

1-1-1994

Unit Route Planning

Sumeet Rajput

Clark R. Karr

Find similar works at: <https://stars.library.ucf.edu/istlibrary>
University of Central Florida Libraries <http://library.ucf.edu>

This Research Report is brought to you for free and open access by the Digital Collections at STARS. It has been accepted for inclusion in Institute for Simulation and Training by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

Recommended Citation

Rajput, Sumeet and Karr, Clark R., "Unit Route Planning" (1994). *Institute for Simulation and Training*. 204.
<https://stars.library.ucf.edu/istlibrary/204>

Contract Number N31339-92-C-0045
December 21, 1994

Unit Route Planning

Written by
Sumeet Rajput and Clark Karr



IST

Institute for Simulation and Training
3280 Progress Dr.
Orlando, FL 32826

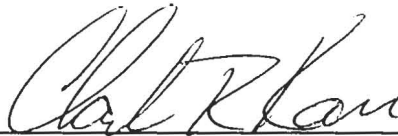
University of Central Florida
Division of Sponsored Research

Unit Route Planning

Written by
Sumeet Rajput and Clark Karr

December 21, 1994

IST-CR-94-42



Submitted by: Clark Karr



Reviewed by: Rhonda Freeman

Unit Route Planning

IST-CR-94-42

Contract N61339-92-C-0045
December 21, 1994

Sumeet Rajput and Clark R. Karr

Table of contents

1. EXECUTIVE SUMMARY	1
2. INTRODUCTION	2
2.1 PURPOSE.....	2
2.2 BACKGROUND.....	2
2.2.1 <i>Distributed Interactive Simulation</i>	2
2.2.2 <i>Computer Generated Forces</i>	2
3. ROUTE PLANNING	3
3.1 GENERAL PATH PLANNING APPROACHES.....	4
4. UNIT ROUTE PLANNING	6
4.1 TERRAIN GRID	6
4.2 OBSTACLE SEGMENT ABSTRACTION	7
4.2.1 <i>Creating Obstacle Segments</i>	8
4.2.2 <i>Example</i>	9
4.3 CREATING ROUTES.....	10
4.3.1 <i>A* Algorithm</i>	10
4.3.2 <i>Cost factors</i>	10
4.4 ADVANTAGES OF THE OBSTACLE SEGMENT ABSTRACTIONS.....	11
5. THE UNIT ROUTE PLANNING ALGORITHM	12
5.1 ALGORITHM PLANROUTE.....	12
5.2 GRID GRANULARITY AUTO-DETECTION	13
5.3 ALGORITHM DETAILS	13
5.3.1 <i>Marking Obstacle Segment Abstractions in Grid Cells</i>	13
5.3.2 <i>Sample points</i>	15
5.3.3 <i>Adjusting start and destination points</i>	16
5.3.4 <i>Movement between grid cells</i>	18
5.3.5 <i>Extended Obstacle Segments</i>	20
5.3.6 <i>Determining Reachable Points</i>	21
5.3.7 <i>Computation of route cost</i>	22
6. RESULTS	25
6.1 GRID GRANULARITY 5 METERS.....	25
6.2 GRID GRANULARITY 85 METERS.....	28
6.3 GRID GRANULARITY 85 METERS WITH HOSTILE UNITS	31
6.4 GRID GRANULARITY 500 METERS.....	34
6.5 GRID GRANULARITY 500 METERS WITH TUNNEL	37
7. CONCLUSIONS AND FUTURE WORK	40
8. REFERENCES	41
9. GLOSSARY	43

List of Figures

FIGURE 4.1-A NEIGHBORING CELLS	6
FIGURE 4.2-A OBSTACLE SEGMENTS	7
FIGURE 4.2.1-A AN OBSTACLE THAT IS REPRESENTED BY AN OBSTACLE SEGMENT	8
FIGURE 4.2.1-B A CELL WITH A HORIZONTAL OBSTACLE SEGMENT	8
FIGURE 4.2.1-C AN OBSTACLE THAT IS NOT REPRESENTED BY AN OBSTACLE SEGMENT	8
FIGURE 4.2.2-A NOTIONAL TERRAIN	9
FIGURE 4.2.2-B OBSTACLE SEGMENT ABSTRACTION GRID.....	9
FIGURE 5.3.1.1-A BOUNDING BOX OVERLAP DOES NOT GUARANTEE AN OBSTACLE SEGMENT	14
FIGURE 5.3.2-A SAMPLE POINTS	15
FIGURE 5.3.3-A START POINT ADJUSTMENT	16
FIGURE 5.3.3-B FLOODFILLING THE GRID CELL	17
FIGURE 5.3.4.1-A A SHARED POINT.	18
FIGURE 5.3.4-A ACCESS TO REGION BETWEEN PHYSICAL OBSTACLES IN TWO CELLS	19
FIGURE 5.3.4-B THE CORRESPONDING OBSTACLE ABSTRACTIONS	19
FIGURE 5.3.4-C DISPLACED OBSTACLE SEGMENTS.....	20
FIGURE 5.3.5-A DIAGONAL MOVE BLOCKED BY ABSTRACT OBSTACLE IN AN ORTHOGONAL CELL.....	20
FIGURE 5.3.5-B EXTENDED OBSTACLE SEGMENT.....	21
FIGURE 5.3.6-A MUTUALLY REACHABLE SETS.....	22
FIGURE 6.1-A 5 METER GRID: TERRAIN, START, AND DESTINATION POINTS	25
FIGURE 6.1-B 5 METER GRID: OVERLAID GRID AND OSAS	26
FIGURE 6.1-C 5 METER GRID: FINAL ROUTE.....	27
FIGURE 6.2-A 85 METER GRID: UNDERLYING TERRAIN WITH START AND DESTINATION	28
FIGURE 6.2-B CANOPY AND RIVERS OSAS, START, AND DESTINATION	29
FIGURE 6.2-C 85 METER GRID: THE FINAL ROUTE	30
FIGURE 6.3-A 85 METER GRID WITH HOSTILE UNITS: TERRAIN, START AND DESTINATION, AND HOSTILE UNIT LOCATIONS.....	31
FIGURE 6.3-B 85 METER GRID WITH HOSTILE UNITS: OVERLAID GRID AND OSAS	32
FIGURE 6.3-C 85 METER GRID WITH HOSTILE UNITS: THE FINAL ROUTE.....	33
FIGURE 6.4-A 500 METER GRID: UNDERLYING TERRAIN, START AND DESTINATION	34
FIGURE 6.4-B 500 METER GRID: OVERLAID GRID AND OSAS.	35
FIGURE 6.5-A 500 METER GRID WITH TUNNELS: UNDERLYING TERRAIN, START, AND DESTINATION.....	37
FIGURE 6.5-B 500 METER GRID WITH TUNNELS: OVERLAID GRID AND OSAS.....	38
FIGURE 6.5-C 500 METER GRID WITH TUNNELS: FINAL ROUTE	39

List of Tables

TABLE 9-A GLOSSARY.....	43
-------------------------	----

1. Executive Summary

Route planning occurs at multiple levels within CGF systems. At the lowest level, routes for individual vehicles are prepared that allow vehicles to move from one point to another. At levels above the vehicle, routes are prepared for groups of vehicles, i.e. units (platoons, companies, battalions). Unit routes occur within "movement corridors" reflecting the fact that, formations and tactics aside, the vehicles' routes are only constrained to be within the corridor. As the hierarchy of units is ascended, the unit's size and therefore the width of the corridor increases.

This report presents a new Unit Route Planning Algorithm based on a novel abstraction called the Obstacle Segment Abstraction (OSA). This Unit Route Planning Algorithm combines the OSA with two route planning approaches, regular grid and vertex graph. A regular grid overlays the terrain. The size and location of the grid is determined by the unit boundaries of the unit planning the route. The scale of the grid is determined by the unit size; a grid scale less than half the typical "frontage" of the unit gives good results. Obstacles on the terrain are encoded in the grid using the OSA. An efficient search algorithm, A*, is applied to the grid to determine the optimal route. Factors in addition to distance are incorporated into the route cost determination to introduce trafficability and cover and concealment in the evaluation of the potential routes.

The OSA is a space efficient representation of obstacles within a polygonal terrain database. Calculations show that the standard SIMNET Ft. Knox Terrain Database (TDB) can be completely represented at the 500 meter grid level as an OSA grid in approximately 200 Kbytes of memory.

The Unit Route Planning Algorithm has several strengths:

1. It is scalable. The size of the underlying grid is based on the size of the unit.
2. It is time and space efficient. An efficient search algorithm, A*, is used to determine routes. The space requirements for the OSA are relatively small.
3. Three factors (distance, trafficability, and cover and concealment) are considered in evaluating candidate routes.
4. Multiple distinct routes can be generated between two locations on the terrain.
5. Chokepoints are identified.
6. It can be used to efficiently find and refine lengthy, precise routes through a process of successive refinement.
7. It can be utilized as an efficient vehicle route planner with the addition of a simple route smoothing algorithm.

This report documents several experiments of route planning at different unit sizes. For example, an optimal 10 km route was determined in approximately 5 seconds on a 60 MHz 486 PC after the grid had been filled with Obstacle Segment Abstractions.

2. Introduction

2.1 Purpose

This technical report is a deliverable under STRICOM contract N61339-92-C-0045, "Intelligent Autonomous Behavior by Semi-Automated Forces in Distributed Interactive Simulation". It is one of two reports satisfying CDRL A007 "Route Planning Technical Report"; the second report is entitled "Dynamic Obstacle Avoidance", IST-TR-94-41.

Its purpose is to report the methodology and results of experimental software development conducted at the Institute for Simulation and Training (IST) under that contract. The goal of the software development was to produce an efficient and effective mechanism whereby battalion, company, and platoon command entities controlled by Computer Generated Forces systems can efficiently generate routes with minimal length and exposure to enemy fire.

2.2 Background

This section provides a brief description of Distributed Interactive Simulation (DIS) and Computer Generated Forces (CGF). It may be skipped by readers familiar with these topics.

2.2.1 Distributed Interactive Simulation

Distributed Interactive Simulation is an architecture for building large-scale simulation models from a set of independent simulator nodes (DIS[1993]). The simulator nodes are linked by a network and communicate via a common network protocol. (The term DIS is also sometimes used to designate a particular network protocol standard; in this document "DIS" refers to the simulation architecture; the DIS protocol standard will be so identified.) In DIS, the simulator nodes each independently simulate the activities of one or more entities in the simulated system and report their attributes and actions of interest to other simulator nodes via the network. The simulated entities coexist in a common environment (for example, a terrain database) and interact by exchanging network packets (Loper et. al.[1991]). Finally, an important characteristic of DIS simulations is that they are real-time; events in the simulation occur at the same rate as their real-world counterparts.

2.2.2 Computer Generated Forces

DIS environments are designed to provide a simulated battlefield which is used for training military personnel. In such a battlefield, the trainees need an opposing force against which to train. One technique is to use a computer system that generates and controls multiple simulation entities using software and possibly a human operator. This type of system is known as a semi-automated force (SAF or SAFOR) or a computer generated force (CGF).

A CGF system will use built-in behavior to react autonomously to the simulation situation or to carry out orders given by its operator. Its behavior may be encoded as algorithms, production rules, formal behavior specifications, or some other form. The intent is for the CGF system's behavior to be autonomous (i.e. not requiring human control) and realistic (i.e. true to doctrine, physics, and human responses) to the greatest possible extent.

Under the sponsorship of ARPA and STRICOM, IST has been conducting research in the area of CGF systems, seeking to increase the realism and autonomy of CGF behavior. A key product of that sponsorship is the IST CGF Testbed. The IST CGF Testbed is a CGF system that provides an environment for testing CGF behavioral control algorithms. It is documented in Danisas et. al. [1990], Gonzalez et. al.[1990], Petty [1992], Smith et. al. [1992a], and Smith et. al. [1992b].

3. Route Planning

Route planning occurs at multiple levels within CGF systems. At the lowest level, routes for individual vehicles are prepared that allow vehicles to move from one point to another. Vehicle routes typically consist of a series of line segments. These piecewise linear routes are represented as a list of points (called "route points") and the vehicle is expected to travel in a straight line between route points. Craft et.al. [1994] describes a mechanism for avoiding moving obstacles while traversing a route.

At levels above the vehicle, routes are prepared for groups of vehicles, i.e. units (platoons, companies, battalions). As the hierarchy of units is ascended, the unit's size and therefore the width of the route increases. Unit routes occur within "movement corridors" reflecting the fact that, formations and tactics aside, the vehicles' routes are only constrained to be within the corridor. A corridor has sufficient width for a unit to move through it. Figure 3-A illustrates a network of movement corridors between two locations on some notional terrain.

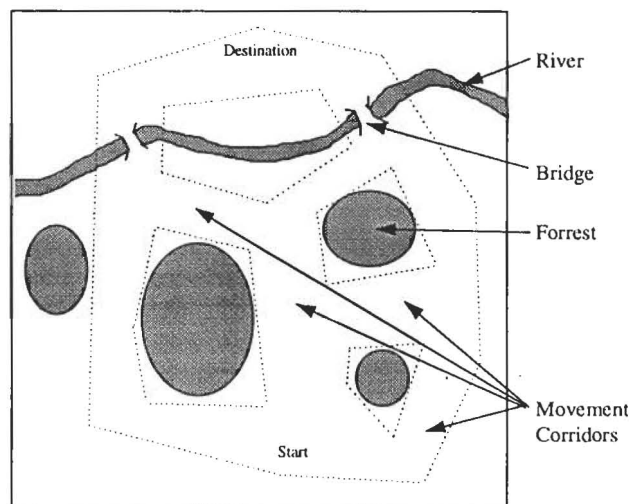


Figure 3-A Movement Corridors

Planning routes for units of differing sizes may seem a simple matter of generating a route for a single vehicle and treating that route as the center of a corridor. While feasible, this approach retains the complexities of creating precise vehicle routes. For example, a route for a vehicle would need to avoid small static obstacles such as buildings. In contrast, a unit route or corridor could ignore the buildings with the understanding that the vehicles within the unit would individually avoid them. Similarly, a vehicle level route would need to specify precisely the path to cross a bridge. A unit route would simply need to enclose the bridge. The complexities of vehicle level route planning increase the computational expense/time above that required for unit route planning.

This report details research into route planning for military units. A novel approach based on the combination of three disparate route planning approaches is presented that:

1. is scaleable, suitable for battalion through vehicle route planning,
2. computationally fast,
3. considers distance, cover and concealment, and trafficability,
4. generates multiple acceptable routes between points within unit boundaries, and
5. finds and reports chokepoints within the routes.

This research has been done in conjunction with the development of a CGF Automated Mission Planner capability. To generate and evaluate multiple courses of action to fulfill a mission, the Mission Planner requires multiple, tactically sound unit routes and the identification of chokepoints along the routes (Lee [1994]).

3.1 General Path Planning Approaches

Motion planning with particular emphasis on robot path planning and robot manipulator path planning has seen considerable work, see Hwang et. al. [1992] for a survey. There are four broad categories of path planning approaches: free space analysis, vertex graphs analysis, potential fields, and grid (regular tessellation) based algorithms (Thorpe [1984]). Each approach has strengths and weaknesses.

In the free space approach, only the space not blocked or occupied by obstacles is represented. For example, representing the center of movement corridors with Voronoi diagrams (Roos et. al. [1991]) is a free space approach. Although Voronoi diagrams are efficient representations, they and other free space approaches have some deficiencies. First, they tend to generate unrealistic paths. Paths derived from Voronoi diagrams follow the center of corridors while paths derived from visibility graphs (Mitchell [1988]) clip the edges of obstacles. Second, the width and trafficability of corridors are typically ignored. Third, distance is generally the only factor considered in choosing the optimal path.

In the vertex graph approach, the endpoints, vertices, of possible path segments are represented (Mitchell [1988]). This approach is suitable for spaces that have sufficient obstacles to determine the endpoints; determining the vertices in "open" terrain is difficult. In addition, representing only path vertices creates three other difficulties. First, trafficability over the path segments is not represented; route segments between arbitrary vertices are typically "open" or "blocked". Second, factors other than distance can not be included in evaluating possible routes. In the military simulation domain, concealment and cover are important factors in route planning. Third, because the width of route segments is not represented one of two problems occurs. Chokepoints (narrow sections of routes) are marked "blocked" or "open". If "blocked", acceptable routes are discarded. If "open", unacceptably long, narrow routes are accepted. Thus, the vertex graph approach has difficulty representing route width.

In the potential field approach, the goal (destination) is represented as an "attractor", obstacles are represented by "repellers", and the vehicle is pulled toward the goal while being repelled from the obstacles (NASA [1991]). There are three difficulties with the potential field approach. First, the vehicles can be attracted into box canyons from which they can not escape (Mitchell [1988]). Second, some elements of the terrain may simultaneously attract and repel. For example, an obstacle to movement, a repeller, may create an area of concealment. A vehicle should be attracted to the obstacle for concealment while being repelled from the obstacle creating the "visibility shadow". Third, route width is not considered.

In the regular grid approach, a grid overlays the terrain, terrain features are abstracted into the grid, and the grid rather than the terrain is analyzed. Each grid cell is typically marked as "open" or "blocked". Quadtrees are an example of the regular grid approach (Mitchell [1988]). Grid routes are converted into terrain routes typically by adding the z-coordinate to the xy-coordinates in the grid route. This approach's advantage is to simplify the analysis but has two disadvantages. First, "jagged" paths are produced because movement out of a grid cell is restricted to four (or eight) directions corresponding to the four neighboring cells (eight if diagonal moves are allowed). Second, the granularity (size of the grid cells) determines the smallest "opening" that can be identified. If the granularity is too large, small openings in

obstacles (e.g. bridges over rivers) are lost. To capture the small openings, a small granularity is required which increases the computational expense of the analysis.

4. Unit Route Planning

The Unit Route Planning algorithm presented here combines a unique obstacle abstraction with two route planning approaches, regular grid and vertex graph. A regular grid overlays the terrain. The size and location of the grid is determined by the unit boundaries of the unit planning the route. The scale of the grid is determined by the unit size; grid scales less than the typical "frontage" of the unit give good results. For example, grid scales from 75 to 125 meters are suitable for platoons, 250 to 500 meters for companies, and 750 to 1000 meters for battalions. Obstacles on the terrain are encoded in the grid using a novel abstraction called the Obstacle Segment Abstraction (OSA). An efficient search algorithm, A*, is applied to the grid to determine the optimal route. Factors in addition to distance are considered in determining route cost to incorporate trafficability and cover and concealment in the evaluation of the potential routes.

4.1 Terrain Grid

CGF systems operating within DIS type environments rely on a representation of terrain termed a Terrain Database. This research was performed using the SIMNET terrain database (TDB) which encodes terrain and terrain features as polygons; these are encoded in turn in edge and vertex lists. This TDB contains features such as treelines, canopies, rivers, and lakes. These features are the physical obstacles. The Unit Route Planning algorithm is applicable to any polygonal TDB format (e.g., ModSAF's CTDB (Smith [1994])) and to any TDB format that represents obstacles as distinct features.

When a route is requested, a regular grid is laid over a portion of the terrain. The location and size of the grid are determined by the unit boundaries and the orientation of the grid is determined by the orientation of the destination to the starting location. The granularity of the grid (the size of each grid cell) is determined by the unit size. Each grid cell has eight neighboring grid cells. The cells to the north (N), south (S), east (E), west (W) are called the orthogonal cells and the cells to the northeast (NE), northwest (NW), southeast (SE), and southwest (SW) are called the diagonal cells.

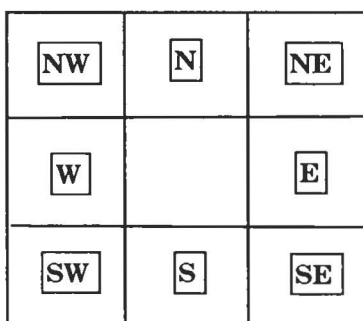


Figure 4.1-A Neighboring cells

4.2 Obstacle Segment Abstraction

The terrain underlying the grid is analyzed and each obstacle is encoded in the grid. The encoding uses a small set of linear segments called the Obstacle Segments (OSs). The different types of OSs are:

- Horizontal and vertical
- Diagonal and
- Tunnel

as shown in Figure 4.2-A.

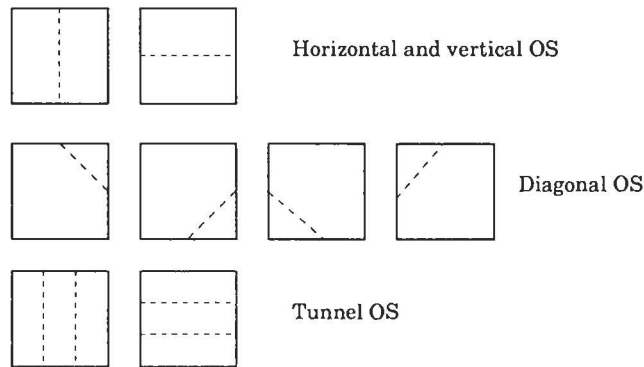


Figure 4.2-A Obstacle Segments

Each OS is identified with an Obstacle Identification Number (OIN). The OSs representing a single terrain feature are assigned the same OIN. Thus, each OSA represents a single terrain feature and is the set of OSs with the same OIN.

Treating physical obstacles as OS abstractions is the basis for OSA route planning. The precise polygonal details of the obstacle are dispensed with and an encoded representation is used in its place. As will be seen, the grid granularity determines the “correlation” error between the feature and its abstraction. So long as the granularity reflects the unit size, the correlation error is not a significant issue.

4.2.1 Creating Obstacle Segments

The intersections of terrain obstacles and the edges of the grid cells are determined and converted to OSs. An OS is created between the sides of the two entry points of an obstacle into a grid cell:

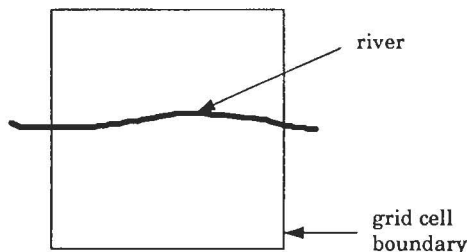


Figure 4.2.1-A An Obstacle that is represented by an Obstacle Segment

The portion of the river in Figure 4.2.1-A is represented by a horizontal OS in Figure 4.2.1-B.

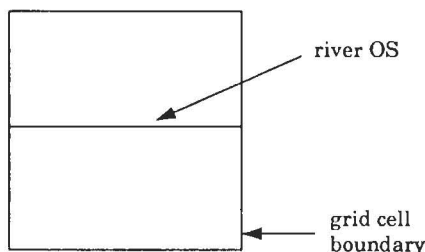


Figure 4.2.1-B A cell with a Horizontal Obstacle Segment

An obstacle that does not exit a cell does not impose a barrier to travel within the cell; a vehicle can simply move around it. Thus, an OS is not created for a feature that enters but does not exit a cell:

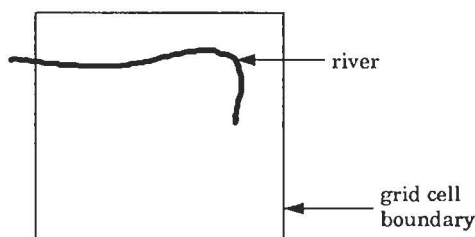


Figure 4.2.1-C An Obstacle that is not represented by an Obstacle Segment

The portion of the river in Figure 4.2.1-C is not represented by an OS.

Obstacles with width (e.g., rivers and lakes) will sometimes be represented by more than one OS in a grid cell. This occurs when the edges of the obstacle cross different grid cell boundaries. The example in section 4.2.2 illustrates this phenomenon.

4.2.2 Example

Figure 4.2.2-A shows some notional area of terrain and Figure 4.2.2-B shows the resulting OSA grid.

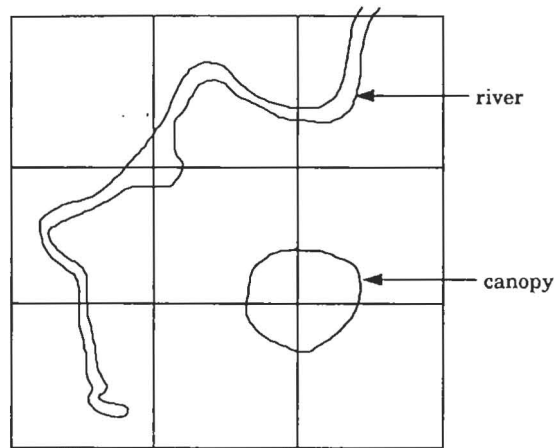


Figure 4.2.2-A Notional Terrain

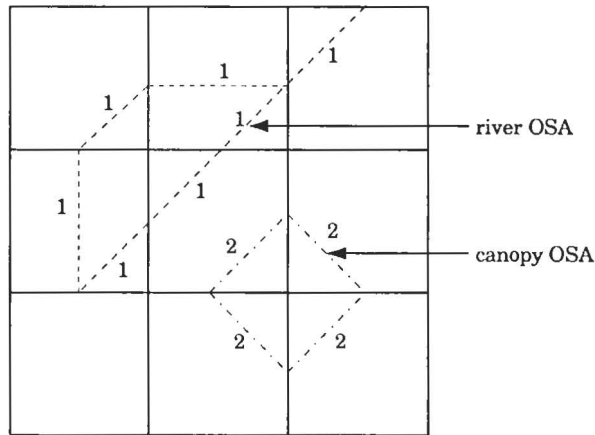


Figure 4.2.2-B Obstacle Segment Abstraction Grid

The OSs representing each obstacle are assigned unique OINs. In this example, the river's OSs are assigned the id "1" and the canopy's OSs are assigned the id "2". The river crossing the corner of a grid cell causes the river's OSA representation to include two cells with multiple OSs.

4.3 Creating Routes

Route planning within the OSA grid is a matter of *searching* the grid for optimal routes. The search begins at the start location and partial routes are extended into each of the neighboring grid cells. Then the partial path which seems “best” is extended. Extending a path creates several new partial paths, one for each neighboring cell at the end of the partial path. These partial paths are evaluated and placed on the list of candidate partial paths. The process of extending the “best” partial path continues until the destination is reached.

The search through the grid utilizes the vertex graph approach to route planning. Within each grid cell, “sample” points (see Section 5.3.2) are isolated. Beginning at the start location, a partial route to each neighboring cells’ sample points is created provided that the partial route does not cross an Obstacle Segment. In turn, each partial route is extended. This is a graph search problem. An efficient algorithm for performing graph searches is A*, Winston [1992]. The A* algorithm’s efficiency results from extending the lowest cost partial path. Undesirable, costly partial paths are ignored in favor of less costly partial paths. The Unit Route Planner calculates route cost from distance, trafficability, and cover and concealment.

4.3.1 A* Algorithm

The A* procedure is a *branch-and-bound* search combined with, first, an estimate of the cost of the remaining route and, second, the *dynamic-programming* principle, Winston [1992]. This estimate of the remaining distance is called the “underestimate”. When the underestimate is a lower-bound on the actual distance, A* produces optimal solutions. A natural underestimate in route planning is the linear distance from the last point on a partial route to the destination. The dynamic programming principle improves search efficiency by retaining only the best partial solution to each point for further analysis.

4.3.2 Cost factors

The cost of a route depends upon:

- length,
- concealment from enemy positions, and
- trafficability over the route.

As the length of a route increases, its cost also increases. Route segments that are not concealed from enemy positions increase total route cost reflecting increased “danger” along those route segments. Poor trafficability similarly increases the cost of route segments. For this work, terrain slope and soil type governed trafficability. Flat terrain was the least costly. Soil types were given costs in relation to the ease of travel over them. Route segment cost was computed through a set of weight factors and equations described in detail in Section 5.3.7.

The Unit Route Planner’s underestimate is the linear distance between the last point on the partial route and the destination considering the rest of the route to be concealed and on flat terrain. This is an underestimate because the least costly route possible between two points is a straight line with maximal concealment over the most easily traversed soil.

4.4 Advantages of the Obstacle Segment Abstractions

The OSA combines the strengths of the vertex graph and the regular grid approaches and solves or alleviates their problems. Consider the shortcomings of the regular grid approach. First, the OSA encoding of obstacles within the grid is more sophisticated than the typical “open” or “blocked” approach. This encoding allows the interiors of cells containing obstacles to be considered for route segments. This removes the restriction that grid granularity is dictated by the smallest opening in the terrain. Second, within each grid cell, candidate route vertices called “sample points” are identified. The introduction of sample vertices into the regular grid solves the “jagged” path problem of regular grids. Consider the shortcomings of the vertex graphs approach. First, determining vertices in open terrain is not a problem because vertices, sample points, are found within many grid cells. Second, trafficability and cover and concealment are encoded in the grid cell and are included in the evaluation of candidate routes.

5. The Unit Route Planning Algorithm

This section discusses the Unit Route Planning Algorithm that was implemented in the IST CGF Testbed. Specifically, Section 5.1 presents the *PlanRoute* algorithm. Section 5.2 discusses how the grid granularity can be set to the optimum value. Finally, Section 5.3 presents the details of the algorithm implementation.

5.1 Algorithm PlanRoute

Algorithm *PlanRoute* plans a route between two points, the start and the destination. The details of *PlanRoute* are explained in following sections.

Algorithm PlanRoute

Input:

The grid to route on, the start, and destination points.

Output:

A list of points defining the route.

Variables:

<i>route_list</i>	A list of routes, sorted in ascending order of route cost. The first route on this list is referred to as <i>route_list₁</i> .
<i>route_to_be_extended</i>	The least costly (and hence the first) route on the <i>route_list</i> . It is the same as <i>route_list₁</i> .
<i>reachable_points</i>	A list of points that are reachable from the last point on <i>route_to_be_extended</i> .

1. [Create the grid]
2. [Fill the grid]
3. [Initialize *route_list*]
route_list = empty
4. [Are we there?]
If the first route on the *route_list* terminates at the destination then
 - 4.1 Return *route_list₁*,else
 - 4.2 [Extend least costly partial route]
 - 4.2.1 *route_to_be_extended* = *route_list₁*.
 - 4.2.2 Determine the points reachable from the last point on *route_to_be_extended* (described in Section 5.3.6).
reachable_points = {*r₁*, *r₂*, ..., *r_n*}.
 - 4.2.3 Expand *route_to_be_extended* to each of the points in *reachable_points* if another, less expensive route to those points does not exist.
 - 4.2.4 Add these new routes to *route_list* in ascending order of route cost and underestimate.
 - 4.2.5 Go to step 4.

Steps 1 and 2 in the algorithm create the grid and populate it with abstract obstacles (see Section 5.3.1 for details on converting polygonal features to OSs). Step 3 initializes the *route_list* to empty. Step 4 finds the route.

In step 4, a check is made to determine if *route_to_be_extended* has reached the destination. If so, the result is sent to the caller; otherwise, *route_to_be_extended* is extended.

Step 4.2 extends the least costly partial route. The first route on the *route_list*, *route_list*, is removed and assigned to *route_to_be_extended*. All the points that are reachable from the end of this route are determined and stored in the list *reachable_points* (see Section 5.3.6 to see how reachable points are determined.) The route is extended to each *reachable point* only if another, less costly route does not exist. These “extended” routes are added to the *route_list* in sorted order and step 4 is repeated.

5.2 Grid granularity auto-detection

It is easy to see that the density of obstacles on portions of terrain can exceed the capacity of the OSA grid to represent them at certain grid sizes. When the granularity of the grid is too coarse for the density of obstacles, the Unit Route Planner attempts to create too many identical OSs. For example, the Planner may attempt to create two NE diagonal OSs in a single grid cell. The Unit Route Planner detects when this occurs and prevents analysis from continuing at the same granularity. There are at least two solutions to the problem of a too coarse grid granularity. First, the algorithm can simply stop and replan the entire route at a finer granularity. This is the simplest solution conceptually. The algorithm simply “halves” the grid width and starts over. The second solution is similar to the quadtree representation. In this approach only the granularity of the grid cell that is too coarse is increased. This approach avoids the computational expense of repeated replanning but is more complex. For this work, the first approach was chosen.

5.3 Algorithm details

Sections 5.3.1 through 5.3.7 describe aspects of the Unit Route Planner in greater detail.

5.3.1 Marking Obstacle Segment Abstractions in Grid Cells

This section describes the details of analyzing the SIMNET Terrain Database to create Obstacle Segment Abstractions.

Physical obstacles (treelines, canopies, rivers, and lakes) that cross a cell’s boundaries are marked as Obstacle Segments for that cell. A cell may lie in one, two, or four terrain database patches. All the physical obstacles in the patches that contain the cell have to be searched to determine Obstacle Segments for the cell.

5.3.1.1 Treeline obstacles

A treeline is described by a list of vertices which determine its shape. A treeline is composed of a number of treeline *segments* which is the part of the treeline between two treeline vertices. A *bounding box* of a treeline is constructed which defines the smallest rectangle that encloses the treeline. The bounding box is tested to see if it overlaps a cell.

If the bounding box intersects the cell at two points on *different* cell edges, the treeline is a candidate for classification as an Obstacle Segment. If there is no overlap between the bounding box and the cell or if the bounding box is completely enclosed by the cell, the treeline is not considered as an Obstacle Segment. Even though the bounding box of the treeline may intersect the cell boundary at two points on different cell edges, this by no means guarantees that the

treeline crosses the cell boundary and is an Obstacle Segment. Such a situation is shown in figure 5.3.1.1-A.

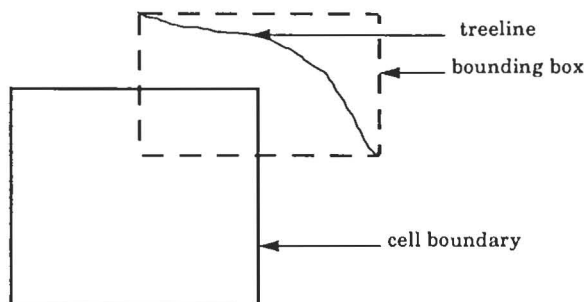


Figure 5.3.1.1-A Bounding box overlap does not guarantee an Obstacle Segment

Additional tests must be done on the treeline to make sure that it does cross the cell boundary in order to classify it as an Obstacle Segment. This is done by constructing a bounding box for each treeline segment and testing each bounding box against the cell for possible overlap. A situation similar to the one described for the treeline bounding box/cell overlap case may develop; a treeline segment bounding box may cross the cell boundary but the treeline segment may not (see Figure 5.3.1.1-A). In such a case, the treeline segment is tested against the cell boundary directly by doing a *line intersection* test. The bounding box tests at the two levels (treeline and treeline segment) and possible treeline segment intersection tests guarantee that treeline obstacles are correctly recognized.

5.3.1.2 Canopy obstacles

In the SIMNET TDB, a canopy is defined as a collection of treelines and base polygons. Canopies can be classified as Obstacle Segment Abstractions by first constructing a bounding box for the entire canopy and testing its overlap with the cell. If the canopy bounding box intersects the cell boundary at two points on different cell edges, the canopy is a candidate for classification as an Obstacle Segment Abstraction. However, this does not guarantee that the canopy *will be* an Obstacle Segment Abstraction.

If the canopy bounding box overlaps the cell, each base polygon in the canopy is tested for intersection with the cell by constructing its bounding box and testing its overlap with the cell. Again, this may not guarantee that the polygon overlaps the cell, in which case intersection tests between the polygon edges and the cell may have to be done to be sure. It may be possible that some canopies may give rise to more than one Obstacle Segments in a cell (for example, a vertical and south west diagonal obstacle may result). In such a case all the Obstacle Segments that are part of the same physical obstacle are given the same obstacle identification number (OIN).

5.3.1.3 River obstacles

A river flowing through a terrain database patch is described by a collection of (possibly) overlapping "water" polygons. The algorithm first determines the *connectivity* of water polygons in a patch. Finding the connectivity of water polygons refers to partitioning them into groups such that all water polygons in one group are connected to each other. This is necessary because two or more sets of connected water polygons may give rise to two or more rivers and each river would have to be marked as a separate Obstacle Segment Abstraction.

Each group of connected polygons is tested for possible classification as an Obstacle Segment Abstraction by constructing bounding boxes for all the water polygons in that group and by checking their overlap with the cell. Intersection tests may have to be done to determine that a polygon whose bounding box overlaps the cell does indeed intersect the cell boundary.

The bounding box/cell overlap tests continue until another water polygon belonging to the same river is found that intersects a cell edge which is different from the one that was intersected by the first water polygon. This indicates that the river cuts through two different edges of the cell making certain regions of the cell unreachable.

5.3.1.4 Lake obstacles

Just like rivers, lakes are described in a similar fashion; the lakes in a patch are composed of a number of water polygons. The same logic is used for recognizing lakes as Obstacle Segment Abstractions as was used for recognizing rivers as obstacles (see section 5.3.1.3).

5.3.2 Sample points

In the typical regular grid approach to route planning, the center of the grid cell is the single available route point. The Unit Route Planner has instead a set of 12 available route points called sample points. The sample points are the vertices used by A* in searching the OSA grid for routes. Figure 5.3.2-A shows the sample points in relation to the diagonal OSs, a horizontal OS, and a vertical tunnel OS. They are arranged so there is at least one sample point on each side of each OS.

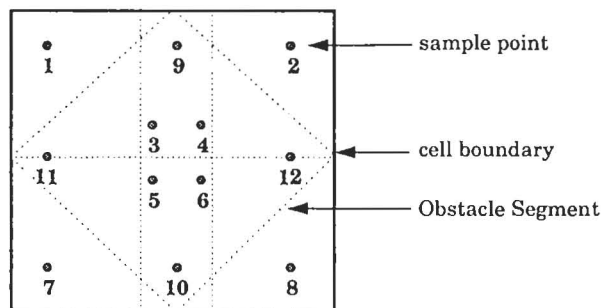


Figure 5.3.2-A Sample Points

All grid cells have at least the first eight sample points. If the grid cell contains a tunnel (see Section 4.2), four additional points (9..12) are added.

Regular grids typically show a “digitization bias”, Mitchell [1988], in which only 4 (8 if diagonal moves are allowed) angles out of each cell are available; these angles correspond to moves to the orthogonal and diagonal cells. This digitization bias causes the “jagged” appearance of routes. Sample points greatly reduce digitization bias. Each cell has between 512 angles (8 sample points * 8 neighbors * 8 sample points per neighbor) and 1152 angles (12 sample points * 8 neighbors * 12 sample points per neighbor). Each partial route can be extended to between 64 and 96 available sample points. Section 5.3.6 discusses how this “explosion” of partial routes is controlled.

5.3.3 Adjusting start and destination points

Because of the correlation error between an obstacle and its Obstacle Segments, the start and destination points can not be translated to the OSA grid directly from their locations on the terrain. Simply put, the start or destination points must remain on the correct “side” of the Obstacle Segments. Figure 5.3.3-A illustrates the problem. The start point in the OSA grid must be to the southwest of the river’s OS.

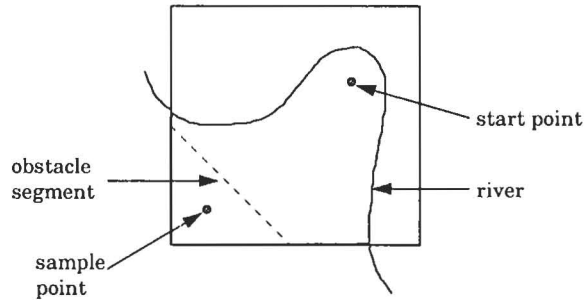


Figure 5.3.3-A Start point adjustment

An algorithm was devised to move the start and destination points so that their positions relative to the abstract obstacles matched their positions relative to the physical obstacles. Specifically, the start or destination point is moved to a sample point (see Section 5.3.2) inside the grid cell. The algorithm is based on the *floodfilling* approach as described in Foley et.al. [1991].

The algorithm starts by subdividing the grid cell containing the terminal point (start or destination) into 10 meter X 10 meter subcells. It then fills the cell with physical obstacles by marking appropriate sub-cells as being “blocked” by physical obstacles.

The algorithm then floodfills the cell starting from the terminal point and determines which edges and corners of the grid cell have been “colored”.

Finally, the sample point replacing the terminal point is determined:

1. First determine which corners are colored. Put each colored corner’s sample point (i.e., the sample point nearest to that corner) onto the list of accessible sample points.
2. If the list contains 0 or more than 1 sample points (zero or more than one corner was colored), examine the edges:
 - 2.1 If the left and top edges are colored, then add sample point 3 (see Figure 5.3.2-A) to the accessible sample points list.
 - 2.2 If the left and bottom edges are colored, then add sample point 5 to the accessible sample points list.
 - 2.3 If the top and right edges are colored, then add sample point 4 to the accessible sample points list.
 - 2.4 If the bottom and right edges are colored, then add sample point 6 to the accessible sample points list.

3. Of all the sample points in the accessible sample points list, find the closest sample point to the actual point and move the terminal point there.

For the case shown in figure 5.3.3-A, the region that is floodfilled and the new starting point is as shown in figure 5.3.3-B.

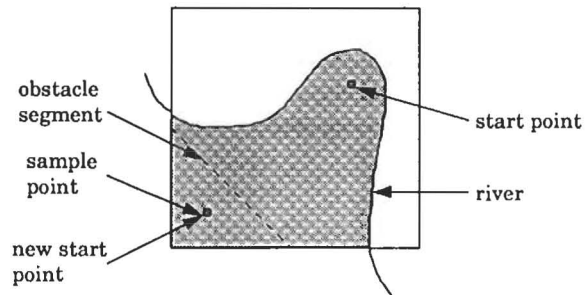


Figure 5.3.3-B Floodfilling the grid cell

5.3.4 Movement between grid cells

Obstacle Segment Abstractions are approximations of the underlying terrain obstacles. OSAs in orthogonal cells may “touch” on the edge between the cells even though the obstacles do not touch. This creates an artificial barrier to routing. To solve this problem the concepts of the “shared point” and “Obstacle Segment Displacement” were introduced.

5.3.4.1 Shared points

Obstacle Segments in adjacent orthogonal grid cells may touch on the boundary between the two grid cells. The point of contact is called a *shared point*. Shared points are the midpoints of the edge separating two cells. For example, in Figure 5.3.4.1-A, *s* is a shared point.

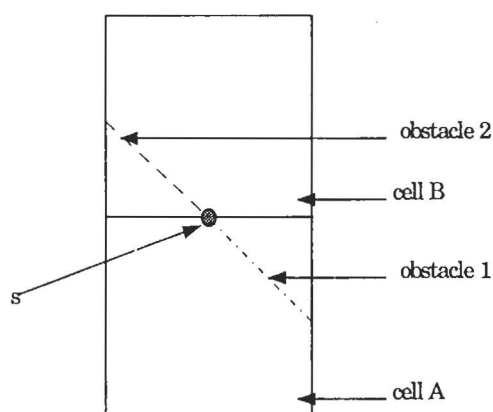


Figure 5.3.4.1-A A shared point.

A shared point exists between grid cells only when

- the destination cell is an orthogonal cell, and
- the two grid cells contain Obstacle Segments that touch at the shared point.

Shared points are used in allowing movement between obstacles whose OSAs meet (see Section 5.3.4.2).

5.3.4.2 Obstacle Segment Displacement

Obstacle Segment Abstractions may be aligned so as to block movement that would otherwise be possible if routes were being generated with respect to the underlying terrain features only. Consider, for example, the movement from cell A to cell B in Figure 5.3.4-A.

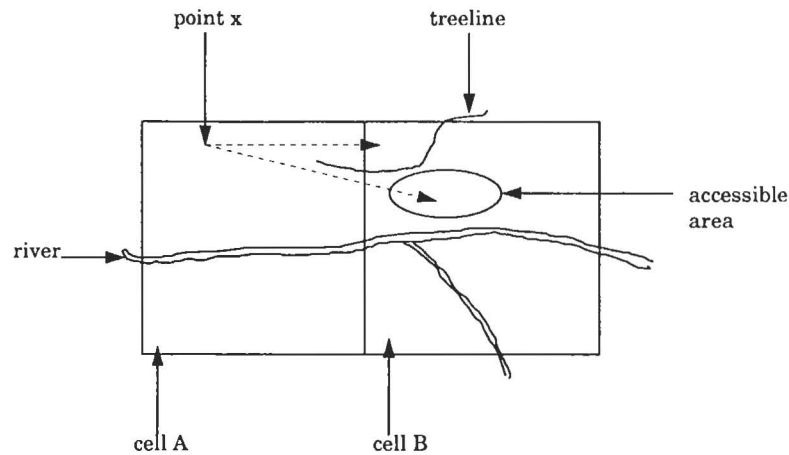


Figure 5.3.4-A Access to region between physical obstacles in two cells

From the figure, it is clear that a route may be extended from point *x* to the accessible area in cell B.

The OSAs for these two grid cells restrict access to the accessible area (Figure 5.3.4-B). Without adjustment, the only move from *x* is to the NW of the treeline.

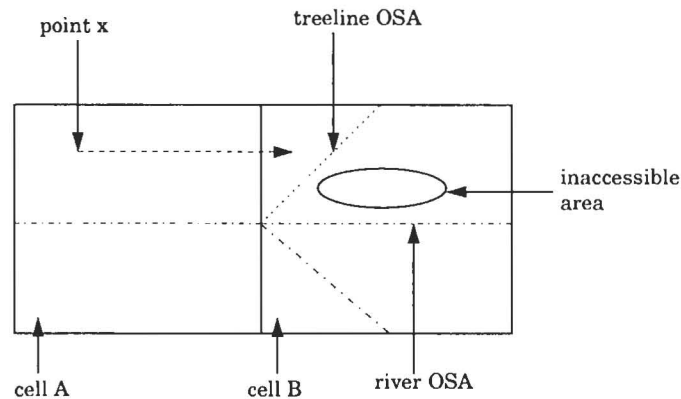


Figure 5.3.4-B The corresponding obstacle abstractions

To prevent “touching” OSAs from unrealistically eliminating route segments, OSAs are “displaced” or moved away from shared points. This is reasonable; the terrain features do not touch (they would be represented as the same OSA if they did), so the OSAs should not touch. In Figure 5.3.4-C, the OSA representing the river is displaced to the South and the treeline OSA is displaced to the North.

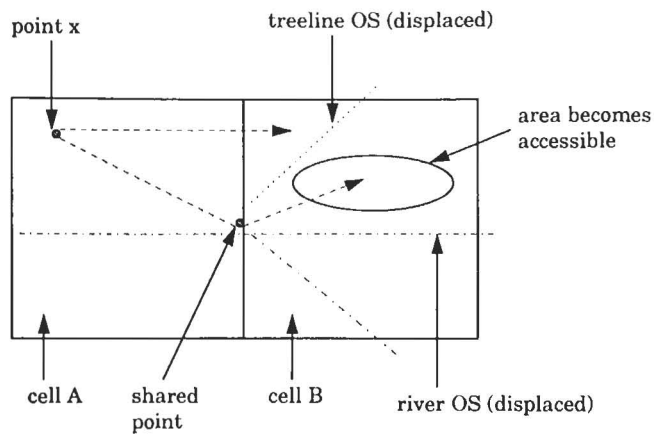


Figure 5.3.4-C Displaced obstacle segments

The displacement opens the gap between different OSAs making movement between them possible.

5.3.5 Extended Obstacle Segments

A move into a diagonal grid cell may sometimes be blocked by an OS in an orthogonal cell. Such a situation is shown in Figure 5.3.5-A.

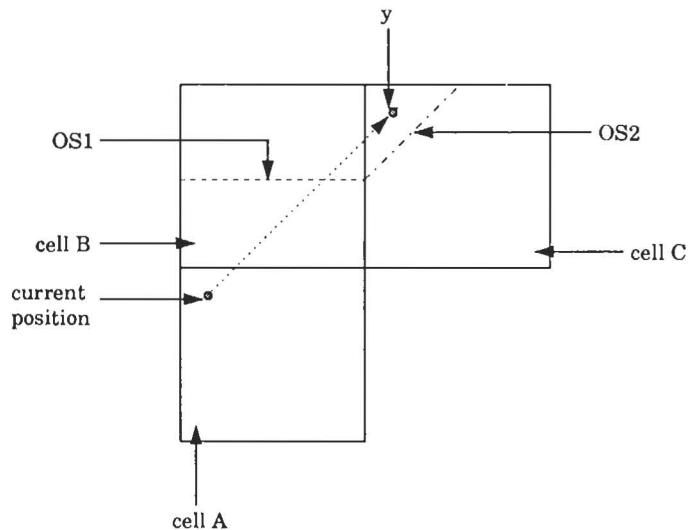


Figure 5.3.5-A Diagonal move blocked by abstract obstacle in an orthogonal cell

The *PlanRoute* algorithm (Section 5.1) considers only two grid cells at a time when extending partial routes. In Figure 5.3.5-A, a move is being considered from “current position” in cell A to *y* in cell C. The move appears to be valid when only cells A and C are considered; knowledge of OS1 in cell B is not used. However, such moves should be disallowed because there may be OSs in cell B that block such a move (e.g. OS 1 in Figure 5.3.5-A.)

To disallow such moves, OSs in destination cells are “extended” as shown for the case given in Figure 5.3.5-A.

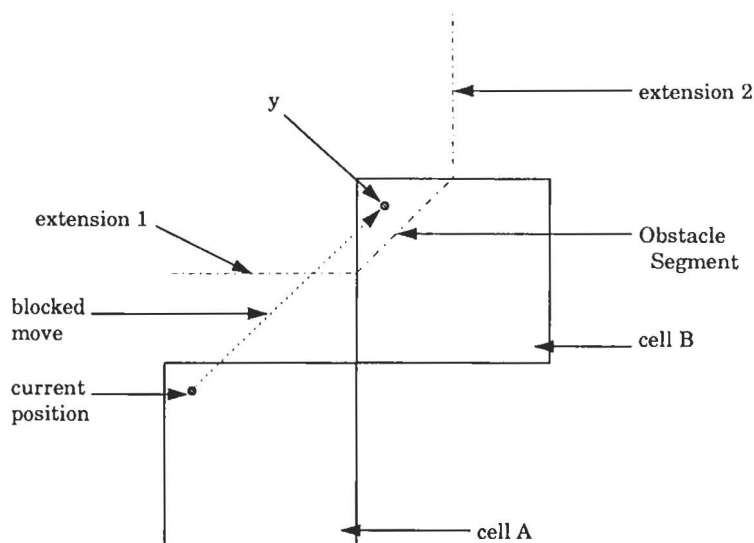


Figure 5.3.5-B Extended Obstacle Segment

An “Extended OS” consists of three components: the Obstacle Segment and its two extensions (one from each endpoint). Each extension is a line segment at a right angle to its edge of the grid cell. When determining if a sample point can be reached, the extensions as well as the OS are checked.

5.3.6 Determining Reachable Points

Routes are extended to “reachable” points in neighboring grid cells. A sample point is reachable if the route segment to the point does not intersect an OS. A route segment is a line segment from the end of the route to a sample point.

If two adjacent grid cells do not contain the shared point (Section 5.3.4.1), line segments are drawn to each of the sample points in the destination grid cell. If a line segment intersects an OS, the sample point is unreachable and discarded.

If two adjacent grid cells contain a shared point, the shared point is checked for reachability. If so, the sample points are checked for reachability from the shared point. Thus, a sample point is reachable if an unobstructed route segment exists to a reachable shared point.

Reachable points are sorted by their `grid_cost` (see Section 5.3.7).

The reachable points are partitioned into *mutually reachable sets* (MRS) which have the property that all points in a MRS are mutually reachable. Reachable points partitioning controls the combinatorial explosion of partial routes introduced by the multiple sample points. If any point in a MRS is involved in a route, all sample points in the MRS are considered to be part of the route and are not considered when extending additional partial routes. Thus, a MRS defines a reachable area and the least costly sample point within a MRS is used for routes into that area.

For example:

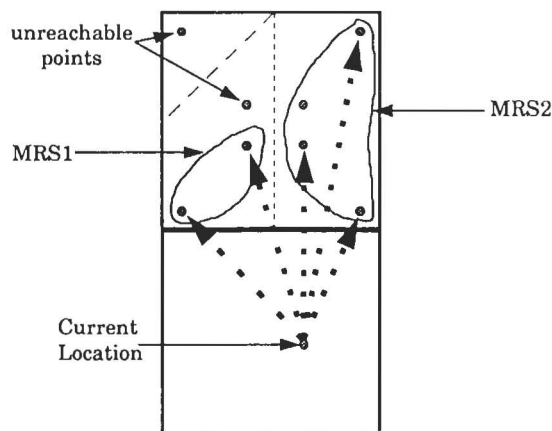


Figure 5.3.6-A Mutually Reachable Sets

In Figure 5.3.6-A, sample points 1 and 3 in the North grid cell are unreachable from “Current Location”. The reachable sample points have been partitioned into two MRSs corresponding to two possible destination areas in the North cell from “current location”. Two partial routes will be created from Current Location into the North cell. The partial route into MRS1 will be to the least costly of MRS1’s two sample points. The partial route into MRS2 will be to the least costly of MRS2’s four sample points. Hence, two, rather than six, partial routes are created for further analysis by A*.

5.3.7 Computation of route cost

The A* algorithm evaluates partial routes on their “cost”. The cost of a route, partial or whole, is the sum of the costs of its route segments. Three factors are considered in determining the segment cost: distance, trafficability, and concealment. Segment cost is the cost of moving between sample points in adjacent grid cells and is made up of three components:

- The *base cost* of the terrain in the destination grid cell.
- The *slope* of the terrain around the destination sample point.
- The *percentage of intervisibility* from an area around the destination sample point to the areas around the enemy locations.

The base cost is an average determined from the types of terrain in the grid cell. The average is calculated from 25 points taken from a regular 5 x 5 grid within the grid cell. For example, the base cost for a grid cell mostly on sand is greater than for a grid cell mostly on dirt. There are other approaches to calculating the base cost. A simple approach would use only the terrain under the sample point. This approach ignores the terrain along the route. Another approach is to sample the terrain along the route segment. This approach considers each route to have zero width. Zero width routes have a major defect. Because unit routes have width (i.e. units travel along corridors), the terrain under the corridor is not introduced into the cost calculation. For example, narrow corridors are weighted the same as wide corridors. Although the approach taken in the Unit Route Planner may consider too much terrain, it has the advantage that the terrain under a corridor is considered. Recall that grid granularity is determined by the size of the unit; so, the expectation is that most of the area under a grid cell will be traversed by the vehicles in the unit. Calculating an average over a grid cell appropriately increases the cost of grid cells with slow-go or no-go terrain.

An additional factor in trafficability is the slope of the terrain under the route. Flat terrain has the lowest cost.

Concealment is considered by calculating a percentage of intervisibility from an area around the destination sample point to the areas around enemy positions known by the routing unit.

Parameters into the algorithm control how much weight is given to trafficability and concealment. Thus, concealment can be weighted more to obtain routes that maximize concealment over trafficability.

Route segment cost is calculated from the length of the segment, trafficability, and concealment:

$$\text{segment_cost} = (\text{trafficability_cost} + \text{concealment_cost}) * \frac{|\text{segment}|}{\text{grid_granularity}}$$

where:

segment_cost is the cost of the route segment
 trafficability_cost is a measure of trafficability of the terrain
 concealment_cost is a measure of concealment of the route
 |segment| is the length of the route segment and
 grid_granularity is the granularity of the grid.

Trafficability_cost is computed as:

$$\text{trafficability_cost} = \text{base_cost} + \text{slope} * \text{slope_weight}$$

where:

trafficability_cost is a measure of trafficability of the terrain
 base_cost is the base cost of the terrain in the destination grid cell
 slope is the slope of the terrain at the destination sample point expressed as a percentage (0.0 = flat terrain, 100.0 = maximum slope a vehicle can handle) and
 slope_weight is the weight given to slope component in the equation.

Concealment_cost is computed as:

$$\text{concealment_cost} = \text{los} * \text{los_weight}$$

where:

concealment_cost is a measure of concealment of the route segment
 los is the intervisibility from the area around the destination sample point to areas around all enemies expressed as a percentage and
 los_weight is the weight given to the los component in the equation.

The cost of the route is the sum of the segment_costs of all the segments that lie on the route:

$$\text{route_cost} = \sum_{i=1}^{n-1} \text{segment_cost}_{i,i+1}$$

where:

route_cost is the cost of the route from the starting location to point labeled n
 segment_cost_{i,i+1} is the cost of moving between points i and i+1 on the route
 n is the total number of points on the route

The base_costs for various soil types used in this research were:

base polygon type	base_cost
dirt soil	20
sandy soil	30
fordable water	50
asphalt road	1
dirt road	10

The weight factors used in this research were:

weight factor	weight
los_weight	10.0
slope_weight	1.0

5.3.8 Multiple "Optimal" Routes and Chokepoints

This research was done in conjunction with the development of a CGF Automated Mission Planning capability. The Mission Planner requires multiple, tactically sound unit routes between points and the identification of chokepoints along the route.

The Unit Route Planner generates multiple "optimal" routes between two points. As each route is generated the terrain covered by the route is made "less desirable" for subsequent routes by increasing the base cost of grid cells under the route. Hence, the base cost of "used" grid cells increases and subsequent routes tend to avoid repeating the previous routes. Each route is "optimal" considering that previously generated routes should be avoided.

Keeping track of previous routes allows chokepoints to be identified. When multiple routes between points are requested, route segments corresponding to narrow corridors and chokepoints must be reused. The Unit Route Planner identifies multiple used route segments as chokepoints.

6. Results

This section shows examples of routes that were generated for different grid granularities. The Unit Route Planner was incorporated into the IST CGF Testbed. Each figure is an image of a CGF Testbed Simulator's screen. There are five examples (Sections 6.1-6.5). Each section is composed of three figures, each a stage in the route generation process. The first figure shows the underlying terrain and the start and destination points marked on it. The second figure shows the grid overlaid on the terrain and the Obstacle Segment Abstractions. Finally, the third figure shows routes that were extended and the final route. All scenarios use the standard SIMNET Fort Knox terrain database, except Section 6.5, which uses the standard SIMNET Fort Hunter-Liggett terrain database.

6.1 *Grid granularity 5 meters*

This scenario shows route planning with a grid granularity of 5 meters. Although the Unit Route Planner was not designed for planning routes for individual vehicles, experiments show that vehicle routes can be generated with a grid granularity of 5 meters.

This 5 meter scenario is included to orient the reader to the Obstacle Segment Abstraction because it is clearly visible.

Figure 6.1-A shows the start and destination that are on opposite sides of a river. A road crosses the river to the north.

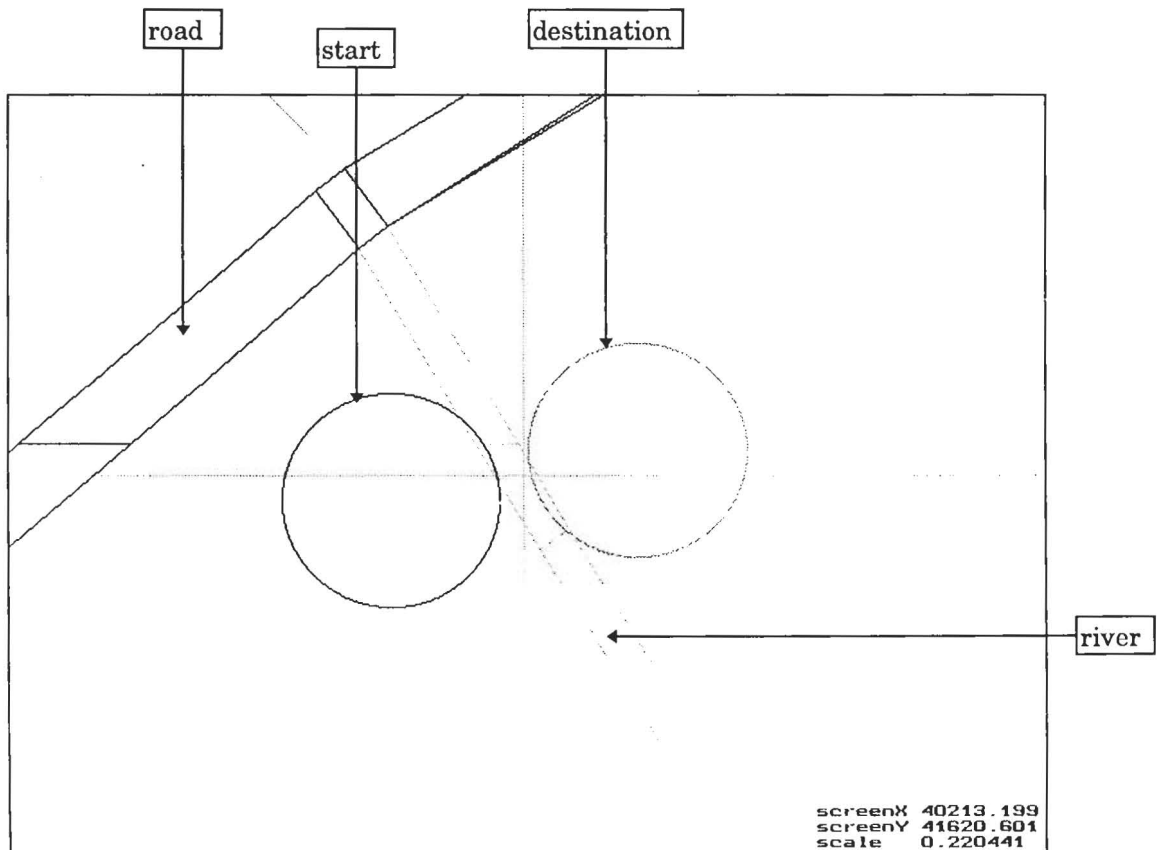


Figure 6.1-A 5 Meter Grid: Terrain, start, and destination points

Figure 6.1-B shows the grid and the OSAs. The OSAs closely follow the underlying terrain. At the 5 meter level, the OSA and its underlying terrain are within 2.5 meters.

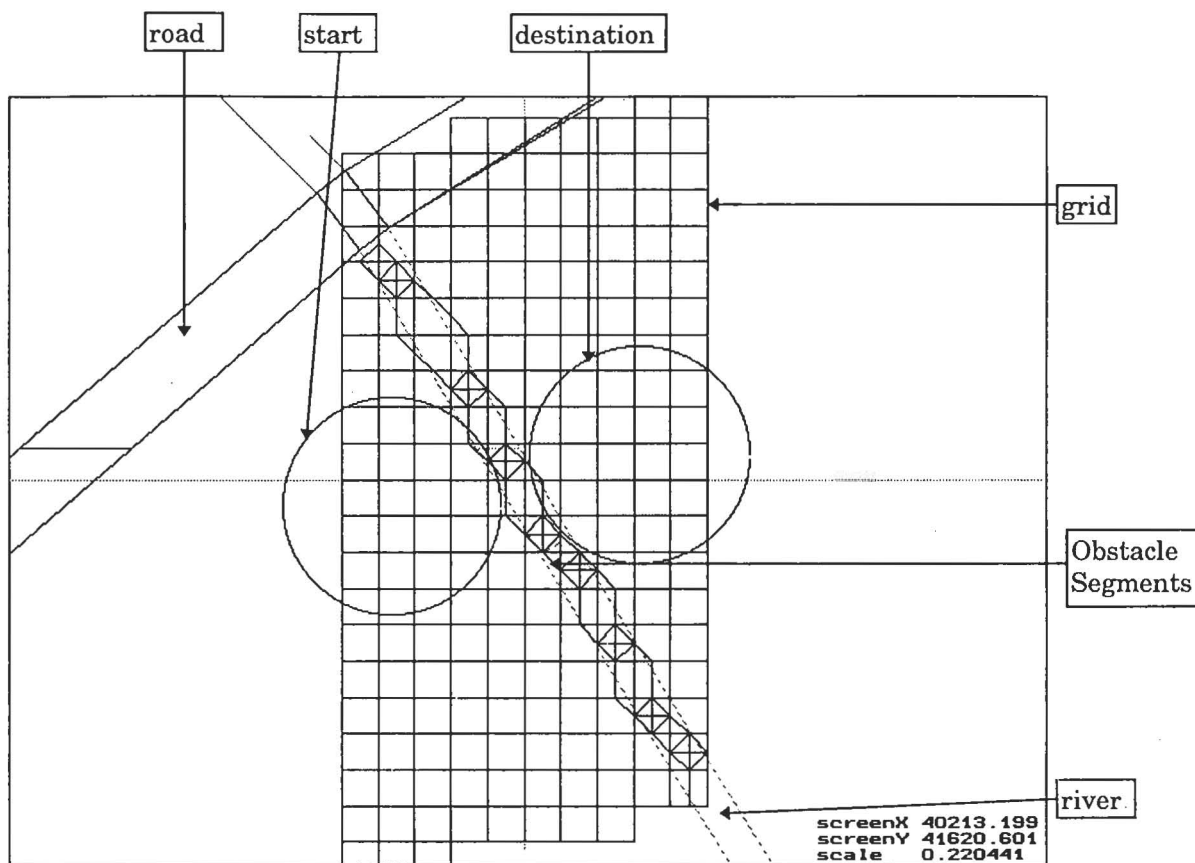


Figure 6.1-B 5 Meter Grid: Overlaid grid and OSAs

Figure 6.1-C shows partial and final routes generated by the A* algorithm. A*'s efficiency is seen after the bridge is found; very few partial routes were generated on the eastern (right) side of the river.

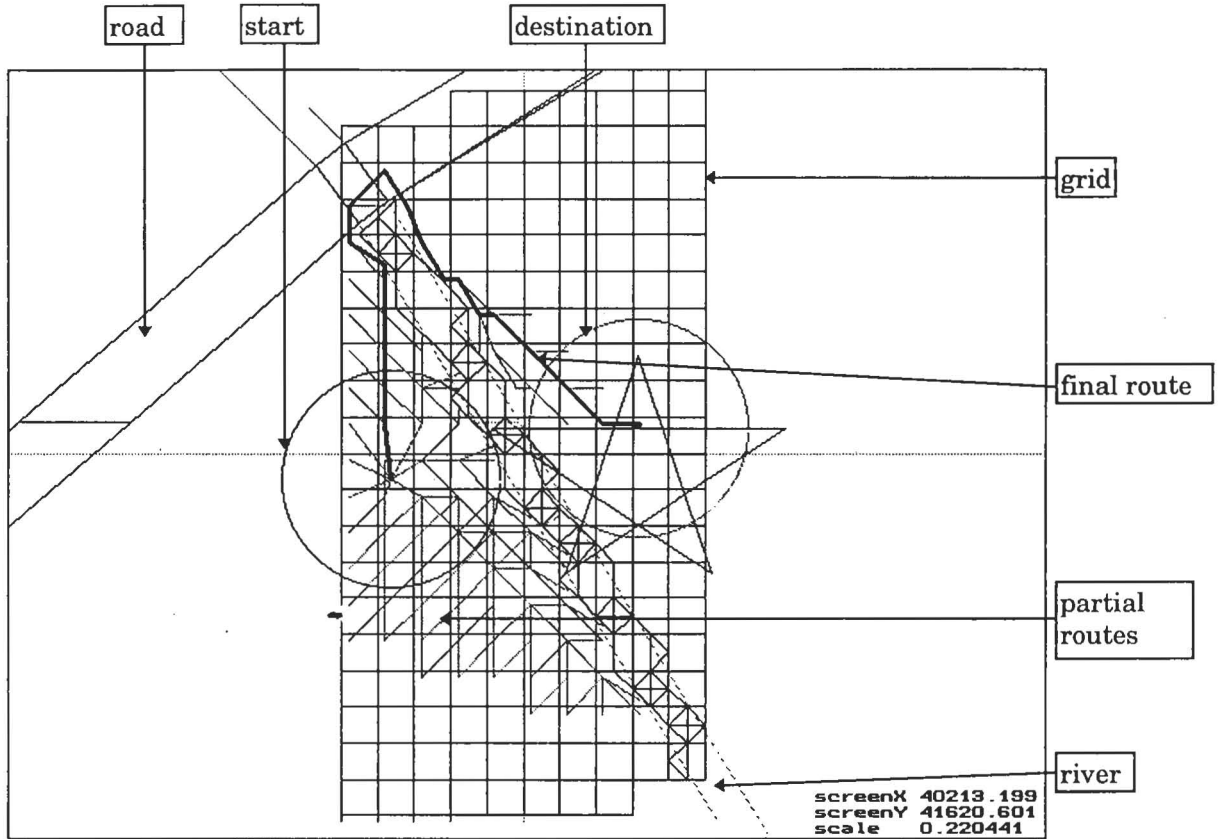


Figure 6.1-C 5 Meter Grid: Final route

Figure 6.1 shows the digitization bias of the Unit Route Planner is considerably less than the typical grid based route planner. Even so, some smoothing of the route would be preferred for a realistic appearing vehicle route. The IST CGF Testbed includes a grid based route smoothing algorithm that could be quickly incorporated into the Unit Route Planner.

6.2 *Grid granularity 85 meters*

This scenario shows route planning with a grid granularity of 85 meters suitable for platoon route planning. The start and destination points are approximately 0.75 km apart.

Figure 6.2-A shows the underlying terrain with the start and destination.

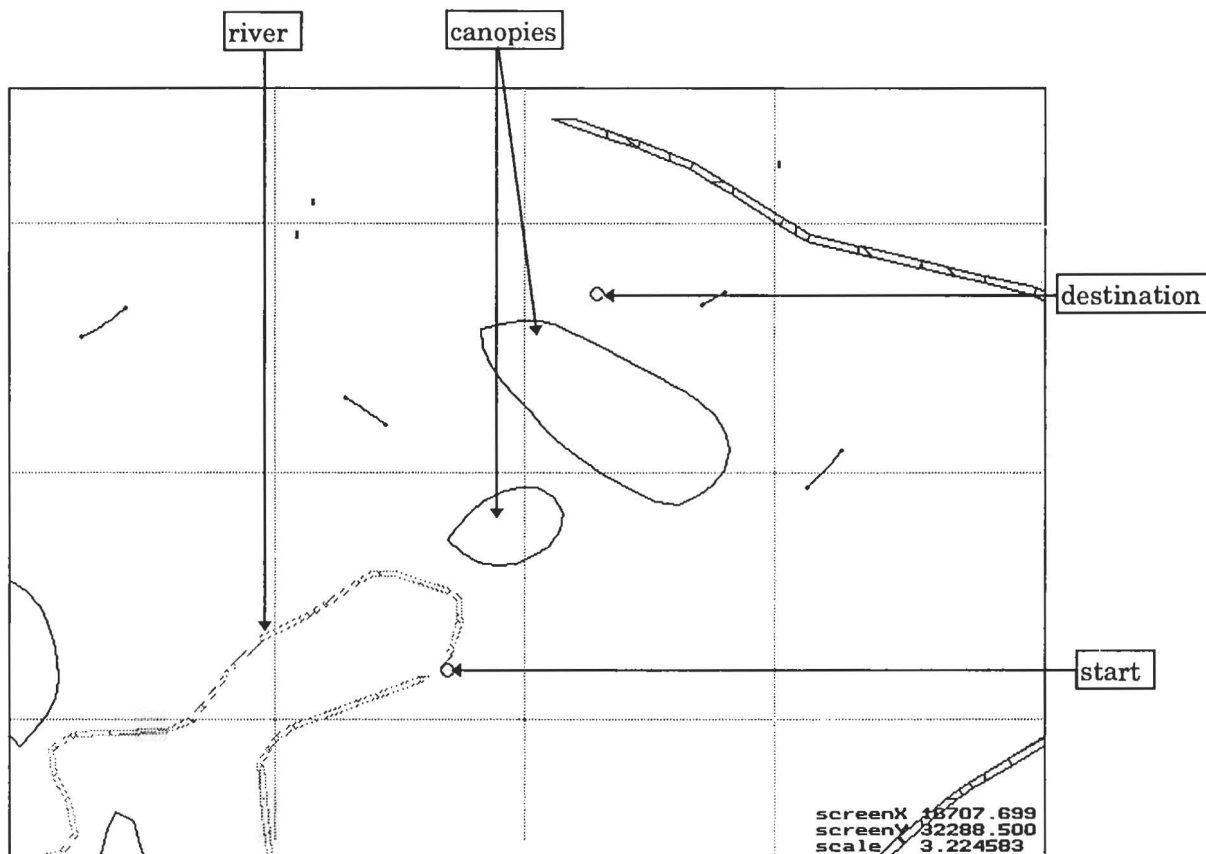


Figure 6.2-A 85 Meter Grid: Underlying terrain with start and destination

Figure 6.2-B shows the overlaid grid and the OSAs.

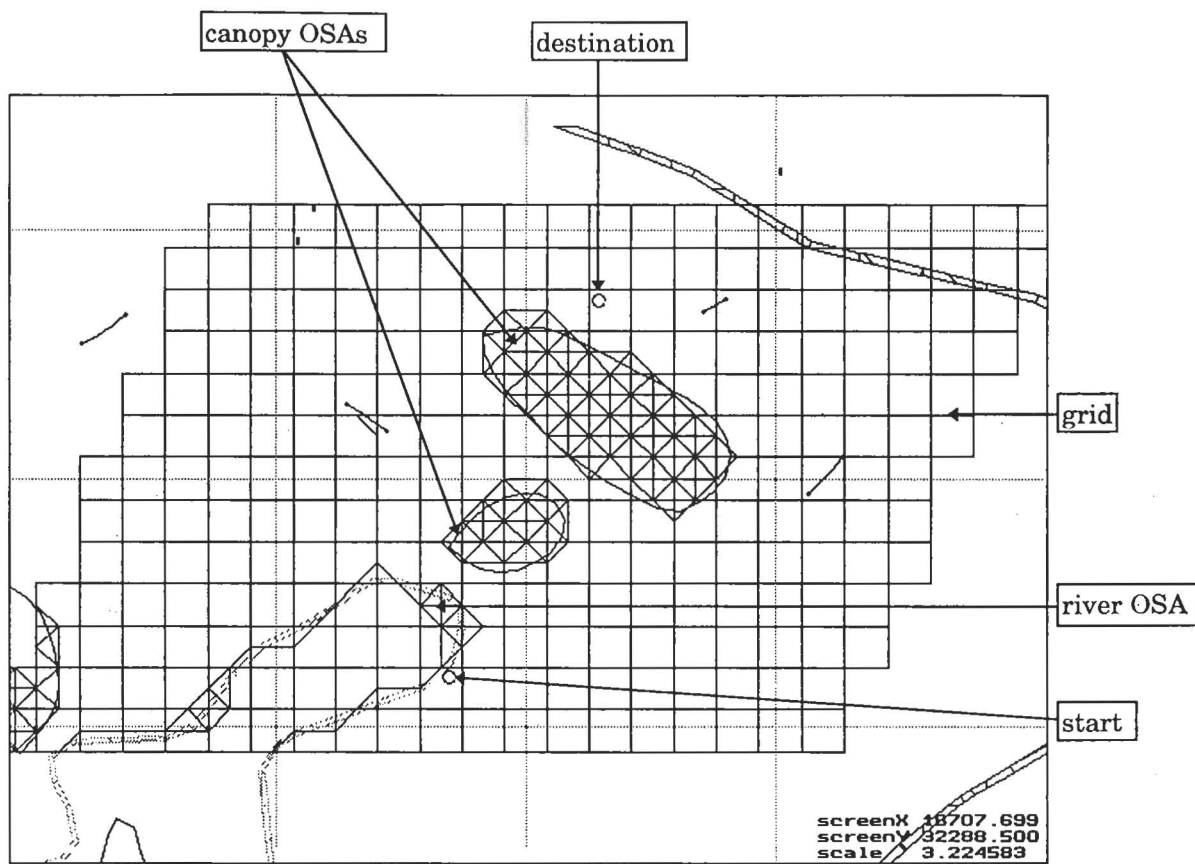


Figure 6.2-B Canopy and rivers OSAs, start, and destination

Figure 6.2-C shows the extended and final routes.

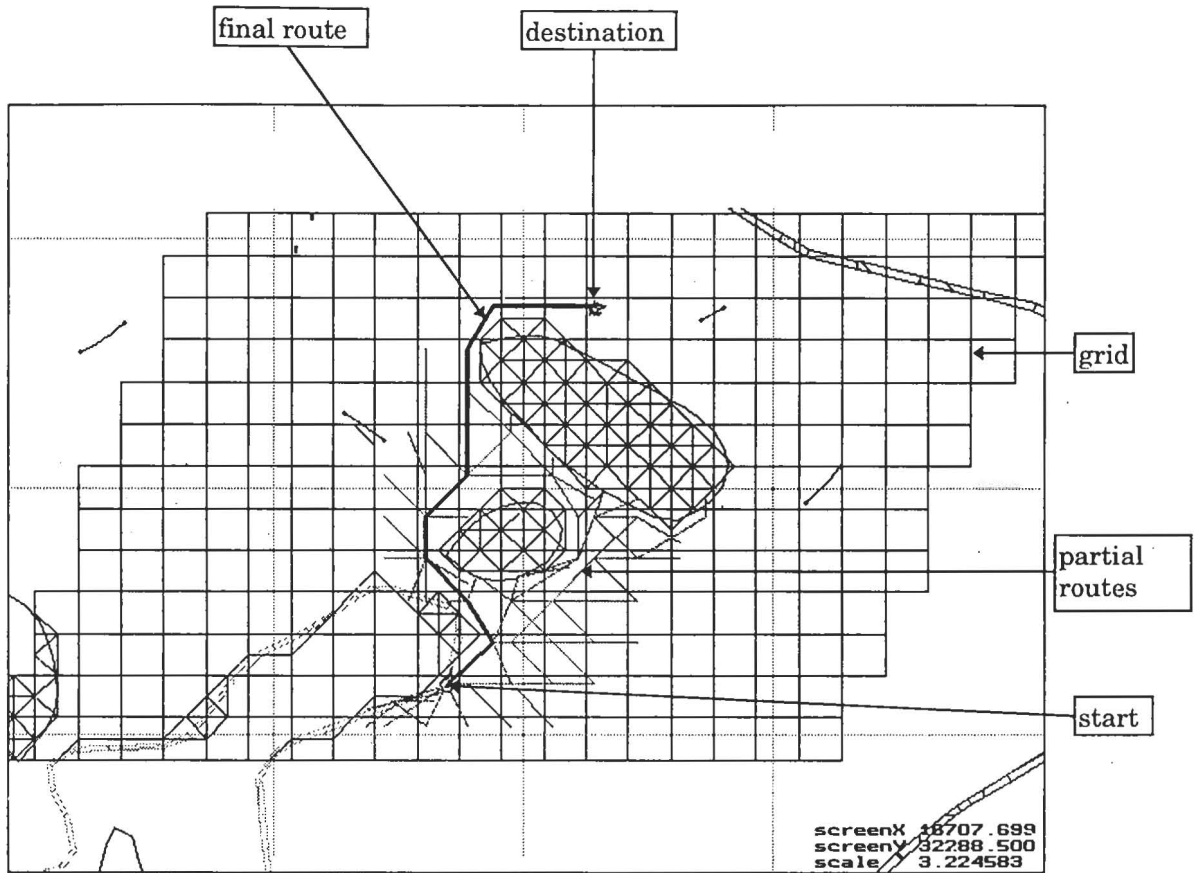


Figure 6.2-C 85 Meter Grid: The final route

Figure 6.2 shows the optimal route from the start to the destination. A platoon can use this route as the center line of a movement corridor. The vehicles within the platoon would be “free” to travel within some distance of this center line.

6.3 Grid granularity 85 meters with hostile units

This scenario is the same as Section 6.2's except for the presence of hostile units. The Unit Route Planning Algorithm searches for routes maximizing concealment from these hostile units.

Figure 6.3-A shows the underlying terrain, start and destination locations, and the hostile units' locations.

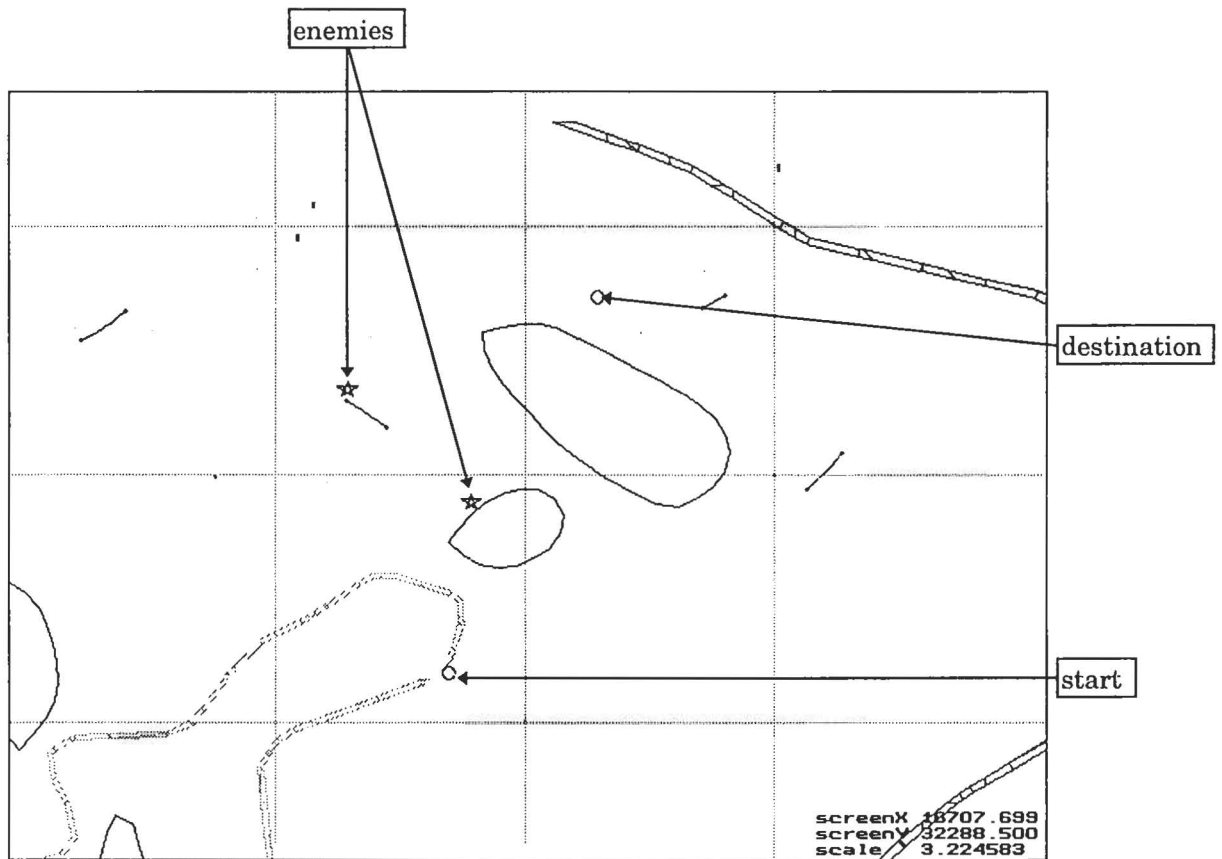


Figure 6.3-A 85 Meter Grid with Hostile Units: Terrain, start and destination, and hostile unit locations

Figure 6.3-B shows the overlaid grid and the OSAs in it.

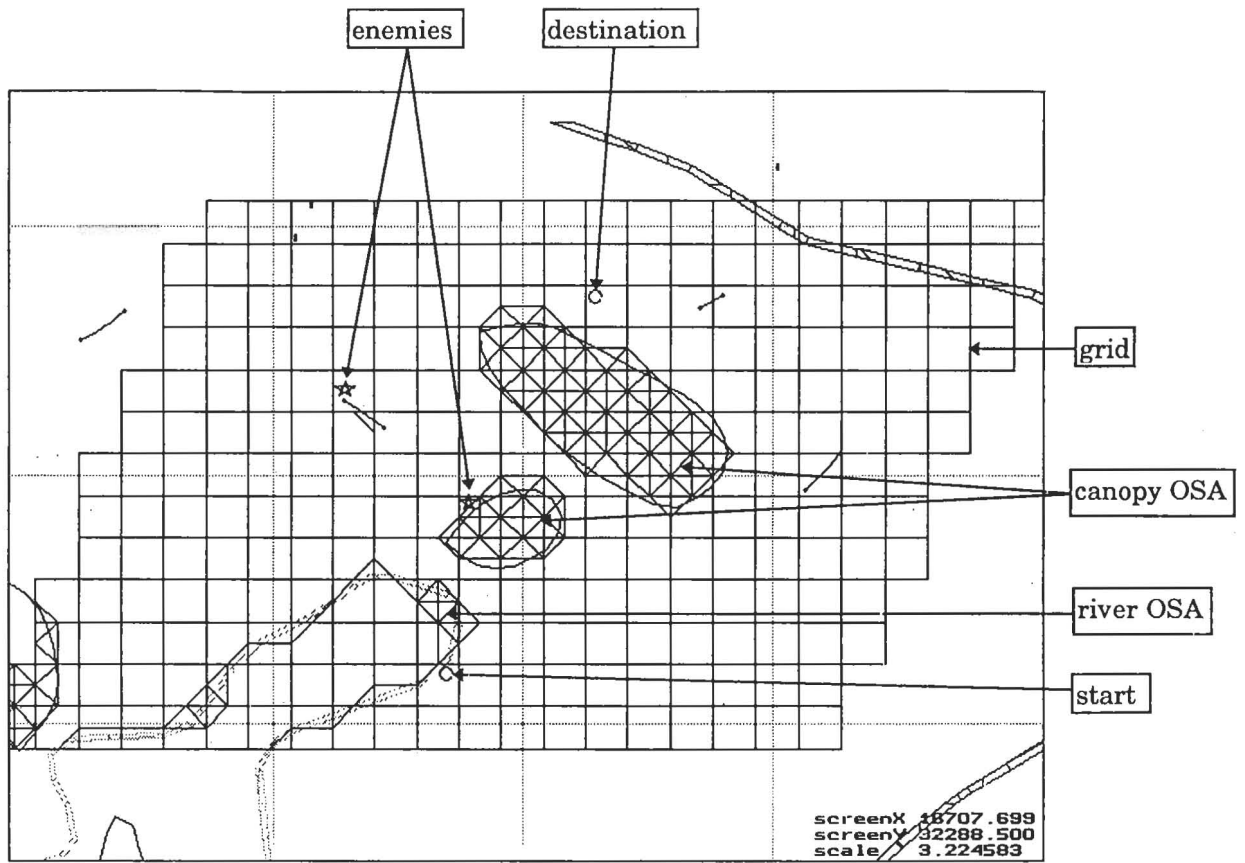


Figure 6.3-B 85 Meter Grid with Hostile Units: Overlaid grid and OSAs

Figure 6.3-C shows the partial and final routes.

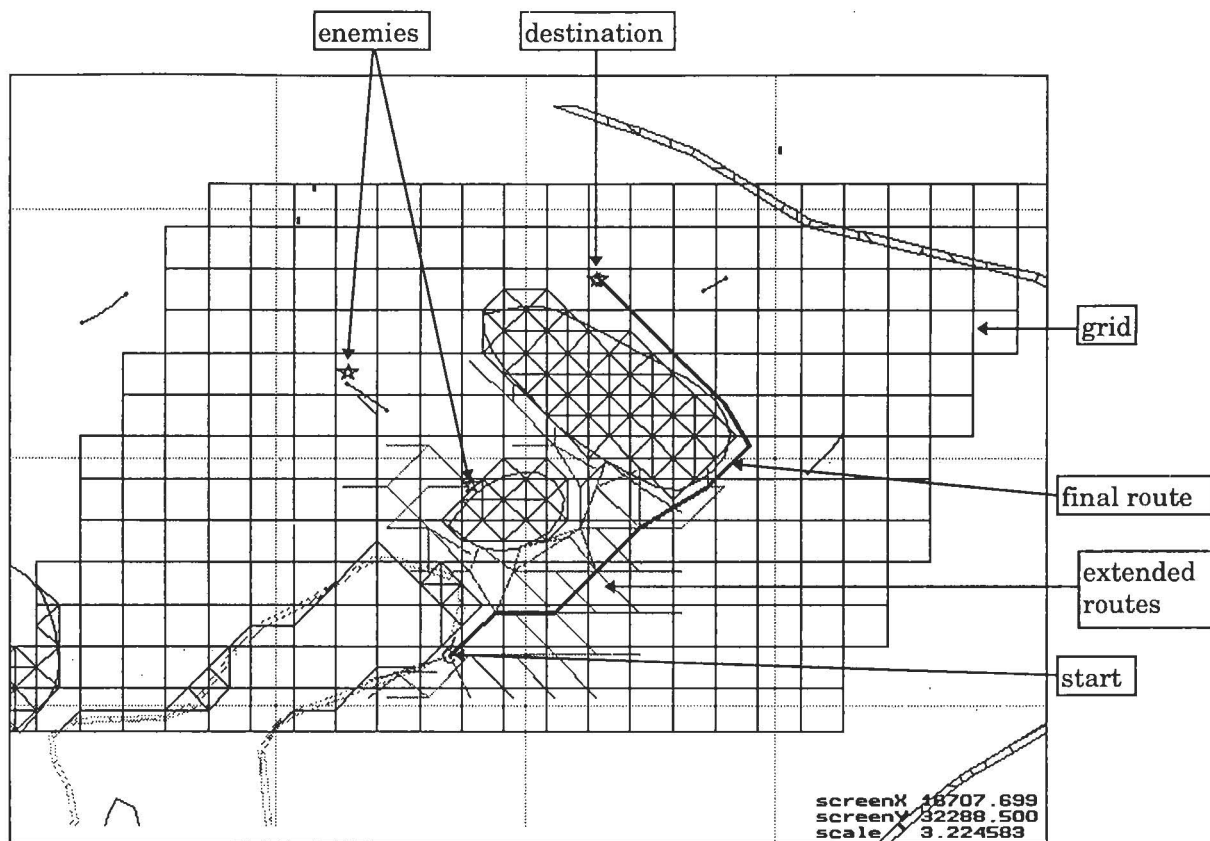


Figure 6.3-C 85 Meter Grid with Hostile Units: The final route

The final routes generated by this and the previous example (Figure 6.2-C) are considerably different. Figure 6.3-C shows the optimal route with these enemy positions. A route of maximal concealment has been generated. Comparison of Figures 6.2-C and 6.3-C show that the area near the two enemy positions was not "explored" by A* due to the lack of concealment. Instead a longer, more concealed route was found to the south of the canopies.

6.4 Grid granularity 500 meters

This scenario shows route planning with a grid granularity of 500 meters suitable for company route planning. The start and destination points are separated by 10 km and numerous rivers. This scenario challenges the Unit Route Planner to find routes across rivers.

Figure 6.4 shows the underlying terrain, start and destination.

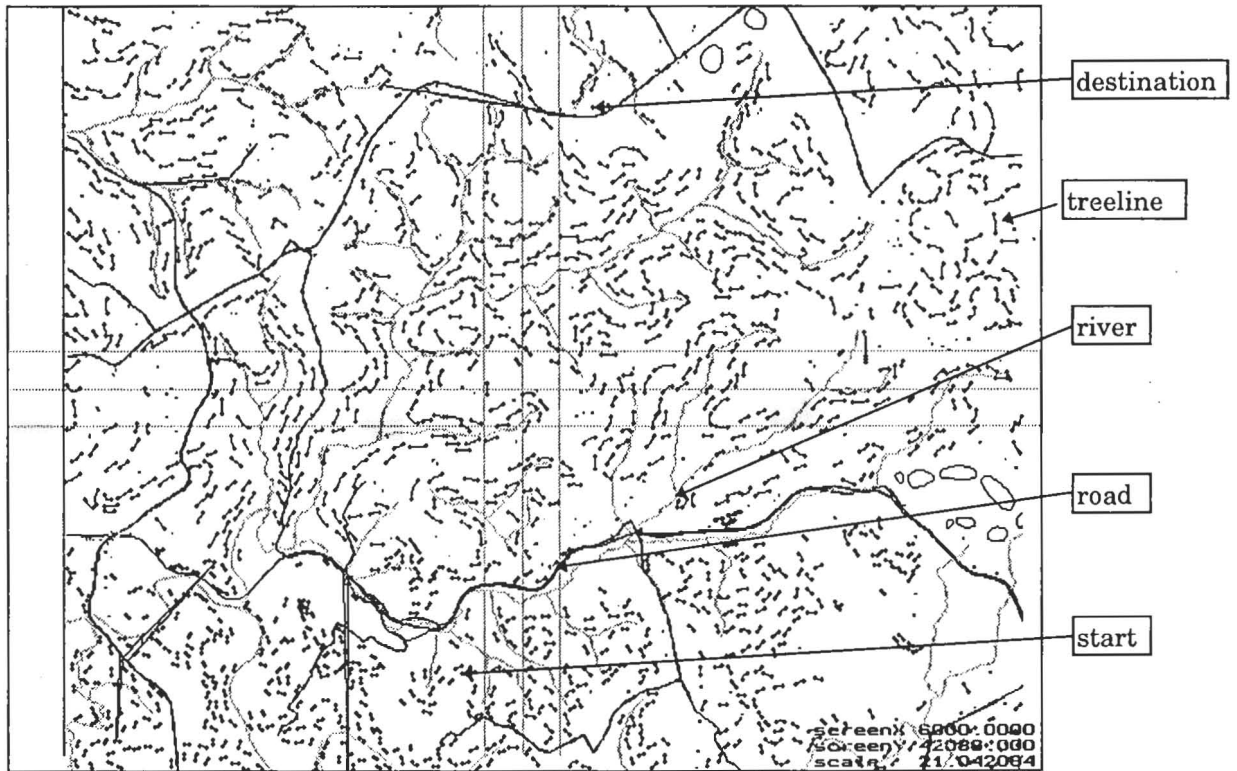


Figure 6.4-A 500 Meter Grid: Underlying terrain, start and destination

Figure 6.4-B shows the overlaid grid and the OSAs.

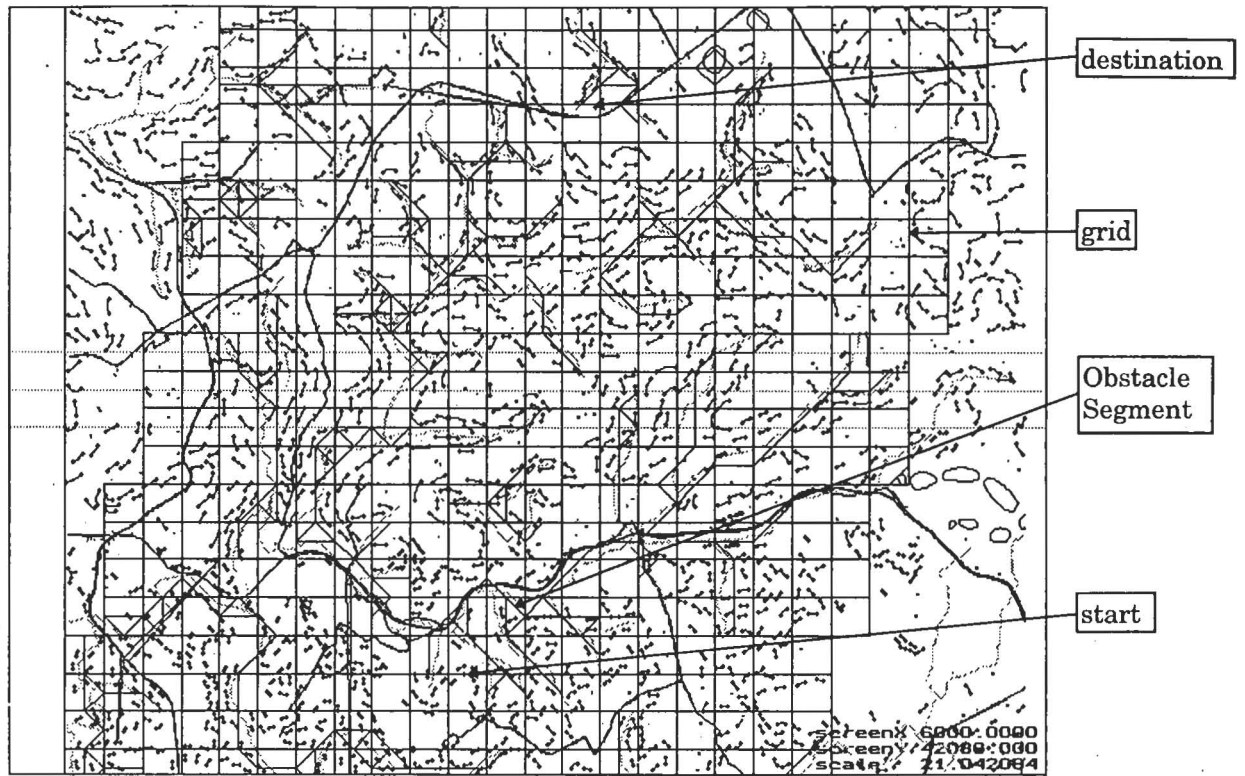


Figure 6.4-B 500 Meter Grid: Overlaid grid and OSAs.

Figure 6.4-C shows the partial and final routes.

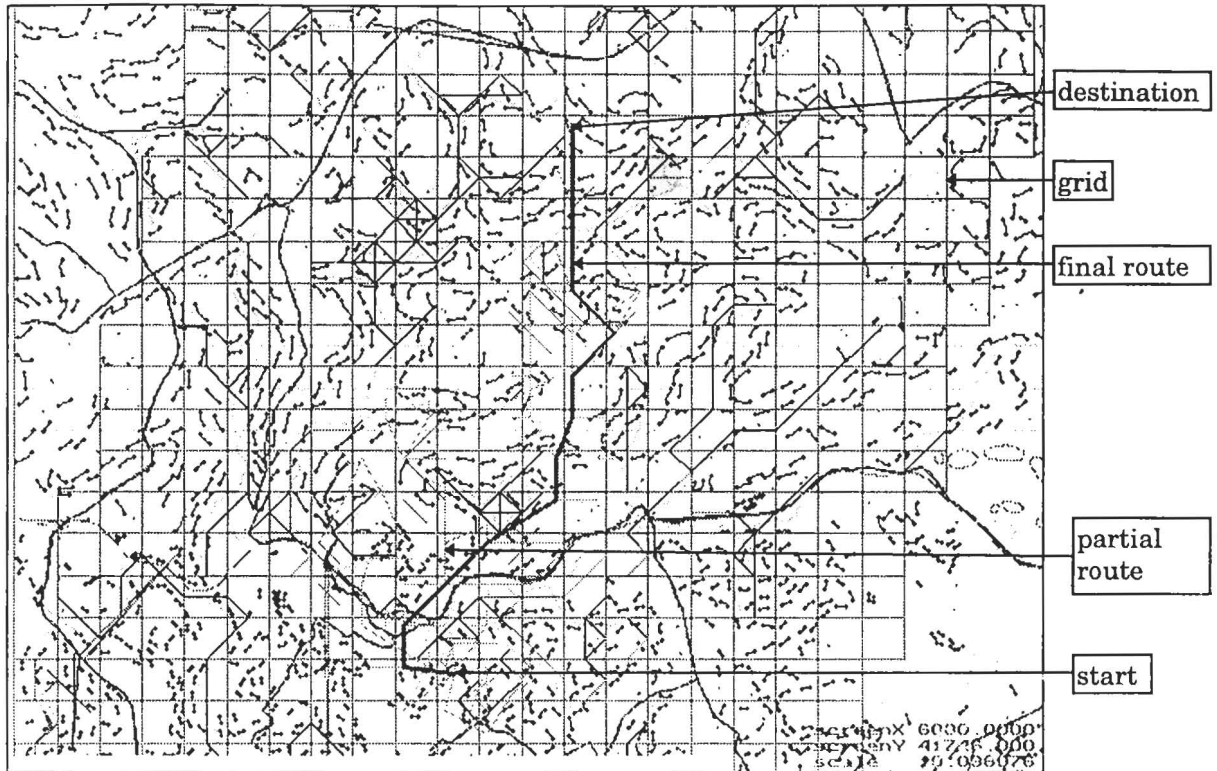


Figure 6.4 500 Meter Grid: Final route

An optimal 10 km route was determined in approximately 5 seconds on a 60 MHz 486 PC after the grid had been filled with Obstacle Segment Abstractions; a process taking approximately 20 seconds. Calculations show that the SIMNET Ft. Knox TDB can be completely represented at the 500 meter grid level as an OSA grid in approximately 200 Kbytes of memory. Such pre-processing of the TDB would improve the runtime performance of the Unit Route Planner by completing the Grid Creation and Filling steps prior to beginning the scenario.

6.5 Grid granularity 500 meters with tunnel

This scenario shows route planning with a grid granularity of 500 meters. This scenario challenges the Unit Route Planer to find narrow corridors between canopies. The "tunnel" Obstacle Segment (Section 6.5) is seen to represent a corridor between two canopies.

Figure 6.5-A shows the underlying terrain, start, and destination points.

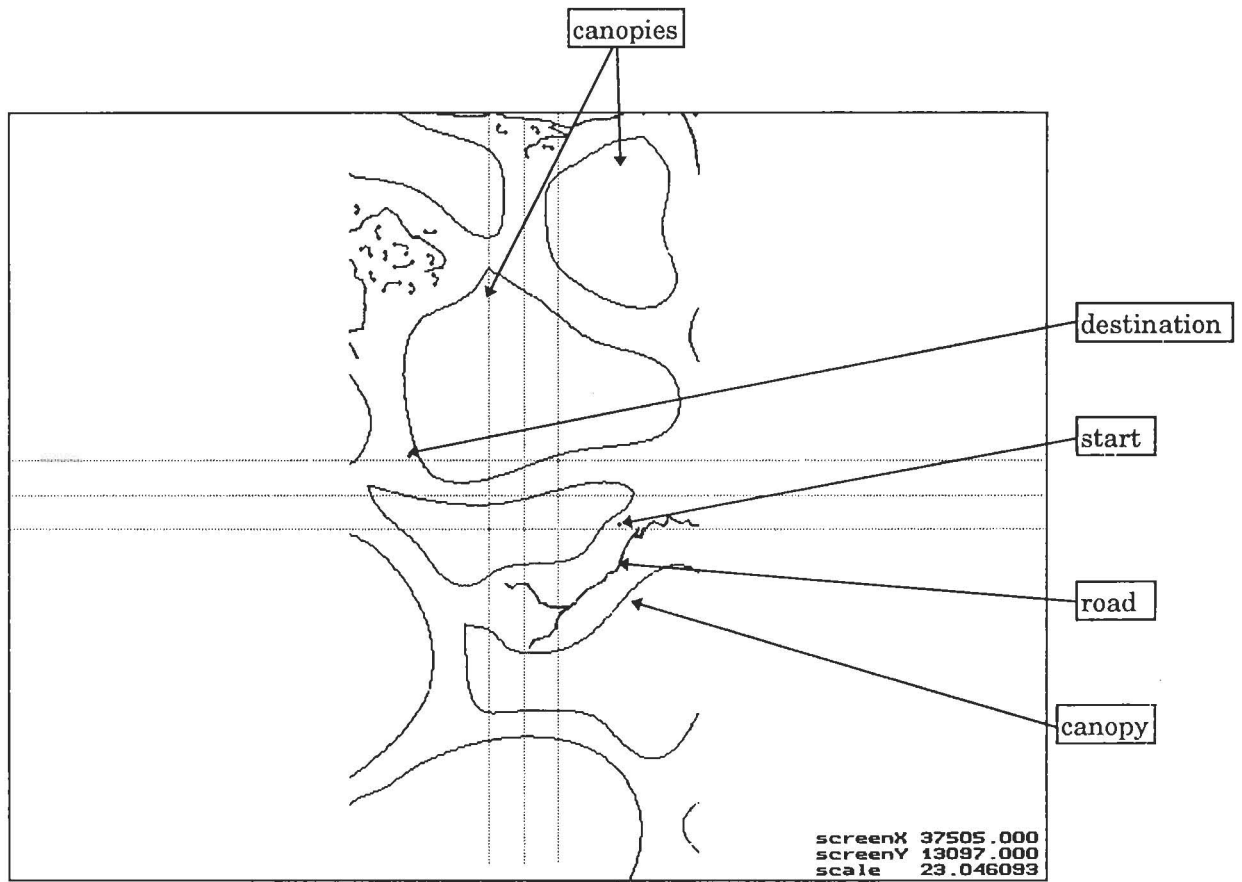


Figure 6.5-A 500 Meter Grid with Tunnels: Underlying terrain, start, and destination.

Figure 6.5-B shows the overlaid grid and the OSAs.

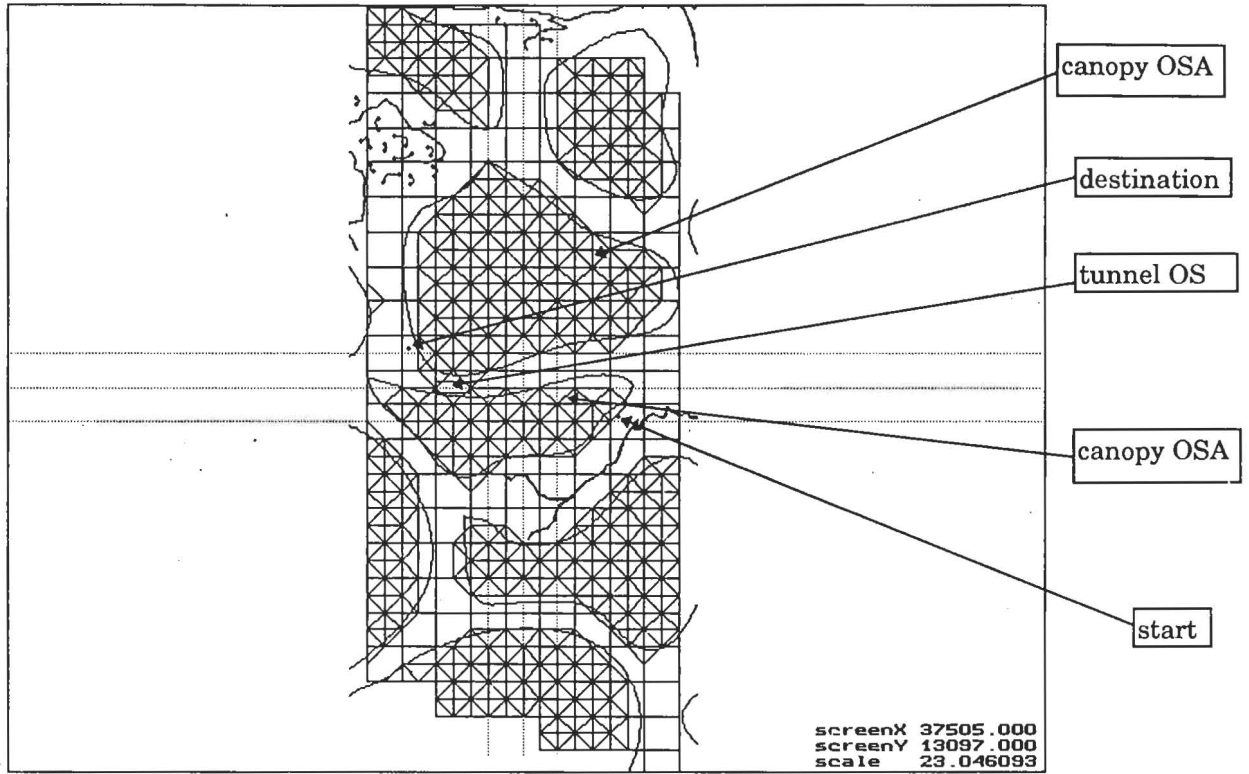


Figure 6.5-B 500 Meter Grid with Tunnels: Overlaid grid and OSAs

A tunnel OS has been created to represent a narrow gap between two canopies. The gap's width is less than the grid granularity.

Figure 6.5-C shows the partial and final routes.

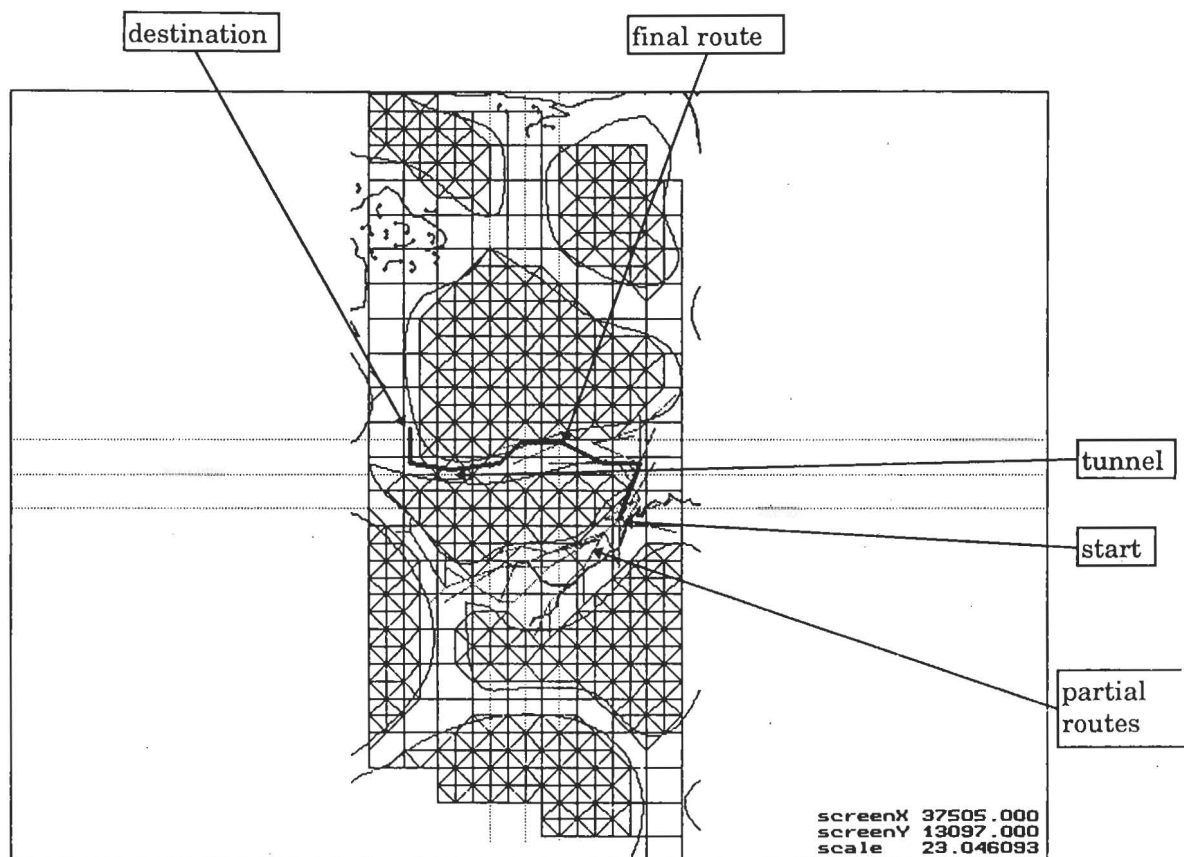


Figure 6.5-C 500 Meter Grid with Tunnels: Final Route

This scenario demonstrates the Unit Route Planner's ability to find and represent narrow sections of movement corridors. The "tunnel" Obstacle Segment is used to represent a corridor between obstacles that are closer than the grid size. In the typical grid based route planner, movement between obstacles closer than the grid size is blocked. The tunnel OS allows a coarser grid granularity and thereby improves computational efficiency.

7. Conclusions and Future Work

The Unit Route Planner is flexible and efficient. Its flexibility comes from its scalability. The granularity (i.e. size) of the underlying grid varies based on the size of the routing unit. The finer the granularity, the closer the Obstacle Segment Abstractions correspond to the physical obstacles. Fine granularity routing is suitable for finding precise vehicle routes while coarser granularity routing is suitable for finding unit routes. The smaller the unit, the finer the required granularity. The Unit Route Planner algorithm can be applied to any polygonal TDB.

The efficiency of the Unit Route Planner derives from three factors. First, the Obstacle Segment Abstraction approach allows obstacles to be represented with sufficient precision for routing but not so precise as to waste computational power. Second, the scaleable grid approach allows the representation of the terrain to correspond to the requirements dictated by the size of the unit. These two factors together prevent unnecessarily precise (i.e. computationally expensive) routes from being generated. Third, the Obstacle Segment Abstraction in combination with the vertex graph approach allows the application of an efficient search technique, A*, to the problem of determining routes.

The Unit Route Planner has additional strengths. First, three routing factors, distance, trafficability, and concealment, are considered in finding optimal routes. The relative contribution of each routing factor is controlled by parameters to the algorithm. Hence, optimal routes of different characteristics are determined. For example, concealment can be weighted more heavily than distance to produce predominately concealed but lengthy routes. Second, the Unit Route Planner generates multiple "optimal" routes between two points. As each route is generated the terrain covered by the route is considered less desirable. Subsequent routes tend to avoid the previous routes. Hence, each route is "optimal" considering that previously generated routes should be avoided. Third, chokepoints are identified. When multiple routes between points are requested, route segments corresponding to narrow corridors and chokepoints must be reused. The Unit Route Planner identifies those route segments as chokepoints. Fourth, narrow corridors between "close" obstacles are found. That is, corridors narrower than the granularity of the grid are represented. Fifth, the Unit Route Planner can be used to find and refine lengthy, precise routes through a process of successive refinement of routes. That is, a coarse route generated with a coarse grid granularity can be refined by applying the Route Planning process to sections of the coarse route at successively finer granularities. This approach would provide an efficient mechanism for planning detailed, lengthy routes. Sixth, although the goal of this research was a scalable, unit route planner, the Unit Route Planner is also an efficient vehicle route planner. It requires only the addition of a simple route smoothing algorithm to plan realistic vehicle routes.

In the current work, only terrain obstacles that are uncrossable (canopies, rivers, and treelines) are abstracted into OSAs. There are other features in the terrain which may prevent or hinder a unit's movement across them. These features are considered "no-go" and "slow-go" areas. For example, extremely steep terrain is a no-go area for many units. No-go areas could easily be represented as OSAs which would further decrease the combinatorial explosion of partial routes within the Unit Route Planner. Sandy and swampy terrain are slow-go areas because vehicles cannot move quickly over such terrain. An interesting area for future research would be to extend the OSA concept to representing slow-go obstacles.

8. References

- Craft, M.A., Cisneros, J. E., and Karr, C. R. (1994). "Dynamic Obstacle Avoidance", *Contract Report IST-CR-94-41*, Institute for Simulation and Training, University of Central Florida.
- Danisas, K., Smith, S. H., and Wood, D. D. (1990). "Sequencer/Executive for Modular Simulator Design", *Technical Report IST-TR-90-1*, Institute for Simulation and Training, University of Central Florida, 16 pages.
- DIS Steering Committee (1993). "The DIS Vision: A Map to the Future of Distributed Simulation", *IST Technical Report*, 47 pages.
- Foley, J. D., van Dam, Andries, Feiner, S. K., Hughes, J. F. (1991). "Compute Graphics Principles And Practice", 2nd edition, Addison-Wesley Publishing Company, Inc.
- Gonzalez, G., Mullally, D. E., Smith, S. H., Vanzant-Hodge, A. F., Watkins, J. E., and Wood, D. D. (1990). "A Testbed for Automated Entity Generation in Distributed Interactive Simulation", *Technical Report IST-TR-90-15*, Institute for Simulation and Training, University of Central Florida, 37 pages.
- Hwang, Y. K. and Ahuja, N. (1992). "Gross Motion Planning--A Survey", *ACM Computing Surveys*, Vol. 24, No. 3, pp.219-291.
- Lee, J. J. and Fishwick, P. A. (1994), "Simulation-Based Planning for Computer Generated Forces", *Proceeding of the 4th Conference of Computer Generated Forces and Behavioral Representation*, Orlando, FL, May 4-6, 1994, pp. 451-459.
- Loper, M. L., Thompson, J. R., and Williams, H. L. (1991). "Simulator Networking: What Can It Offer The Training Community?", *Military Simulation & Training*, Issue 4 1991, pp. 11-14.
- Mitchell, J. S. B. (1988). "An Algorithmic Approach to Some Problems in Terrain Navigation", *Artificial Intelligence*, Vol. 37, pp. 171-201.
- NASA (1991), "Potential-Field Scheme for Avoidance of Obstacles by a Robot", *NASA Tech Brief KSC-11491*, John F. Kennedy Space Center, Fl., pp. 1-43.
- Petty, M. D. (1992). "Computer Generated Forces in Battlefield Simulation", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 56-71.
- Roos, T. and Noltemeier, H. (1991). "Dynamic Voronoi diagrams in Motion Planning", pp 227-236, *Computational Geometry: Methods, Algorithms, and Applications (1991)*, Proceedings of the International Workshop on Computational Geometry (1991), Bern, Switzerland.
- Smith, J. E. (1994), *ModSAF Programmer's Guide: LibCTDB*, Loral Advanced Distributed Simulation, Cambridge, Massachusetts.
- Smith, S. H., Karr, C. R., Petty, M. D., Franceschini R. W., and Watkins, J. E. (1992a). "The IST Computer Generated Forces Testbed", *Technical Report IST-TR-92-7*, Institute for Simulation and Training, University of Central Florida.

Smith, S. H., and Petty, M. D. (1992b). "Controlling Autonomous Behavior in Real-Time Simulation", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 27-40.

Thorpe, C. E. (1984). "Path Relaxation: Path Planning for a Mobile Robot", *CMU-RI-TR-84-5*, Carnegie-Mellon University The Robotics Institute Technical Report, April 1984.

Winston, Henry Patrick (1992). *Artificial Intelligence*, Third Edition, Addison-Wesley, 1992.

9. Glossary

Table 9-A Glossary.

Term	Description
ARPA	Army Research Projects Agency.
CGF	Computer Generated Forces.
DIS	Distributed Interactive Simulation.
MRS	Mutually Reachable Set.
OIN	Obstacle Identification Number.
OS	Obstacle Segment.
OSA	Obstacle Segment Abstractions.
sample points	Points in a grid cell that lie on the extended and final routes.
shared point	Common point between two or more different Obstacle Segments that lie in adjacent grid cells.
SAFOR	Semi-Automated FORces.
SIMNET	SIMulation NETwork.
STRICOM	Simulation TRaining and Instrumentation CCommand.
TDB	Terrain DataBase.

