The Multi-Pole Occlusion Camera

Mihai Mudure

April 18, 2007

Reconstruction of point datasets

- Hard problem
- Reconstructing global models
  - Challenging offline process
- Reconstruction the point dataset for each desired view
  - Estimating the footprint of a point in the output image is difficult

Goal

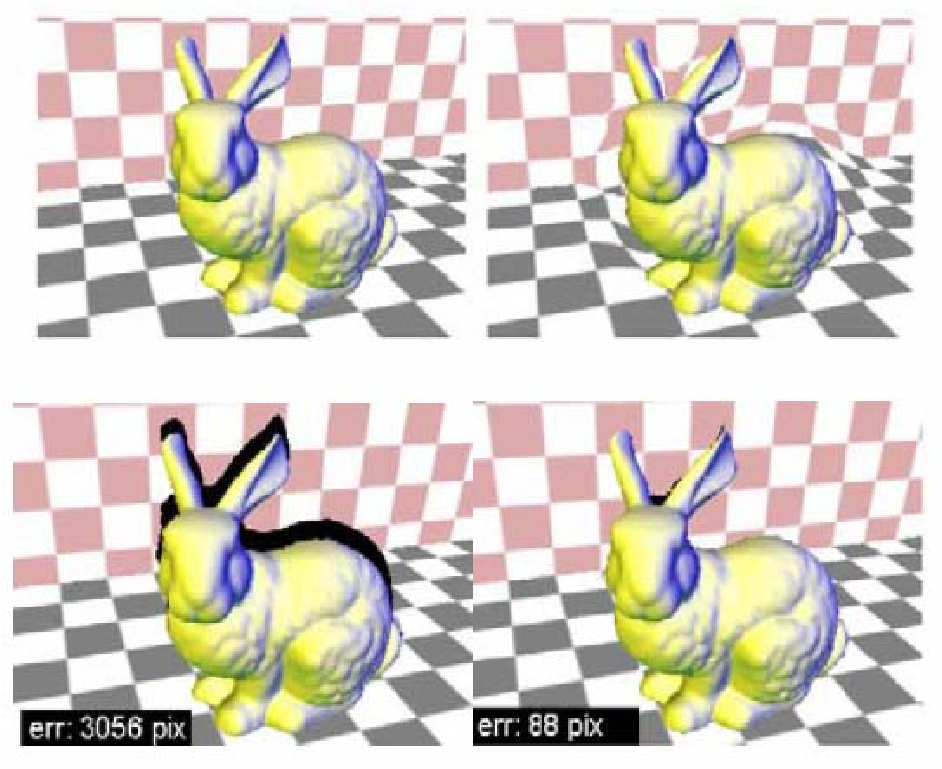
- Intermediate approach between computing a global model and single viewpoint reconstruction

- Reconstruction that can be used over a continuum of viewpoints around a reference view

Idea

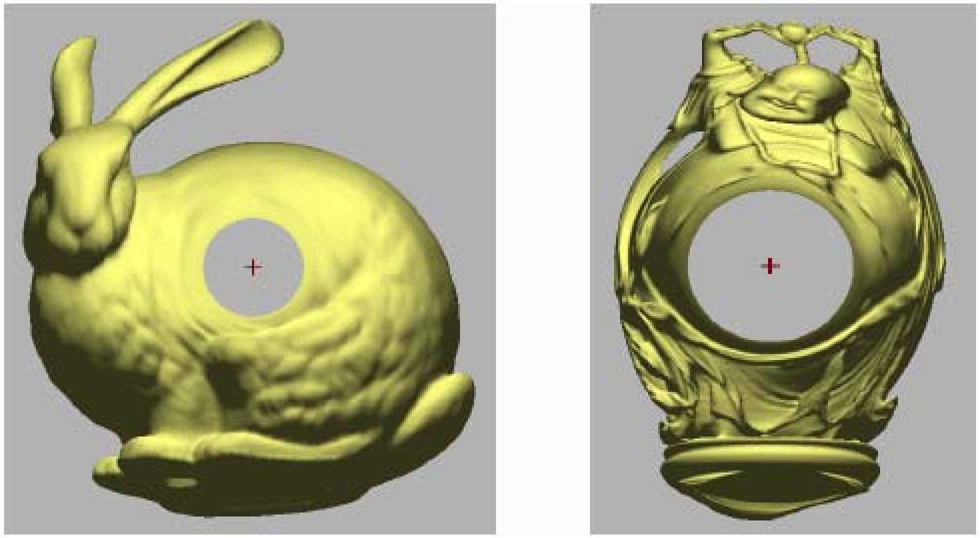
- Use occlusion cameras
- Non pinhole class of cameras
- Produce images that are robust to disocclusion errors

Occlusion cameras



V. Popescu, D. Aliaga, "Depth Discontinuity Occlusion Camera," *ACM Symposium on Interactive 3D Graphics*, Mar., 2006

Occlusion cameras



C. Mei, V. Popescu, E. Sacks, "The Occlusion Camera,"
Computer Graphics Forum, Eurographics 2005, Sep., 2005

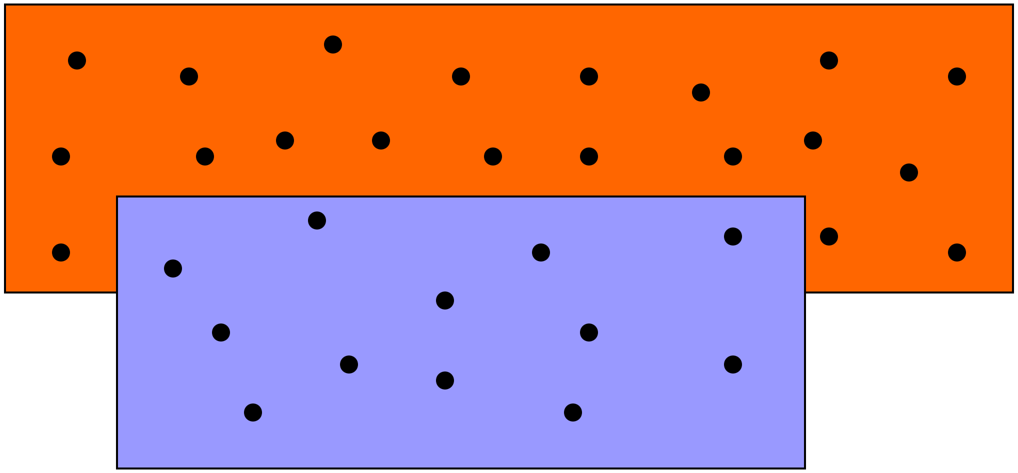
Multi-pole occlusion camera

- Camera model
- Construction
- Projection
- Mesh reconstruction
- Results
- Future work

Camera Model

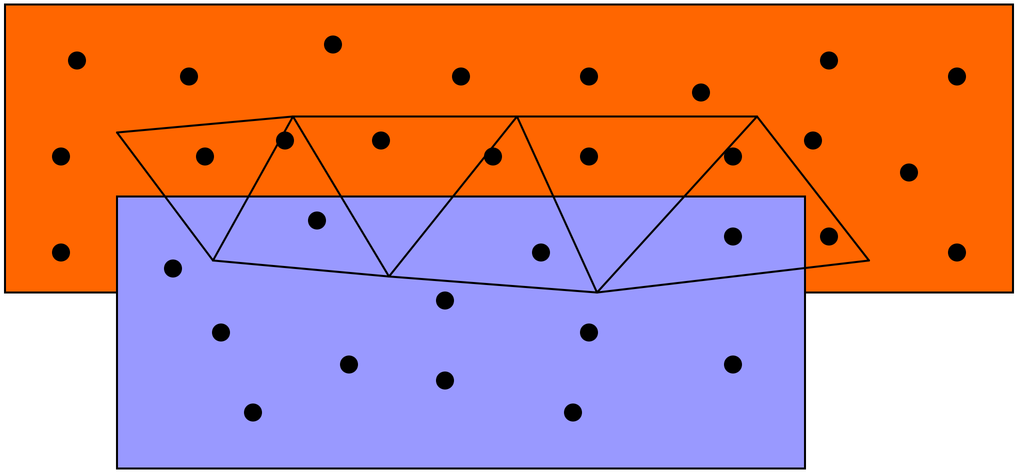
- Reference camera PHC0
- 2D triangulation in the image plane of PHC0 with vertices Pi (poles)
- Labeling of Pi as front/back

Camera Model

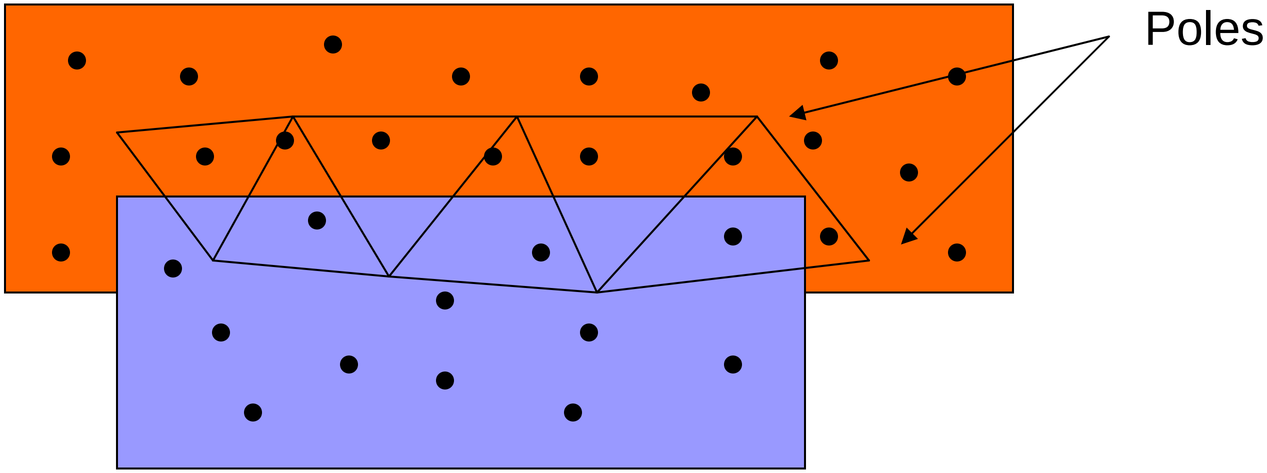


Sample scene

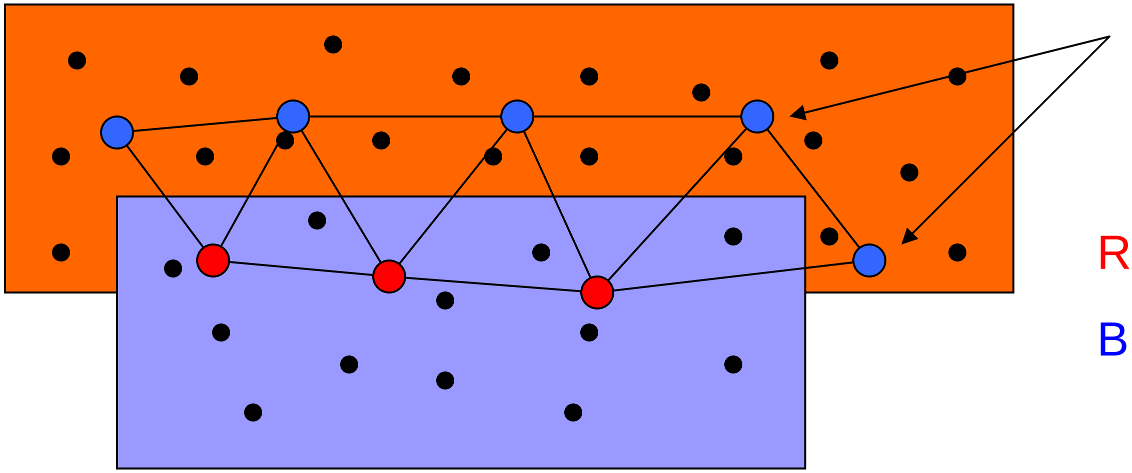
Camera Model



Camera Model



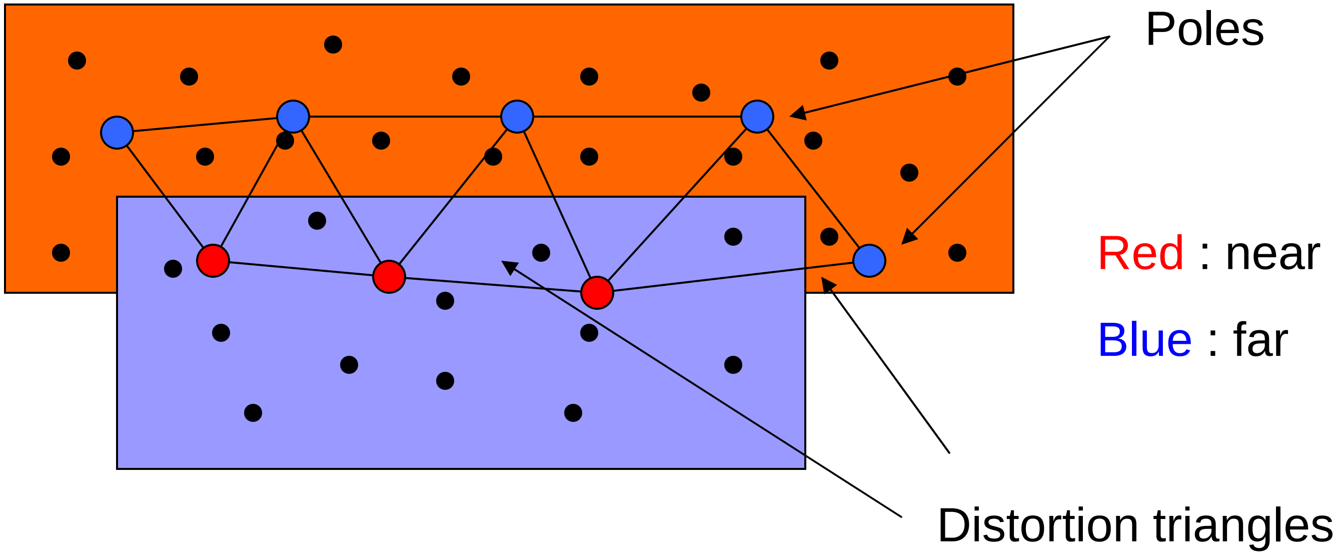
Camera Model



Red : front

Blue : back

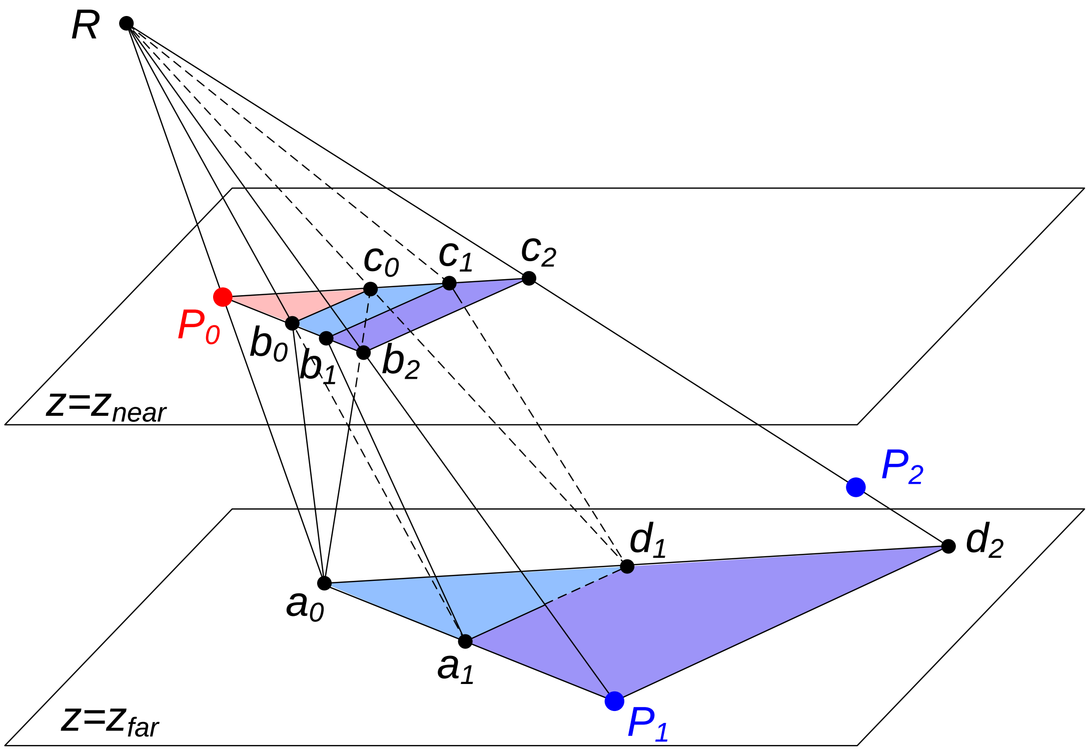
Camera Model



Distortion triangles

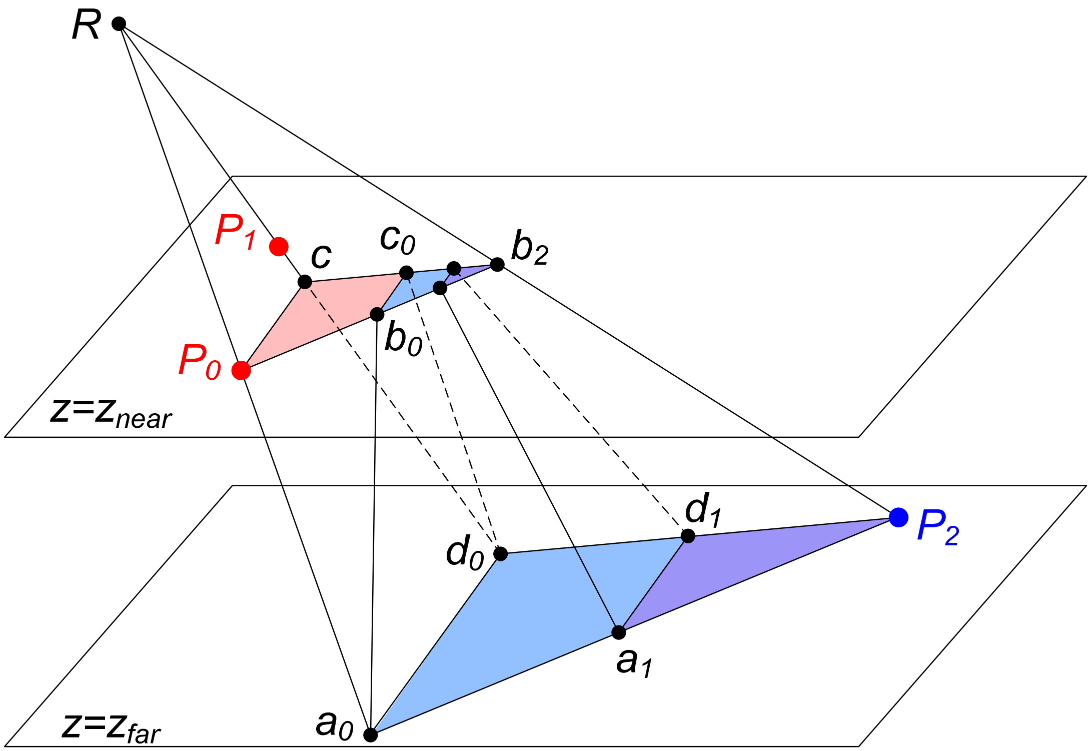
- Determine the rays of the distorted camera
- Triangles that have the same labeling for all vertices do not introduce any distortion
- Distortion depends on the depth of the distorted point
    - Z <= znear : no distortion
    - Z >= zfar : maximum distortion

Distortion triangles



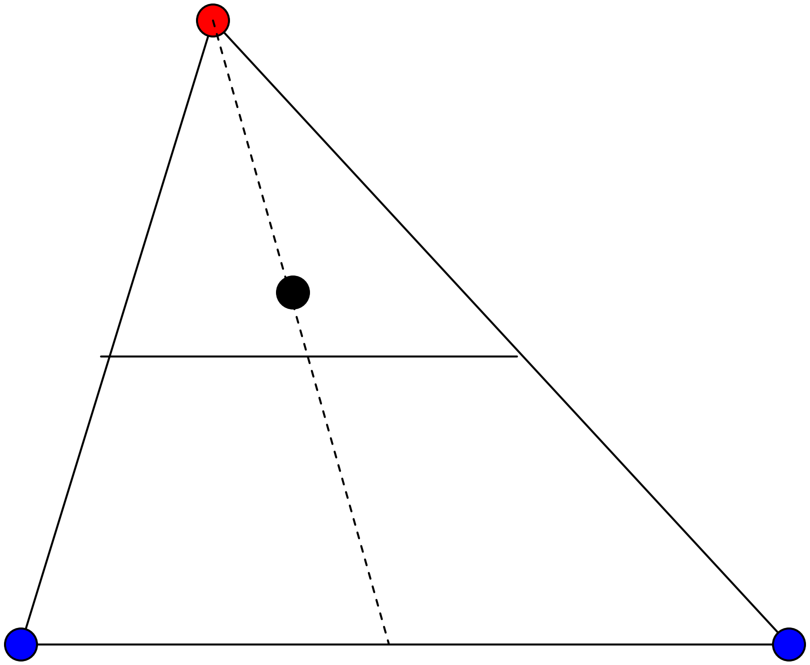
Distortion triangle with front, back, back labeling

Distortion triangles



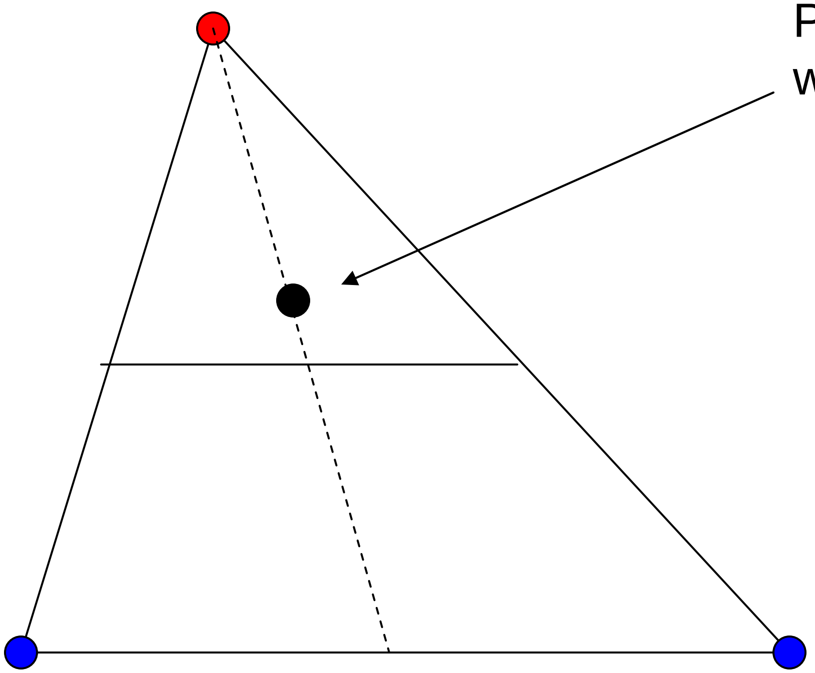
Distortion triangle with front, front, back labeling

Distortion triangles



Distortion illustration for a (front,back,back) triangle

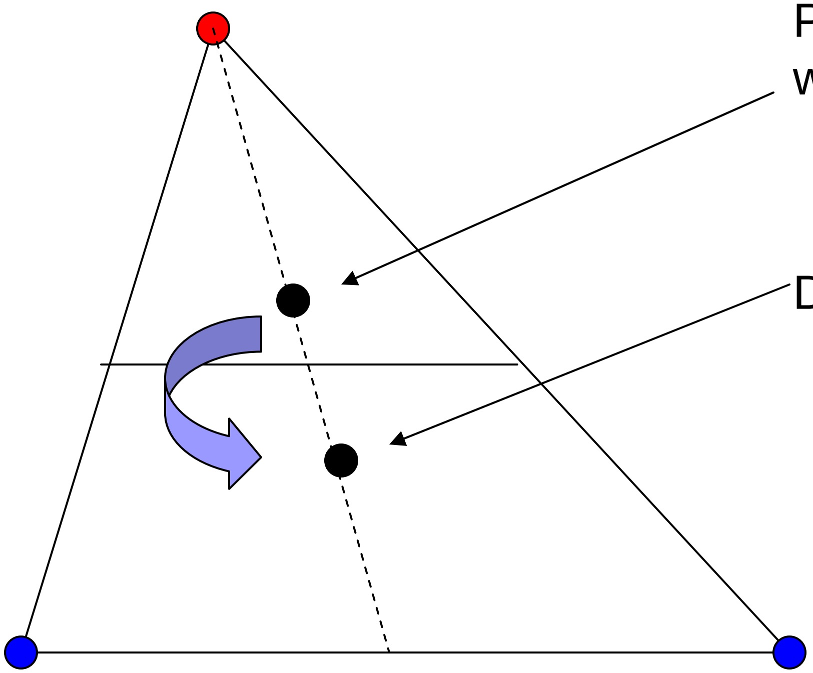
Distortion triangles



Point to be distorted with z=zfar

Distortion illustration for a (front,back,back) triangle

Distortion triangles

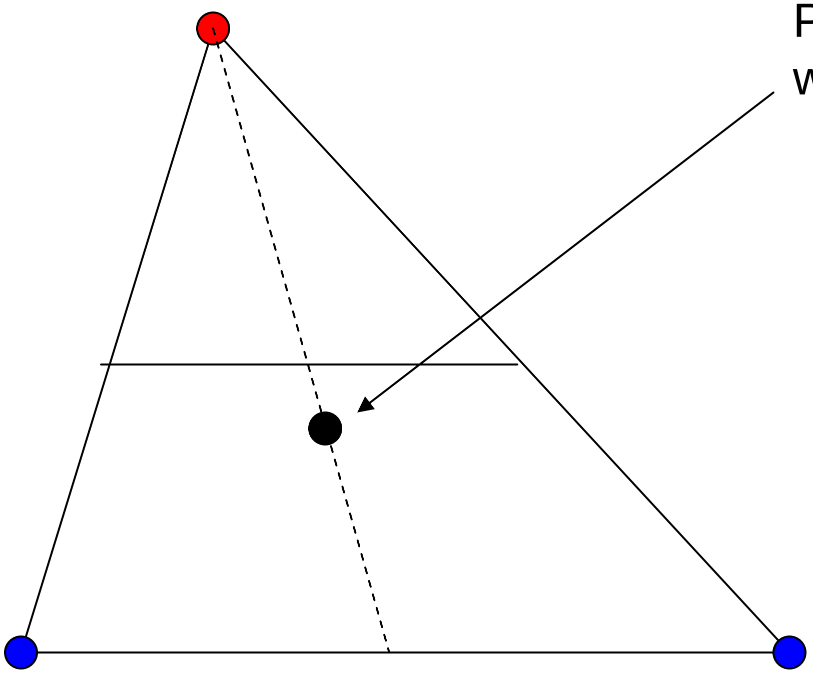


Point to be distorted with z=zfar

Distorted point

Distortion illustration for a (front,back,back) triangle

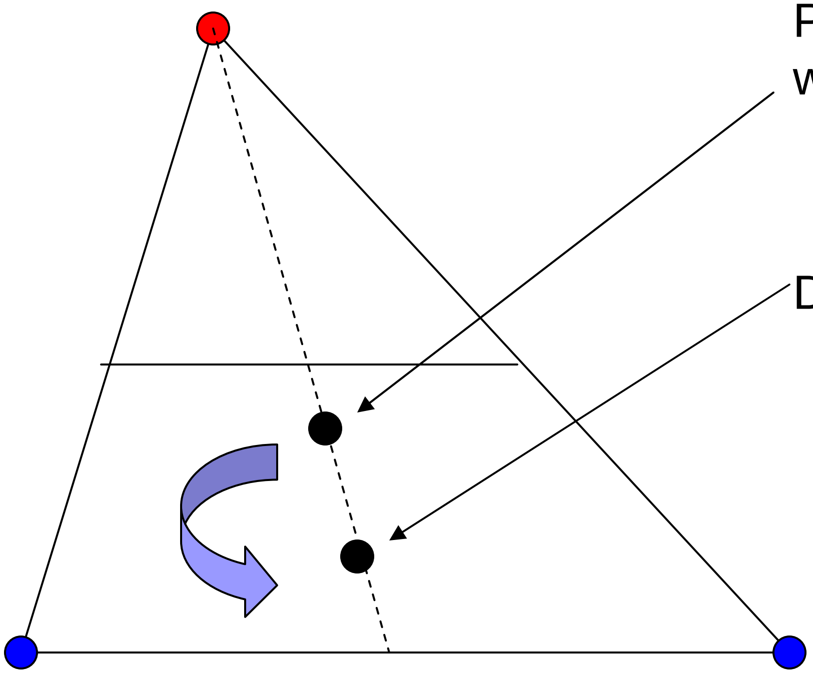
Distortion triangles



Point to be distorted with z=zfar

Distortion illustration for a (front,back,back) triangle

Distortion triangles

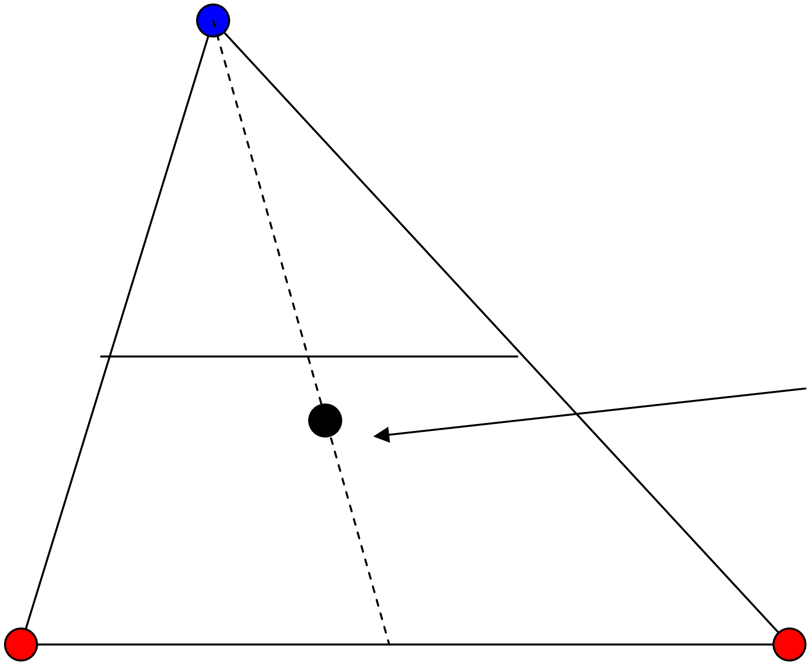


Point to be distorted with z=zfar

Distorted point

Distortion illustration for a (front,back,back) triangle

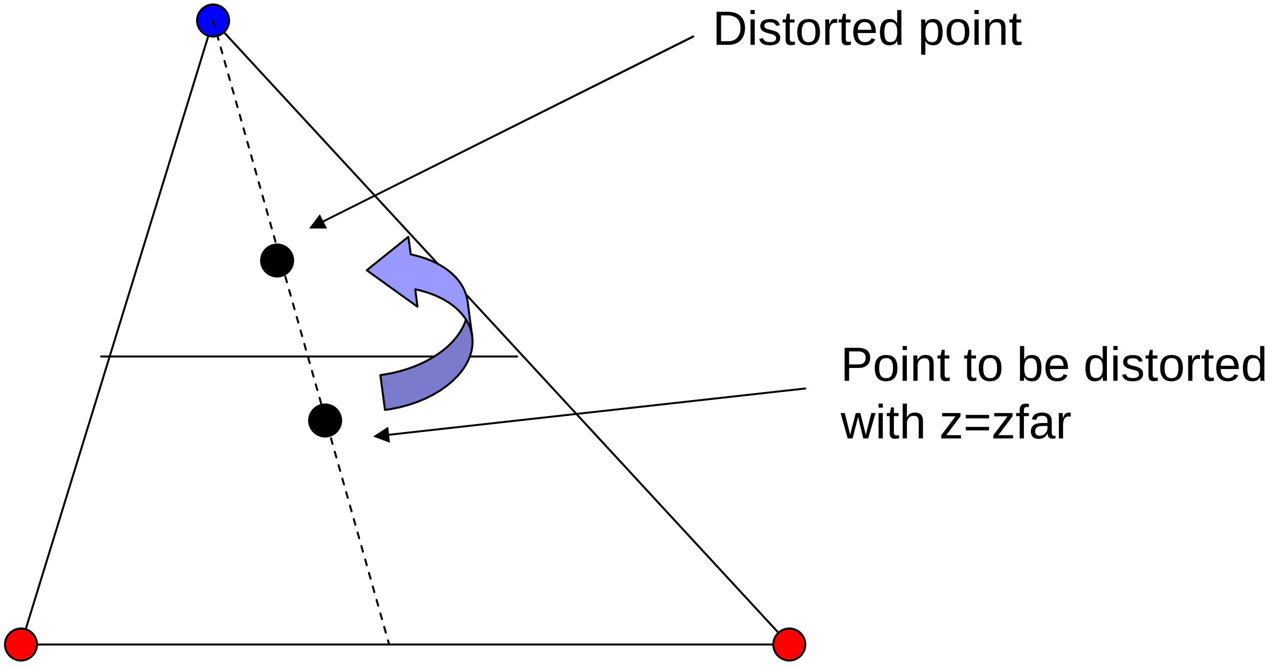
Distortion triangles



Point to be distorted with z=zfar

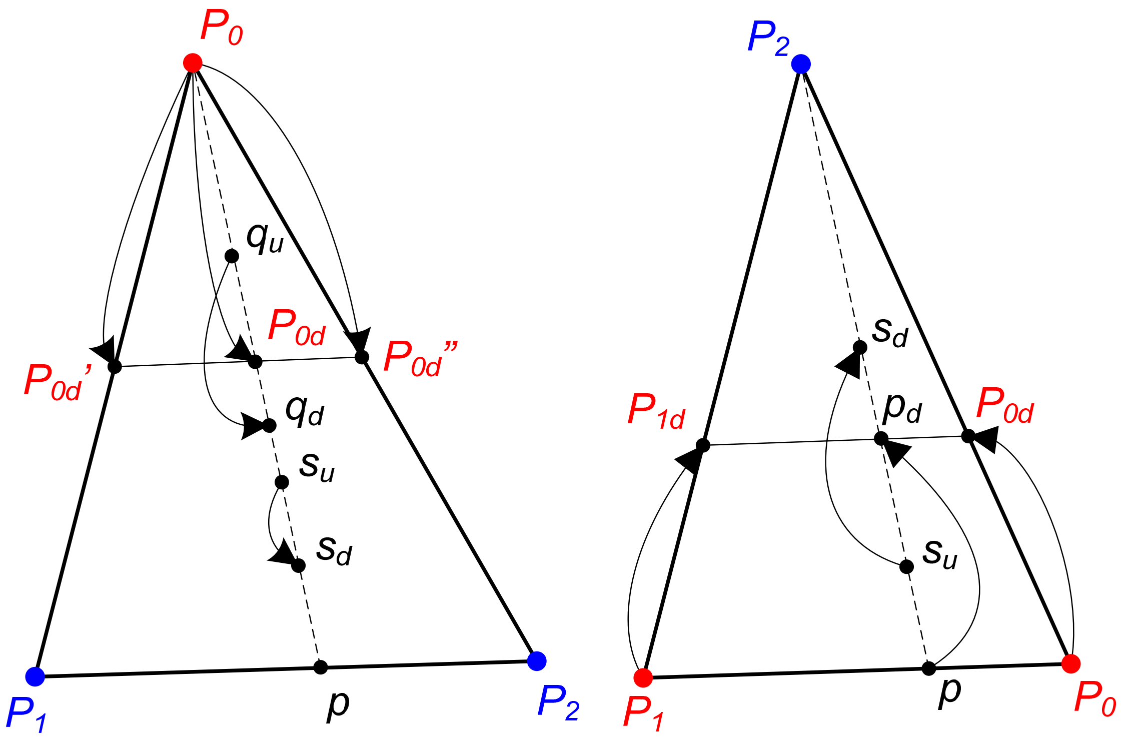
Distortion illustration for a (front,front,back) triangle

Distortion triangles

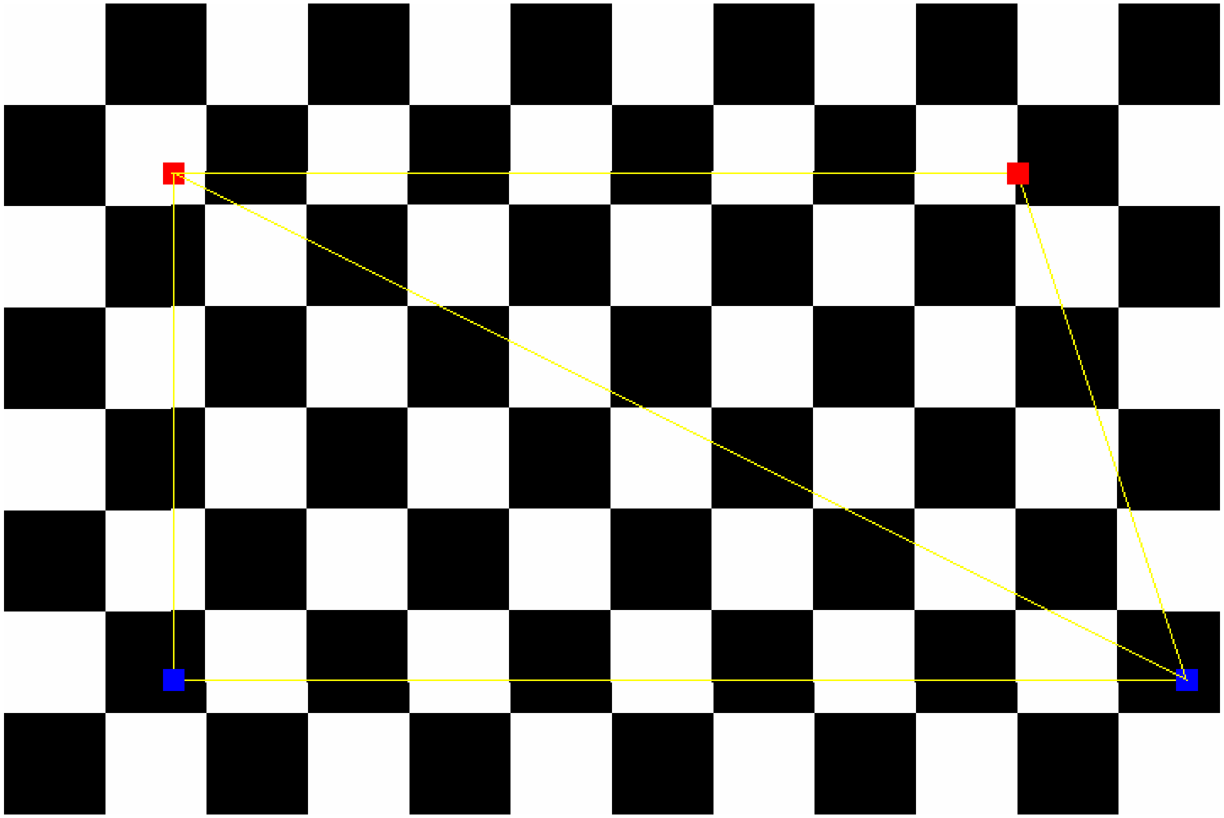


Distortion illustration for a (front,front,back) triangle

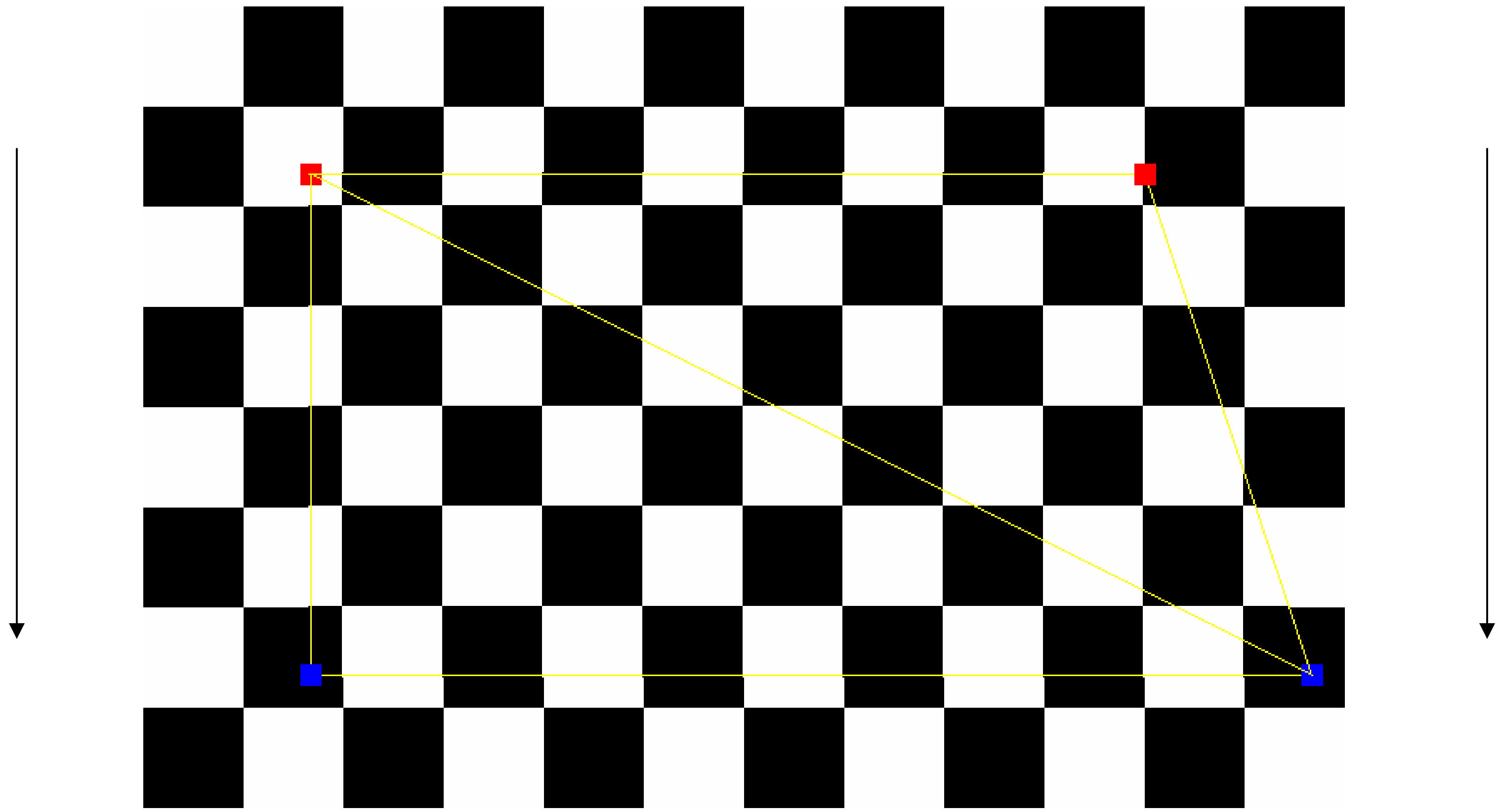
Distortion triangles



Distortion example

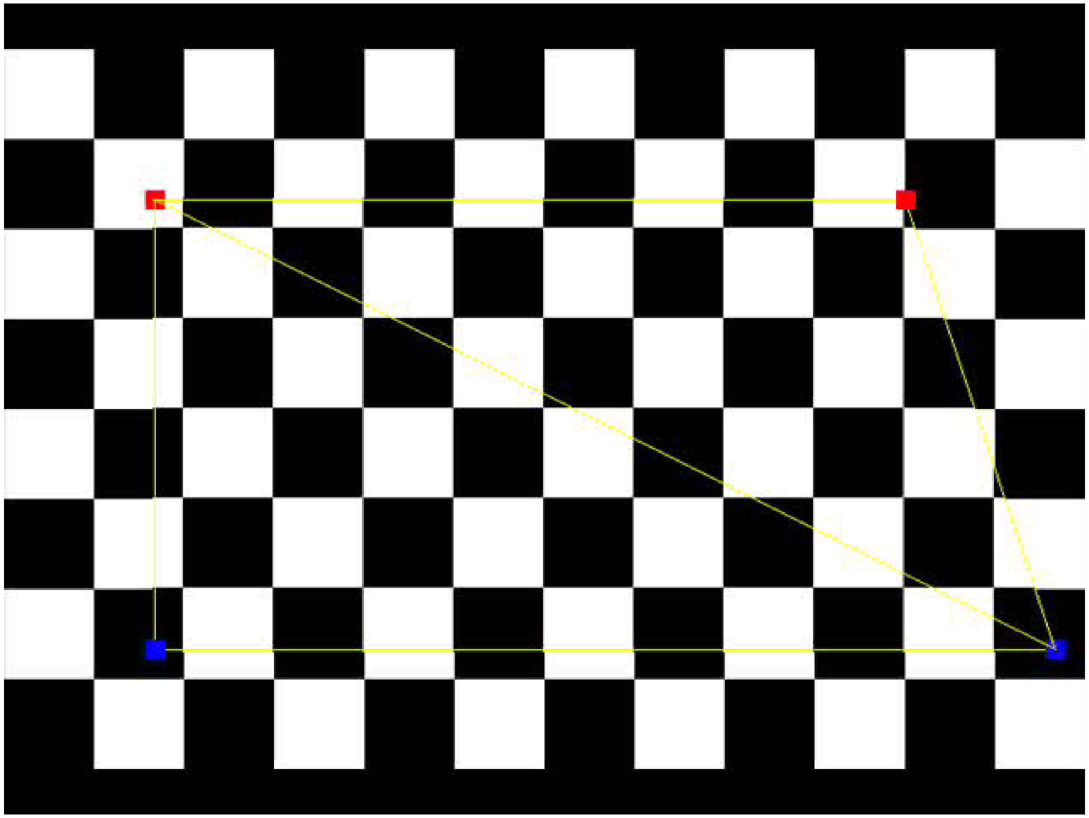


Distortion example



The amount of distortion increases with the depth of the distorted points

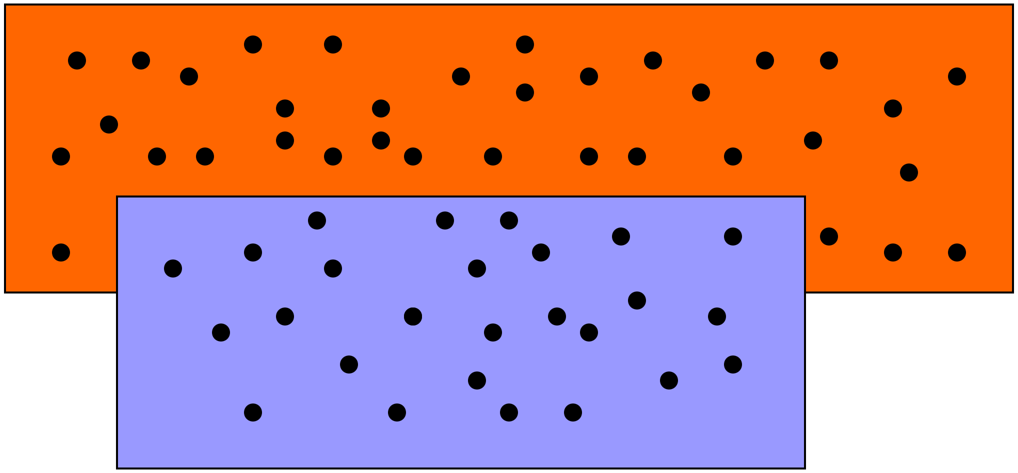
Distortion example



Multi-pole occlusion camera

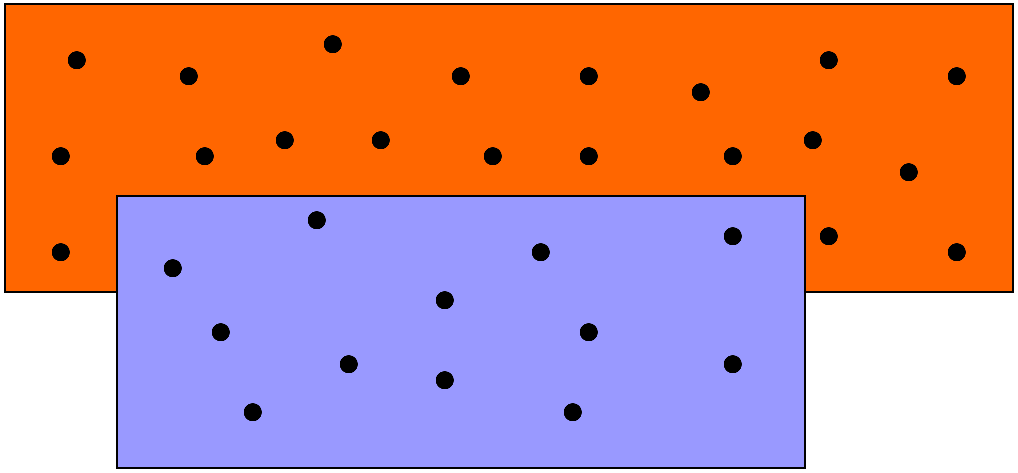
- Camera model
- Construction
- Projection
- Mesh reconstruction
- Results
- Future work

Construction



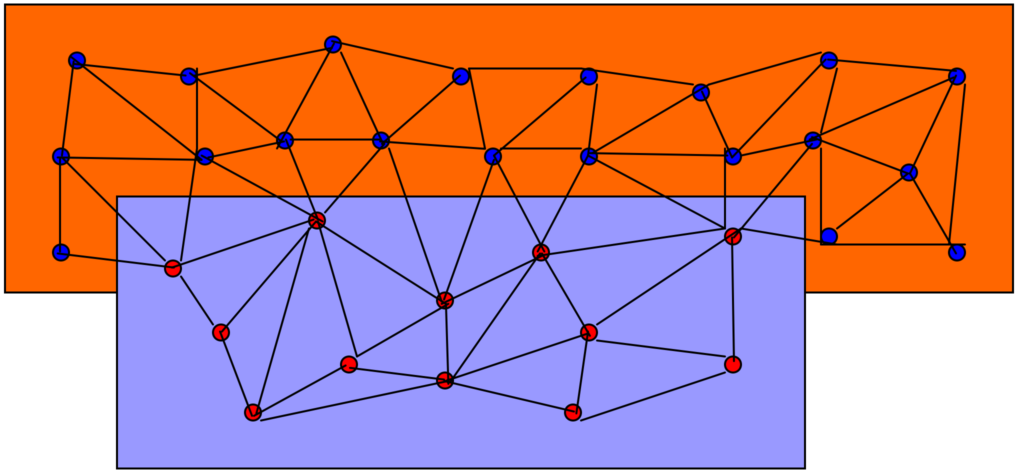
Example scene

Construction



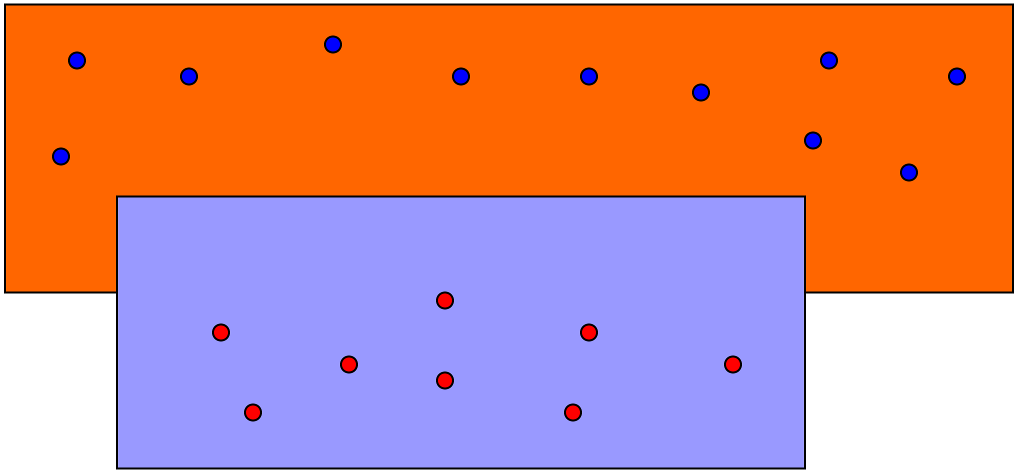
Splat to compute the visible points for the reference view

Construction



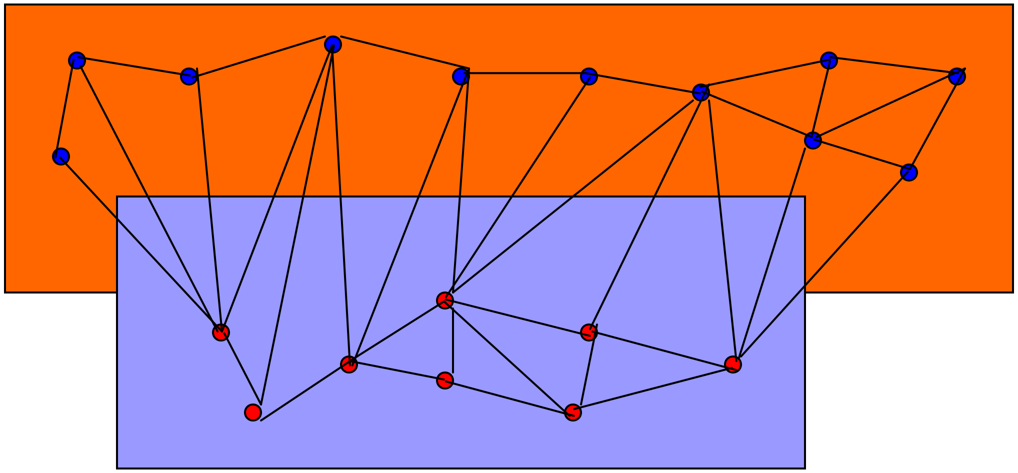
Delaunay triangulate in 2D and mark vertices as front or back

Construction



Eliminate vertices of mixed triangles

Construction



Retriangulate

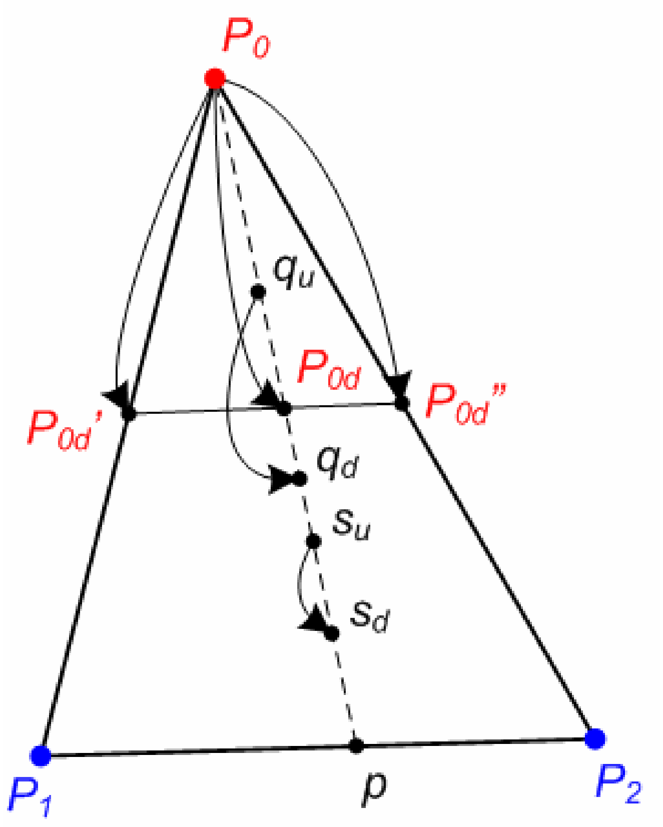
Multi-pole occlusion camera

- Camera model
- Construction
- Projection
- Mesh reconstruction
- Results
- Future work

Projection

- Projection of point P
  - Project onto the reference camera PHC0 to obtain (u0,v0) and the depth z
  - Use (u0,v0) to determine its corresponding distortion triangle Di
  - Distort (u0,v0) to (ud,vd) using the triangle Di

Projection



$$s_u = (u_u, v_u), \quad s_d = (u_d, v_d)$$

$$P_0 = (u_0, v_0), \quad p = (u_p, v_p)$$

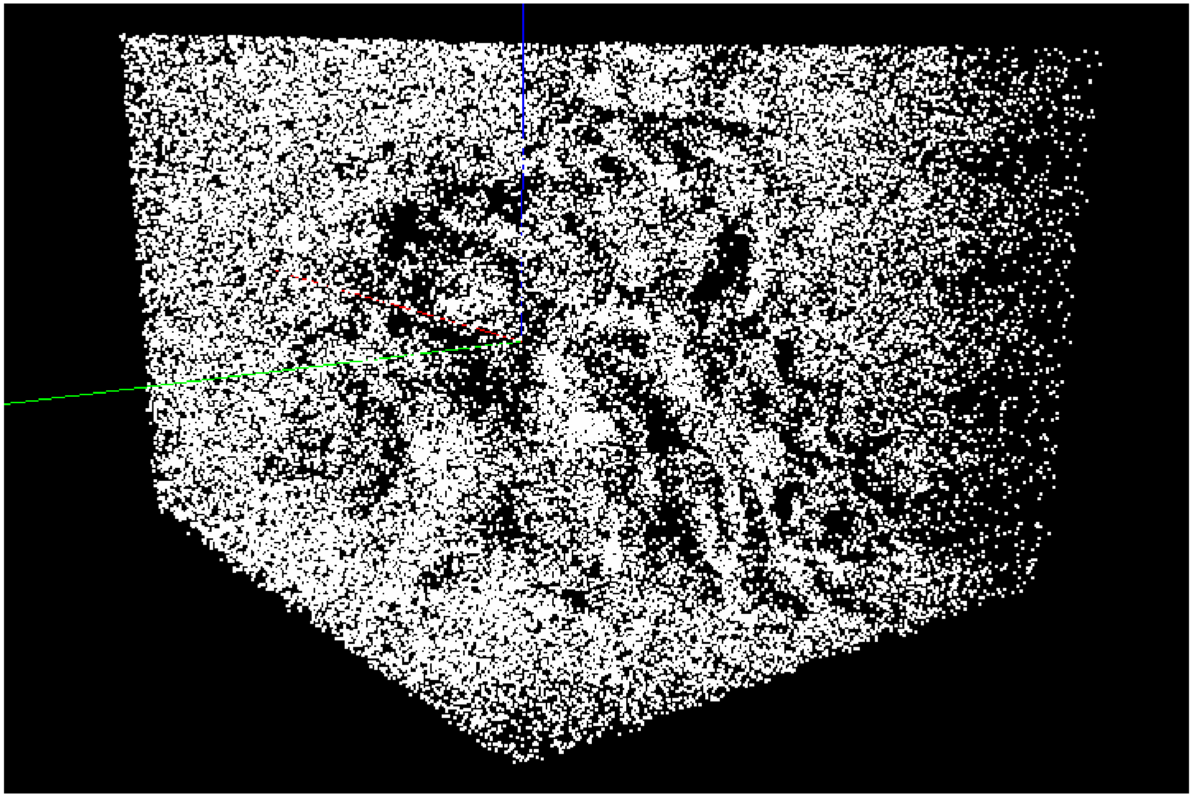
$$(u_d, v_d) = (u_u, v_u) + \left[(u_p, v_p) - (u_u, v_u)\right] f_k\, f_z$$

$$f_z = \begin{cases} 0, & if \quad z \le z_{near} \\ \dfrac{\dfrac{1}{z_{near}} - \dfrac{1}{z}}{\dfrac{1}{z_{near}} - \dfrac{1}{z_{far}}}, & if \quad z_{near} < z < z_{far} \\ 1, & if \quad z_{far} \le z \end{cases}$$

Multi-pole occlusion camera

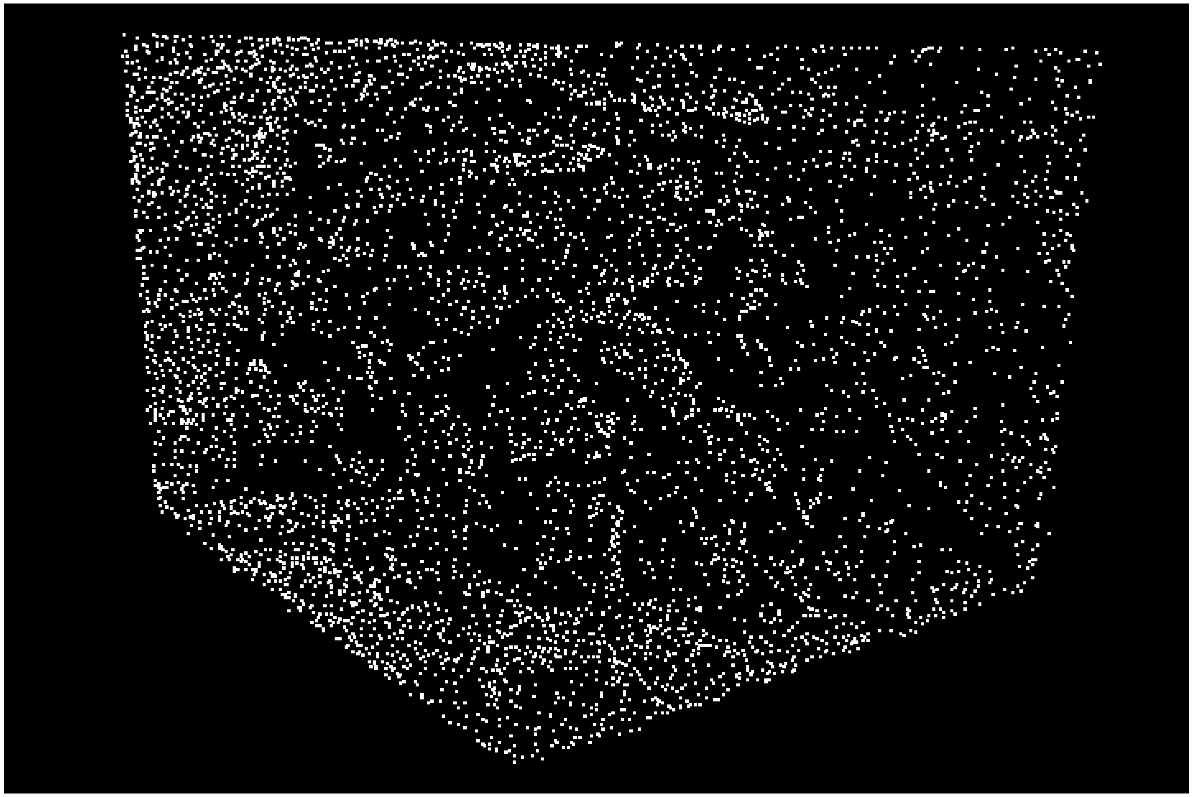
- Camera model
- Construction
- Projection
- Mesh reconstruction
- Results
- Future work

Mesh reconstruction



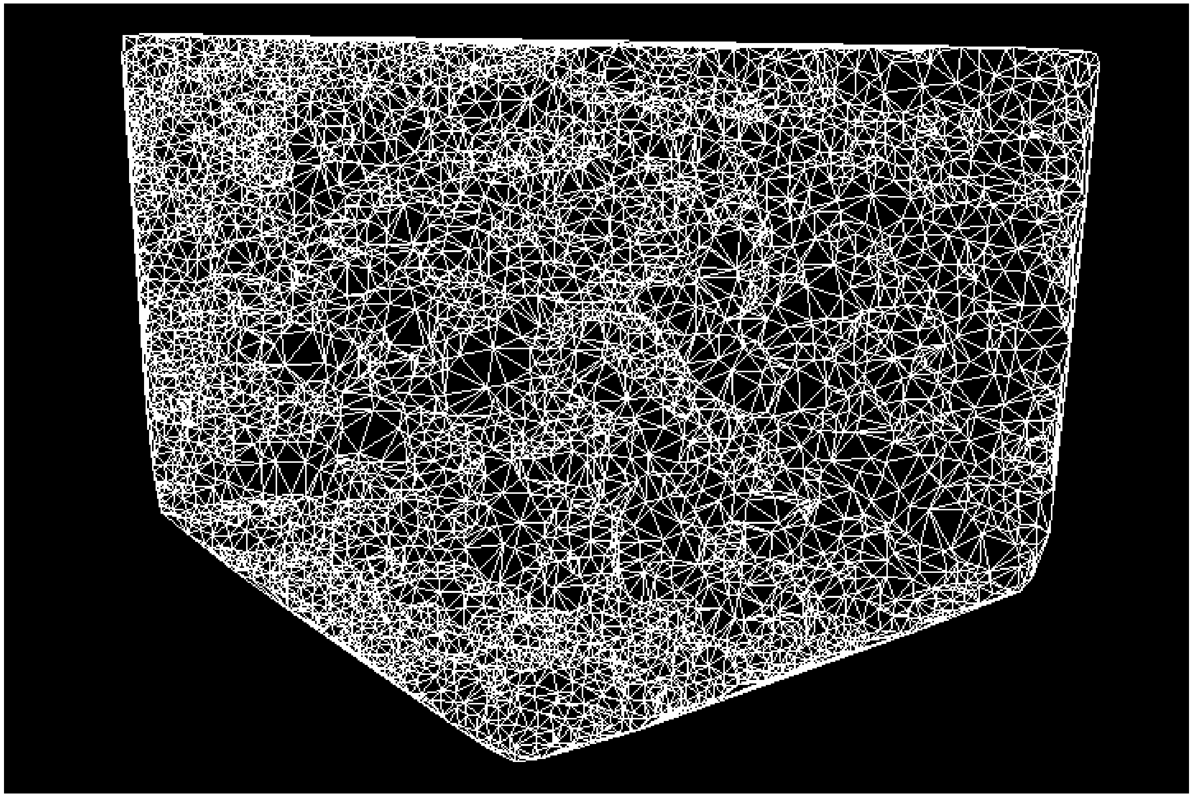
Reconstruction using a pinhole camera

Mesh reconstruction



Compute visible points

Mesh reconstruction



Compute 3D mesh

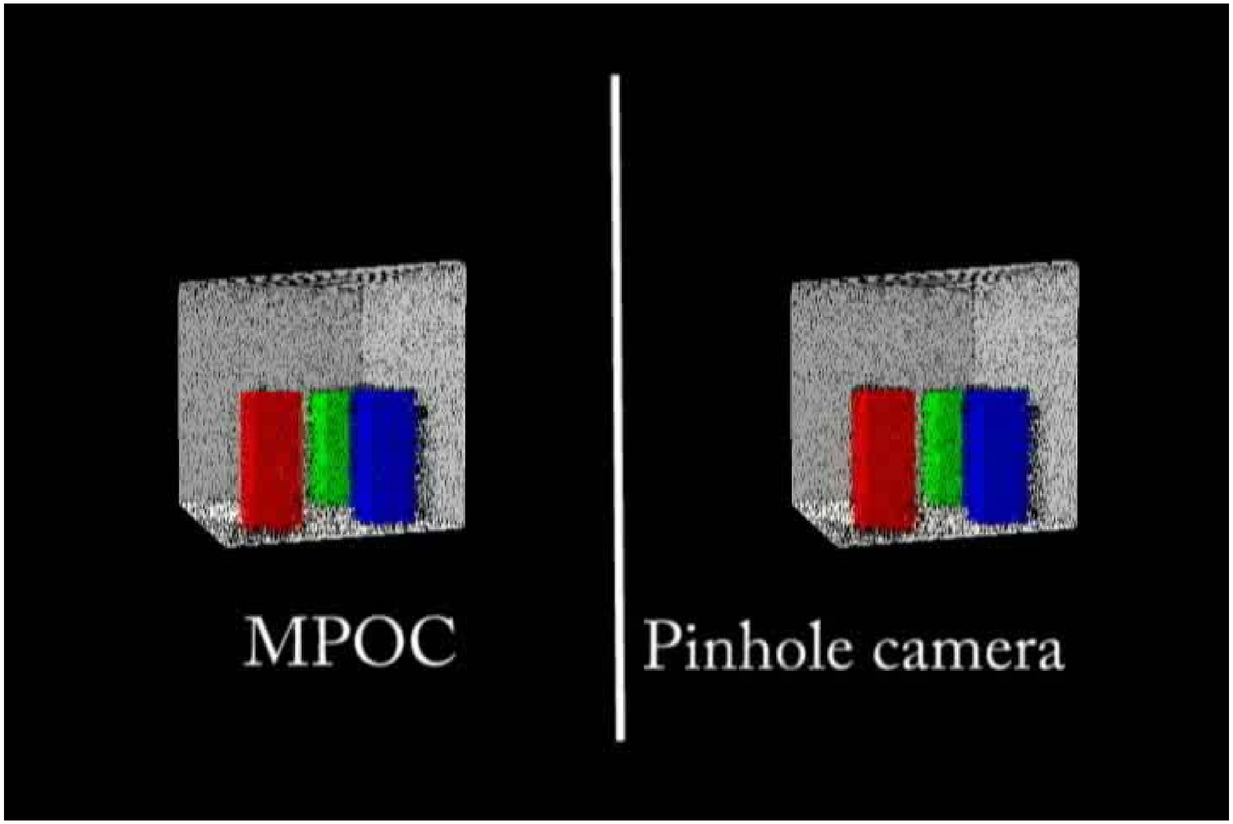
Mesh reconstruction

- Mesh reconstruction using the MPOC
  - Project the points using the MPOC
  - Splat to compute visible points
  - 2D triangulation in the distorted image
- Separate visibility from reconstruction

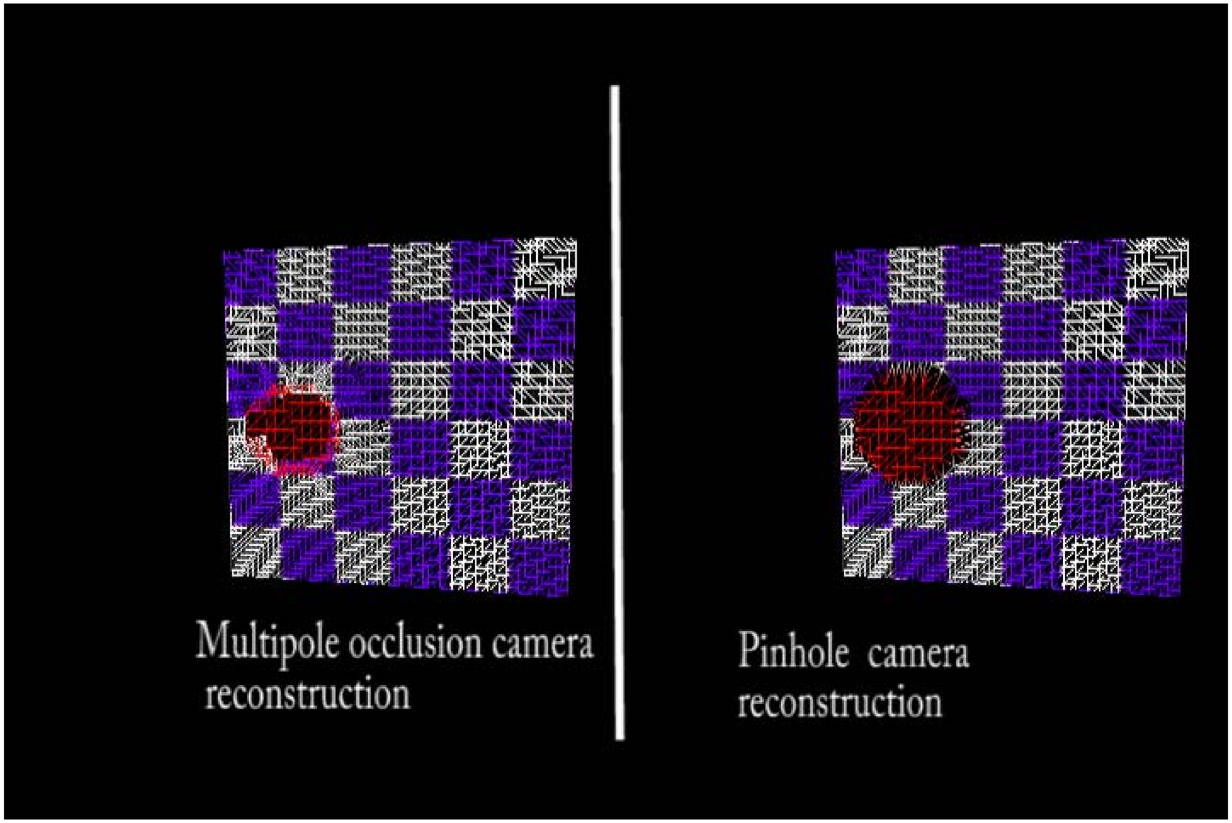
Multi-pole occlusion camera

- Camera model
- Construction
- Projection
- Results
- Future work

Results

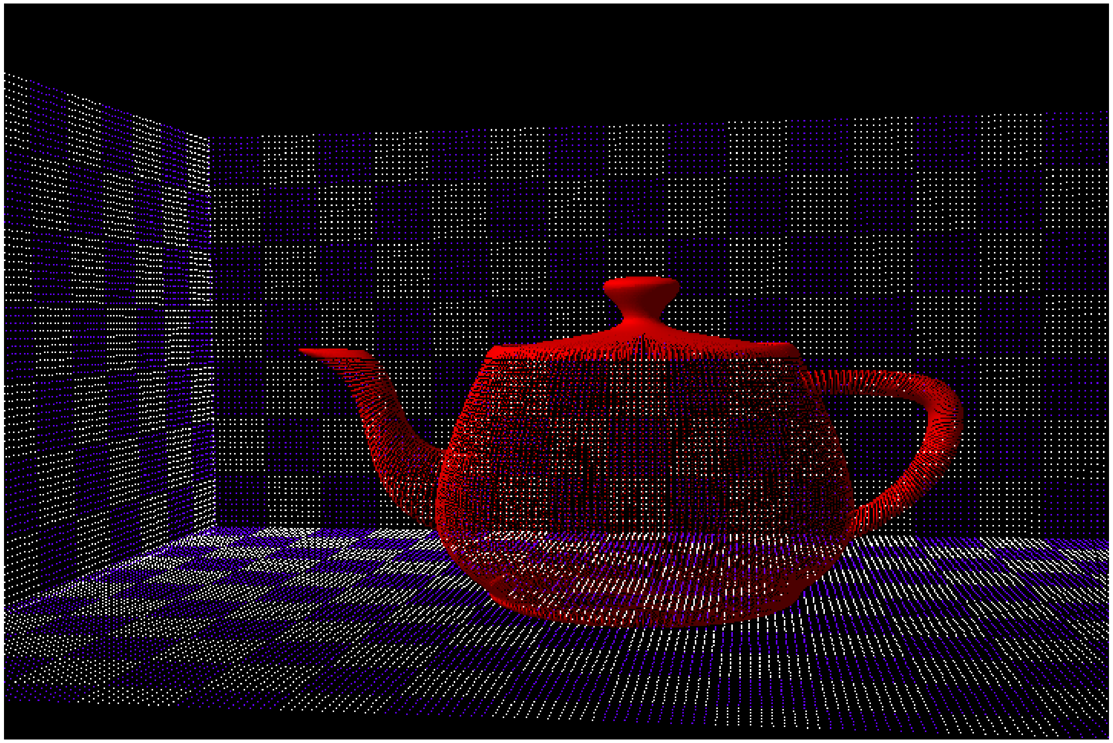


Results



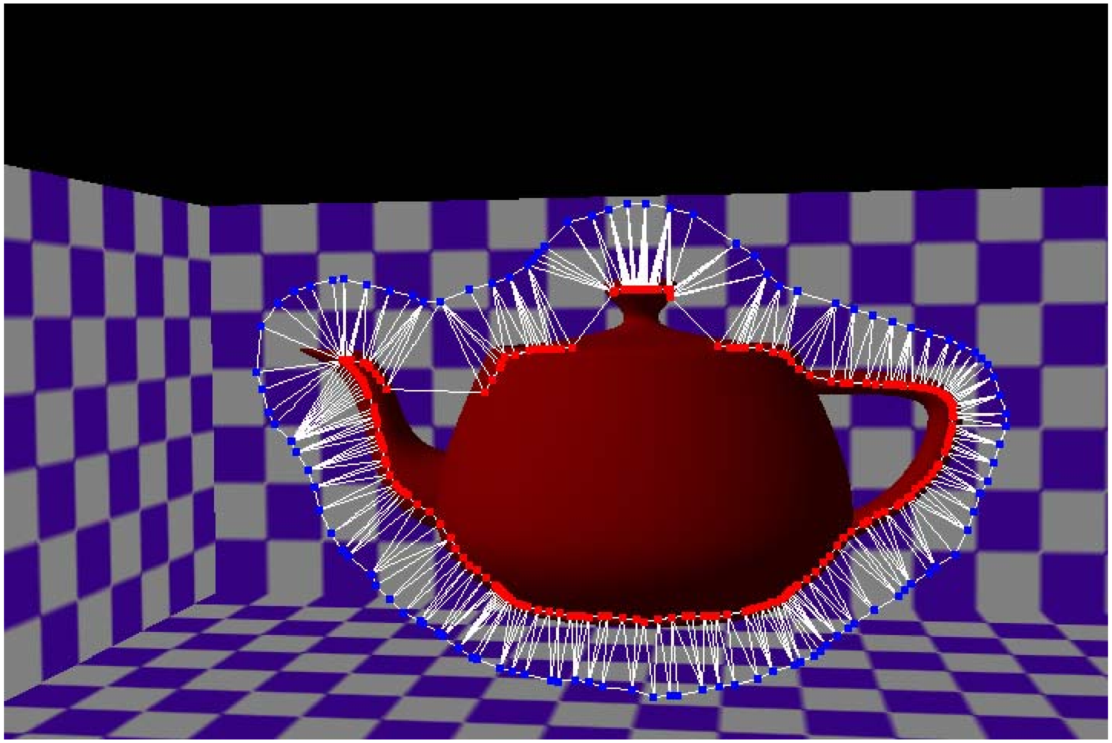
Multipole occlusion camera reconstruction
Pinhole camera reconstruction

Results



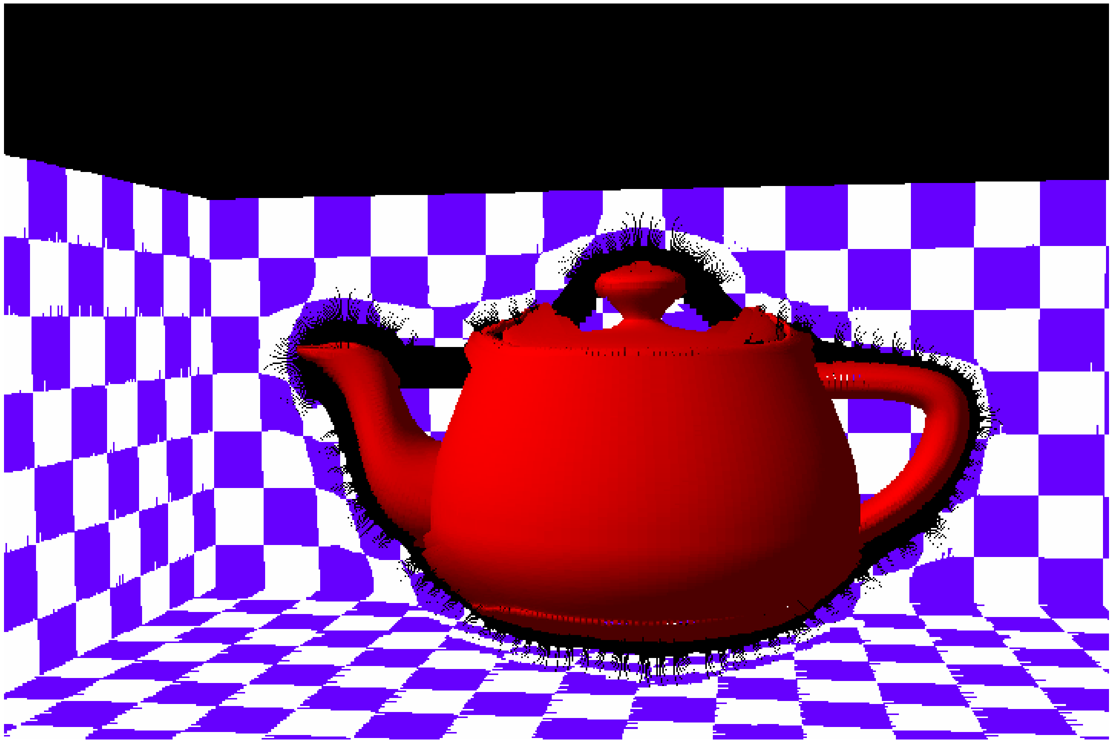
Test case

Results



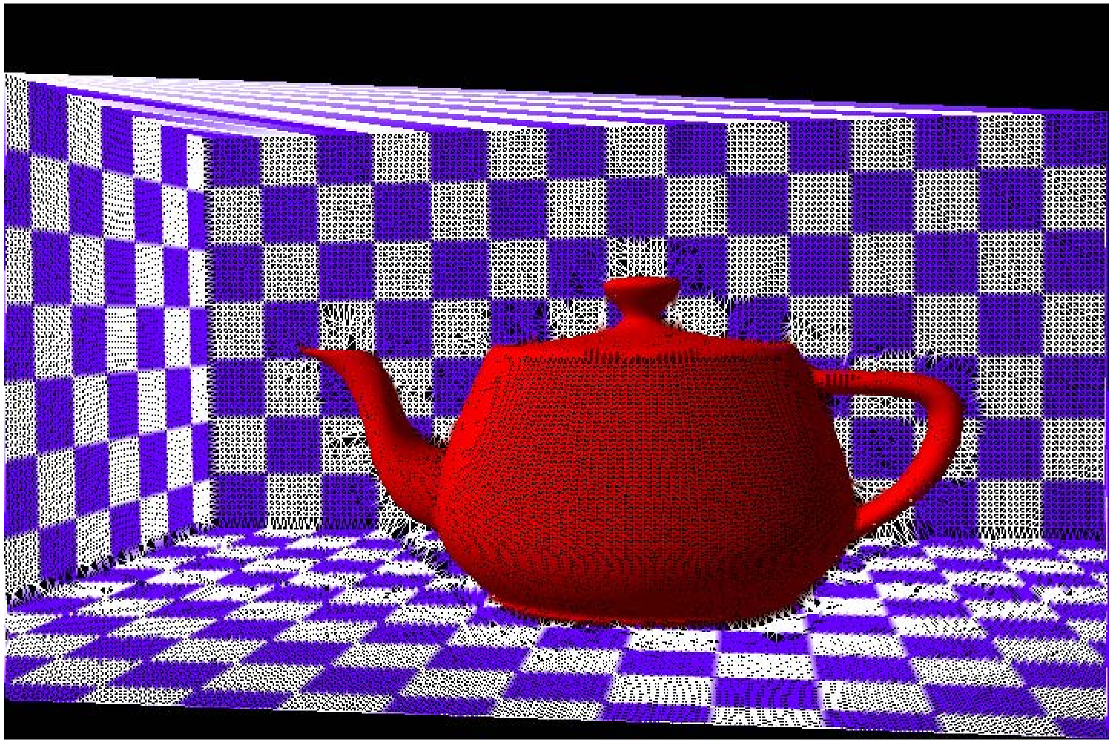
Test scene showing the occlusion camera poles

Results



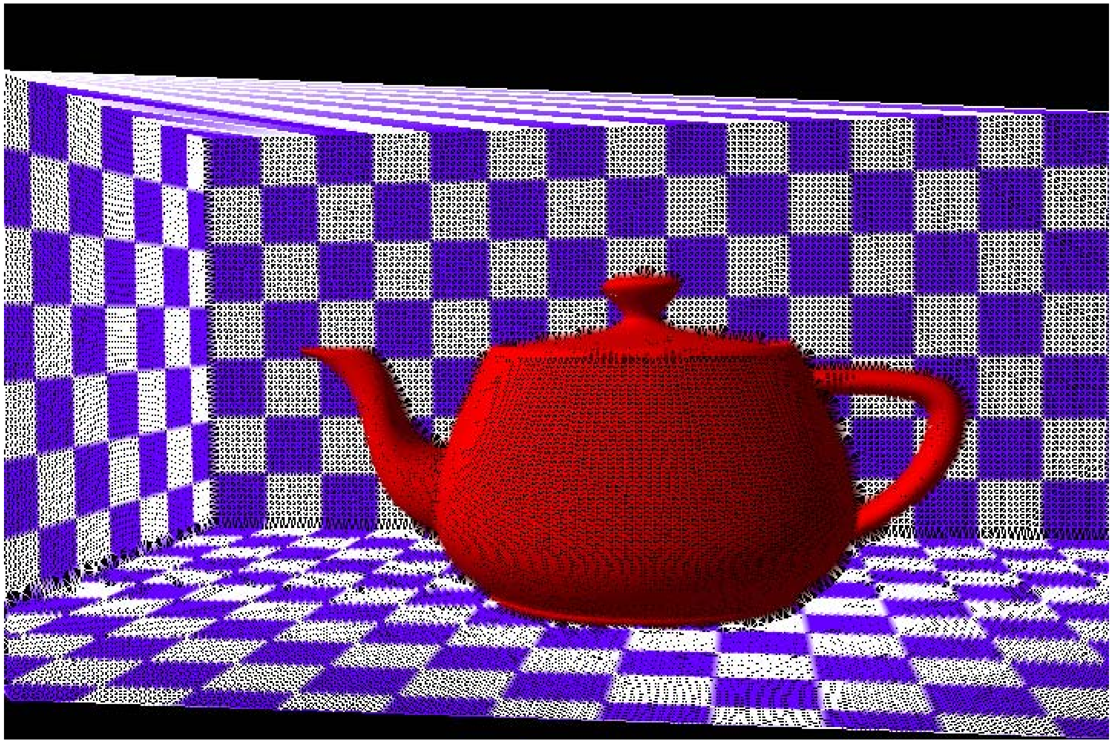
Multipole occlusion camera reference image

Results



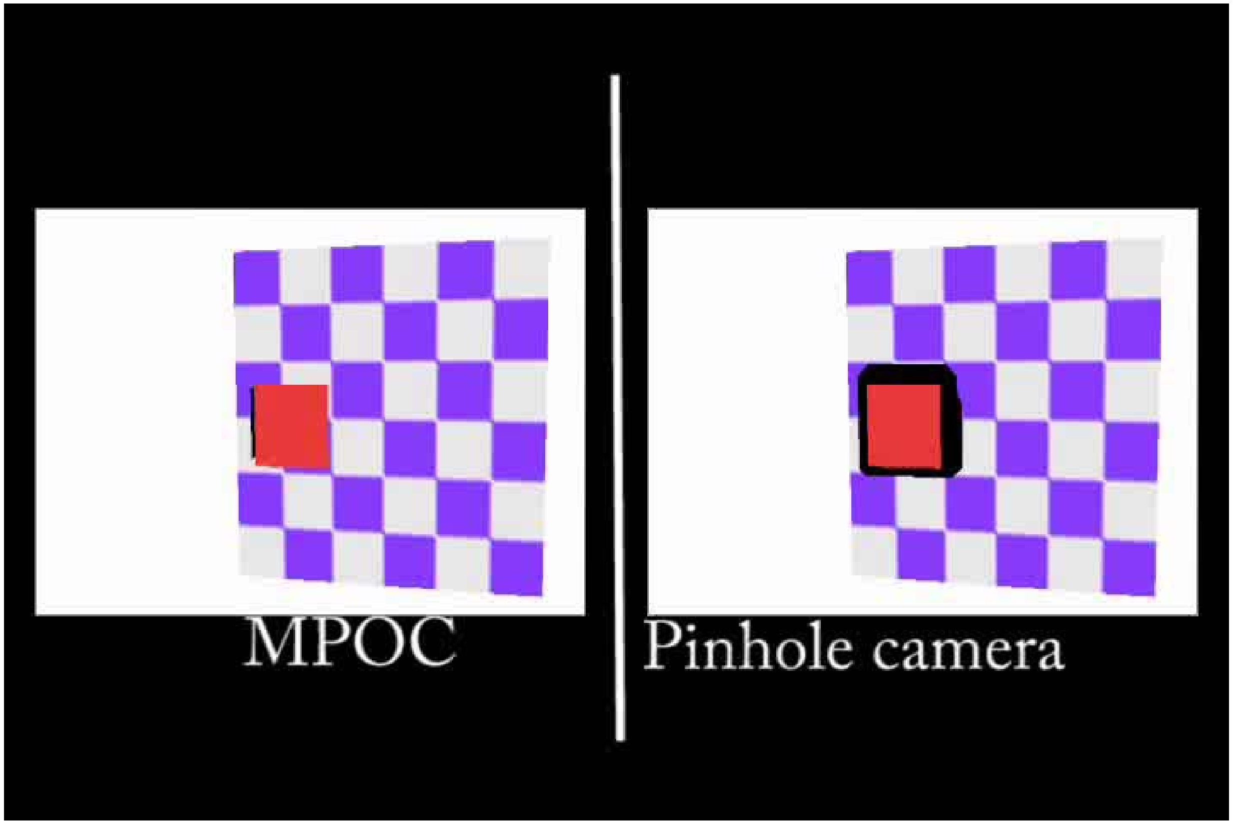
Mesh reconstructed using the multipole occlusion camera

Results



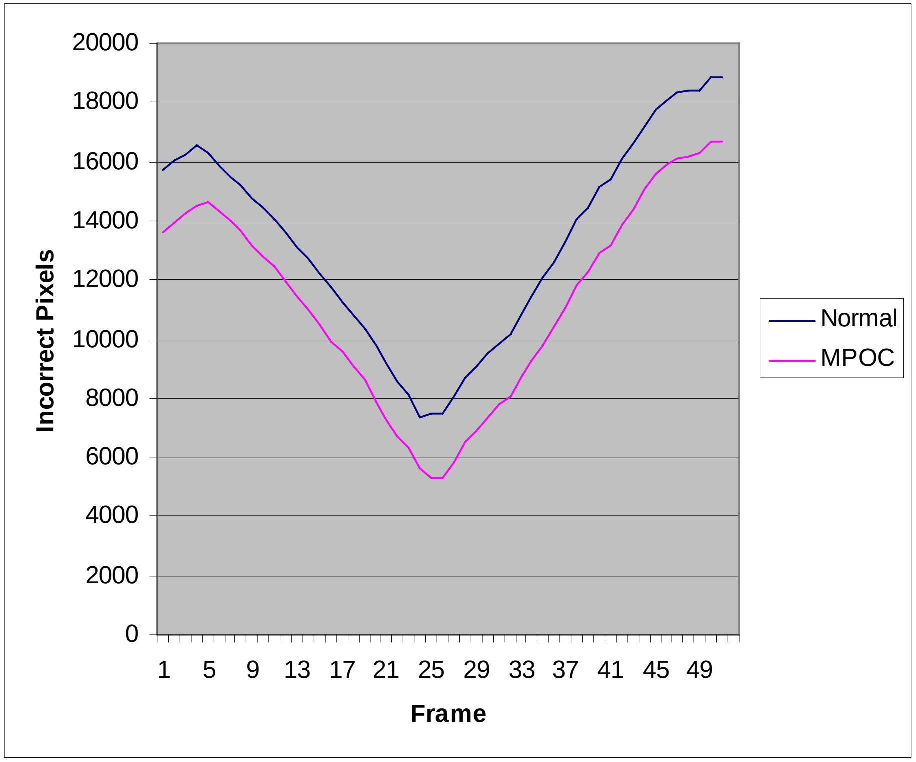
Mesh reconstructed using a pinhole camera

Results



Error comparison

Results



Boxes scene

Errors when using the same mesh for multiple views

Future work

- Better pole selection method
- Speed-up rendering
- Integrate into the ModelCamera

Thank you