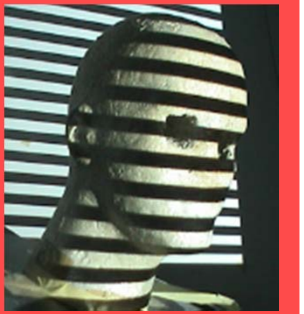
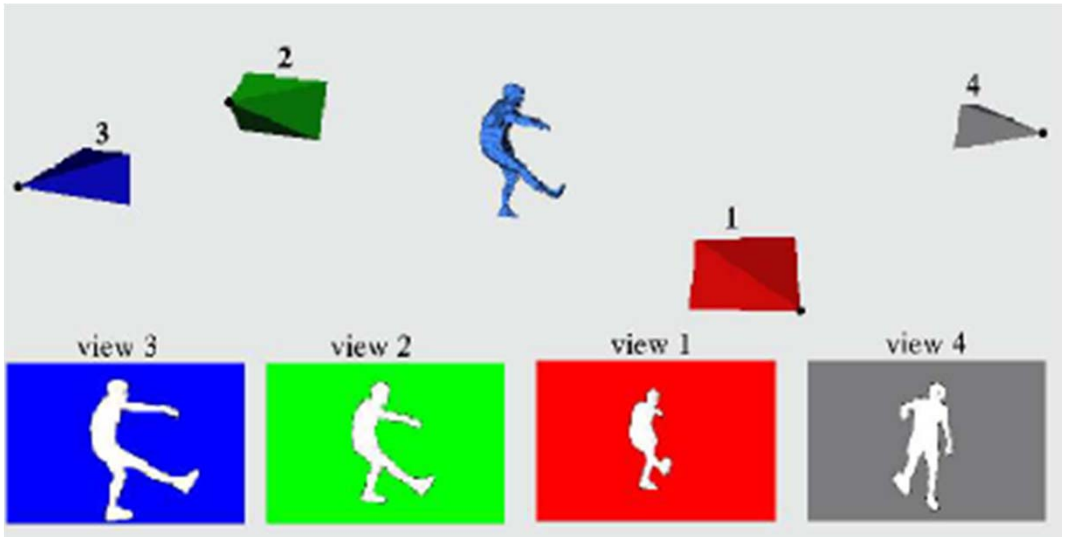




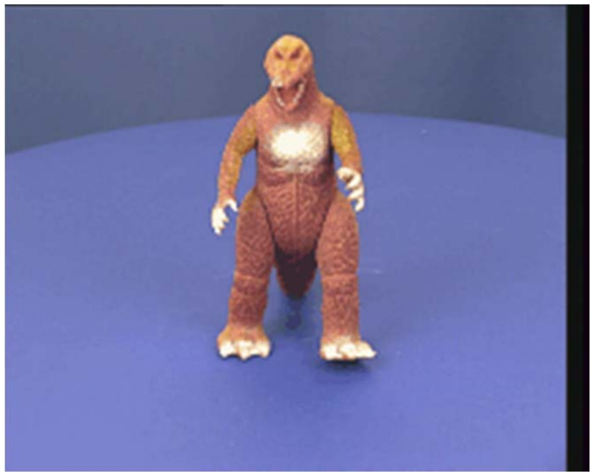
# Shape from Silhouettes II

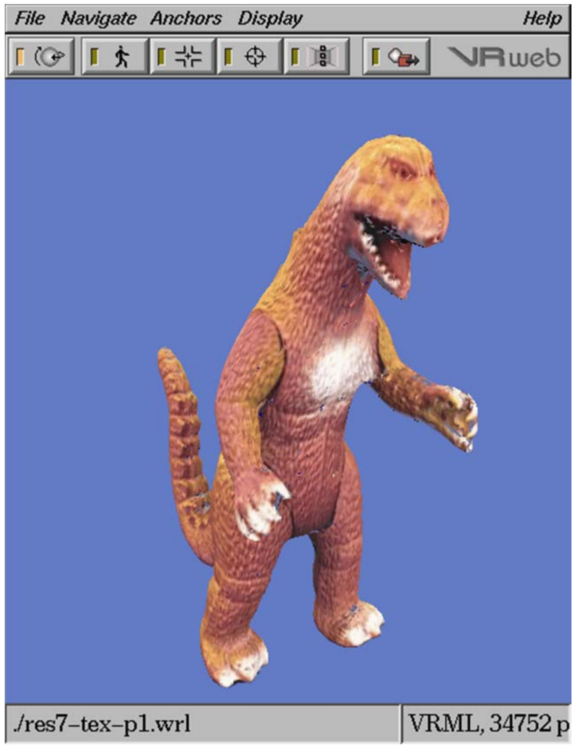
Guido Gerig

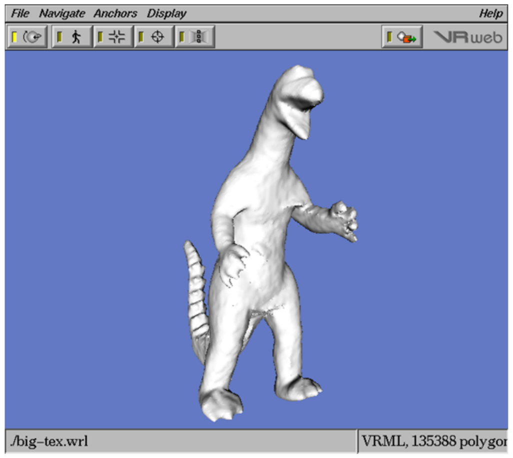
CS 6320, S2013

(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







Slides from Lazebnik, Matusik Yerex and others

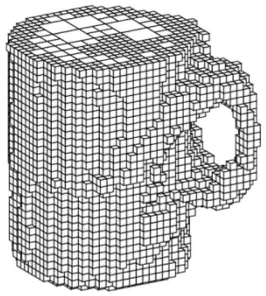
Automatic 3D Model Construction for Turn-Table Sequences, A.W. Fitzgibbon, G. Cross, and A. Zisserman, SMILE 1998

## Outline

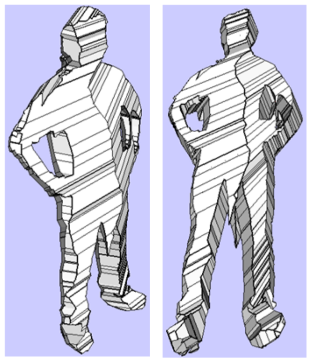
- Silhouettes
  - basic concepts
  - extract silhouettes
  - fundamentals about using silhouettes
  - reconstruct shapes from silhouettes
  - use uncertain silhouettes
  - calibrate from silhouettes
- Perspectives and interesting ideas

# Algorithms

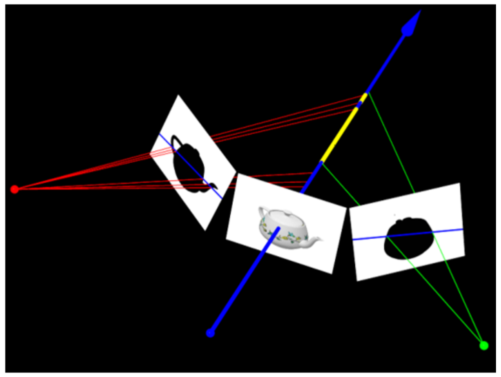
- Standard voxel based method



- Exact polyhedral methods



- Image-based visual hulls



# IBVH: Image based visual hulls

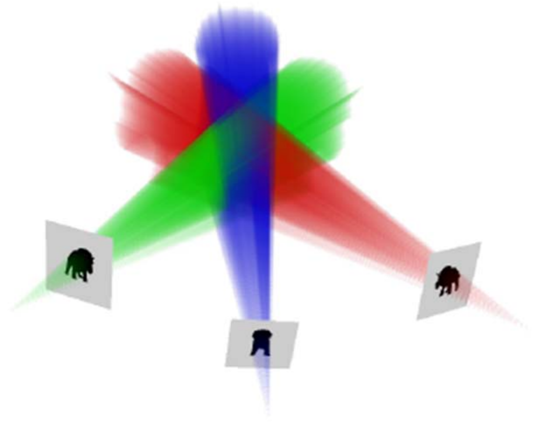


Figure 1 - The intersection of silhouette cones defines an approximate geometric representation of an object called the visual hull. A visual hull has several desirable properties: it contains the actual object, and it has consistent silhouettes.

- VH without constructing auxiliary geometric/volumetric representation
- All steps of rendering are in 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

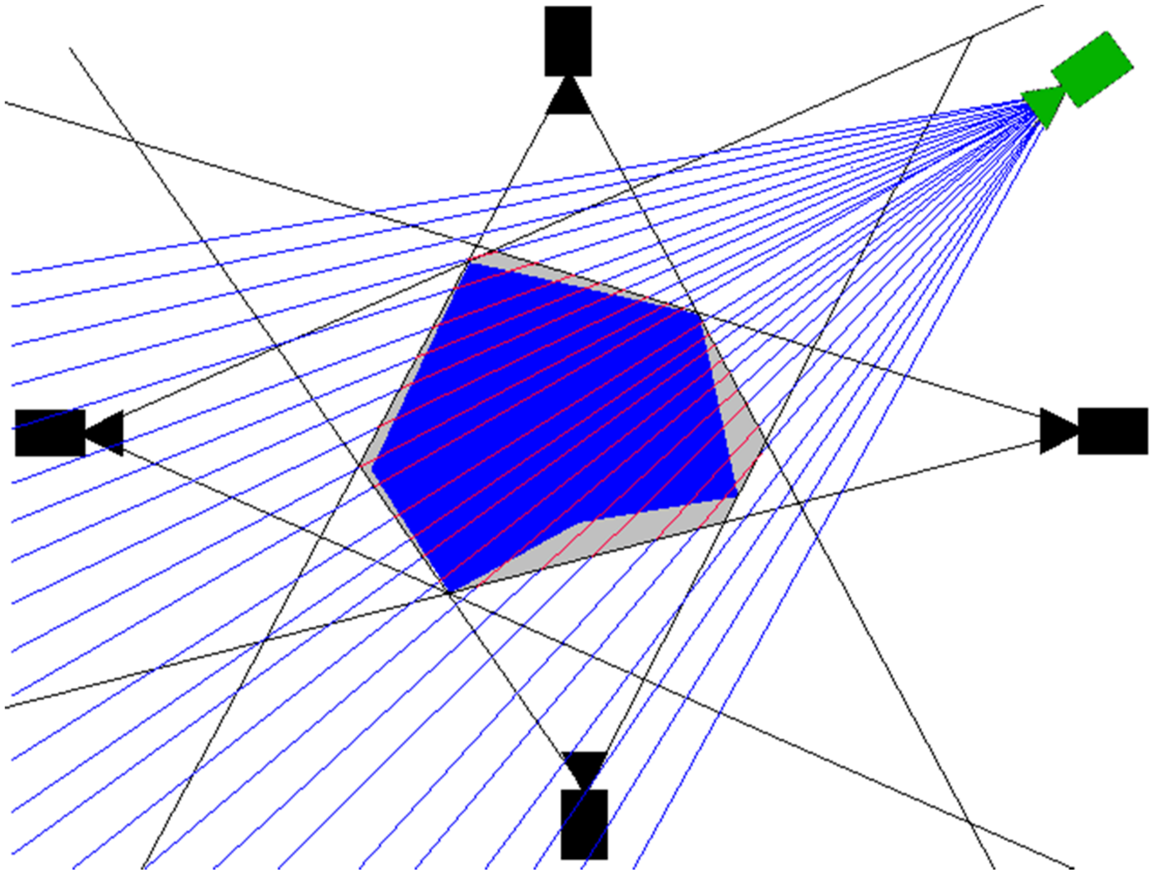
Wojciech Matusik, *Image Based Visual Hulls*

## 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

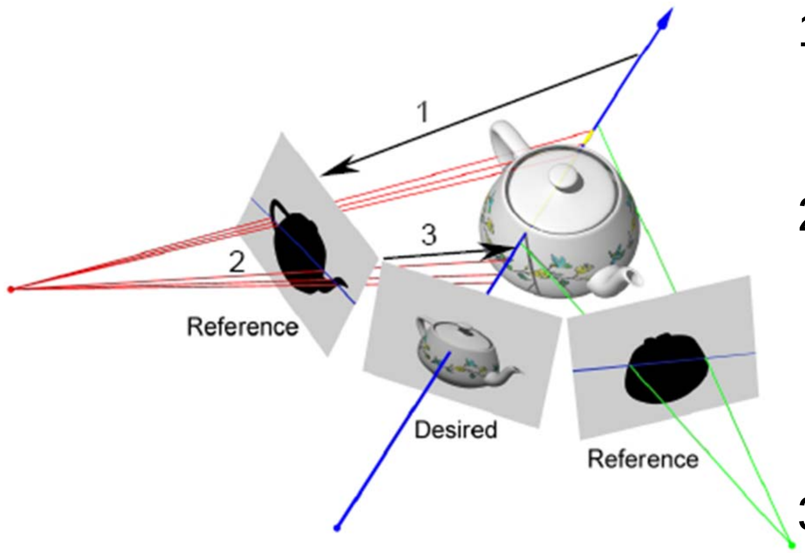
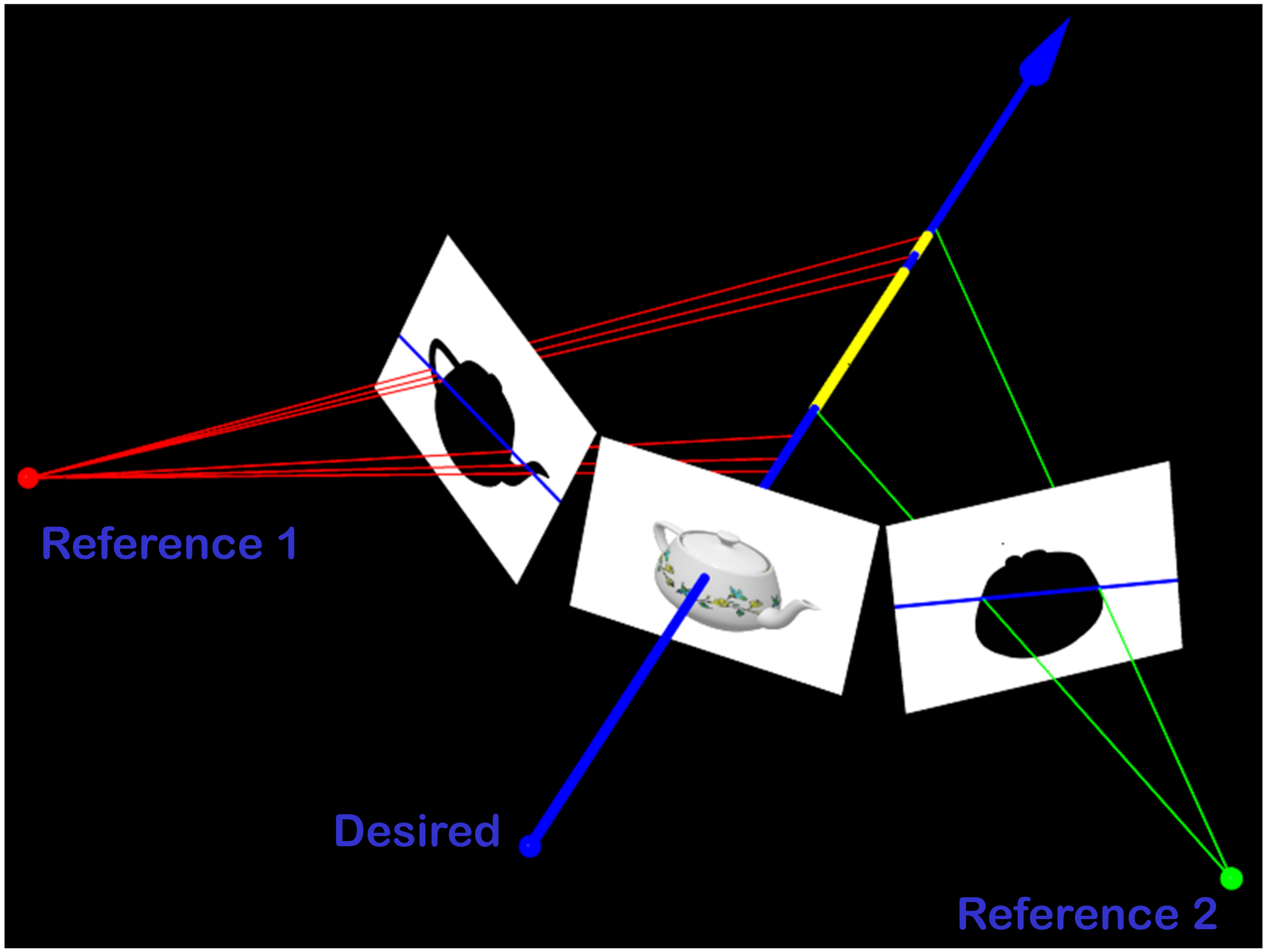


Figure 2 – Computing the IBVH involves three steps. First, the desired ray is projected onto a reference image. Next, the intervals where the projected ray crosses the silhouette are determined. Finally, these intervals are lifted back onto the desired ray where they can be intersected with intervals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

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

# Image-Based Computation



# Image-Based Computation

Naïve Algorithm

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

Efficiency for desired view:

n: #pixels per scanline

$O(n^2)$: #pixels in image

k: #images

l: average #times ray intersects silhouette

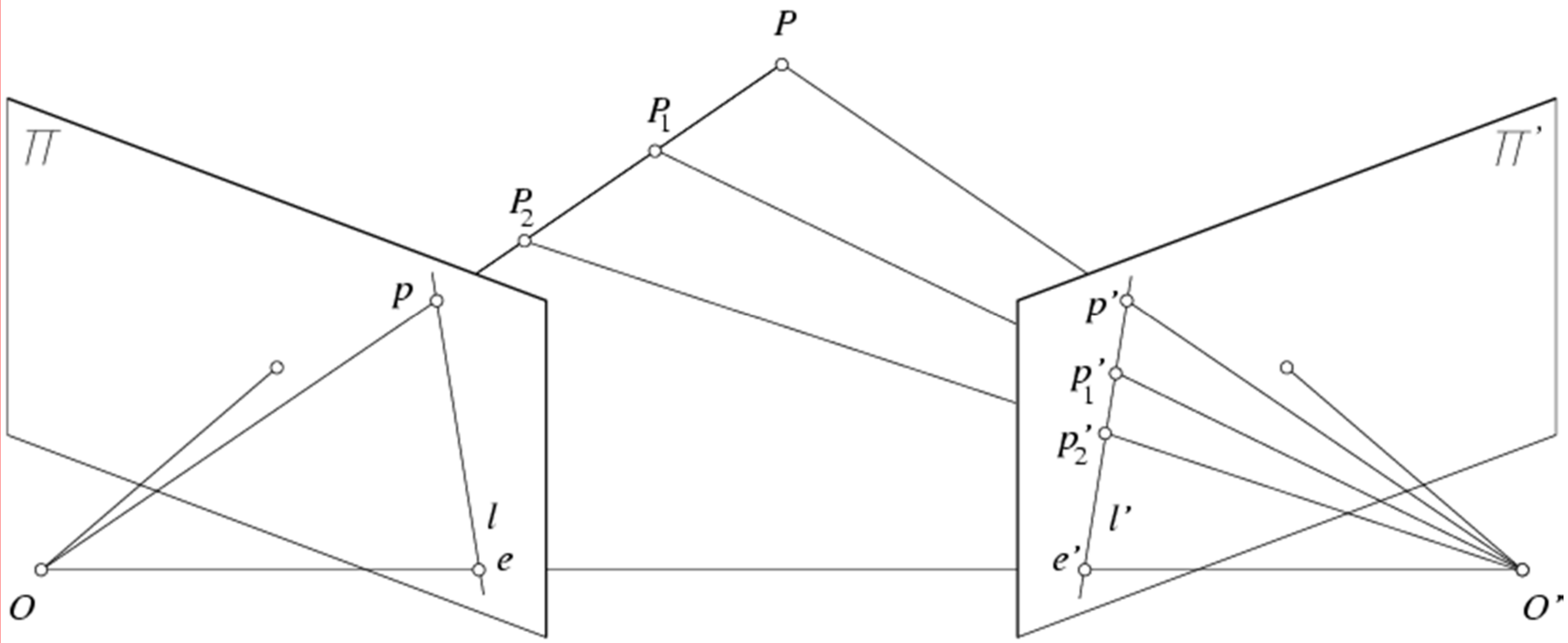
$O(lk)$: #silhouette edges (search along epip lines)

**$O(lkn^3)$: 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(lkn²)**

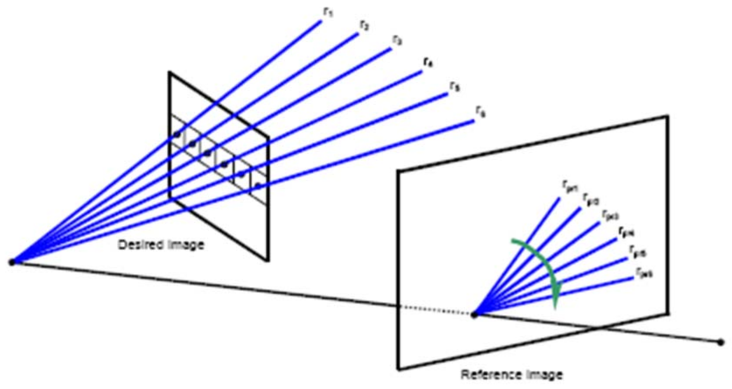


Figure 3 – The pixels of a scanline in the desired image trace out a pencil of line segments in the reference image. An ordered traversal of the scanline will sweep out these segments such that their slope about the epipole varies monotonically.

Traverse scan line of desired view

Each ray project to line through epipole in reference image

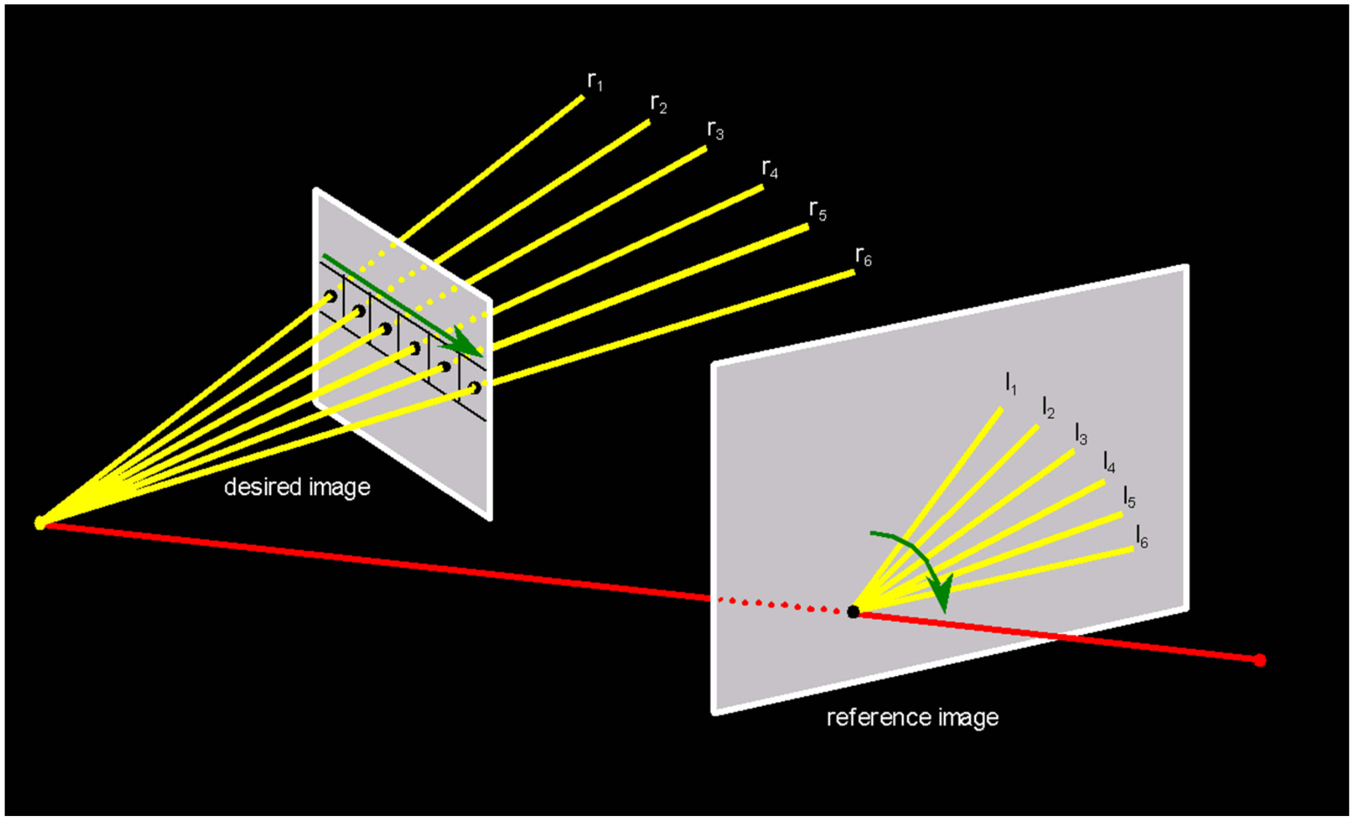
Set of rays in desired image: pencil of epipolar lines in reference image

Monotonic change of slope

Compute silhouette intersections for whole scan line incrementally (trick: presorted silhouette edges relative to epipole)

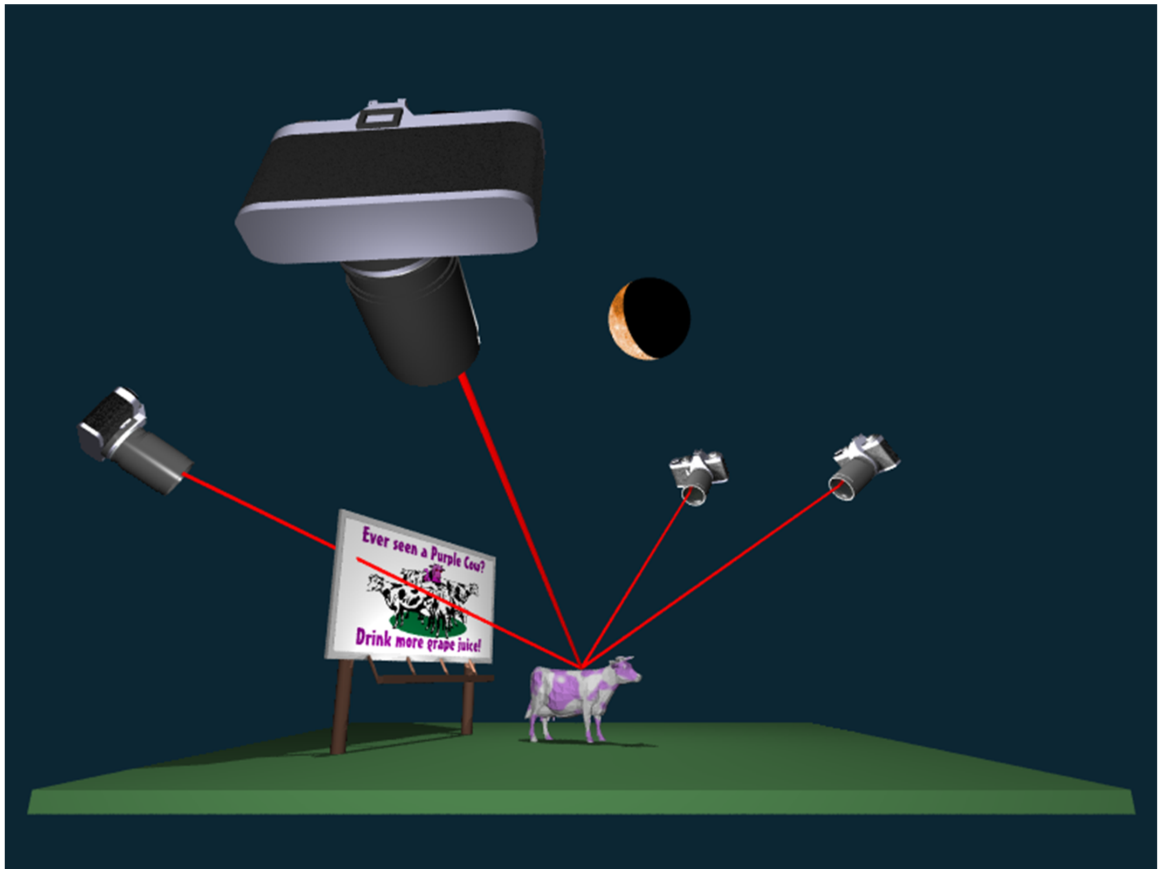
## Observation

- 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

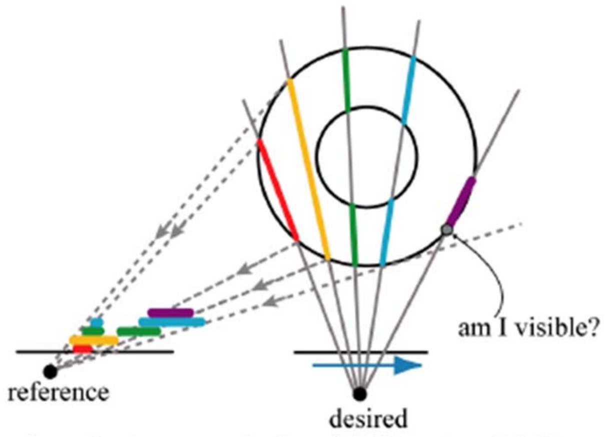
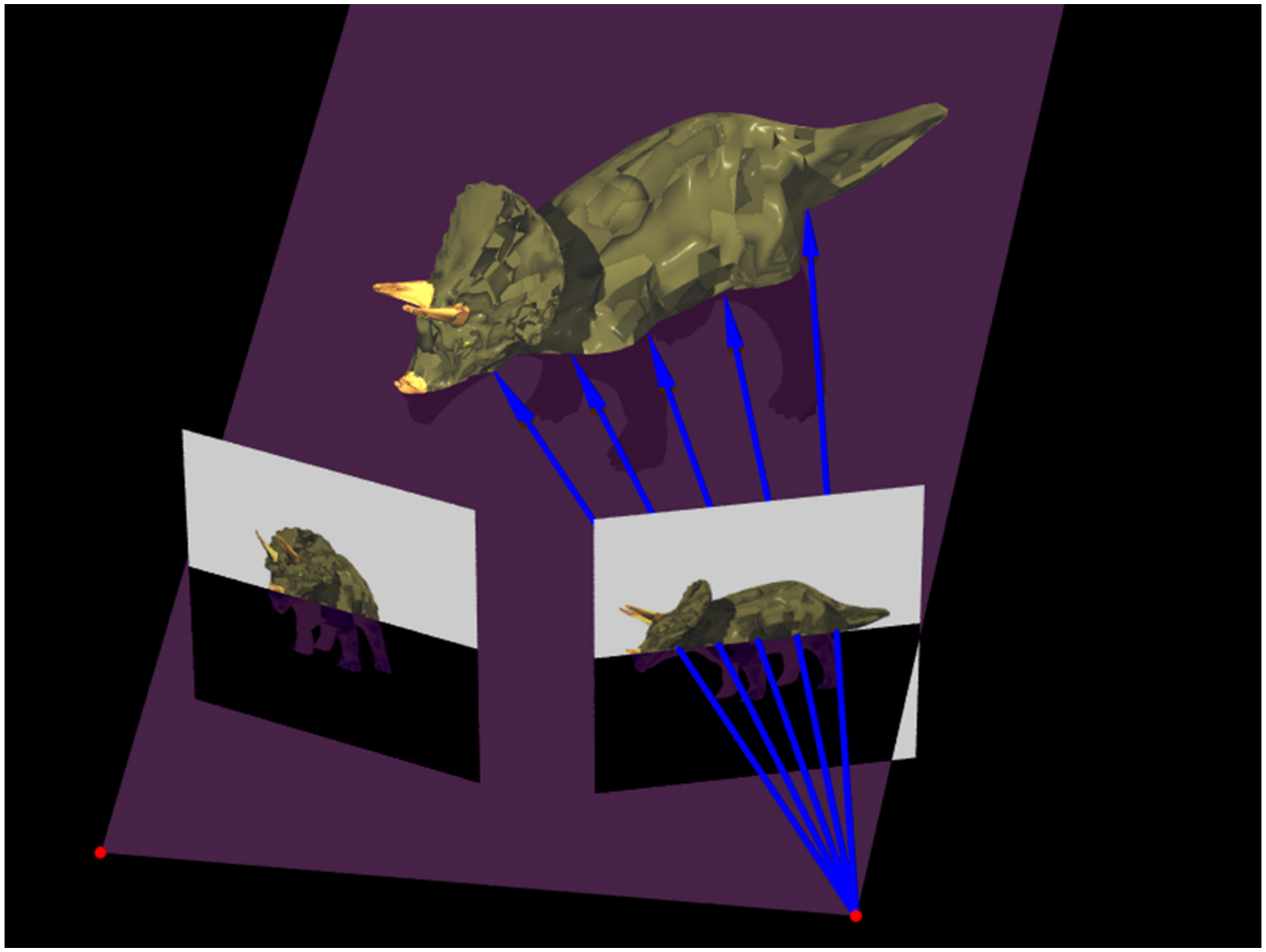
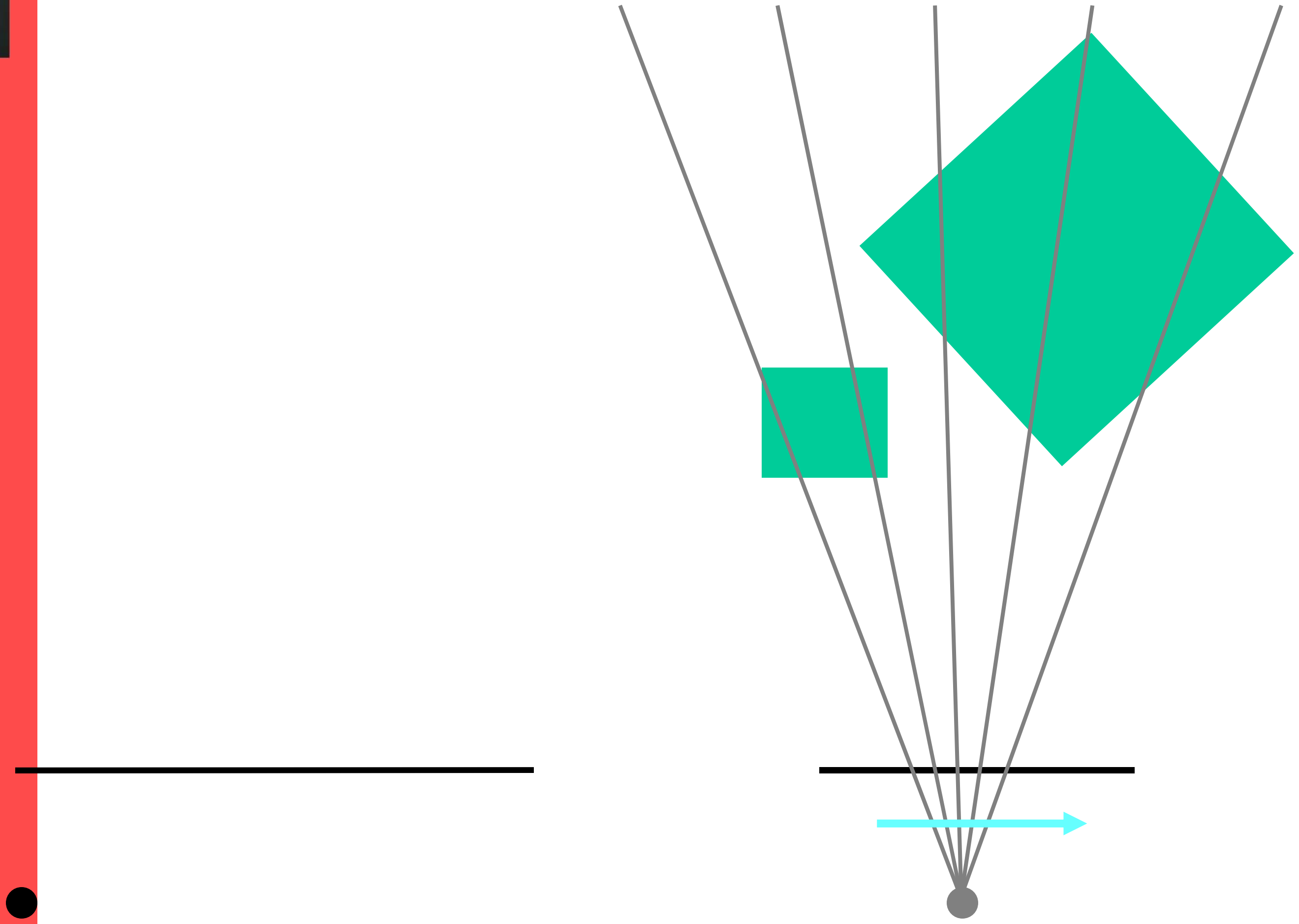


Figure 4 – In order to compute the visibility of an IBVH sample with respect to a given reference image, a series of IBVH intervals are projected back onto the reference image in an occlusion-compatible order. The front-most point of the interval is visible if it lies outside of the unions of all preceding intervals.

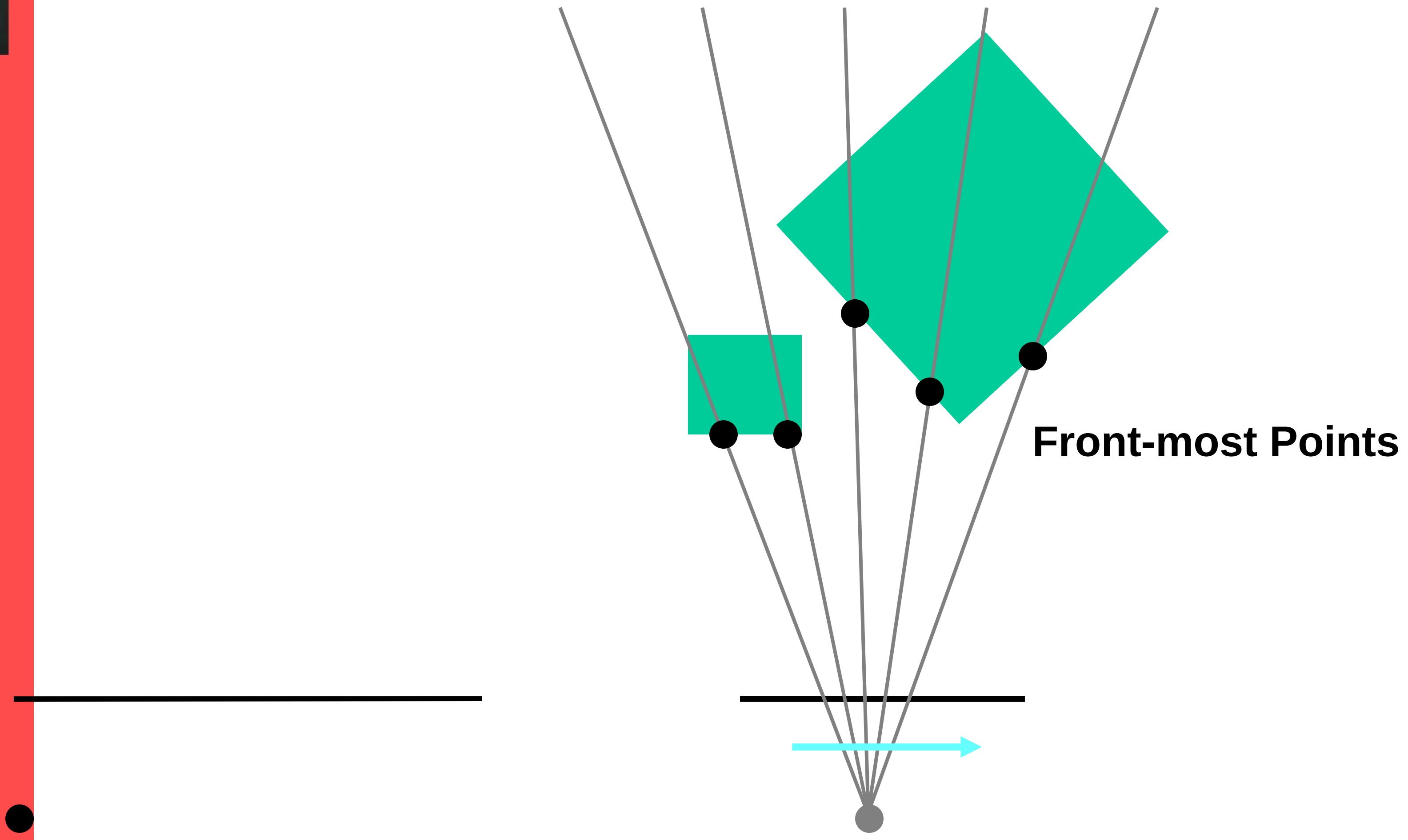
# Visibility Algorithm



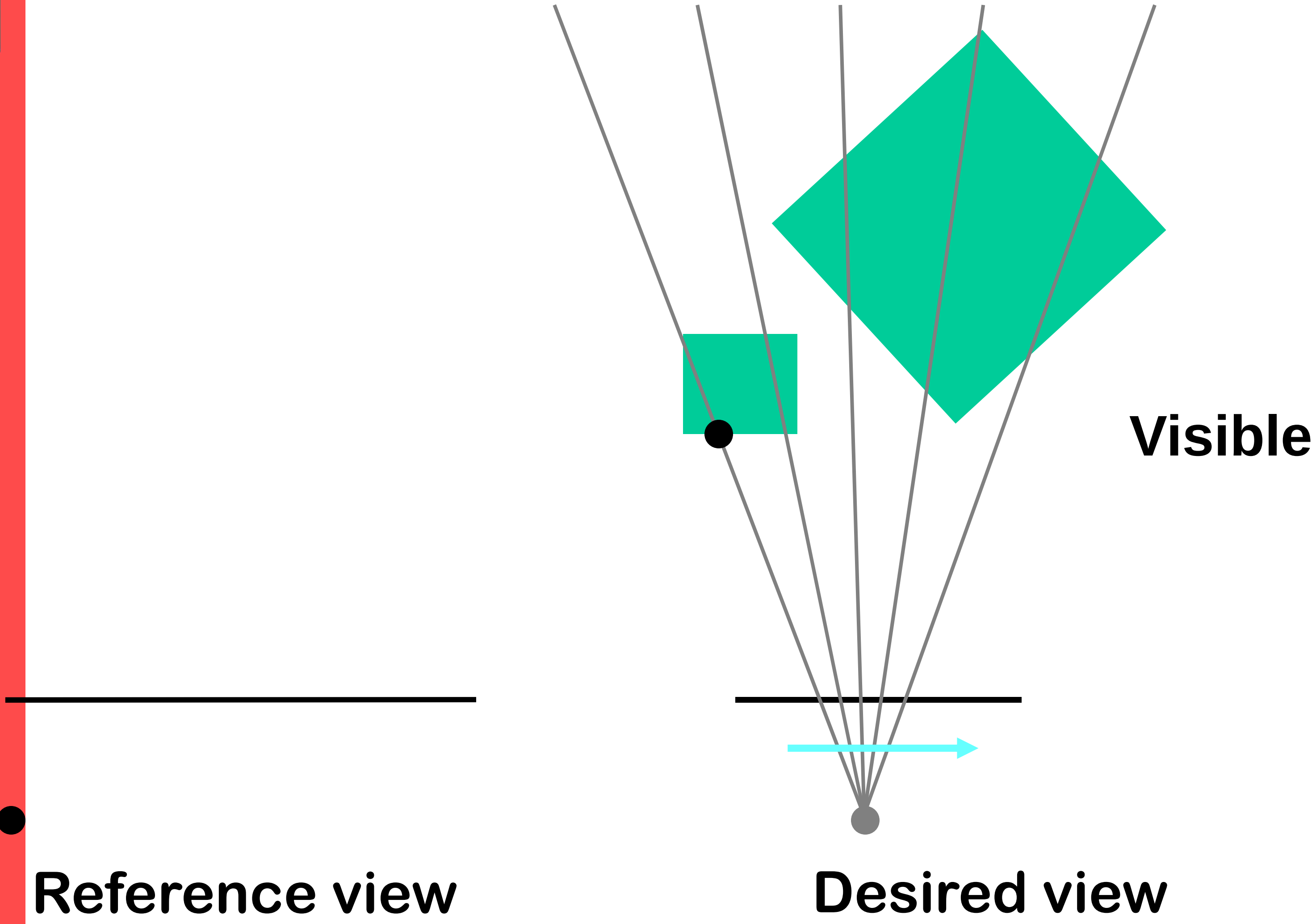
# Visibility in 2D



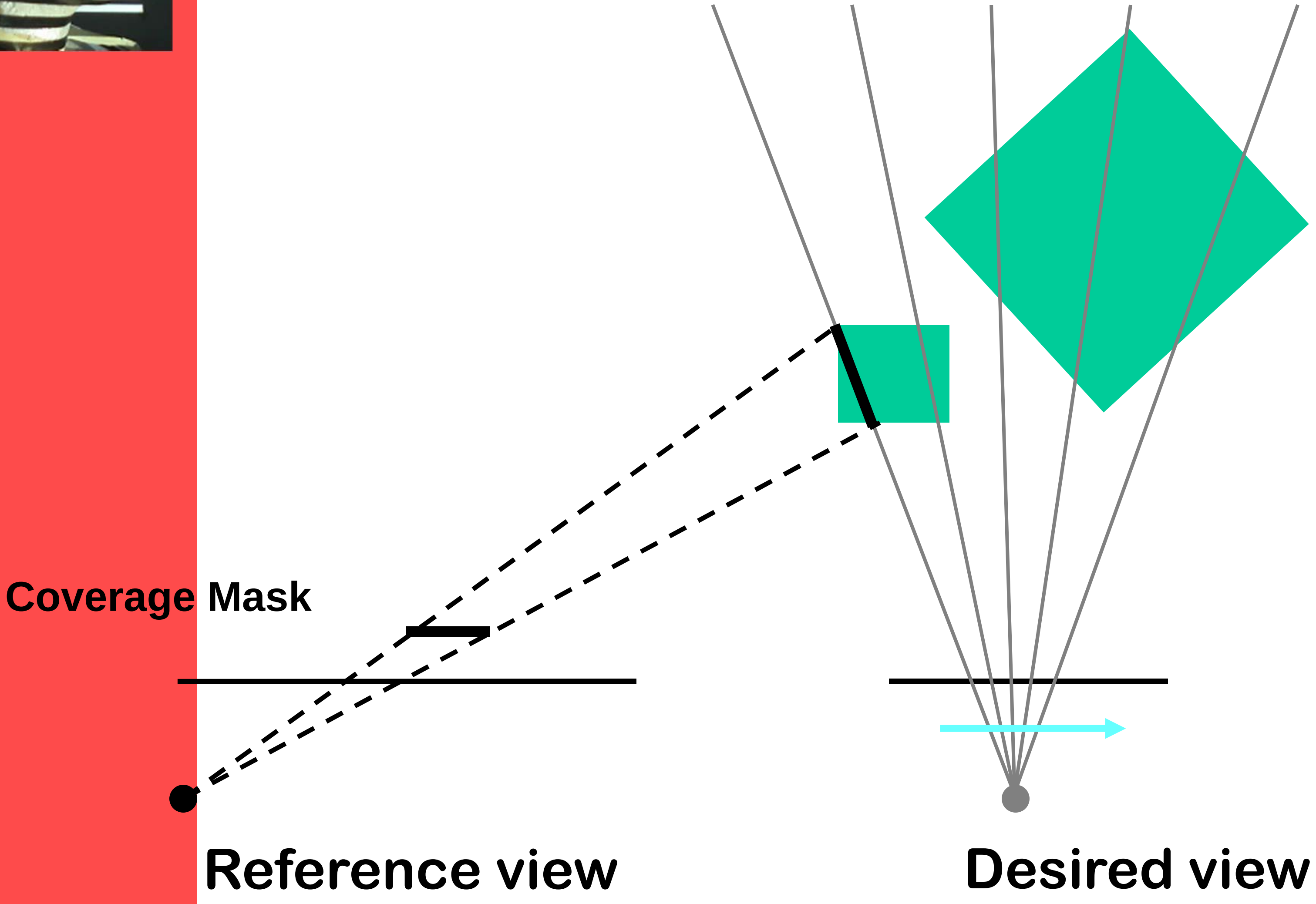
## Visibility in 2D



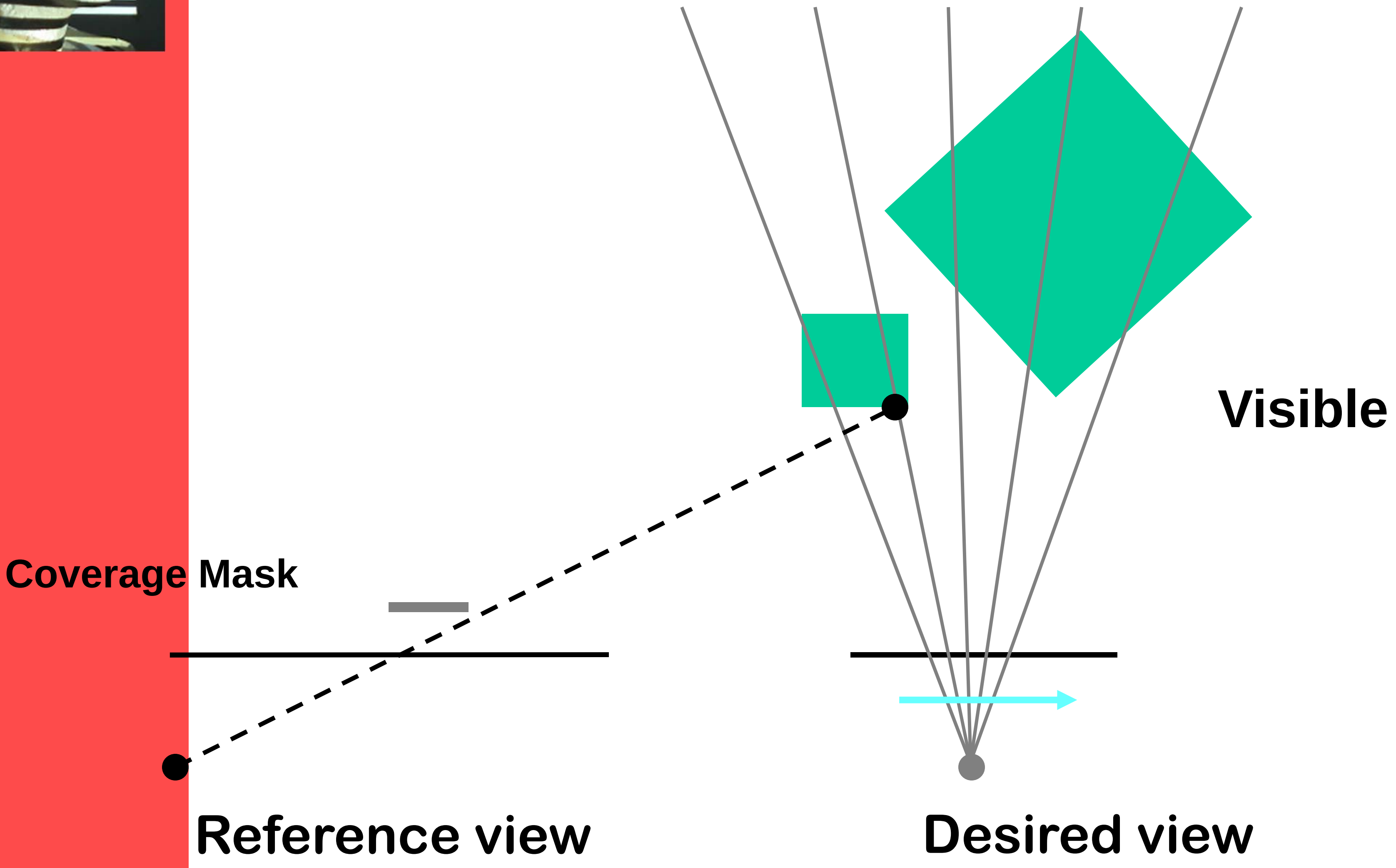
## Visibility in 2D



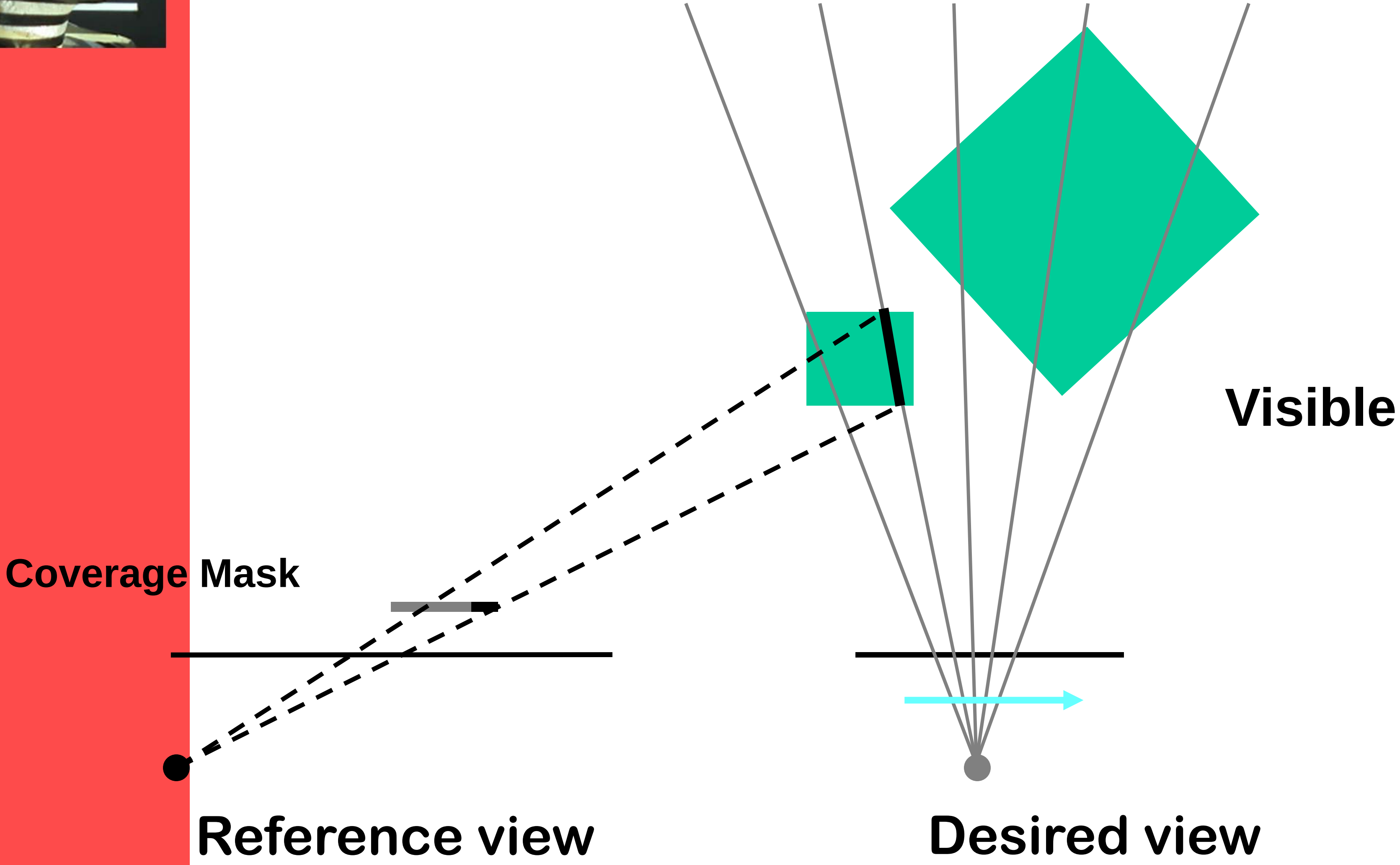
# Visibility in 2D



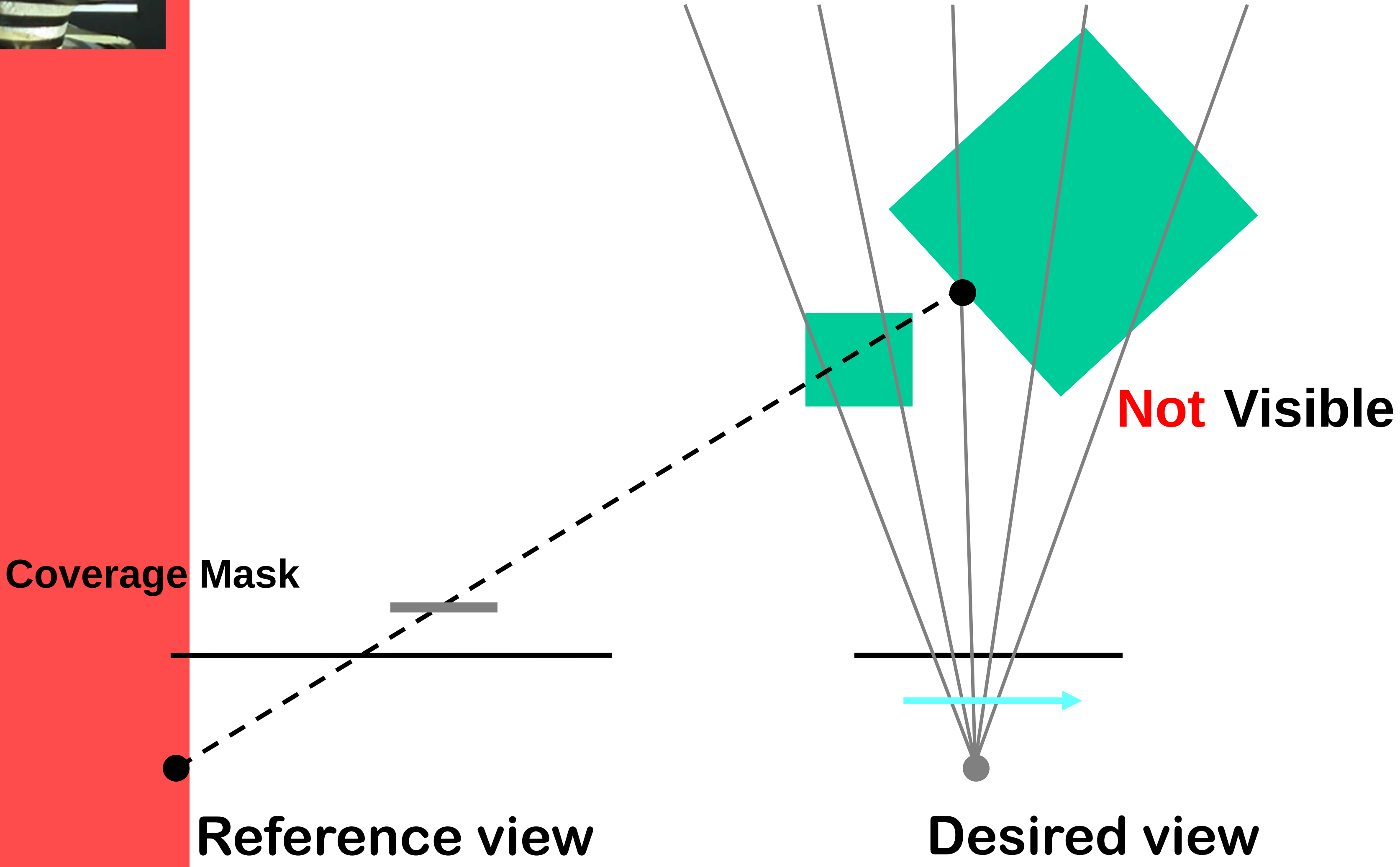
# Visibility in 2D



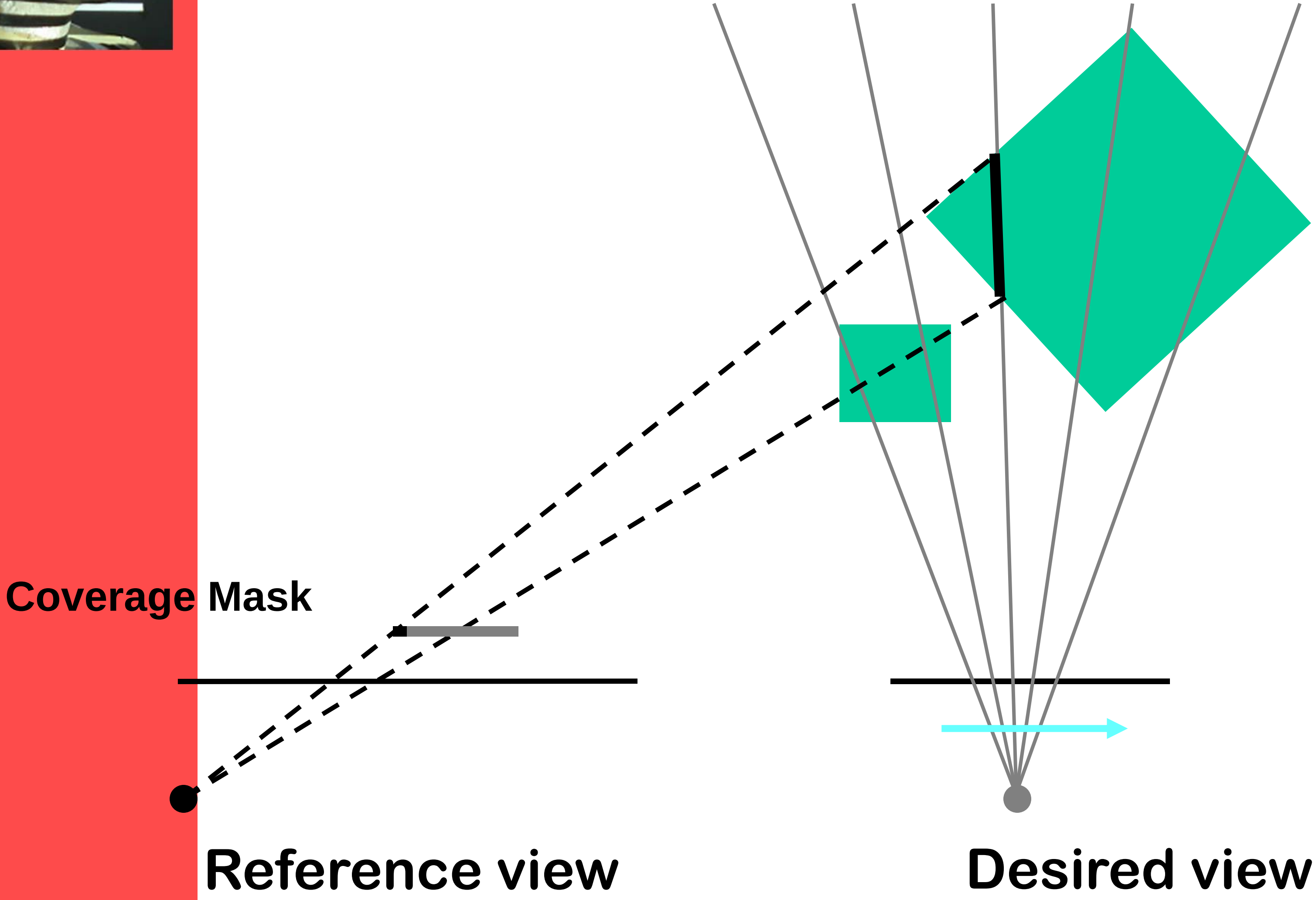
## Visibility in 2D



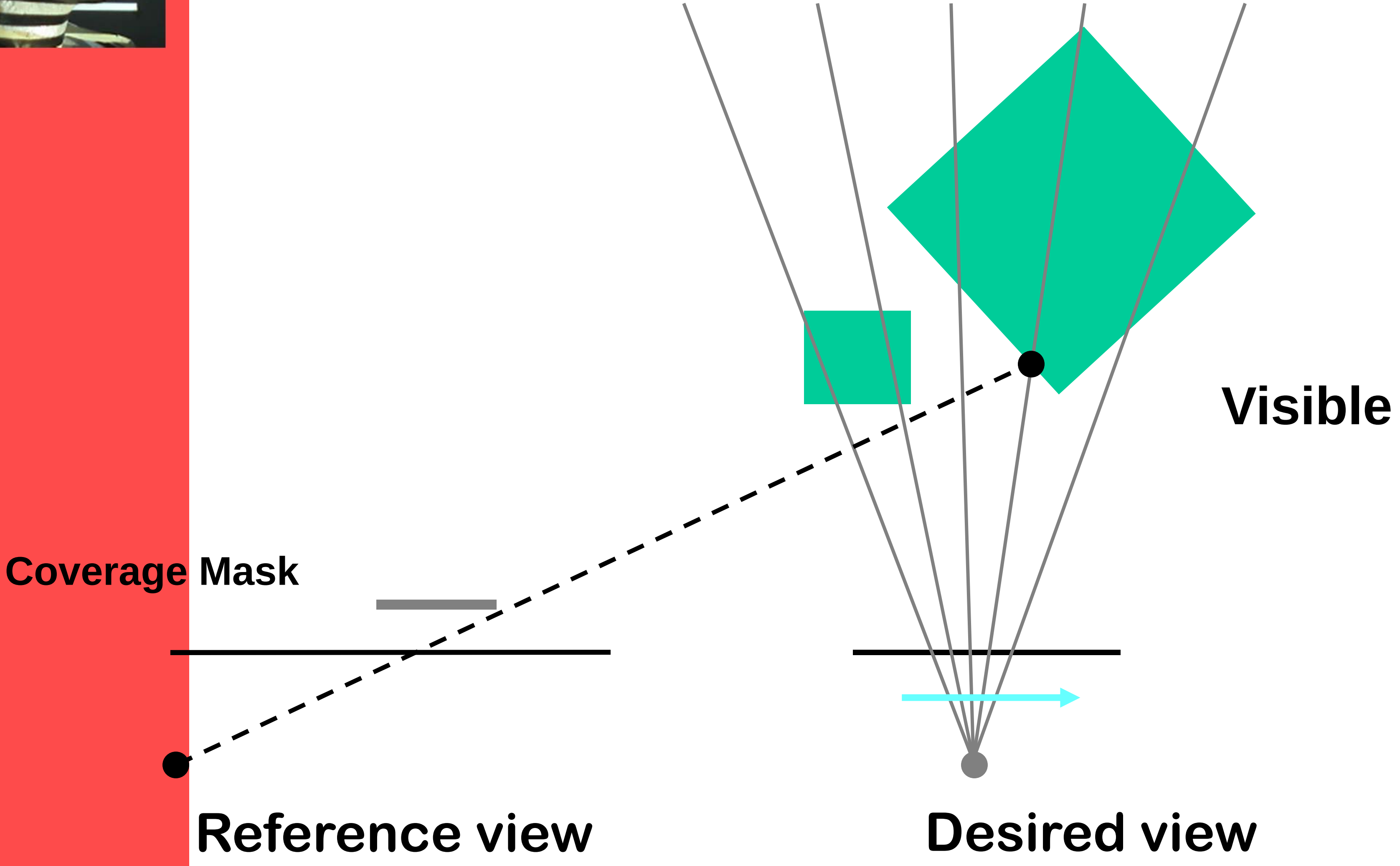
## Visibility in 2D



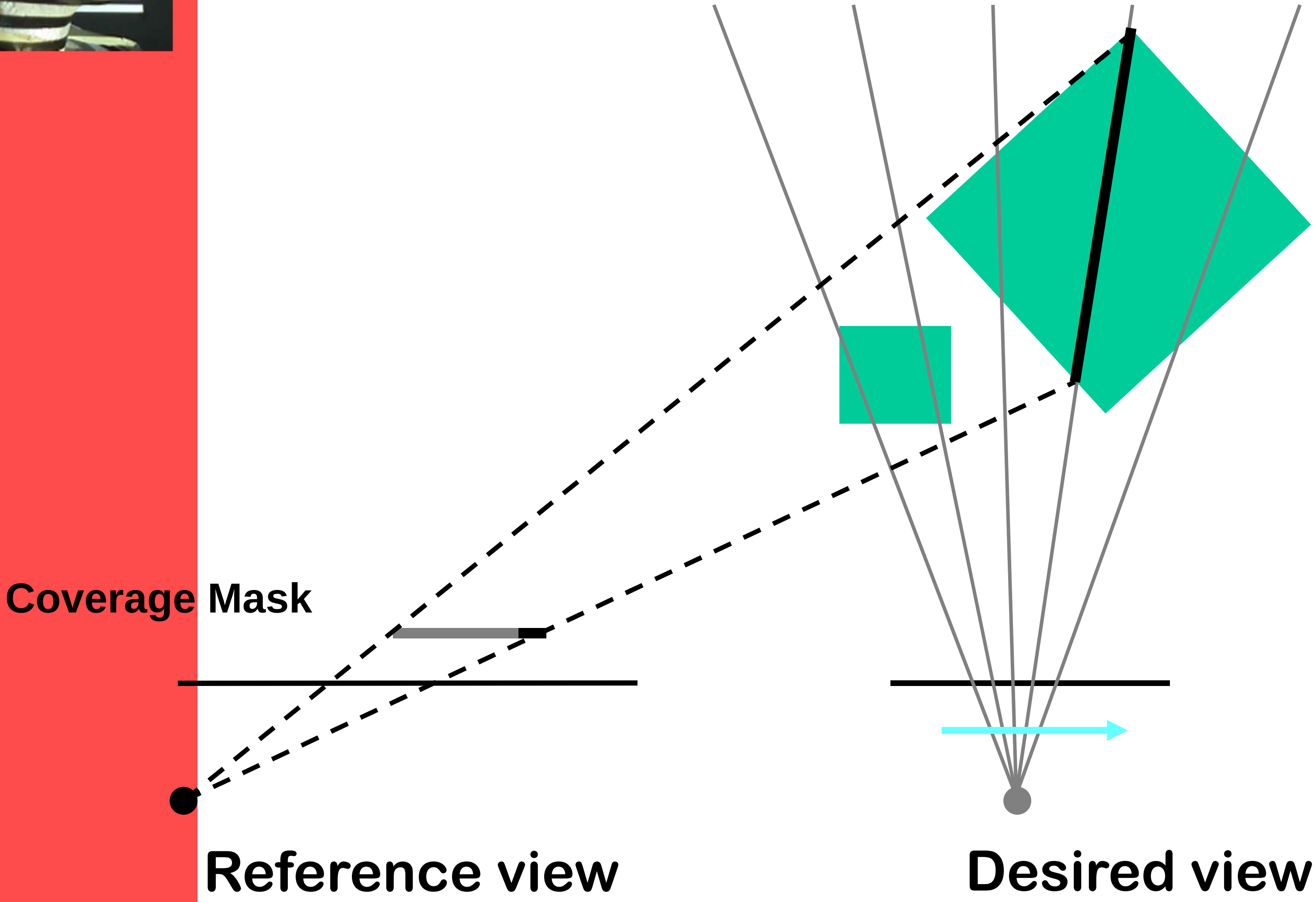
## Visibility in 2D



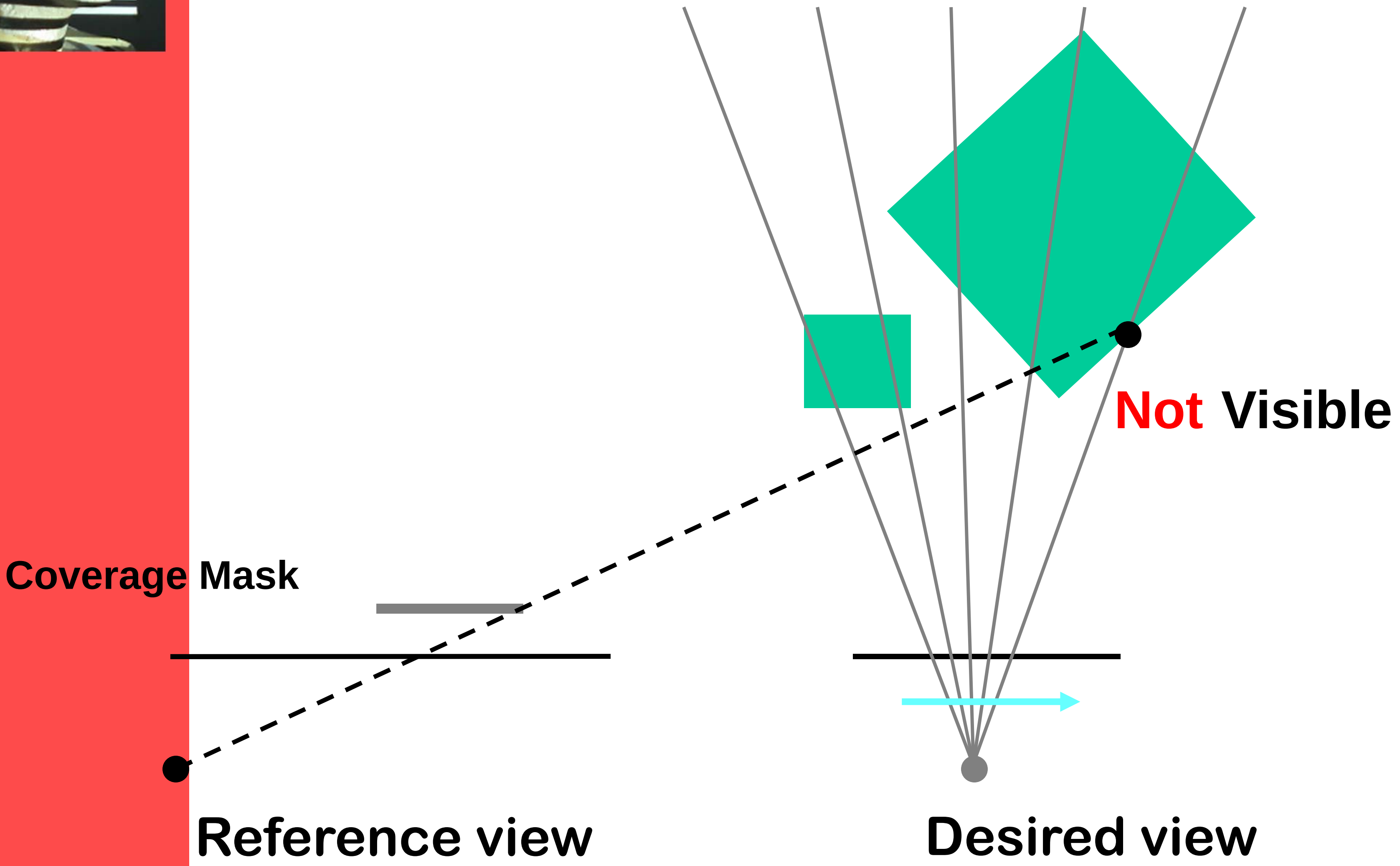
## Visibility in 2D



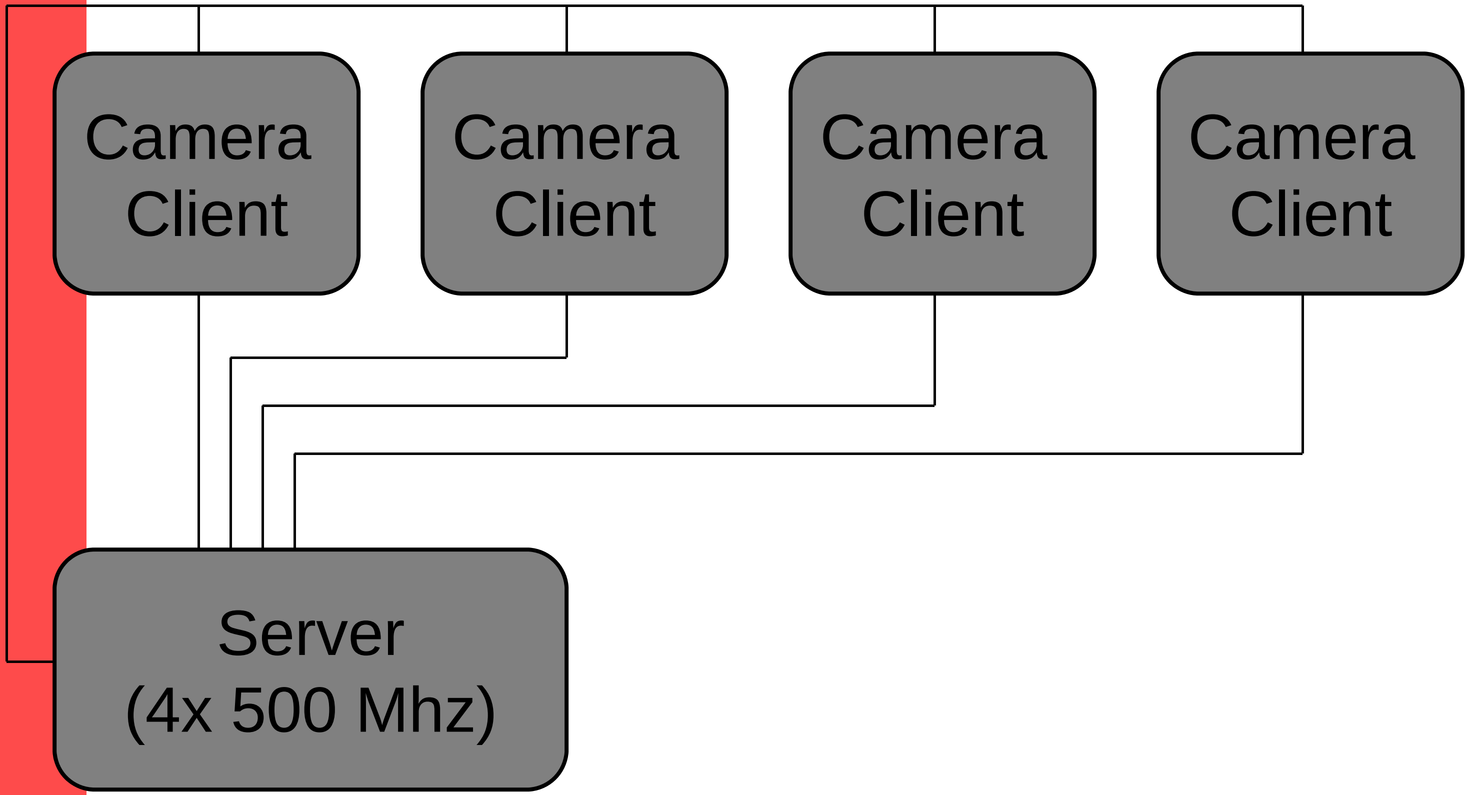
## Visibility in 2D



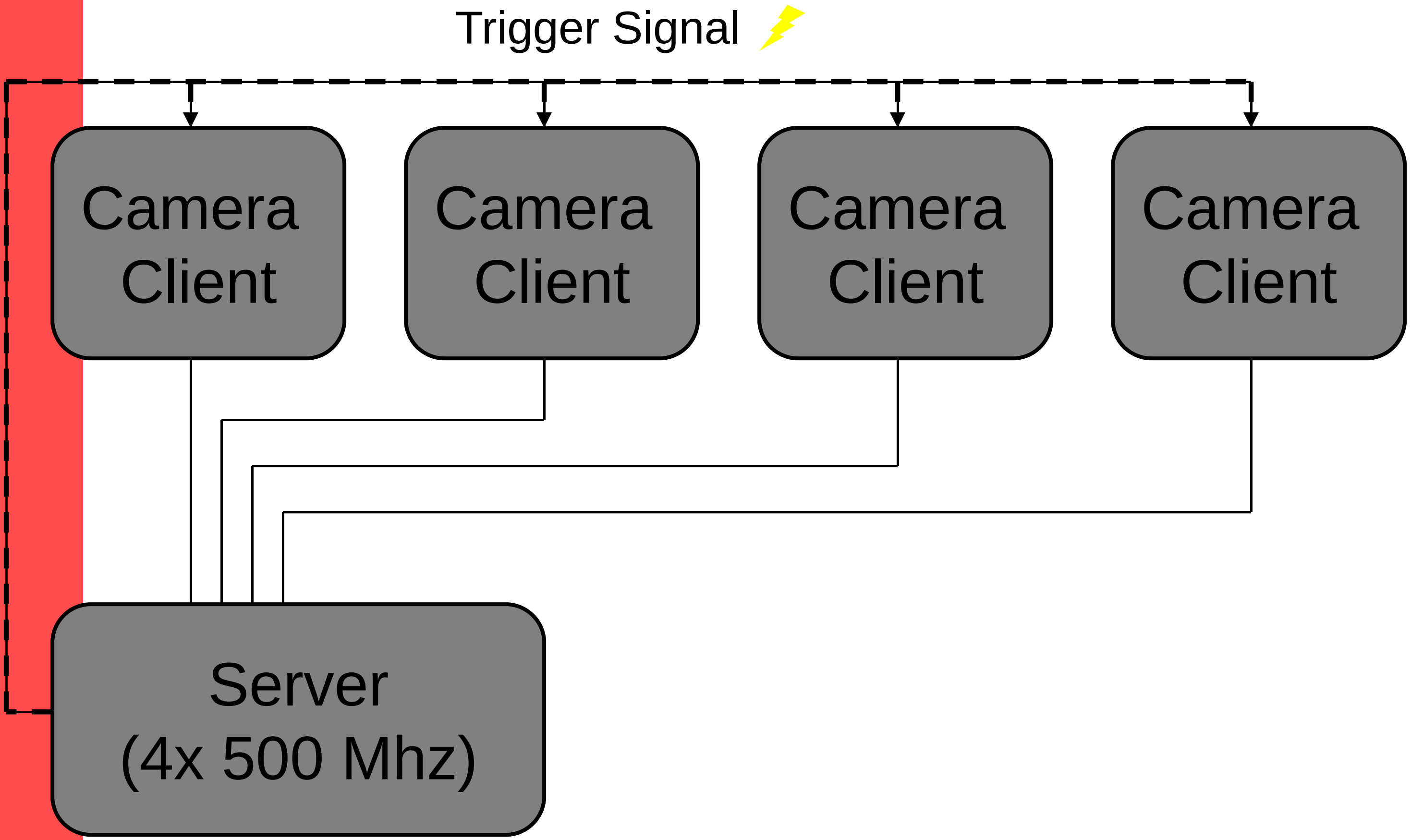
# Visibility in 2D



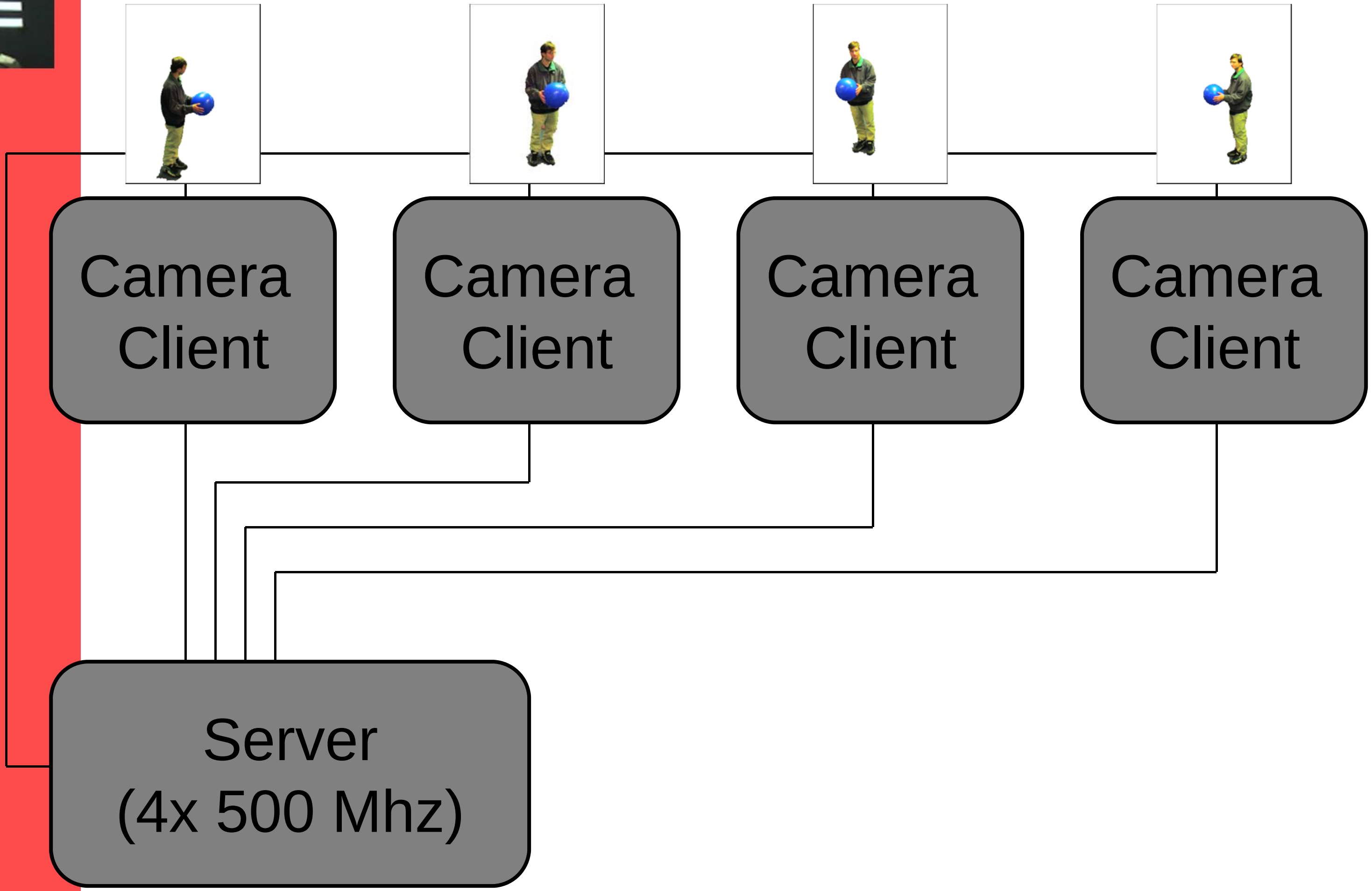
## System


Camera Client
Camera Client
Camera Client
Camera Client
Server (4x 500 Mhz)

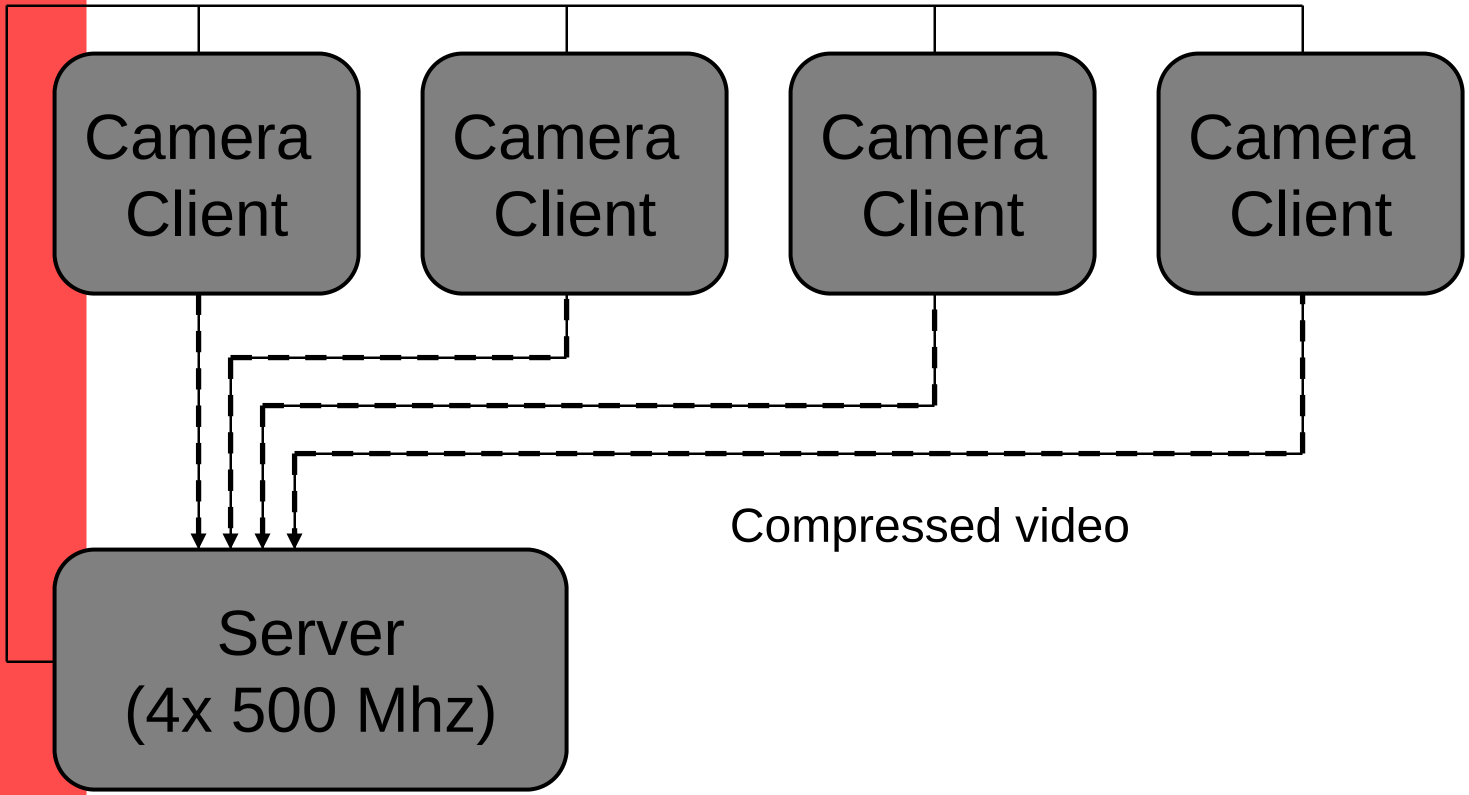
## System


Trigger Signal
Camera Client
Camera Client
Camera Client
Camera Client
Server (4x 500 Mhz)

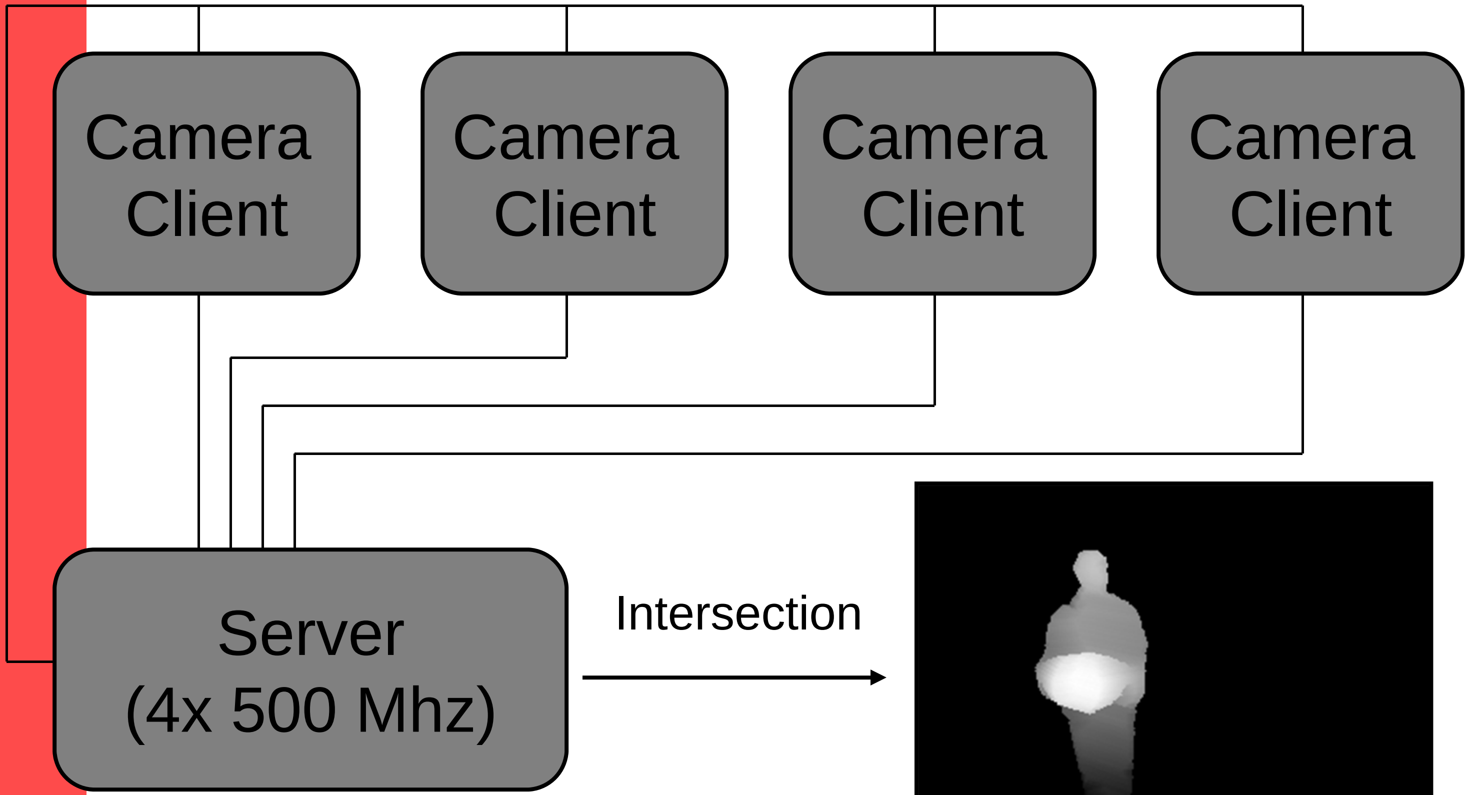
## System


Camera Client
Camera Client
Camera Client
Camera Client
Server (4x 500 Mhz)

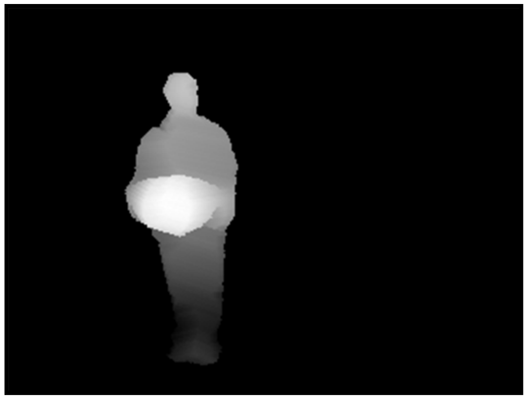
## System


Camera Client
Camera Client
Camera Client
Camera Client
Compressed video
Server
(4x 500 Mhz)

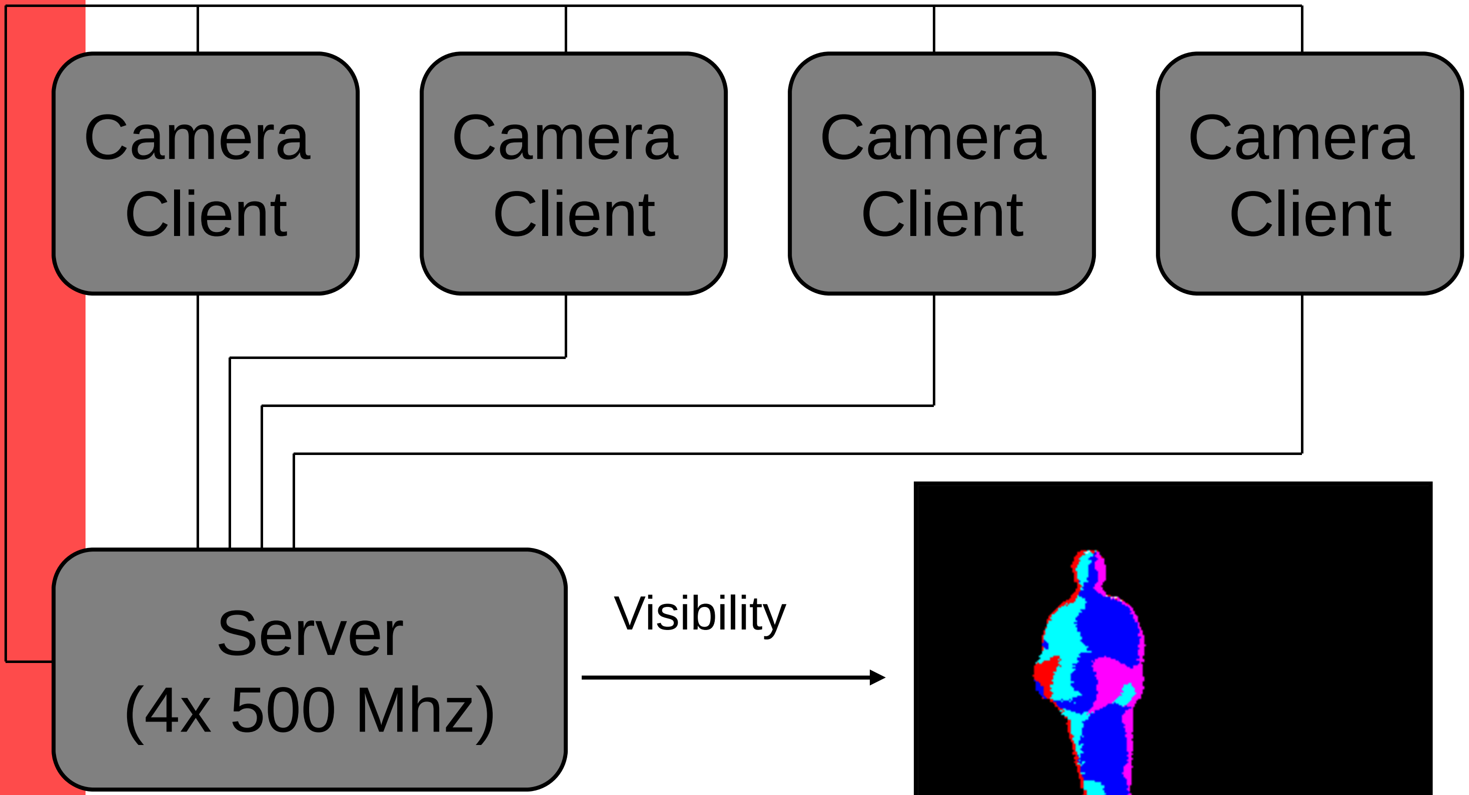
## System

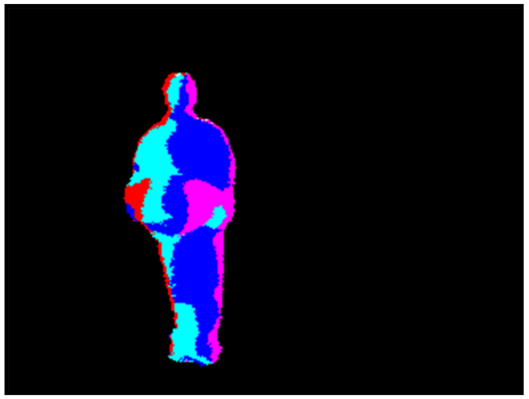


Camera Client
Camera Client
Camera Client
Camera Client
Server (4x 500 Mhz)
Intersection

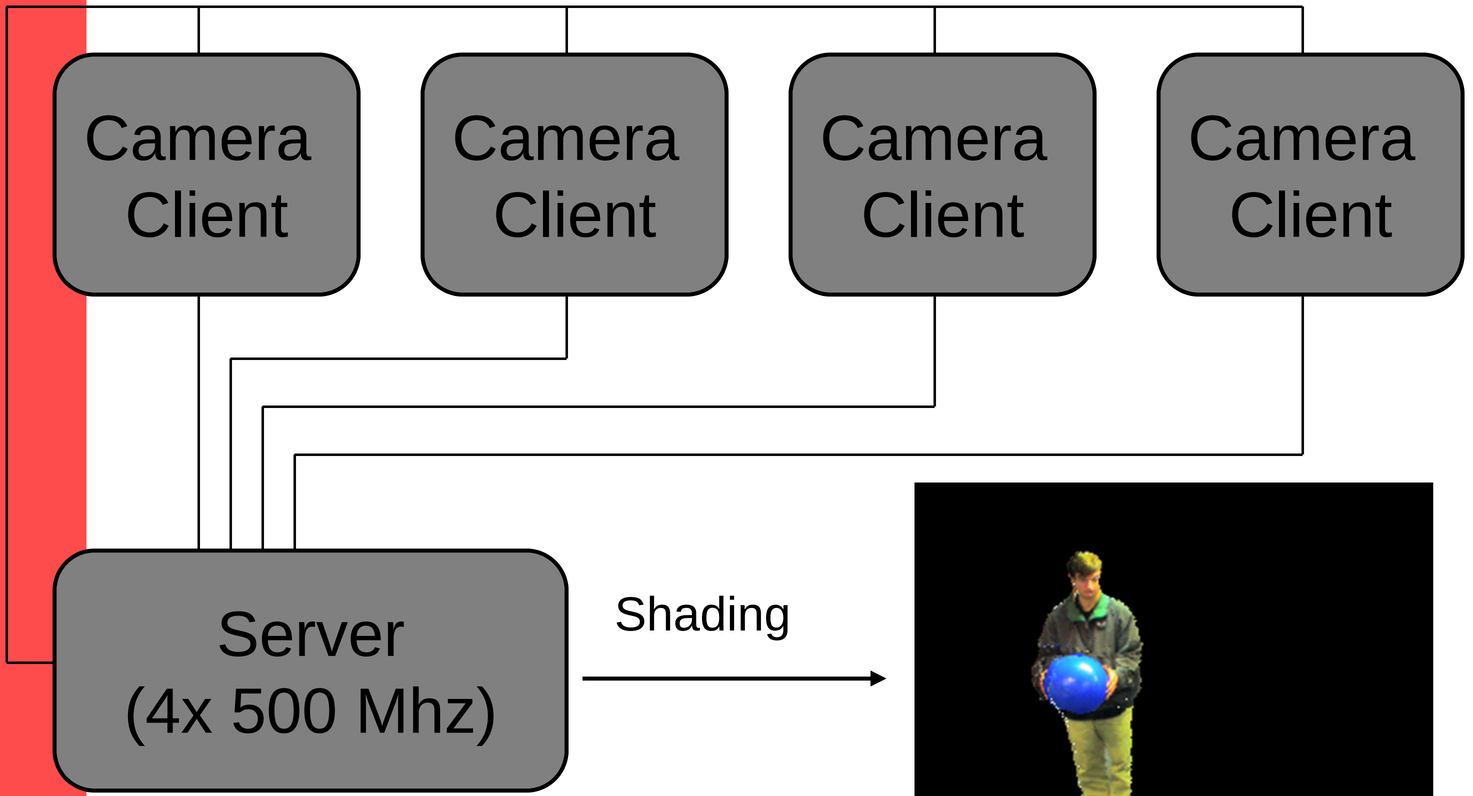


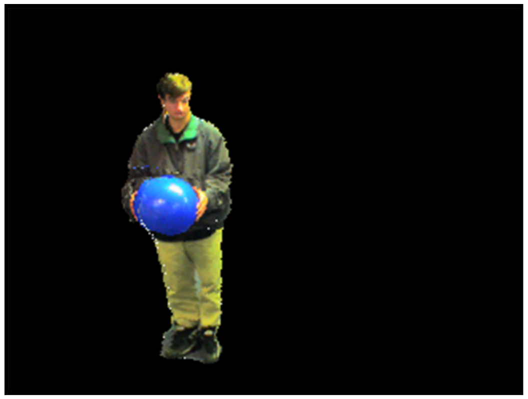
## System


Camera Client
Camera Client
Camera Client
Camera Client
Server (4x 500 Mhz)
Visibility



# 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

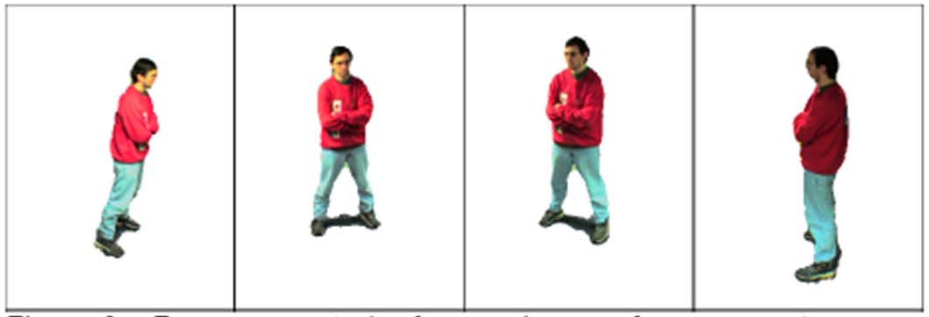


Figure 6 – Four segmented reference images from our system.

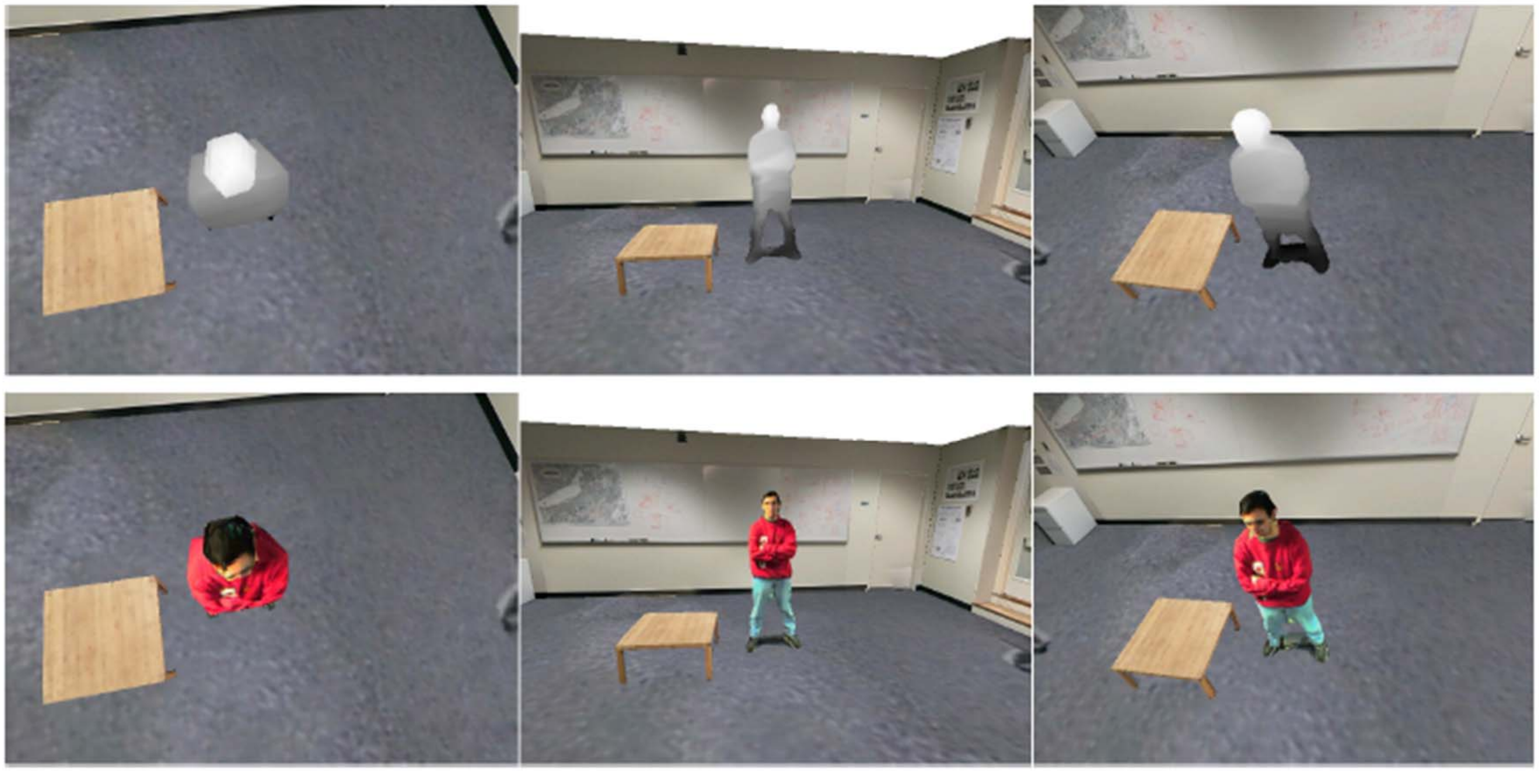
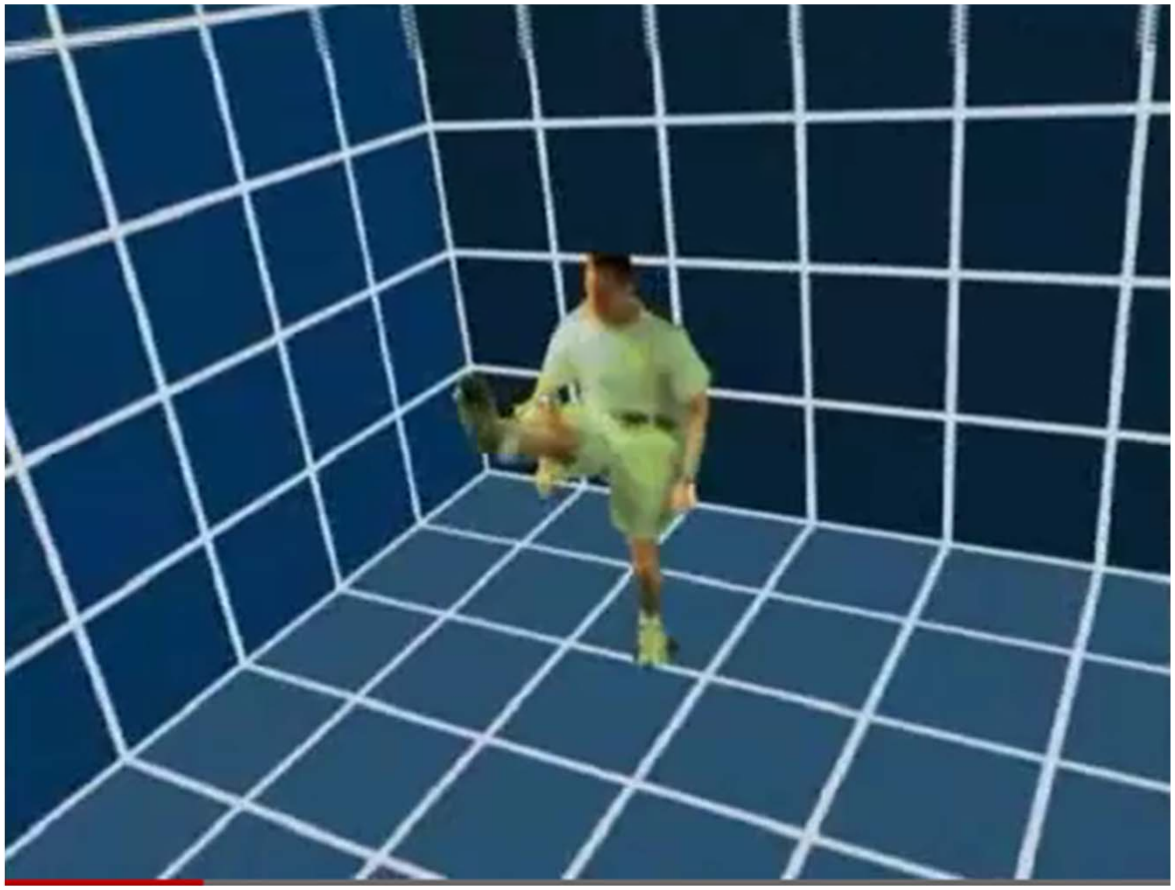


Figure 8 - Example IBVH images. The upper images show depth maps of the computed visual hulls. The lower images show shaded renderings from the same viewpoint. The hull segment connecting the two legs results from a segmentation error caused by a shadow.

# 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