Rapid rendering of apparent contours of implicit surfaces for real-time tracking.

Ed Rosten, Tom Drummond

University of Cambridge

Curved surfaces

- Why?
  - Many real-life objects are not polyhedral.
  - We want to track them.

- Real time tracking
  - Need an efficient way of dealing with them.

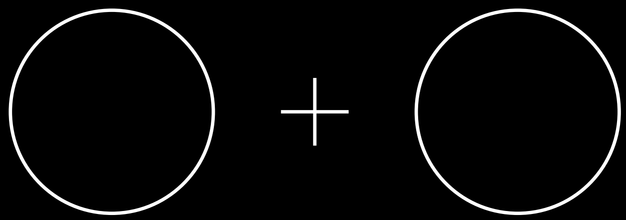
Representing curved surfaces

- Polyhedral approximation.
  - Needs lots of polygons.
- Quadrics (and other curved primitives)
  - Works best for quadric-shaped objects.

Implicit surfaces

Surface defined by $f(\underline{\mathbf{x}}) = 0$, where $f(\underline{\mathbf{x}})$ is a scalar field.

- $f(\underline{\mathbf{x}})$ is a sum of primitives.
  - We use Gaussians.

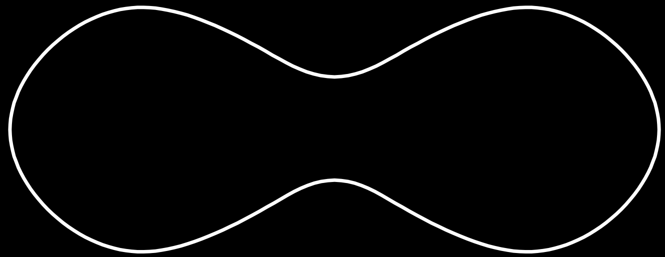


- **Advantage**: Very general purpose.
- **Disadvantage**: Difficult to deal with.

Implicit surfaces

Surface defined by $f(\underline{\mathbf{x}}) = 0$, where $f(\underline{\mathbf{x}})$ is a scalar field.

- $f(\underline{\mathbf{x}})$ is a sum of primitives.
  - We use Gaussians.

$=$ 

- **Advantage**: Very general purpose.
- **Disadvantage**: Difficult to deal with.

Using implicit surfaces

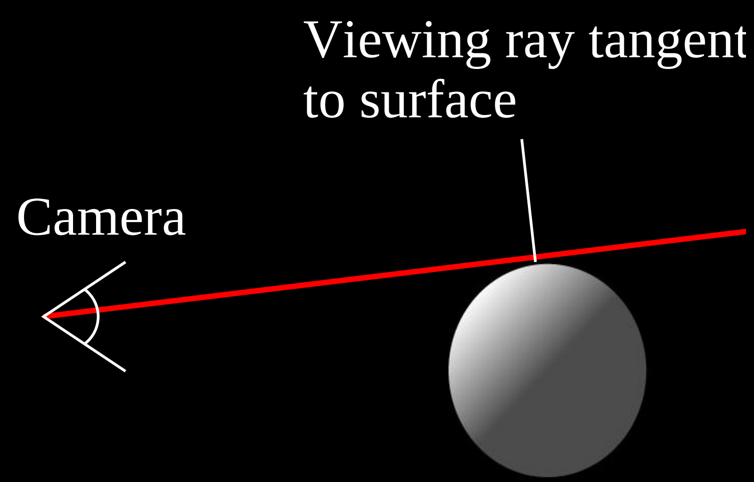
- Need to choose a feature to track.
- Apparent contour is the key feature.

- For real-time tracking, this must be calculated rapidly.

Apparent contour

Which points lie on the apparent contour?

1. Point is on the surface:
   - $f(\underline{\mathbf{x}}) = 0.$
2. Viewing ray is tangent to the surface:
   - $\nabla f(\underline{\mathbf{x}}) \cdot \underline{\mathbf{x}} = 0.$



3. Point is not occluded by the shape.

Approach

1. **Solve the first two conditions first.**
2. Then find which parts are visible.

Apparent contour

- Simple, inefficient methods
  1. Test all space.
  2. Precompute surface points then test each one.

- We present an efficient method
  - Generate whole contour from starting point.
  - Can derive a differential equation to do this.

Apparent contour

- Conditions for point on apparent contour:

$$f(\underline{\mathbf{x}}) = 0$$

$$\nabla f(\underline{\mathbf{x}}) \cdot \underline{\mathbf{x}} = 0$$

- We present an efficient method
    - Generate whole contour from starting point.
    - Can derive a differential equation to do this.

Apparent contour

- Conditions for point on apparent contour:

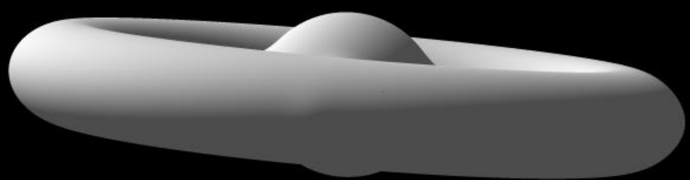
$$f(\underline{\mathbf{x}}) = 0$$

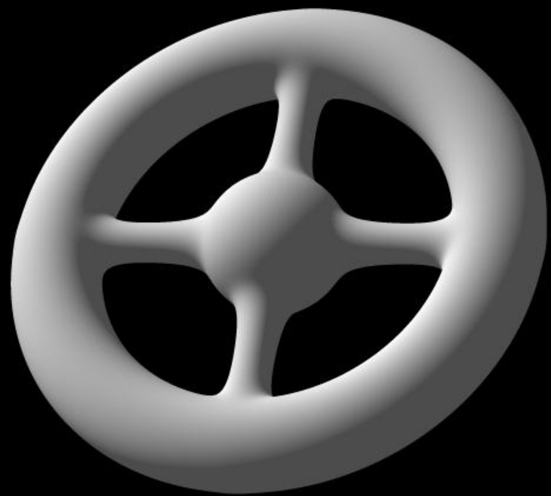
$$\nabla f(\underline{\mathbf{x}}) \cdot \underline{\mathbf{x}} = 0$$

- We present an efficient method
  - Generate whole contour from starting point.
  - Can derive a differential equation to do this.
  - $\dot{\underline{\mathbf{x}}} = \widehat{\mathcal{H}[f(\underline{\mathbf{x}})]\,\underline{\mathbf{x}} \times \nabla f(\underline{\mathbf{x}})}$

Apparent contour

- Differential equation for the apparent contour:
  - $\dot{\underline{\mathbf{x}}} = \widehat{\mathcal{H}[f(\underline{\mathbf{x}})]\,\underline{\mathbf{x}} \times \nabla f(\underline{\mathbf{x}})}$
- Solve with 4$^{\text{th}}$ order Runge-Kutta solver.





Apparent contour

- Differential equation needs boundary conditions.

- Precompute sparse point set on the surface.

- Find a point where $\widehat{\nabla f(\underline{\mathbf{x}})} \cdot \hat{\underline{\mathbf{x}}} \approx 0$
  - Move it on to the contour.

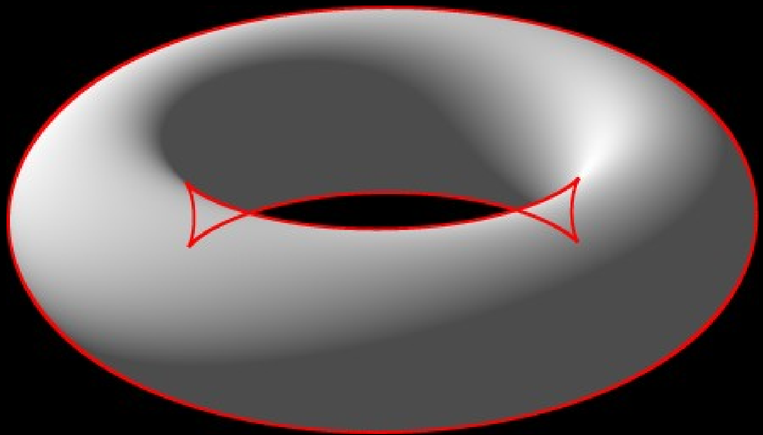
- Avoid calculating contours more than once.
  - Reject points near existing contours.
  - Efficient rejection using fast search.

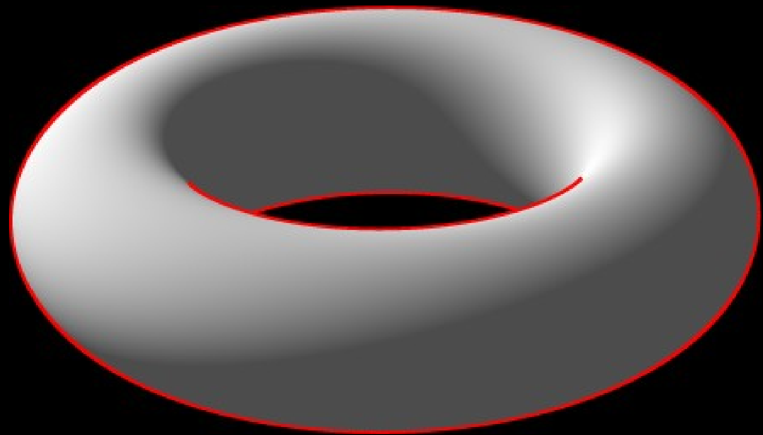
Approach

1. Solve the first two conditions first.
2. **Then find which parts are visible.**

Contour visibility

- Contours are correctly drawn.
- Too much is visible.

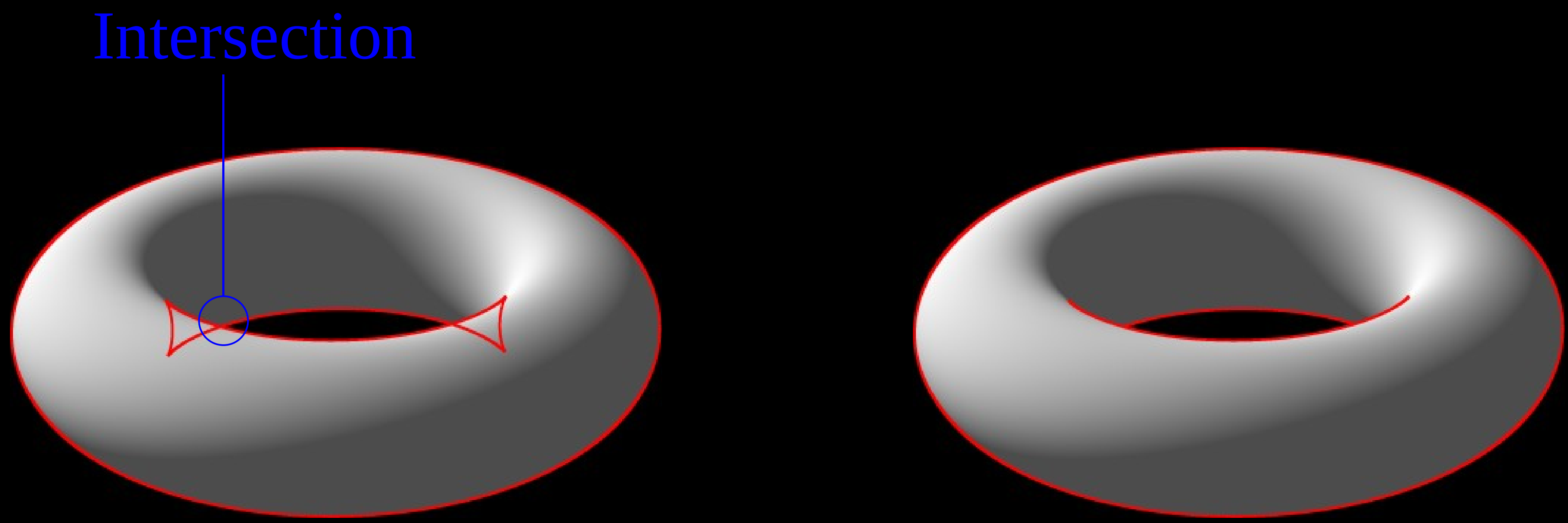




- Search for surface along ray is *very* inefficient.
- Strong constraints on how the visibility can change.

Contour visibility

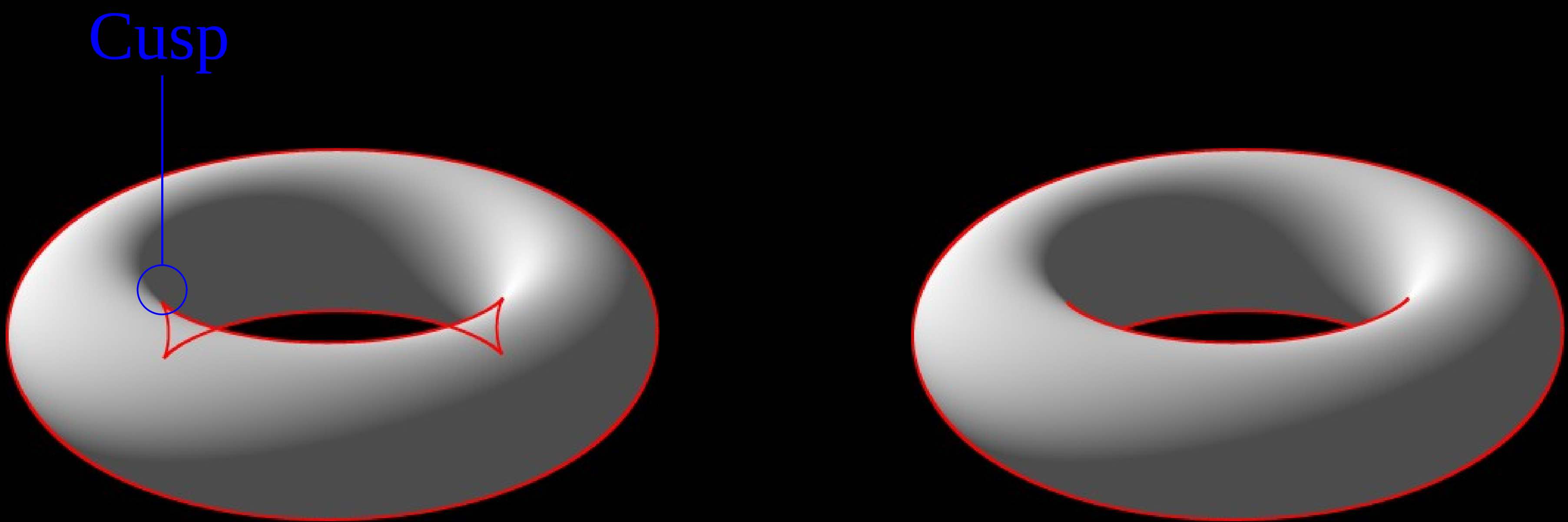
- Contours are correctly drawn.
- Too much is visible.



- Search for surface along ray is *very* inefficient.
- Strong constraints on how the visibility can change.

Contour visibility

- Contours are correctly drawn.
- Too much is visible.

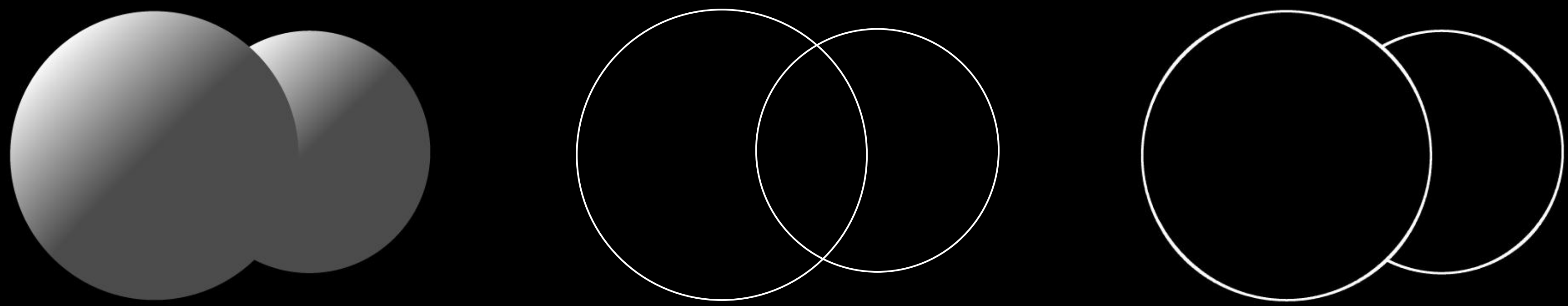


- Search for surface along ray is *very* inefficient.
- Strong constraints on how the visibility can change.

Occlusion depth

Defined as: *The number of surfaces between a point on the apparent contour and the camera.*

- Depth can change by ± 2 at an intersection.

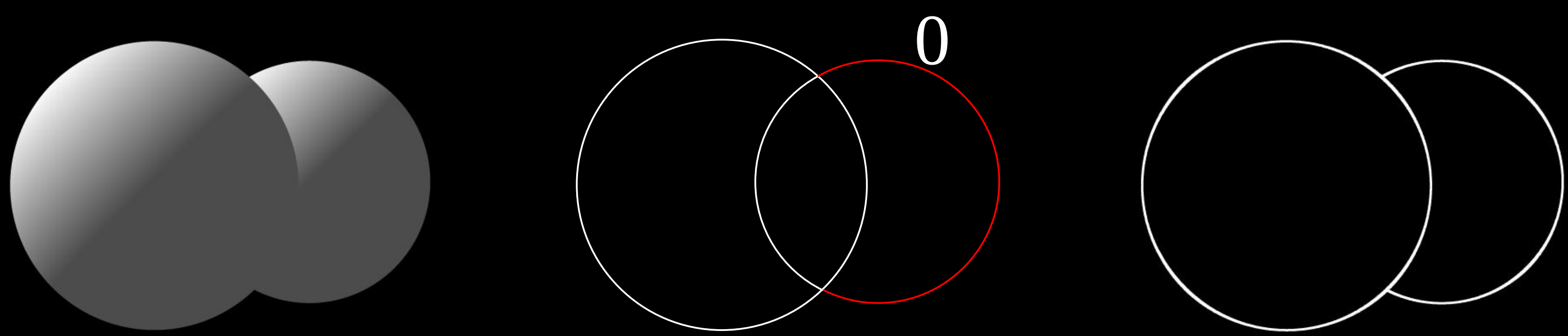


- This corresponds to an occlusion.

Occlusion depth

Defined as: *The number of surfaces between a point on the apparent contour and the camera.*

- Depth can change by ± 2 at an intersection.

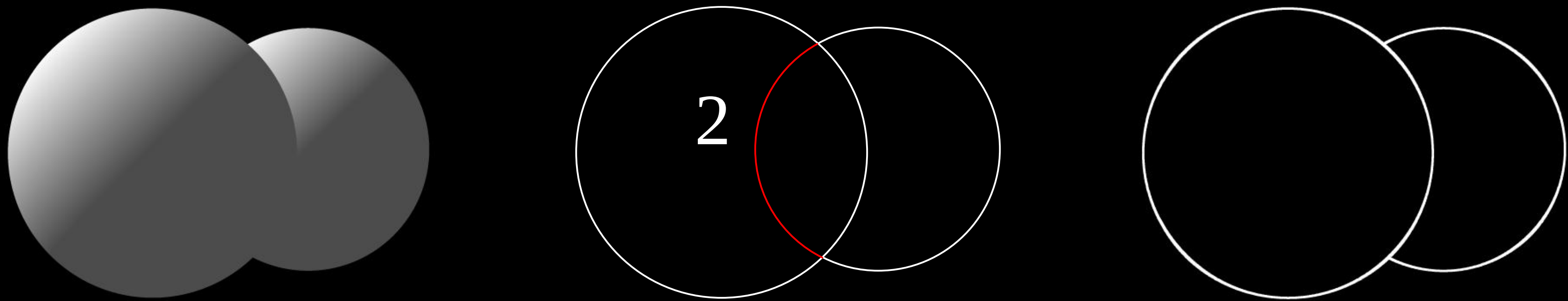


- This corresponds to an occlusion.

Occlusion depth

Defined as: *The number of surfaces between a point on the apparent contour and the camera.*

- Depth can change by ± 2 at an intersection.

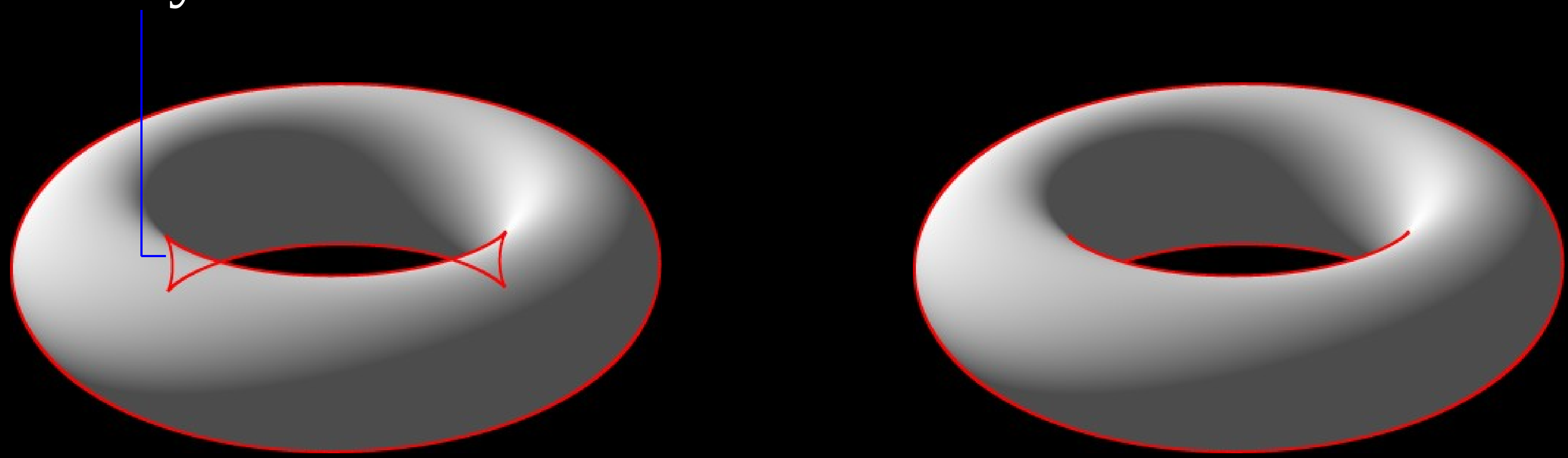


- This corresponds to an occlusion.

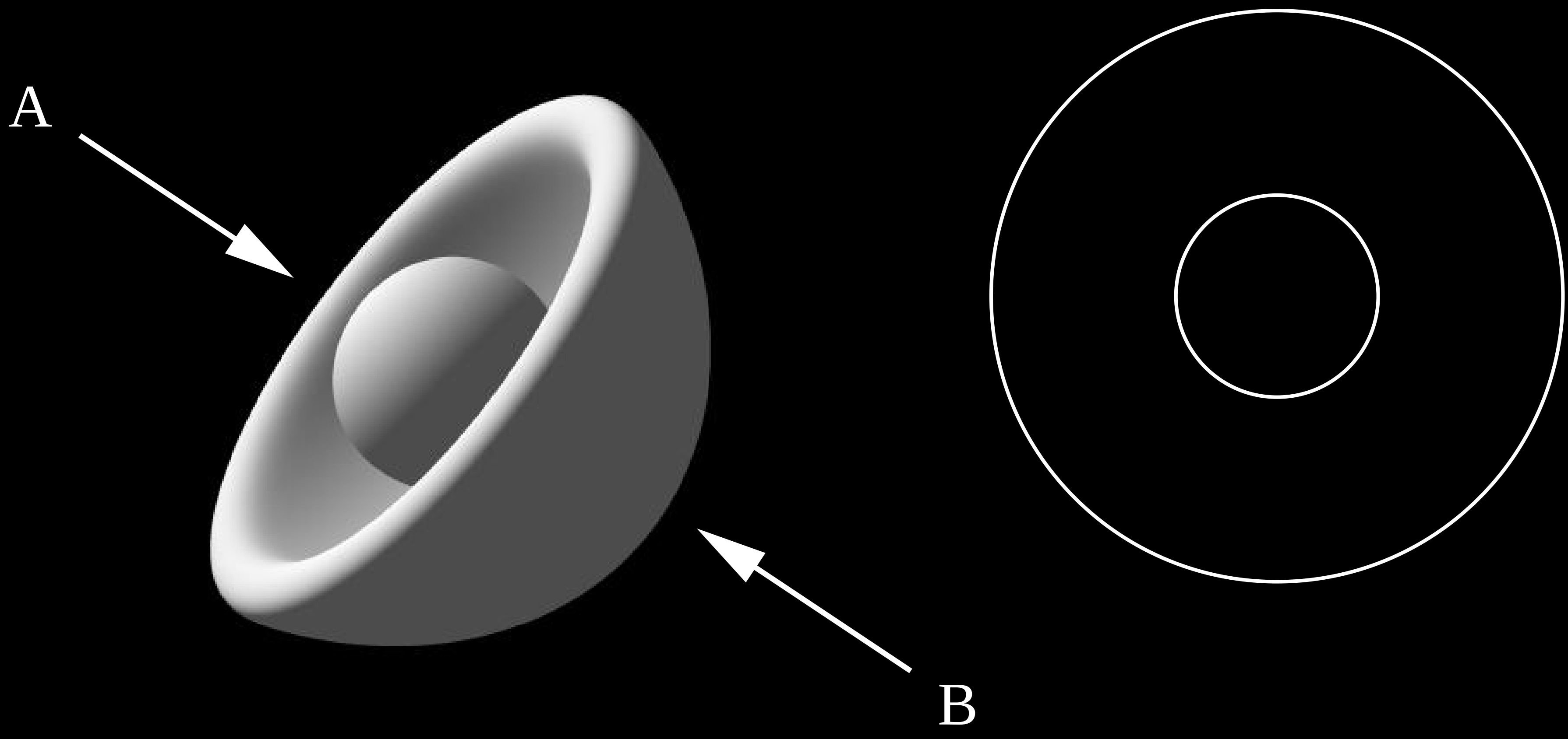
Occlusion depth

Defined as: *The number of surfaces between a point on the apparent contour and the camera.*

- Depth can change by ± 1 at a cusp



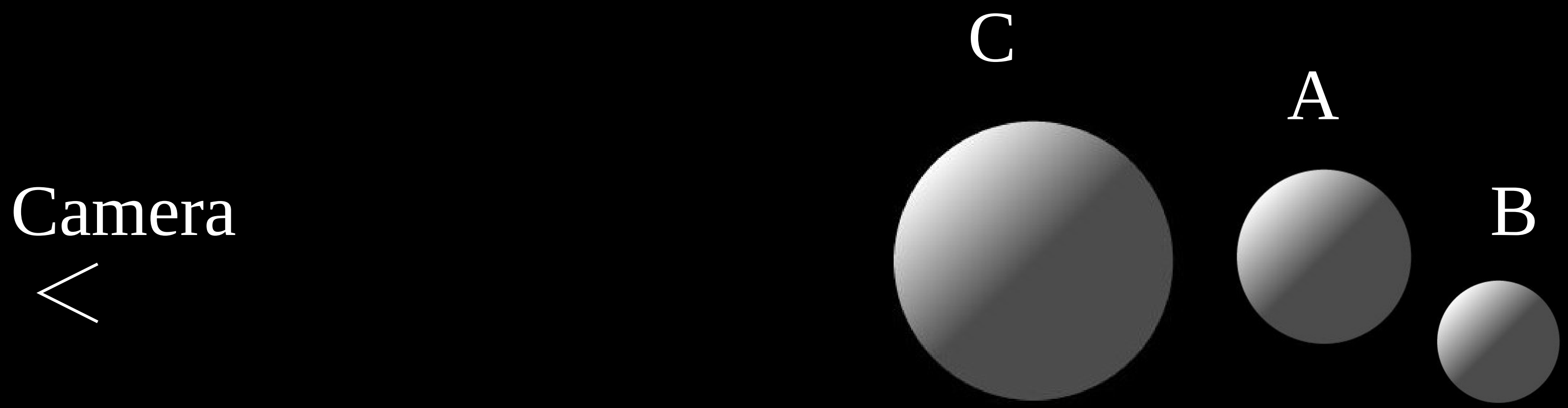
Constant of integration

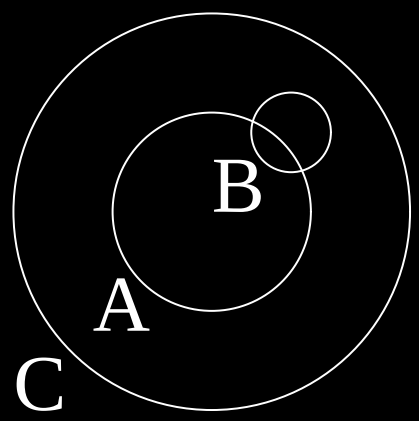


This cannot be determined completely without searching.

Avoiding searches

The amount of searching can be further reduced.





A partially hides B

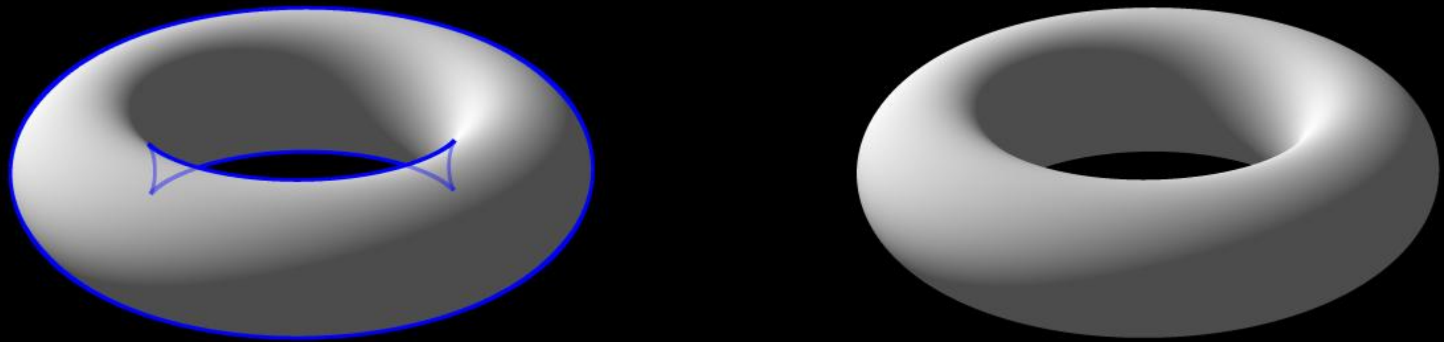
Searching shows C hides A

A is hidden

∴ B is hidden

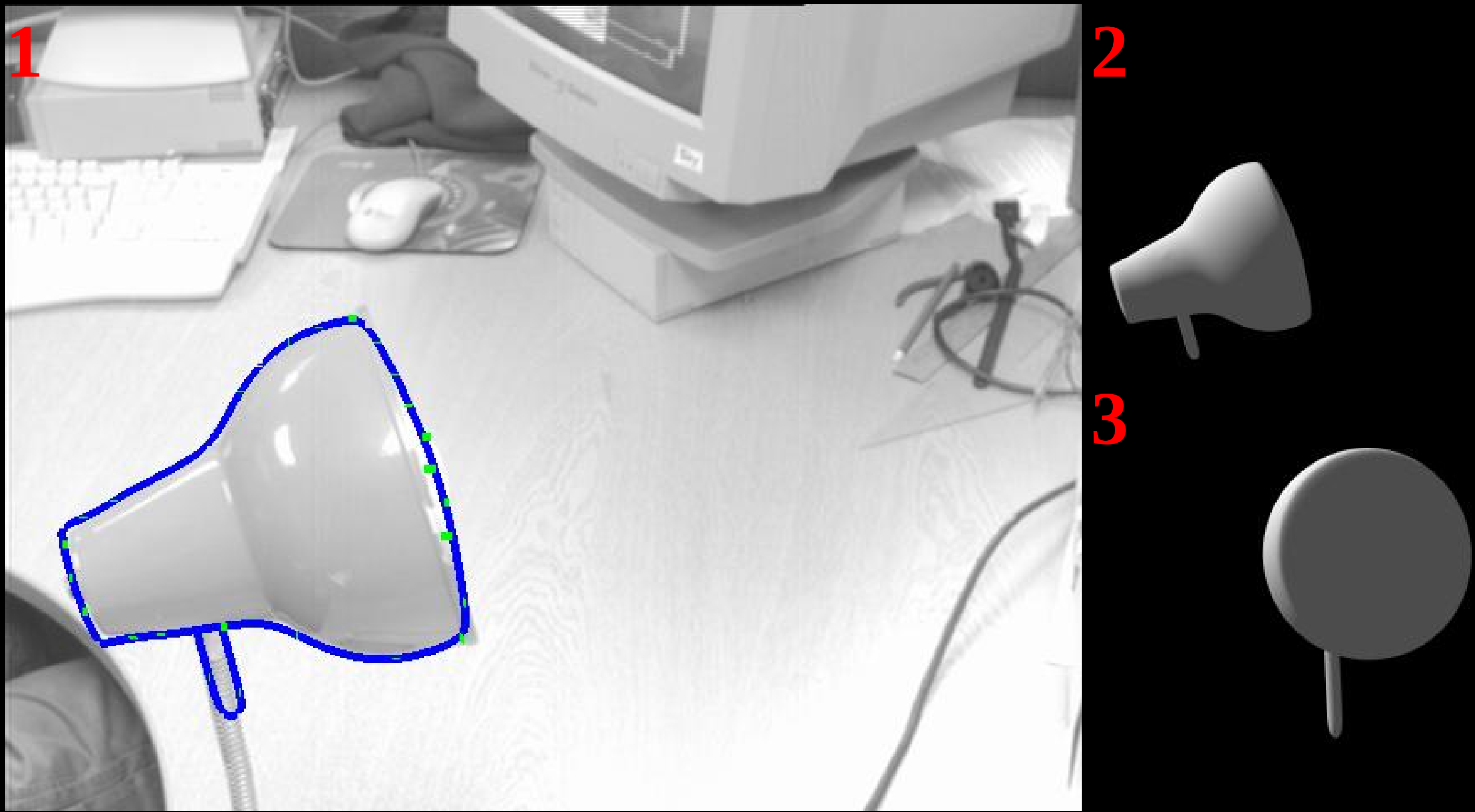
Tracking the contour

- Image motion of contour due to:
  - Change in viewpoint.
  - Slip of contour.



- Tracking system linearizes image motion with respect to 6 pose parameters.
- Slip is not visible in linear approximation.

Results



1. The tracker in operation.
2. Lamp rendered from the camera's view.
3. A synthetic view from in front of the lamp.

Any questions?