ACCELERATING ROUTE PLANNING AND COLLISION DETECTION FOR COMPUTER GENERATED FORCES USING GPUS

David Tuft, Russell Gayle, Brian Salomon, Naga Govindaraju, Ming Lin, and Dinesh Manocha
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599

Maria Bauer and Angel Rodriguez
US Army RDECOM Simulation and Training Technology Center
Orlando, FL 32826

Michael Macedonia
DTO
Fort Meade, MD 20755

ABSTRACT

We present algorithms to accelerate route planning and collision detection for computer generated forces. Our algorithms exploit the parallel computing capability of Graphics Processing Units (GPUs) along with their ability to perform geometric culling. We combine the GPU accelerated computations with exact intersection tests on the CPU. Our approach supports dynamic terrains and multiple feature intersections in parallel. Our technique has been integrated into OneSAF block D build 24. Our route planning technique is a 30x – 50x speedup and has demonstrated an overall speedup of 10x. Our collision detection code is a 5x – 10x speedup over existing collision detection techniques.

1 INTRODUCTION

Computer Generated Forces (CGFs) are computer systems that emulate the battlefield entities and units whose tactical behaviors and decisions are either made in part by human operators (Semi-Automated Forces) or automated decision algorithms (Automated Forces). A number of products have been developed to support army applications in three modeling and simulation domains: Training, Exercise, Military Operations (TEMO); Advanced Concepts and Requirements (ACR); and Research, Development and Acquisition (RDA). Over the last few years, major efforts have been directed towards Semi-Automated Forces (OneSAF and JLCCTC) operational requirements. For example, the OneSAF refers to a composable, next generation CGF that can represent a full range of operations, systems, and control processes from individual combatant and platform to battalion level, with a variable level of fidelity that

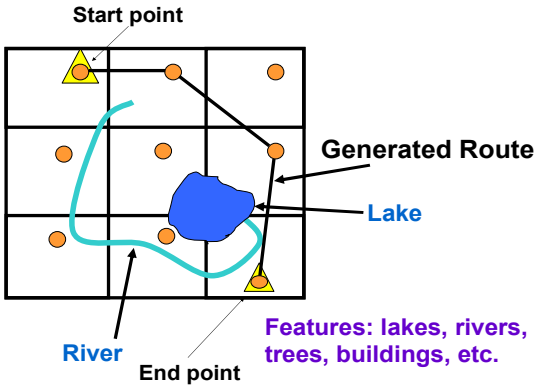

Figure 1*: Route generated avoids obstacles Segments between grids are tested against features to cull segment set.*

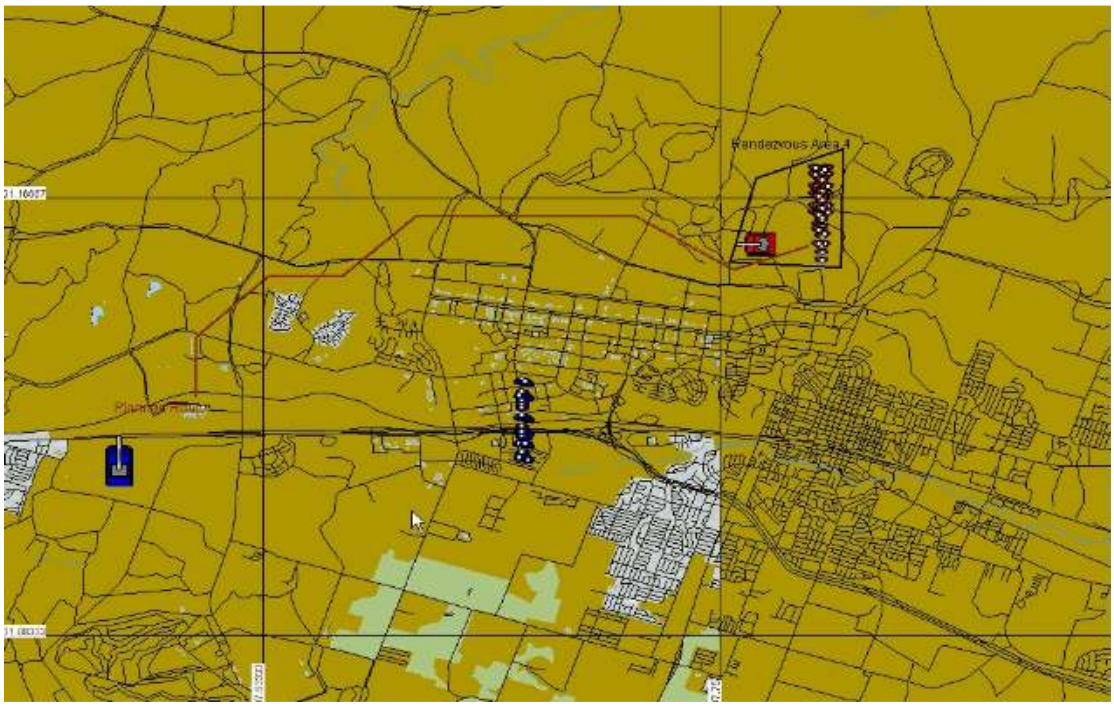

Figure 2*: Scenario in OneSAF. This scenario demonstrates the path segments that have been determined for a given route. The route planned for the tank is displayed in red. This scenario shows a 15x improvement with our GPU based techniaue enabled.*

supports all models and simulation (M&S) domains. A key component of OneSAF is the ability to add new equipment, units, behaviors, physical models and synthetic environment representations. In this paper, we focus on some of the behavior and terrain representations with respect to OneSAF, as well as use of simulation and planning-based technologies to effect the realistic movement of vehicles through the synthetic battle-space.

Route Planning and Collision Detection:
Terrain reasoning services can consume a significant portion of computing resources in Modeling and Simulation (M&S) applications. In the OneSAF Objective System (OOS), much of the available CPU is allocated to the dynamics agent some of this sub-allocated to collision detection and to route planning. This limits the amount of CPU resource given to cognitive models. Collision detection, route planning, and LOS computations can be major bottlenecks in computer generated force systems dealing with a high number of entities. This constraint severely limits the entity count sustainable by the simulation system. However, current CPU algorithms can be relatively slow and may not be able to perform their computations in this time frame.

GPU-based Computations:
We exploit the capabilities of GPUs to accelerate intersections tests. Modern GPUs have built in special purpose hardware designed for calculating visibility. Salomon et al. used this hardware to cull Line of Sight (LOS) rays [Salomon et al. 2004]. This technique checks LOS rays with the same hardware that has been used to cull LOS rays for intersections.

Our work accelerates route planning and collision detection by testing segments of routes and entities against a buffer of features. This process is able to quickly condense the

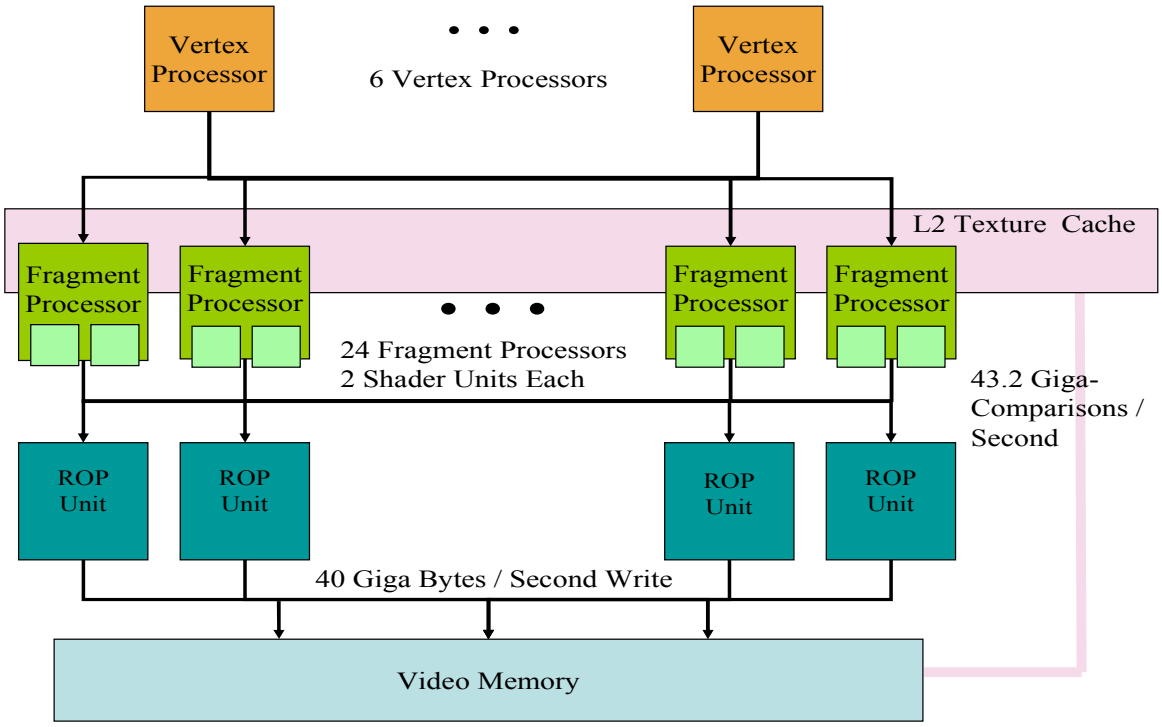

Figure 3: *In this abstraction of a GPU-pipeline, the Vertex Processors transform triangles which are then rasterized into fragments. Fragments are then shaded and fed into the ROP units. The ROP unites perform fixed function updates to the depth stencil and color buffers. Our algorithms use the vertex and fragment processors to render the objects, and the ROP unites for the visibility tests.*

set of non blocked routes or non colliding entities. Once this set has been reduced, we can determine final collision detection and non-blocked routes with traditional CPU based approaches.

2 GRAPHICS PROCESSORS

Fast graphics hardware including 3D rasterization, texturing, and dedicated vertex and pixel processing has become as ubiquitous as floating-point hardware. It is nearly impossible to buy a PC without dedicated 3D rasterization and texturing hardware. Moore's Law, the highly parallel nature of graphics rendering algorithms, and the computation demands for simulating visual reality have converged to drive the development of faster and more capable graphics hardware. The ubiquity and performance of this hardware leads us to consider the extent to which this hardware can be harnessed to solve scientific, simulation and visibility problems beyond the conventional domain of image synthesis for the sake of pretty animation.

GPUs have been progressing at a rate faster than Moore's Law. The same growth rate is expected to continue for the next 3-5 years, giving us the capability to perform GFlops of computation on a $350 COTS GPU. This makes the GPU an excellent candidate for performing scientific, geometric and compute-intensive algorithms.

Furthermore, GPUs have greatly increased capabilities. The current GPUs are optimized for rasterization of 3D geometric

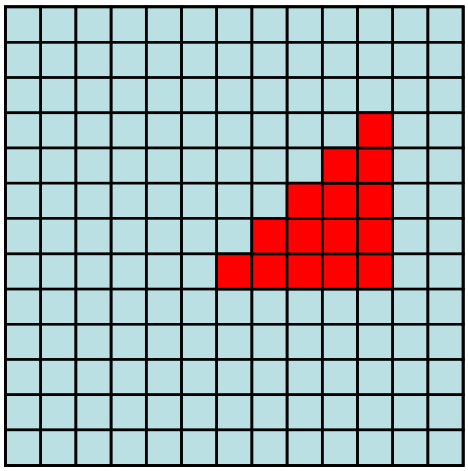

Figure 4: *15 pixels are returned for the occlusion query corresponding to the red triangle. The Occlusion query returns the visibility of an object as count of pixels rendered.*

primitives. They also have efficient image processing capabilities. Moreover, the vertex and fragment processors provide the application programmer a great deal of flexibility and power. Because of these capabilities, an incredible array of new algorithms and real-time implementations of previous algorithms have been made possible. Furthermore, as graphics hardware becomes more programmable, the barrier between the CPU and the GPU is being redefined. The new languages and compilers for programming the GPUs make it much easier to use them for a variety of applications. Finally, the underlying precision of the GPUs is increasing as well. Current GPUs can support 32-bit floating point frame-buffers.

Figure 3 shows some of the key features of a modern GPU pipeline. At the top of the figure are the Vertex Shaders. These MIMD processing units operate on the data associated with each vertex. After the Vertex shaders, the data is broken up into fragments by the rasterization hardware. The fragment processors operate on all the data associated with a fragment. These processors output to the fixed function Raster Operators (ROPs). The ROPs perform fixed function operations

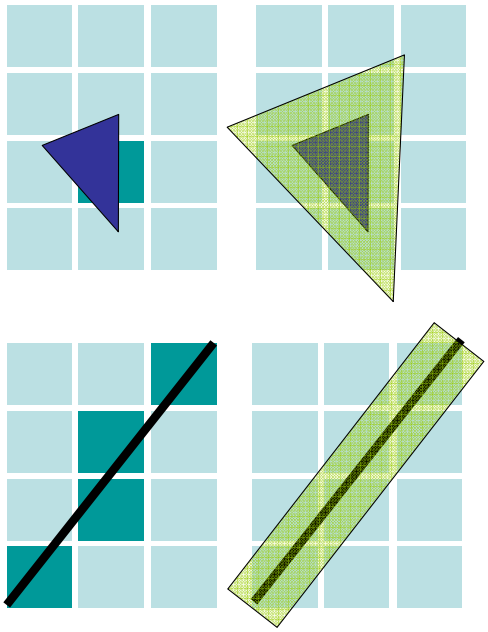

Figure 5: *Conservative rasterization of terrain triangles and Route path. We ensure that each pixel intersected by a feature or route segment generates a fragment with a conservative depth value.*

between the output fragments and the various buffers (e.g. depth, stencil, and color).

There are several features of current GPUs that are necessary for feature intersection technique. These features include: the depth buffer, depth buffer flags, and occlusion query capabilities. The depth buffer is a 2D array of scalers. Each scaler in the depth buffer is represented by 24 bits of memory scaled between 0 and 1. This buffer is used in depth-buffered rendering to keep track of the fragment nearest the viewer with a simple comparison test. If the depth of the pixel is less than the current one, the color and depth value are overwritten with the incoming fragment. Modern GPUs support expanded depth operations. The type of comparison operator can be set by the application (e.g. less, greater, equal, etc). Furthermore, the GPU can provide a count of the number of fragments passing the depth test while preserving the contents of the depth buffer using write masks. Fragment counts are established using *occlusion queries* that track the number of fragments passing the depth test

while rendering a batch of primitives.. Figure 4 is a graphics representation of a triangle that has been resterized. An occlusion query returns the visibility of a given primitive as a count of pixels rasterized.

3 OVERVIEW

Feature intersection is the algorithmic process of determining if the primitives making up a feature geometrically intersect with the primitives that represent a route segment or entity. A key component of our approach is a new feature intersection test that is used both for route planning as well as collision detection.

Our feature intersection algorithm leverages off of the previously mentioned commodity graphics hardware features. Our algorithm works by testing objects against features.

Each feature analysis task can be generalized to determine the exact set of features that a single object intersects. Our GPU-accelerated algorithm performs a quick and conservative culling of objects and terrain features to accelerate the computations step produces a near minimal set of objects and features which require CPU-based exact intersection tests.

Culling is performed using the GPU's occlusion query capability. These queries are used to determine whether two objects are overlapping when rendered from a particular viewpoint. Overlaps between objects and features are determined simply by rendering the objects in succession.

The terrain features being tested for intersection can be very complex. This may result in small details of the features being smaller than one pixel after rendering. Due to this inaccuracy, an image-based technique may miss intersections. In order to ensure that all potential intersections are found, we conservatively expand the size of the objects as in [Govindaraju et al 2004].

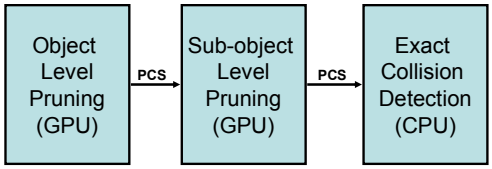

Figure 6: *System Architecture: The overall pipeline of the collision detection algorithm for large environments.*

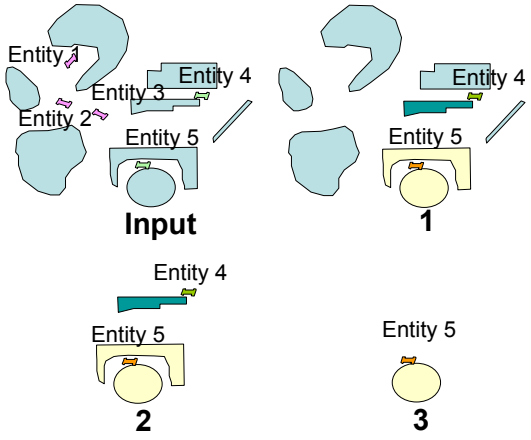

Figure 7: *Our technique proceeds in three phases: First, entities are checked against the set of all features. Second, all features are checked against remaining entities. Third, the potentially colliding features are checked against their prospective entities.*

4 COLLISION DETECTION

The problem of collision detection can be reduced to that of feature intersection. In this case, each entity must be tested against all features to determine collision. The collision detection information can be used by simulators to add collision avoidance and entity damage simulations. Collision detection must happen every time entities move to be accurate. This problem is complicated because entities and features can be complex objects and there may be many dynamic entities.

Calculating collision detection for every entity in a simulation can be an expensive task. To get around this high cost, most collision detection algorithms perform bounding sphere intersections or use

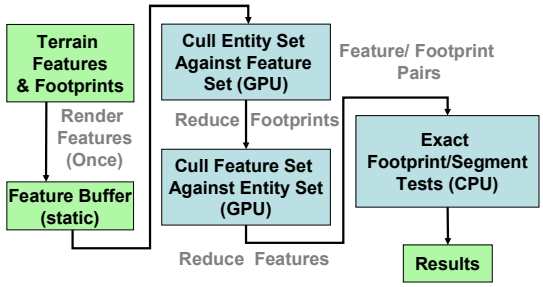

Figure 8: *We use two forms of GPU culling to limit the number of entities and features per entity that must be checked.*

hierarchies. Sphere intersections are not as reliable as true collision detection. Hierarchies require memory overhead and are not well suited to dynamic features. Our method is built on Cullide [Govindaraju et al 2003] and calculates accurate collision detections for high res entities and features.

Collision detection is necessary to add realism and realistic entity behavior to a scenario. In such a case entities must be tested each frame against features to create realistic movement. Collision detection allows OneSAF to create realistic entity movement. It also allows for realistic entity interactions such as vehicle damage.

Our GPU based collision detection algorithm is a culling approach. This approach relies on hardware visibility tests to cull features and entities from a potentially colliding set.

Collision detection proceeds in three phases:

- An entity is checked against a feature buffer to see if that entity can be culled
- Features are then checked against each entity that cannot be culled
- The resulting features and entities are then checked on the CPU for collision

Our algorithm works as a 2 part culling technique followed by an exact CPU test. In part 1 the entities are tested against a buffer containing all features. This step is very fast

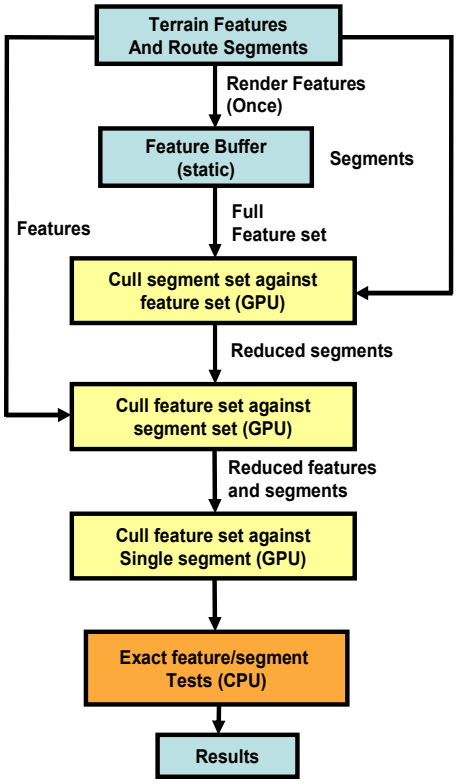

Figure 9: *We use three forms of GPU culling before performing exact intersection tests between features and segments.*

and conservatively culls most colliding entities that are not in near proximity to features. Next, for each entity a set of potentially colliding features is created. The GPU compares features against individual entities. This reduces the number of potentially colliding features per entity. This small subset is then very rapidly tested on the CPU to determine if a collision has occurred.

5 ROUTE PLANNING

The topic of route planning has received considerable attention largely due to the high computational complexity associated with it. Good solutions to route planning have applications in many areas, including autonomous navigation or planning among

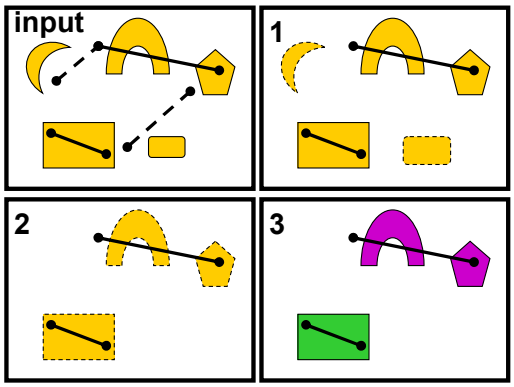

Figure 10: *Our GPU-based culling algorithm proceeds in three phases. First, non-intersecting segments are pruned. Second, non-intersecting features are pruned. Third, potentially intersecting features are paired with segments.*

collaborating agents [Kennet et al. 1999, Varadhan et al. 2005].

In a CGF application, quick and accurate route planning is critical for accurate and effective unit movement. However, a planning task for a single unit or group can be very expensive. When large numbers of units require route planning, the task becomes too daunting to be performed in a reasonable amount of time. In the OneSAF system, the bottleneck (taking over 50% of CPU time) is feature analysis computation. In order to improve the performance of planning in the OneSAF system, we need to improve this task.

Our route planning method performs conservative culls similar to our collision detection technique. This technique proceeds in three phases:

- The number of segments is reduced by culling them against the full feature set
- The number of features is reduced by culling them against the reduced set of segments
- The reduced feature set is culled against each individual segment in the reduced segment set

Figure 9 shows an outline of the steps taken. First, all features are rendered into a buffer. We then test individual route segments against the buffer. Next, the features are culled against a buffer of segments. Finally, individual segments are checked against the buffer.

Integration into OneSAF:

Route planning is broken up into feature read and feature analysis. In Feature read, geotiles are broken up into nodes that can later be used for grid based A* path finding. The A* algorithm requires segments connecting paths to be tested against features. Feature analysis is the process of determining what route segments intersect features. The GPU route planning technique accelerates feature analysis. The OneSAF feature intersection algorithm was run on all segments stemming from a single node at once. Verdesca et al. describe the integration process in more detail [Verdesca et al.]. Our techniques have been integrated into OneSAF block D build 24.

6 RESULTS

Our system performs rapid feature intersection checks for potential route segments and entites. Our system is also able to perform multiple feature intersections in parallel. Using this method we were able to Conservative reduction of the number of feature intersection tests resulting in a 30-50x speedup in feature analysis computation for route planning. In collision detection, we demonstrated 5-20x speedup for dynamic collision avoidance.

7 CONCLUSIONS

We believe that exploiting the high computational power of GPUs is essential to increase the complexity of simulations that can be performed in real-time for computer

generated forces using systems such as OneSAF. Our algorithm for accelerating route planning and collision detection using the GPU is one example.

Our algorithm is currently integrated into OneSAF which will reduce the time of route planning and collision detection computation allowing more complex scenarios to be run.

ACKNOWLEDGEMENTS

We would like to acknowledge AMSO and DARPA/RDECOM Contract N61339-04-C-0043. We would also like to acknowledge Eric Root, Marlo Verdesca, and Jaeson Munroe from SAIC for their involvement in the integration of this technology into OneSAF. We are also grateful to LTC John Surdu and the OneSAF team for their support.

REFERENCES

Henderson, D. L., 1999: Modterrain: A proposed standard for terrain representation in entity level simulation, MS thesis, Naval PostGraduate School.

Messina, P., et al., 1999: Synthetic Force Express: A New Initiative in Scalable Computing for Military Simulation.

Govindaraju, N., Redon, S., Lin, M. and Manocha, D., 2003: CULLIDE: Interactive collision detection in large environments using graphics hardware. *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware.*, 25-32.

Govindaraju, N., Lin, M., and Manocha, D., 2004: Fast and Reliable Collision Culling using Graphics Processors, *Proc. of ACM VRST*.

Verdesca, M., Munroe, J., Hoffman, M., Bauer, M., Manocha, D., Using Graphics Processor Units to Accelerate OneSAF: A Case Study in Technology Transition *Proc. of I/ITSEC* 2005. Best paper from Research Development Category.

Salomon, B. Govindaraju, N. Sud, A., Gayle, R, Lin, M., Manocha, D, 2004, Accelerating Line of Sight Computations Using Graphics Processing Units, 24th Army Science Conference Proceedings 2004.

Hoff Kenneth, Culver T., Keyser J, Lin M. Manocha D., 2000: Interactive Motion Planning Using Hardware-Accelerated Computation of Generalized Voronoi Diagrams. *IEEE Conference on Robotics and Automation* 2931-2937 vol.3

Gokul Varadhan, Dinesh Manocha, "Star-shaped Roadmaps - A Deterministic Sampling Approach for Complete Motion Planning", *Robotics Science & Systems* 2005