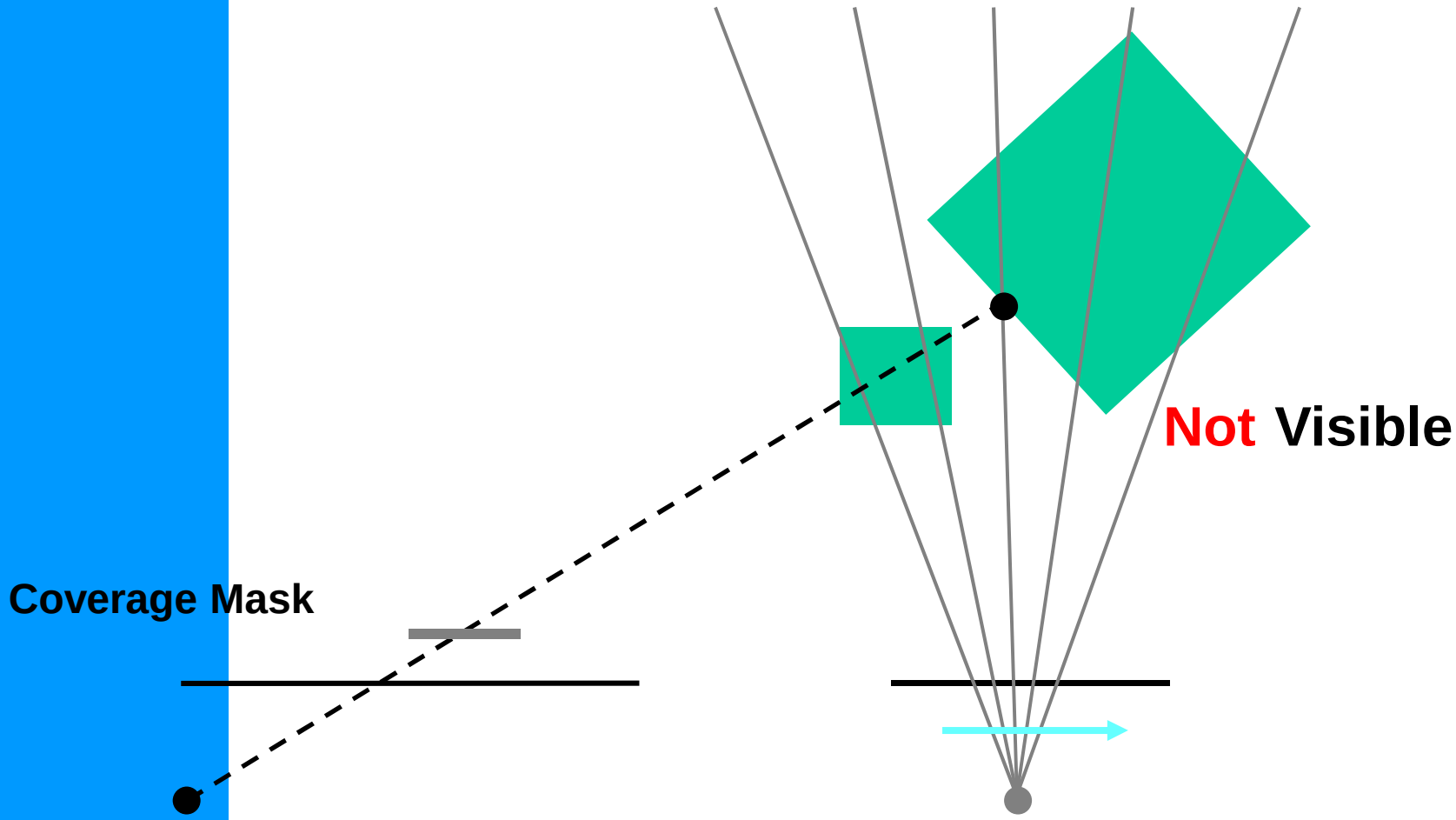
ETH

Visibility in 2D



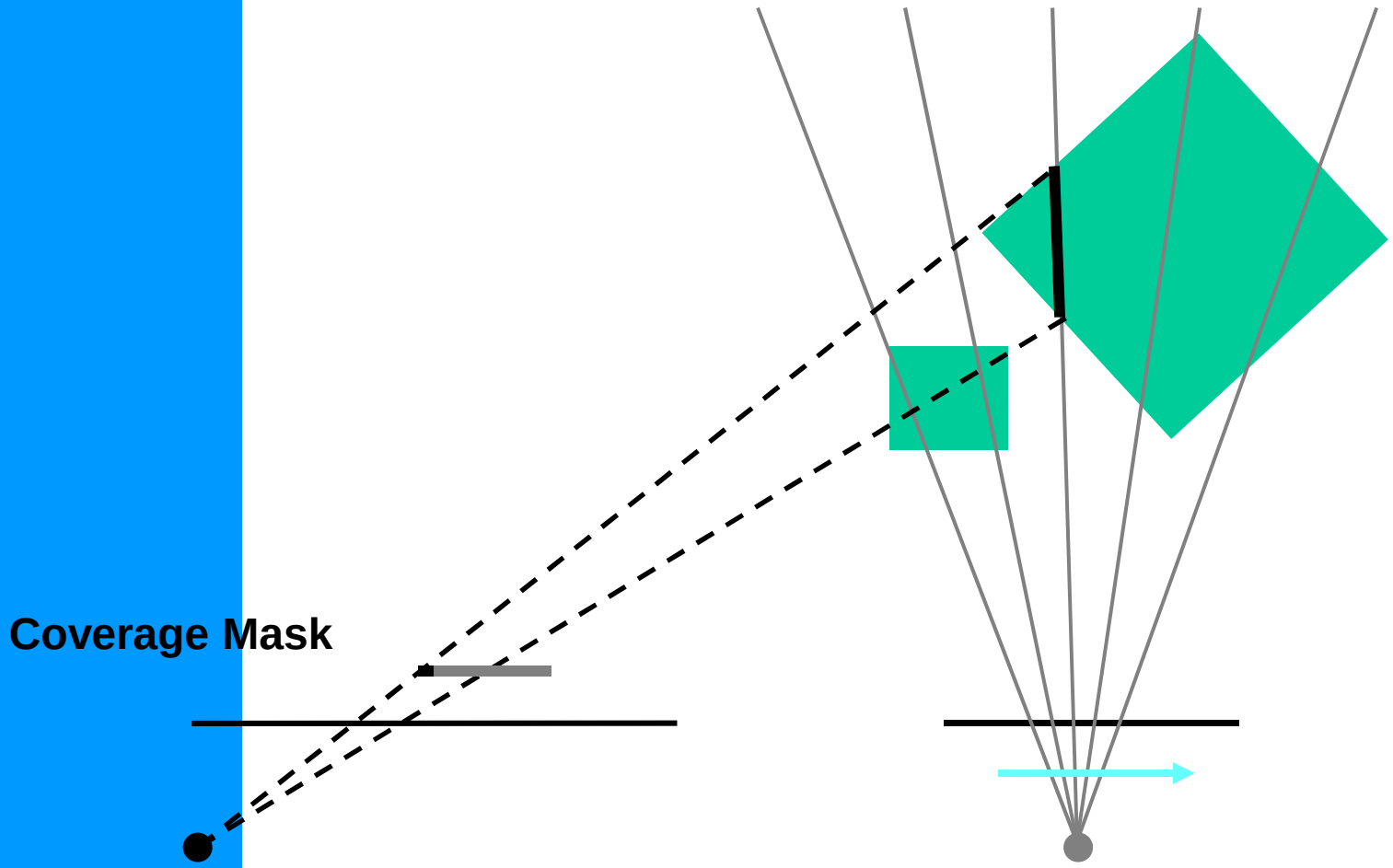
ETH

Visibility in 2D



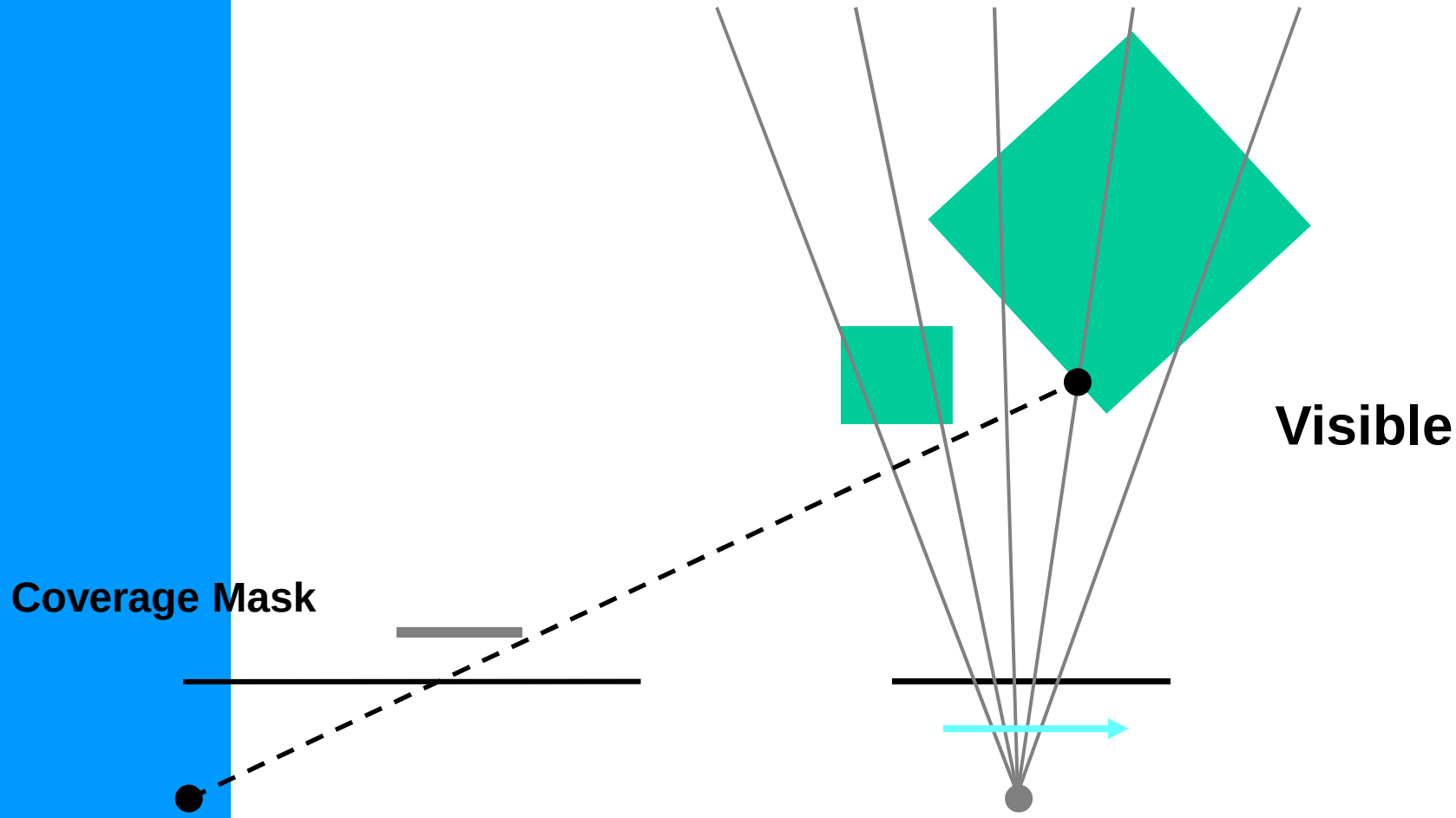
ETH

Visibility in 2D



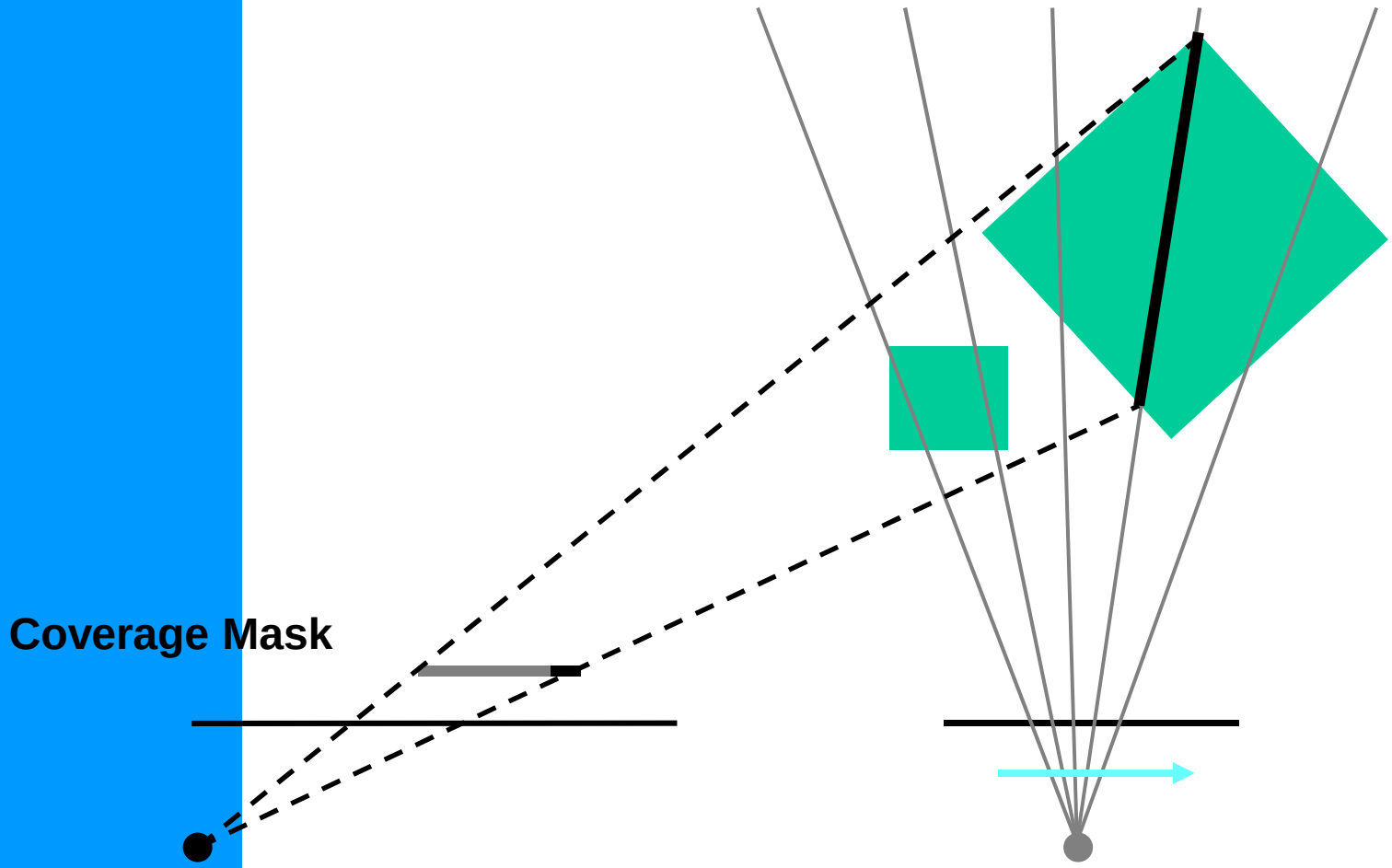
ETH

Visibility in 2D



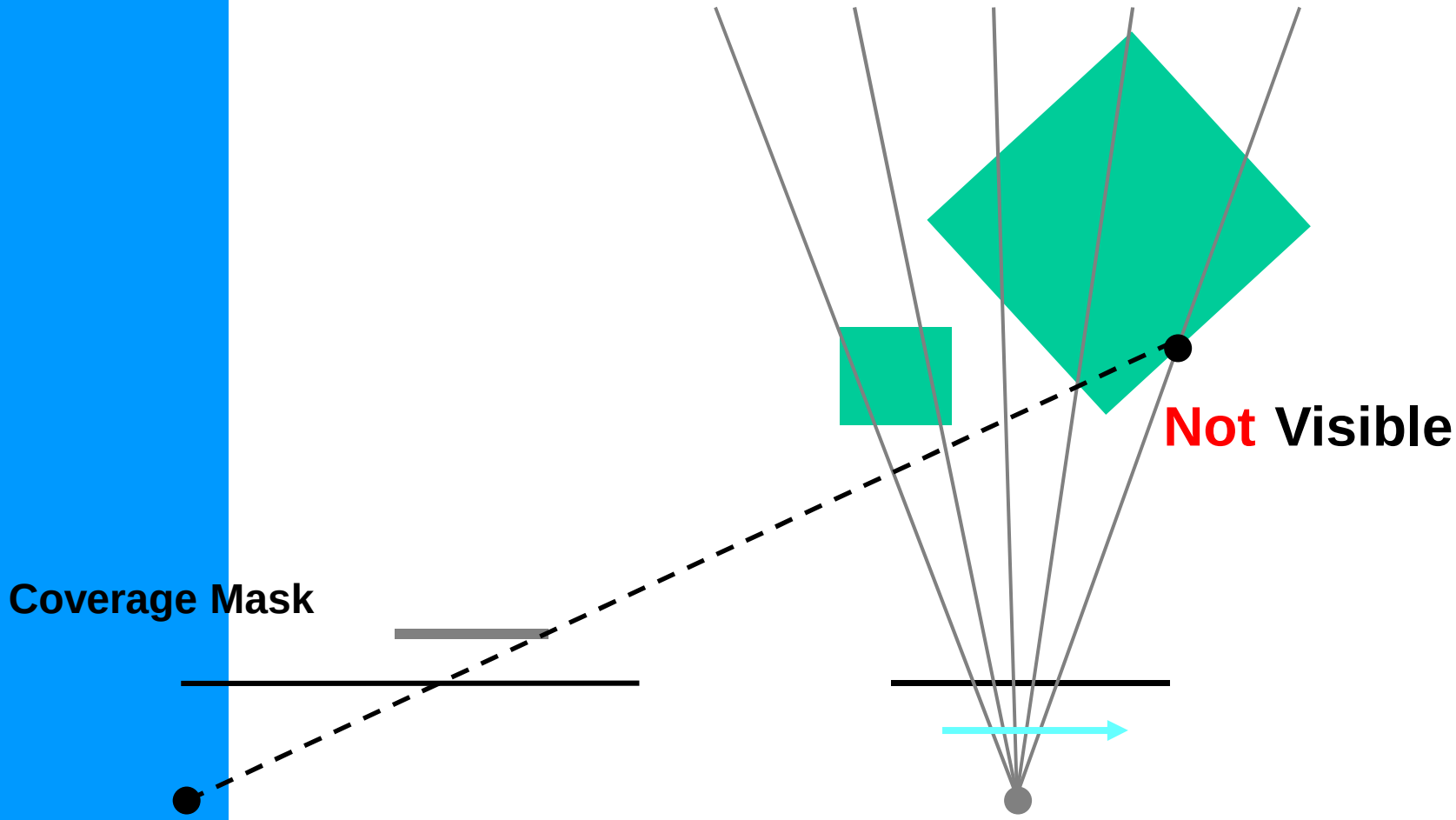
ETH

Visibility in 2D



ETH

Visibility in 2D



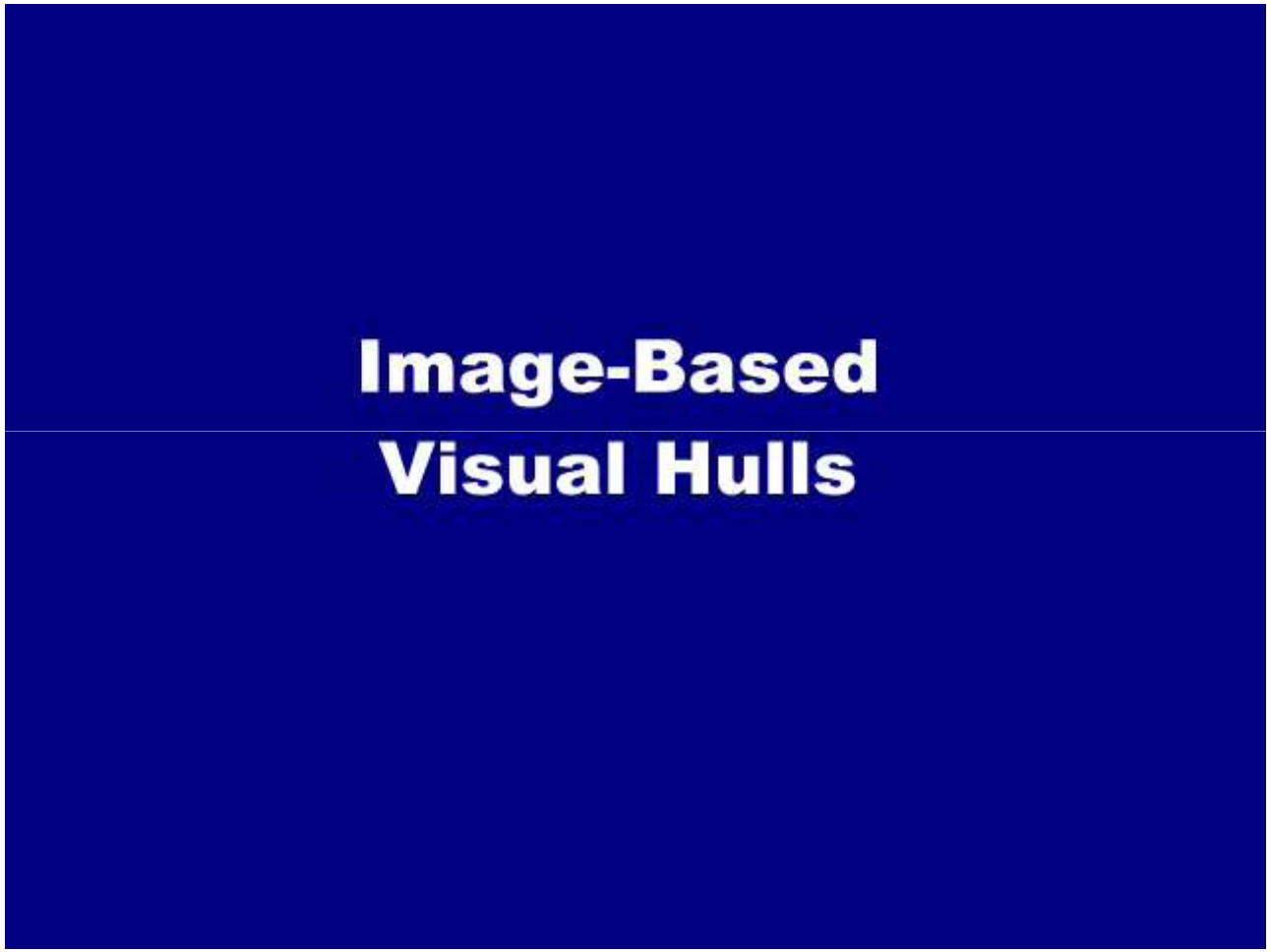
ETH

IBVH Results

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

ETH

IBVH Results



Movie

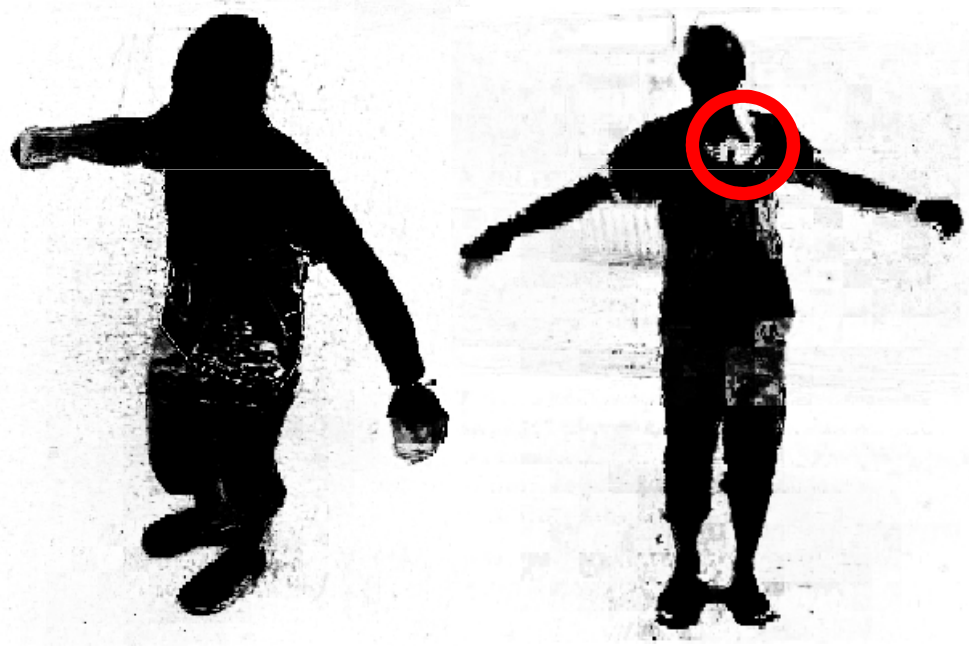
ETH

Outline

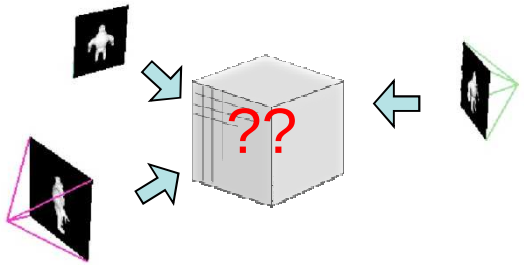
- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes

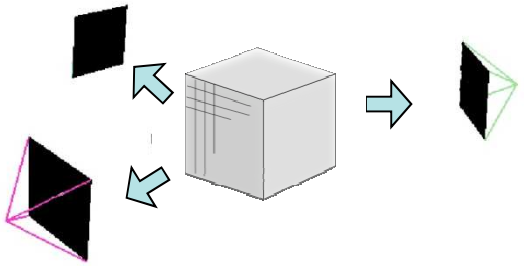
ETH

What if my silhouettes are noisy?



What if my silhouettes are noisy?



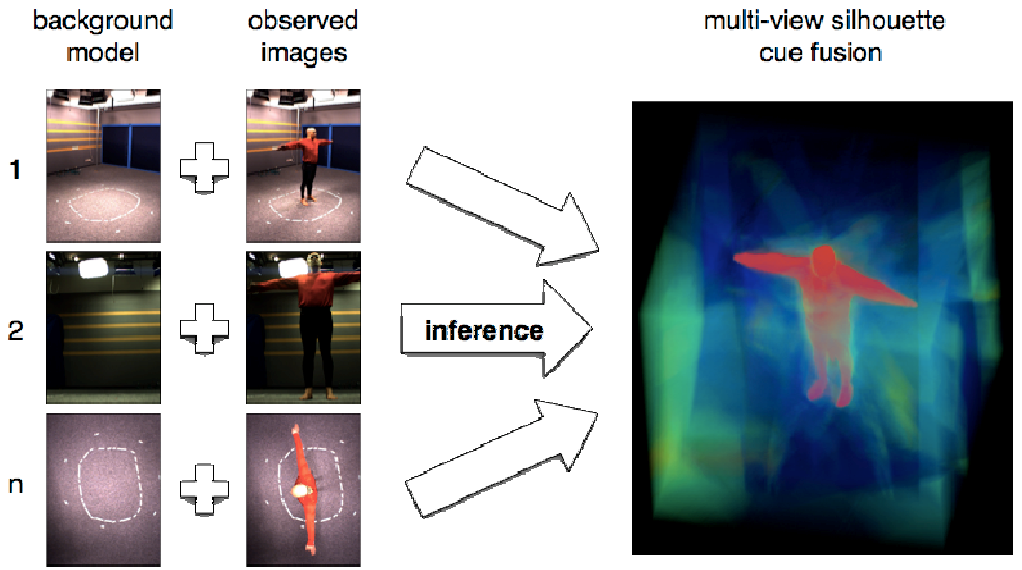


- Idea: we wish to find the content of the scene from images, as a probability grid
- Modeling the forward problem - explaining image observations given the grid state - is easy. It can be accounted for in a *sensor model*.
- Bayesian inference enables the formulation of our initial inverse problem from the sensor model
- Simplification for tractability: independent analysis and processing of voxels

ETH

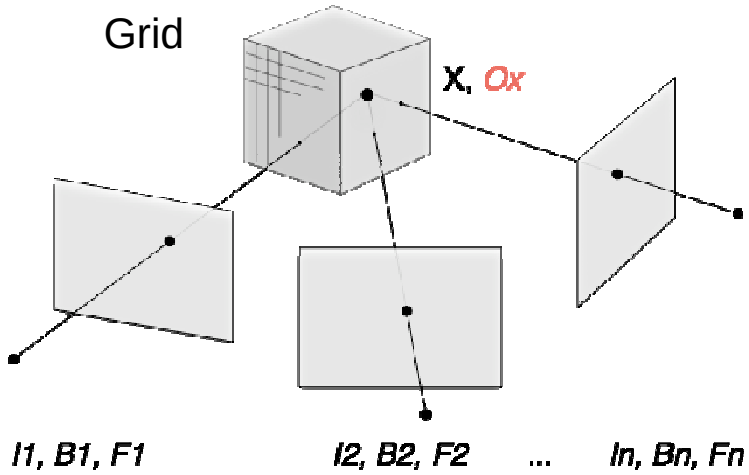
Idea

- Unreliable silhouettes: do not make decision about their location
- Do sensor fusion: use all image information simultaneously



ETH

Modeling



Sensor model:

$$P(I \mid O_X) = \sum_F P(I \mid F, B) P(F \mid \mathrm{O}_X)$$

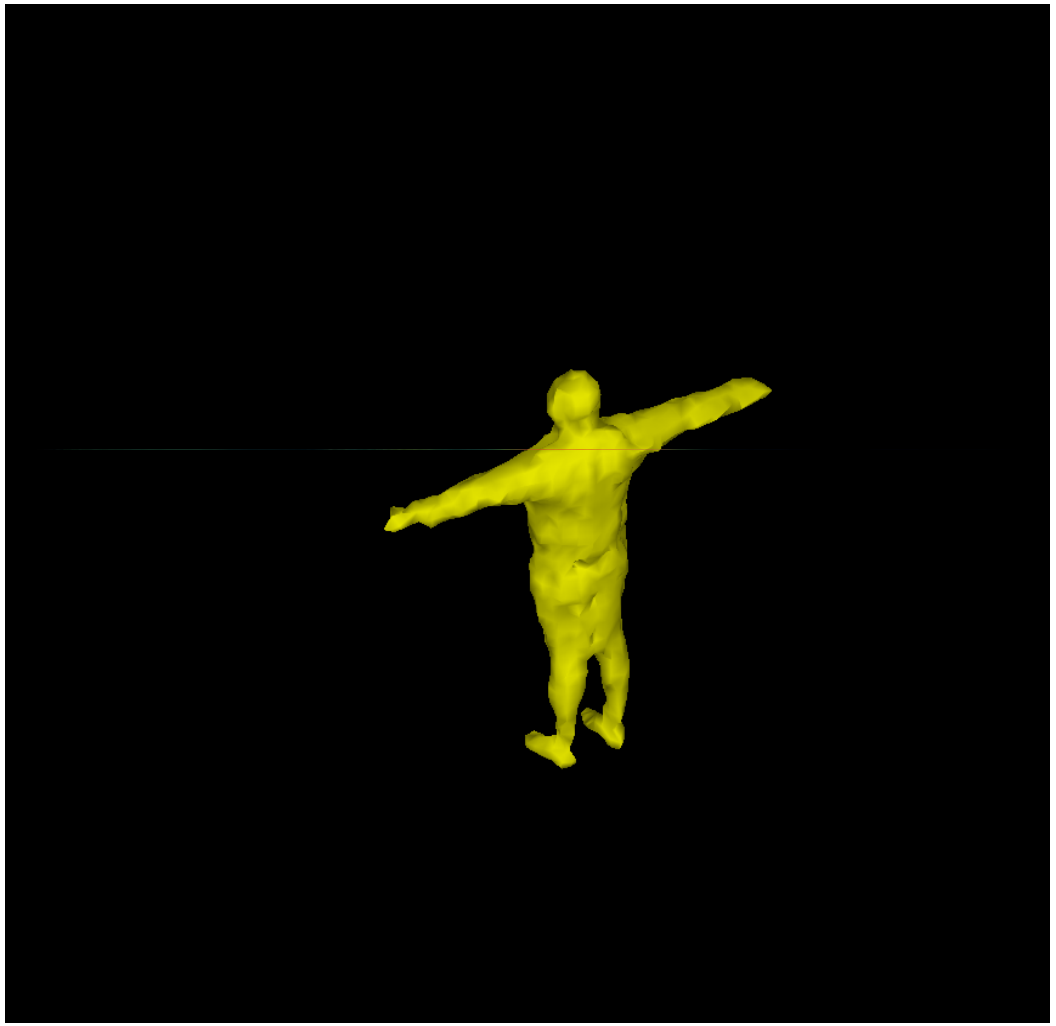
Inference:

$$P(O_X \mid I) = \frac{\prod_{img,pixel} P(I_{img,pixel} \mid \mathrm{O}_X)}{\sum_{O_X} \prod_{img,pixel} P(I_{img,pixel} \mid \mathrm{O}_X)}$$

- I: color information in images
- B: background color model
- F: silhouette detection variable (0 or 1): hidden
- Ox: occupancy at voxel X (0 ou 1)

ETH

Visualization



ETH