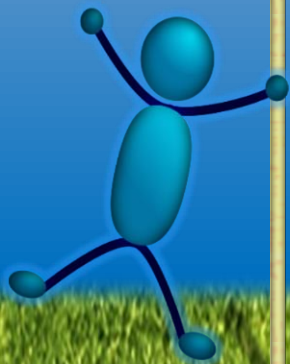


Infinite Resolution Textures










Alexander Reshetov David Luebke

NVIDIA












Motivation:

Computer Graphics at 30,000 feet










DISTANCE ASSETS			
3D Models			
2D Textures			

Motivation:

Computer Graphics at 30,000 feet

DISTANCE ASSETS			
3D Models			
2D Textures			

Method & Apparatus

DISTANCE ASSETS			
3D Models			
Infinite Resolution Textures			



IR texture

= raster image

+ silhouettes @ grid



```
float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
```

IRT from an application standpoint

- instead of

```
float4 color = colorMap.SampleLevel(colorSampler, uv, lod);
```

- use

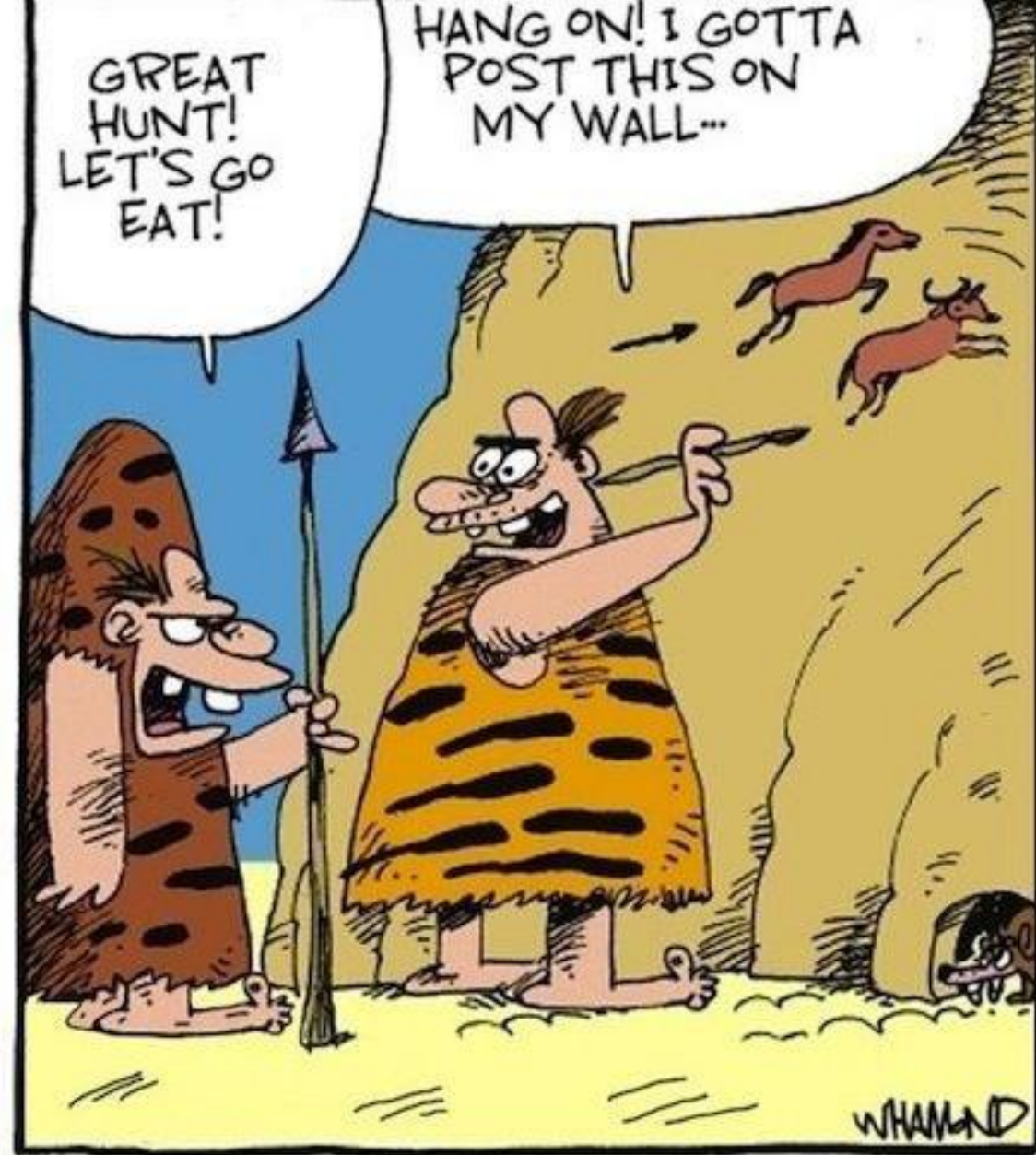
```
float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
```

IRT calculates **duv** at runtime by evaluating distances to the precomputed silhouette edges

Just by tempering **duv**, we can blend between

- IRT (@ closeups) and
- traditional textures at a distance

Prior Art



Early Facebook

it can be filtered



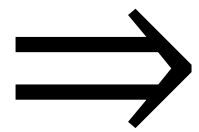
$$\text{IRT} = \text{edges} + \text{raster image}$$



hi freq



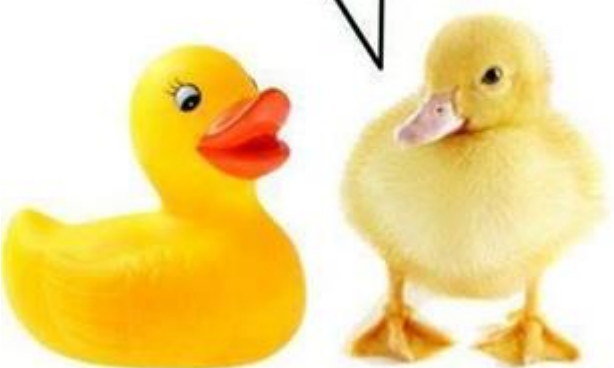
low freq



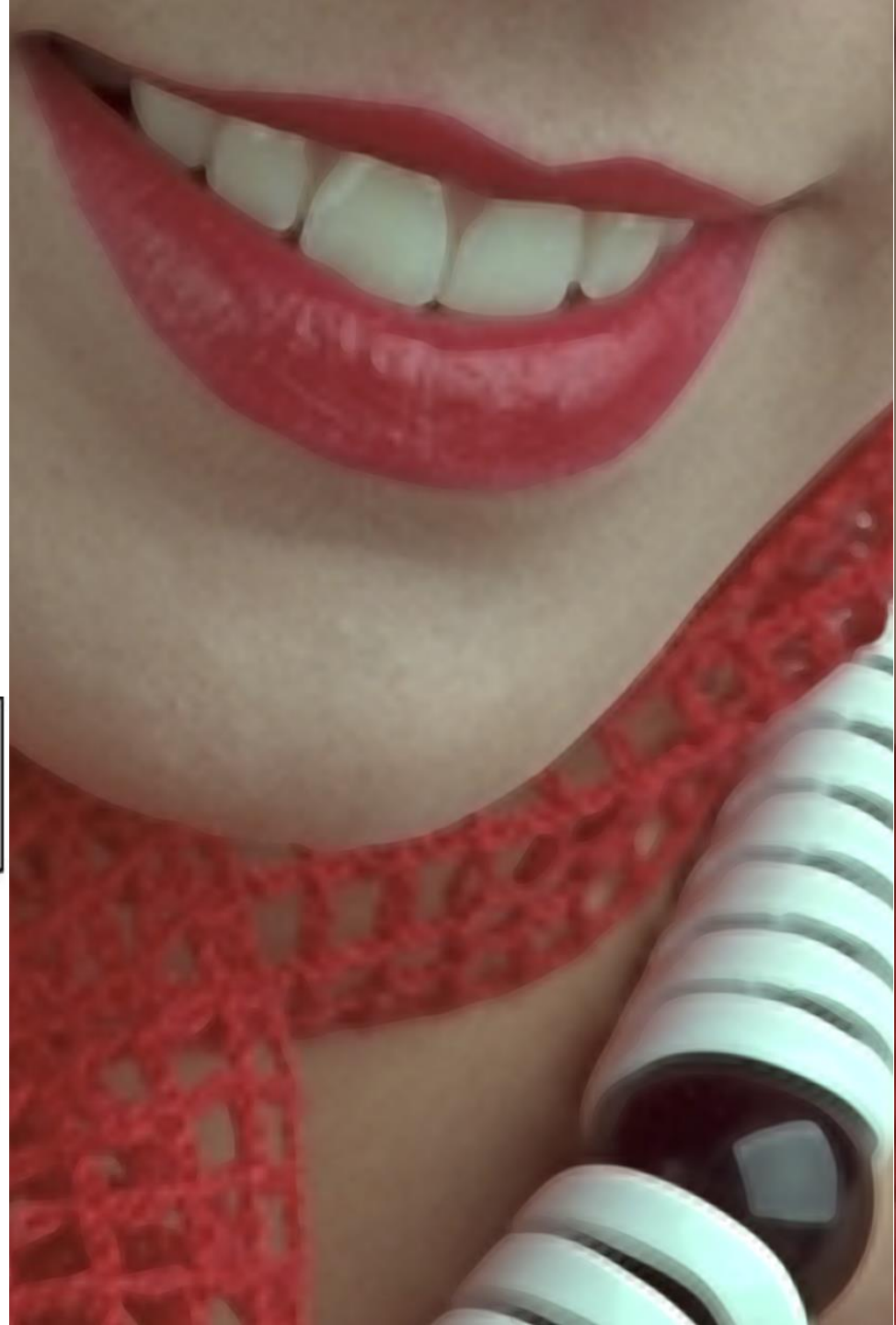
IRT



I CAN'T BELIEVE YOU GOT PLASTIC SURGERY!!!

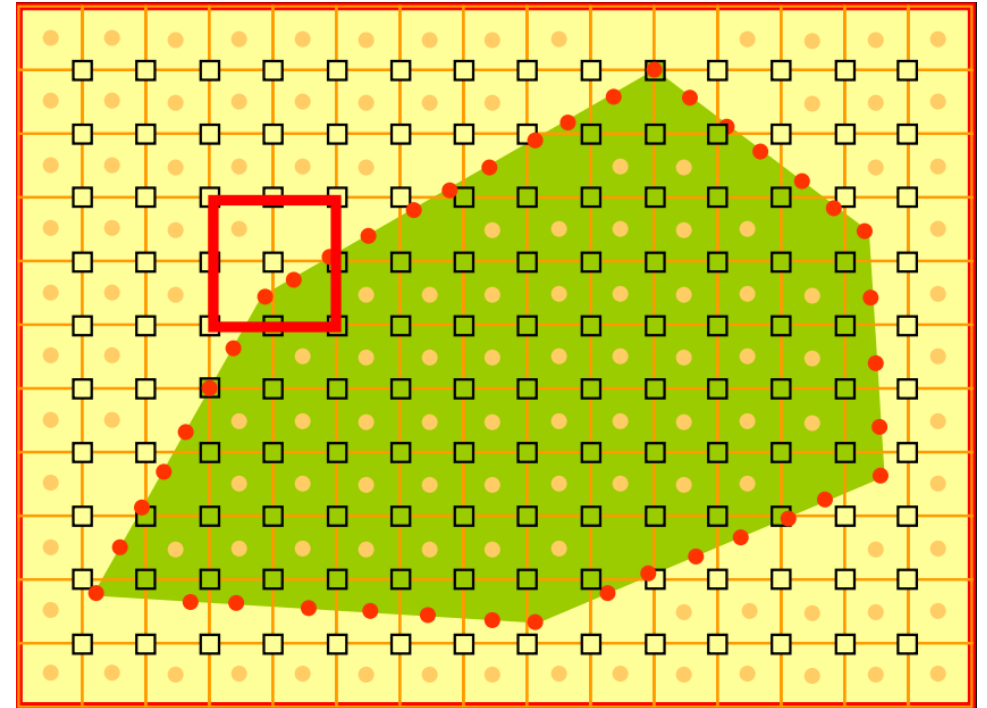


pinned from Scarlett Image



Prior Art circa 2005

- **Silmaps**
Pradeep Sen
- Bixels
*Jack Tumblin,
Prasun Choudhury*
- Vector Texture Maps
Nicolas Ray et al
Curvilinear Contours
Stefan Gustavson
- Pinchmaps
*Marco Tarini,
Paolo Cignoni*



- piecewise-linear edges
- always interpolating colors on the same side of the edge
- with a custom interpolation scheme

Prior Art circa 2005

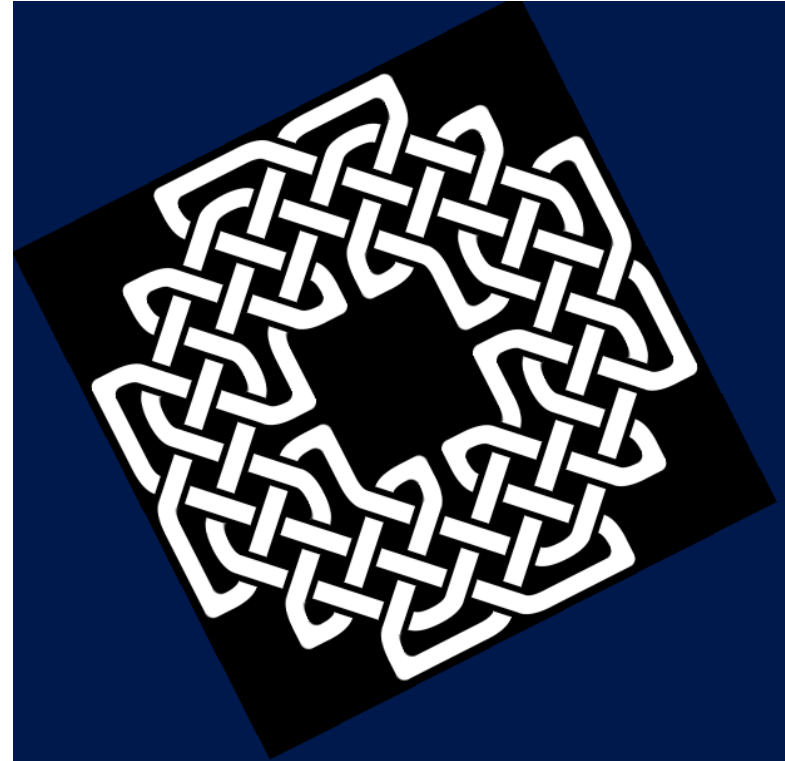
- Silmaps
Pradeep Sen
- **Bixels**
Jack Tumblin,
Prasun Choudhury
- Vector Texture Maps
Nicolas Ray et al
Curvilinear Contours
Stefan Gustavson
- Pinchmaps
Marco Tarini,
Paolo Cignoni



- decompose the texture plane into patches with straight boundary segments
- 10 patch functions

Prior Art circa 2005

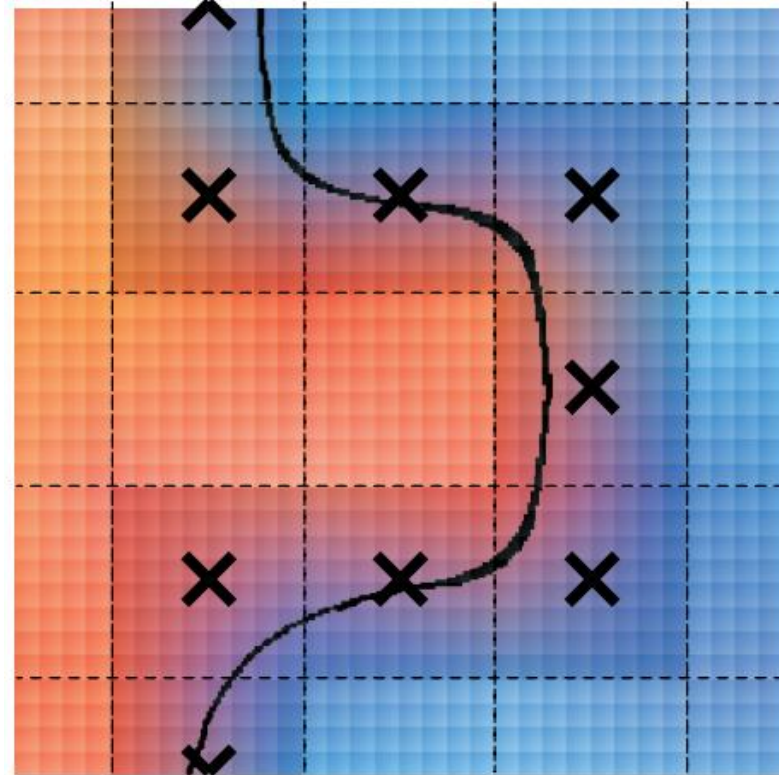
- Silmaps
Pradeep Sen
- Bixels
*Jack Tumblin,
Prasun Choudhury*
- **Vector Texture Maps**
Nicolas Ray et al
Curvilinear Contours
Stefan Gustavson
- Pinchmaps
*Marco Tarini,
Paolo Cignoni*



- implicit cubic polynomials for edges
- binary classification function defines a patch

Prior Art circa 2005

- Silmaps
Pradeep Sen
- Bixels
*Jack Tumblin,
Prasun Choudhury*
- Vector Texture Maps
Nicolas Ray et al
Curvilinear Contours
Stefan Gustavson
- **Pinchmaps**
**Marco Tarini,
Paolo Cignoni**



- a single quadratic silhouette edge per *pinchmap* texel
- use distance to the edge to compute new **uv**

pinchmaps



IRT



pinchmaps



IRT

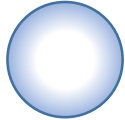

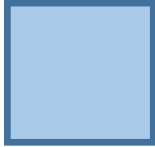
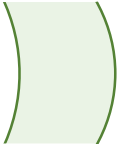


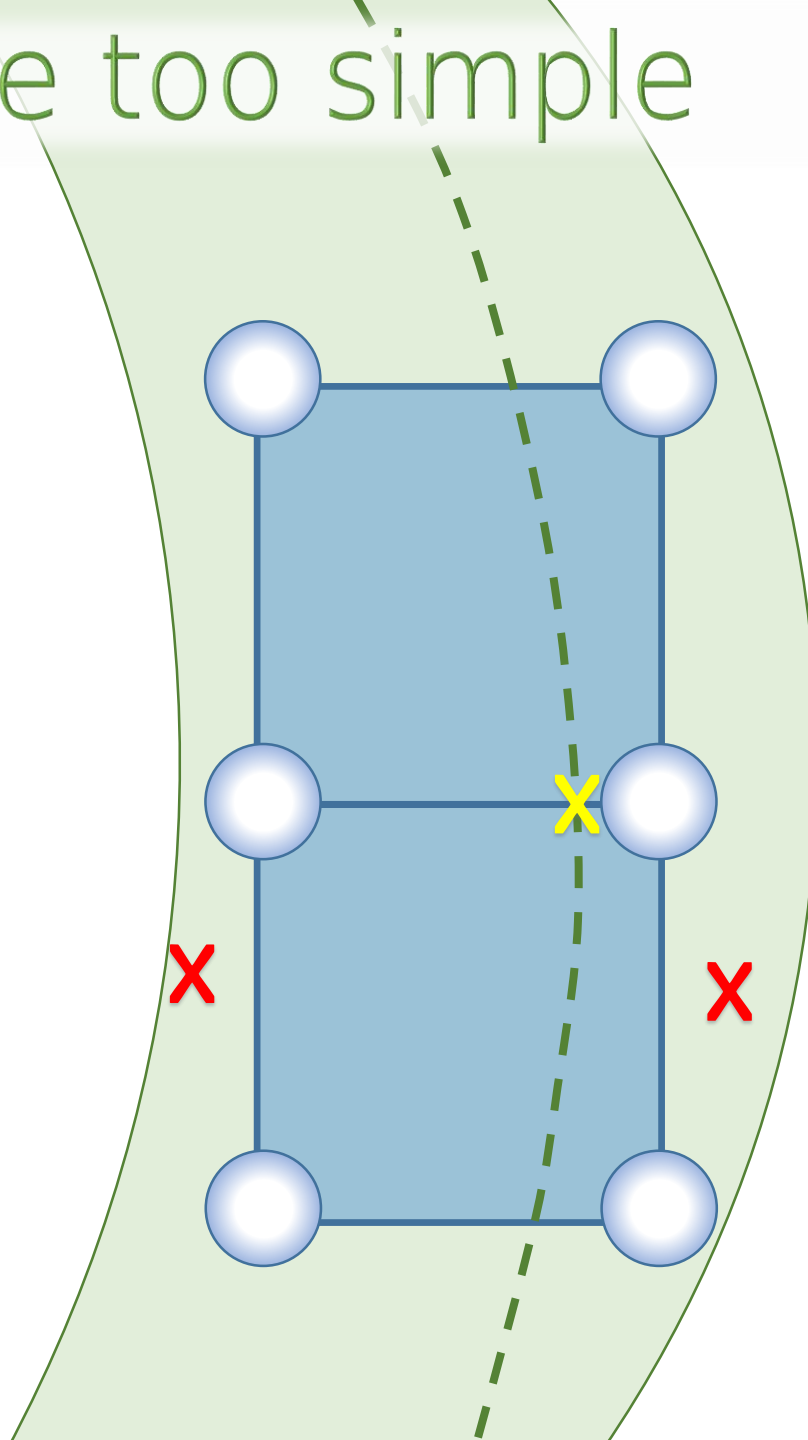
Occam vs Einstein

- **Occam's Razor**
the simpler one is usually better
- **Einstein Principle**
“a scientific theory should be as simple as possible, but no simpler”




Pinchmaps are too simple

- pinchmap texels 
- define an implicit quadratic curve, 
- so all samples that have 4 pinchmap texels... 
- ...will be resampled from the original texture 



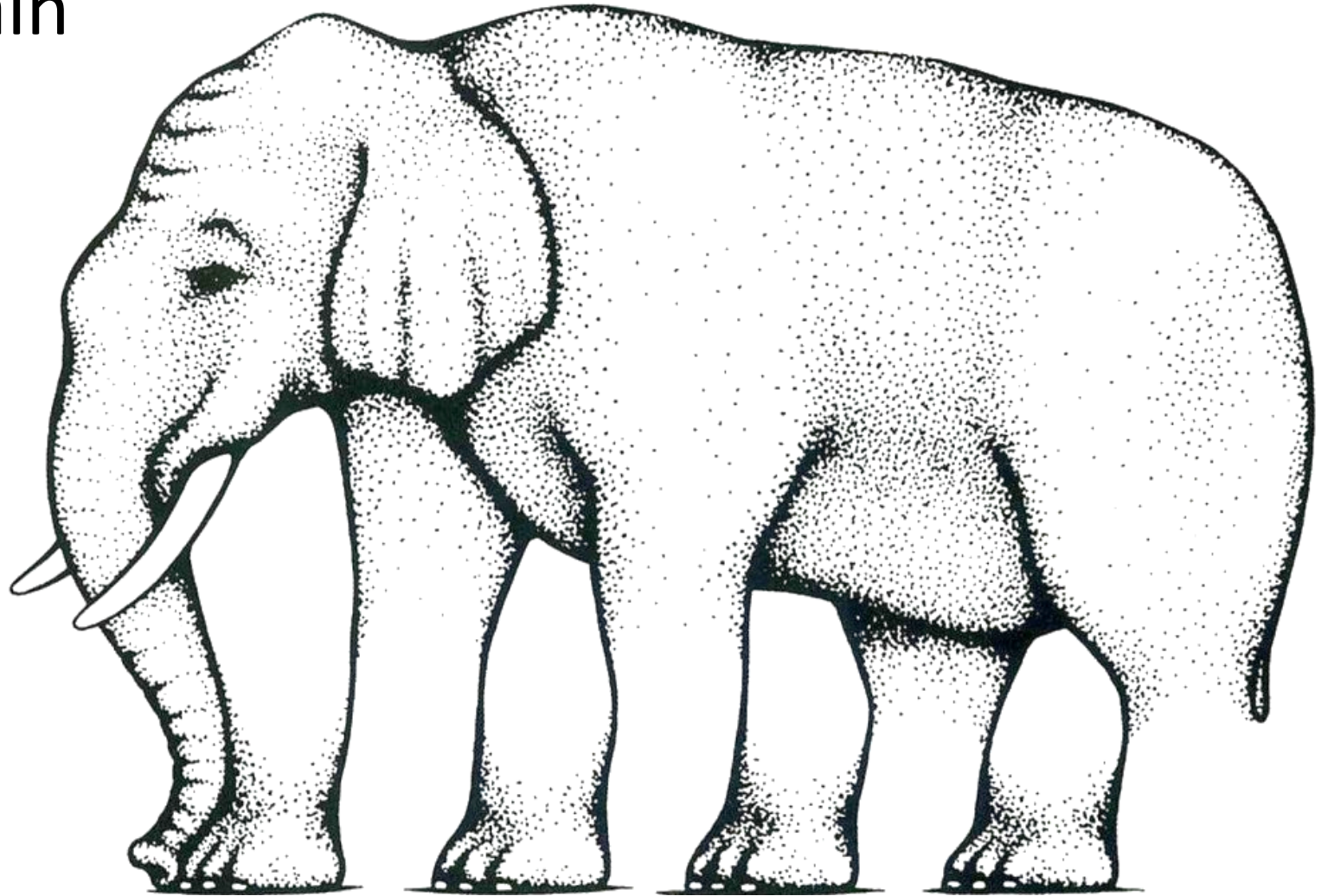
Issues

- No intersections 
- 2nd degree only
- zero adjustment for all 'outside' samples \Rightarrow discontinuous duv

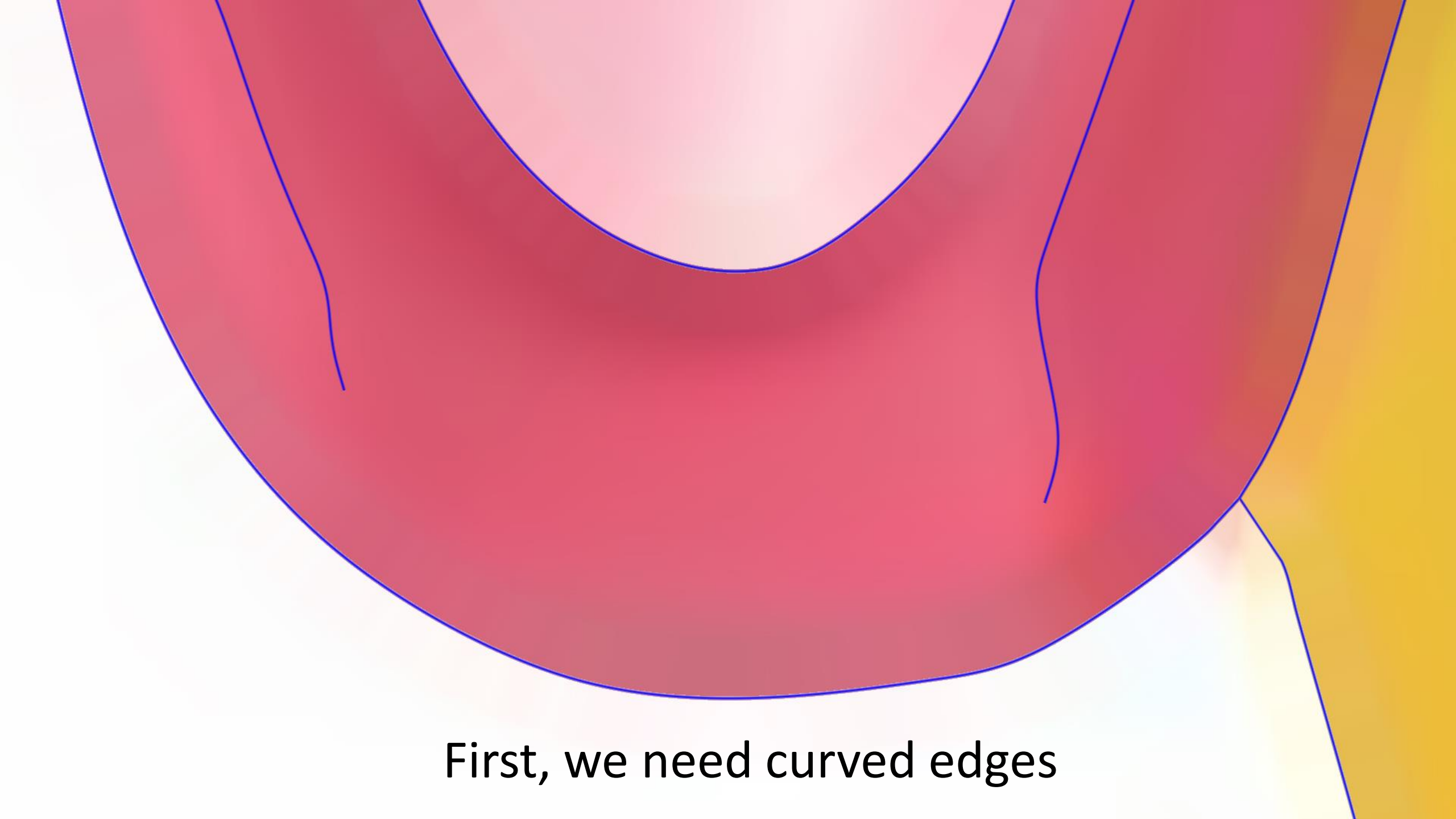
To address these problems...

...IRT uses more evolved processing...

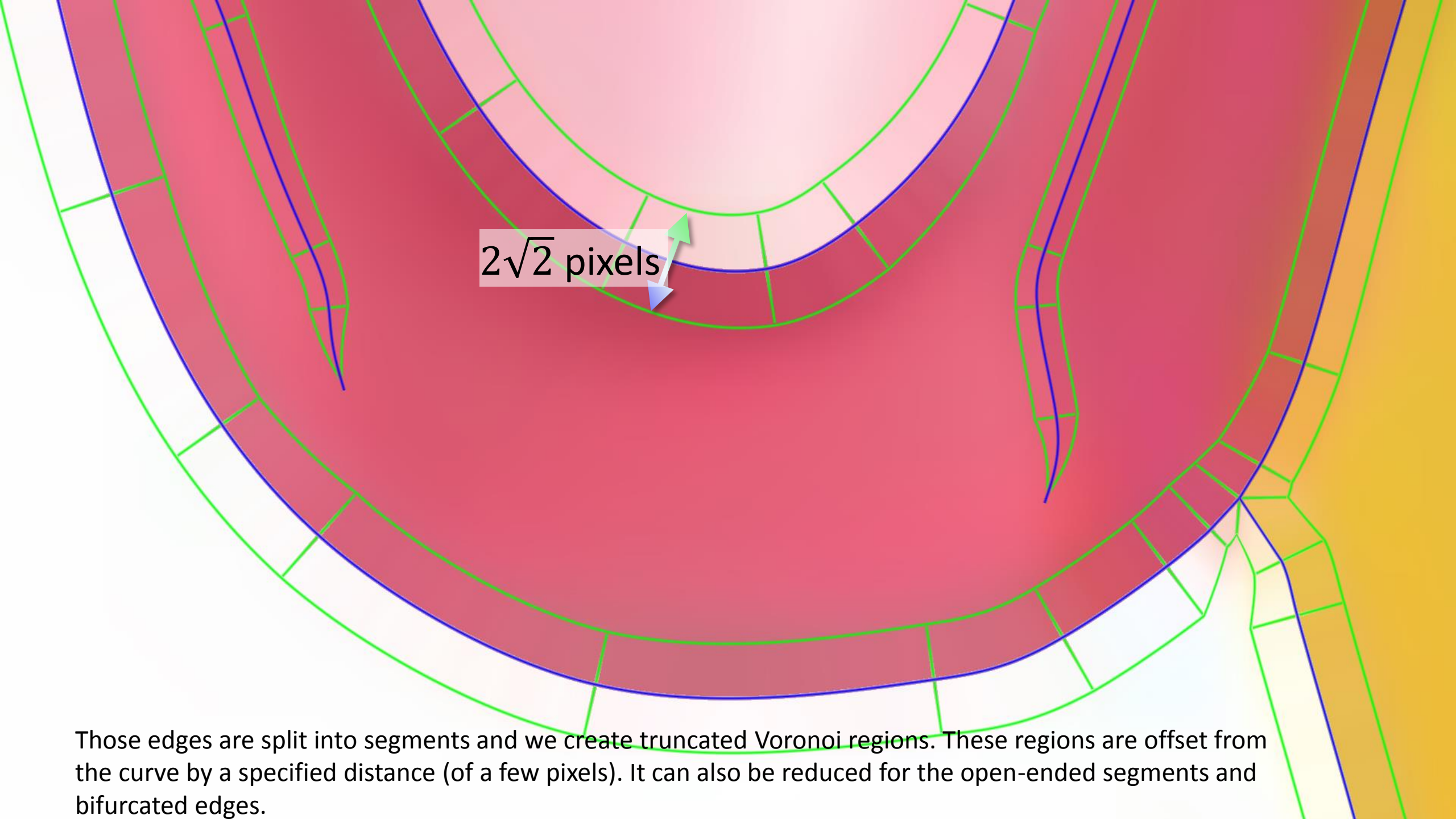
...that is easier to explain



legs-isential quandary
by Roger N. Shepard



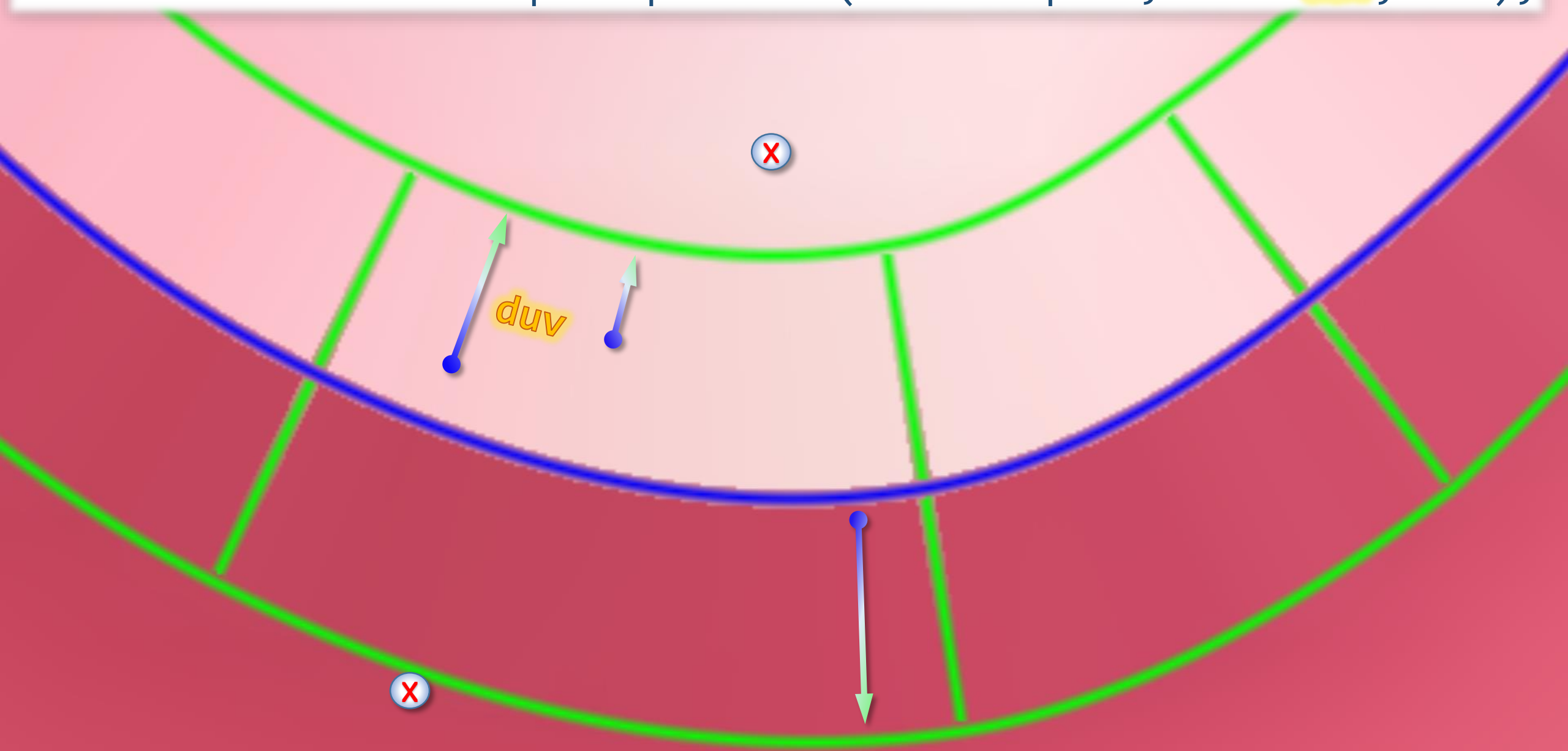
First, we need curved edges



$2\sqrt{2}$ pixels

Those edges are split into segments and we create truncated Voronoi regions. These regions are offset from the curve by a specified distance (of a few pixels). It can also be reduced for the open-ended segments and bifurcated edges.

```
float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
```



At run time, we just move the sample away from the edge. The samples outside Voronoi regions will have zero *duv* offset.

opportunities

1. Temper* raster and vector modes just by scaling the texture coordinate adjustment **duv** using pixel/texel ratio as

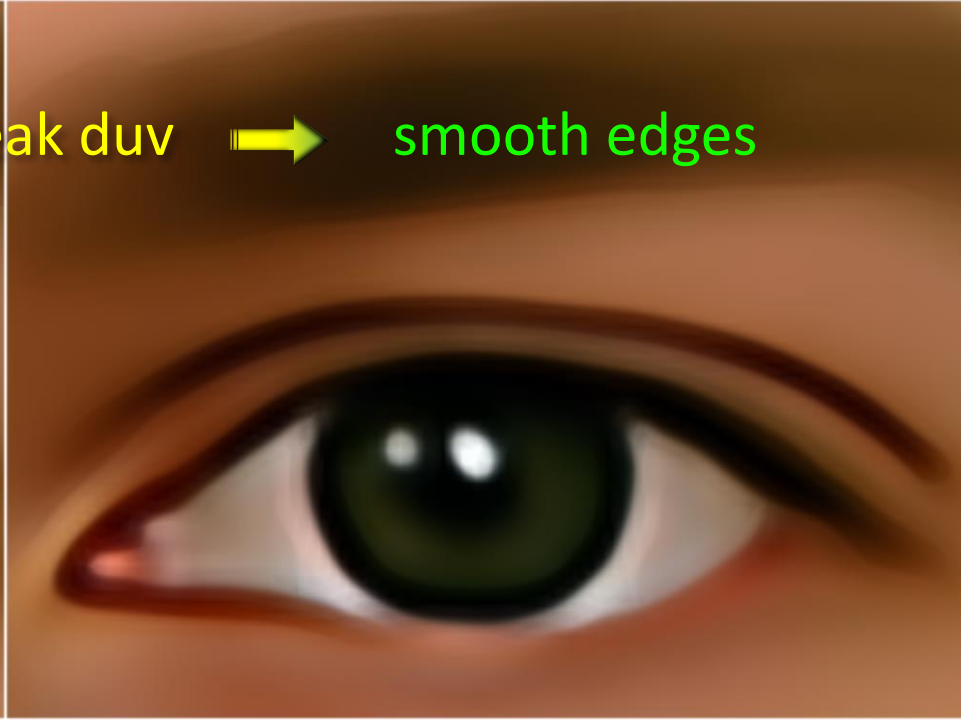
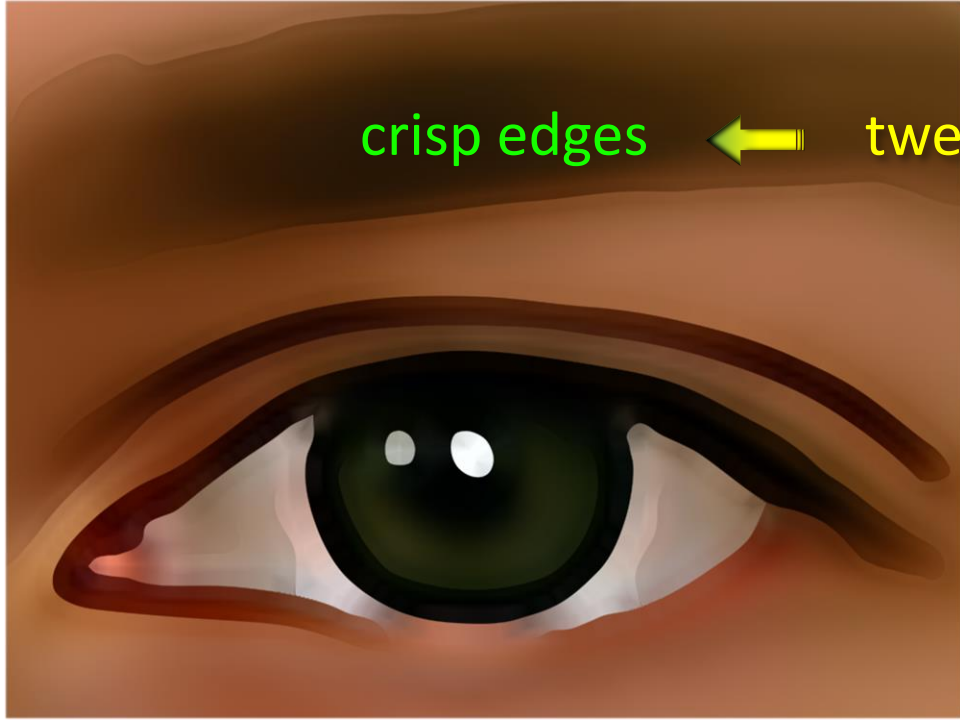
```
float pixratio = 0.5*length(fwidth(uv * texdim));  
duv *= min(1, 2 * (1 - pixratio)) / texdim;
```

2. Perform antialiasing in a single fetch by adjusting lod
3. Do whatever we like with it (like 'soft landing')

* having the elements mixed in satisfying proportions


<http://www.merriam-webster.com/dictionary/tempered>

soft
landing

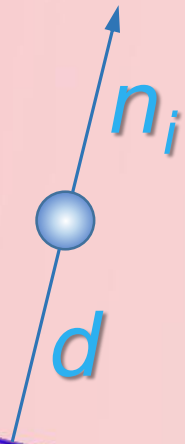


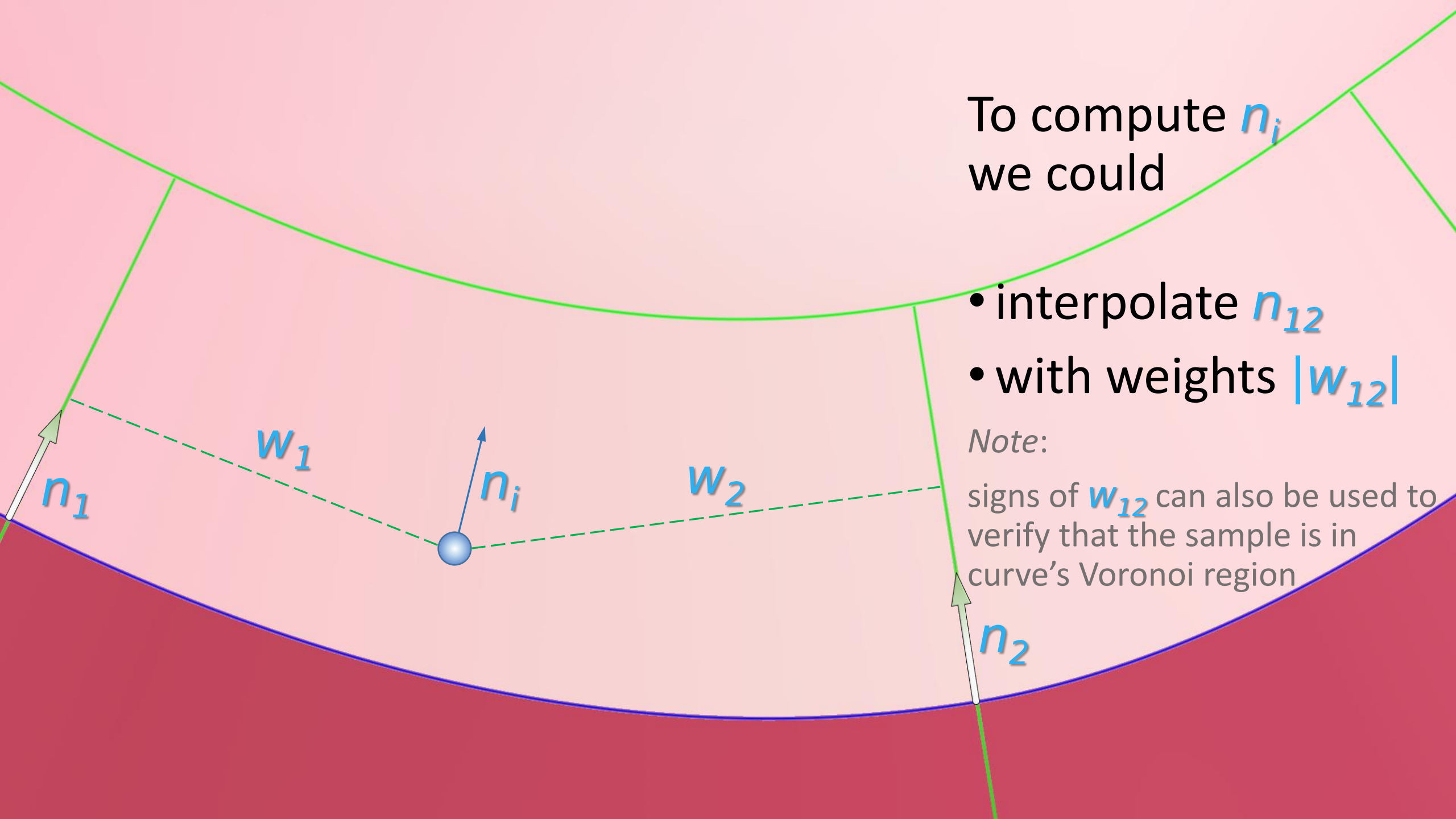
crisp edges ← tweak duv → smooth edges

details

For each sample , we need

- scalar distance to the curve d
- offset vector n_i





To compute n_i
we could

- interpolate n_{12}
- with weights $|w_{12}|$

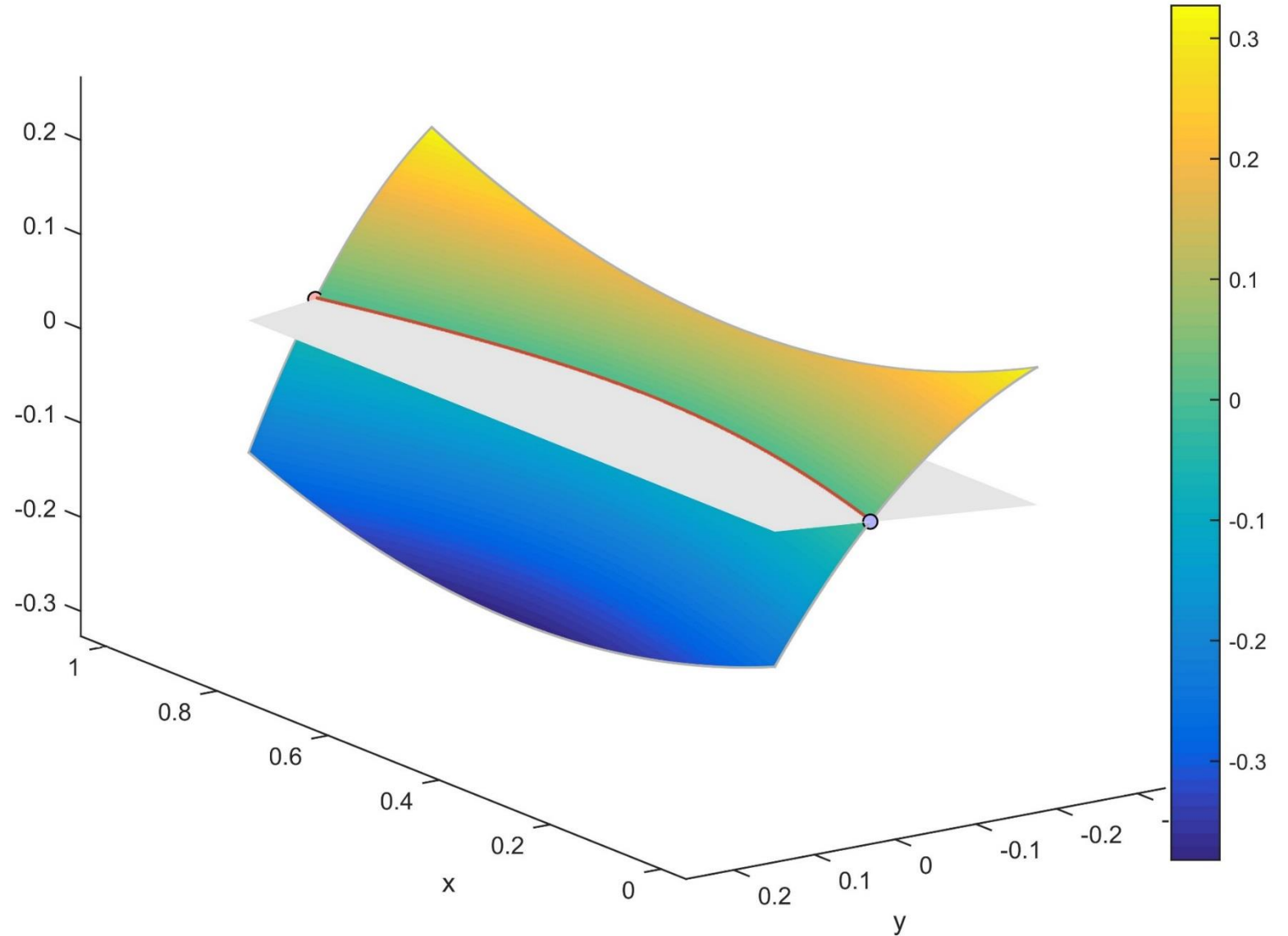
Note:

signs of w_{12} can also be used to verify that the sample is in curve's Voronoi region

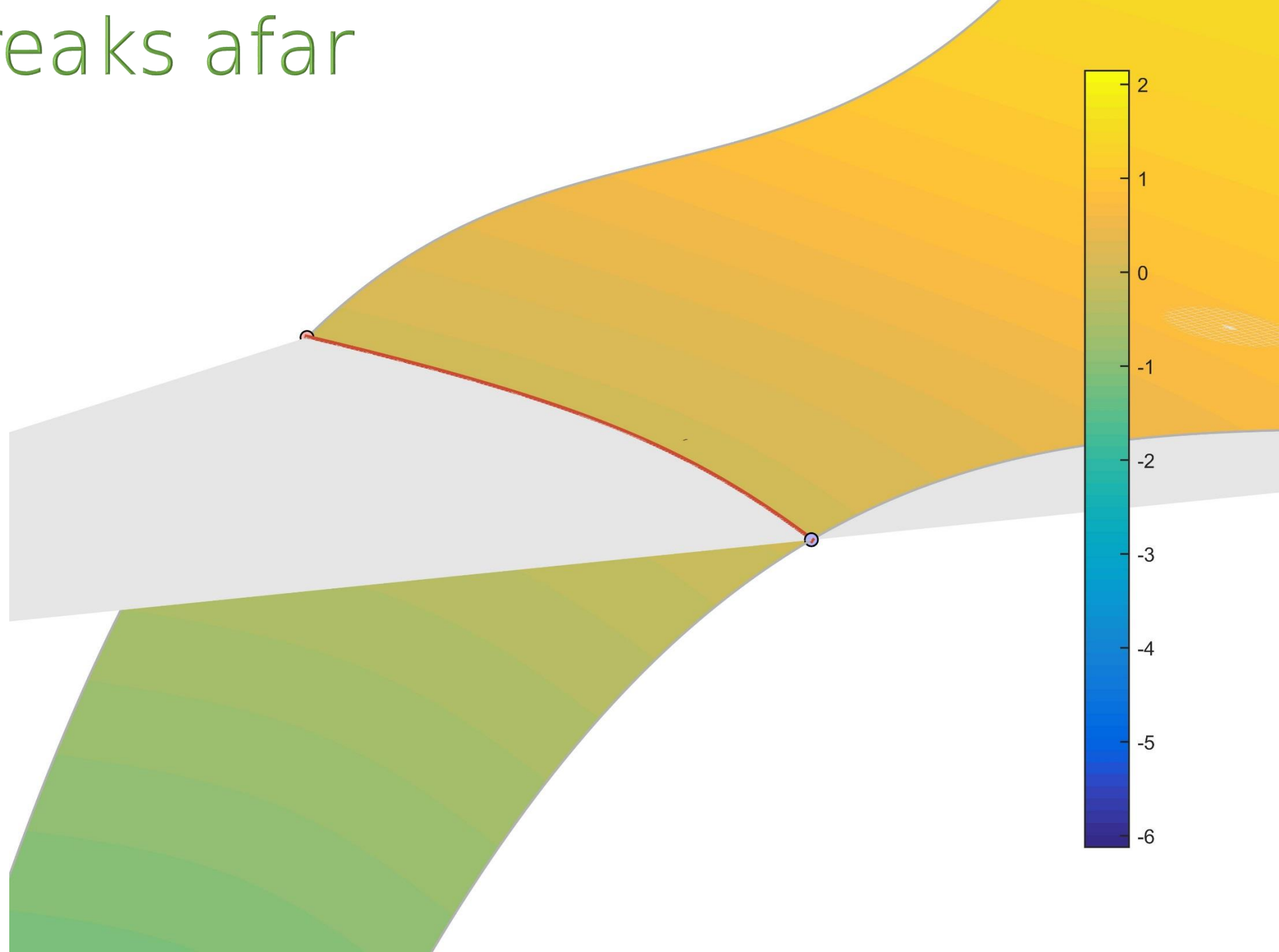
Distance to the curve

- \exists numerous prior art approaches
- To compute it even faster, we propose two algorithms:
 1. Implicit representation of cubic Bézier curves – using barycentric coordinates (savings: 6 terms instead of 10)
(see also “Rendering Cubic Curves on a GPU with Floater's Implicitization” by Ron Pfeifle in JGT 2012)
 2. A quotient of two multivariate polynomials over variables that we choose (to make life easier)
 \approx beefed up Phong interpolation in 1D

Using a value of implicit cubic polynomial as a distance proxy works near the curve...

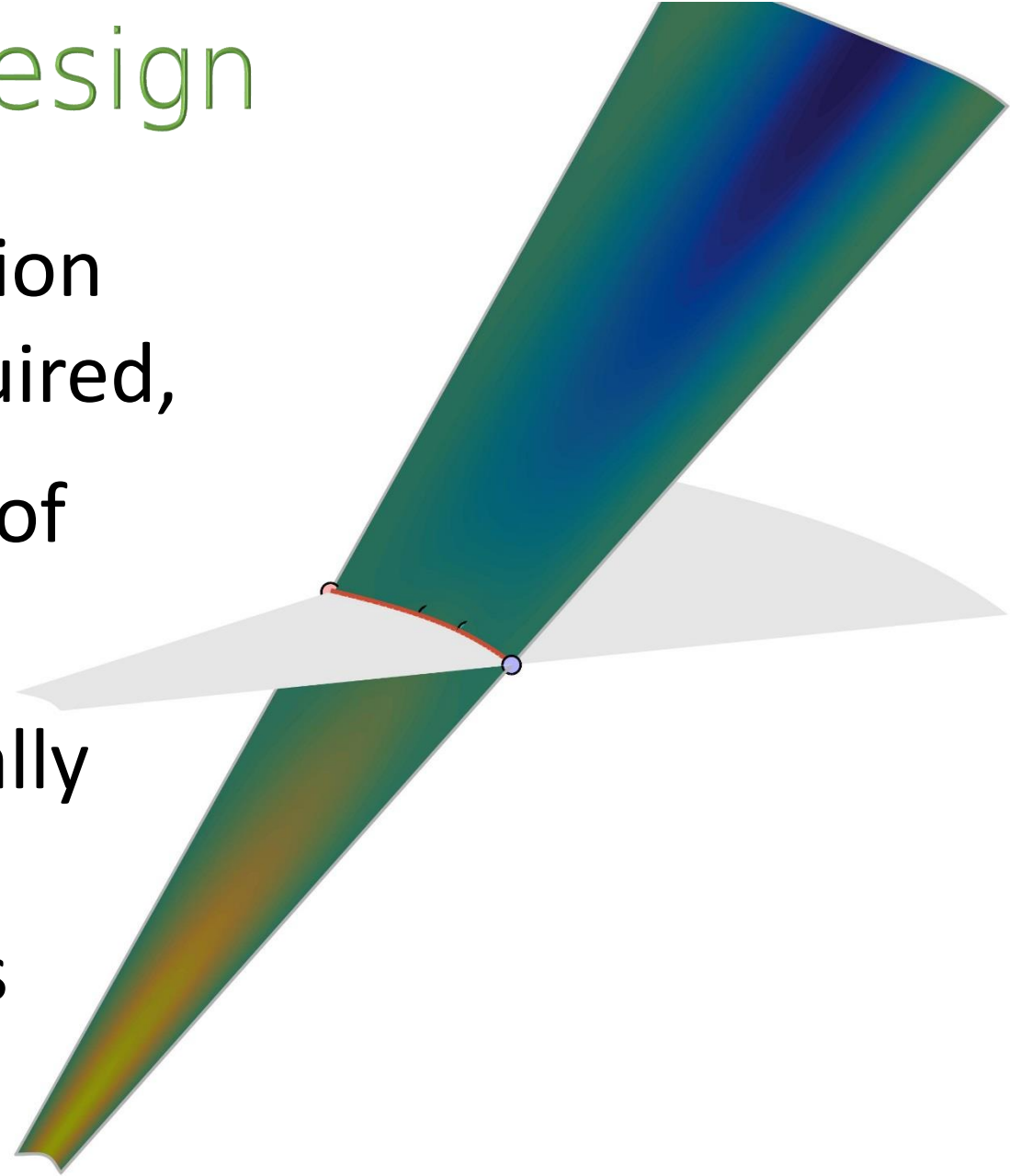


...but breaks afar



P_{n+1}/P_n is stable by design

- unless strict reproduction of Bézier curves is required,
- ✓ it should be a method of choice
- ✓ since it is unconditionally stable; there are other interesting possibilities as well



Plans

