Grid Adaptation by Deformation in 2D and 3D

Michael Hein, Matthias Miemczyk, Raphael Münster

7. December 2006
Outline

1 Raphael
   - Objects in a numerical simulation
   - Strategy for minimum distance computation
   - Dynamic Hierarchical Data Structures
   - Rejection criterion for distance computation
   - Further acceleration methods

2 Matthias and Michael
   - 3D search algorithm
   - Efficient 3D search algorithm
   - Numerical results

3 Future Work
   - Future Work (Topics for diploma thesis?)
   - Bibliography
Objects in a numerical simulation

<table>
<thead>
<tr>
<th>Objects in Featflow</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Circles, spheres, boxes, rectangles or compound objects of these primitives</em></td>
</tr>
<tr>
<td><em>Goal: Use of multiple arbitrary shaped objects</em></td>
</tr>
<tr>
<td><em>Represent objects by closed NURBS curves or NURBS surfaces</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirements for ficticious boundary methods and grid adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Solve the containment problem</em></td>
</tr>
<tr>
<td><em>Compute the minimum distance between a grid point and an arbitrary shaped object</em></td>
</tr>
</tbody>
</table>
Strategy for minimum distance computation

Ray tracing methods in distance computation

- Use of hierarchical data structures to subdivide search space
- Quadtrees, octrees, k-d-trees, bounding volume hierarchies (spheres, circles, rectangles, AABBs)
- Quickly reject objects/regions that are not part of the solution
Dynamic Hierarchical Data Structures

Choice of acceleration structures

- Hierarchy of axis aligned bounding rectangles (AABR tree) of the objects
- Quickly reduce the candidates on the bounding rectangle level
- Hierarchy of circles to further decompose the objects
- Structures can be updated quickly without rebuilding them
- Reuse data structures for collision detection
Rejection criterion for distance computation

Lower bound / upper bound principle

- Elements (nodes) in a hierarchy supply methods to compute lower and upper bounds for the distance
- For each level in the hierarchy an upper bound is computed
- When a node’s lower bound is worse than the upper bound it can be pruned
Lower bound / upper bound search example

Raphael

Matthias and Michael

Future Work
Further acceleration methods

Incorporating time and space coherency

- A global upper and lower bound for the current grid point can be constructed from the preceding grid point
- Lazily update the bounding rectangle hierarchy
- Make use of information from the last time step (point inside, long distances,...)
Distancefield
3D search algorithm

problems in existing implementation

- slow & inaccurate raytracing
- ray-quad intersection was calculated by projecting the quad into the plane
- raytracing failed quite often, especially for non-planar faces
- fallback to brute-force search caused unacceptable time complexity
Search with distance information

**general idea**
- no expensive raytracing
- only cheap calculation of distances
- next element is found by calculating the smallest distance to the search point

**Algorithm one**
- check if point is within element
- distance from search point to the center of each face
- only cheap calculation of distances for six faces
- next element is adjacent element to the face with smallest distance
Algorithm one
## Search with distance information

### Algorithm two

- check if point is within element
- distance from searchpoint to the center of the element
- distance calculation for six adjacent elements
- go to the element with smallest distance
Algorithm two
Algorithm three, combination of cheap distance search and raytracing

- go to element with smallest global distance
- check if point is within element
- if not, start raytracing search
### Conclusion for distance-searches

- for quite regular elements distance search is an option
- problems arise as soon as the grid contains distorted elements
- every distance-search had an Achilles heel
- pure raytracing is not slower than combination
New raytracing implementation

- separate the quad faces into two triangles
- calculate ray-triangle intersection
- fast and accurate, fails only for heavily distorted elements
Numerical results

![Graph showing grid quality against deformation step. The graph indicates a decrease in grid quality as the deformation step increases.]
Numerical results
Numerical results
Numerical results
Numerical results
Numerical results
Numerical results
Numerical results
Numerical results
Future Work: Numerical analysis (Raphael)

- comparison with numerical methods (eikonal equation)
- extensive performance tests
- error estimation
Future work

Possible Todo’s

- Laplace and Umbrella-Smoothing (work in progress)
- parallel calculation of monitor function (OpenMP?)
- work on memory usage, for larger problems
- fix multigrid solver
Future work

Complex geometries (Diploma thesis Michael)

- Objects described by NURBS-surfaces and NURBS-patches
- Requirement: Efficient method for distance calculation
- Requirement: Efficient data structures
- Example below: 1578 patches (original data from Volkswagen)[1]
Future work

Diploma thesis (Matthias)

- Objects described by surface triangulation
- Grid deformation with fictitious boundary
- Versus
- Exact grid deformation and refinement on the boundary of the simulated object
- Or
- Combination of both methods
future work
REUSCHE, L.: Conversion of Trimmed NURBS Surfaces into Subdivision Surfaces.
Diplomarbeit, Technische Universität Carolo-Wilhelmina zu Braunschweig, 2005.