DiFi: Fast 3D Distance Field Computation using Graphics Hardware

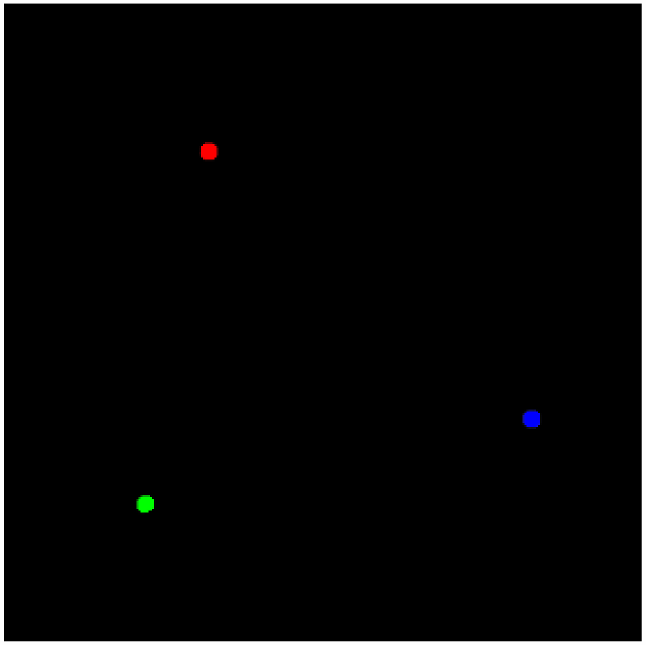
Avneesh Sud, Miguel A. Otaduy
and Dinesh Manocha

University of North Carolina at Chapel Hill

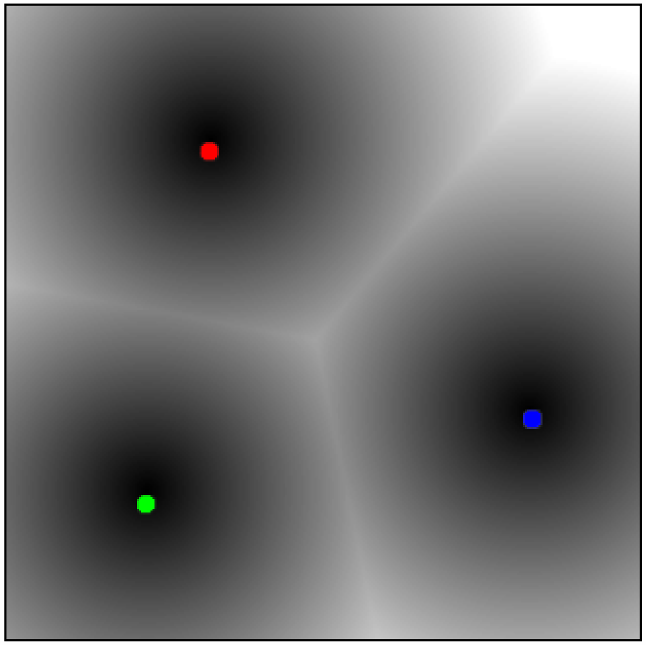
http://gamma.cs.unc.edu/DiFi

Distance Field

Given a set of geometric primitives (sites), it is a scalar field representing the minimum distance from any point to the closest site



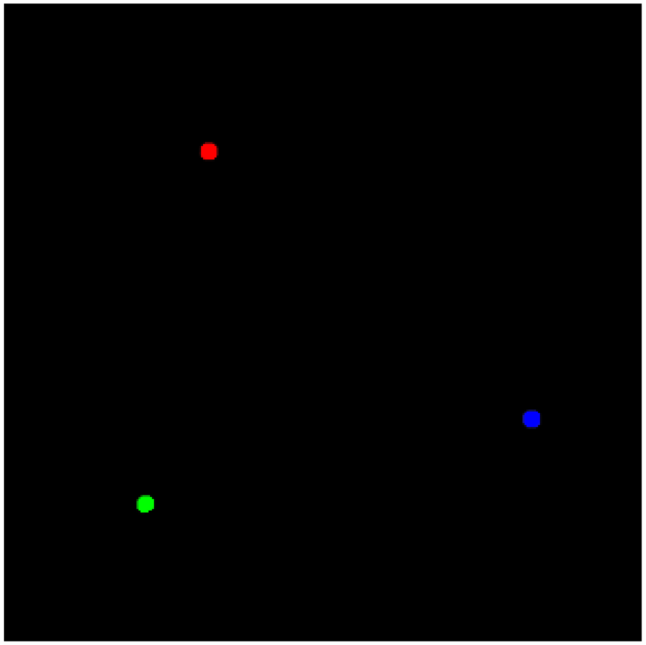
3 point sites



2D Distance field

Voronoi Diagram

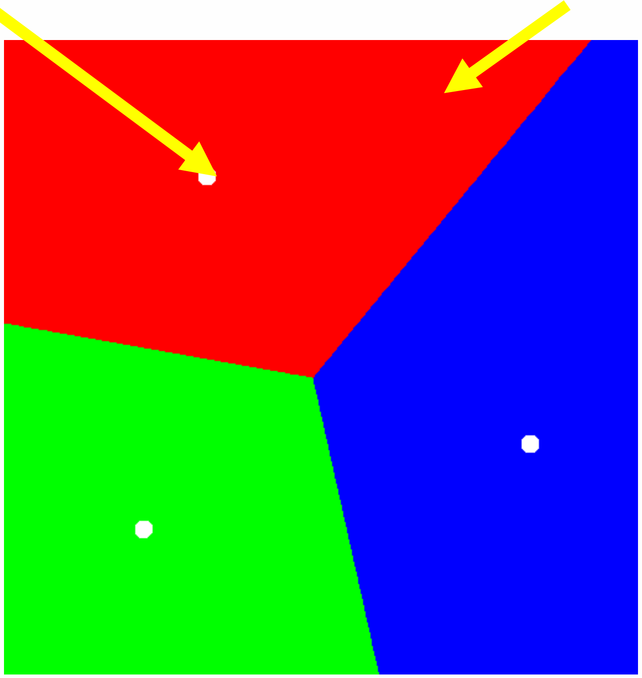
Given a collection of sites, it is a subdivision of space into cells such that all points in a cell are *closer* to one site than to any other site



3 point sites

Voronoi Site

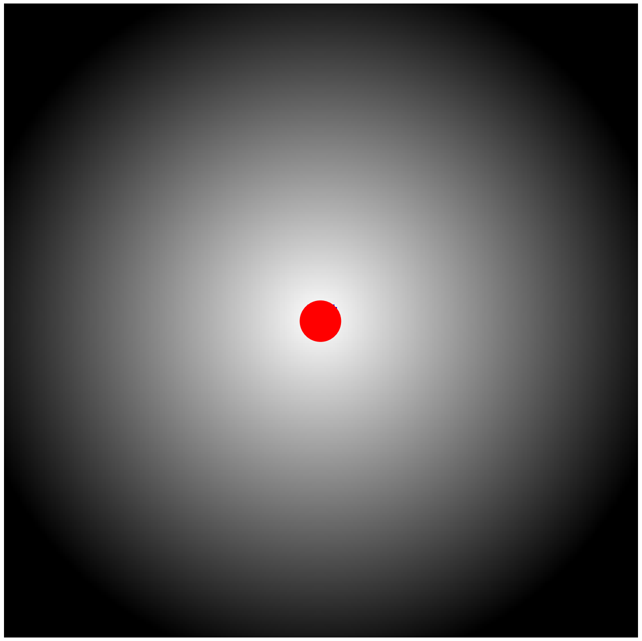
Voronoi Region



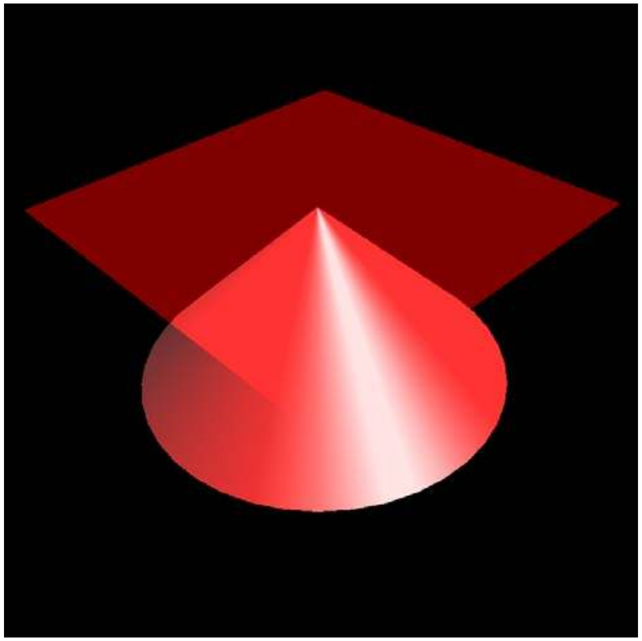
Voronoi diagram

Distance Function

A scalar function $f(\mathbf{x})$ representing minimum distance from a point **x** to a site



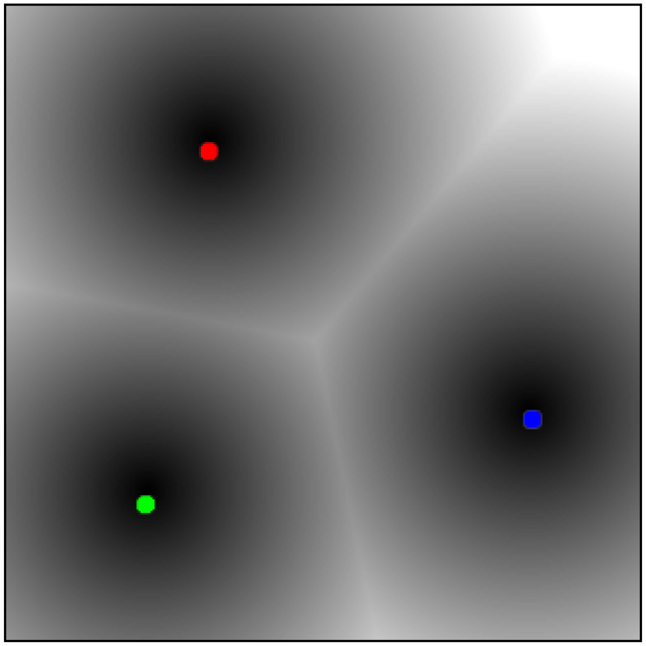
$f(x,y)=\sqrt{x^2+y^2}$

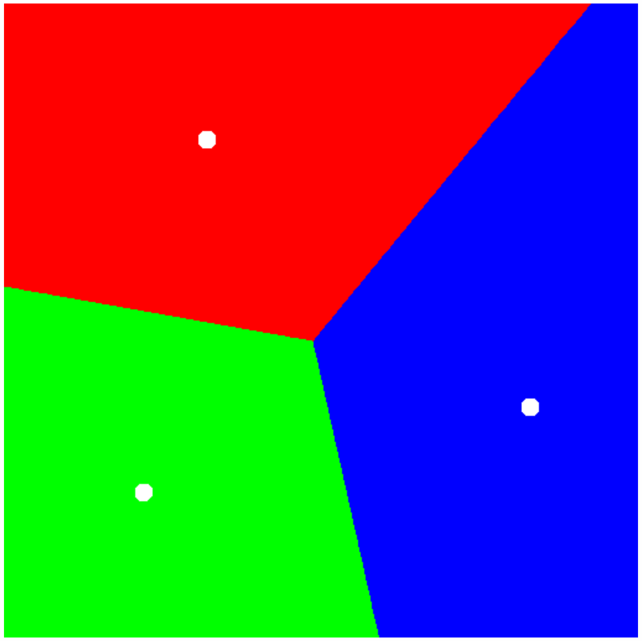


graph $z = f(x,y)$

Voronoi Diagram and Distance Fields

Region where distance function contributes to final distance field = Voronoi Region





Why Should We Compute Them?

Useful in a wide variety of applications

Collision Detection

Surface Reconstruction

Robot Motion Planning

Non-Photorealistic Rendering

Surface Simplification

Mesh Generation

Shape Analysis

Goal

Distance field algorithm:

- Fast computation
- Applicable to complex and generic models
- No preprocessing

Outline

- Related Work
- Fast GPU based algorithm (DiFi)
- Applications and Results
- Conclusions

Outline

- Related Work
- Fast GPU based algorithm (DiFi)
- Applications and Results
- Conclusions

Related Work

- Geometric models: Polygonal data
- Volumetric models: Image data

Related Work

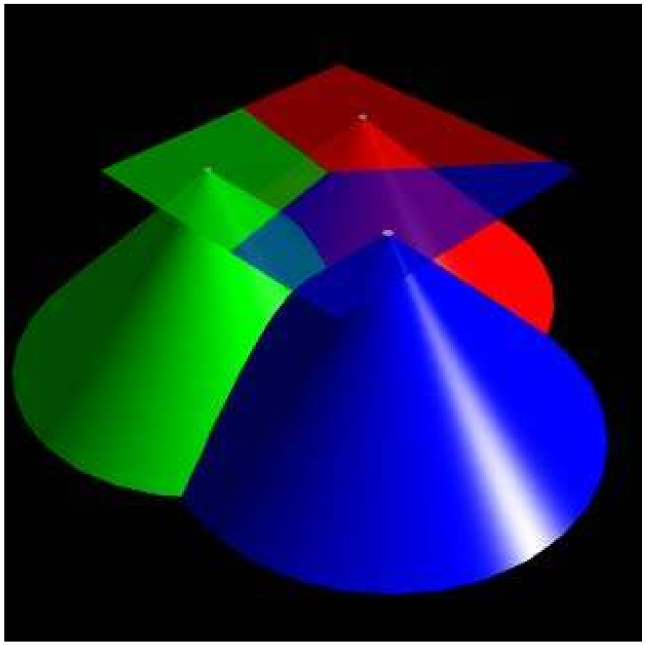
- Geometric models: Polygonal data
  - Adaptive Grids [Vleugels97, Frisken00]
  - Uniform Grids [Sethian96, Hoff99, Mauch00, Sigg03, Denny03, Furhmann03]
- Volumetric models: Image data

Related Work

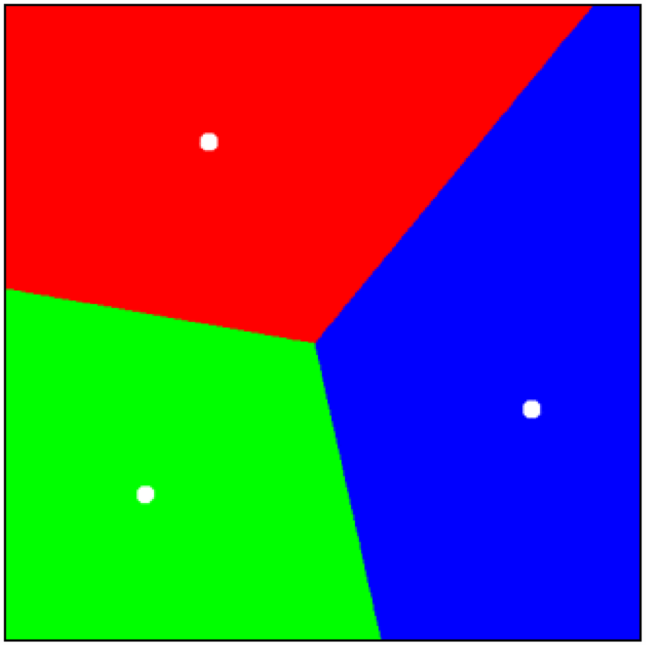
- Geometric models: Polygonal data
- Volumetric models: Image data
  - Approximate Distance Fields [Danielsson80, Sethian96]
  - Exact Distance Fields [Mulikin92, Breen00]
  - Surveyed in [Cuisenaire99]
  - Linear time algorithms for 2D [Breu95] and k-D [Maurer03]

GPU Based Computation

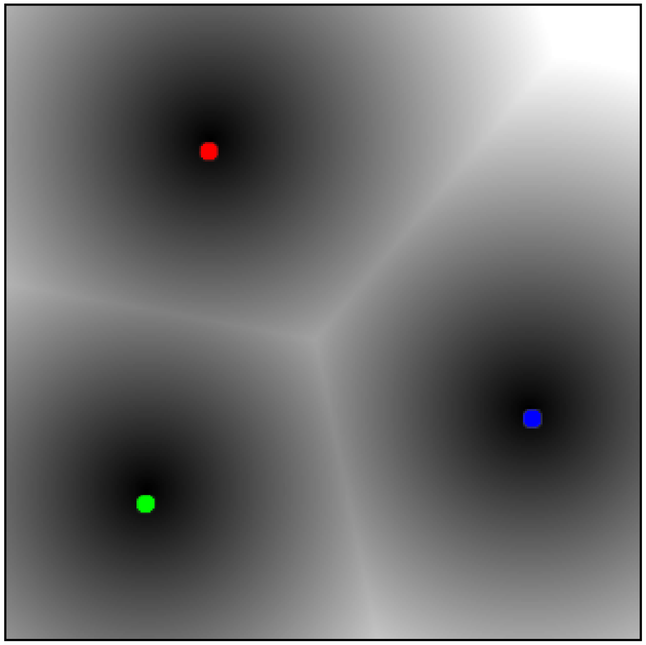
- Accelerate using graphics hardware [Hoff99]
  - Rasterization to compute distance values
  - Depth test to perform minimum operator



Render distance functions



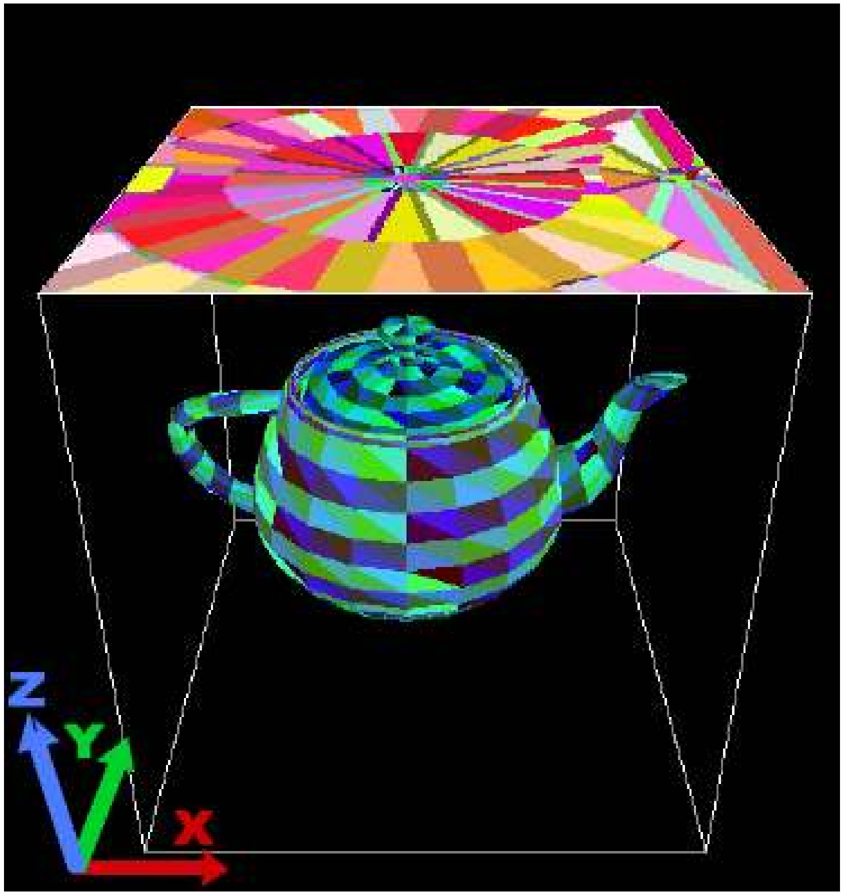
Color buffer



Depth buffer

GPU Based Computation

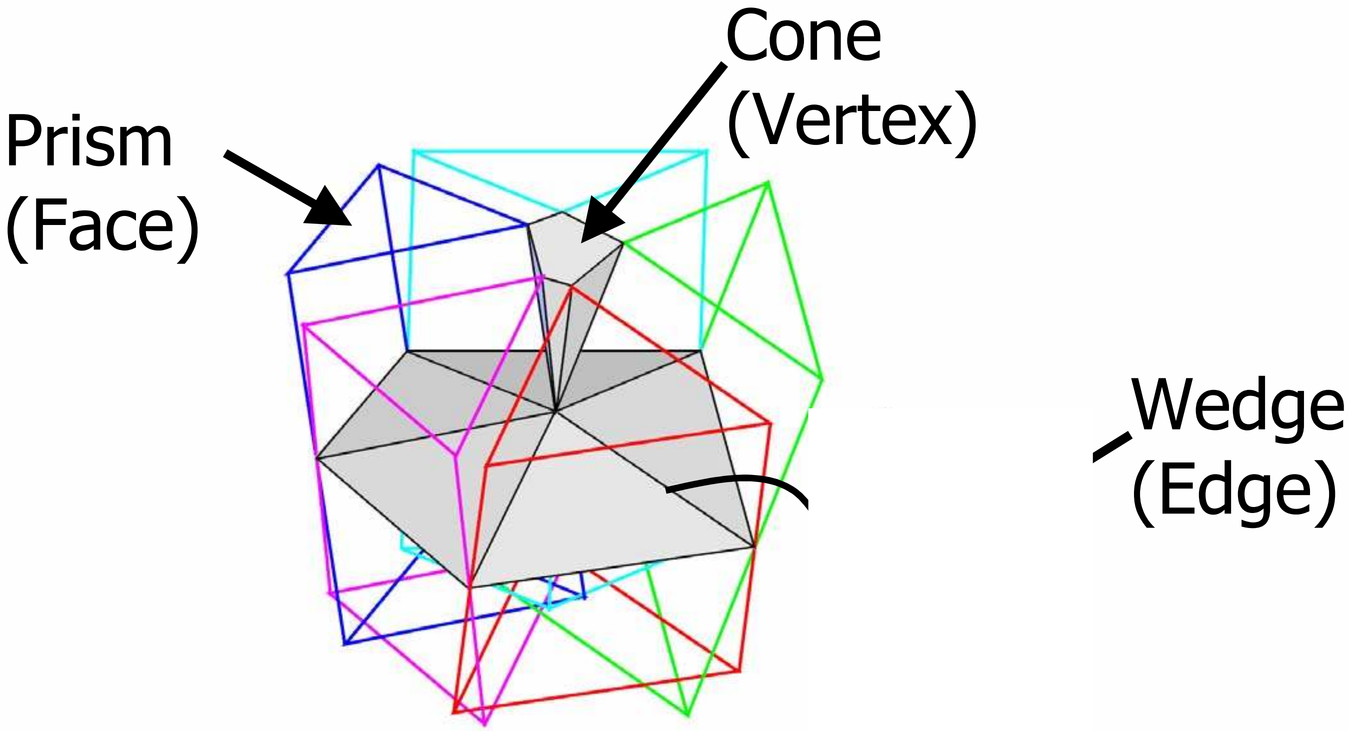
- Graphics hardware can generate one 2D slice at a time
- Sweep along 3rd dimension (Z-axis) computing 1 slice at a time
- Slow for large number of sites and high grid resolutions



3D Voronoi Diagram

GPU Based Computation

- For manifold objects, Voronoi regions bounded by prisms, wedges and cones [Mauch00, Sigg03]



GPU Based Computation

- Compute distance functions inside Voronoi region bounds using programmable GPU [Sigg03]
- Best suited for computation in small neighborhood of the boundary

- Not applicable to non-manifolds
- Inefficient for global computation

Contributions

- A fast 3D distance field computation algorithm
- Reduces computation using geometric properties and spatial coherence
  - Culling
  - Clamping
- Applicable to complex polygonal and image models
- No preprocessing

Outline

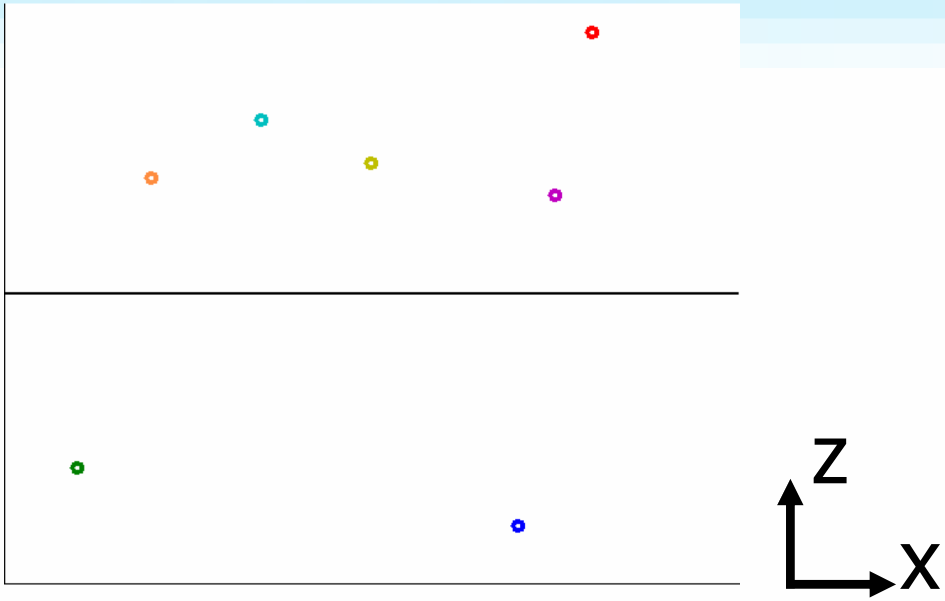
- Related Work
- Fast GPU based algorithm (DiFi)
  - Motivation
  - Geometric properties
  - Site classification
  - Culling algorithm
  - Clamping algorithm
- Applications and Results
- Conclusions

Outline

- Related Work
- Fast GPU based algorithm (DiFi)
  - Motivation
  - Geometric properties
  - Site classification
  - Culling algorithm
  - Clamping algorithm
- Applications and Results
- Conclusions

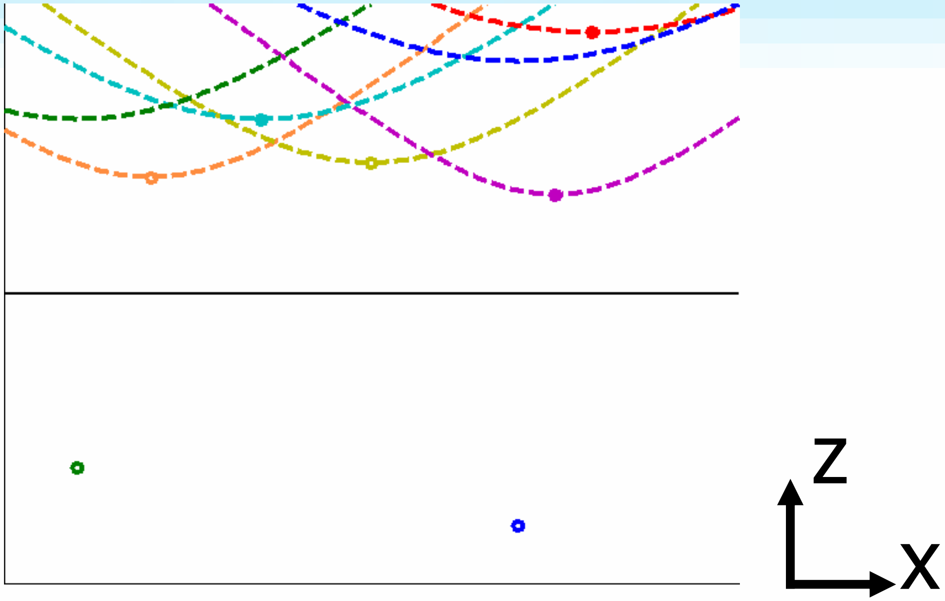
Motivation

- Not all sites contribute to distance field of a slice



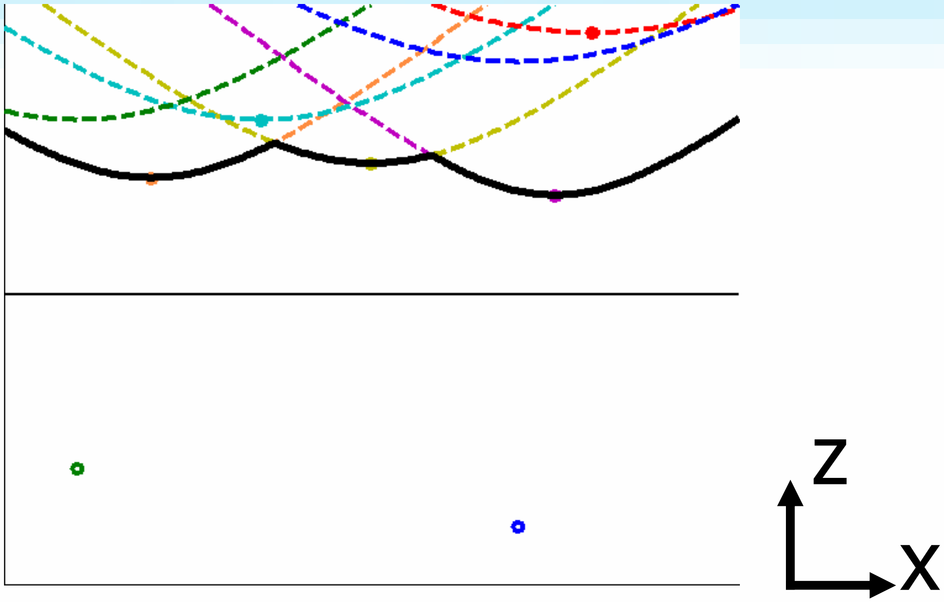
Motivation

- Not all sites contribute to distance field of a slice



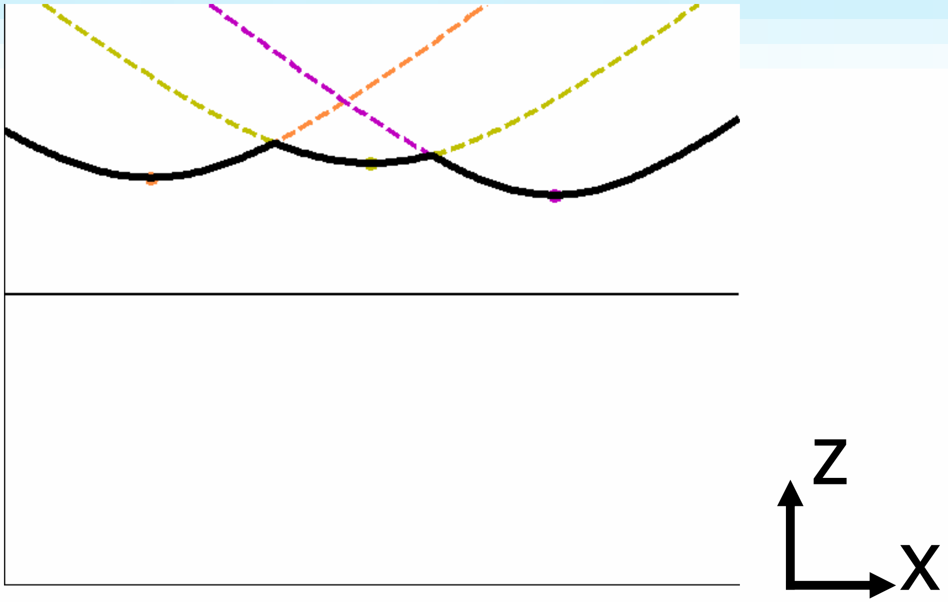
Motivation

- Not all sites contribute to distance field of a slice



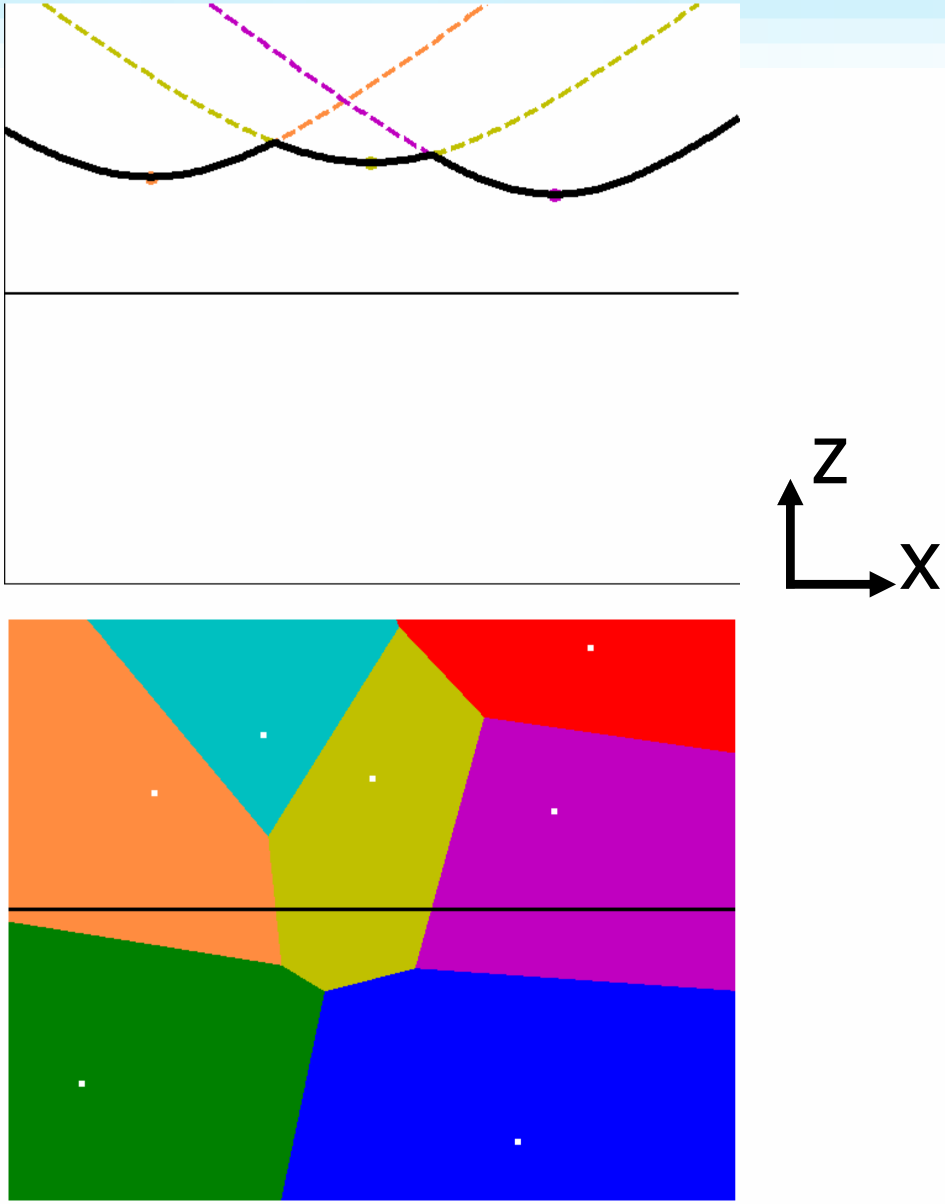
Motivation

- Not all sites contribute to distance field of a slice



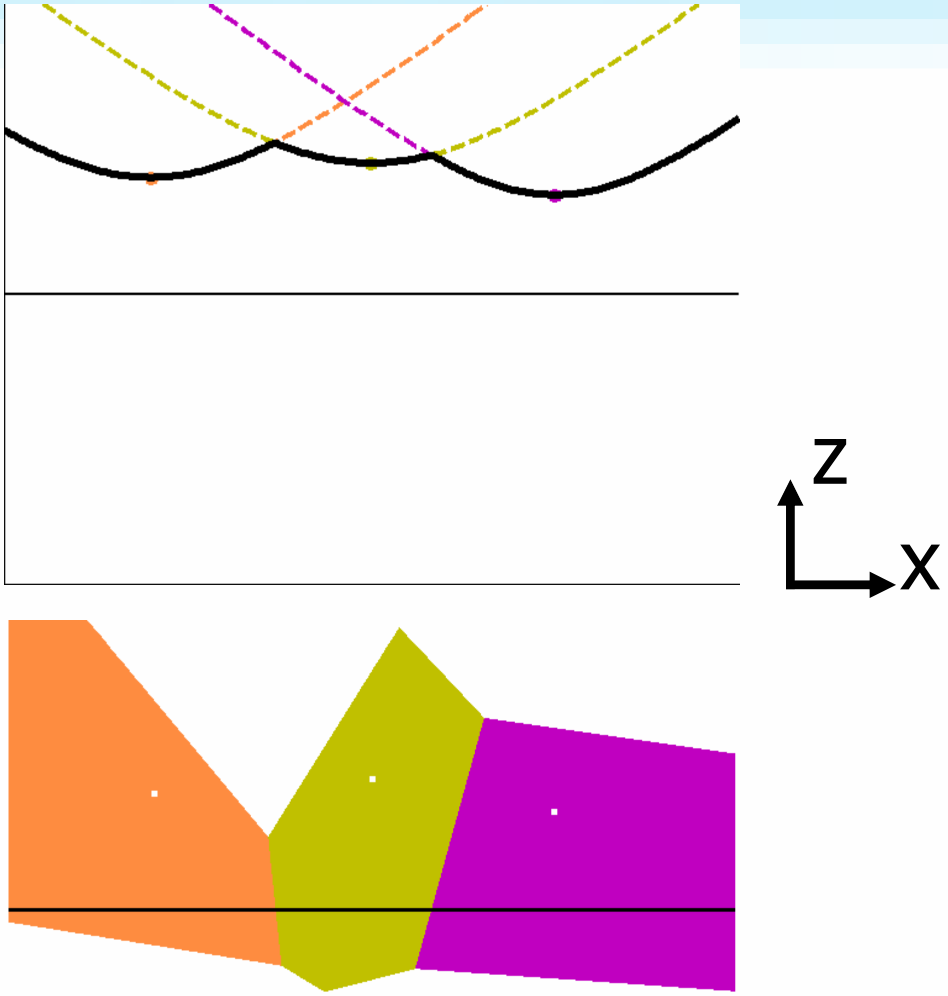
Motivation

- Not all sites contribute to distance field of a slice



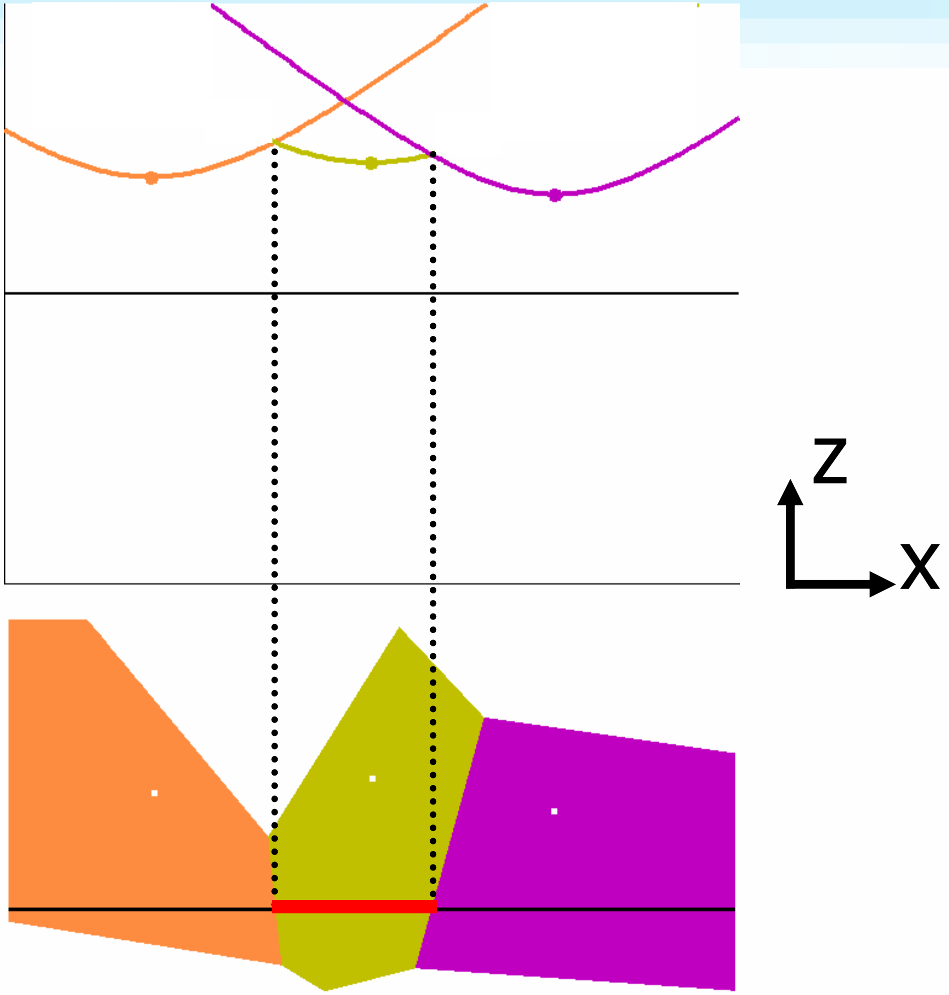
Motivation

- Sites whose Voronoi regions intersect the slice contribute to distance field
  - Small number of sites contribute
  - *Cull* remaining sites



Motivation: Goals

- Sites whose Voronoi regions intersect the slice contribute to distance field
  - *Cull* remaining sites
- Compute distance function in domain where Voronoi region intersects slice
  - *Clamp* domain of computation

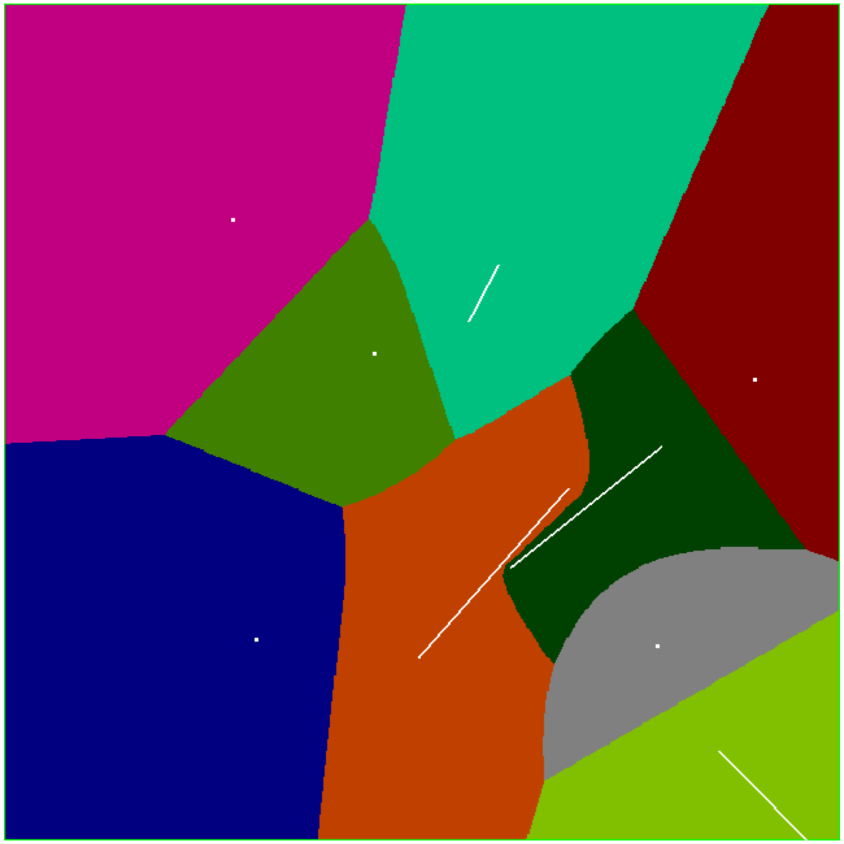


Outline

- Related Work
- Fast GPU based algorithm (DiFi)
  - Motivation
  - Geometric properties
  - Site classification
  - Culling algorithm
  - Clamping algorithm
- Applications and Results
- Conclusions

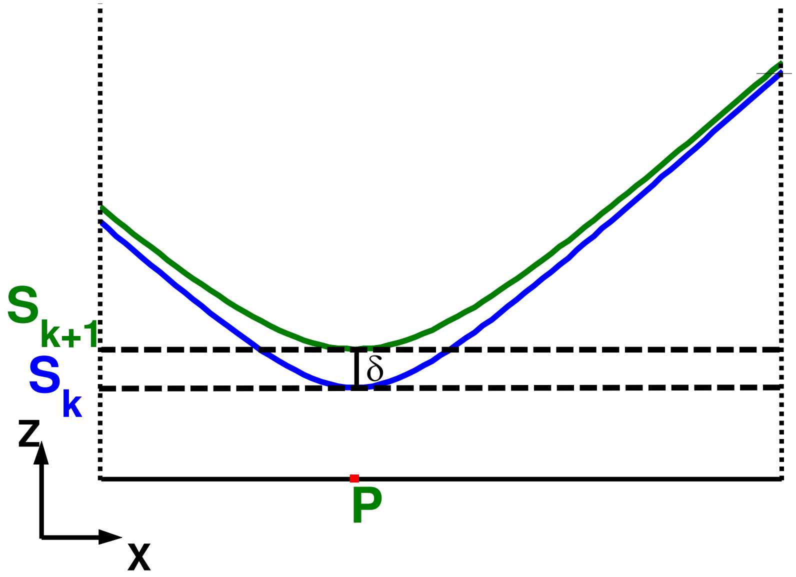
Geometric Properties

- *Connectivity:* Voronoi regions are connected for all L_p norms
  - Used for culling



Geometric Properties

- *Connectivity:* Voronoi regions are connected for all L_p norms
  - Used for culling
- *Coherence:* Change in distance field between adjacent slices is bounded
  - Used for clamping



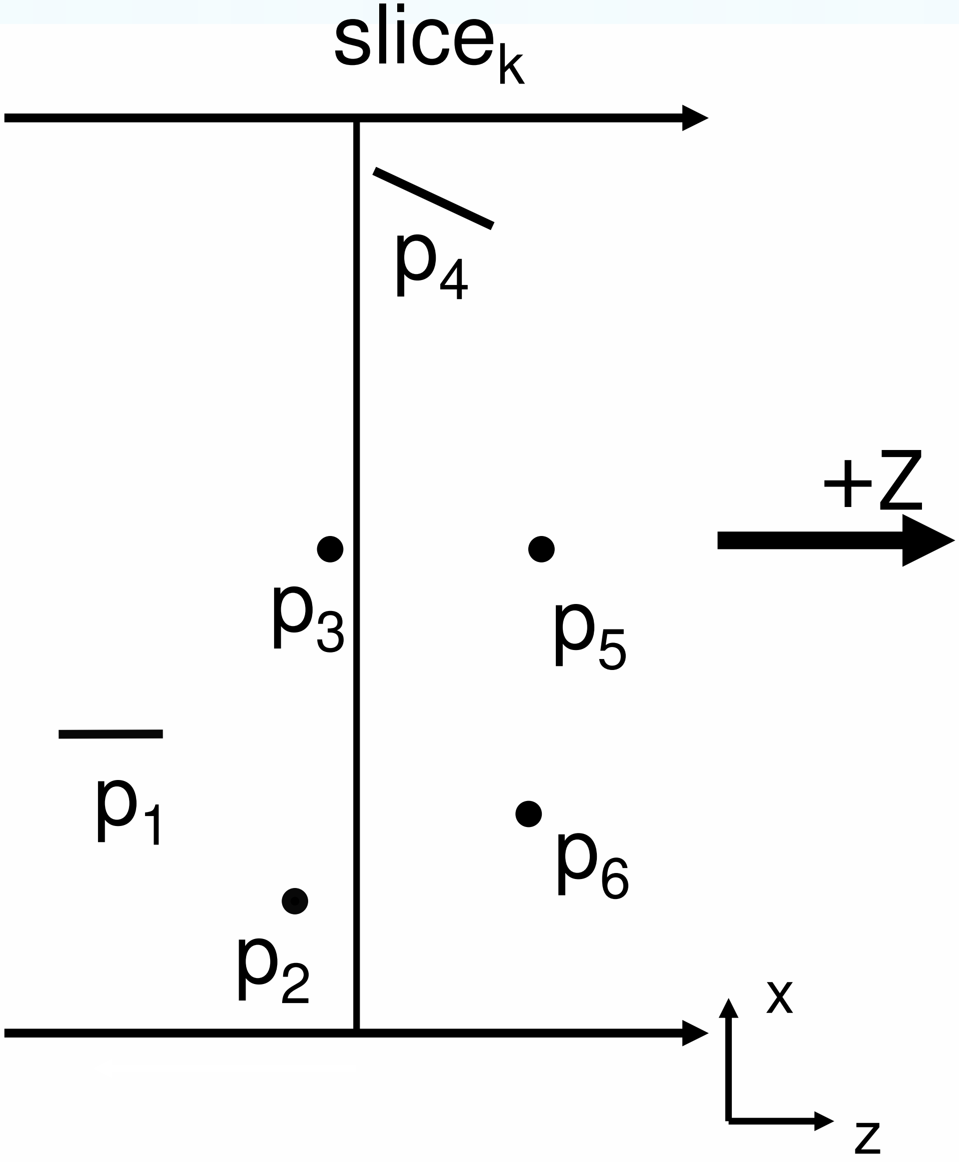
Distance functions for a point site P_i to adjacent slices

Outline

- Related Work
- Fast GPU based algorithm (DiFi)
  - Motivation
  - Geometric properties
  - Site classification
  - Culling algorithm
  - Clamping algorithm
- Applications and Results
- Conclusions

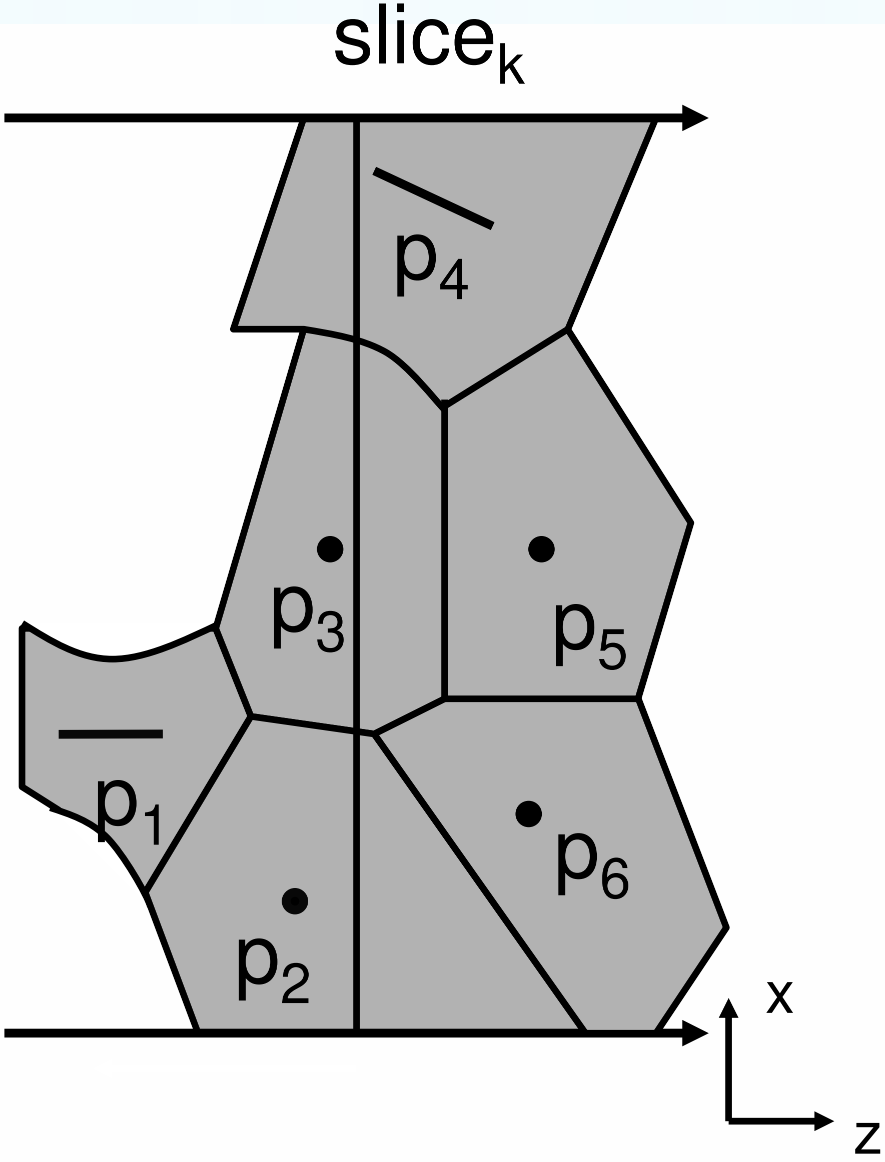
Site Classification

- For each slice partition the set of sites



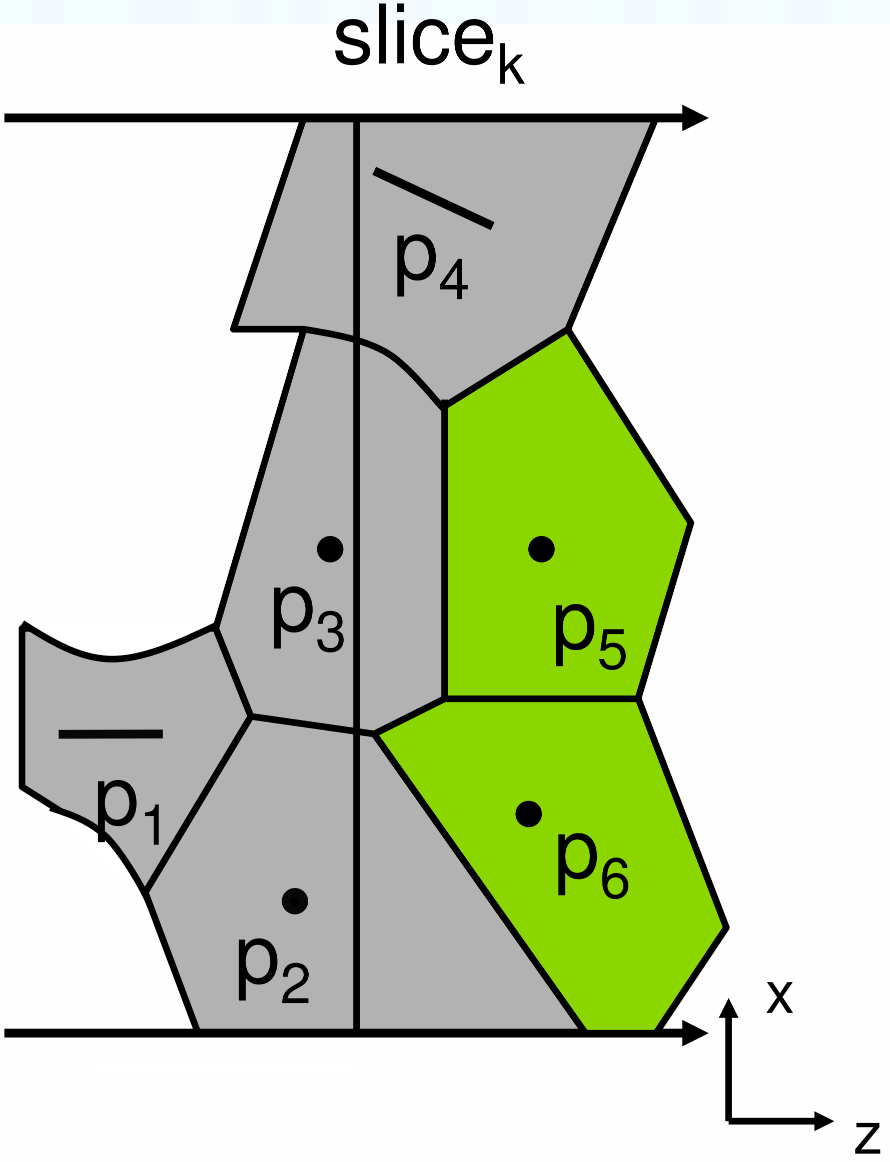
Site Classification

- For each slice partition the set of sites using Voronoi region bounds:



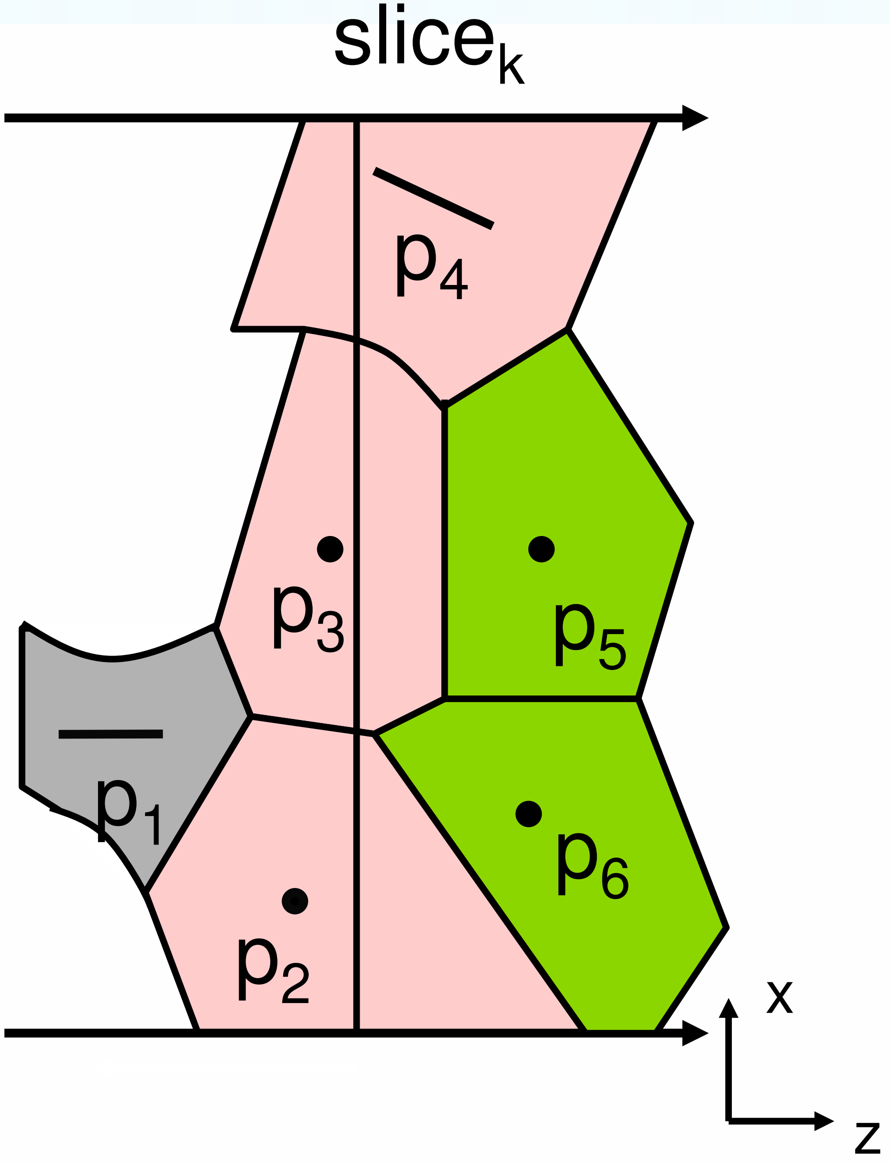
Site Classification

- For each slice partition the set of sites using Voronoi region bounds:
  - Approaching



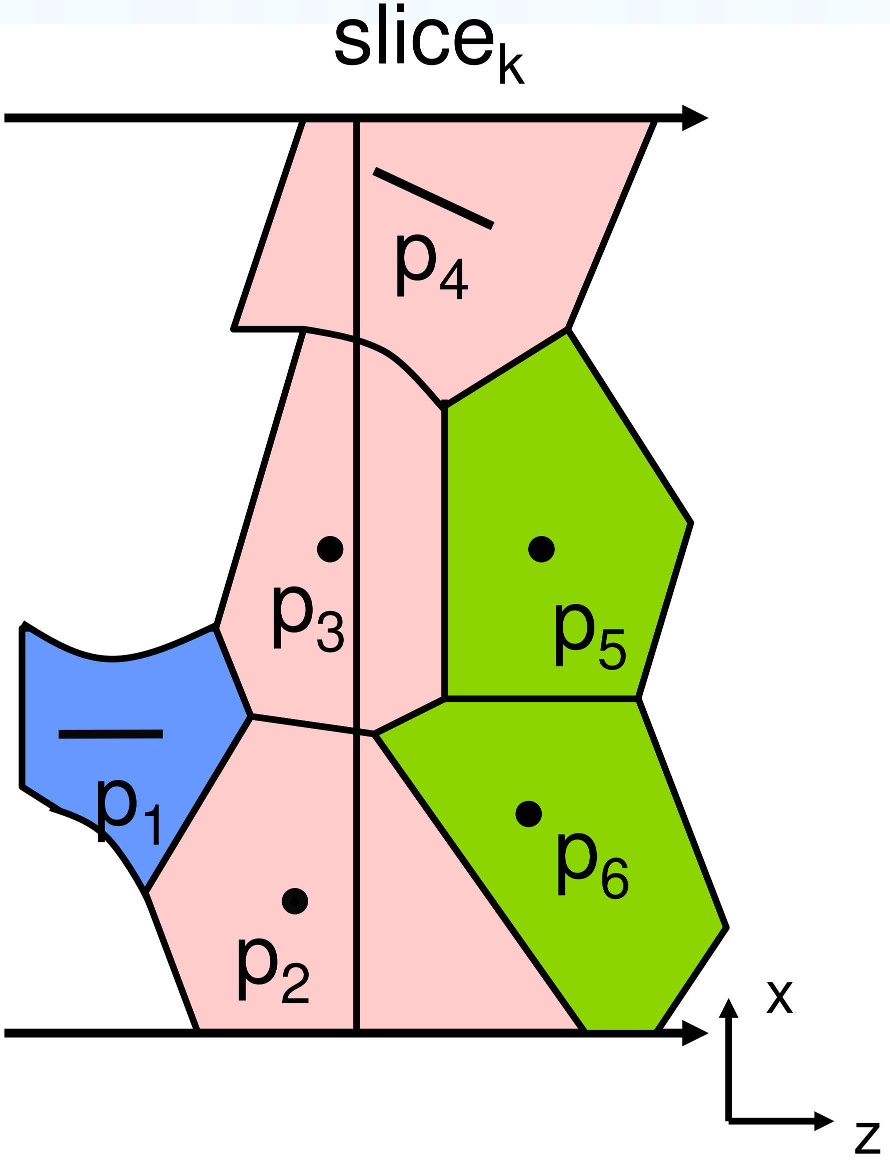
Site Classification

- For each slice partition the set of sites using Voronoi region bounds:
  - Approaching
  - Intersecting



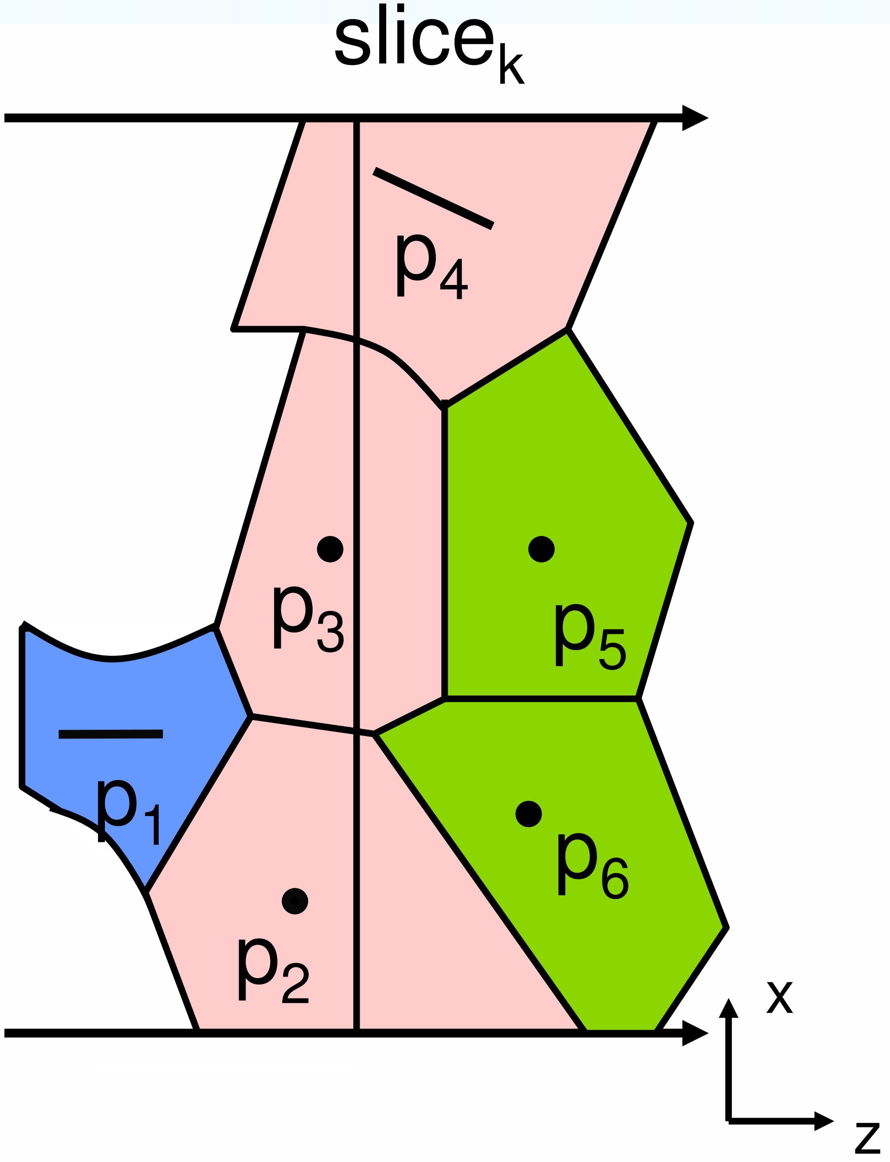
Site Classification

- For each slice partition the set of sites using Voronoi region bounds:
  - Approaching
  - Intersecting
  - Receding



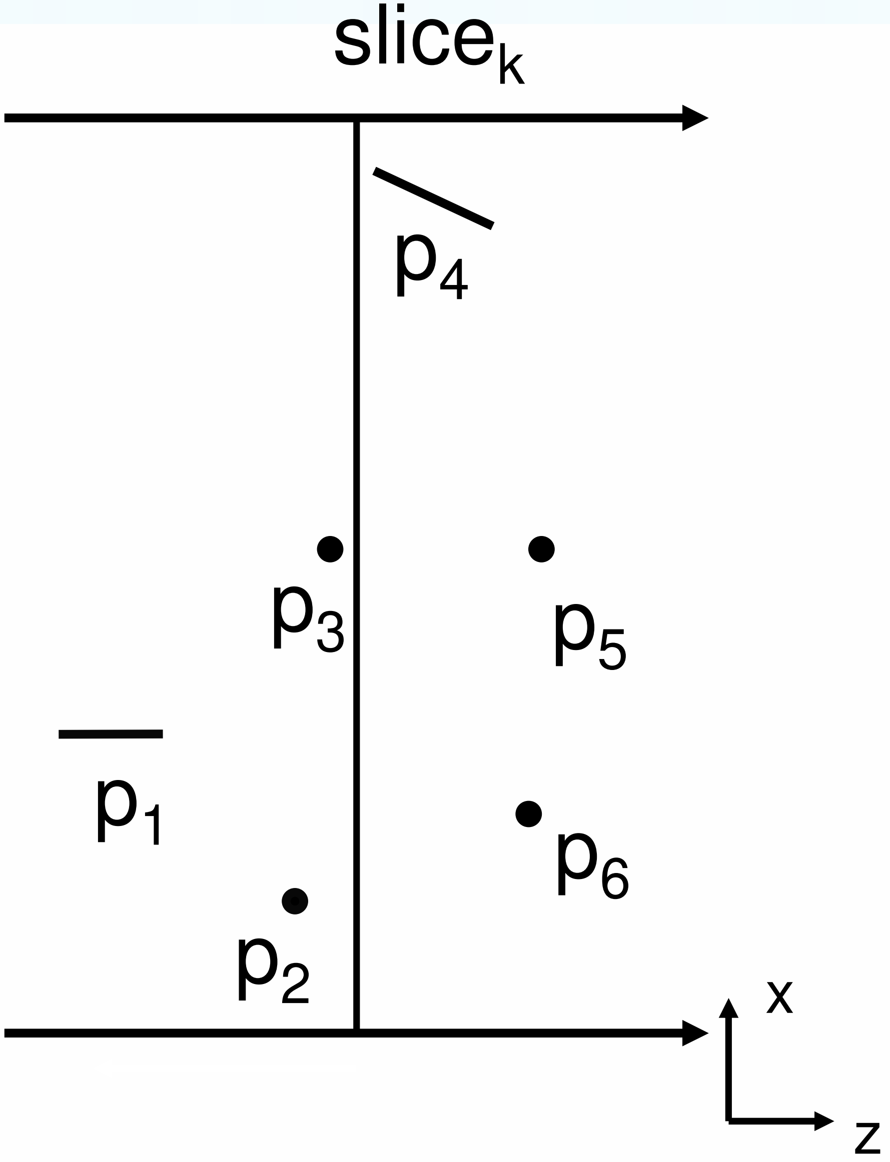
Site Classification

- For each slice partition the set of sites using Voronoi region bounds:
  - Approaching
  - Intersecting
  - Receding
- Only Intersecting sites contribute to distance field



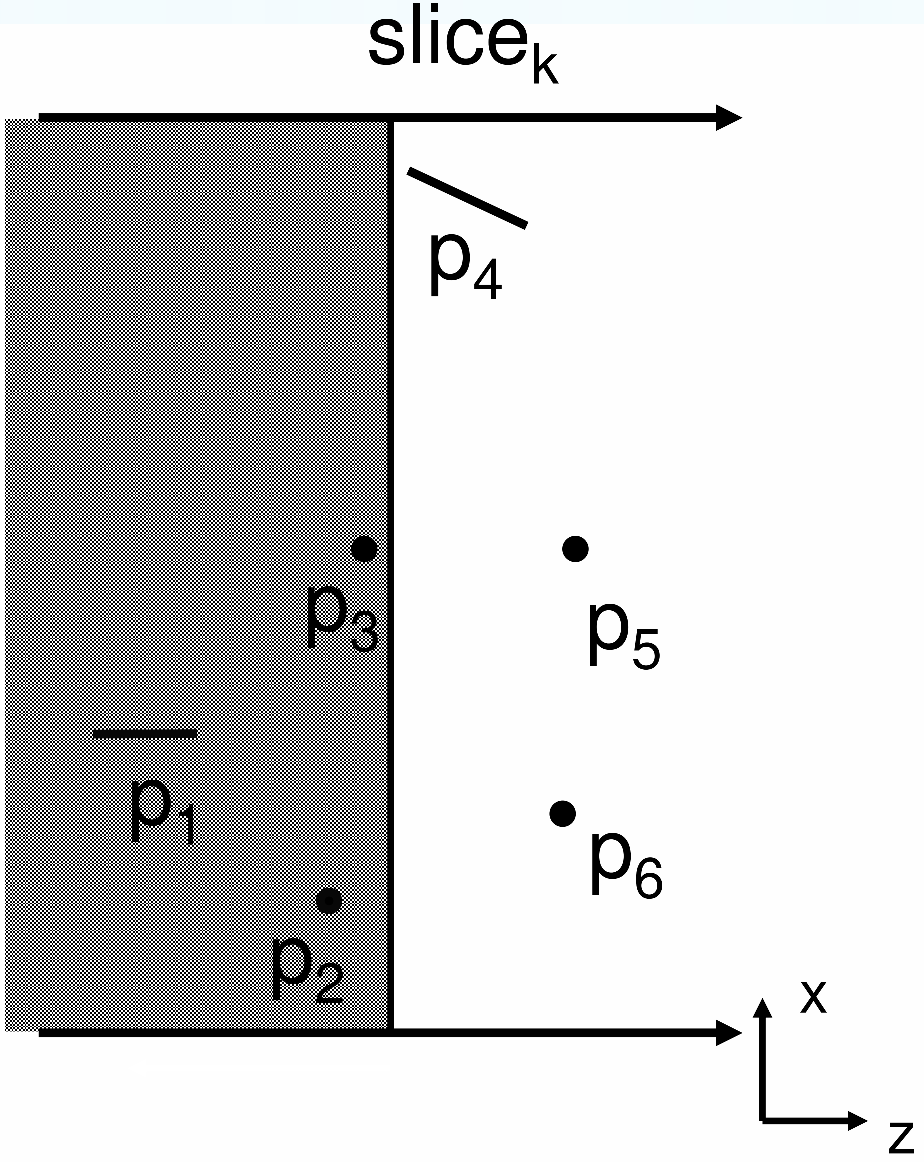
Site Classification

- For each slice, also partition set of sites using sweep direction



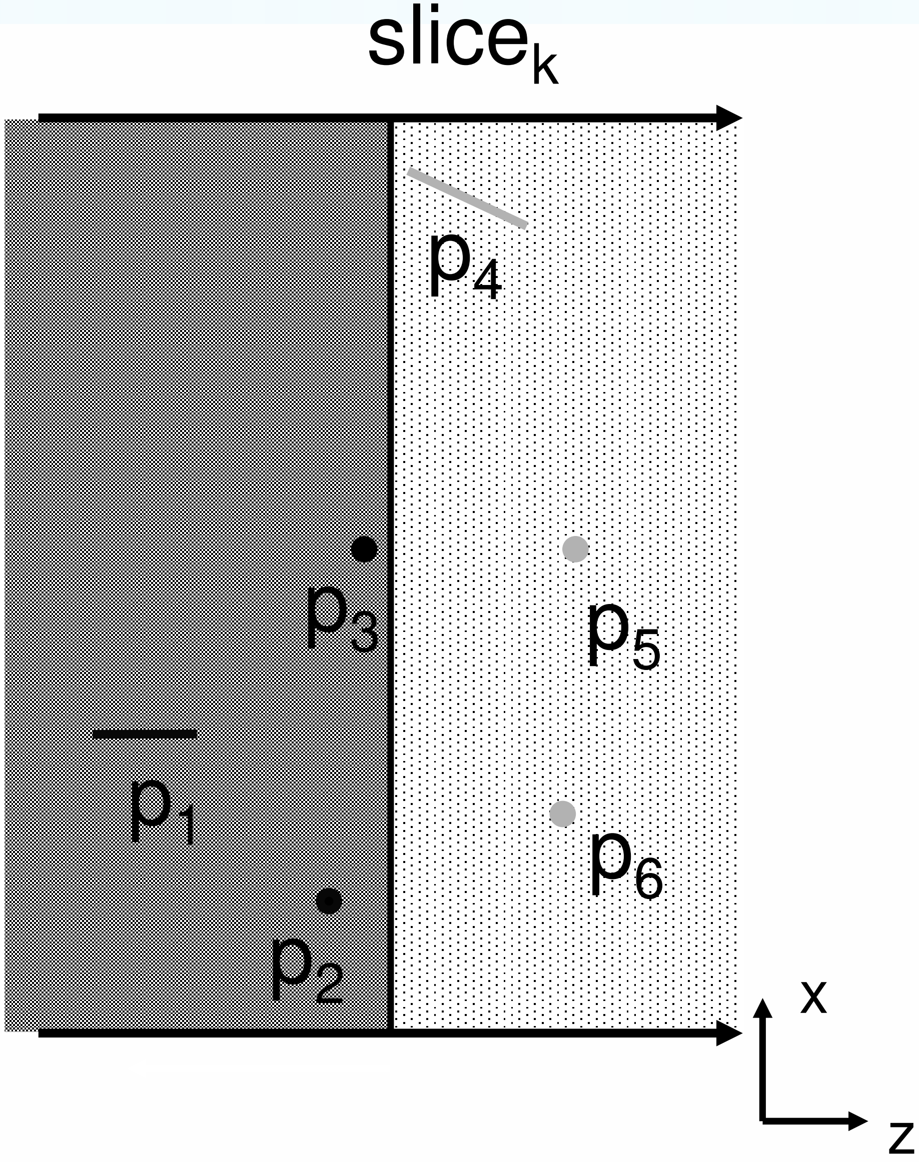
Site Classification

- For each slice, also partition set of sites using sweep direction
  - Swept



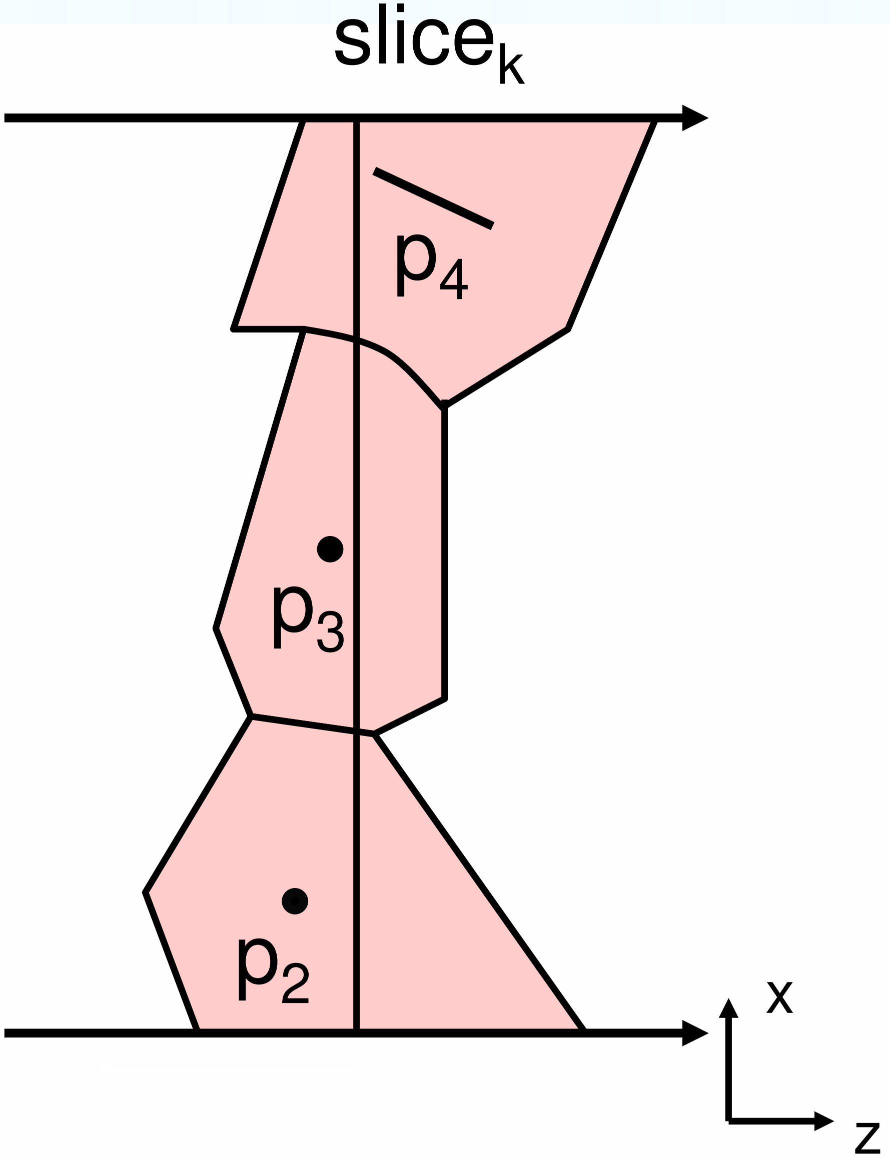
Site Classification

- For each slice, also partition set of sites using sweep direction
  - Swept
  - Unswept



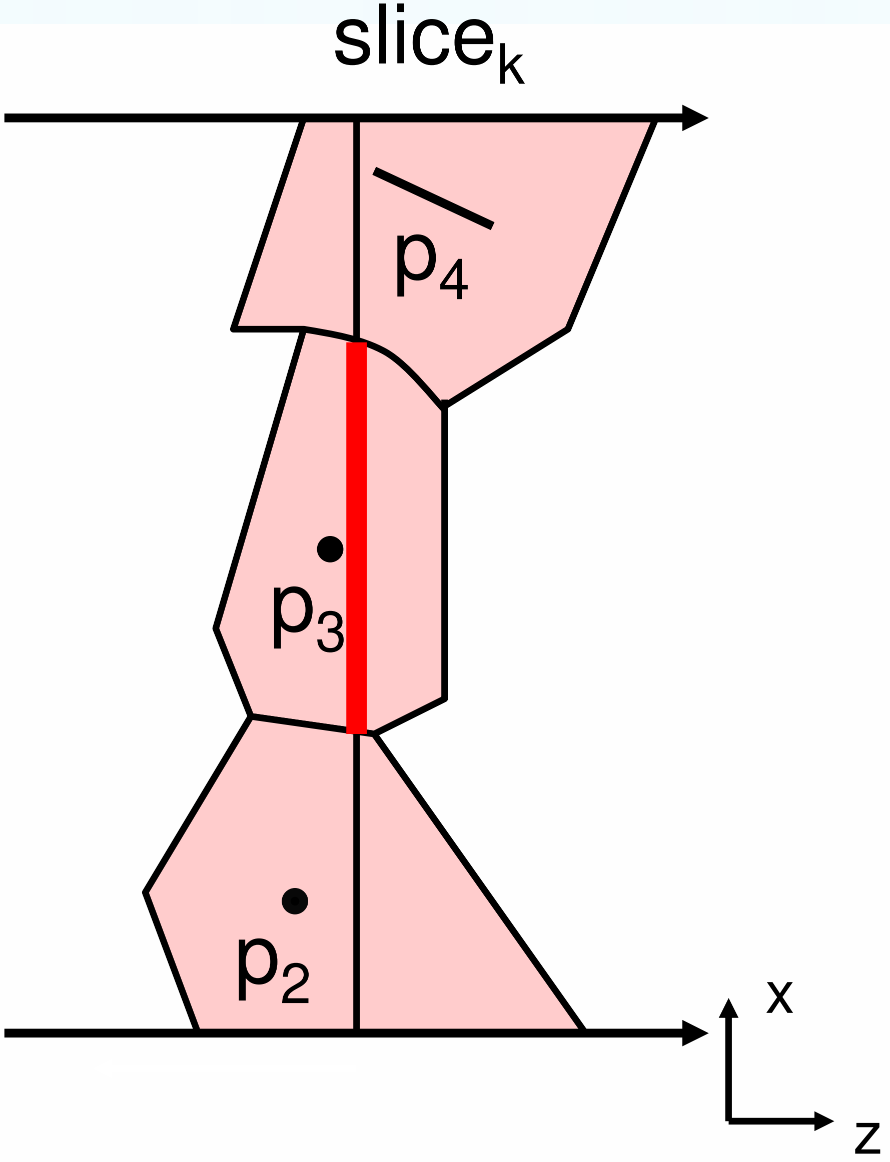
Acceleration Techniques

- *Culling:* Render distance functions for intersecting sites only



Acceleration Techniques

- *Culling:* Render distance functions for intersecting sites only
- *Clamping:* For each intersecting site, clamp domain of computation

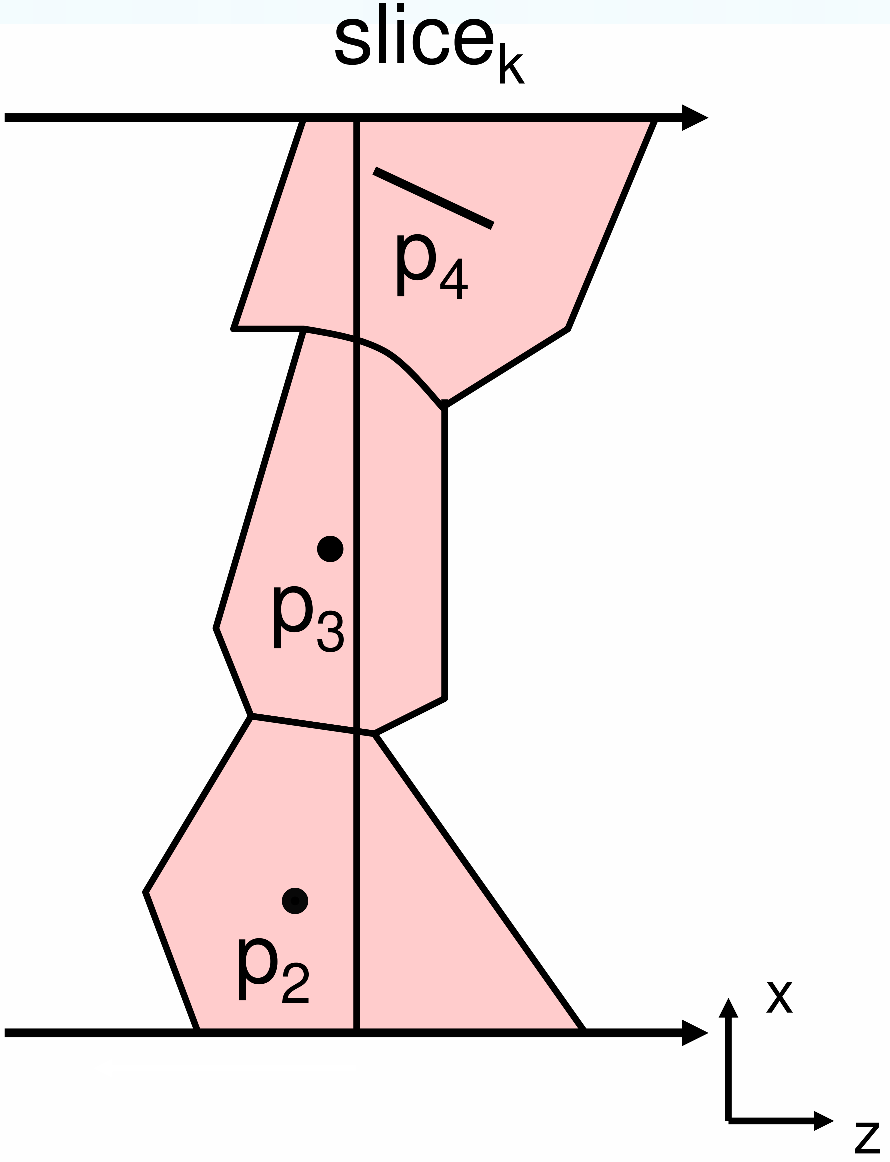


Outline

- Related Work
- Fast GPU based algorithm (DiFi)
  - Motivation
  - Geometric properties
  - Site classification
  - Culling algorithm
  - Clamping algorithm
- Applications and Results
- Conclusions

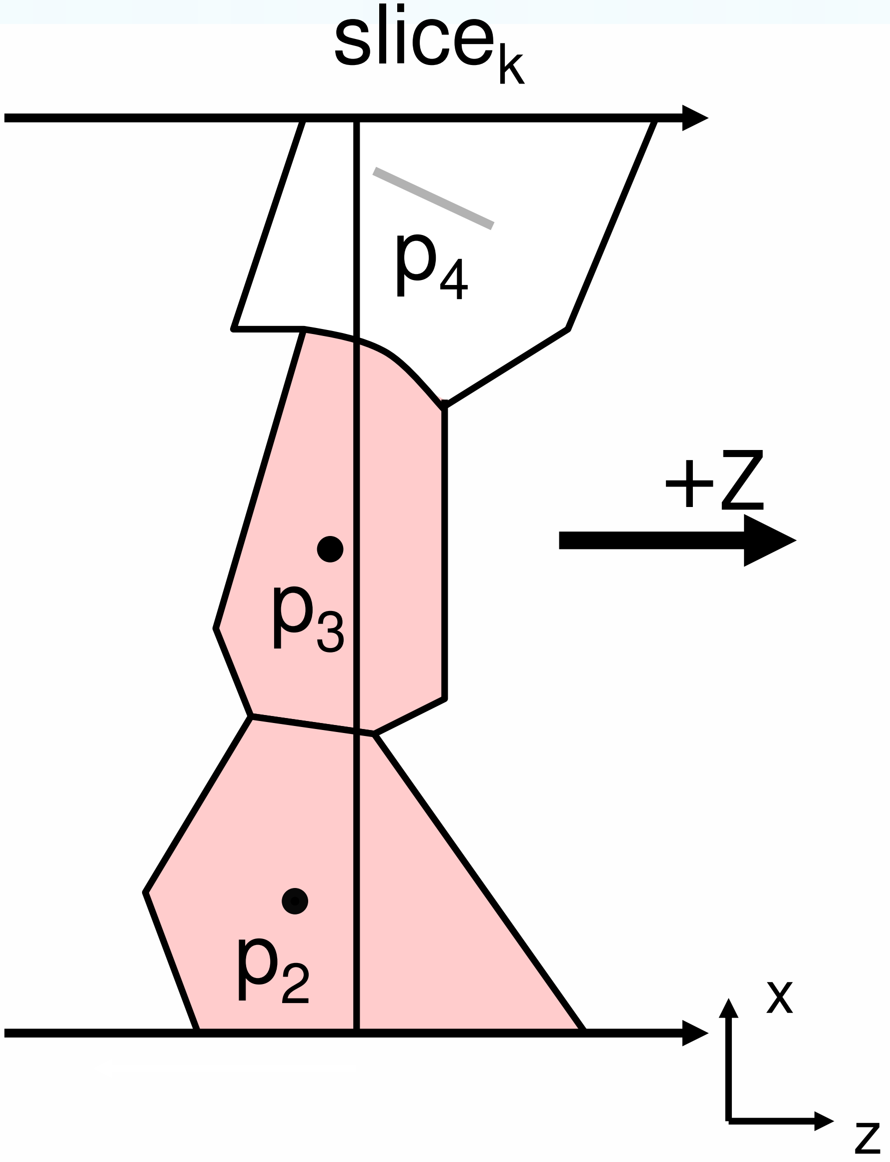
Culling: Goal

- Render distance functions for *intersecting* sites only



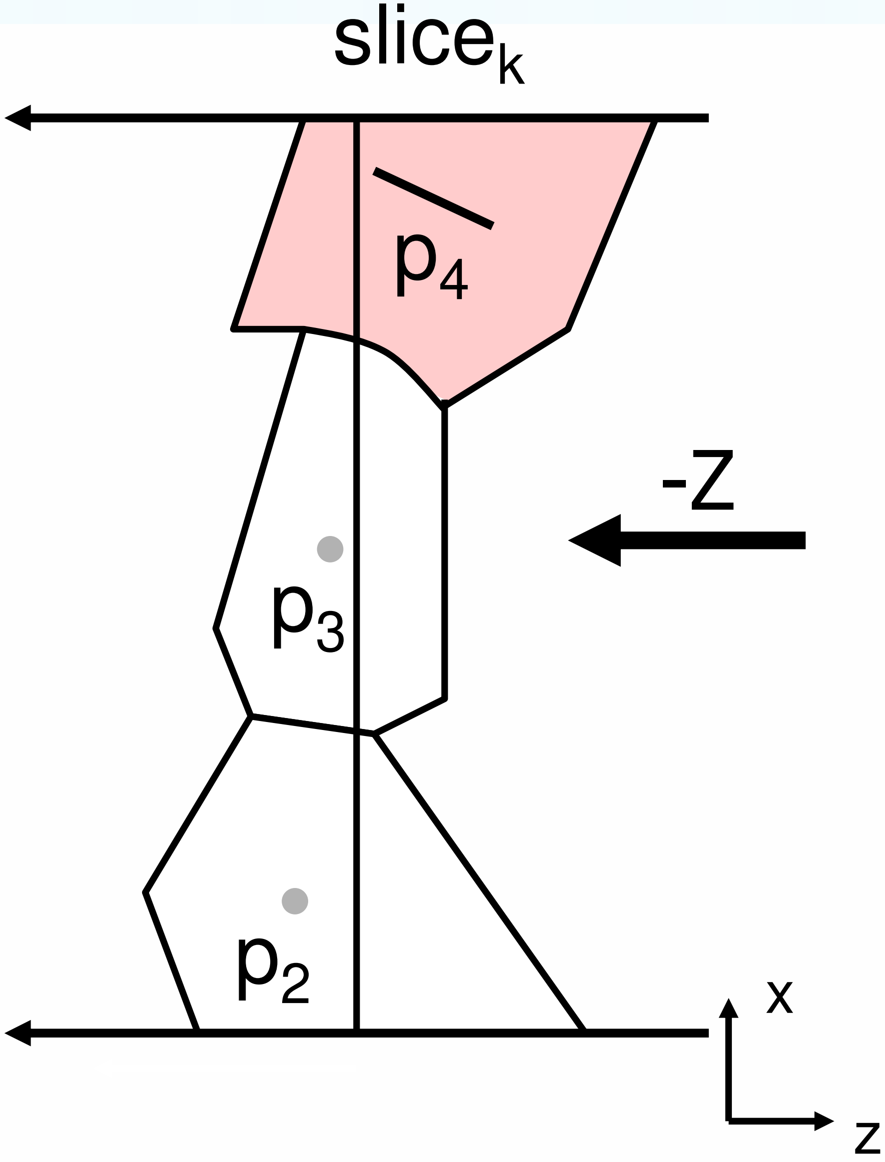
Culling: 2 Pass Algorithm

- Render distance functions for *intersecting swept* sites: +Z pass



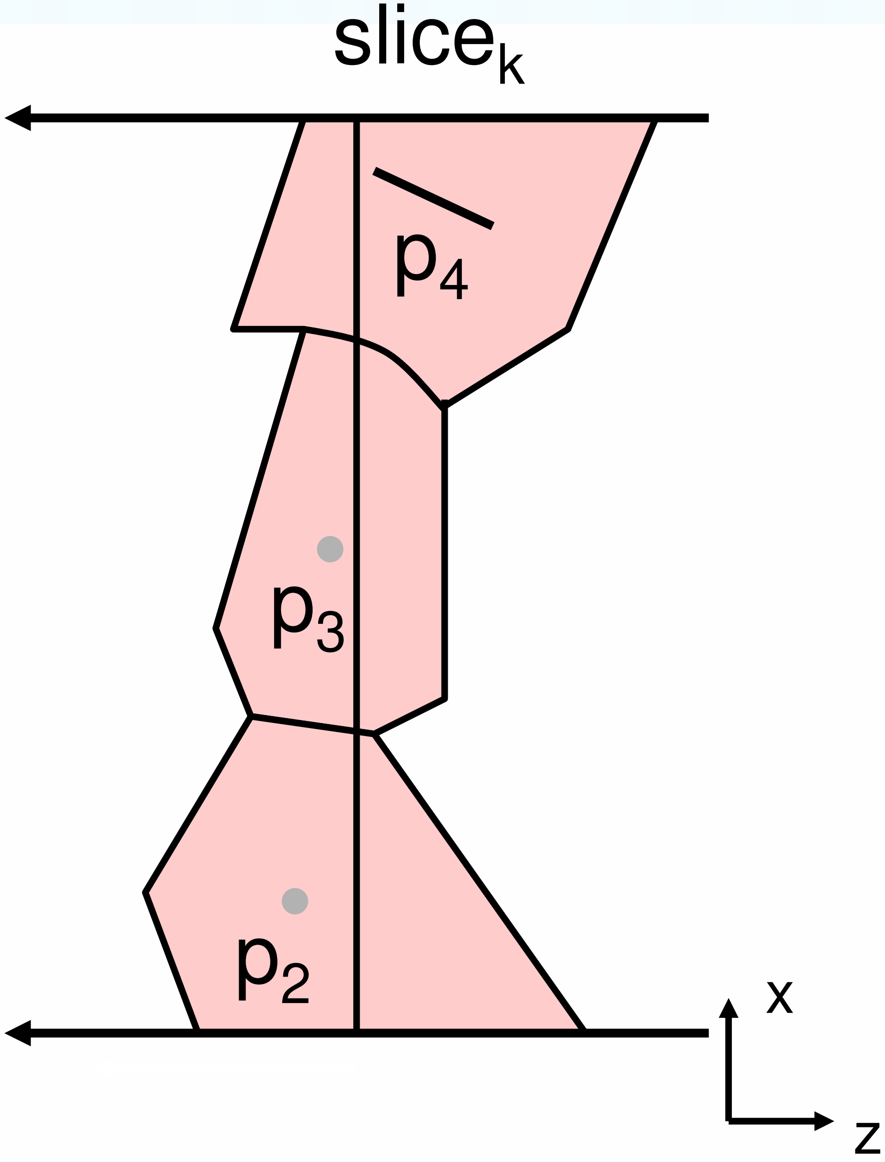
Culling: 2 Pass Algorithm

- Render distance functions for *intersecting swept* sites: -Z pass



Culling: 2 Pass Algorithm

- Render distance functions for *intersecting swept* sites
- Final distance field obtained after both passes

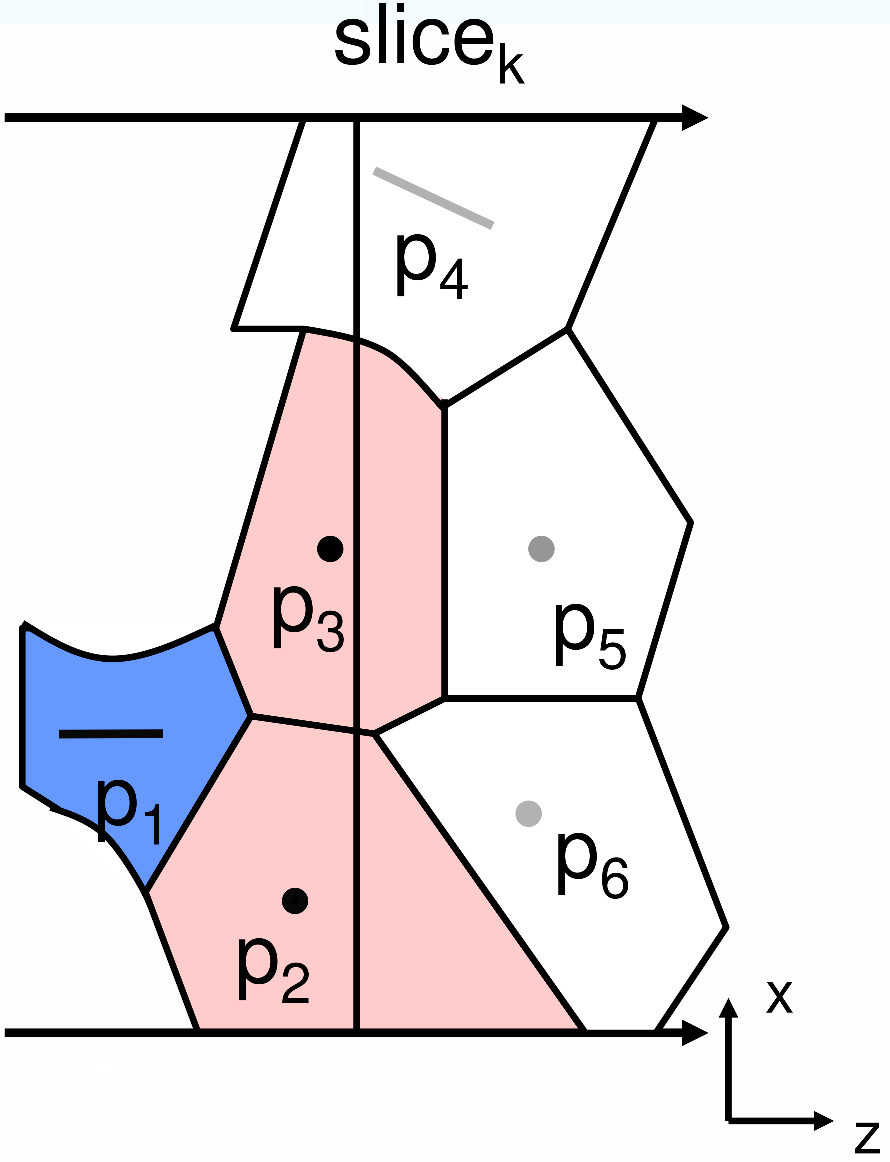


Culling

- Computing exact intersecting set = Exact Voronoi computation
- Swept set easy to compute
- Compute a set of *potentially intersecting swept (PIS)* sites
- Use hardware based occlusion queries to compute *PIS*

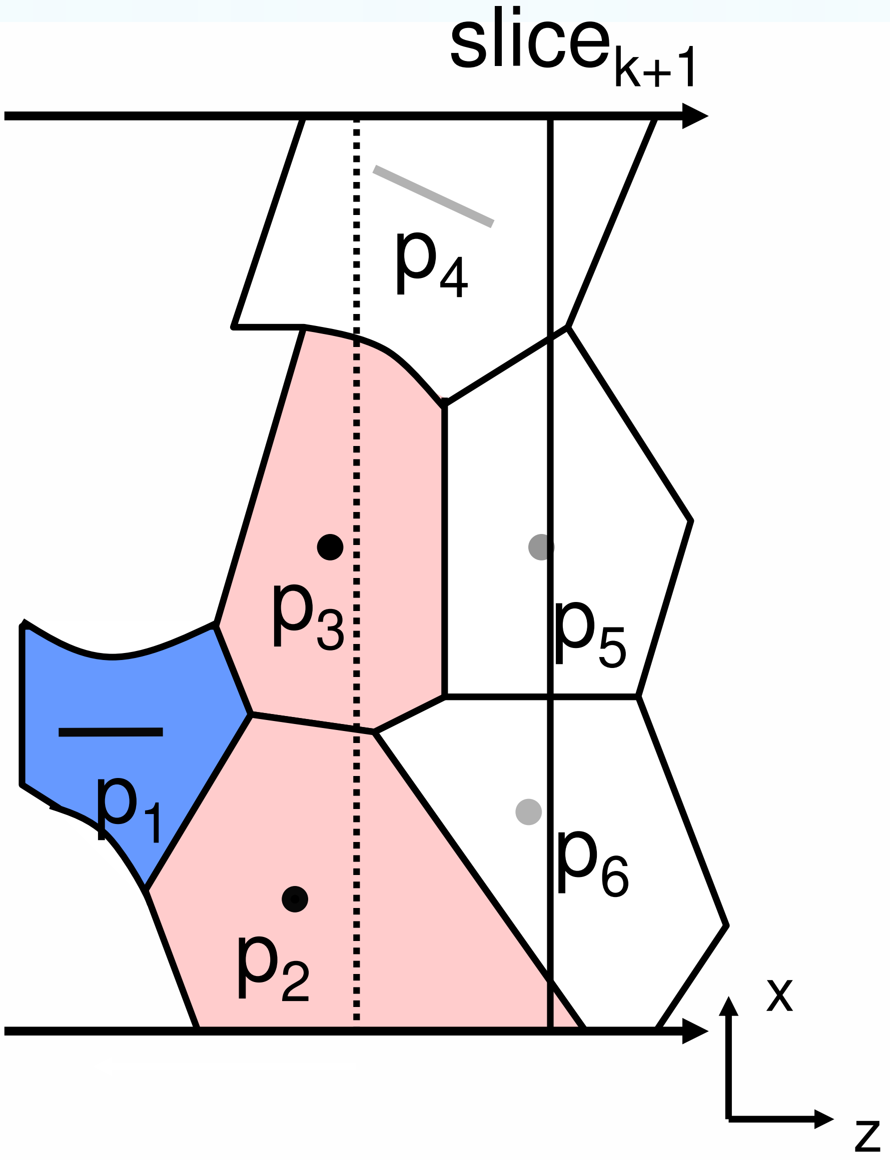
Culling: Computing *PIS*

- Given the *potentially intersecting swept* set for slice k



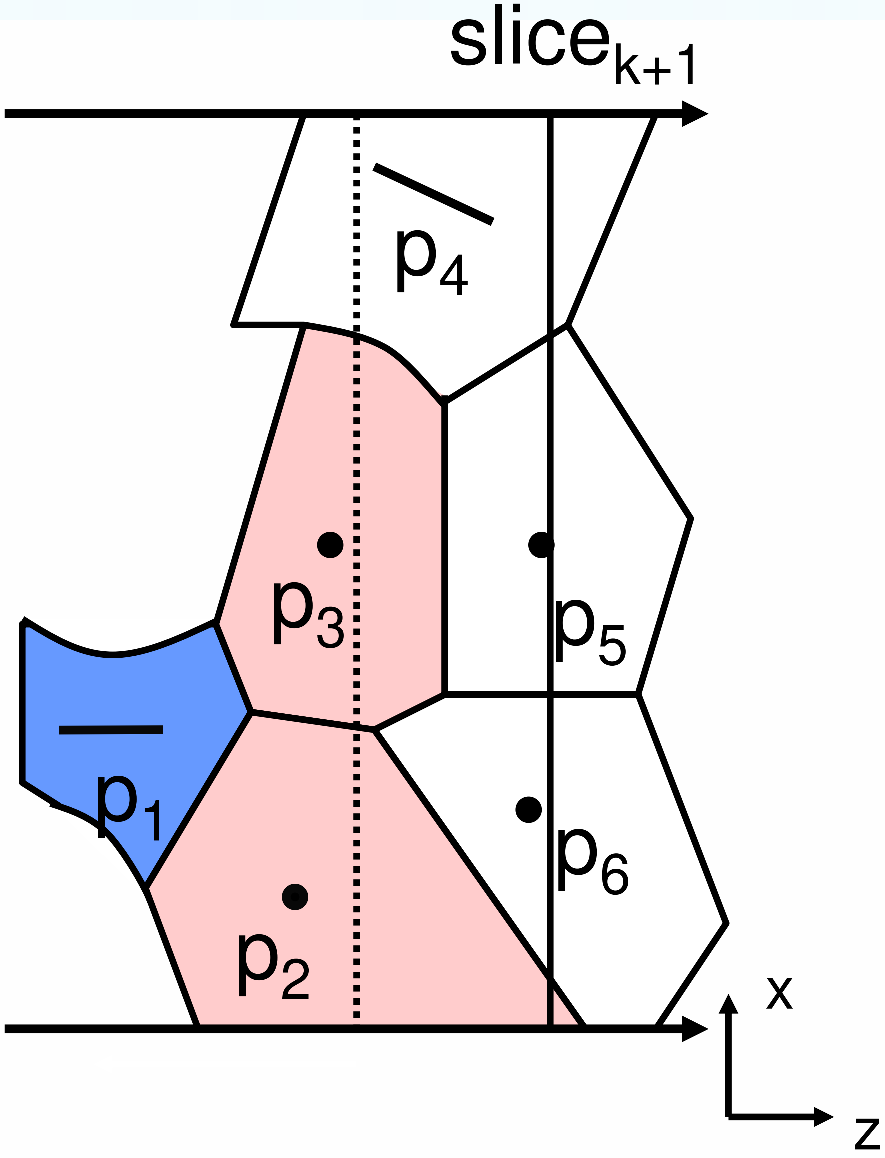
Culling: Computing *PIS*

- For slice k+1:



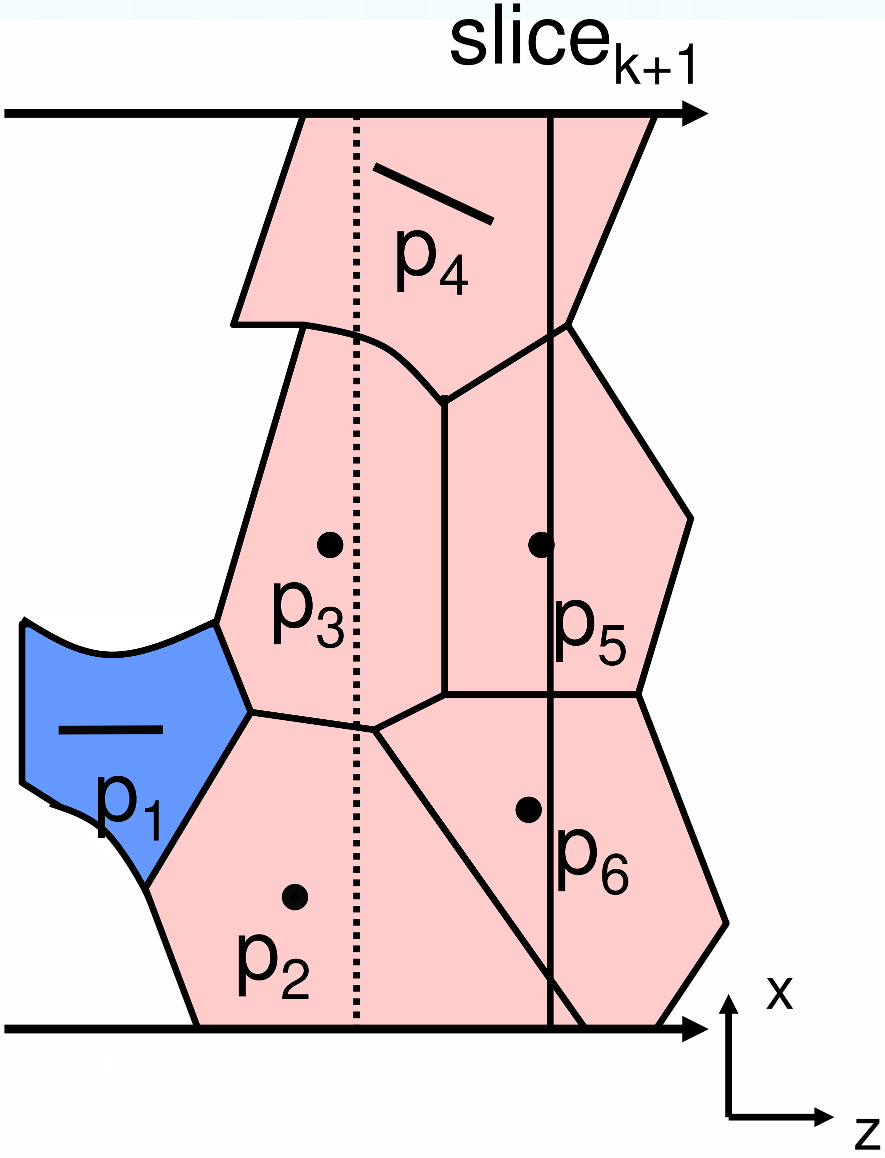
Culling: Computing *PIS*

- For slice k+1:
  - Add newly swept sites to *PIS*



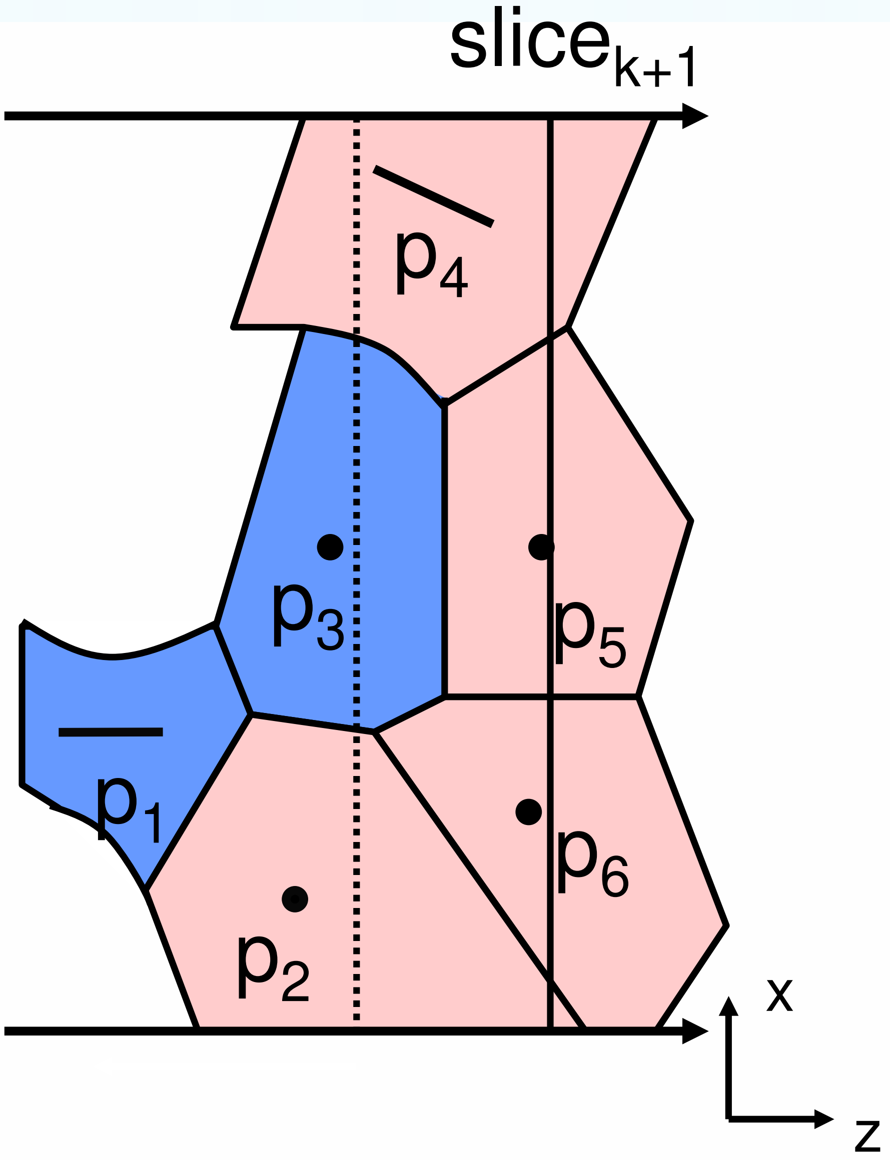
Culling: Computing *PIS*

- For slice k+1:
  - Add newly swept sites to *PIS*
  - Draw distance functions of new *PIS*



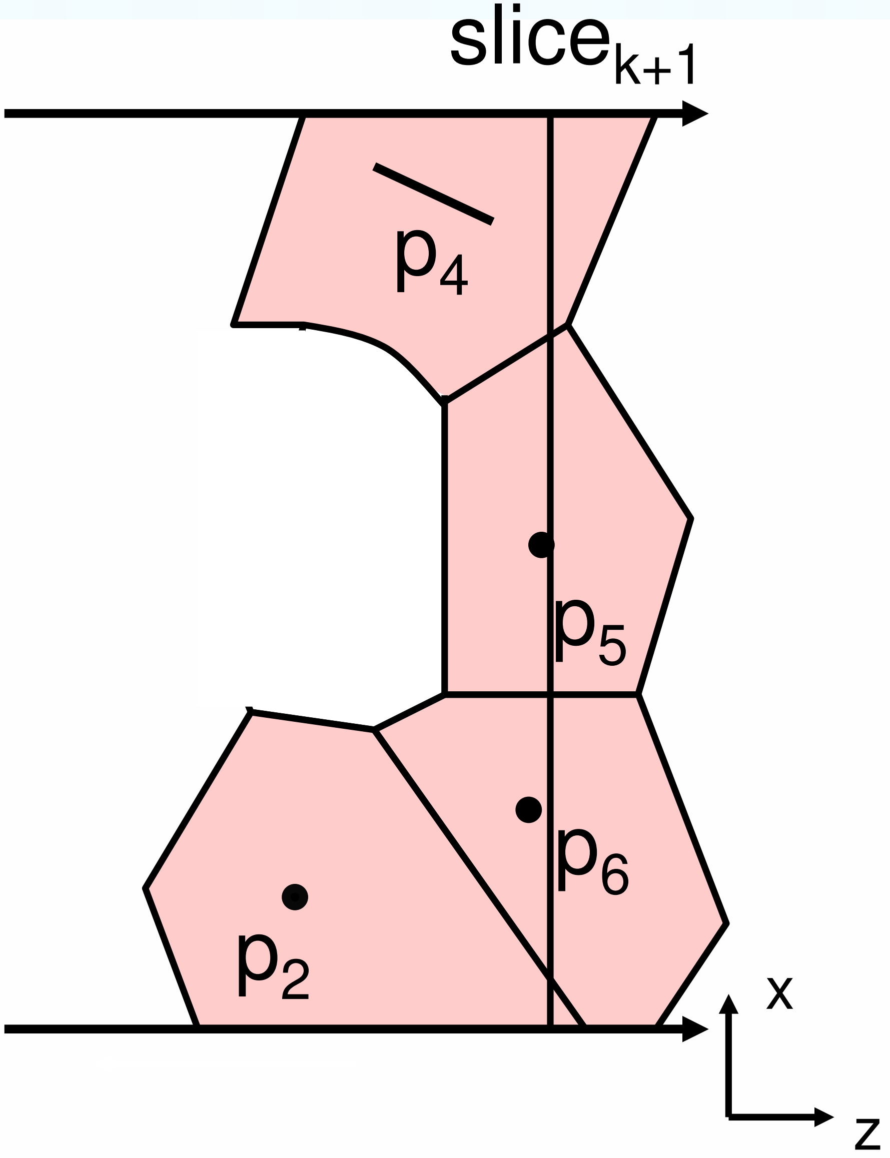
Culling: Computing *PIS*

- For slice k+1:
  - Add newly swept sites to intersecting set
  - Draw distance functions of new intersecting set
  - Check visibility and update receding set



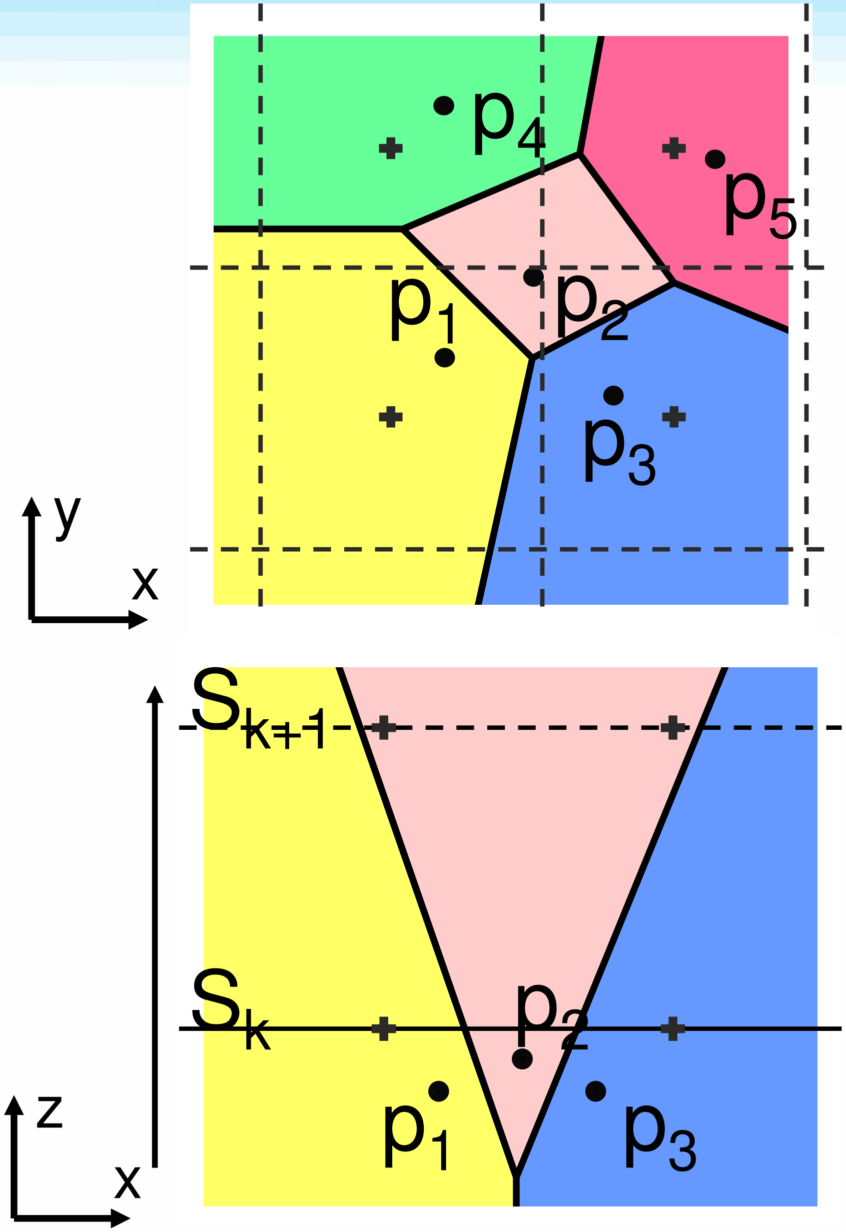
Culling: Computing *PIS*

- For slice k+1:
  - Add newly swept sites to intersecting set
  - Draw distance functions of new intersecting set
  - Check visibility and update receding set
  - Get final intersecting swept set for slice k+1



Culling: Conservative Sampling

- Issue: Image space occlusion query may under sample a Voronoi region
  - Wrongly classifies a site as receding
- Solution: “Grow” the Voronoi region by pixel size (details in paper)

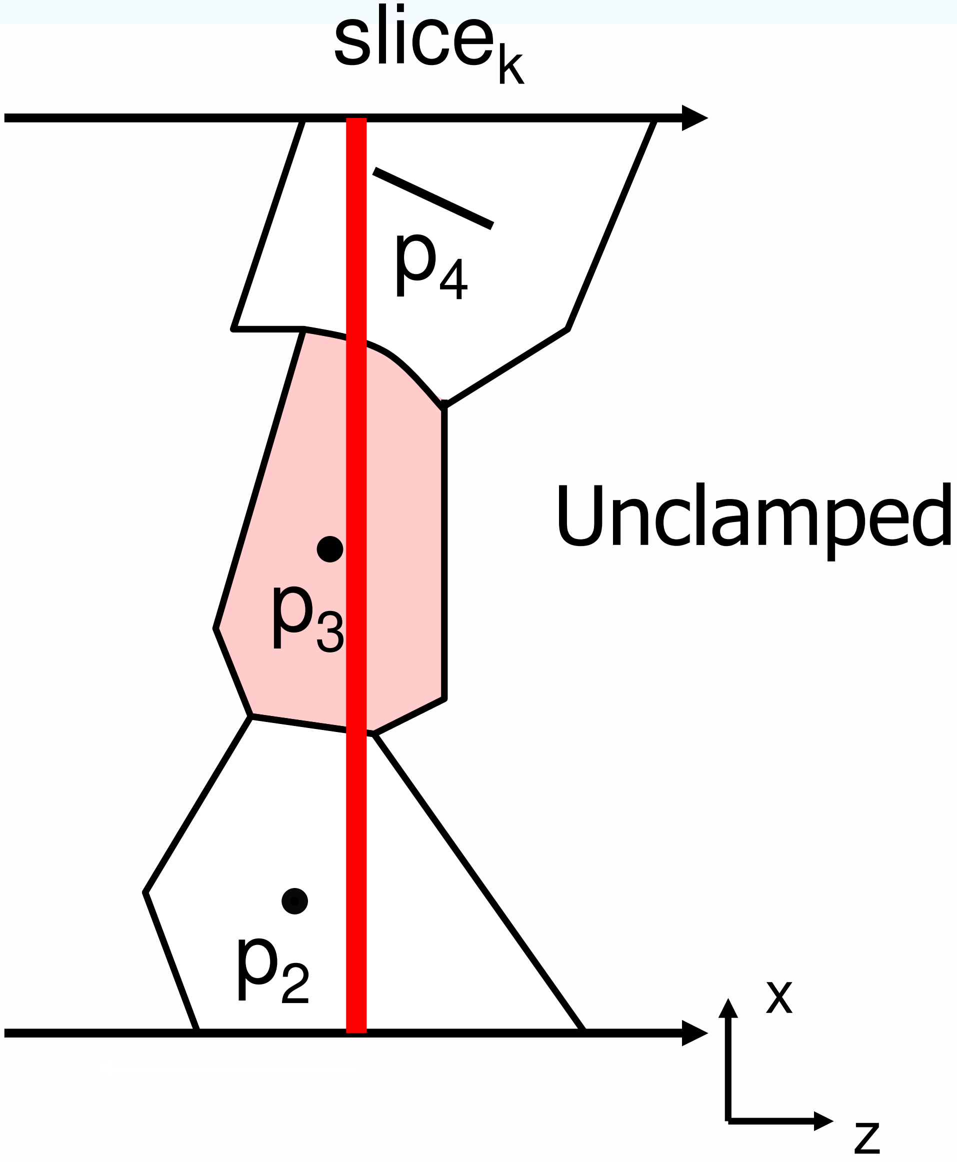


Outline

- Related Work
- Fast GPU based algorithm (DiFi)
  - Motivation
  - Geometric properties
  - Site classification
  - Culling algorithm
  - Clamping algorithm
- Applications and Results
- Conclusions

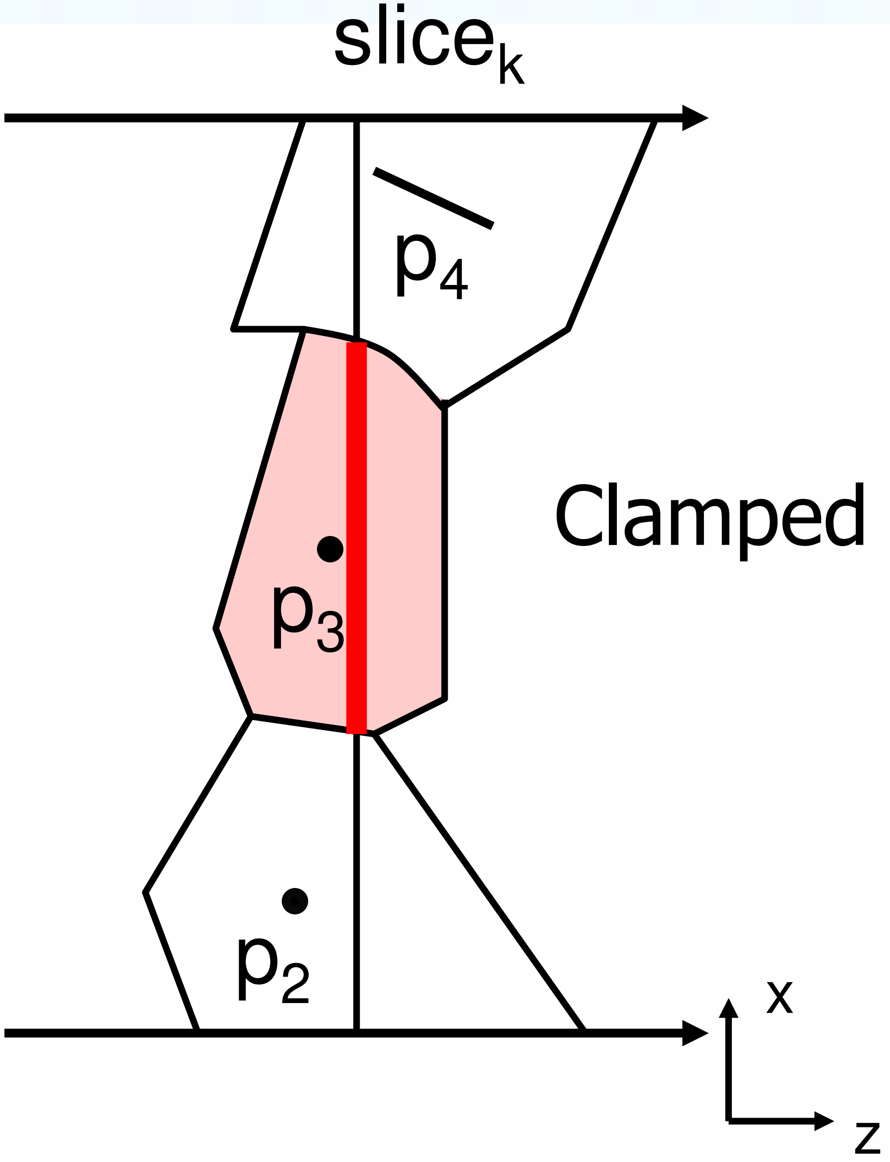
Clamping: Goal

- *Clamping:* For each intersecting site, clamp domain of computation



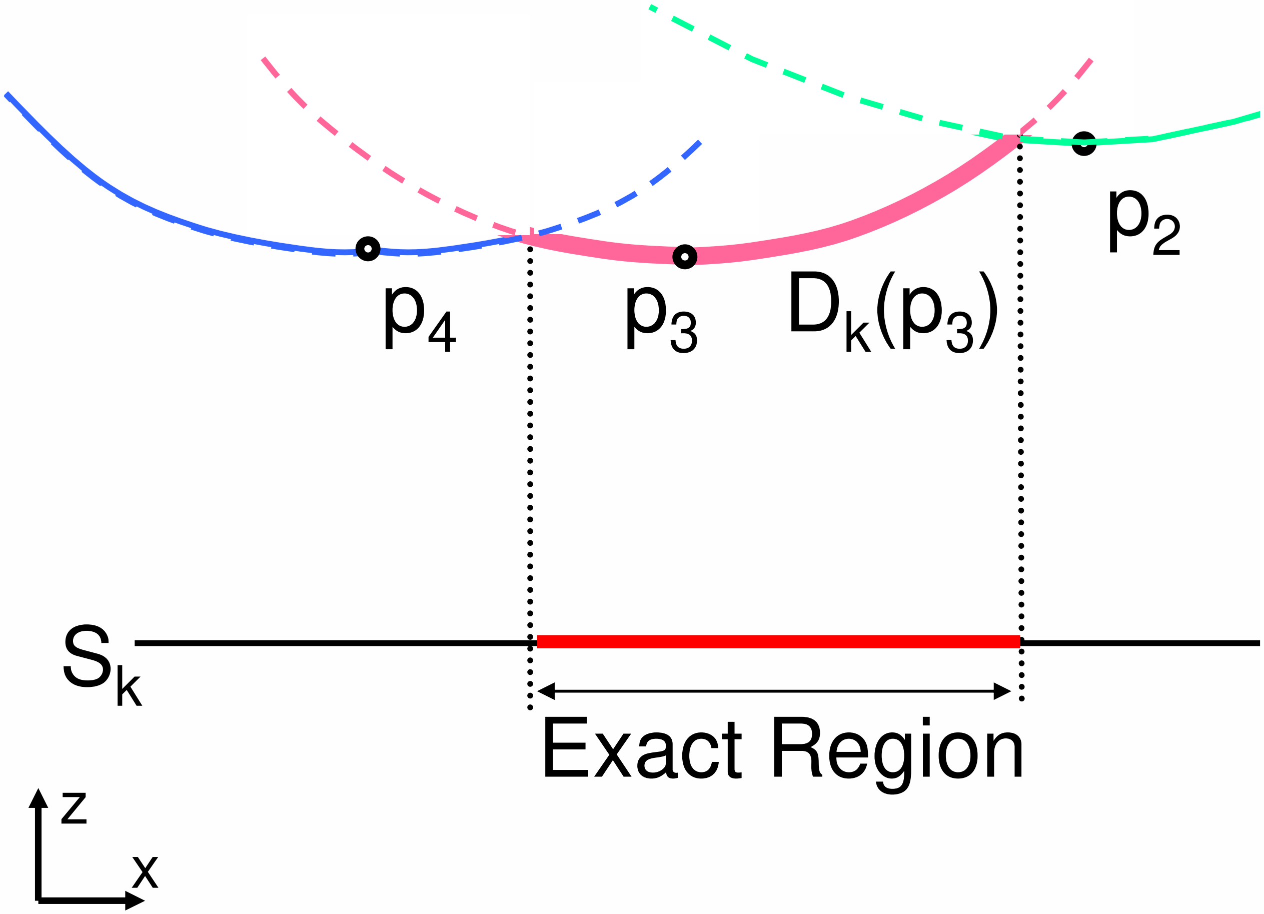
Clamping: Goal

- *Clamping:* For each intersecting site, clamp domain of computation
- Domain of computation = Intersection of Voronoi Region with slice



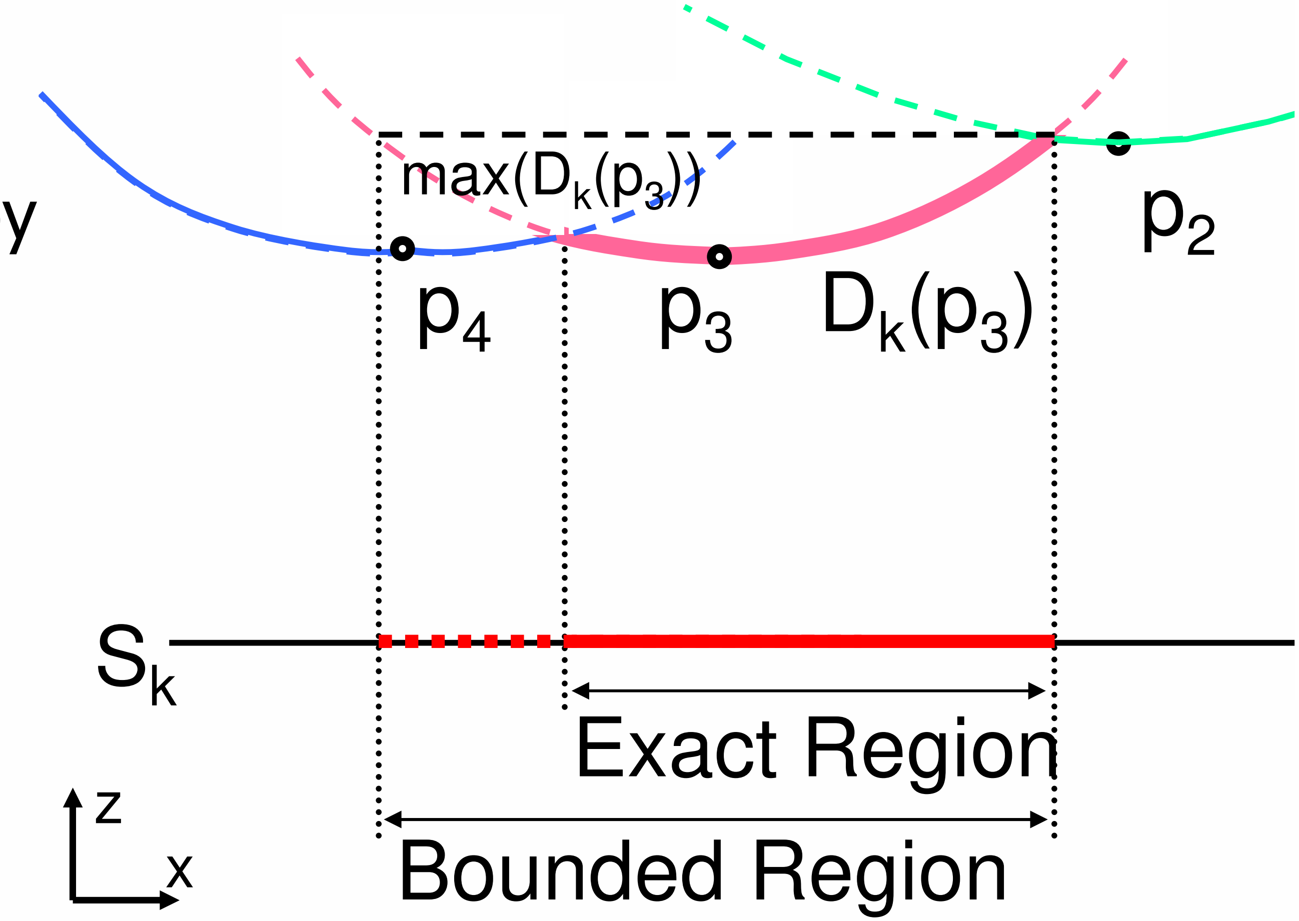
Clamping

- Distance function of each site is monotonic
- The exact Voronoi region



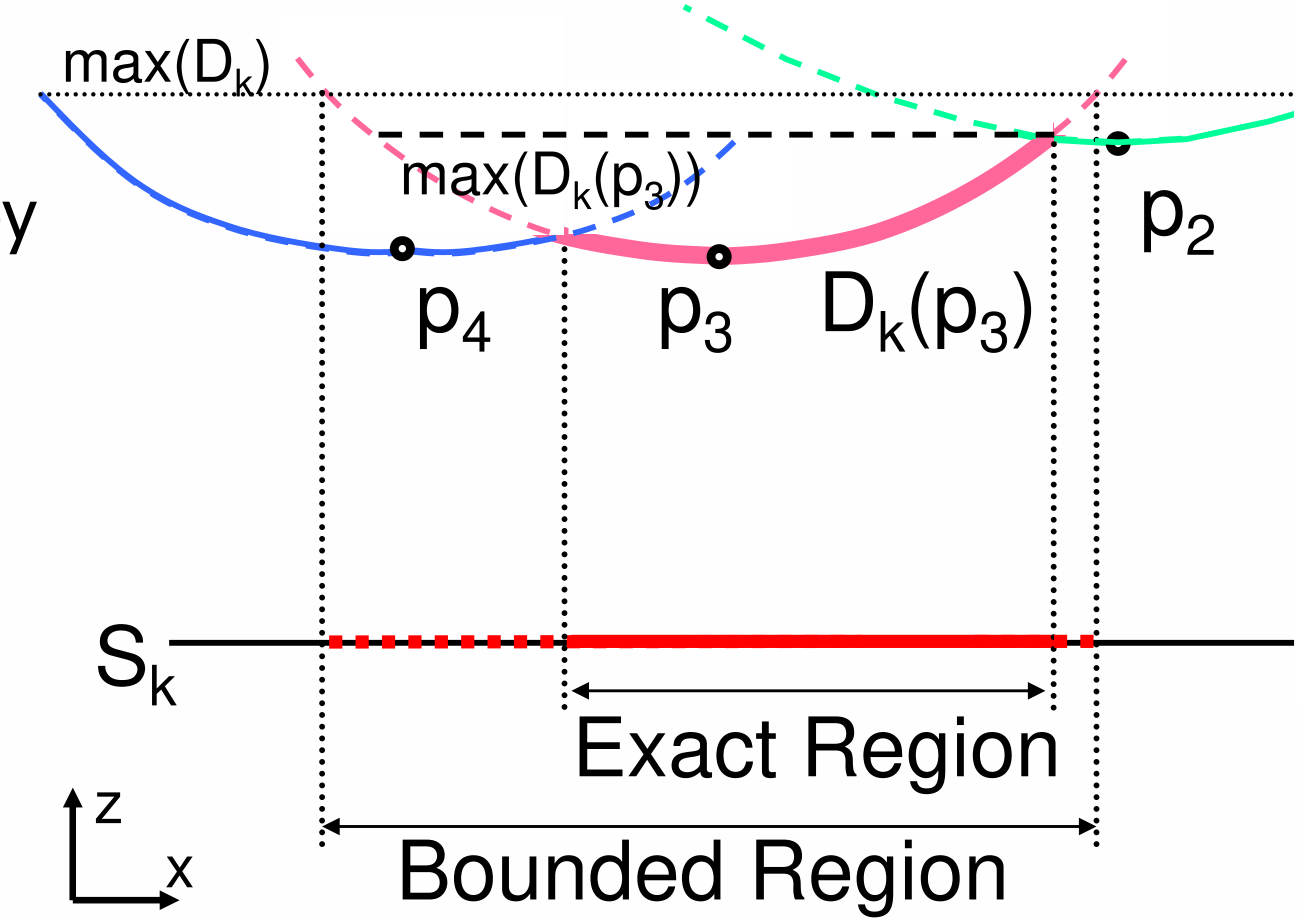
Clamping

- Distance function of each site is monotonic
- The exact Voronoi region is bounded by max of distance function



Clamping

- Distance function of each site is monotonic
- The exact Voronoi region is bounded by max of distance function, which is bounded by max of distance field, $\max(D_k)$



Clamping

- Compute max(D_{k+1}) for slice k+1 incrementally using max(D_k)

Lemma: Let distance between adjacent slices be δ_z . Then change in maximum value of distance field between slices S_k and S_{k+1} is given by:

$$max(D_{k+1}) \leq max(D_k) + \delta_z$$

- Use max(D_{k+1}) for clamping

Clamping: Manifold Sites

- Voronoi region bounded by prisms, wedges and cones [Mauch00, Sigg03]
- For each *manifold site*, refine Voronoi region bounds using prism, wedge or cone bounds

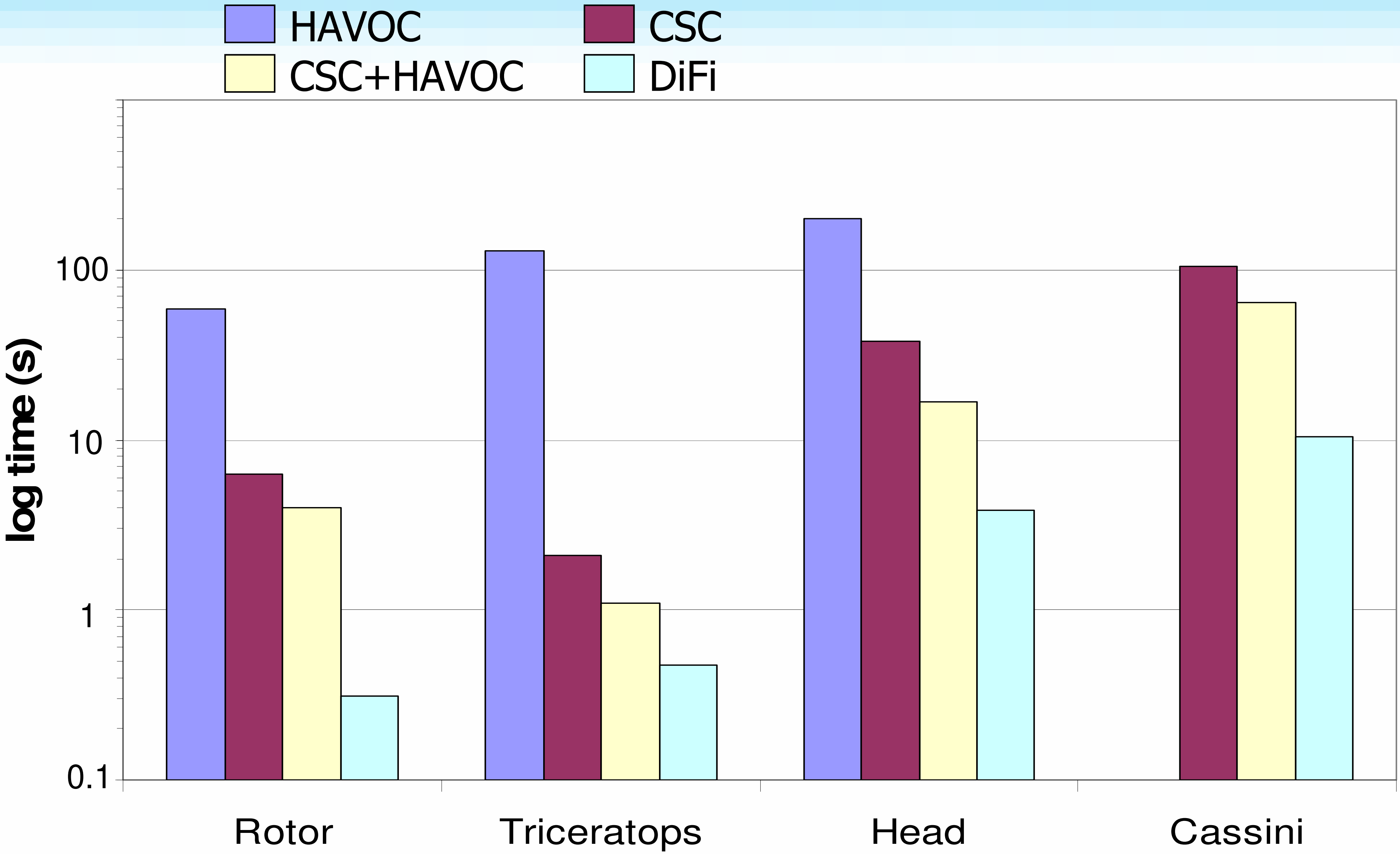
Outline

- Related Work
- Fast GPU based algorithm (DiFi)
- Applications and Results
- Conclusions

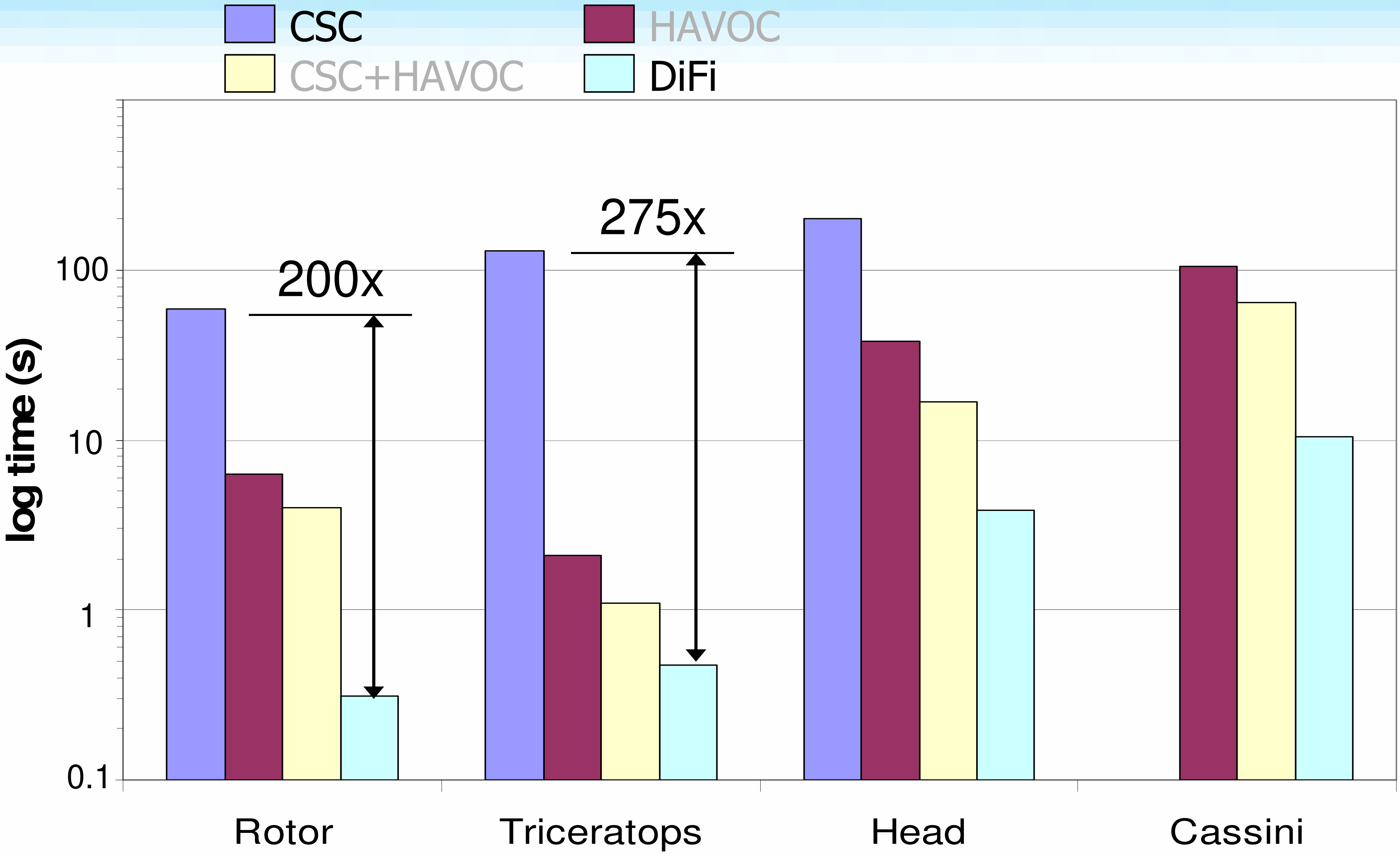
Implementation

- Pentium4 2.8Ghz, 2GB RAM
- NVIDIA GeForce FX 5900 Ultra, 256MB Video RAM
- Windows XP, OpenGL
- HAVOC3D [Hoff99]

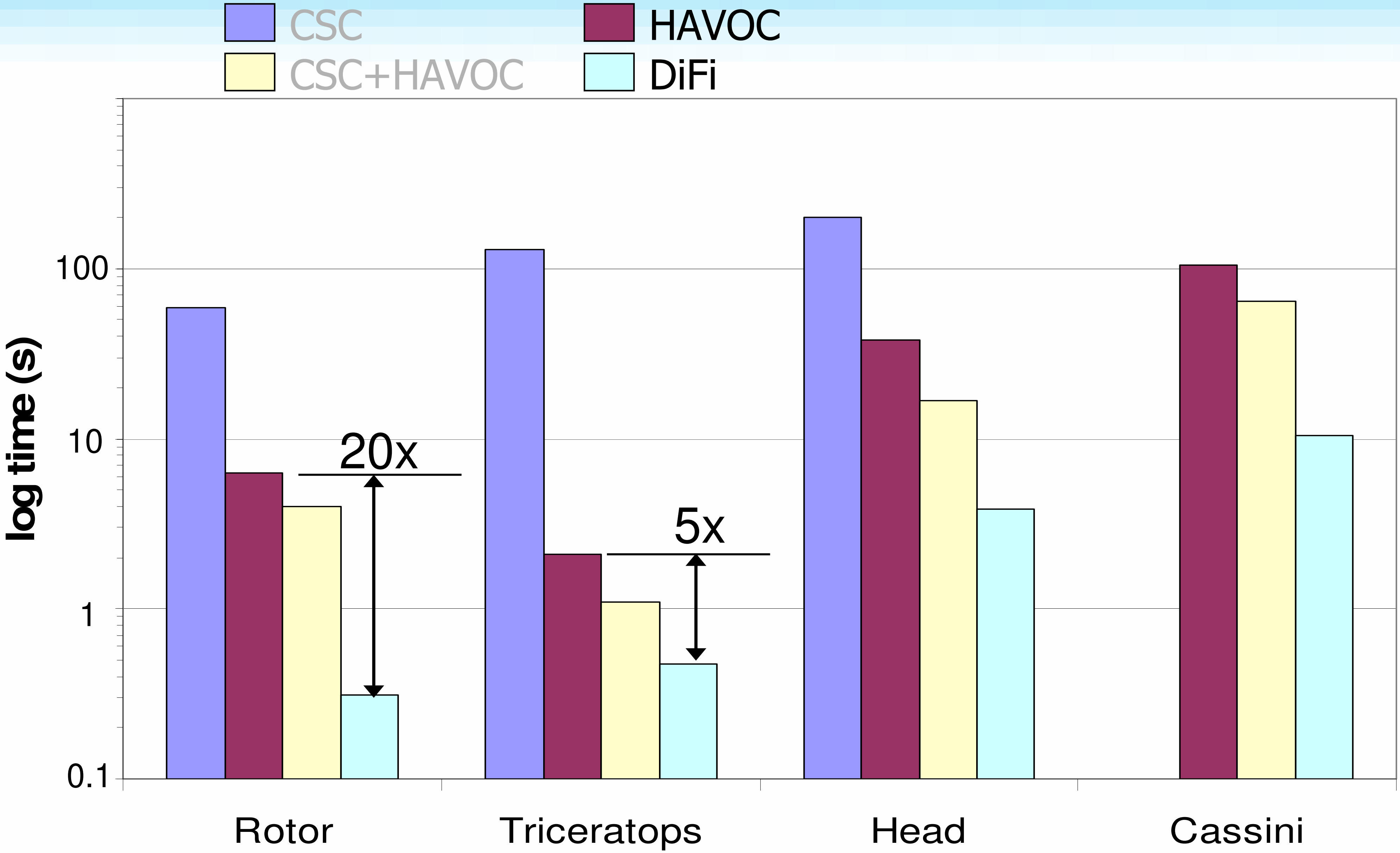
Results: Distance Field



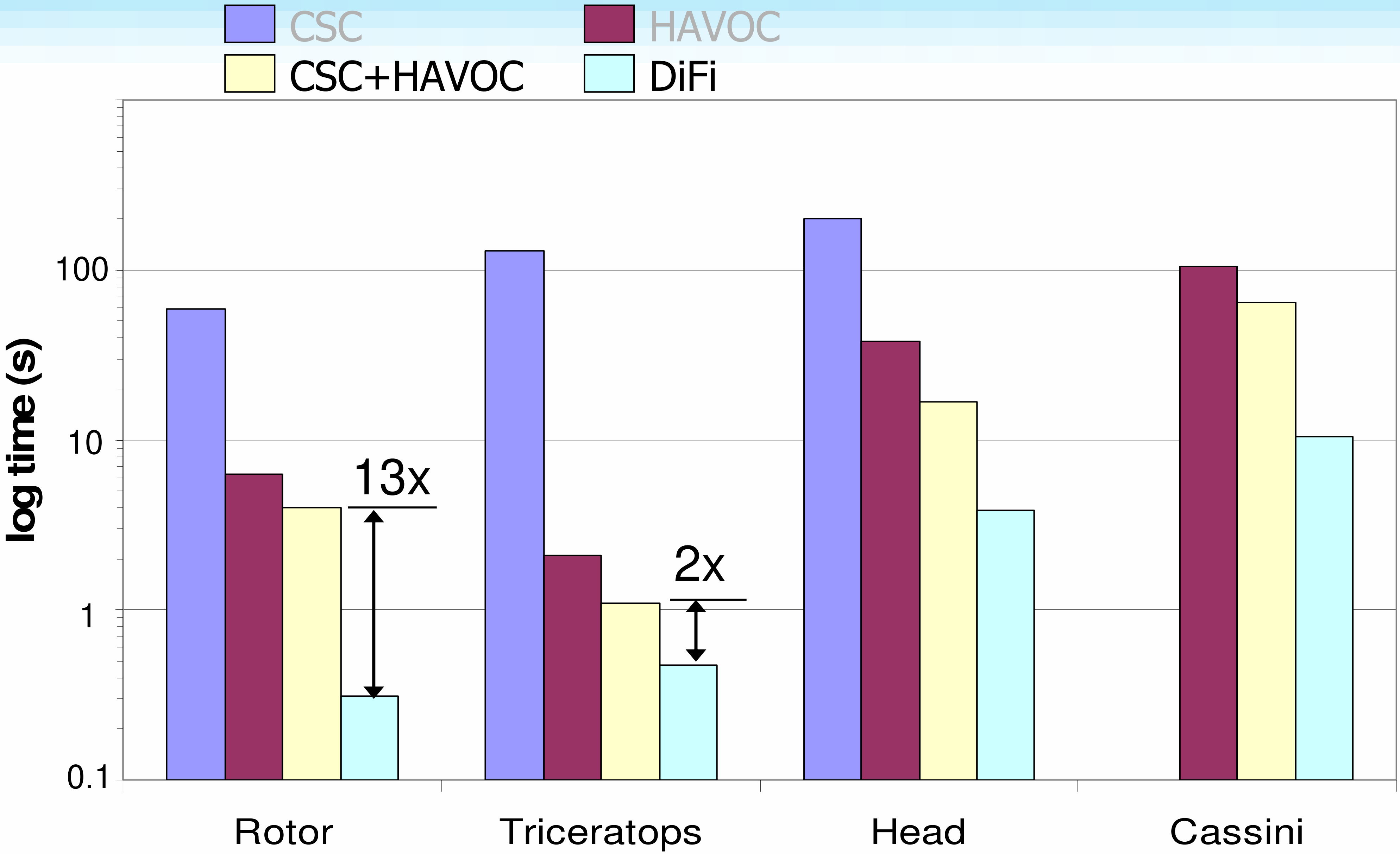
Results: Distance Field



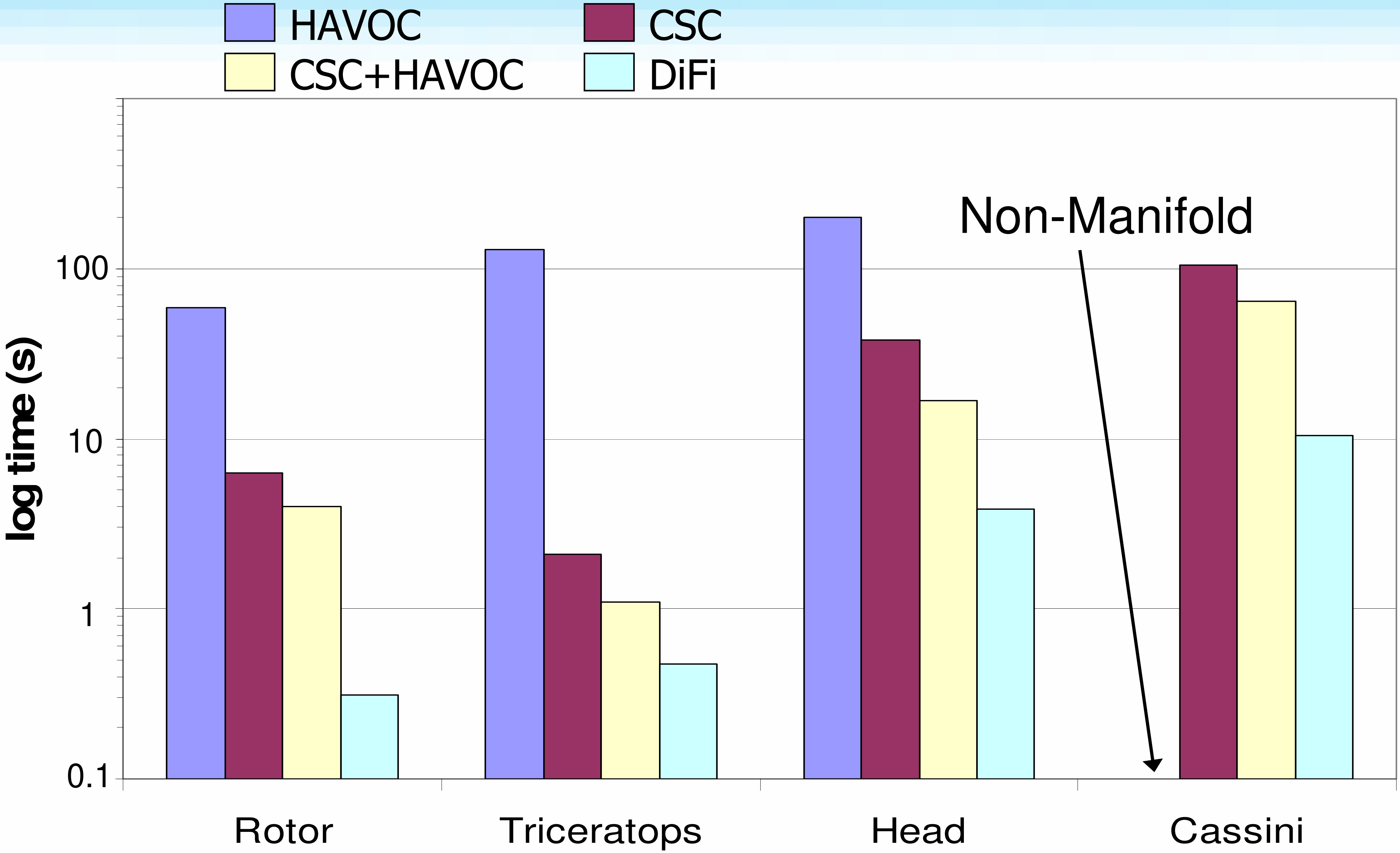
Results: Distance Field



Results: Distance Field

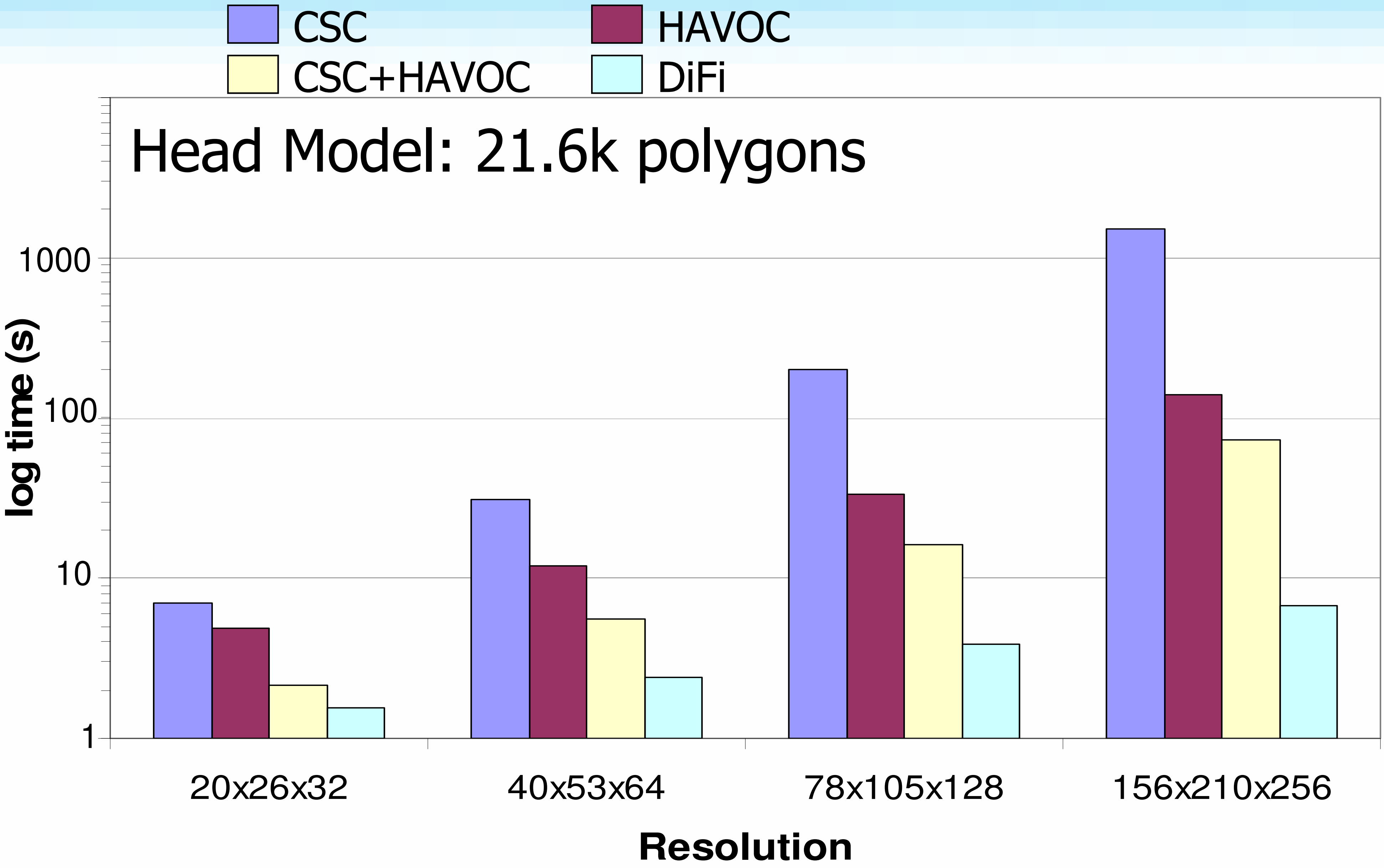


Results: Distance Field

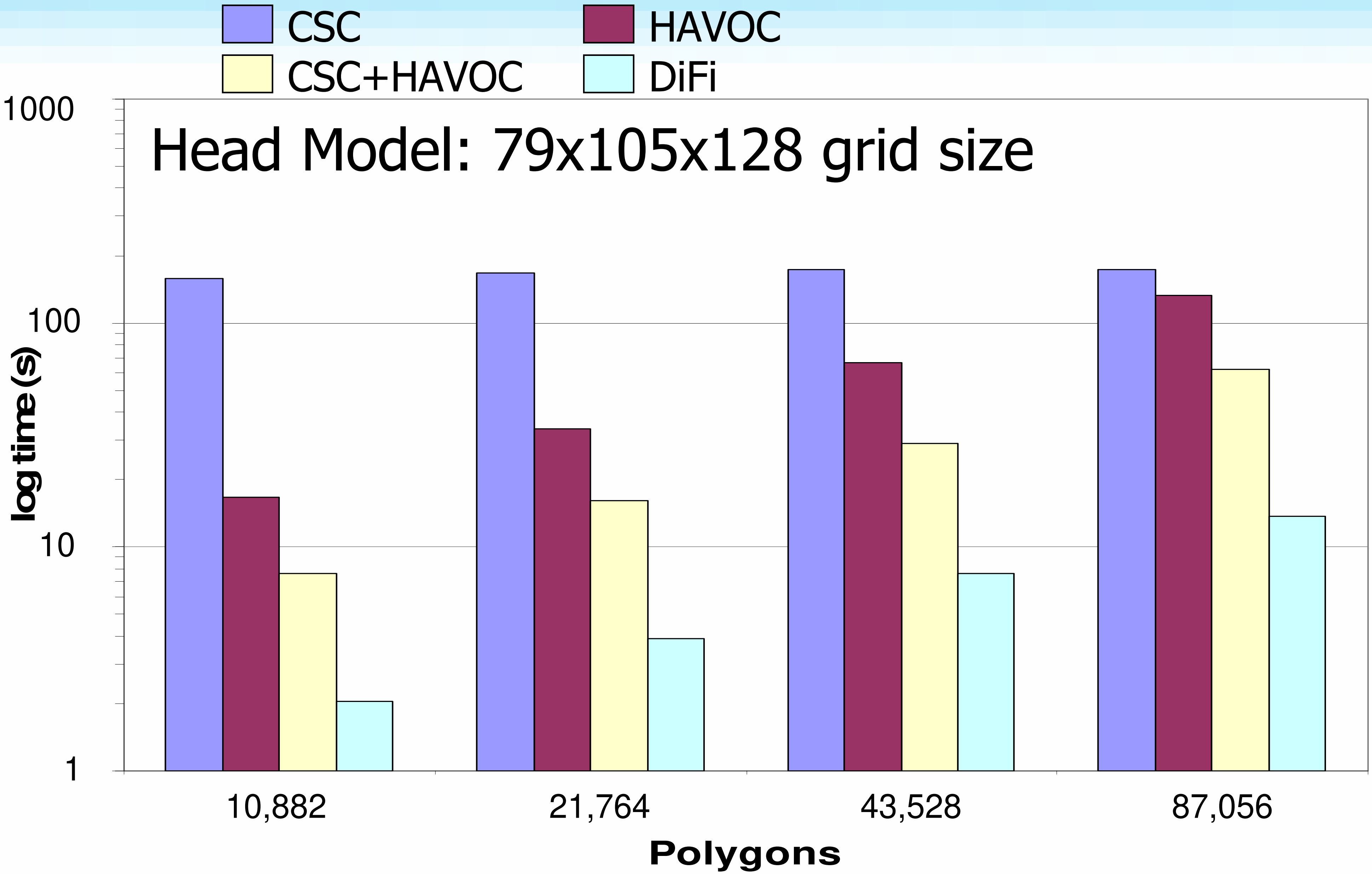


Non-Manifold

Results: Varying Resolution

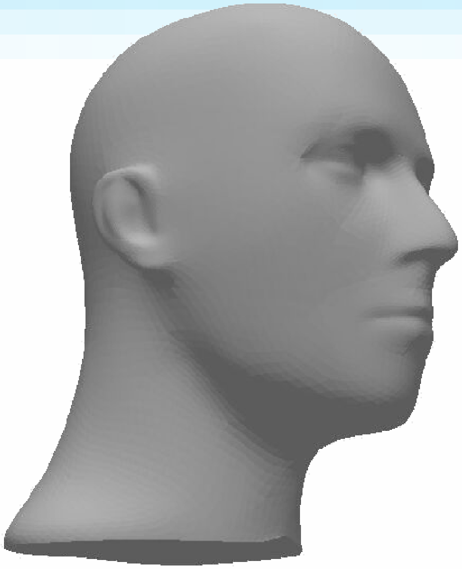


Results: Varying Polygon Count

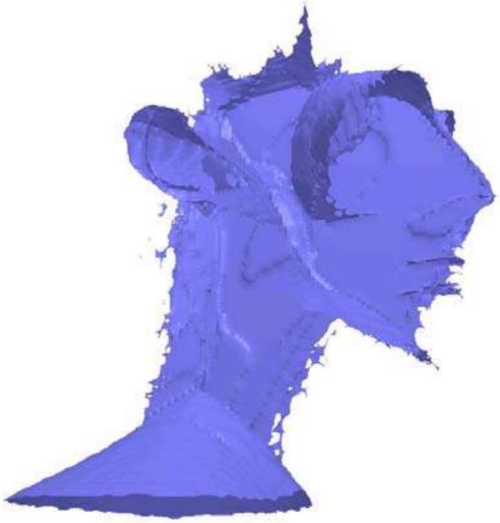


Applications: Medial Axis

- Compute a *simplified medial axis* using gradient of distance field [Foskey03]
- Stable subset of exact medial axis

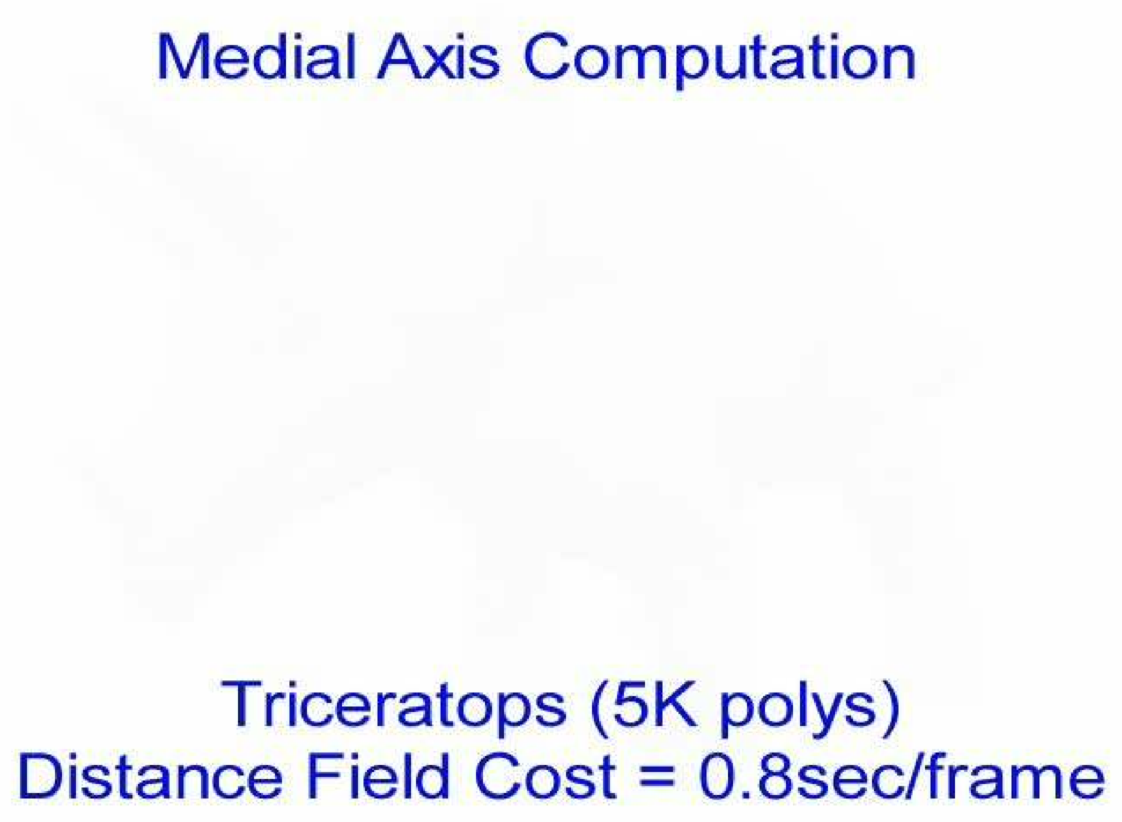


Head model



θ-simplified medial

Applications: Medial Axis



Medial Axis Computation

Triceratops (5K polys)
Distance Field Cost = 0.8sec/frame

Applications: Motion Planning

- DiFi used in a constraint-based planner [Garber02]
- Voronoi diagram → Estimated path
- Distance field → Proximity queries

Live Demo

- Laptop
  - Pentium4 3.2Ghz, 2GB RAM
  - NVIDIA GeForce FX Go5700, 128MB Video RAM
  - Windows XP, OpenGL

Outline

- Related Work
- Fast GPU based algorithm (DiFi)
- Applications and Results
- Conclusions

Conclusions

- A fast 3D distance field computation algorithm with an order of magnitude speedup
  - Almost interactive for complex 3D models
- Applicable to complex polygonal and image models
- No preprocessing
  - Applicable to dynamic environments

Conclusions

- Use geometric properties to reduce computations
  - Culling
  - Clamping
- Exploit spatial coherence for incremental computation
- Perform geometric tests efficiently on GPU
  - Overcome undersampling

Limitations

- Best suited for *global* distance field computation in complex environments
  - Culling involves occlusion query overhead
  - Clamping bounds depend on distribution of sites
- Computes distance field on uniform grid
  - Size limited by GPU memory
- Application may require distance field readback to CPU

Future Work

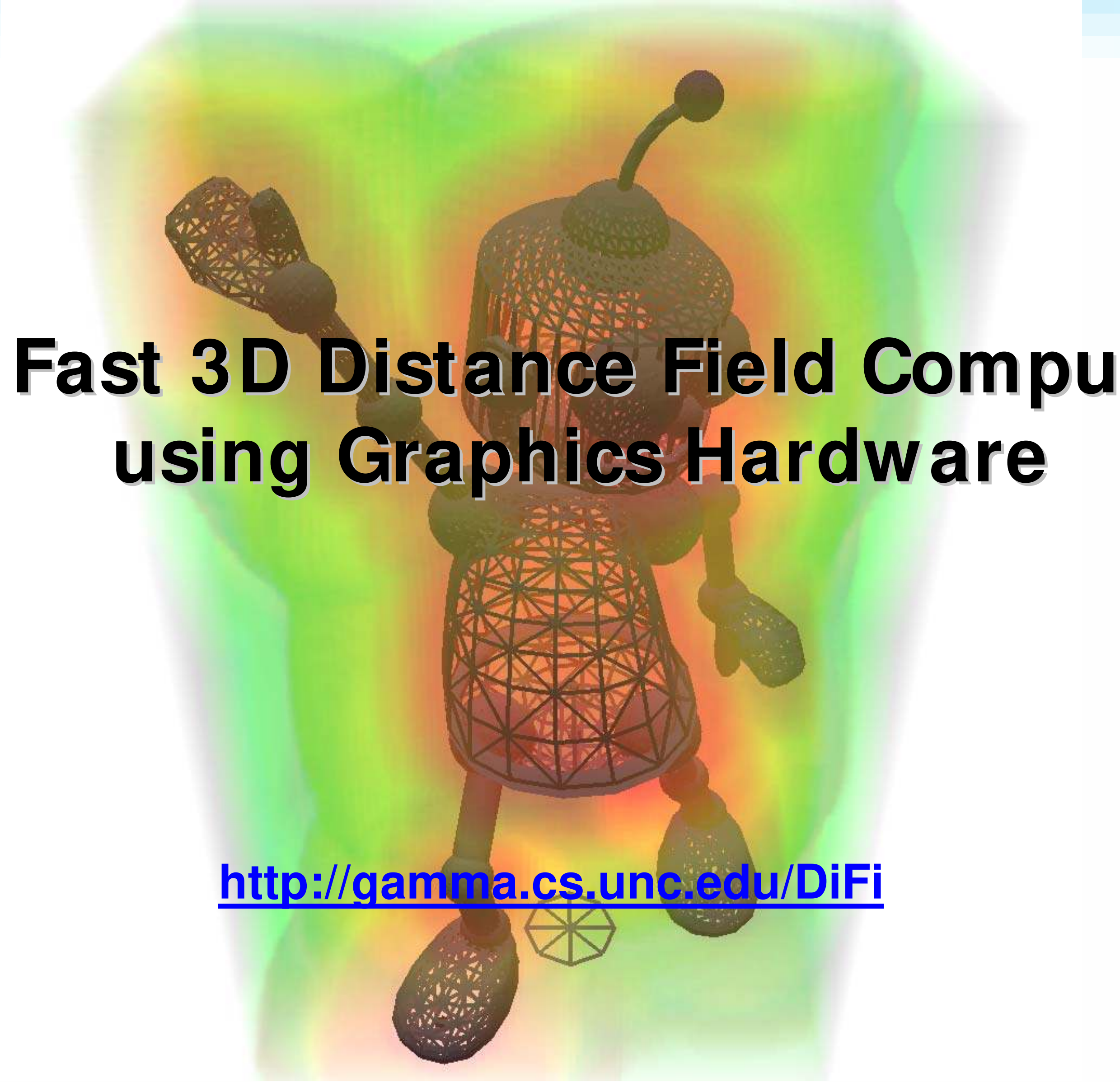
- Efficient clamping for manifold sites [Sigg03]
- Explore temporal coherence for dynamic and deformable models
- Extend to k-th order Voronoi diagrams
- Further applications like dynamic simulation, morphing and database queries

Acknowledgments

- ARO
- NSF
- ONR
- DARPA
- Intel Corporation
- Anonymous reviewers
- Ming Lin
- Mark Foskey, Luv Kohli
- Mark Harris, Greg Coombe, Naga Govindaraju
- UNC GAMMA group

Questions?

Questions?



DiFi: Fast 3D Distance Field Computation using Graphics Hardware

http://gamma.cs.unc.edu/DiFi