Heterogeneous Spline Surface Intersections

Sverre Briseid (sbr@sintef.no)

Trond Hagen (trr@sintef.no)

Geometric and Physical Modeling 2009, San Francisco

Outline

- Heterogeneous architectures
- OpenMP & CUDA
- Spline surface intersection/self-intersection
- Multi-core approach
- Results
- Conclusions

Heterogeneous architectures

- More than one type of architecture in a system
  - Multi-core CPU
  - Specialized accelerated cores
- Different programming models
- Sequential algorithms a bottleneck
- Split problem into independent task, run in parallel
- Utilize the strengths of the different architectures

Multi-core CPU & OpenMP

- 2-4 cores in modern desktop computers
- Requires parallel algorithms
- OpenMP
  - API for shared memory parallel programming
  - C/C++
  - Compiler pragmas
  - Easy syntax

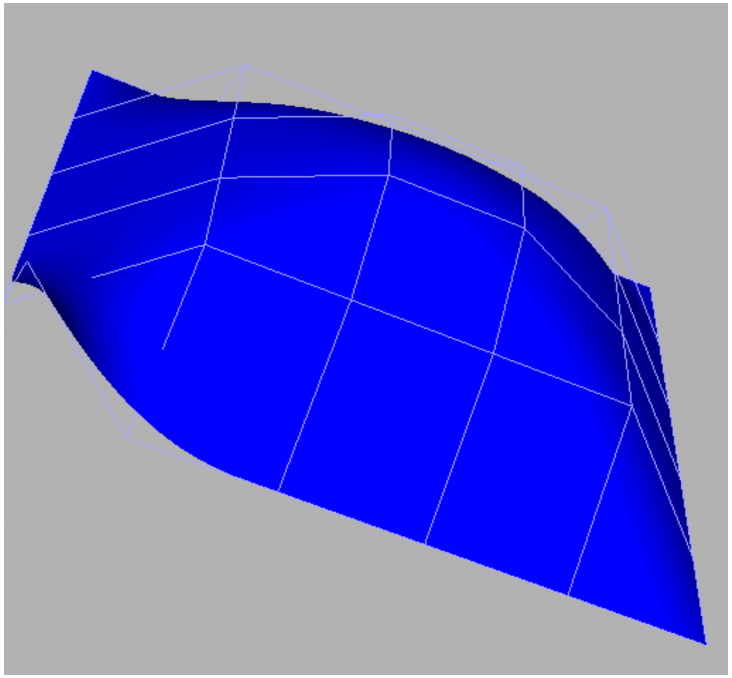
```
int i, m=10, N=1000
double A[N], B[N], C[N];
#pragma omp parallel for
for (i=0; i<N; i++) {
A[i] = B[i] + m*C[i];
}
```

GPU & CUDA

- GPU (Graphics Processing Unit)
  - All modern computers has one
  - Massively parallel – Up to 500 cores
  - Computational power: Up to 2 teraflops
  - 32-bit precision at full speed – 64-bit precision at half speed
- CUDA
  - API for using NVIDIA graphics cards
  - GPU computing for the masses
  - Syntax based on C/C++
  - Computational kernels

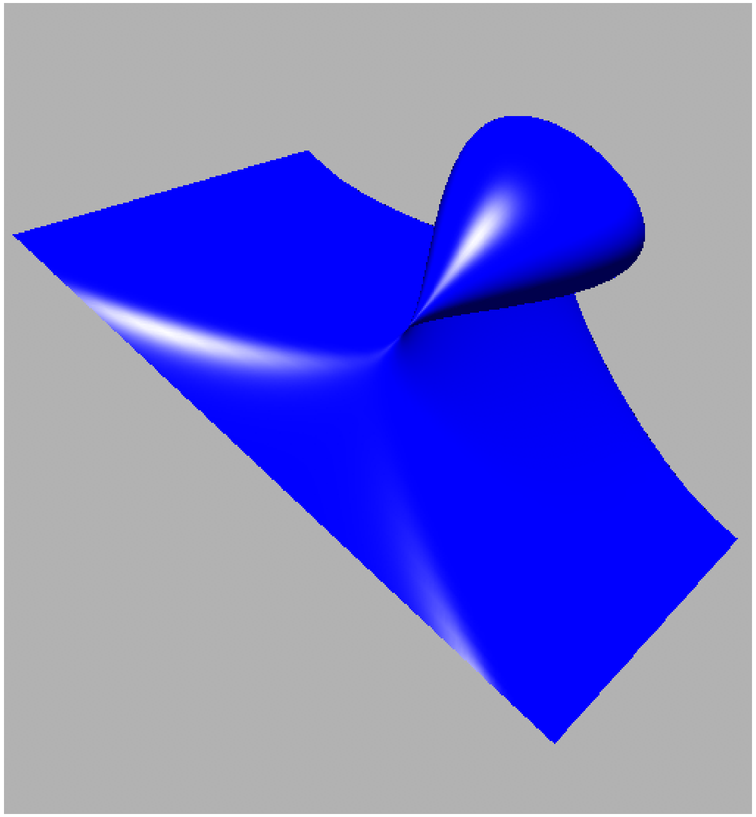
Spline surface

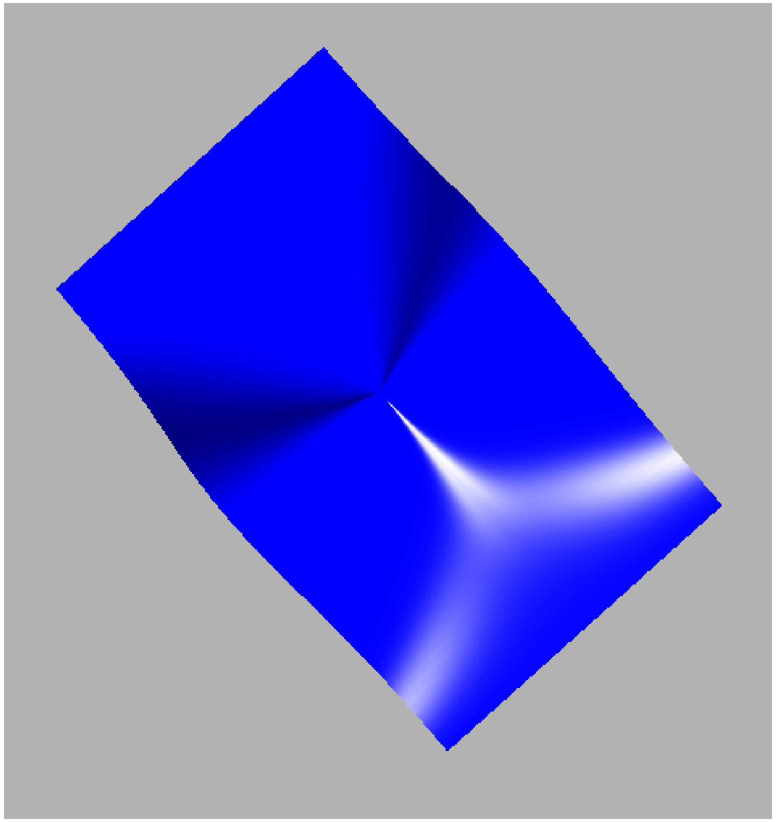
- Parametric
- Controlled by a regular polygon mesh



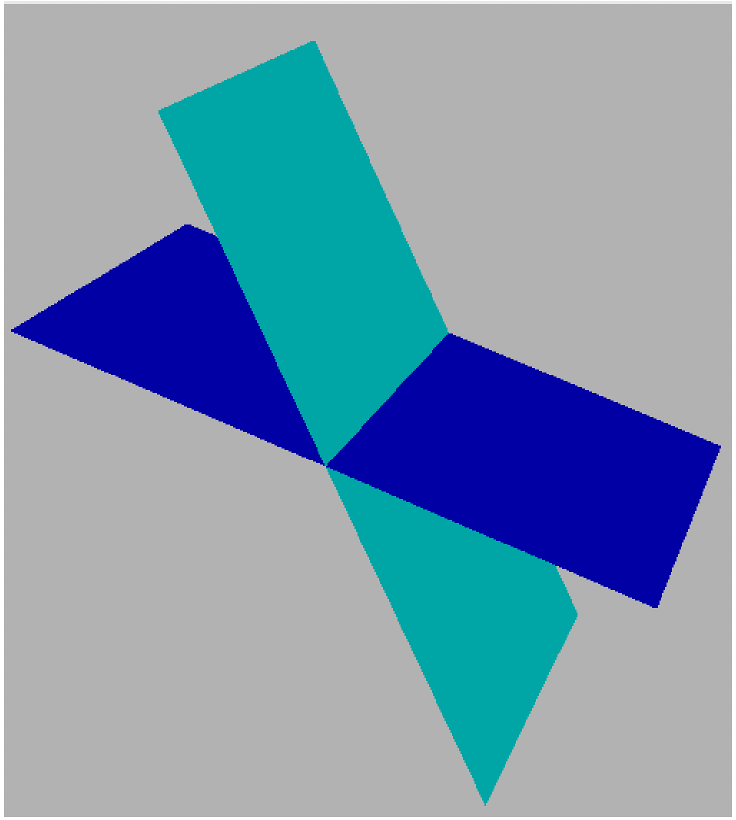
$$S(u,v) = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \mathbf{c_{i,j}} B_{i,d}(u) B_{j,d}(v)$$

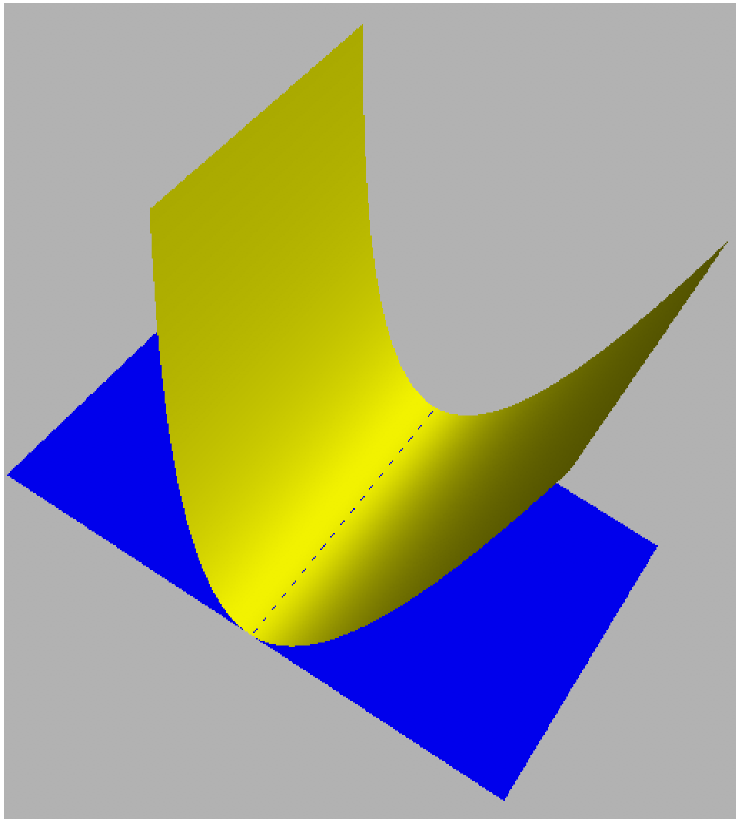
Self-intersection - Singularities





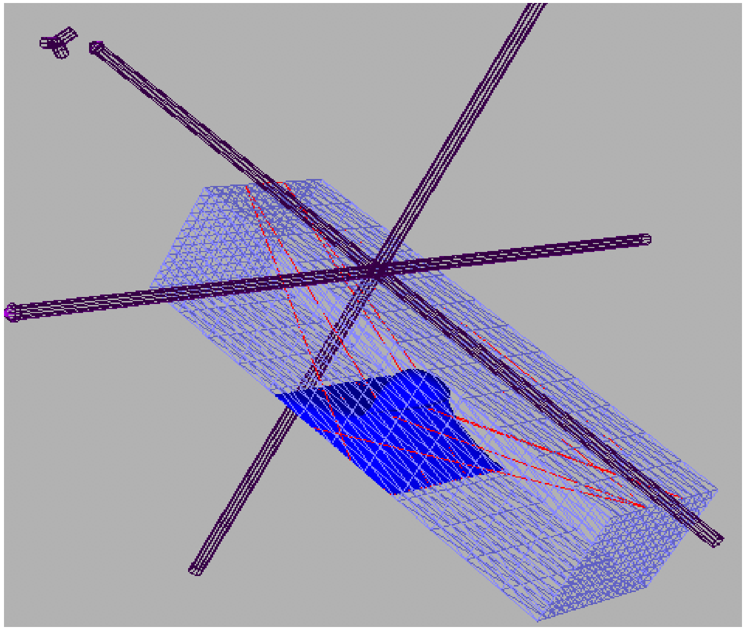
Transversal and tangential intersections

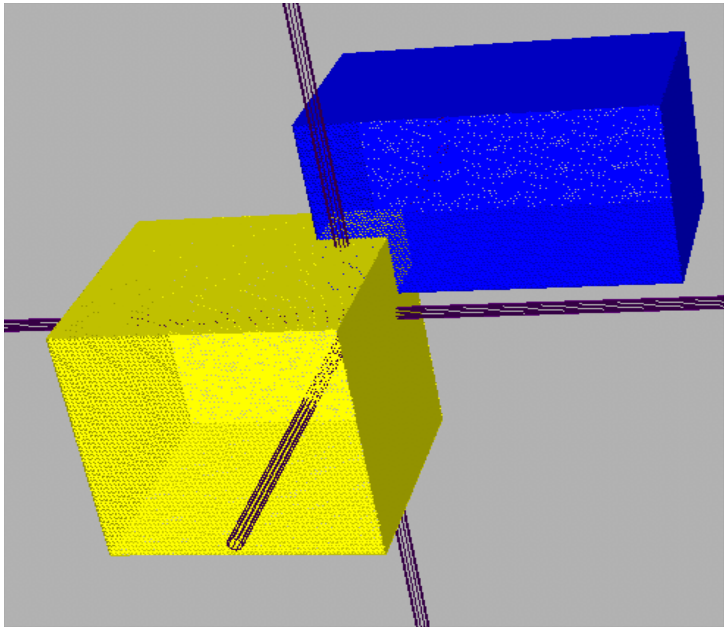




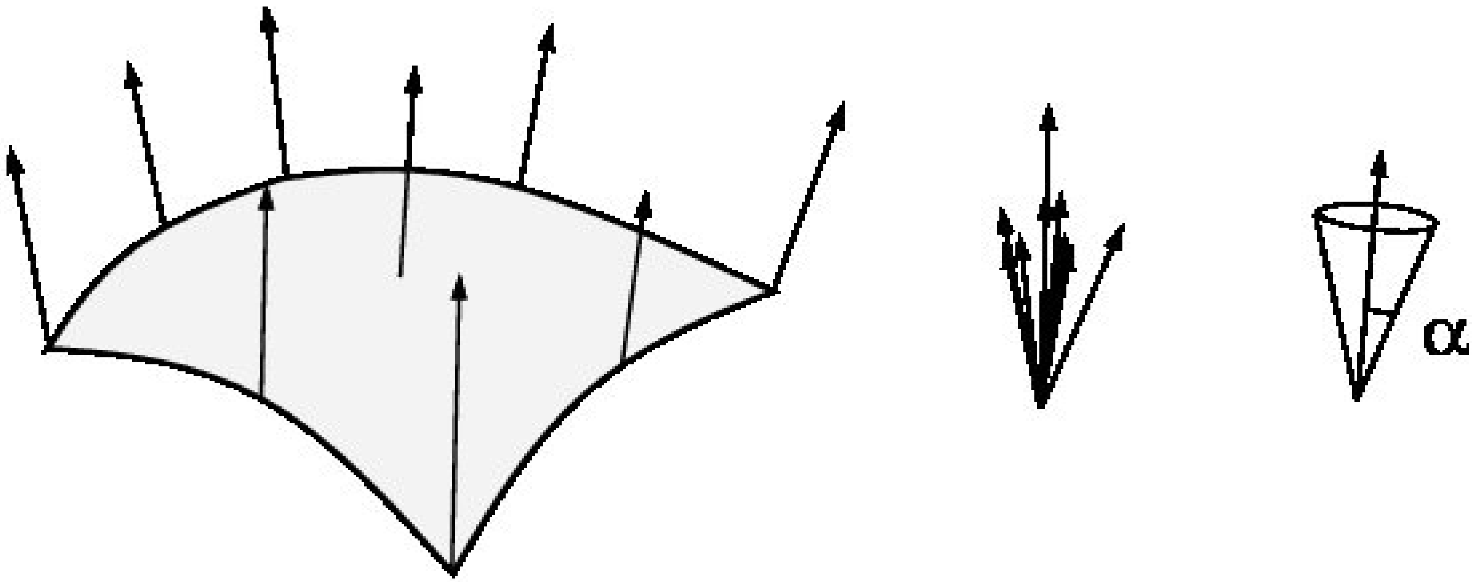
Multi-core approach

- Zoom in on problematic areas using parallel resources, analyze
- Let the CPU trace out the intersection curves in a sequential manner
- Overlap-test
  - Massive uniform subdivision down to Bezier level. Level n => 2^n Bezier segments in each parameter direction
  - Create axis aligned bounding boxes
  - Box-box overlap-test





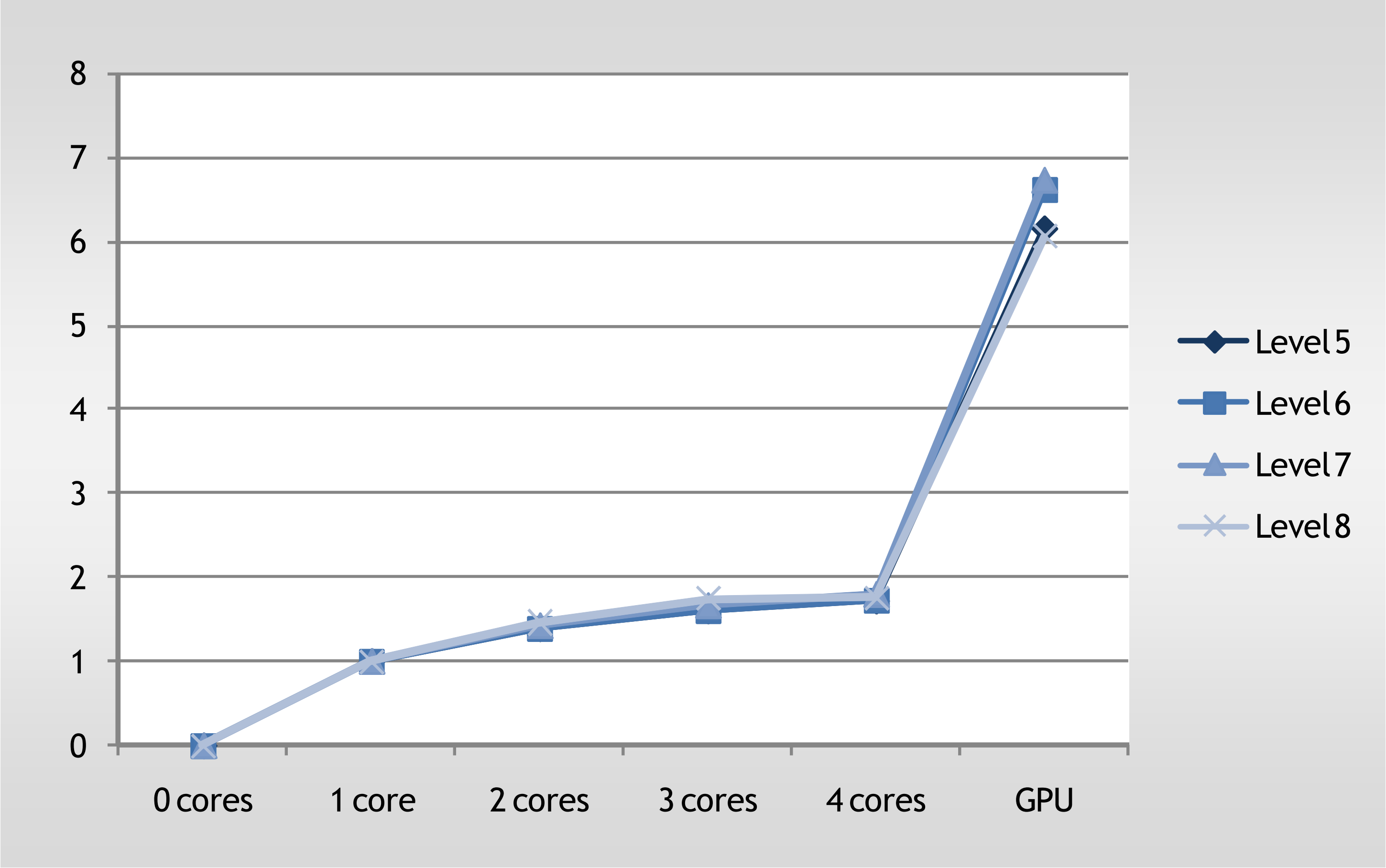
- Intersection-analysis
    - Subdivide normal surface to the same level as the surface
    - Check if sub-patches contain the origin
    - Create direction cones for the bezier normal patches
    - Check if cone span is less than pi => no self-intersection
    - Check all pairs of normal cones whether they overlap => possibly a tangential intersection, given that bounding boxes overlap



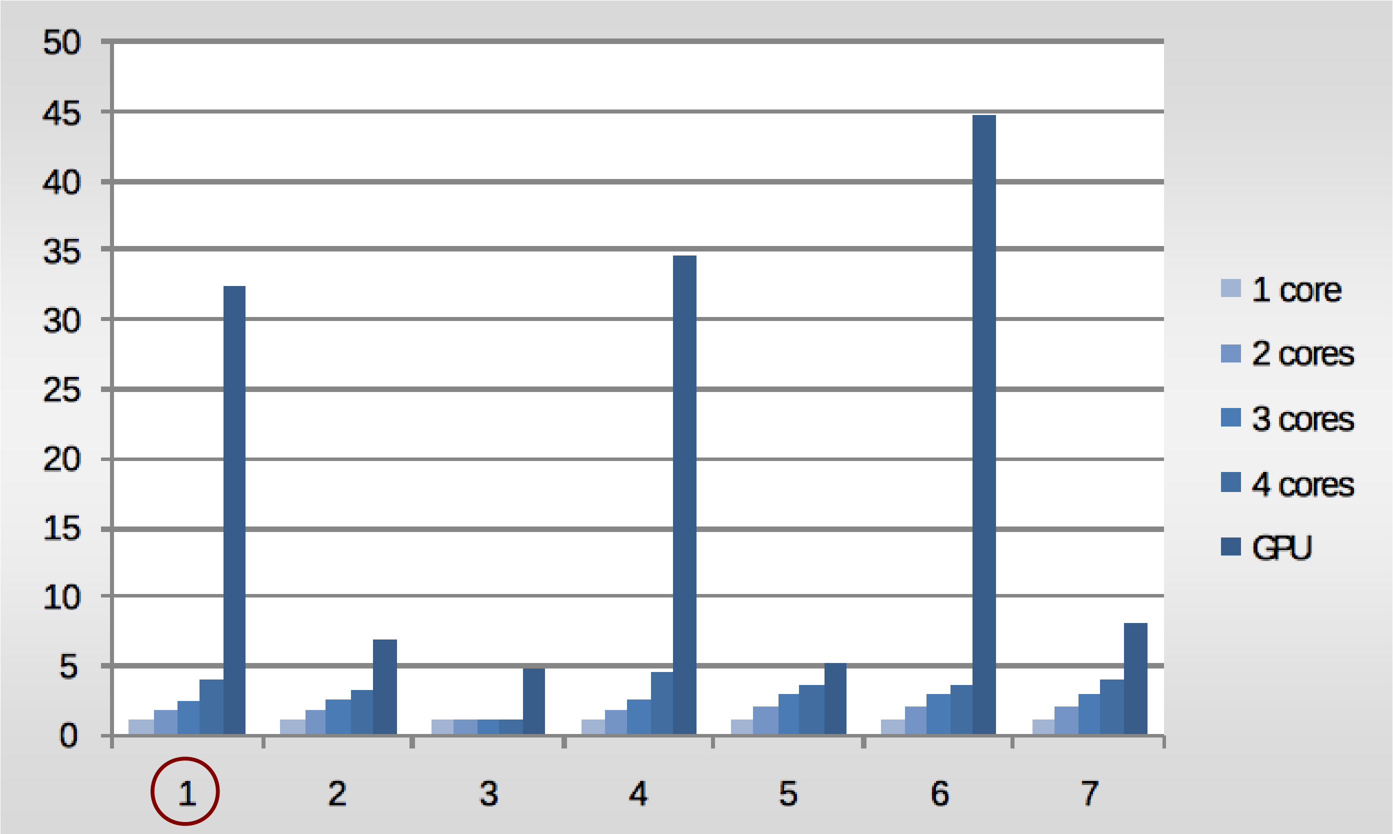
The intersection-test modules

1. Spline surface refinement
   - Localize the possible intersections
2. Bounding box generation
   - Axis aligned boxes containing the Bezier subpatches
3. Box-box overlap-test
   - See if two Bezier subpatches may overlap
4. Normal surface refinement
   - Refine to the same level as the spline surface
5. Degeneracy-test
   - Check if bounding boxes of refined normal surface contain the origin
6. Normal cone generation
   - Compute the span of the normals for each Bezier subpatch
7. Cone-cone overlap-test
   - Check if we may have a tangential intersection

Speedups for subdivision levels 5-8 Input: Cubic Bezier surface & corresponding quintic normal surface

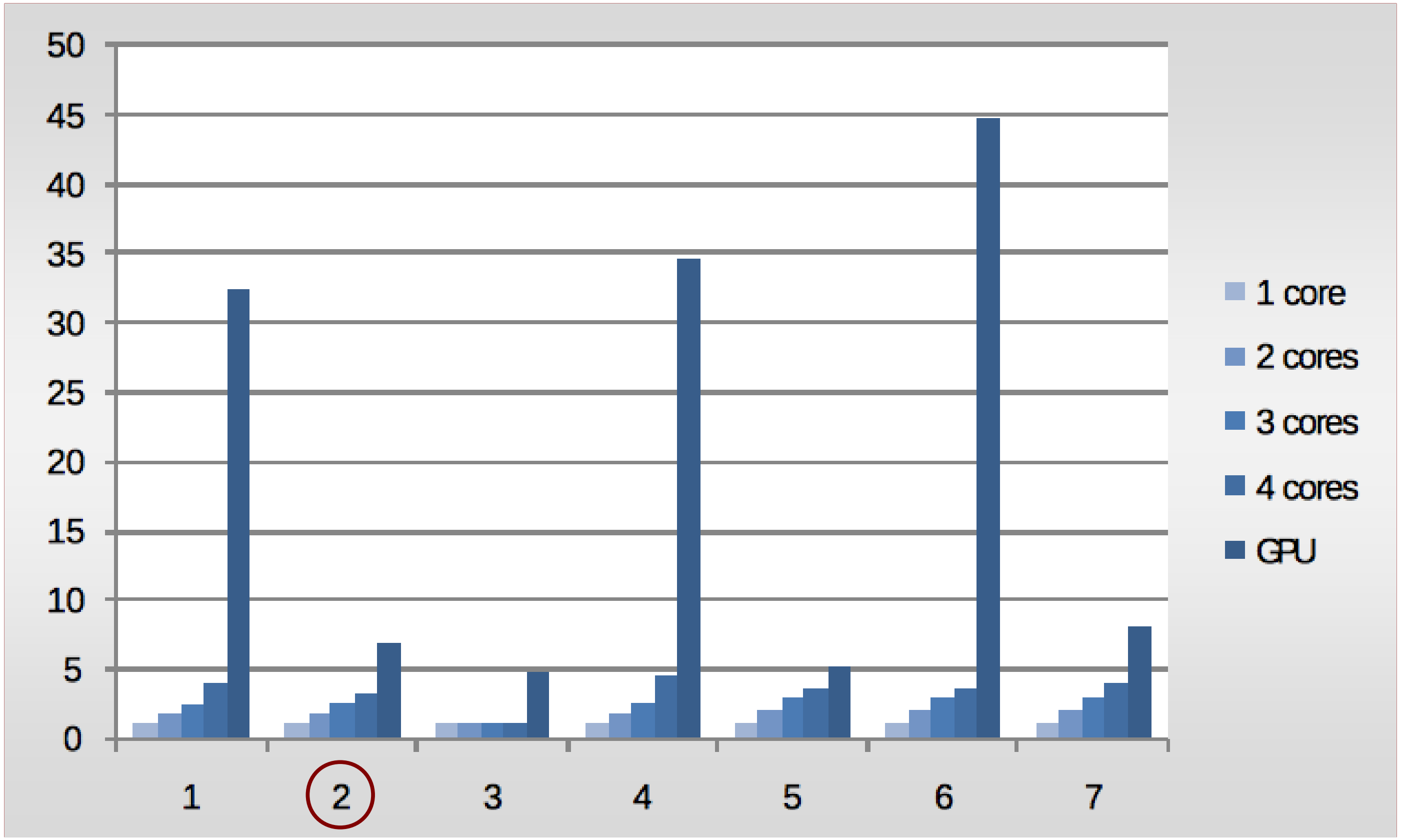


Kernel speedups – subdivision level 8



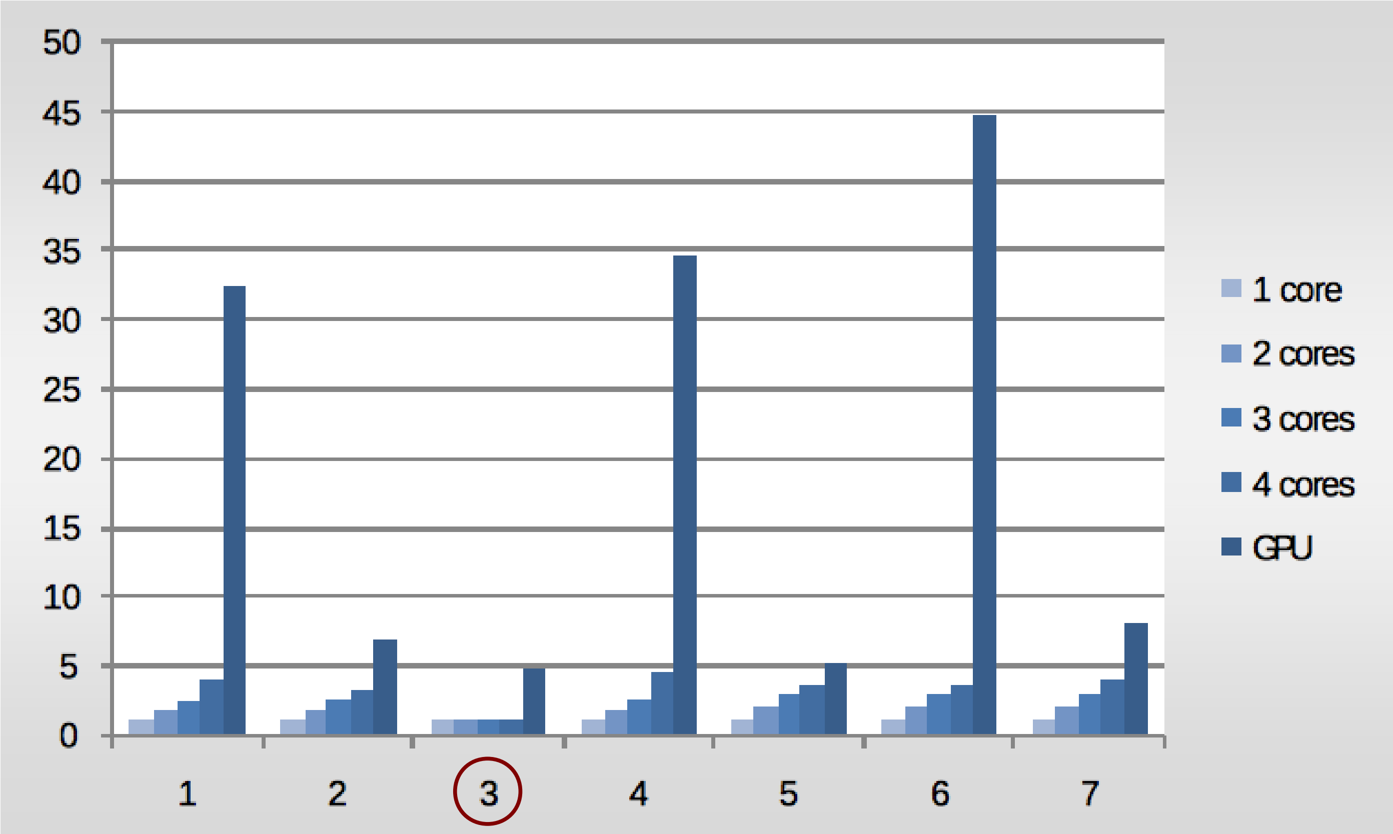
Kernel 1 – Surface refinement

Kernel speedups – subdivision level 8



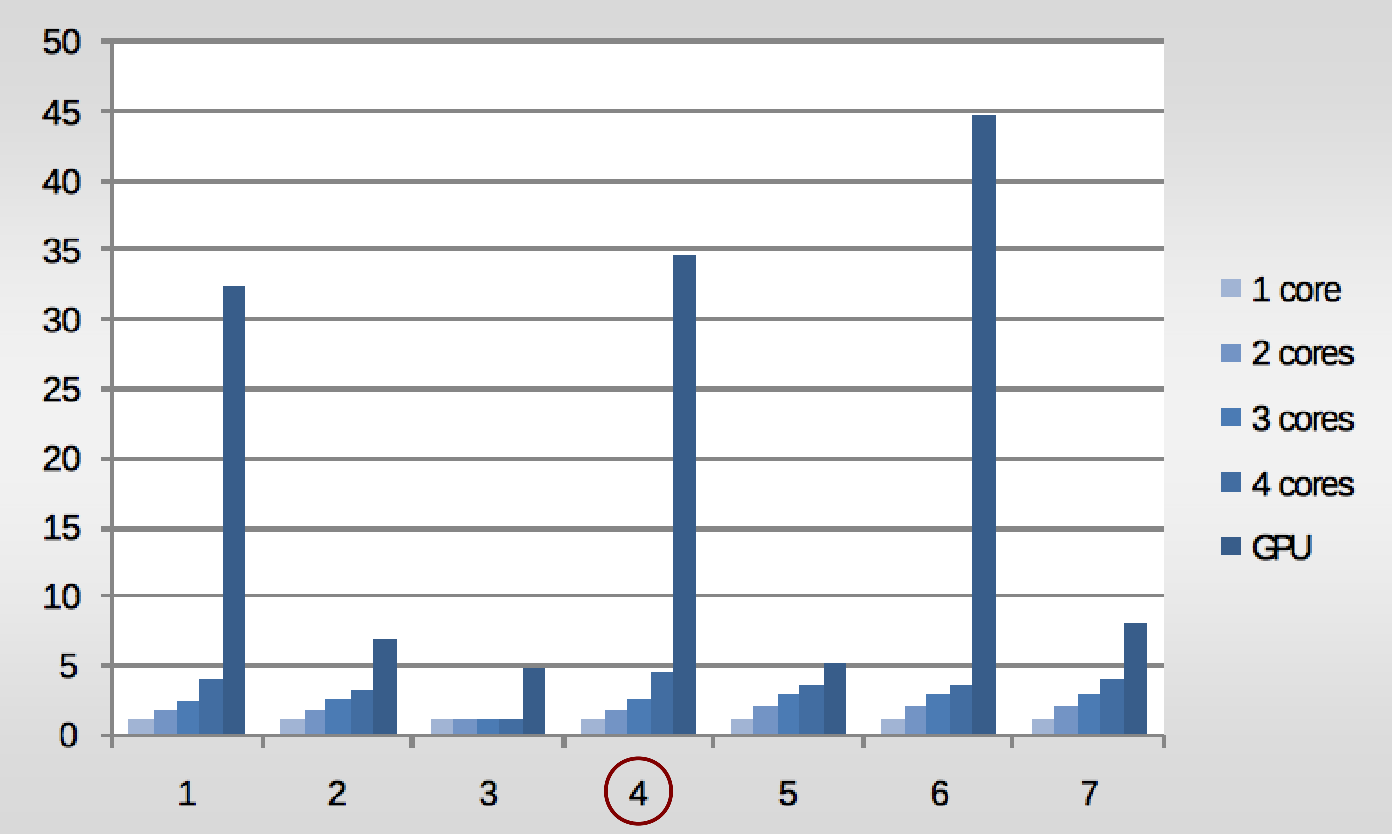
Kernel 2 – Bounding box generation

Kernel speedups – subdivision level 8



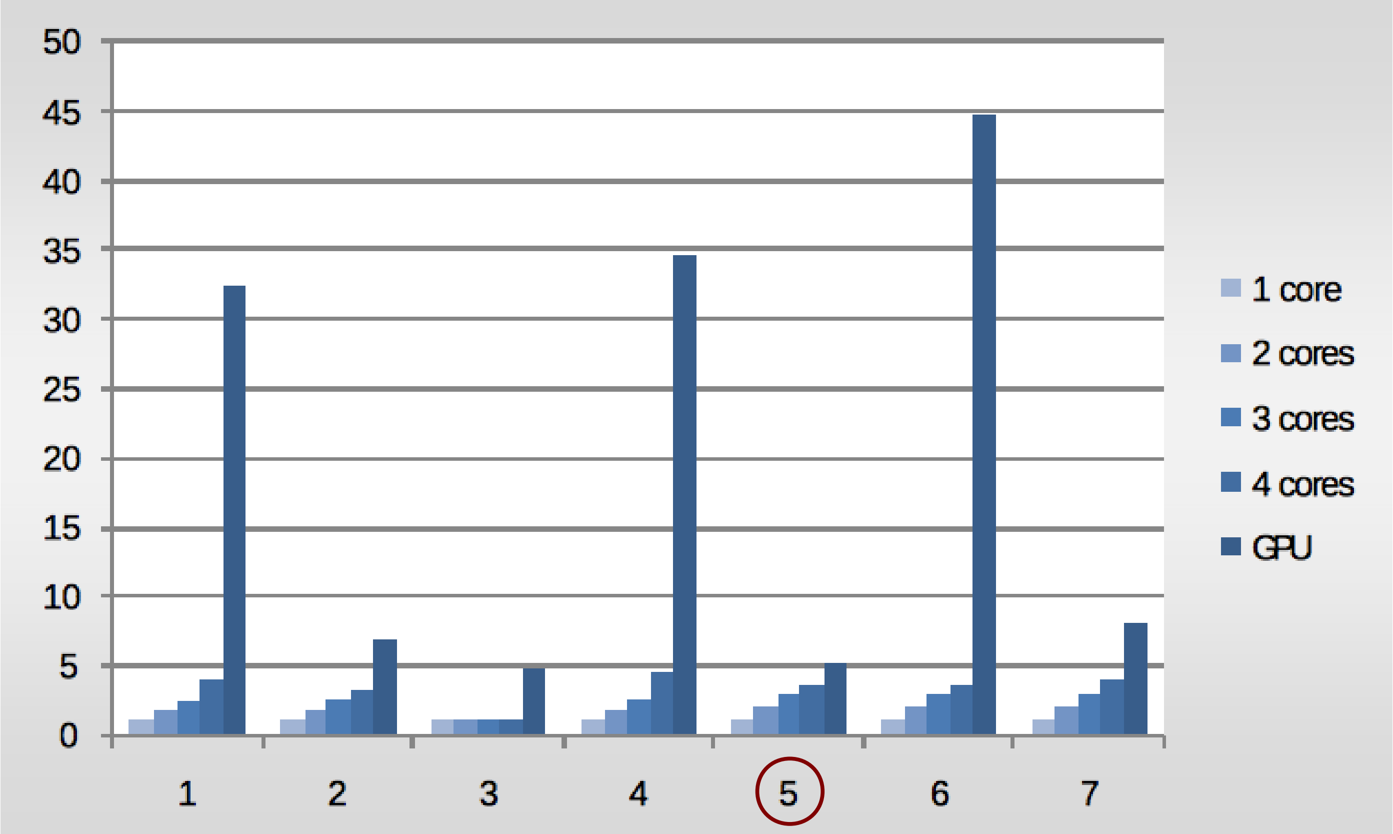
Kernel 3 – Box-box overlap test

Kernel speedups – subdivision level 8



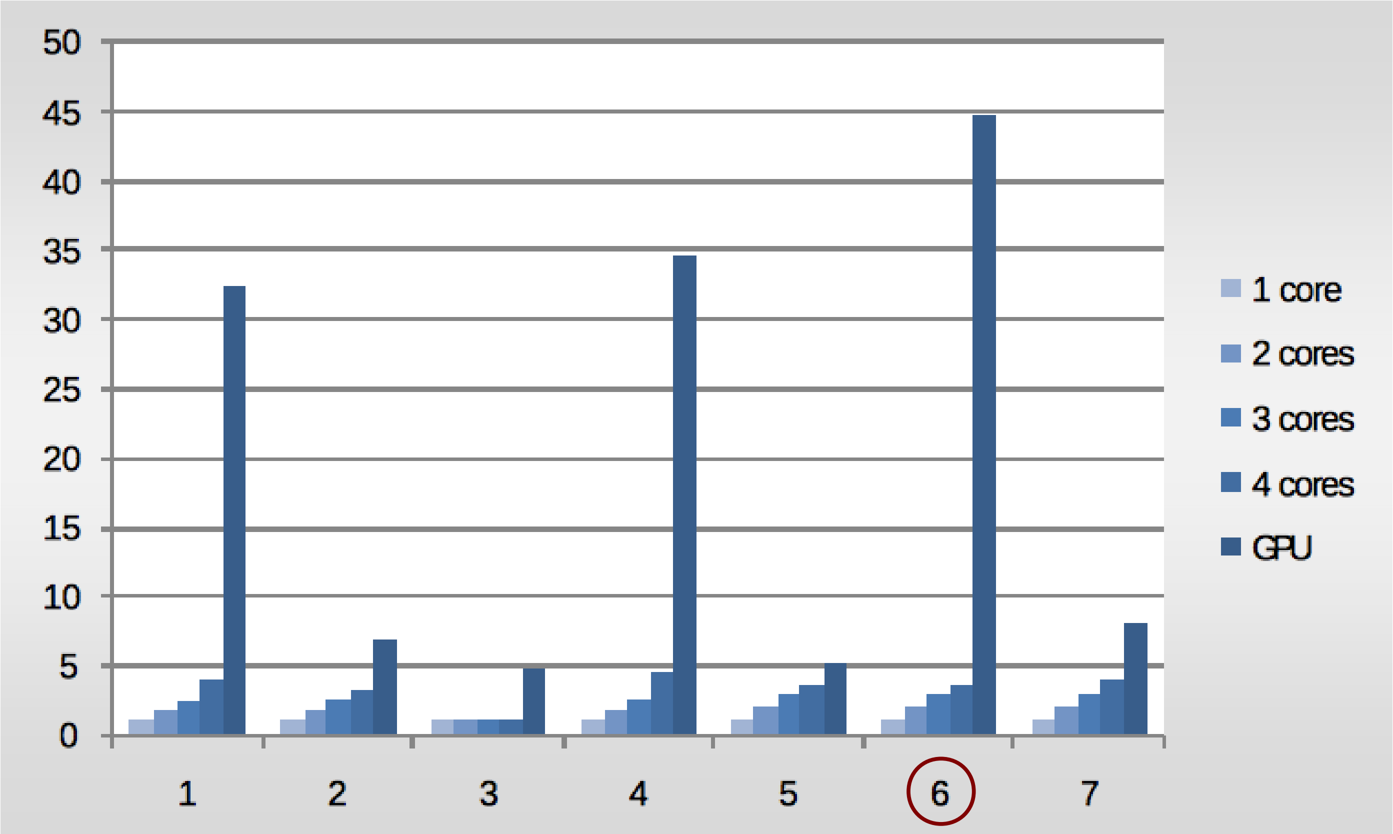
Kernel 4 – Normal surface refinement

Kernel speedups – subdivision level 8



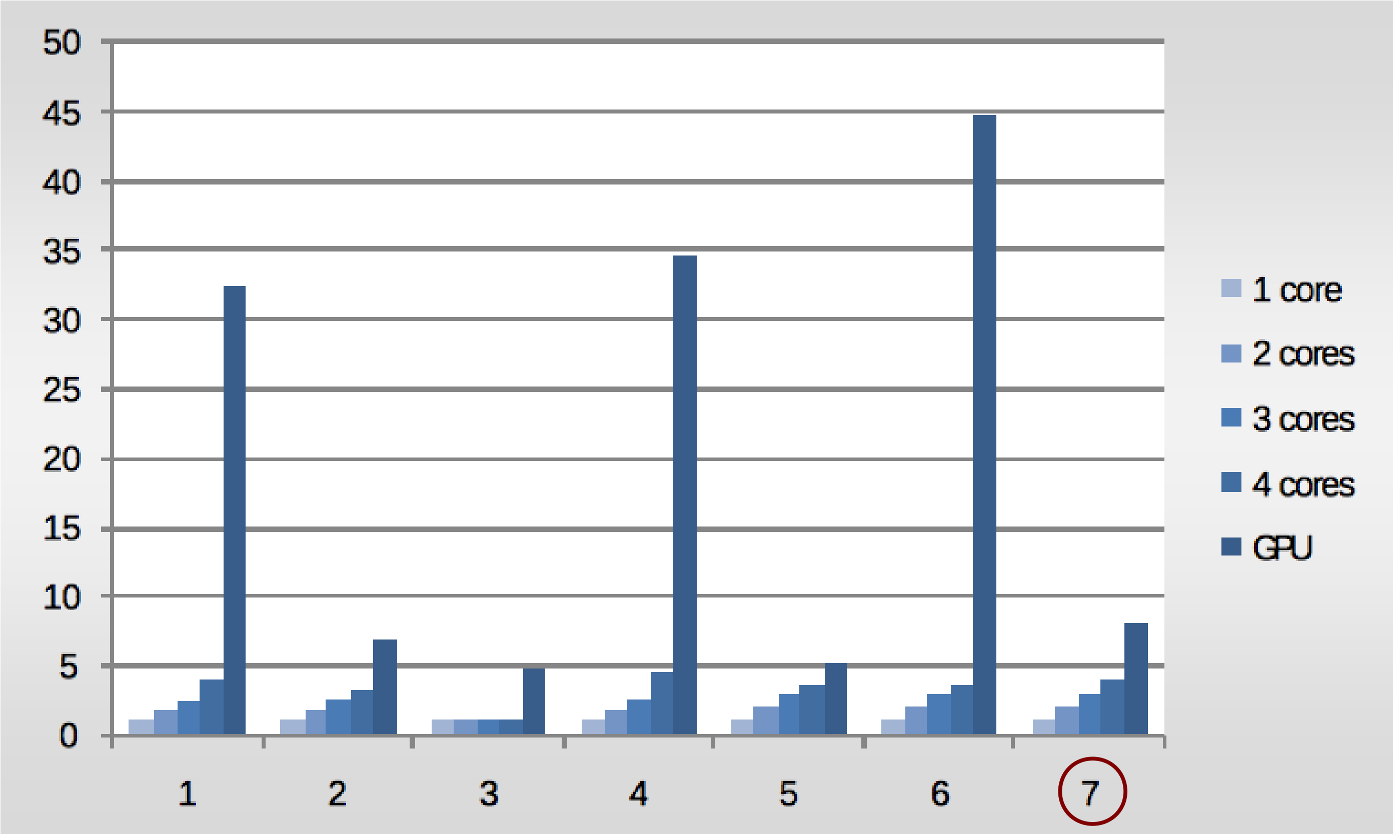
Kernel 5 – Normal surface degeneracy test

Kernel speedups – subdivision level 8



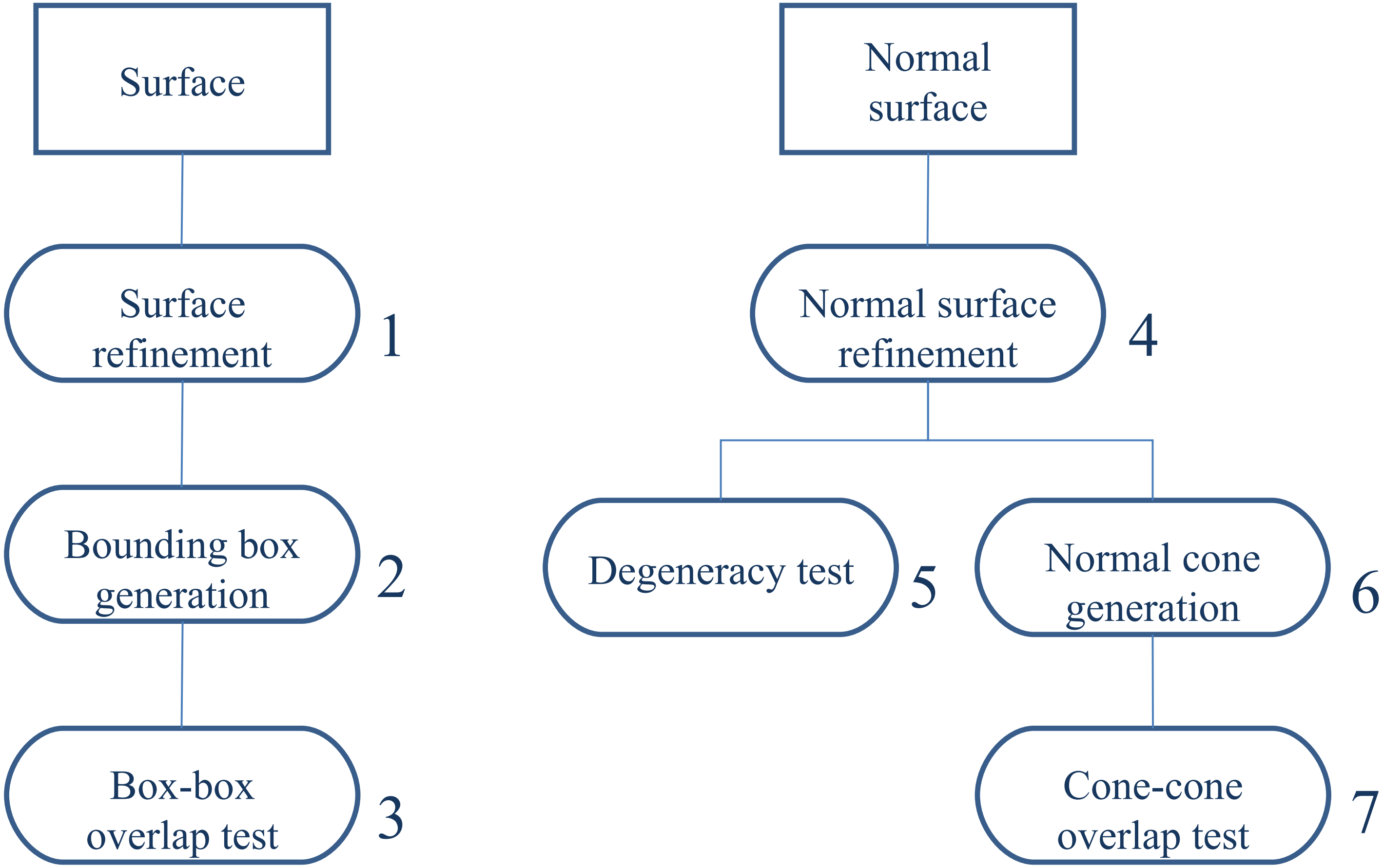
Kernel 6 – Normal cone generation

Kernel speedups – subdivision level 8



Kernel 7 – Cone-cone overlap test

Pipeline – Heterogeneous parallelization



Conclusions

- Heterogeneous intersections a good idea?
  - It does seems like it
- What about the algorithmic approach?
  - Well suited for difficult cases
  - Scales well on the CPU for most of the kernels
  - Good speedup on the GPU
  - Parallel pipeline allows load balancing between CPU & GPU
- Is the algorithm futureproof?
  - Future processors will get even more parallel
  - Faster CPU-GPU inter-communication reduces overhead
  - Heterogeneous algorithms will get even more important

Thank you for your attention!

Questions?