GPU Algorithms for NURBS Minimum Distance and Clearance Computations

Adarsh Krishnamurthy, Sara McMains
University of California, Berkeley

GPU-Algorithm Development

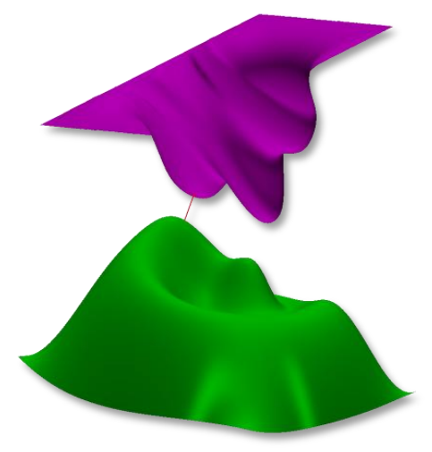
Challenges

- GPU/CPU hybrid operations
  - Distribution of work between CPU and GPU
  - Some operations are inherently serial
- GPU restrictions
  - Restrictions on dynamic loops
  - Restrictions on texture memory writes
- GPU performance guidelines
  - Coherent memory reads
  - Branchless kernels
  - Reduced data read-back from GPU
- Multiple GPU vendors
  - Algorithms should run on any massively parallel architecture
  - Should be easy to port to many-core architecture

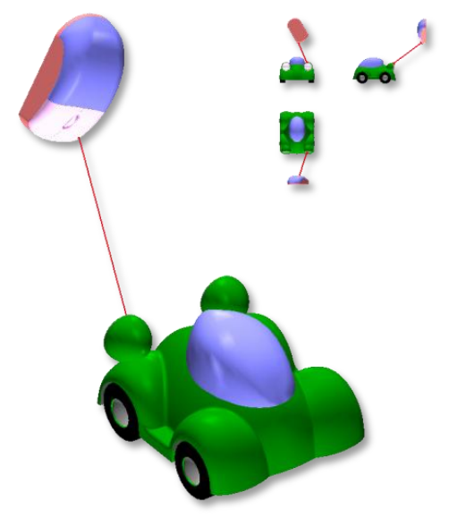
Strategies

- Separation of CPU/GPU operations
  - Example: NURBS evaluations

- Imposing artificial structure to the computations
  - Example: Surface minimum distance computations

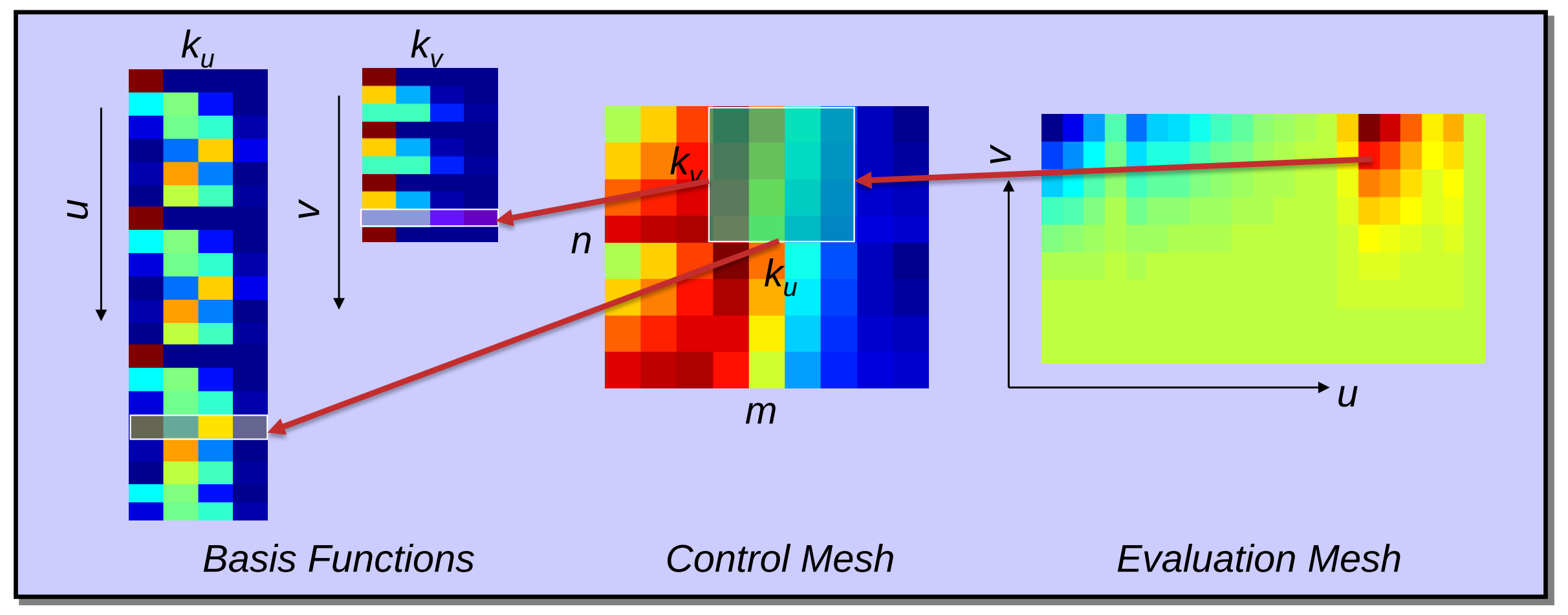


- Separating problem into multiple stages
  - Example: Object clearance computations

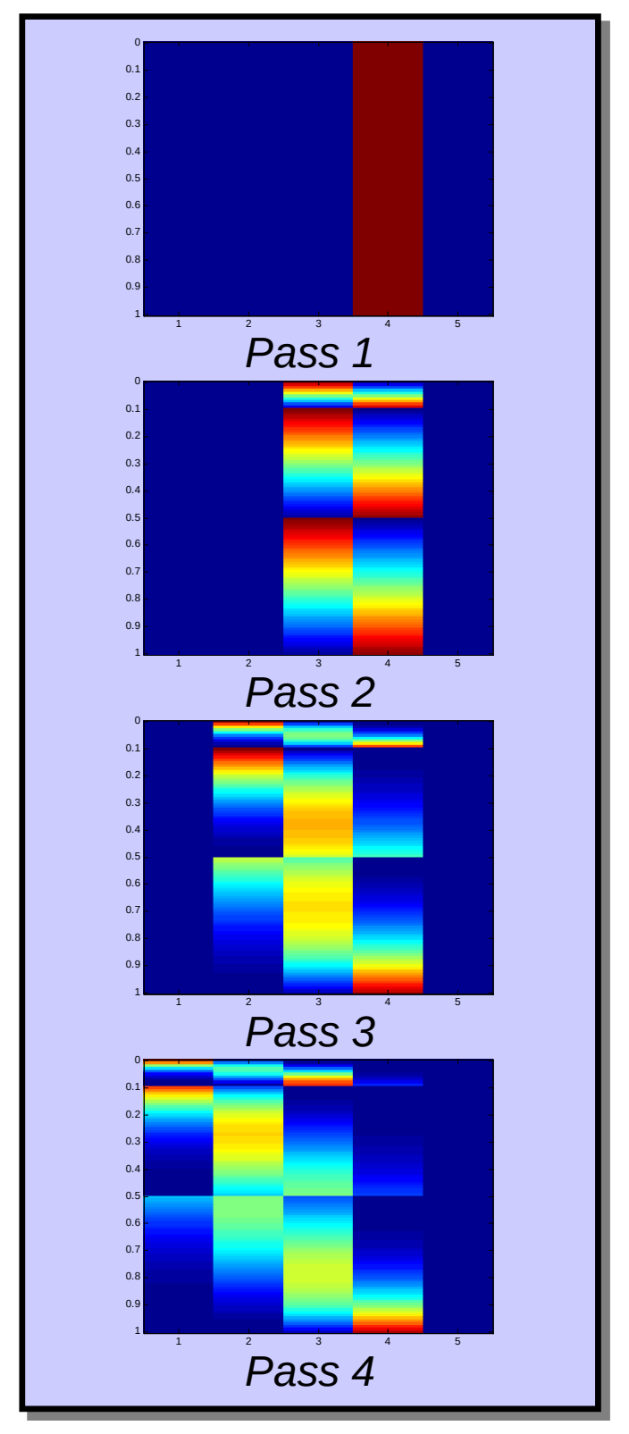


NURBS Evaluation

- Two step operation
  - Evaluate basis function values in multiple passes
  - Multiply basis function values with control points to get surface coordinates
- Strategies
  - Ping-pong technique to overcome dynamic loop limitation of older GPUs
  - Perform loop operations on the CPU and perform the only the multiplication operation in the GPU kernel



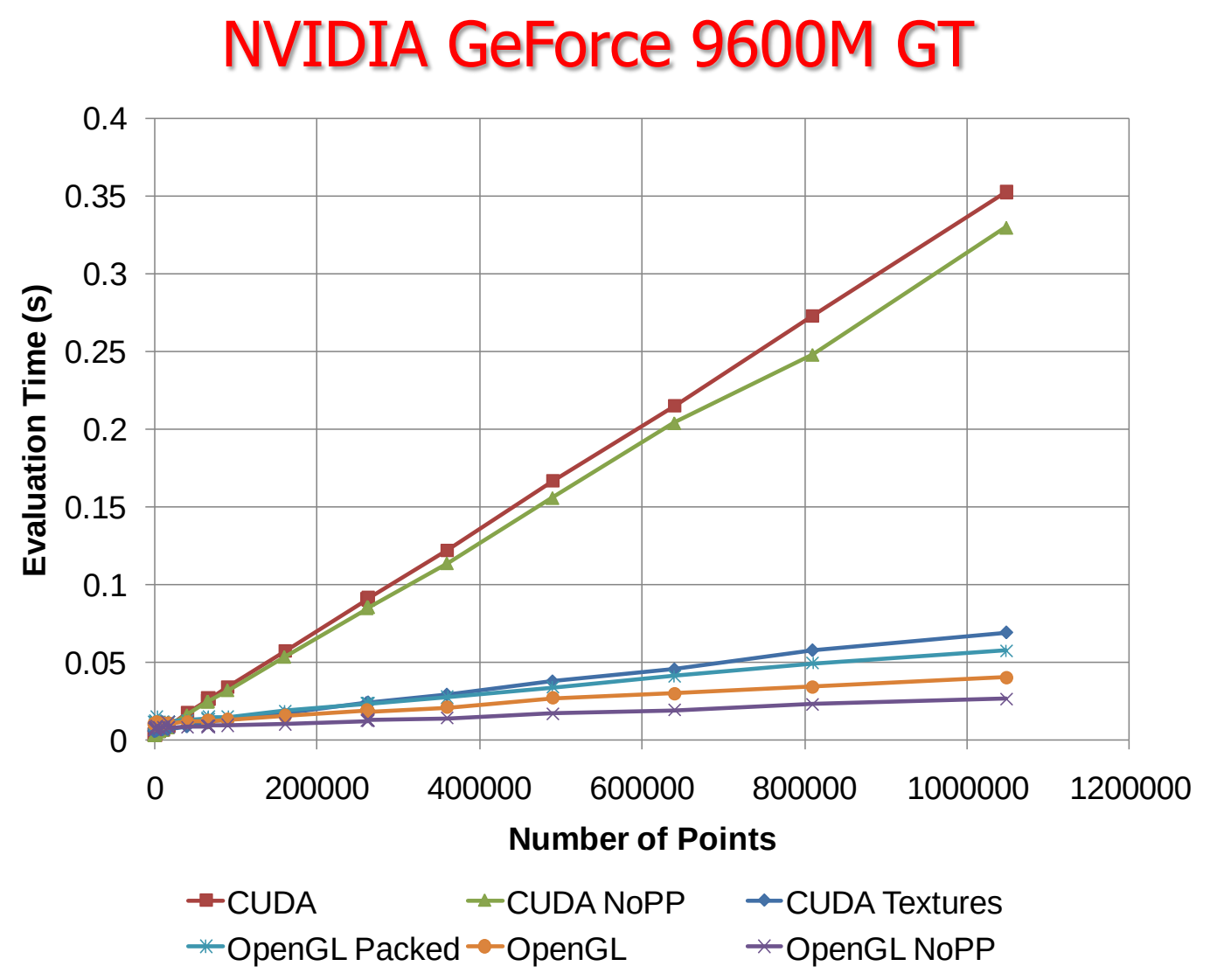
GPU NURBS Evaluation Algorithm

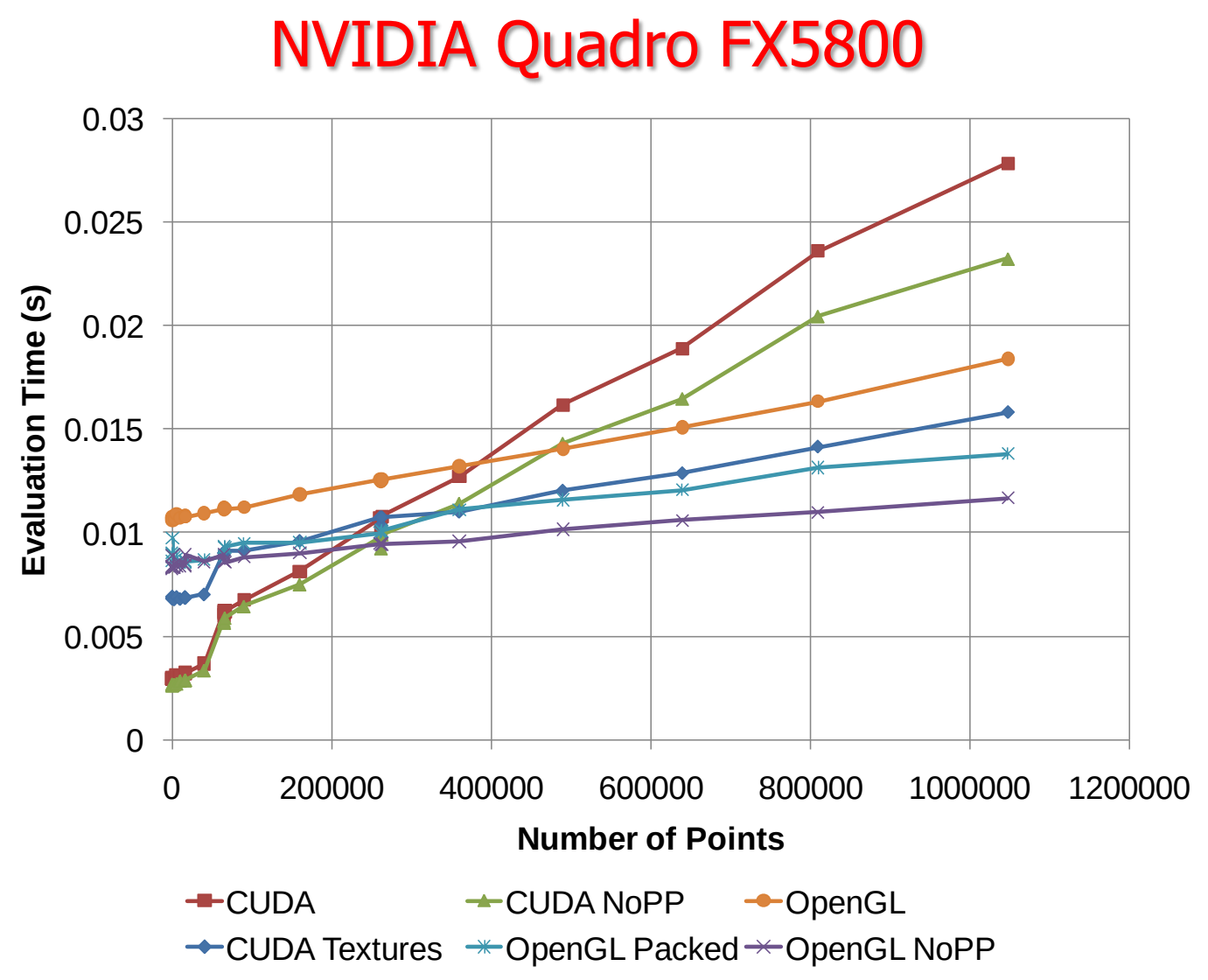


Intermediate output while computing basis function

NURBS Evaluation Results

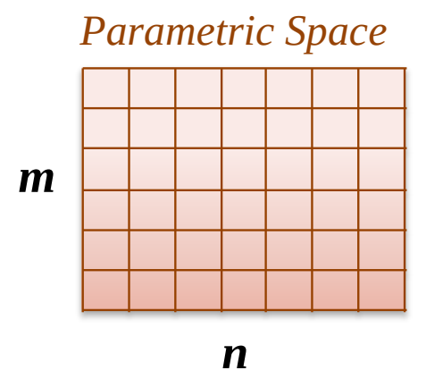
- Graphs comparing NURBS evaluation times for different implementations on two different graphics cards
- Implementations tested
  - CUDA
  - CUDA with texture memory (CUDA Textures)
  - CUDA without ping-pong (CUDA NoPP)
  - GPGPU (OpenGL)
  - GPGPU with texture packing (OpenGL Packed)
  - GPGPU without ping-pong (OpenGL NoPP)





Bounding Boxes for NURBS Surfaces

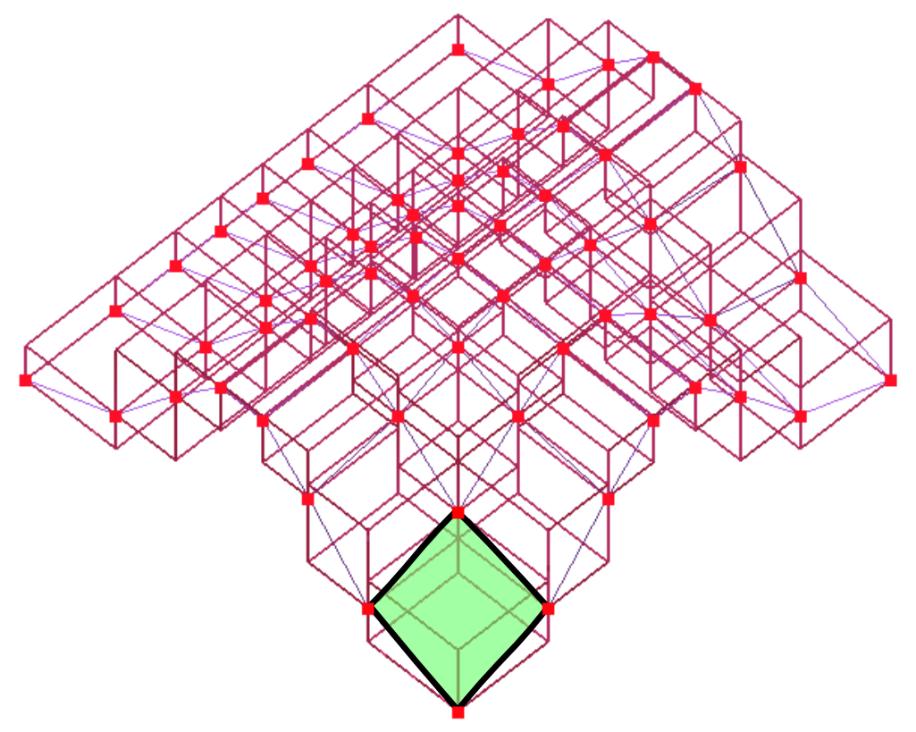
- Build bounding-boxes for NURBS surfaces to accelerate geometric operations
  - Use grid of points on surface already evaluated
  - Find min, max x, y, & z coordinate of four adjacent points and build the basic bounding-box
  - Find the maximum possible deviation, *K* of the actual surface from a piecewise-linear approximation
  - Expand the size of the bounding-box by *K*, which will guarantee that the bounding-box contains the surface

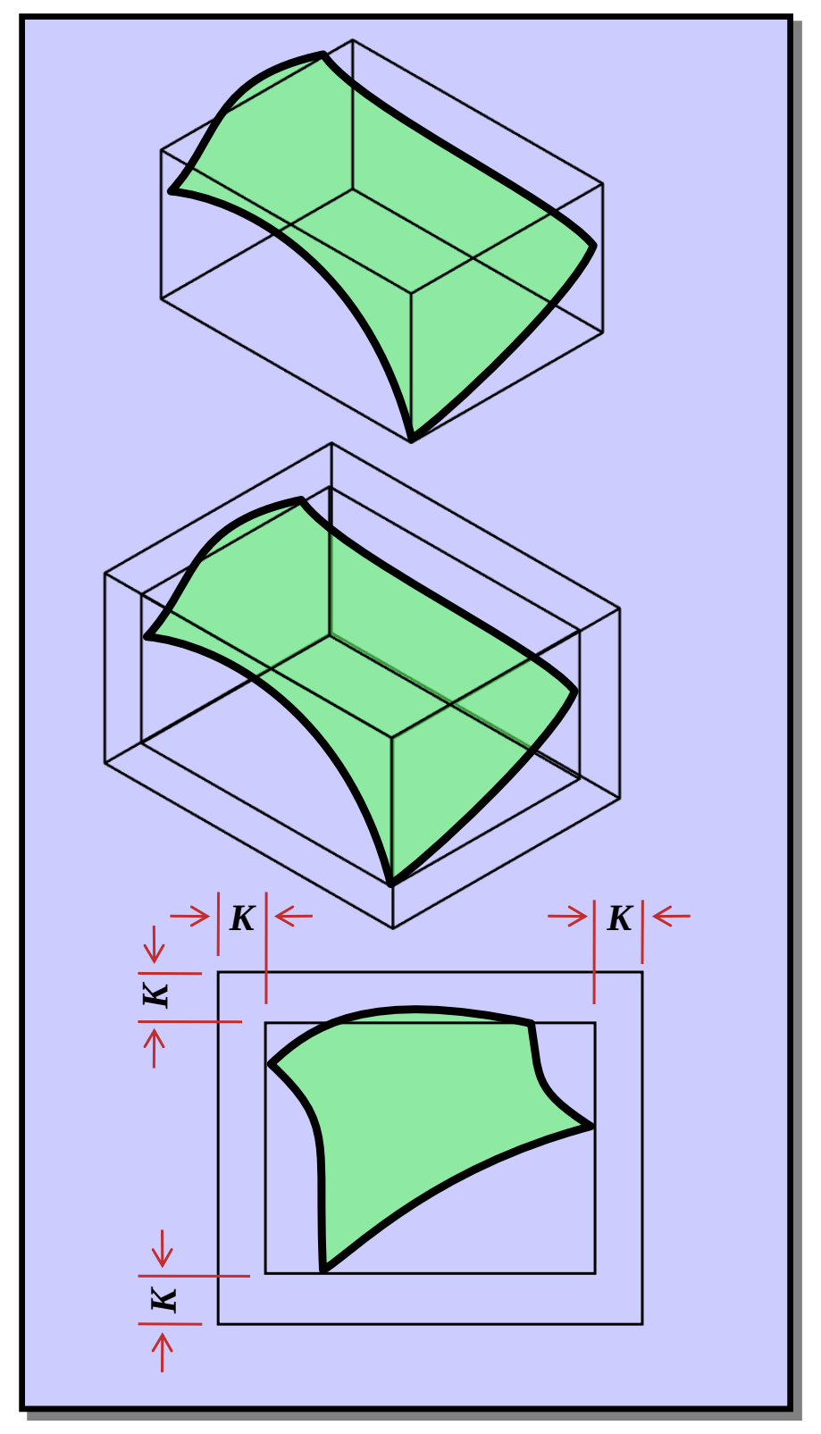


$$M_1 = \mathrm{Max}(\partial^2 S/\partial u^2)$$
$$M_2 = \mathrm{Max}(\partial^2 S/\partial u\,\partial v)$$
$$M_3 = \mathrm{Max}(\partial^2 S/\partial v^2)$$

$$K = \frac{1}{8}\left(\frac{1}{n^2}M_1 + \frac{2}{nm}M_2 + \frac{1}{m^2}M_3\right)$$

[Filip et al. 1986]

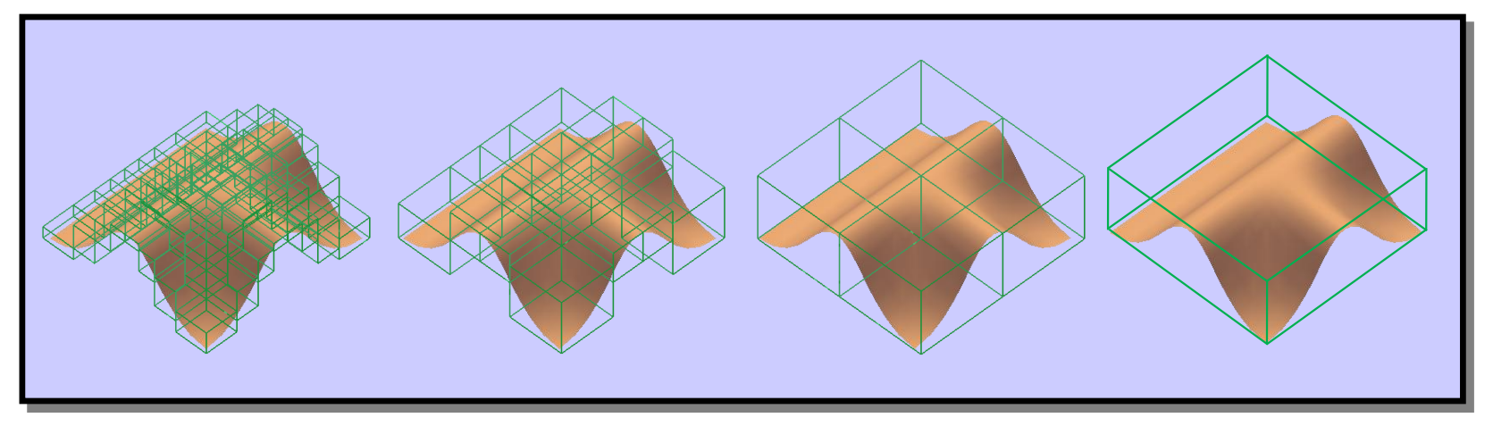




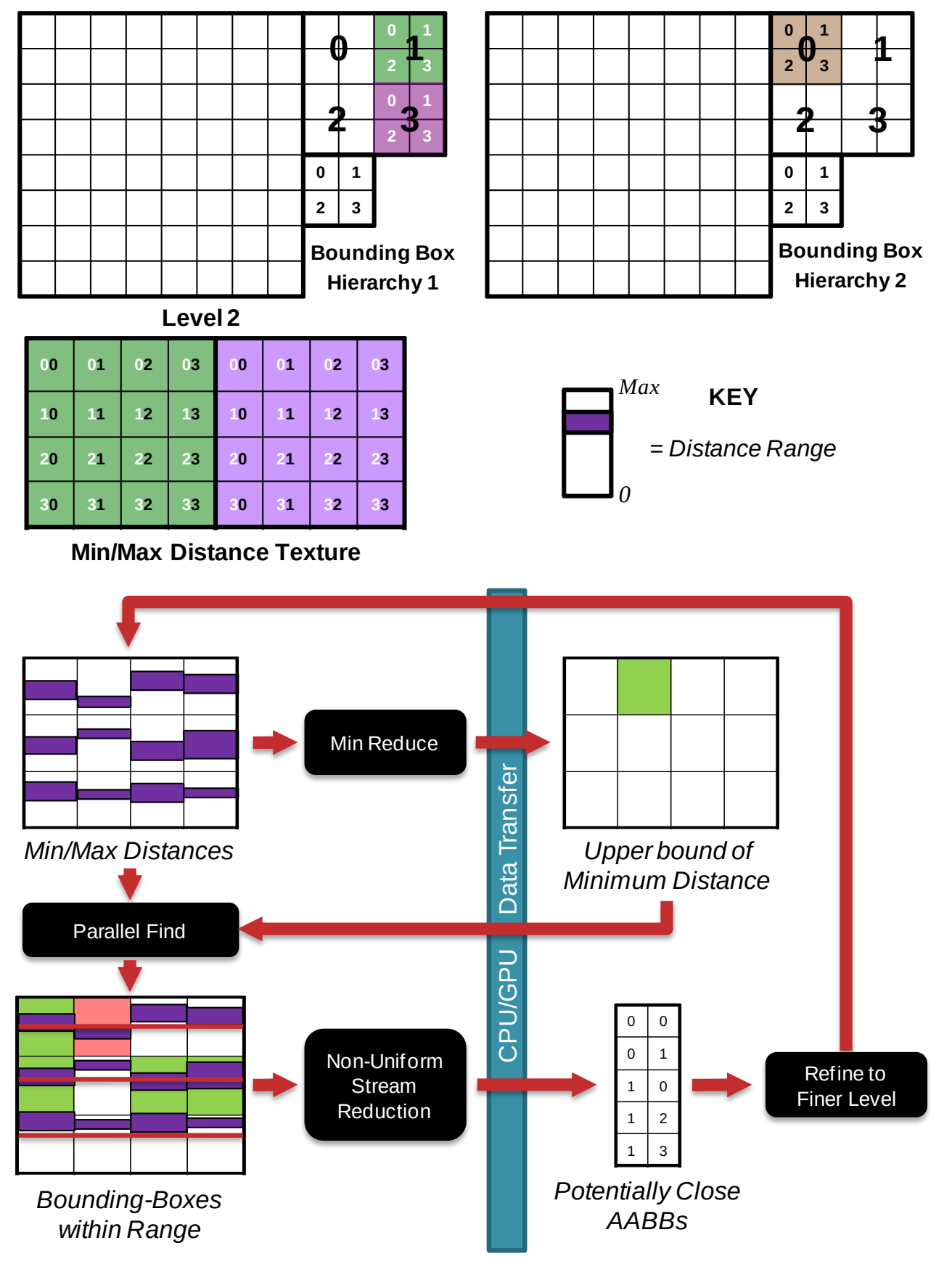
Steps in creating bounding-boxes for NURBS surfaces

Surface Minimum Distance Computations

- GPU acceleration strategy
  - Construct bounding box hierarchies
  - Map sets of 4 boxes from each surface at each level of hierarchy to a min/max distance texture
  - Find set of potentially-close bounding-boxes at this level
  - Refine to the finer level
  - GPU acceleration effective when number of potentially-close bounding-boxes increase at lower levels
- Minimum distance computation
  - Linearize the surface patches inside the finest level of potentially-close bounding-boxes with triangles
  - Find the minimum distance between these triangles

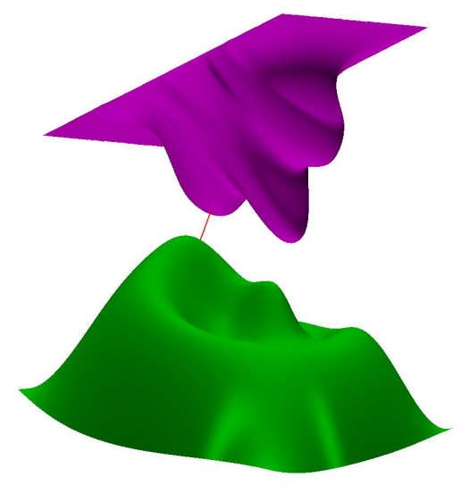


Above: Bounding –Box hierarchy used in the minimum distance computation algorithm.
Right: Minimum distance computation algorithm overview

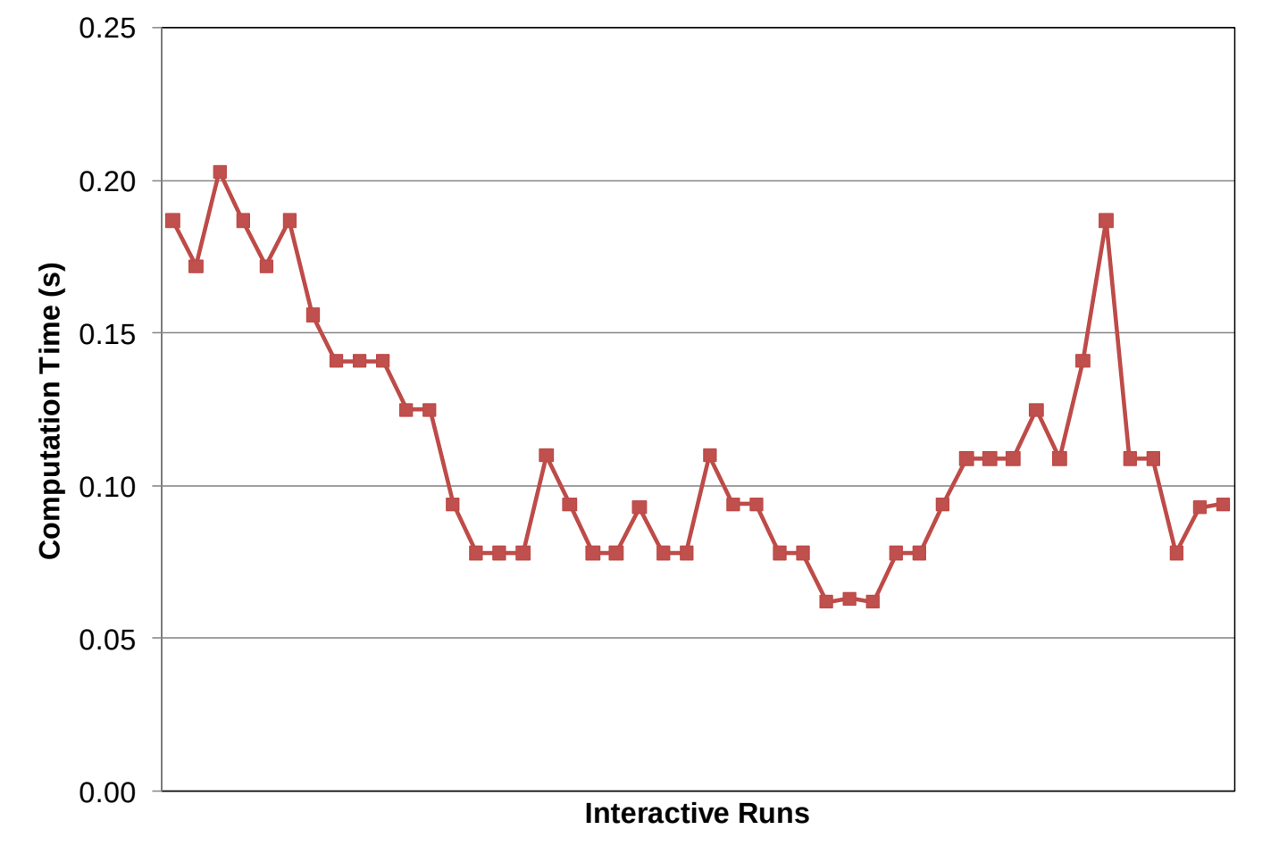


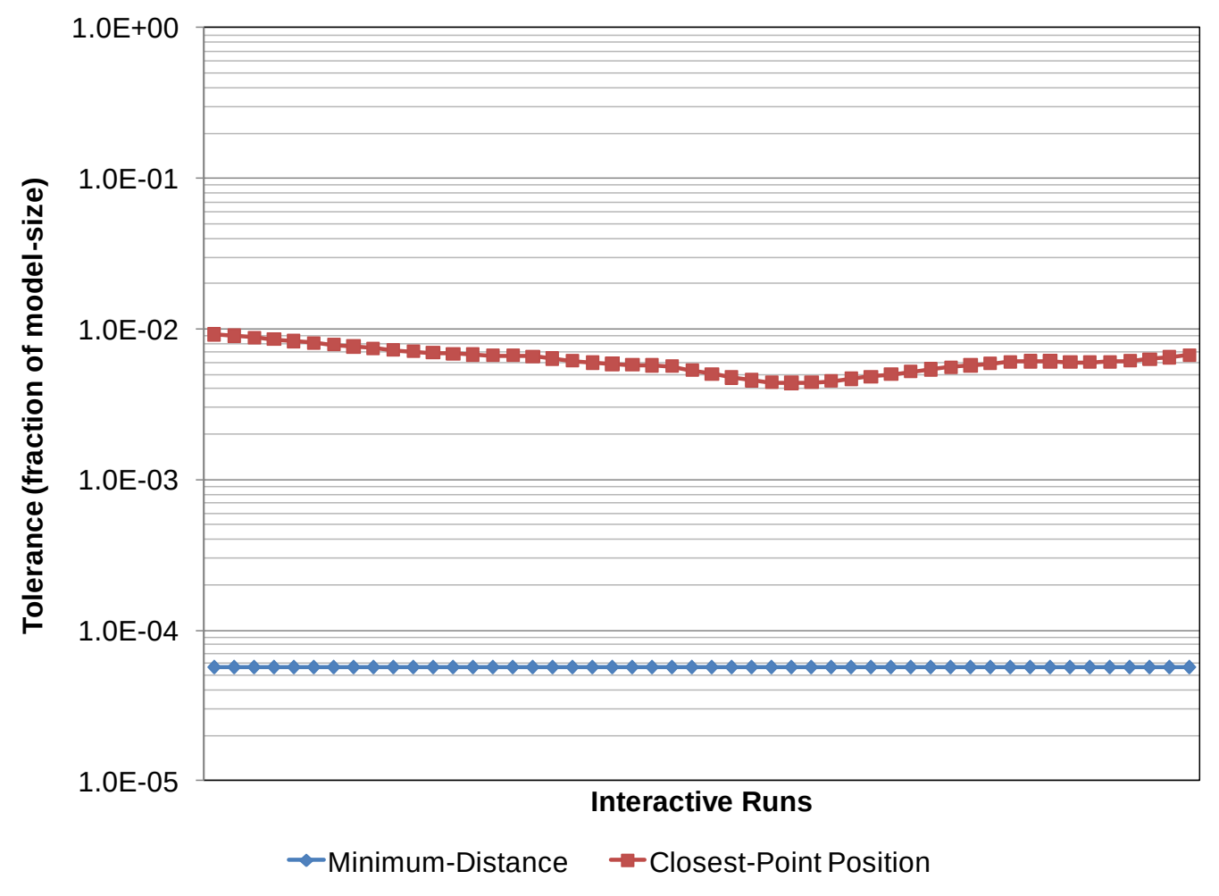
Timing Results

- Interactive minimum distance computations
  - Using NVIDIA Quadro FX5800
  - One surface interactively rotated and translated with respect to a fixed surface
  - Average computation time : 0.11s
  - Tolerance bounds were < 0.01 for all computations



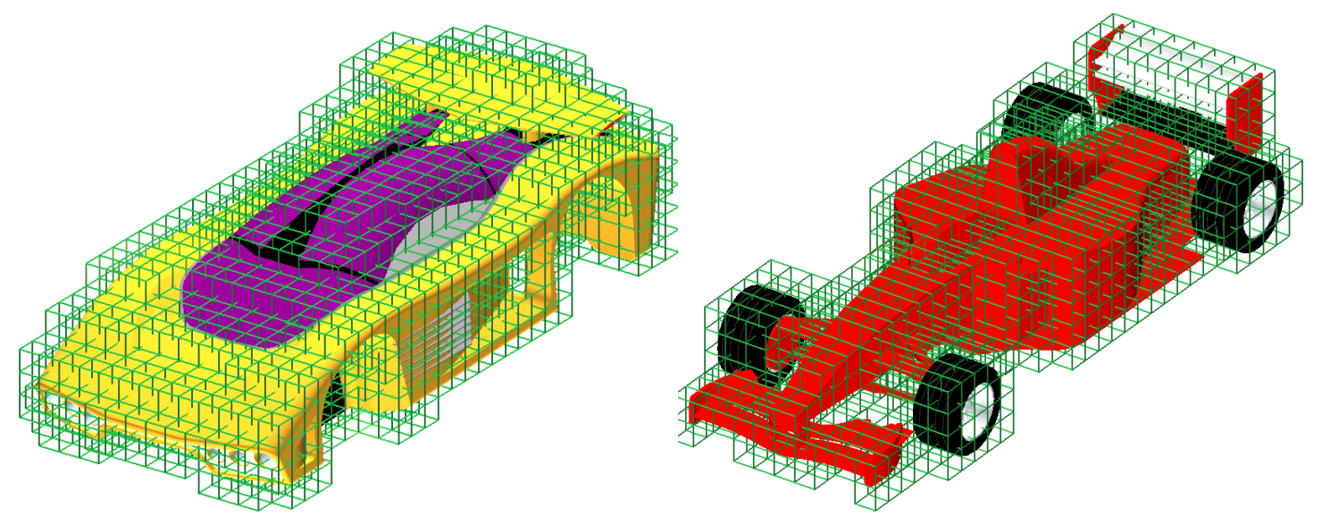
NURBS Surfaces used for timing

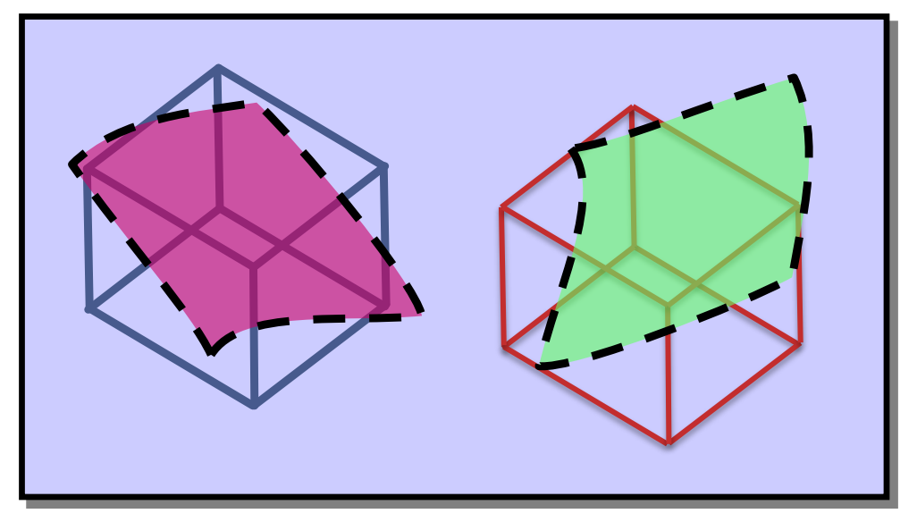




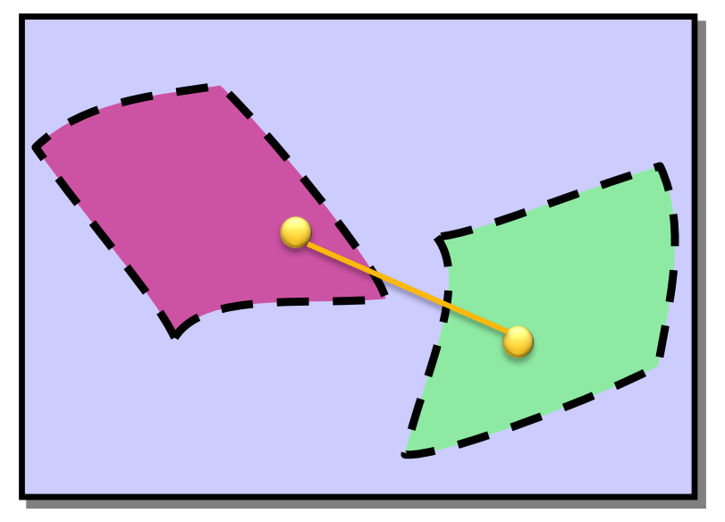
Object Clearance Computations

- Two-stage computations
- Voxel-based first stage
  - Voxelize the object using the coarse tessellation used for display
  - Find list of potentially-close voxel pairs
- Surface-based second stage
  - Create a list of potentially-close surfaces that intersect the potentially-close voxel pairs
  - Compute minimum distance between the surfaces

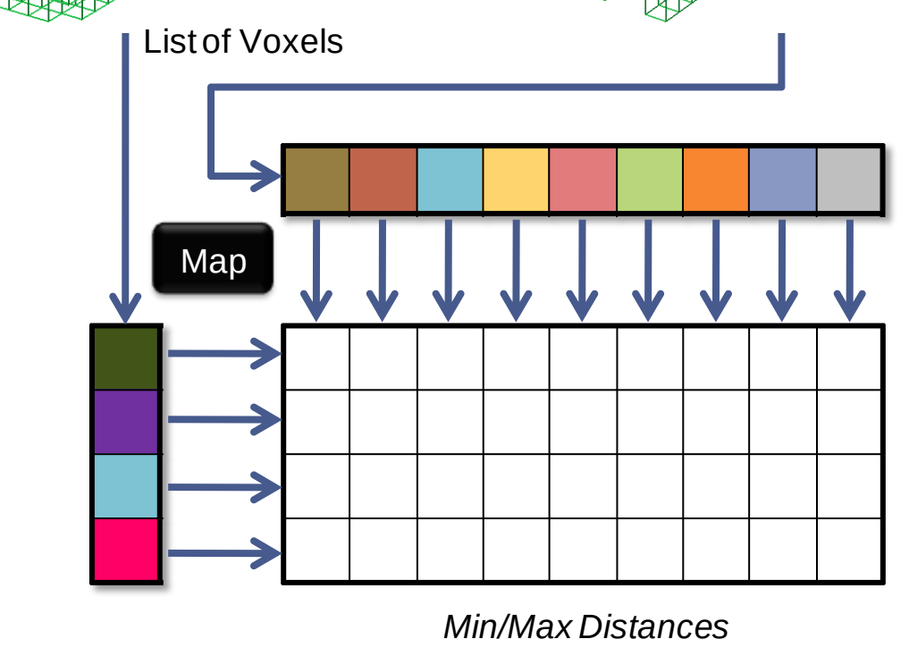




Surfaces intersecting potentially-close voxel pairs

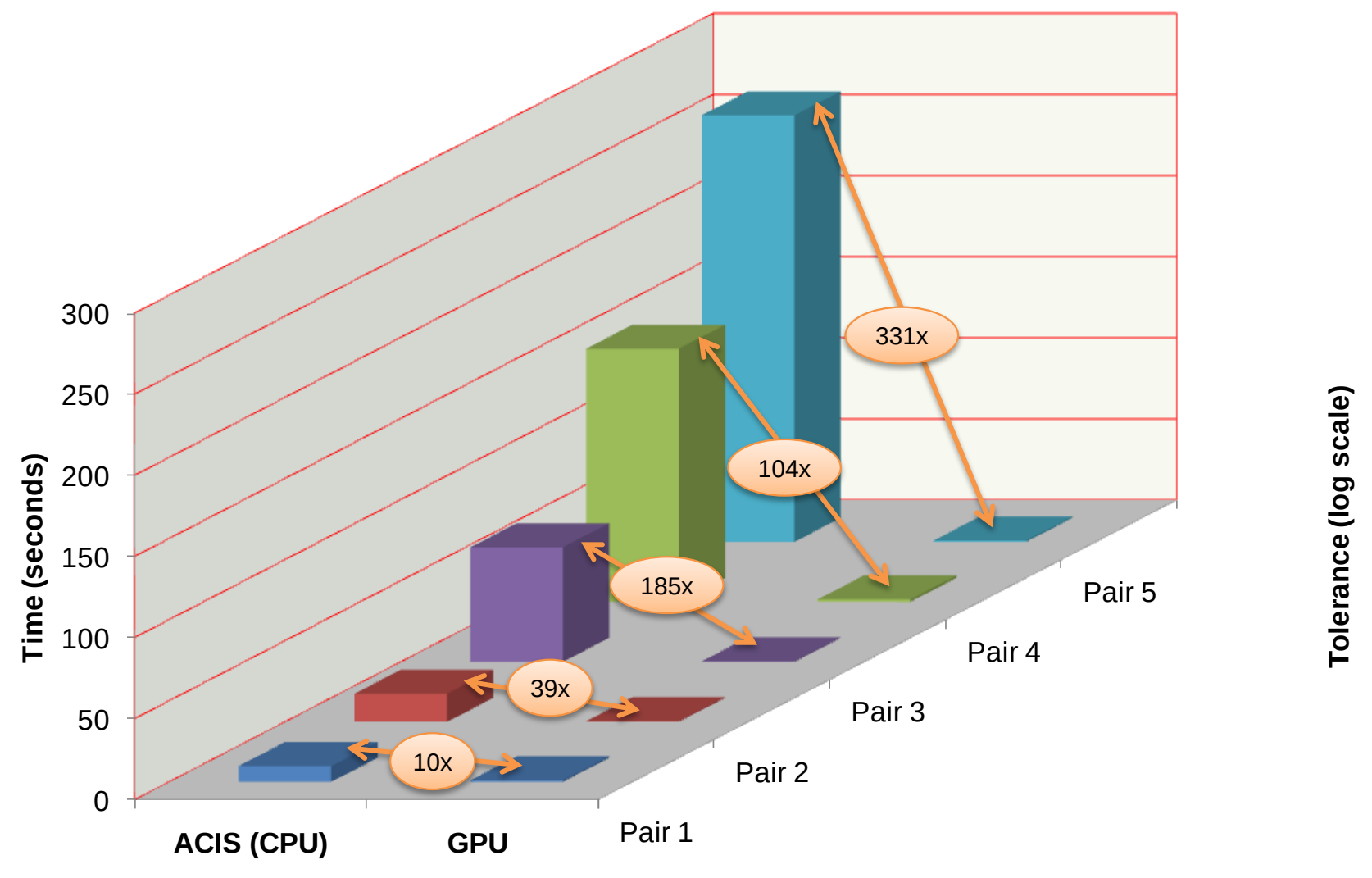


Surface minimum distance computation

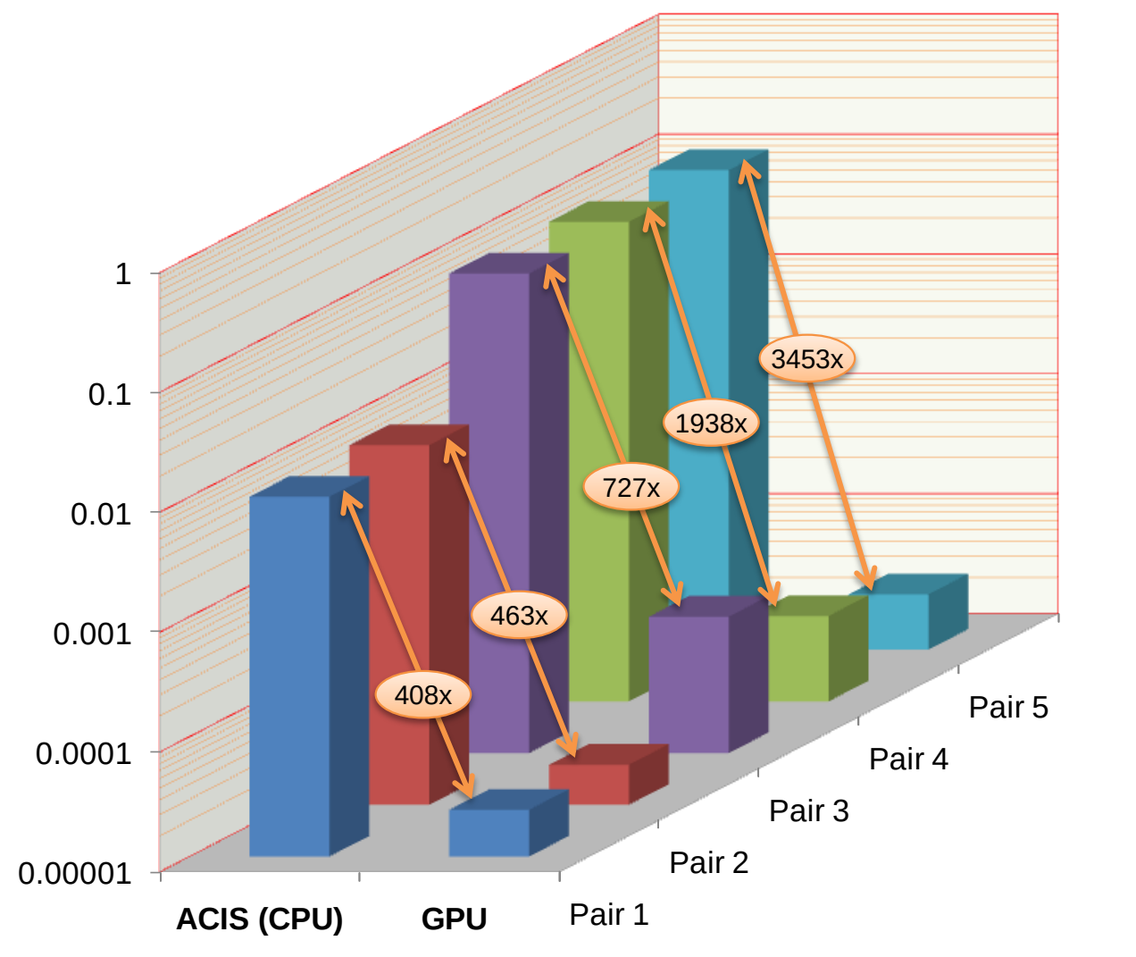


Map voxels to rows and columns of a Min/Max distance texture

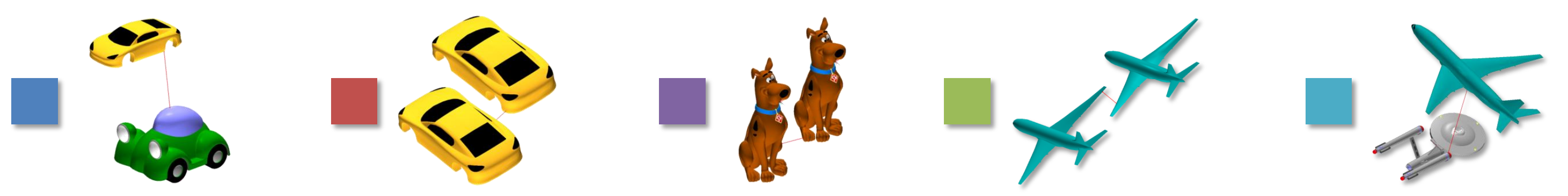
Object Clearance Results



Computation Time



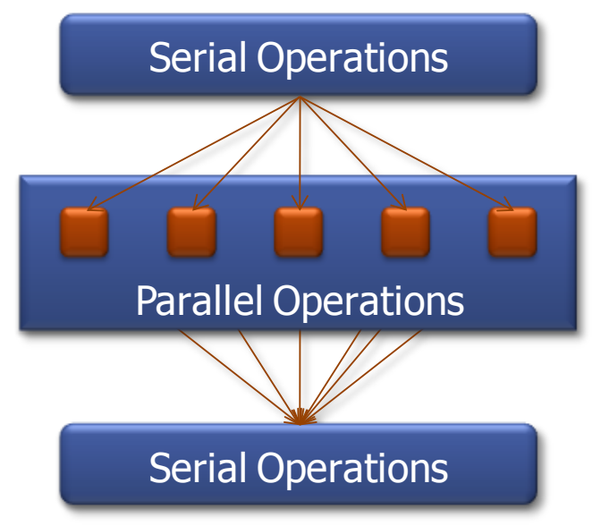
Computed Tolerance Bounds

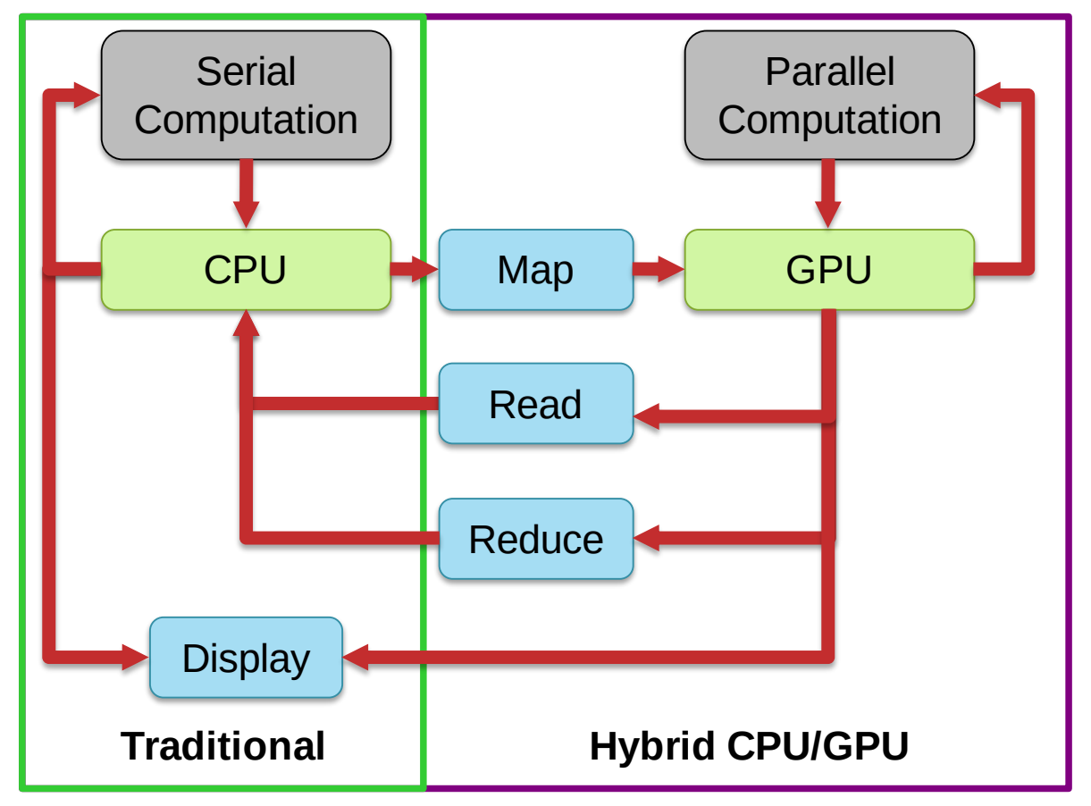


Conclusions

GPU Programming Insights

- Dramatic performance gains
  - Frequently orders of magnitude improvement
  - But requires GPU-optimized algorithms
- Hybrid CPU/GPU algorithms
  - Some parts of algorithms are inherently serial
  - Use CPU in such cases for better work-load balancing
- Guaranteed user-specified tolerances
  - Enables direct adoption of GPU algorithms in CAD
- GPU framework
  - Reduces development time for new algorithms
  - Helps in performance tuning and optimization





Proposed GPU framework for CAD algorithms

Computer-Aided Design and Manufacturing Laboratory, Mechanical Engineering Department, University of California, Berkeley