Tactical Movement Planning for Individual Combatants

Douglas A. Reece
Matt Kraus
Science Applications International Corporation
12479 Research Parkway
Orlando, FL 32826
reeced@saic.com, krausm@siac.com

Paul Dumanoir
U.S. Army Simulation, Training and Instrumentation Command (STRICOM)
12350 Research Parkway
Orlando, FL 32826-3276
Paul_Dumanoir@stricom.army.mil

Keywords:
Terrain reasoning, path planning

ABSTRACT: *Moving computer generated forces realistically has challenged CGF and computer game developers for years. The CGF community has mainly addressed ground vehicle movement, and through a variety of approaches has provided solutions to many entity movement problems. In our project (DISAF), we are faced with the problem of moving individual soldiers rather than vehicles. The vehicle-based movement behaviors and controllers in our software baseline (ModSAF) have proved inadequate for the task. This paper describes the tactical movement task for individual combatants, provides an analysis of path planning algorithms from ModSAF s and the robot planning literature, and presents our algorithm for planning movement. The tactical movement behavior we developed, considers slopes, seeks concealed routes, and adjusts posture and speed when in threatening areas. It uses a combination of grid search and graph search to find paths outside and inside of buildings.*

1. Introduction

While many military simulations have addressed individual vehicle movement behavior, none of them has adequately addressed realistic infantry movement. We are developing a comprehensive movement capability for synthetic individual combatants (ICs) in small units. In particular, we have developed movement planning and control behaviors that allow a simulated IC to move tactically in a complex environment. This capability is being developed as part of the DISAF project funded by STRICOM and performed by SAIC.

We have implemented selected parts of the movement behavior architecture, including squad and fire team fire and maneuver behaviors and individual tactical movement. This paper describes the development of the individual tactical movement behavior. Individual tactical movement must integrate terrain obstacles, trafficability, and cover from threats into route selection, and must include speed and posture changes into movement control. The next section reviews the requirements for tactical movement. Section Algorithm analyzes movement planning algorithms and the impact of the requirements on algorithm selection. Section Implementation describes the implementation and results of path planning, and the final section provides concluding remarks.

2. Requirements

2.1 Motion planning

Movement can be viewed at several levels of abstraction. Higher levels of abstraction generally correspond to longer distances and periods of time. The actions required to plan and execute movement vary with the level. The relevance of different movement levels to individual tactical movement are as follows:

- Long distance route planning–a human operator or simulation task sets a destination many kilometers from the individual, and the individual plans and executes movement to that destination. We assume that long range moves are generally performed by higher level units and use vehicular transport.

Long distance movement is thus not a part of individual movement.

- Intermediate distance route planning–an operator or task sets a destination a few hundred meters to a couple of kilometers away and the individual moves there. This range is far enough that detailed planning around small obstacles such as individual trees is not appropriate, but planning is still required at a high level to avoid large obstacles such as bodies of water and to find roads and bridges. The individual may consider, at a coarse level, ground trafficability and slope, proximity to threats, exposure to threats, etc. in planning its movement.
- Short distance movement planning–an individual moves to a point no more than about a hundred meters away. Individual tactical movement behavior primarily addresses this level of movement. Individual tactical movement must move the soldier around all obstacles, attempt to find an efficient path through terrain of varying slope and trafficability, and avoid areas exposed to enemy threats. The effect of slope varies with direction of movement. Other moving agents must also be considered. In general, the other agents might be moving in related paths for formation movement, bounding movement, or room clearing; for the purposes of individual tactical movement, other movment agents are assumed to be on unrelated paths and are treated as moving obstacles.
- Fine motion planning–the individual changes postures to stay in cover while moving; changes posture and orientation to move through small apertures, duck under obstacles, or pass by other people in narrow passageways; sprints across exposed areas; climbs in windows, climbs up ladders, jumps over obstacles, jumps down from obstacles, kicks open doors, etc. In our DISAF simulation, most fine motion is abstracted away, so we consider only walking, running, and crawling actions in the tactical movement task.

2.2 Environment

The nature of the simulation environment–the types of features, the density of features, and their native representation–has some bearing on the type of movement planning algorithm that works best for the behavior. As mentioned earlier, the movement behavior was developed for DISAF. In DISAF the typical representation of terrain skin and terrain features is as polygons. The terrain databases of interest may have small terrain polygons (on the order of meters or less), closely spaced individual trees and man-made structures, and buildings with interiors (Multi Elevation Structures, or MESs). The MESs constrain movement a great deal, requiring precise movements accurate to tens of centimeters to avoid colliding with walls. The MES data structure in DISAF includes the topology of the rooms and the locations of doors and windows connecting the rooms.

3. Algorithm Analysis

3.1 Path finding algorithms

There are several good reviews of path finding algorithms in the literature [1] [2]. This section briefly reviews important ones for the tactical movement behavior described above.

3.1.1 Cell decomposition

In a cell decomposition approach, free space and obstacles are represented as a grid of small, uniform cells. Cells may be hexagonal or irregular in an obstacle-dependent way, but are most often squares. Cells cannot represent the shape of an obstacle exactly, but can get arbitrarily close if the size of the cells is small. Of course, decreasing the size of the cells increases the size of the search space. In addition to discriminating between free space and obstacles, cells can be assigned values that represent anything related to movement, e.g., trafficablity, ground slope, or exposure to threats. This feature of the cell decomposition approach is well illustrated by [3] who used factors such as distance-to-enemy to generate continuously varying cell values.

Once the terrain has been cast in a grid representation, a path can be found by searching the grid from the start point to the destination. The A* algorithm is typically used to control search, with straight-line distance to the goal used as a heuristic function. If cell values represent only the distance across the cell, then the search will yield the shortest distance to the goal; if other functions are used to define the cell value, then the path will be optimum with respect to those cost functions.

Cell search has the disadvantage that if high resolution planning is desired, the cells must be small. High resolution is needed in our domain to plan motion down narrow hallways, across bridges, and past other constrictions. Search cost increases even if there are few obstacles in the terrain and most of the terrain is uniform in cost. To represent critical

detais in the environment without paying the search cost everywhere, hierarchical approaches have been used. Kambhampati [4] describes multi-resolution A* search that uses quadtrees of cells. Yahja [5] gets fewer grid discretization artifacts in the path by using large uniform cells framed by high-resolution cells. Karr [6] uses a grid to search large areas for units, but marks important linear features in the cells so that the search process may accurately determine if one cell may be reached from another. For example, rivers are small in size relative to the cells, but they may obstruct movement across a cell in one direction.

3.1.2 Skeletons

Skeleton approaches reduce free space to a network of one-dimensional lines. Common representations are visibility graphs and Voronoi diagrams. A visibility graph is a collection of lines that connects the visible vertices of obstacles with each other. A Voronoi diagram is the set of points equidistant from two or more objects. Path planning with these skeletal free space representations involves finding a path from the start point to the nearest skeleton line, doing likewise with the goal point, and then using a graph search technique to find the lowest cost path from start to goal along the skeleton. The visibility graph solution yields a true shortest path, taut-rope solution (at least for the portion of the path on the graph), while the Voronoi diagram solution yields a path that stays as far as possible away from the obstacles.

A variation of this planning technique is used to find "short" paths in ModSAF [7] and CCTT, another U.S. Army simulation system [8]. In these systems, no representation of free space is built. Instead, candidate free space routes are generated and evaluated. The first candidate route goes directly from the start to the goal. If it intersects an obstacle, it is rejected but two other candidate routes are created by computing "skirt" points around the obstacle in each direction. If the skirting paths intersect obstacles, additional routes will be generated by skirting those obstacles, and so forth. The route selected will be the candidate that doesn't intersect obstacles and is the shortest (or best by some other criteria).

ModSAF has another path search algorithm for planning paths inside of MESs [9]. This algorithm uses the topological information available in MESs as a graph of free space. This approach takes advantage of the fact that all paths through the building have to pass through choke points in the doorways between rooms. The doorways are nodes in the graph; the graph is searched using A*. Unfortunately, this path search algorithm is not integrated with the path search algorithm described above.

ModSAF has yet another algorithm for planning concealed routes [10]. This algorithm uses cells to represent concealed areas initially, then transforms individual areas into nodes of a graph. Concealed areas are generated only by terrain features such as trees and buildings, but not by the terrain skin itself (e.g. hills). The nodes are joined by edges whose weights are the distances between the concealed areas. The A* algorithm is used to search this "concealment space" graph for the lowest-cost path. The algorithm does not also plan routes around obstacles; the algorithm for doing that is described above. Because the mechanisms developed for these two requirements are incompatible, the two requirements are handled sequentially. The result is that concealed routes may be infeasible due to obstacle blockage, or the obstacle-free routes generated from concealed waypoints may detour into exposed areas.

3.1.3 Weighted regions

The cell decomposition of movement space described above is really a specialization of a decomposition into arbitrarily shaped weighted regions. It is possible with arbitrary polygonal regions to represent large uniform areas very economically while at the same time representing small terrain features with small regions. A minimal cost path across a space of polygonal regions will bend only at region boundaries. Mitchell [1] describes how optimal paths will bend at region boundaries according to Snell's Law of Refraction; i.e., the ratio of the sines of the angles of incidence and refraction of the optimal path will be equal to the ratio of the weights of the regions.

A commonly used algorithm to find optimal paths in weighted regions is described in [11]. The essential idea is that rays are cast out in all directions from the start point; each ray bends at region boundaries to obey the optimality criterion outlined above. The goal point must be "trapped" between two rays. The path found will be closer to optimal if more rays are used. This path planning approach was used in a variation of ModSAF used to control U.S. Marine Corps infantry units [12]. Regions given very high weights if they were impassable, high weights if they

were exposed to threats, and low weights otherwise (this is a modification of a standard "0-1-∞" weighting set).

3.1.4 Potential functions

A final approach to path planning is to construct a scalar potential (as in energy) function that steadily decreases to a minimum as the distance to the goal decreases. The potential function is high at obstacle boundaries and decreases as the distance to the obstacle increases. The path is determined by following the steepest descent down the potential value surface until the goal is reached.

There are two difficulties with the potential function approach. The first is that the net potential function from multiple obstacles may have a local minimum that traps the algorithm. In other words, the obstacles may form a dead end alley. This problem can be addressed to some degree by using better obstacle repelling functions [13] or by making the potential a function of velocity [14]. The second difficulty is that regions cannot be weighted to encourage or discourage movement through them. Thus, there is no way to encourage movement *near* obstacles to gain from their protection from enemy observation. While these two issues make potential fields awkward for some route finding problems, the issues don't usually arise if a route planner has found a generally clear course and the potential fields are just used to make local corrections. Thus they are useful as a low level movement controller that is following a path that is known to be satisfactory from a global perspective.

3.2 Algorithm evaluation

The algorithms above can be evaluated in terms of the requirements of individual tactical movement. The key requirements of individual tactical movement are 1) planning a path through an obstacle field, 2) planning an optimal path through regions of different cost, 3) planning an optimal path through terrain of continuously varying cost (for example slope in the direction of movement), and 4) using exposure to threats as a terrain cost. The approaches are 1) cells, 2) skeletons, 3) regions, and 4) potential functions.

1. **Path planning through an obstacle field**. Cells handle this well, but suffer from computational cost when fine motion is required. This can be a big problem inside buildings. They also produce paths with grid artifacts (jagged paths). Skeletons handle the problem efficiently. The weighted regions approach also works well, but is probably more expensive than skeletons. Potential functions are adequate for uncluttered problems, but have difficulty with problems that require a global path search, e.g. the mazes that may be encountered in urban areas.
2. **Planning an optimal path through regions of different cost.** Cells handle this well. Skeletons cannot do this at all; the paths are only generated to avoid obstacles, without consideration of other terrain cost. The only use of terrain costs is to choose between candidate obstacle-avoiding paths. The weighted regions algorithm efficiently produces the best paths when there are large regions of uniform cost. Potential functions do not consider weighted regions.
3. **Planning an optimal path through terrain with continuously varying cost.** Cells handle this well. Skeletons cannot do this. The weighted regions no longer works when cost varies continuously, for the paths in general curve and the optimal-path "rays" cast from the start point are difficult to compute. Potential functions do not consider weighted regions.
4. **Using exposure to threats as a terrain cost**. This only applies to the cell and region approaches; the other two are discussed above. The only difference between this and (2) above is that threats are dynamic, whereas obstacles, ground slopes, etc. are static. While a system might build a representation of terrain regions once, exposed danger areas must be reconstructed each time a path is planned. For the cell approach, this means checking visibility from threats to the cells that are searched; for the weighted regions algorithm, it means checking the entire movement space for visibility to threats, constructing polygonal danger areas from the sample points, and tessellating the movement space into convex polygons before search can begin.

The strengths and weaknesses of the different approaches are summarized in Table 1.

Requirement	Planning Approach			
	Cell decomp.	**Skeleton**	**Weighted region**	**Potential Fn.**
Free path	Fair	Good	Good	Fair
Varying costs	Good	Poor	Good	Poor
Continuous variation	Good	Poor	Poor	Poor
Threats	Good	Poor	Poor	Poor

Table 1. Suitability of path planning approaches to soldier-agent planning requirements

From the table, it is apparent that there is no approach that is good for all the requirements, but the cell decomposition approach is at least fair for all. We have selected cell decomposition as the path planning mechanism. We did not elect to use the ModSAF short term planning algorithm or the ModSAF cover and concealment algorithm because of the limitations of the skeleton approach. We did use the graph search idea for building interiors that the current ModSAF algorithm uses, although we had to implement it differently to integrate it with the cell decomposition approach. Section Implementation describes how our implementation integrates the cell decomposition and building interior search and how it overcomes some of the shortcomings of the cell based approach.

4. Implementation

The individual tactical movement behavior uses a cell-based movement planner to find a good path through static terrain, and uses a reactive obstacle avoidance technique to avoid moving agents. The path planner can plan routes outside and inside MES buildings; it consider ground slope in path cost; it uses an adjustable, non-linear (with time) cost function for exposed areas; it adjusts posture to hide from threats if possible; and it adjusts speed to rush across exposed areas. This section describes the implementation of the path planning algorithm.

4.1 Path planning algorithm

4.1.1 Defining the search space

The search space is a grid of 1 meter squares no more than 250 meters long by 200 meters wide. It is oriented along the direction from start to goal, and is longer than that vector and about that wide. Thus the plan can go away from the goal to some degree, and deviate to the side a relatively large amount. If a path cannot be found in that area, the algorithm fails.

In general, the path planner finds paths by searching over the grid of cells. However, a 2-dimensional grid is not adequate for planning inside buildings; therefore we use a hybrid combination of a 2-dimensional grid and a 3-dimensional network for the search space. The 3D network is the topological network of rooms in the building, where each node in the network is a doorway between rooms. Outside of buildings, search nodes are cells are expanded to the 8 nearest neighbors in a rectangular grid; inside of buildings, nodes are doorways and are expanded directly to other doorways. The trick, of course, is doors to the outside; these special search nodes are expanded both to inside doorways and to outside grid cells.

Although the world location of a search node is implicit in the cell's grid position, the exact world location of the node is recorded in the search node data structure. This allows us to represent the exact location of search nodes that don't fall exactly on cell centers. In particular, while the search space is defined so that the start location fall on a cell center, the goal location and doorway locations may not on cell centers. Also, in the future a continuous relaxation process could move node locations off of grid points.

4.1.2 Populating the search space

Terrain obstacles, which are polygonal, are "drawn" onto the search grid using a standard line rasterizing algorithm. Obstacles that are no bigger than one cell are ignored; it is assumed that dynamic obstacle avoidance can handle these and that they shouldn't affect planning. As each obstacle cell is identified in the rasterization process, an obstacle node is created and stored in a hash table of "visited" nodes.

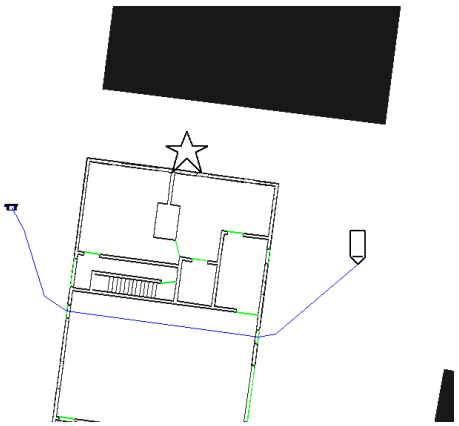


Figure 2. Path planned through building to avoid exposure to threat (star).

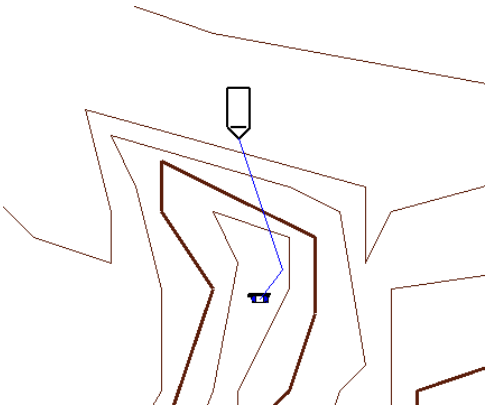
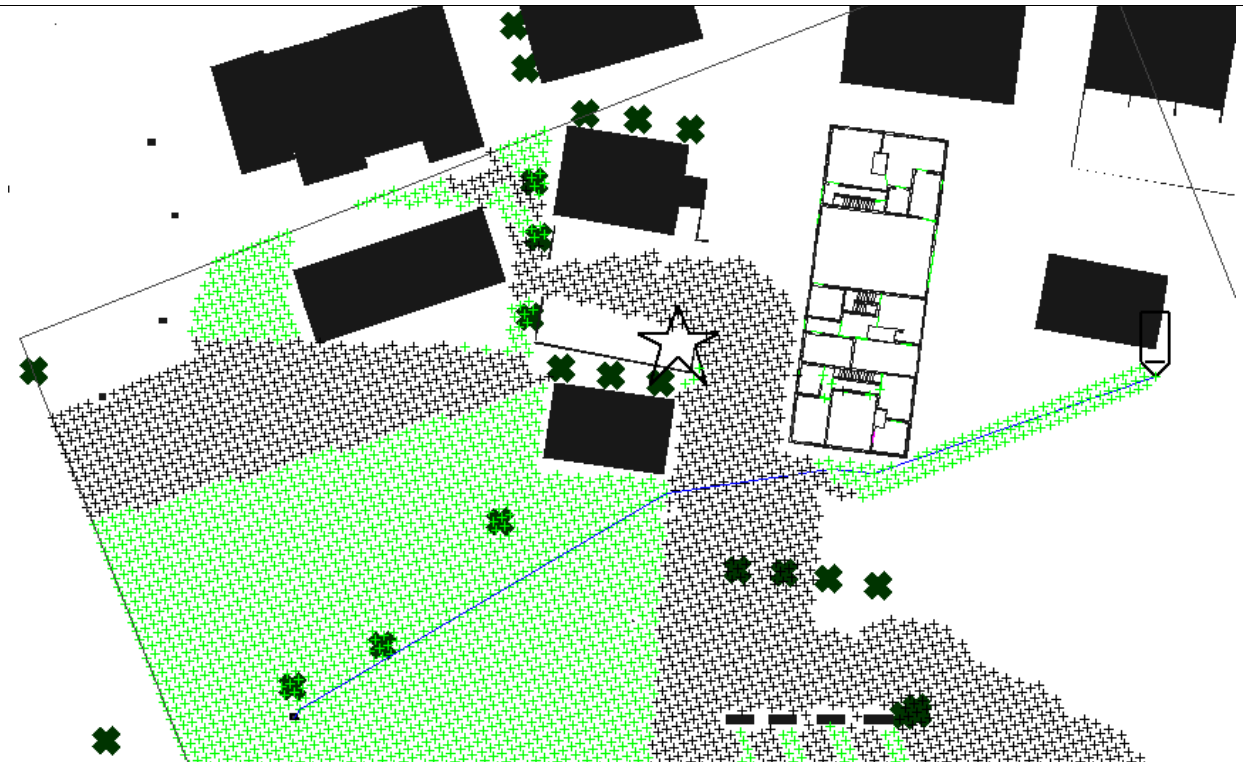


Figure 3. Path planned up a steep hill. The path goes laterally and then traverses the hill at an angle so that the effective slope is reduced.



After the obstacles are identified, all outside doors of MESs are identified. These doors create doorway search nodes which replace the obstacle nodes in the same location. No interior building walls are are drawn in the main grid, as the grid is too coars for planing inside the building.

Other than the obstacle, outside-door, and start location nodes, no other nodes in the search space are created at first. Line-of-sight checks and elevation lookups needed to compute path costs are only computed as needed when the search process visits a node.

4.1.3 Search

The search space is explored using the A* algorithm. As the search proceeds, the 3-dimensional distance from a node to the goal is used as the heuristic function. Using 3 dimensions is necessary to drive the search to the correct floor of a building. The cost function is based on the time needed. In particular:

- if the ground slopes up, the cost goes up inversely with the rate at which an individual's maximum speed would be decreased. This rate is determined by the DISAF physical model of human movement. Thus, a path up a slope costs more than a path that traverses a slope.
- crawling costs more than walking so that walking is preferred, although crawling is still better if standing would expose the agent to threats.

If the node is exposed to any enemy threats, it costs more to move to that node. We check visibility from threats assuming that the moving individual is standing, and if visible test again assuming the individual is prone. If it is visible while standing but concealed while prone, the node is marked as being concealed, but prone. The number of consecutive exposed nodes is used to set the cost of the next exposed node; very short exposed runs (i.e., about a half second) cost little more than concealed moves, but as the runs get longer the cost becomes much (e.g. 50 times) higher.

Although some movement planning algorithms set the cost of exposed movement to be infinitely higher than that of concealed movement, using these costs in a grid search would result in all concealed nodes being explored before an exposed node is explored at all. Making the cost of short exposures relatively small improves search performance considerably. Making the cost of exposure a search parameter allow the planner to produce different tactical plans; for example, setting the cost of exposure very high produces the most stealthy plan, while reducing it to zero produces a path that considers speed only.

4.1.4 Path post processing

After the path has been found, it is post-processed to improve it. First, exposed segments of the path are grown by one node (i.e., the nodes before and after the exposed segment are marked as exposed) so that the entity will have a chance to accelerate to a rushing speed before being exposed. Next, segments requiring the agent to be prone are grown by one node so that the agent will be sure to be prone before it enters the area where it needs to be prone. Third, standing segments between prone segments are checked to see how long they are. Short standing segments are replaced by prone segments so that the agent does not waste time standing up for just a short segment. If the standing segment is exposed, the distance threshold is increased so that the agent will more likely stand and run.

Finally, the path is examined to see if uniform segments—segments of the path that don't change direction more than 45 degrees, have the same exposure and same posture—can be reduced to a straight line. Segments can be reduced if the straight line from the start to the end does not cross any of the polygonal obstacles or exposed areas (for concealed segments). If it does, a binary search is used to find an intermediate point that a straight line can be drawn to, and any points between the start and intermediate point are removed from the path plan.

This latter plan processing step is a coarse form of relaxation. Relaxation can improve a grid-based path plan in different ways. Krogh [14], for example, showed how small perturbations to the node location could optimize and smooth the path of a robot avoiding obstacles. In the implementation of grid search described here, none of the node cost function terms varies in a way that would allow optimization by using small perturbations. We may add such terms in the future, and at that time replace the path-straightening relaxation by an iterative smoothing procedure.

4.2 Path planning results

Figure 1 shows a search from a point on the left to a point on the right; the star is a threat location. The light colored crosses represent concealed locations, while the black crosses show exposed locations. The line shows the resulting path. Because crossing the exposed area is relatively expensive, the search

covers a wide area to the left and the right of the eventual path before deciding to cross the exposed area at its smallest point. The small rectangles in the bottom right of the figure are small obstacles only about a meter high; the concealed nodes below them require the individual to be prone.

Figure 2 shows the route planned by an individual from one side of a building to another; because of the threat in the alley, the path was planned through the building.

Figure 3 shows a short move up a hill. For this example, the cost of moving up a slope was increased artificially high. The lines in the figure are contour lines. The computed path does not go straight to the destination but bends so that it can climb the hill at an angle and have a lower effective slope.

5. Conclusion

We have developed a tactical movement behavior for individual soldiers. Functionally it performs path planning around obstacles and through buildings, staying concealed from threats and avoiding difficult terrain as much as possible.

In order to select the best algorithm for planning tactical movement we had to first determine all of the relevant requirements. We found that several previously implemented approaches to movement planning were not appropriate given the functional requirements, but we were able to develop and implement an approach that satisfies the requirements. The lesson we learned is that if a particular approach seems to be very powerful, but it fails to satisfy one or more parts of the problem, then it must be augmented or replaced by another approach.

There are several directions for future work in this system. Path relaxation could be introduced so that it could adjust and optimize a path created using more sophisticated evaluation functions. Our vision for representing paths is to move away from geometric waypoints and toward action-based waypoints such as “go through door” or “move to tree.” We have had some success using the same unit planner with a larger cell size to perform unit planning; this process could be developed more to support different travelling modes and to allow switching to single file formations in restricted terrain.

6. Acknowledgements

This work was performed as part of contract N61339-99-C-0007 awarded to SAIC by the U.S. Army’s Simulation, Training and Instrumentation Command (STRICOM).

7. References

[1] Mitchell, J. S. B. “An Algorithmic Approach to Some Problems in Terrain Navigation.” *Artificial Intelligence* 37: 171-201. (1988).

[2] Hwang, Y.K. and Ahuja, N. “Gross Motion Planning—A Survey.” *ACM Computer Surveys* Vol. 24, No. 3, pp 219-291. (1992)

[3] Tyler, J., Booker, L., Brisbols, G. and Canova, B. “Route Planning for Individual Combatants Using Genetic Algorithms.” In *Proceedings of the Spring Simulaiton Interoperability Workshop.* University of Central Florida. (1997)

[4] Kambhampati, S. and Davis, L. “Mutliresolution Path Planning for Mobile Robots.” In *IEEE Journal of Robotics and Automation,* Vol RA-2, No. 3. (1986)

[5] Yahja, A., Stentz, A., Singh, S., and Brummit., B. “Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments.” In *Proceedings, IEEE Conference on Robotics and Automation.* (1998)

[6] Karr, C. and Rajput, S. “Unit Route Planning.” In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation.* University of Central Florida. (1995).

[7] Smith, J. “Near-term Movement Control in ModSAF.” In *Proceedings of the 4th Conference on Compter Generated Forces and Behavioral Representation.* University of Central Florida. (1994)

[8] Campbell, C., Hull, R., Root, E., and Jackson, L. “Route Planning in CCTT” In *Proceedings of the 5th Conference on Computer Generated Forces and Behavioral Representation.* University of Central Florida. (1995)

[9] Stanzione, T., F. Chamberlain, L. Mabius, M. Sousa, A. Evans, C. Buettner, J. Fisher, and H. Lu. “Multiple Elevation Structures in the Improved Computer Generated Forces

Terrain Database." In *Proceedings of the 6th Conference on Computer Generated Forces and Behavioral Representation.* University of Central Florida. (1996).

[10] Longtin, M. and Megherbi, D. "Concealed Routes in ModSAF." In *Proceedings of the 5th Conference on Computer Generated Forces and Behavioral Representation.* University of Central Florida. (1995)

[11] Mitchell, J.S. B. and Papadimitriou, C. H. "The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision." *Journal of the Association for Computing Machinery*, 38 (1): 18-71. (1991)

[12] Hoff, B., Howard, M.k and Tseng, D. "Path Planning with Terrain Utilization in ModSAF" In *Proceedings of the 5th Conference on Computer Generated Forces and Behavioral Representation.* University of Central Florida. (1995)

[13] Kim, J-O. and Khosla, P. "Real-Time Obstacle Avoidance Using Harmonic Potential Functions" *IEEE Transactions on Robotcs and Automation* Vol. 8(3): 338-349. (1992)

[14] Krogh, B. and Thorpe, C. "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles." In *Proceedings of the IEEE Conference on Robotics and Automation*. (1986)

8. Author Biographies

DOUGLAS REECE is a Senior Scientist at SAIC in Orlando. He has been developing physical and behavioral models for individual combatant CGFs for four years. He was the principal investigator on the project to develop Computer Controlled Hostiles for the Marine Corps' Team Target Engagement Simulator, and now is the software architect for DISAF. He received his Ph.D. in Computer Science from Carnegie Mellon University in 1992.

MATT KRAUS is the Project Director for DISAF at SAIC. He has a Bachelor of Science degree in Computer Science from Western Michigan University and a Master of Science degree in Simulation Modeling and Analysis from the University of Central Florida. Prior to his work with SAIC he worked as an engineer and ModSAF instructor with the Institute for Simulation and Training and as a systems engineer with PIXAR. His research interests are in the areas of distributed computing, artificial intelligence and computer graphics.

PAUL DUMANOIR is the lead systems engineer for Individual Combatant (IC) simulations at the U.S. Army STRICOM. Mr. Dumanoir leads the DoD Defense Technology Objective (DTO) for IC and Small Unit Operations Simulation, and is involved with the IC Science & Technology Objective (STO). Prior to his involvement with IC simulations, he worked as software and systems engineer on the various M&S programs. His current interests include IC CGFs and Human-In-The-Loop (HITL) networked simulators. He earned his B.S. in Electrical Engineering from the University of South Alabama in 1987 and his M.S. in Computer Systems from the University of Central Florida in 1991.