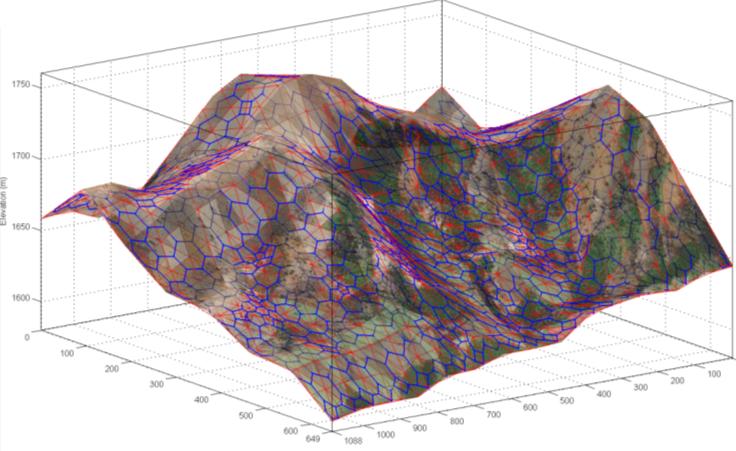
**User’s Manual for the**

**Sensor Suite Evaluation System (SSES)**

**Prototype Version 1.0**



**6 May 2009**

Table of Contents

[About SSES 4](#_Toc229457013)

[Installing SSES 4](#_Toc229457014)

[Running SSES under MATLAB 4](#_Toc229457015)

[Running SSES without MATLAB 5](#_Toc229457016)

[Hardware and software requirements 5](#_Toc229457017)

[Technical support 5](#_Toc229457018)

[Running SSES 6](#_Toc229457019)

[Display and environment layout 6](#_Toc229457020)

[SSES data and layers 7](#_Toc229457021)

[SSES menus 7](#_Toc229457022)

[Initializing a project 8](#_Toc229457023)

[Loading and saving projects 9](#_Toc229457024)

[Importing external data 9](#_Toc229457025)

[Figure manipulation 10](#_Toc229457026)

[Control panel selection 10](#_Toc229457027)

[Terrain and obstacle editing 10](#_Toc229457028)

[Terrain network general comments 15](#_Toc229457029)

[Sensor Plan Editing 16](#_Toc229457030)

[Sensor editing mode 17](#_Toc229457031)

[Display options 22](#_Toc229457032)

[Sensor Plan Performance 29](#_Toc229457033)

[SSES data structures and algorithms 33](#_Toc229457034)

[Site data 33](#_Toc229457035)

[Terrain data 33](#_Toc229457036)

[Mobility data 34](#_Toc229457037)

[Obstacle data 35](#_Toc229457038)

[Sensor data 35](#_Toc229457039)

[Display data 36](#_Toc229457040)

[Detection calculation 37](#_Toc229457041)

[Line-of-sight calculation 38](#_Toc229457042)

[Intruder path calculations 38](#_Toc229457043)

[Reference 40](#_Toc229457044)

[Figure 1: SSES main window. 6](#_Toc229457901)

[Figure 2: Toolbar controls. 8](#_Toc229457902)

[Figure 3: Edit Terrain control panel. 11](#_Toc229457903)

[Figure 4: Terrain network construction. 13](#_Toc229457904)

[Figure 5: Sensor Editor control panel. 16](#_Toc229457905)

[Figure 6: Volume sensor azimuth limits. 20](#_Toc229457906)

[Figure 7: Effect of sensor elevation on surveillance coverage. 21](#_Toc229457907)

[Figure 8: Display Options control panel. 23](#_Toc229457908)

[Figure 9: Face Pd display. 24](#_Toc229457909)

[Figure 11: Relative movement time display. 25](#_Toc229457910)

[Figure 12: Site with obstacles and line sensors. 27](#_Toc229457911)

[Figure 13: SSES display legend. 28](#_Toc229457912)

[Figure 15: Minimum path comparison. 32](#_Toc229457913)

[Figure 16. Site data structure. 34](#_Toc229457914)

[Figure 17. Terrain data structure. 34](#_Toc229457915)

[Figure 18. Mobility data structure. 35](#_Toc229457916)

[Figure 19. Obstacle data structure. 35](#_Toc229457917)

[Figure 20. Sensor plan data structure. 36](#_Toc229457918)

[Figure 21. Sensor element structure. 36](#_Toc229457919)

[Figure 22. Sensor plan data structure. 37](#_Toc229457920)

[Figure 23. Line-of-sight algorithm. 38](#_Toc229457921)

[Figure 24. Generic shortest path algorithm. 39](#_Toc229457922)

**SSES Project Team:**

Helen Anderson

Steven Beres

Joseph Shaw

Timothy Valadez

Sponsor: Dr. K. C. Chang

Faculty Advisor: Dr. Thomas Speller

**User’s Manual principal author:**

Joseph Shaw

© 2009, SSES Applications. All rights reserved.

# About SSES

The Sensor Suite Evaluation System (SSES) is a graphical user interface (GUI) based software application that supports the design and evaluation of electronic security systems (ESS) used to detect intrusion into defended sites. SSES includes functionality to support characterization of a defended site and construction of mathematical representations of site topography and features. SSES provides the capability to select, place and orient sensors, view the effective coverage area of individual multiple sensors, and calculate the resulting probability of detection (Pd) against specific threat types. SSES automatically calculates and displays “worst case” threat intrusion routes to minimize travel time, minimize Pd, or minimize expected defender reaction time. These capabilities support iterative design and assessment of sensor plans.

# Installing SSES

SSES was developed primarily using MATLAB Version 7.4.0 (2007a) running under Microsoft Windows. The prototype has also been tested under MATLAB Versions 7.5.0 (2007b) and 7.7.0 (2008b).

The SSES CD contains two versions of the SSES application, along with several sample SSES project files.

## Running SSES under MATLAB

MATLAB source code is provided in the **SSES\_prototype\_code** folder. To run SSES under MATLAB, simply copy the contents of the **SSES\_prototype\_code** folder from the CD to a location on the MATLAB path. The **ssesPrototype.m** and **ssesPrototype.fig** files should be placed in the same folder. The SSES application is launched by typing **ssesPrototype** at the command line. It is not necessary to place the project folders in the MATLAB path. However, setting the MATLAB current directory to the applicable project folder will reduce the number of steps to access project data.

Although not required, installation of the MatlabBGL library is recommended. The library, written by David Gleich, implements shortest path algorithms using precompiled .mex files that are significantly faster than the simple shortest path algorithm provided with SSES. The MatlabBGL library may be downloaded from the MATLAB File Exchange at:

http://www.mathworks.com/matlabcentral/fileexchange/10922

## Running SSES without MATLAB

For Microsoft Windows users without access to MATLAB, SSES is also provided as a compiled executable application. To run the executable version:

1. Copy all contents of the **SSES\_prototype\_compiled** folder to the desired location on a local drive.

2. Install the MATLAB Component Runtime Version 7.7 by launching the **MCRInstaller** application located in the **MATLABruntimeComponents** subfolder.

3. Add the component runtime to the Windows path by opening the command window and entering (as a single line) the path for 32 or 64 bit operating systems as appropriate:

PATH=C:\Program Files\MATLAB\MATLAB Component Runtime\v77\runtime\win32;%PATH%

PATH=C:\Program Files\MATLAB\MATLAB Component Runtime\v77\runtime\win64;%PATH%

4. Run SSES by launching the **ssesPrototype.exe** application

For additional details refer to the readme.txt file in the **SSES\_prototype\_compiled** folder.

## Hardware and software requirements

SSES minimum hardware and software requirements have not been established. The application has been tested successfully on a Pentium 4 with 1 GB of ram under Windows XP. SSES is graphics and display intensive. Displays have been optimized for a screen resolution of 1680x1050. A minimum display resolution of 1280x1024 is recommended. Lower resolutions will result in significant clipping of GUI labels and elements encroaching on each other.

## Technical support

Formal technical support for SSES does not exist. While a reasonable attempt has been made to ensure the code is robust, the SSES prototype remains very developmental and has not been comprehensively tested. Please contact the author with problems or suggestions for improvement at: [jshaw@spa.com](mailto:jshaw@spa.com) or [jshaw3@gmu.edu](mailto:jshaw3@gmu.edu).

# Running SSES

## Display and environment layout

Users interact with SSES primarily through the SSES main window shown in Figure 1. The SSES display is implemented as a MATLAB figure GUI with both display and control elements.

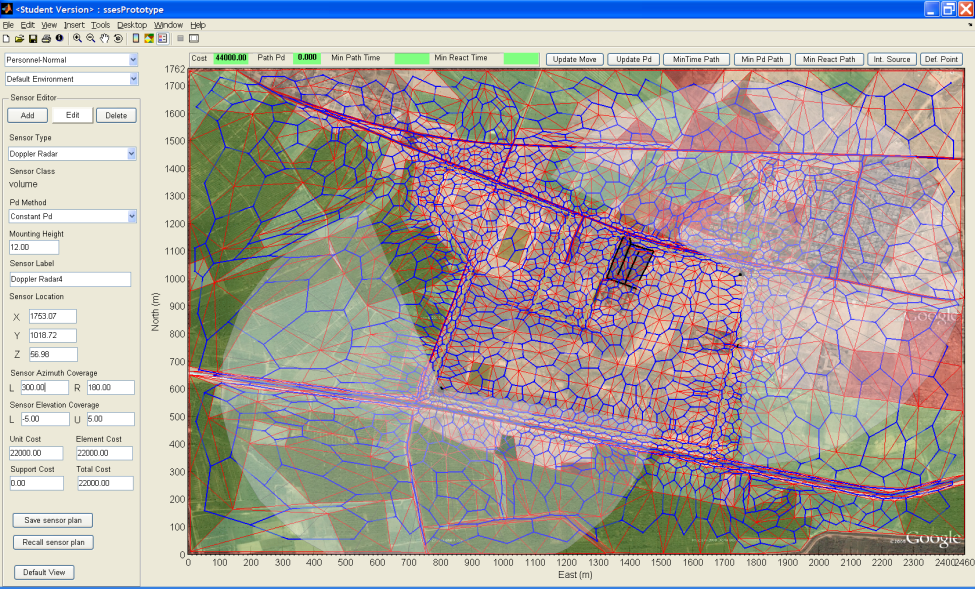


Figure 1: SSES main window.

The SSES window is divided into three main areas. The graphic window occupying most of the screen provides a layered display of the site and current sensor plan. Controls and displays for sensor plan measures of effectiveness (MOE) are located immediately above the main display window. The vertical box along the left side of the display contains control panels for 1) creating and manipulating the site terrain model, 2) developing and modifying the sensor plan, and 3) controlling display options, as well as selecting the threat and environment. The toolbar along the top edge of the window contains controls for loading and saving SSES projects, importing external data, and selecting the display, terrain editing or sensor control panels, as well as core MATLAB figure manipulation tools. These displays and controls are described in detail in the following sections.

## SSES data and layers

SSES data and algorithms are organized around a core network model consisting of a terrain partition network and a mobility network model. The terrain network partitions the site into a set of triangular terrain faces defined by terrain network edges and vertices. The terrain network forms a sparsely connected planar graph where terrain within each face is treated as homogenous and assigned a specific terrain type. Although the terrain network is planar in a graph theory sense, the terrain network incorporates elevation data to form a 3-dimensional polygonal surface. Sensor line-of-site blockage is calculated based on intersection of the line connecting the sensor and target locations with terrain faces. Construction of the terrain network in the SSES prototype version is primarily a manual process and is described in detail in the terrain editing section below. Obstacles and line sensors are associated with terrain network edges. Volume sensor detection probabilities are calculated for terrain network faces. Display controls are provided for terrain faces, edges and vertices.

The mobility network is the dual graph of the terrain network and is formed by connecting the centroids of the terrain network faces. All intruder movement is assumed to occur along the edges of the mobility network. The mobility network is generated automatically by the SSES application. Mobility network base edge costs are determined by threat type, type of terrain traversed, and the distance traveled. Movement costs may be increased by obstacles placed on the network edges. Display controls are provided for mobility network edges and vertices.

To assist in construction of the terrain network, SSES includes an imagery “layer” that allows imagery obtained from external sources to be displayed as either a 2-D or 3-D surface. An elevation data layer contains elevation grid data from an external source such as Digital Terrain Elevation Data (DTED) or interpolated contour data.

Sensor plans are implemented as a sensor “layer” superimposed on the terrain network. Line sensors are associated with terrain network edges and may not be positioned independently of the associated edge. Volume sensors are positioned in 3-dimentional space by translating the sensor in the east-west and north-south directions and specifying sensor mounting height above the terrain surface.

## SSES menus

Figure 2 shows the toolbar located in the upper left part of the SSES window.

Initialize project

Open saved project

Save current project

Import external database data

Zoom out main display

Zoom in main display

Pan main display

Rotate main display (3D)

Select terrain editing control panel

Select display control panel

Select sensor editing control panel

Not used

Print



Figure 2: Toolbar controls.

## Initializing a project

 To initialize a new project start a new session of SSES then left-click the **Initialize SSES Project Interactively** icon. This initiates a query script that allows the user to specify site data, coordinate limits, and display limits, and to load imagery and elevation data.

Image data can be imported from most typical formats including .jpg, .gif, .png, and .bmp. SSES represents texture image data as indexed images and performs image conversion using the MATLAB Image Processing Toolbox **rgb2ind** function. If the Image Processing Toolbox is not available color images will be converted to grayscale.

Elevation data must be read from a MATLAB .mat file that contains the variables “xMesh”, “yMesh”, and “zElev”, where xMesh and yMesh are plaid x and y coordinate arrays produced using the MATLAB **meshgrid** function or equivalent and zElev contains the corresponding elevation data. This data must be produced externally, for example by using the MATLAB Mapping Toolbox **dteds** and **meshgrat** functions and converting the latitude and longitude results to Cartesian coordinates. The site imagery is “texture wrapped” onto the elevation data grid so the coordinate limits of the elevation data and the image data should match or image distortion will result.

Note that site initialization may be performed with an existing project but this is generally not recommended since the existing terrain and sensor data is not erased and will usually not match the new site coordinate system. An exception to this rule is that it may be desirable to load a baseline project with terrain, threat, and sensor database data (but without a sensor network or sensor plan) prior to initializing to eliminate the need to import external data as discussed below.

## Loading and saving projects

To save the current project left-click the **Save SSES Project** icon. SSES projects are saved as MATLAB .mat files that contain the project data structure variables. By default SSES project files are saved as \*SSES.mat. The file selection windows follow standard windows directory conventions. A description of SSES project variables is provided in the data structures and algorithms section. Experienced MATLAB users may find it convenient to manipulate project data from the MATLAB command line to perform operations that are not supported in the SSES prototype version.

The print control is held over from MATLAB core figure functionality and has not been tailored for use with SSES. The print button can be used to print the GUI window but does not provide control over output formatting and generally produces a truncated print. Refer to MATLAB documentation for instructions on printing figures from the MATLAB command line. The most effective way to print SSES images is to screen capture the display and paste the contents of the clipboard into another application such as Midrosoft Word or PowerPoint.

To load a previously saved project left-click the **Open SSES Project** icon. This opens a navigation window to allow selection of the applicable file.

## Importing external data

To import external database data left-click the **Import SSES Project Data** icon. This will open an **Import External Data** panel with **Import Threat Data**, **Import Terrain & Feature Data**, and **Import Sensor Data** pushbuttons. An **Import Environment Data** pushbutton is also shown but is not functional in the SSES prototype version. Left clicking the threat, terrain or sensor pushbuttons opens a file selection window allowing the user to select an Excel data file in .xls or .xlsx format. A description of the data and format for each type of file is provided in the data structures and algorithms section. The terrain data file includes both terrain and obstacle data. The SSES prototype does not include functionality to edit or save threat, terrain or sensor databases. This data must be edited in Excel or directly from the MATLAB command line. Once loaded, the data is stored in the threat, terrain and sensor project variables and loaded when the saved project is recalled. The Excel import function relies on MATLAB being able to start Excel as a COM server from MATLAB. If your system does not have this capability data import may not work. In this case it may be necessary to manually copy the **ter.terrainList**, **obs.obstacleList**, **ess.sensorList**, **threat.threatDB** fields from one of the provided demo projects to a new project from the MATLAB command line. Importing data is a relatively slow operation so it may be desirable to start SSES, import threat, terrain and sensor data, and save the resulting project. This project may then be reloaded prior to site initialization to avoid the need to re-import data.

## Figure manipulation

SSES figure manipulation tools operate in the same manner as standard MATLAB figure manipulation tools. Refer to MATLAB help on the **zoom**, **pan**, and **rotate3d** functions. MATLAB command line functions, (e.g., view, xlim, ylabel, etc.) will operate on the main display. For large projects (large image files, terrain maps with many points, dense elevation data grids) figure manipulation using the tools may be slow and it may be convenient to perform these operations from the command line.

## Control panel selection

The **SSES display options pushbutton** selects display mode, activates the display options control panel, and deactivates terrain or sensor editing mode if currently selected.

The **Edit Terrain** pushbutton selects terrain edit mode, activates the terrain editing control panel, and deactivates sensor editing or display options mode if currently selected.

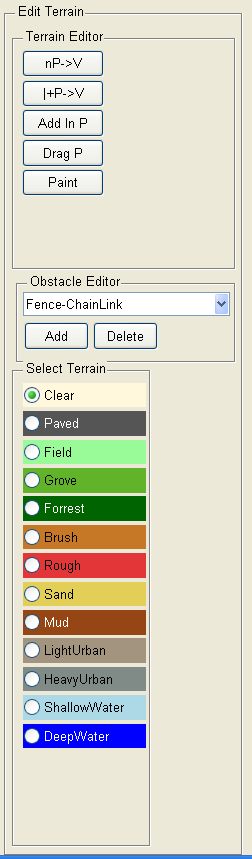
The **Edit Sensors** pushbutton selects terrain edit mode, activates the terrain editing control panel, and deactivates sensor editing or display options mode if currently selected.

Each of these modes is discussed in detail below.

## Terrain and obstacle editing

The Edit Terrain provides controls for constructing and adjusting the terrain network model, and adding and deleting obstacles. Figure 3 shows the layout of the **Edit Terrain** control panel

There are two basic methods of constructing the terrain partition network. The first method, which must be used to start a new project, is to connect a set of vertices to form a network. The second method is to extend the network by adding new points to an existing network or connecting vertices that are not currently connected. Controls are also provided to drag existing vertices to new positions, and to assign terrain type to terrain faces. These operations do not affect network topology.

A limitation of the SSES prototype version is that while vertices can be added to create new network edges and faces, there is currently no mechanism to delete unwanted elements. The presence of “extra” faces and edges does not present a significant problem since they can always be positioned to obtain the desired partition with some faces having extra subdivisions. Large numbers of extra elements can have some impact on graphics Pd, and path calculation performance. Terrain editing operations are described in the following sections.

Terrain type selection

Construct terrain network from n new vertices

Extend terrain network using existing edge and new vertex

Subdivide terrain face adding new internal vertex

Drag terrain network vertex

Change terrain face type to the current terrain type selection

Select obstacle type

Add new obstacle

Delete existing obstacle

Figure 3: Edit Terrain control panel.

The **Make terrain from n vertices** pushbutton allows the user to place an arbitrary number of vertices by left-clicking on the main display. Vertex X-Y locations are determined by the cursor location. Vertex Z location is determined by interpolating the elevation grid data and placing the vertex at the estimated surface elevation. Vertex locations are normally selected by “tracing” over the terrain image to define significant terrain features and identify areas with similar terrain type. To assist with vertex placement the cursor current X and Y location is displayed at the above the upper left corner of the main display. Vertices are displayed as red dots as they are laid down. When the desired vertices have been positioned, left-clicking on a vertex terminates vertex entry and the application automatically generates a terrain partition network using Delaunay triangulation to ensure that the resulting network is planar, and generally has well proportioned faces. Note that while the user controls the placement of the vertices the edges are assigned automatically. The corresponding dual mobility network forms a Voronoi diagram. The current terrain type is assigned to all faces.

Sections of the terrain partition can be constructed sequentially by repeating the process. In order to ensure the terrain network remains planar and triangular, new vertices are not permitted to be placed within existing terrain faces, and new edges are not permitted to intersect existing network edges. Vertices are tested to ensure they do not fall existing terrain faces as they are entered and disallowed vertices are not placed. Since the edge locations are not known until the Delaunay triangulation has been performed they cannot be checked in advance. Instead, the edges are tested after the new network has been constructed and any impermissible edges are deleted. Any vertices and edges that do not form part of an allowed face are also deleted. It is possible for the entire new network to be disallowed.

Manual partitioning of the entire site can be a tedious process. To reduce operator workload, the first time **Make terrain from n vertices** is selected the user is offered the option to start with a regular terrain grid. If this option is selected, a popup box allows the user to specify either the length of the base, or the length of both the base and height of the grid triangles. The triangular grid is generated, oriented with the triangle bases parallel to the X axis. Once the grid covering the site has been generated, the user may continue to add additional vertices as described above. The automatic grid generation option is only available if no terrain vertices have been placed previously. In addition, use of the regular grid option will effectively preclude follow on use of the **Make terrain from n vertices** method since any new vertices would by definition fall within an existing face. Grid faces may be subdivided using the **Add interior vertex** control or distorted by dragging vertices.

Note that multiple disconnected terrain networks may be generated. This does not pose a problem during network construction, but can cause failure (e.g., infinite looping) of the path and line-of-sight algorithms in the SSES prototype version. A fully connected terrain network should be formed prior to calculating sensor coverage or intruder paths.

The **Extend terrain network with edge and vertex** pushbutton allows the user to extend the terrain network one face at a time by left-clicking on an edge to select it, then left-clicking on an existing vertex or on the main display to add a new vertex. If the resulting new network face is allowable then it is incorporated into the network along with any new edges and the new vertex if applicable. In order to ensure the resulting network remains planar, the edge selected must be an exterior edge (i.e. adjacent to a single terrain face), any new vertex must not lie within any existing terrain face, any existing vertex must be adjacent to an exterior edge, and any new edges must not intersect any existing edge. The new face is assigned the current terrain type. In addition to the X and Y coordinates of the cursor, the lengths of the two prospective new edges are displayed at the top of the main display window.

Note that it is possible to create networks that have internal “holes” as shown in Figure 4.

**e**

**d**

**c**

**a**

**b**

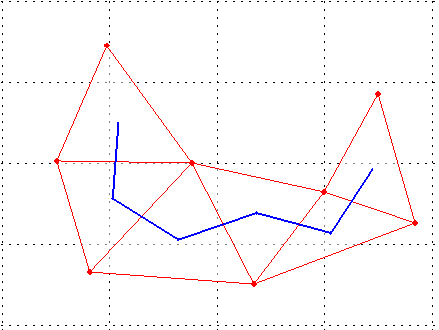
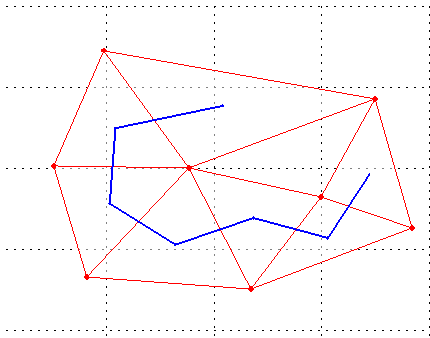
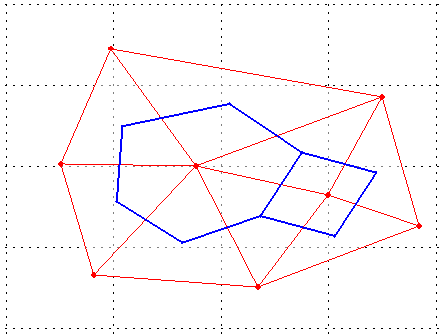


Figure 4: Terrain network construction.

The panel on the left side of Figure 4 shows a typical terrain network which includes edge **a** and vertex **b**. Connecting edge **a** and vertex **b** results in the well formed network shown in the middle panel that does not include the triangle formed by edges **c**, **d**, and **e**, which are exterior edges of the network. By selecting any of these edges and the opposite vertex the “missing” face is added removing the hole as shown in the right hand panel.

The existence of terrain network “holes” does not cause a problem for the path generation functions, but sensor lines-of-sight that traverse the “hole” may show as blocked even though the intervening terrain lies below the line-of-sight.

Since new faces are always created using existing edges networks constructed using the **Extend terrain network with edge and vertex** function are always connected.

The **Add interior vertex** pushbutton enables addition of new vertices within existing terrain network faces in order to subdivide the face into three smaller triangular faces by connecting the face vertices to the new interior vertex. The resulting network will always remain planar. The new vertex may be placed anywhere within an existing terrain face. One of the new terrain network faces will inherit the terrain type of the original face. The other faces will be assigned the current terrain type.

The **Drag vertex** pushbutton enables vertex dragging. To drag a vertex select it by left-clicking on it (selection is indicated by the vertex color changing to green), then while continuing to hold the mouse button down, dragging the vertex to the new position and releasing the mouse button. Vertex dragging is terminated if the new vertex location would result in edges intersecting. Vertex position checking is not robust in the SSES prototype version and care should be taken to avoid dragging the vertex to locations that result in edge intersections.

The **Paint terrain type** pushbutton enables modification of terrain face type. When enabled, left-clicking on a terrain face will assign the currently selected terrain type to that face. Terrain type must be assigned one terrain face at a time.

The edit terrain pushbutton controls are, together with the Obstacle Editor controls, mutually exclusive. Selecting any of these controls will automatically deselect the active control. The active control is indicated by the pushbutton color changing from gray to white. Clicking the active control will deselect it. It is not necessary to have an active control.

The **Select Terrain panel** allows the user to select a “current” terrain type. The current terrain type will be assigned to all new terrain faces created using the terrain editing functions. The current terrain type may also be assigned to existing terrain faces when using the **Paint terrain type** control. The terrain type selections are mutually exclusive. The button colors provide a ready reference to the terrain face color coding when the terrain type is selected as the terrain partition face color option.

The **Obstacle Editor** panel controls are used to add and delete mobility obstacles in the site design. Obstacles are always associated with a terrain partition edge (for display purposes) and the corresponding mobility network edge (for calculating movement effects.) The pull down menu at the top of the panel is used to select the obstacle type. When **obstacle editing** **add** mode is enabled, left-clicking on a terrain network edge will place the current obstacle type on that edge. The SSES prototype version only allows a single obstacle per edge. Adding a new obstacle to an edge will replace the existing obstacle. No warning is issued before replacement. When **obstacle editing delete** mode is selected left-clicking on an obstacle deletes it. Obstacle editing controls are mutually exclusive with terrain editing controls.

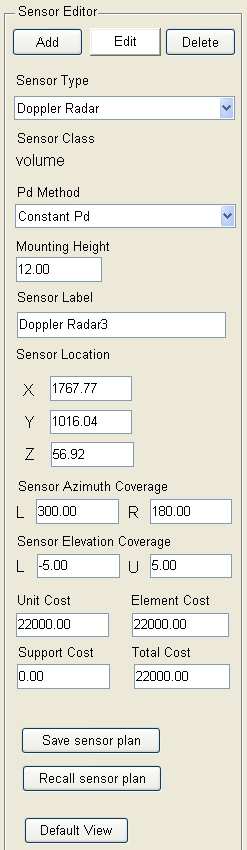
## Terrain network general comments

Terrain network selection involves a tradeoff between the size of the terrain faces, and the number of faces needed to cover the region of interest. Use of smaller terrain faces improves the accuracy and resolution of movement time and sensor coverage and Pd calculations, and will allow a tighter fit between the terrain faces and the elevation grid data. Conversely, increasing the number of terrain faces increases storage and processing requirements. The primary limitations appear to be the speed of MATLAB graphics operations, the time required for sensor line-of-sight calculations, and calculation of intruder paths. A reasonable compromise may be to use a relatively fine terrain network (small faces) in areas of high interest (e.g., near fence lines, around defended areas, and near choke points) and in areas where sensors are likely to be placed, and use a coarser (large faces) in areas of lower interest. The SSES prototype has been tested successfully using terrain networks with ~1000 vertices, ~3000 edges, and ~2000 faces.

When operating in **3D Imagery** mode the site image is “wrapped” over the site elevation data. Because terrain network vertices will generally not coincide with elevation grid data points, terrain faces will often not fit the image surface. As a result, the terrain network faces and image surface will intersect. This can make it difficult to accurately identify terrain features from the image, or to see the structure of the terrain network. It may be necessary to adjust the opacity of the terrain faces and image data, or to force the image data to “underlay” the terrain network by selecting **2D Imagery** mode. Use of these controls is discussed in the display control section.

Although main display can be rotated in 3D to examine site topology and sensor coverage, terrain editing should normally be conducted in the default top-down view. The cursor position algorithm in the SSES prototype version assumes a top-down view and does not correct for rotation effects. As a result, the cursor position may not be projected correctly onto the terrain surface when the display is rotated.

## Sensor Plan Editing

SSES supports two classes of sensors: “line sensors” and “volume sensors”. Line sensors are always associated with a terrain network edge and the corresponding mobility network edge, and detect intruders as they traverse the edge.

Sensor unit cost edit box

Edit mode selection controls

Sensor type pull down menu

Sensor class

(not editable)

Sensor Pd method pull down menu

(Not implemented in the SSES prototype version)

Sensor mounting height edit box

Sensor position edit boxes

Sensor azimuth coverage limit edit boxes

(Volume sensors only)

Sensor label edit box

Sensor elevation coverage limit edit box

(Volume sensors only)

Sensor support cost edit box

Sensor element cost edit box

Sensor total cost edit box

Save sensor plan pushbutton

Recall sensor plan pushbutton

Set default view pushbutton

Figure 5: Sensor Editor control panel.

Line sensors do not have orientation, mounting height, or effective range parameters, and do not require a line-of-sight to operate. Volume sensors may be placed anywhere on the site and detect the presence of intruders within terrain network faces / at the corresponding mobility network vertices. Volume sensors generally have limited field-of-view in azimuth and elevation, and only detect intruders within their effective range. Volume sensors also require a clear line-of-sight from the sensor to the target location for detection. Figure 5 shows the layout of the Sensor Editor control panel

## Sensor editing mode

When **Add** mode is selected the user can add an additional sensor of the current type to the sensor plan by left-clicking in the main display. Line class sensors may only be added along a terrain network edge. The new line sensor will be displayed as a colored line overlaying the selected terrain edge. The SSES prototype version only allows a single line sensor to be placed on each terrain network edge. If a sensor already exists on the edge a popup window will ask the user to confirm the sensor placement. If confirmed the current sensor will be replaced with the new sensor type. Otherwise the operation is canceled and the current sensor is retained. Volume sensors may be placed anywhere on the site and may be located arbitrarily close to other volume sensors or line sensors. Volume sensors are indicated by a triangular icon showing the sensor location and and a colored overlay showing nominal sensor coverage. New sensors are assigned default values which may be edited immediately from the **Add** mode, or at a later time by using **Edit** mode.

When **Edit** mode is selected the user can select sensors for editing by left-clicking on the sensor display. Line sensors are selected by clicking on the terrain network edge where the sensor is located. Volume sensors are selected by clicking on the sensor icon Volume sensor positions may also be adjusted by clicking the sensor icon and dragging the sensor to a new position while holding the left mouse button down. For complex terain networks, the sensor position may lag the cursor position by several seconds. It is not necessary to wait for the sensor to “catch up” before releasing the mouse button The final sensor position will be the point where the mouse button is released. Line sensors can not be dragged since the sensors are “attached” to the terrain network edges. To adjust line sensor position the terrain network edge vertices must be adjusted using the **Drag vertex** terrain editing control. Once a sensor is selected its parameters may be adjusted using the controls described in the following sections.

When **Delete** mode is selected sensors may be deleted by clicking on the applicable sensor display in the same manner as for sensor editing. If user clicks the sensor using the left mouse button the user is prompted to confirm deletion. If not confirmed the deletion operation is cancelled. If the user right clicks on the sensor it is deleted without confirmation.

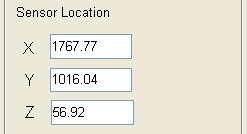
The **Sensor Type** pull down menu is used to select the current sensor type when in **Add** mode. Selecting a sensor type sets the sensor editing display to reflect the sensor type and uses these settings for subsequent sensor additions. It does not affect existing sensor parameters. Changing the sensor type of emplaced sensors is not permitted the SSES prototype version. Instead the sensor must be deleted and re-added.

The Sensor Class display indicates whether the current sensor is line class or volume class. It provides information only and is not editable.

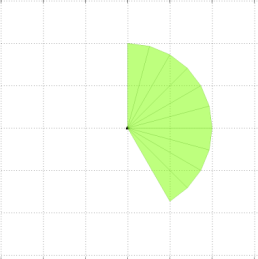
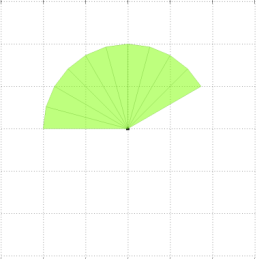
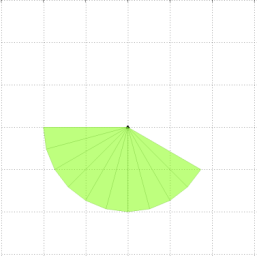
The **Pd Method** pull down menu and the model and lookup Pd methods have not yet been implemented in the SSES prototype version. Attempting to select a Pd method other than “Constant Pd” has no effect. 

The **Mounting Height** edit control allows the user to adjust the mounting height of volume sensors only. Mounting height specifies the volume sensor distance relative to the ground. The mounting height is added to local ground elevation to determine actual volume sensor elevation. The sensor Z coordinate is updated automatically. Increasing the sensor mounting height may improve volume sensor field-of-view and effective coverage area by allowing the sensor to see over terrain that blocks sensor line-of-sight. However, volume sensors typically have a limited elevation coverage extent and increasing sensor mounting height can also increase the size of the “blind zone” in the vicinity of the sensor. Consequently, it may be desirable to check and adjust sensor elevation limits when mounting height is changed. The mounting height for line sensors is always zero and may not be adjusted with the **Mounting Height** control.

New sensors are assigned a label based on the sensor type and a sensor index. An alternate name may be provided using the **Sensor Label** edit box. In the SSES prototype version the sensor label is only used for display purposes and multiple sensors may be assigned the same label. In future versions the label may be used to select sensors via pull down menu and unique sensor labels will be required. The use of unique sensor labels is recommended to allow compatibility with future SSES versions.

For volume sensors the **Sensor Location** edit control provides a precise method for specifying sensor location and elevation. The user may enter sensor X, Y, and Z coordinates to update the current sensor position in either **Add** or **Edit** mode. Manually editing the X or Y coordinate will also update the Z coordinate to reflect local terrain elevation plus the specified mounting height. Manually editing the Z coordinate will cause the mounting height to be updated to reflect the mounting height needed to provide the specified sensor height for the local terrain height. If a line sensor is selected a pair of X, Y, and Z coordinates will be displayed indicating the location of the line sensor location / associated terrain edge vertices. These values are not editable. Line sensor location can only be adjusted by dragging the associated terrain edge vertices.

The **Sensor Azimuth Coverage** edit control allows the user to specify the left (i.e. counter-clockwise) and right (clockwise) limits of the sensor field-of-view. Most volume sensors have a field-of-view of less than 360 degrees. Only intruders that are within the sensor field-of-view are detectable. When a new volume sensor is created the left azimuth is set to 000 degrees (north) and the right azimuth limit is set to the maximum azimuth extent. The left and right azimuth limits may be adjusted in any order. If adjusting either azimuth limit would result in a azimuth extent that is greater than the maximum field-of-view allowed for that sensor type, the other limit will be adjusted to limit the field-of view to the maximum allowed. The azimuth limits may be adjusted to limit the field-of-view to less than the maximum. There is normally no operational advantage to doing this, but reducing the field-of-view can reduce the time required to perform line-of-site and Pd calculations. Figure 6 shows the effect of adjusting azimuth limits for a sensor with a maximum azimuth extent of 150 degrees. The left panel shows the default limits for the new sensor 000-150 degrees relative to the positive Y axis. The center panel shows the effect of adjusting the left azimuth limit to 270 degrees – the right azimuth limit is automatically set to 060 degrees. The right hand panel shows the effect of adjusting the right azimuth limit to 270 degrees – the left azimuth limit is now automatically set to 120 degrees. Azimuth limits are not applicable to line sensors and the azimuth edit controls are not shown when a line sensor is selected.



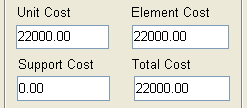
Effect of setting right azimuth limit to 270

Effect of setting left azimuth limit to 270

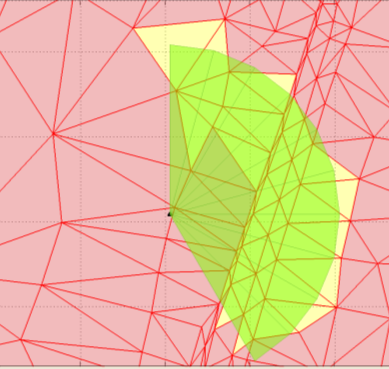
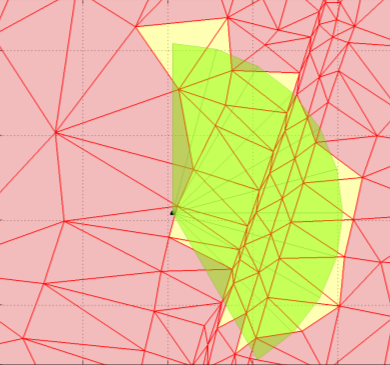
Default azimuth limits 000-150

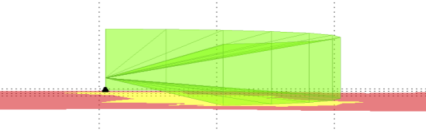
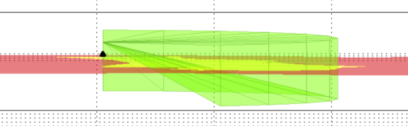
Figure 6: Volume sensor azimuth limits.

The **Sensor Elevation Coverage** edit control operates similar to the **Sensor Azimuth Coverage** edit control, and allows the user to specify the lower and upper elevation limits of the sensor field-of-view. Most volume sensors have a field-of-view of significantly less than 180 degrees.  Only intruders that are within the elevation field-of-view are detectable. When a new volume sensor is created its elevation field-of-view is centered on the horizon. The sensor’s lower and upper elevation limits are set to –1/2 and +1/2 of the maximum elevation limits respectively. Sensor lower and upper elevation limits may be adjusted in any order. If adjusting an elevation limit would result in the elevation coverage exceeding the maximum allowed, the other limit will be adjusted automatically to the maximum limit. Limited sensor depression angle can create a detection gap in the vicinity of the sensor. Figure 7 illustrates this effect and how it can be mitigated by adjusting elevation limits. Elevation limits are not applicable to line sensors and the elevation edit controls are not shown when a line sensor is selected.

Sensor cost data is displayed, and may be edited for both line and volume sensor using the **Unit Cost**, **Element Cost**, **Support Cost**, and **Total Cost** edit boxes.

Unit cost is a function of the specific type of sensor selected and is loaded automatically from the sensor database when the sensor is placed. For volume sensors unit cost represents the nominal cost of the sensor plus any fixed installation and support costs. For line sensors Unit cost is the nominal cost per meter of the installed sensor. Unit cost may be overridden by entering a new value in the **Unit Cost** edit box. This change is applied to the current sensor. When in add mode, the overridden cost will be used for additional sensors of the same type. However, the sensor database is not updated and selecting a new (or the same) type sensor from the Sensor Type pull down menu will restore the default values. The updated unit cost will automatically be propagated to the element and total costs as well.





Sensor with -14 to 1o elevation coverage

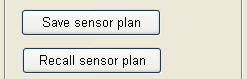
Sensor with -5 to 10o elevation coverage

Figure 7: Effect of sensor elevation on surveillance coverage.

Element cost is the cost of a specific instance of the sensor. For volume sensors element cost is the same as sensor unit cost. For line sensors element cost is determined by multiplying the unit cost (per meter) by the length of the sensor run (length of the associated terrain network edge.) SSES calculates this value automatically when the sensor is placed. The SSES prototype version currently does not update the element cost if terrain network is subsequently adjusted by dragging terrain vertices. The user may override the calculated value by entering a new cost in the Unit Cost edit box. The new element cost is automatically propagated to total cost.

Support cost provides a mechanism for the user to enter additional support costs, e.g., the cost of sensor mounting or network equipment. Sensor cost is always set to zero by default and may be modified by entering a new value in the **Support Cost** edit box. The manually entered support cost will carry forward to new sensors until a different (or the same) sensor type is selected.

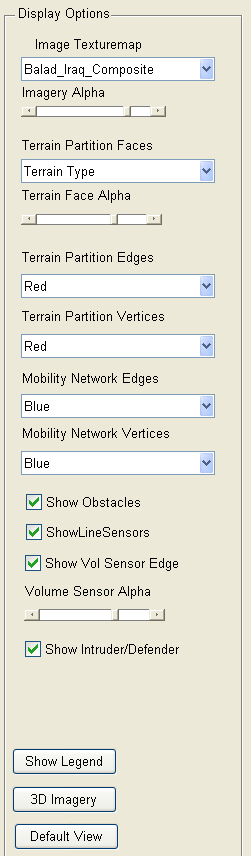
Total cost is calculated automatically as the sum of the element and support costs. Total cost may be manually overridden by entering a value in the **Total Cost** edit box. Manually adjusting total cost does not affect the current unit, element, or support costs. The adjusted total cost also does not carry forward to follow on sensors. In addition, total cost will be recalculated if new values are entered via the unit, element, or support cost edit boxes. The sum of the total cost of all sensors is displayed in the cost field on the left side of the MOE display, and represents the estimated cost of the entire sensor suite.

Development of ESS designs using SSES will typically involve multiple iterations of sensor selection and placement, and performance evaluation. The user can save the current sensor plan by left-clicking the **Save sensor plan** pushbutton. This opens a popup window where the sensor plan name may be entered. MOE data is not saved with the sensor plan in the SSES prototype version so it may be convenient to include metrics of interest (e.g. cost and Pd) in the plan name for future reference. Subsequently left-clicking on the **Recall sensor plan** pushbutton opens a pull down menu listing all saved sensor plans. Selecting a plan from the list will cause the plan to be reloaded, replacing the current plan. The current plan will be lost if it was not saved. Saved sensor plans are stored with the current SSES project, and will be available in future SSES sessions (provided the project is itself saved prior to exiting SSES.)

In the course of placing and orienting sensors, and examining resulting coverage and performance, it is often necessary to zoom, pan and rotate the display, as well as change display options. For convenience, the **Default View** pushbutton is provided to restore the default display options (top down, full site view) with a single mouse click.

## Display options

The SSES display consists of multiple data layers in several different formats. For most operations it is neither practical nor desirable to simultaneously display all layers. To allow users to tailor the display to the task at hand a Display Options control panel is provided. The panel is show in Figure 8.



Show intruder start point and defended point checkbox

Show obstacles checkbox

Show line sensor checkbox

Image selection pull down menu

Image transparency slider

Terrain network face display selection pull down menu

Terrain network edge display select pull down menu

Volume sensor coverage transparency slider

2D -3D Imagery toggle button

Set default view pushbutton

Terrain network face display transparency slider

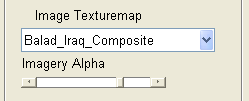
Terrain network vertex display select pull down menu

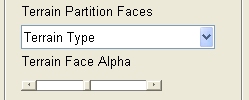
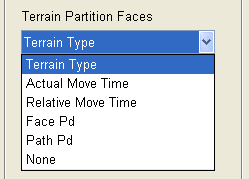
Terrain network edge display select pull down menu

Terrain network vertex display select pull down menu

Show volume sensor edge checkbox

Figure 8: Display Options control panel.

To guide the user in developing the terrain network model site imagery, typically overhead imagery generated by satellite or aircraft sensors may be displayed. The images are stored within SSES as a “texturemap” that can be wrapped over the site elevation grid, or plotted below the terrain network. The site images are loaded as part of site initialization. If more than one image is loaded the **Image Texturemap** pull down menu can be used to select the image to display. Once the terrain network has been generated it may be desirable to deemphasize the site image or hide it completely. The image can be deemphasized by adjusting image transparency using the **Imagery Alpha** slider bar. When the slider is positioned at the right-hand end of the bar the image will be fully opaque and will obscure any features at lower elevation. When the slider is positioned at the left hand of the bar the image is fully transparent and underlying graphics will be visible. Making the image fully transparent also suppresses rendering and may reduce the time required for graphic operations slightly. The **Imagery Alpha** slider control discussed in the next section can be used in conjunction with the Terrain Face Alpha slider control to balance the visual impact of site imagery and the network partition.

Terrain network face color can be used to encode several different types of data. The type of data shown is selected using the **Terrain Partition Faces** pull down menu. The default terrain face display is **Terrain Type** which colors the terrain faces to match the colors shown on the terrain editing control panel. This selection is typically used during construction of the terrain network. Figure 1 shows the **Terrain Type** face display.

Once the terrain network has been generated and development of the sensor plan has begun, assessing sensor coverage and detection probability becomes the primary focus. Sensor coverage can be shown in terms of either the “static” probability of detection provided by volume sensors within each terrain face, or the worst case (for the defender) cumulative probability of detection provided by all volume and line sensors against a threat traveling from the threat source to each terrain face. Figure 9 shows the Face Pd display for a sensor plan consisting of three closed circuit TV (green) and four passive IR (orange) sensors. The red faces indicate areas of low (in this case zero) Pd while the light areas around the sensor field of view show areas of high Pd.

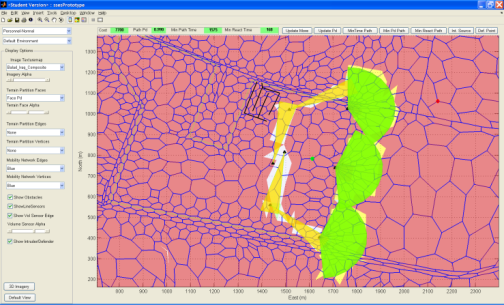
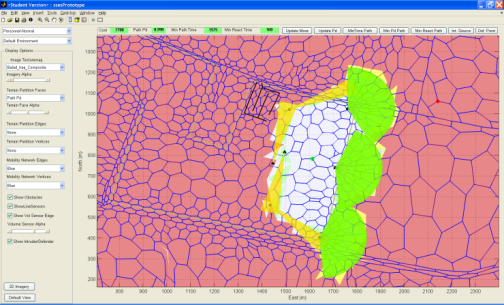


Figure 9: Face Pd display. Figure 10: Path Pd display.

Figure 10 shows the path Pd display. Faces within the sensor plan perimeter are now shown in light colors indicating a high cumulative Pd, even though the faces are not directly observable by any sensor. This occurs because to reach these areas the intruder must pass through coverage by one or more sensors. Removing any of the seven sensors in the plan would result in a Path Pd display that looked similar to Figure 9, since intruders would now have paths to reach these areas without passing through sensor coverage and being detected. In addition to Pd, face colors may encode the actual or relative time required for an intruder to travel from his starting point to that face may be shown.

Figure 11 shows the relative time needed to reach various parts of the site from the specified starting point. In this display the mobility edges have also been coded with the relative movement cost along each edge, and the route that provides the minimum reaction time for the defender is shown in magenta.

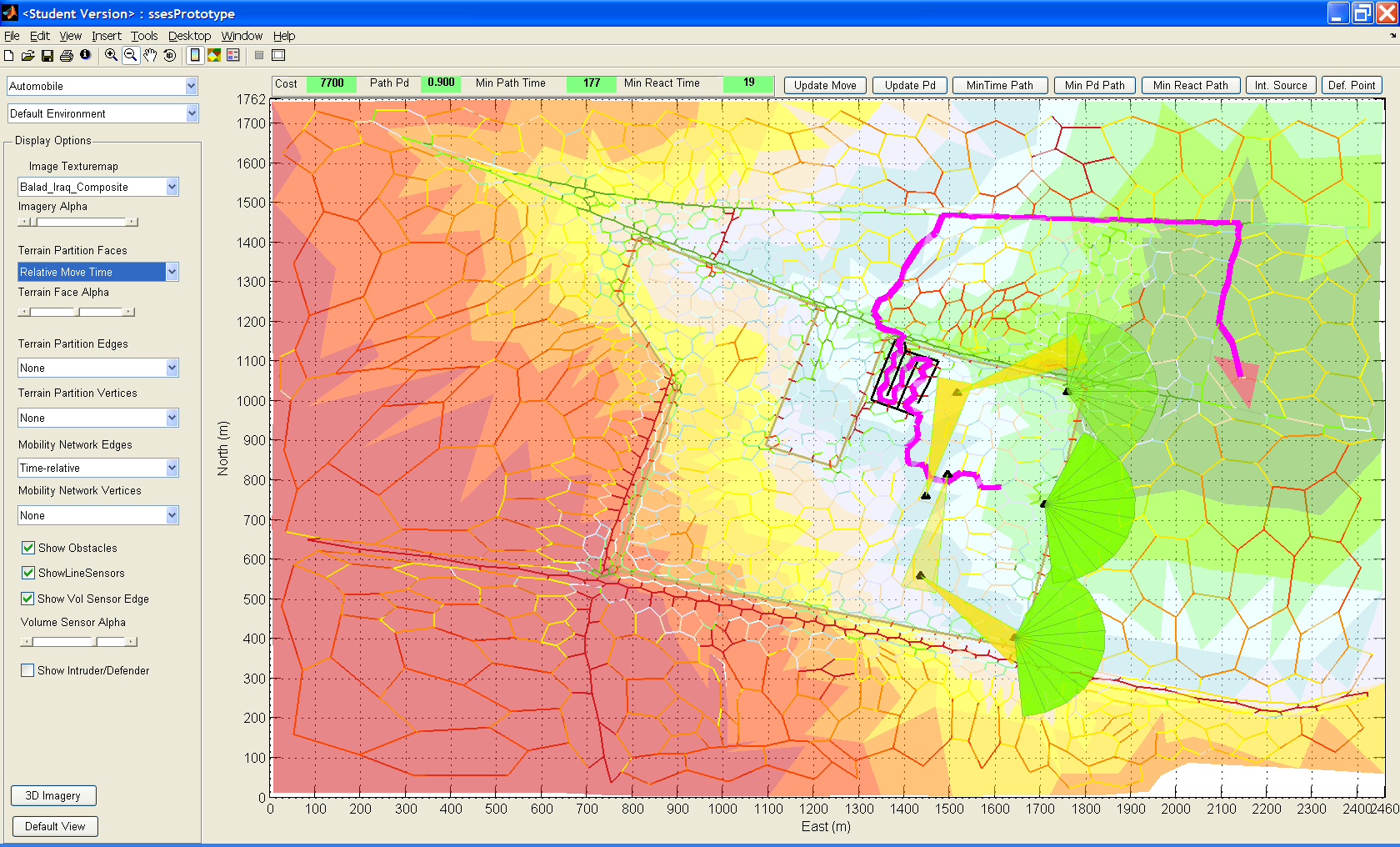
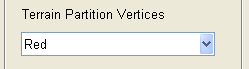
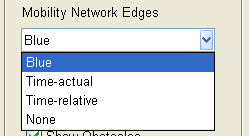
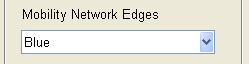


Figure 11: Relative movement time display.

Terrain partition edges are not used to encode information in the SSES prototype version. Their primary functions are to support terrain network construction, and to act as guides for positioning line sensors and obstacles. Terrain edges are displayed by selecting **Red** in the **Terrain Partition Edges** pull down menu and are hidden by selecting **None**.

Terrain network vertices are not used to encode information in the SSES prototype version. Their function is to provide the user with something to “grab” when adjusting the terrain network by dragging its vertices. Terrain vertices are displayed by selecting **Red** in the **Terrain Partition Vertex** pull down menu and are hidden by selecting **None**.

Mobility network edges may be used to encode the absolute or relative time required to traverse the edge. The type of data shown is selected using the **Mobility Network Edges** pull down menu. The default mobility network edge display is **Blue** which merely shows network connectivity, but provides good contrast with the terrain face encodings. The **Time-actual** option sorts travel times in seconds into bins with <10, <20, <30, <40, <50, <100, <200, <300, <400, <500, <1000 and > 1000 second limits and colors the edges from green (minimum time) to red (maximum time). The bin limits are not adjustable in the SSES prototype version. The **Time-relative** option operates in a similar manner, but instead of using fixed cutoff times, edges are sorted into 12 bins in order of increasing time and colored using the same palette used for **Time-actual**. The intent of the coding is to provide a graphic indication of attractive threat paths. Figure 11 provides an example. Display of mobility network edges can be suppressed by selecting the **None** option.

Mobility network vertices are only used to show the location of terrain network face centroids. The SSES prototype version bases volume sensor line-of-sight and detection probability calculations for each face using the face centroid location. When positioning volume sensors it may be convenient to de-clutter the display by turning terrain and mobility network edges off and selecting **Blue** in the **Mobility Network Vertices** pull down menu. Select **None** .to hide the vertices.

The **Show Obstacles** and **Show Line Sensors** check box allows the user to control the display of obstacles and line sensors respectively. Obstacles and line sensors are displayed when the applicable boxes are checked and hidden otherwise. Obstacle and line sensor display is independent of the associated terrain network edge display. However, terrain edges are needed to allow placement of obstacles and sensors. Figure 12 shows a site secured by reinforced chain link fence obstacle (inner tan perimeter) and buried seismic sensors (light green perimeter surrounding the fence. Vehicle traffic entering the site is restricted using a “serpentine” formed using jersey barriers (black ‘maze’.) Terrain and mobility network edges and vertices are suppressed in this example to improve readability.

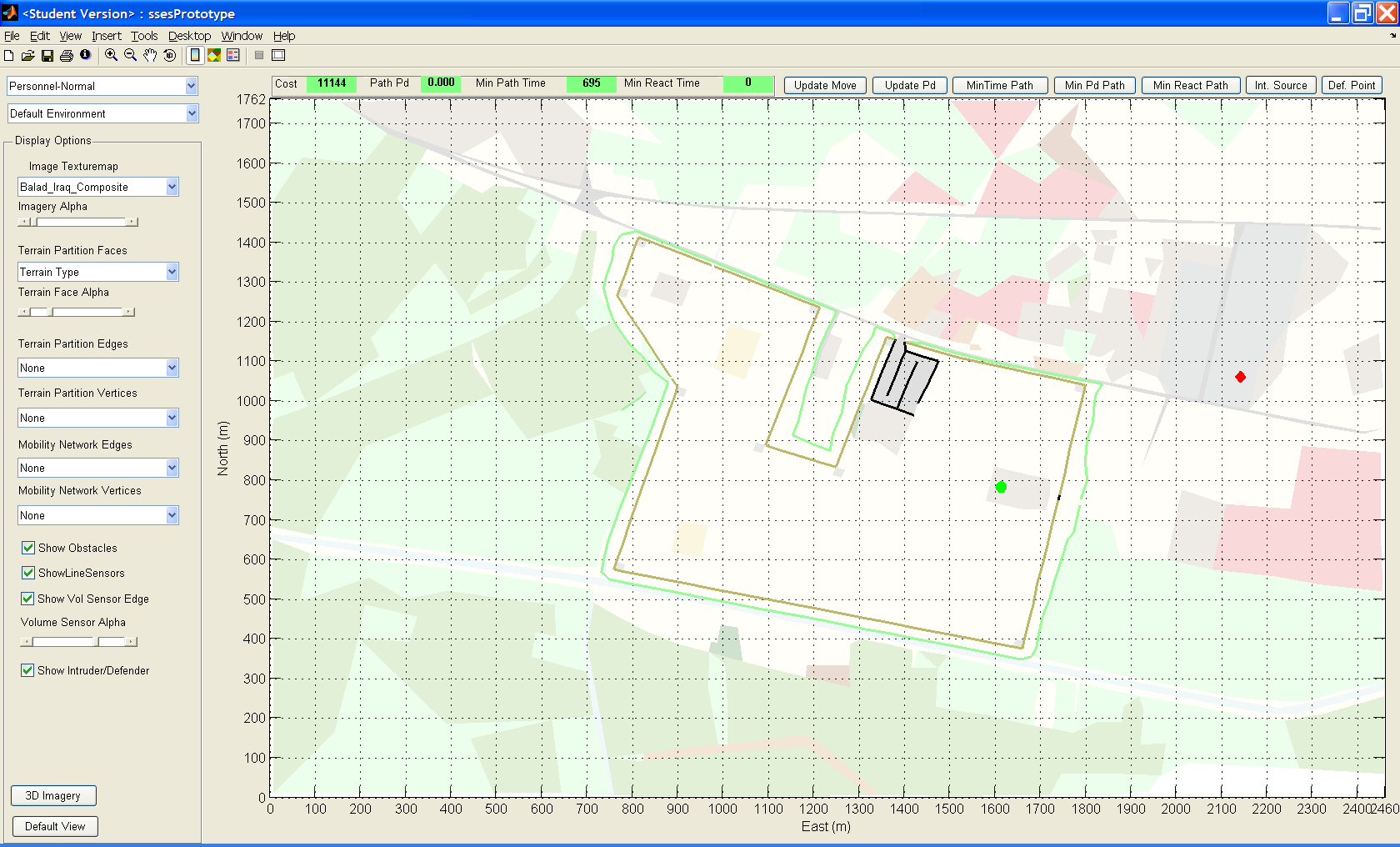


Figure 12: Site with obstacles and line sensors.

Separate controls are provided for the volume sensor position and coverage outline, and the coverage area fill. Checking **Show Vol Sensor Edge** checkbox displays the volume sensor position icon and outlines the coverage area. Sliding the **Volume Sensor Alpha** slider to the right end of the bar sets volume sensor fill to fully opaque. Moving the slider to the left end of the bar makes sensor coverage fill fully transparent, effectively hiding it.

Intruder path calculations require the operator to select an intruder start point (red diamond) and defended point (green dot). The Show Intruder/Defender check box allows the user to control display of these points.

SSES displays use a number of graphical representations to encode site, sensor plan, and performance information. To provide a ready reference, the **Show Legend** pushbutton generates a legend display in a separate window. Figure 13 shows a sample legend. The specific types, colors, and styles for terrain, obstacles, and sensors depend on the data that is loaded from the external databases.



Figure 13: SSES display legend.

As discussed above, terrain network faces will normally not fit the elevation grid data exactly. The goodness of fit will depend on the size of the terrain faces and the elevation grid spacing. While it is often useful to examine site topography in 3 dimensions, elevation mismatches will cause the terrain face display to intersect the site imagery and make it difficult to edit the terrain network accurately. To mitigate this problem the 2D/3D Imagery toggle button allows the user to switch the image between a 2 dimensional and 3 dimensional view. The default setting is 2 dimensional. In this mode the image is shown as a plane tangent to the lowest point in the elevation grid data. This ensures that terrain network elements are plotted on top of the image. When the display is in 2D mode the toggle button displays the **3D Imagery** label. Clicking on the button will switch to 3D mode and the image data will be wrapped onto the elevation data. This allows display of the site contour, but may cause portions of the terrain network to be obscured. When in 3D mode the toggle button label displays **2D Imagery**.

Operation of the **Default View** pushbutton is identical to operation in sensor editing mode. Clicking the button restores the default display options (top down, full site view) with a single mouse click.

## Sensor Plan Performance

The ultimate objective of the SSES is to assist the user to design effective sensor suites. The primary measures of effectiveness selected for the SSES prototype version are cost, the overall probability of detecting an intruder attempting to penetrate the site, and the expected reaction time available to the defender.

The SSES assists the user in assessing sensor suite performance by automatically calculating these metrics, and identifying and displaying potential worst case intruder ingress routes. These calculations take a noticeable amount of time to complete so it may not be desirable to update MOEs after each incremental change to the site representation or sensor plan. Therefore, Pd and path calculations are initiated by the user.

Intruder paths can be found to minimize either intruder travel time (“quickest” or “min time” path), minimize cumulative probability of being detected at some point along the path (“min Pd path”), or minimize the weighted average expected reaction time for the defender (“min reaction time path”).

Figure 14 shows the Measures of Effectiveness toolbar that is used to calculate and display the MOEs. A description of the function and use of each of these displays and controls is provided below. The algorithms used to determine Pd performance and intruder paths are discussed in the algorithms and data structures section of the manual.

Path probability of detection

ESS design total cost

Expected reaction time for worst case path

ESS design total cost

Update movement cost

Update probability of detection

Calculate minimum time path

Calculate minimum Pd path

Calculate minimum reaction time path

Select Intruder starting point

Select defended point



**Figure 14: Measures of Effectiveness toolbar.**

**ESS total cost** is calculated automatically as sensors are added to and deleted from the plan. Cost should always be current and no manual action is required to update this field.

**Path Pd** indicates the cumulative probability of intruder detection as the intruder travels along the most recent optimum (from the intruder’s point of view) path. **Path Pd** is updated when the **Min Time Path**, **Min Pd Path** or **Min Reaction Time Path** are calculated.

**Path Time** indicates the time (in seconds) required for the intruder to travel from the source to the defended point along the most recent optimum (from the intruder’s point of view) path. **Minimum Path Time** is updated when the **Min Time Path**, **Min Pd Path** or **Min Reaction Time Path** are calculated.

**Path Reaction Time** indicates the expected value of the reaction time (in seconds) available to the defender to respond measured from intruder detection to intruder arrival at the defended point along the most recent optimum (from the intruder’s point of view) path. **Path Reaction Time** is updated when the **Min Time Path**, **Min Pd Path** or **Min Reaction Time Path** are calculated.

The **Update Move** pushbutton is used to manually update movement costs following changes to site model (e.g. changing terrain types, adjusting vertex locations, or adding/removing obstacles), or when the threat type is changed. Movement costs are recalculated automatically if required when the minimum time path, minimum Pd path, or minimum reaction time paths are calculated. The pushbutton face color changes to yellow while movement costs are being computed to indicate that calculations are in progress. Update move may be used independently to cause the **Terrain Partition Face** and **Mobility Network Edge** displays to be updated with the current movement costs.

The **Update Pd** pushbutton is used to manually update detection probabilities following changes to the sensor plan or when the threat type is changed. Detection probabilities are recalculated automatically if required when the **minimum time path**, **minimum Pd path**, or **minimum reaction time path** is calculated. The pushbutton face color changes to yellow while movement costs are being computed to indicate that calculations are in progress. **Update Pd** may be used to cause the **Terrain Partition Face** **Face Pd** display to be updated with the current detection probability. It does not cause **Path Pd** values or display to be updated. To update the **Path Pd** display the minimum Pd path or minimum reaction time path functions must be executed.

The **Min Time Path** pushbutton is used to calculate and display the minimum time (quickest), path from the intruder start point to the defended point. The minimum time path is calculated using Dijkstra’s shortest path algorithm where the network edge costs are based on travel time, including terrain and obstacle effects. The minimum time path is shown as a black overlay on the mobility network if the mobility network option is any selection other than **None**. Probability of detection and reaction time are not considered when selecting the minimum time path, but the cumulative probability of detection and expected reaction time for the path are calculated and displayed. The Path Time display window is highlighted in green to indicate that the **Path Time**, **Path Pd**, and **Path React Time** MOE displays reflect the minimum time path. If required, mobility network costs and detection probabilities will be recalculated automatically.

The **Min Pd Path** pushbutton is used to calculate and display the minimum path from the intruder start point to the defended point that provides the minimum cumulative probability of detection. The minimum time path is calculated using a modified version of Dijkstra’s shortest path algorithm. This algorithm is described in the following section. The minimum Pd path is shown as a yellow overlay on the mobility network if the mobility network option is any selection other than **None**. The time required to traverse the minimum Pd path and the expected reaction time path are calculated and displayed. The Path Pd display window is now highlighted in green to indicate that the MOE displays reflect the minimum time path. As for the minimum time path, mobility network costs and detection probabilities will be recalculated automatically if necessary.

The **Min React Path** pushbutton is used to calculate and display the path from the intruder start point to the defended point that provides the minimum expected reaction time for the defender. This path is also calculated using a modified version of Dijkstra’s shortest path, but the path is now chosen to minimize the weighted sum of detection probabilities at each node multiplied by the time for the intruder to reach the defended point along the shortest path from that node. The algorithm is described in the following section. The minimum reaction time path is shown as a green overlay on the mobility network if the mobility network option is any selection other than **None**. The cumulative path Pd and time required to traverse the path are also calculated and displayed. In this case the Path React Time display is highlighted in green to indicate that the MOE displays reflect the minimum reaction time path. As for the other path calculations mobility network costs and detection probabilities are recalculated automatically if necessary.

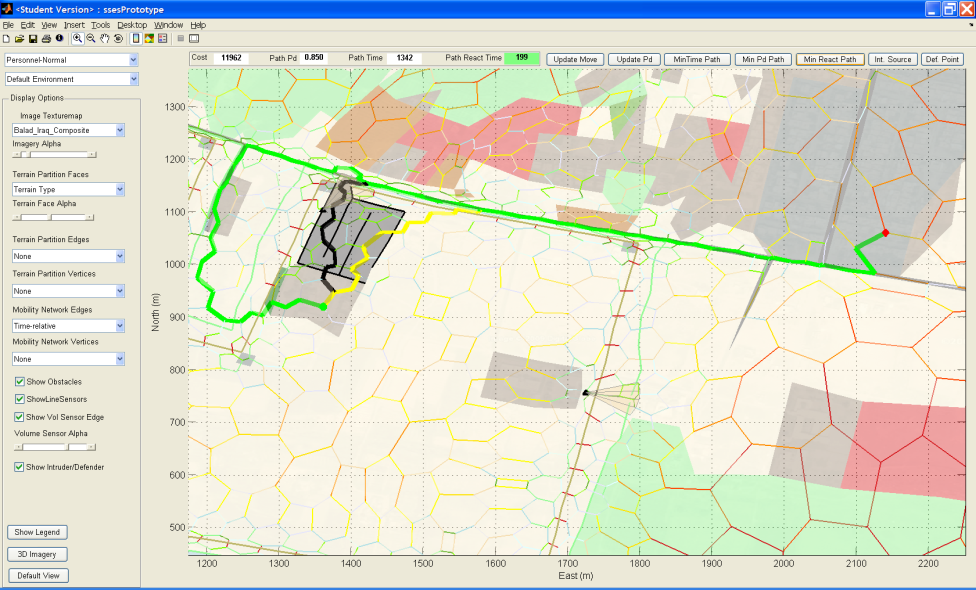


Figure 15: Minimum path comparison.

Figure 15 illustrates the results of the three path algorithms. The site shown is completely surrounded by reinforced fence and a buried line seismic sensor with a Pd of 0.85, except at the main entrance on the north side of the site and a pedestrian gate on the east side. These entrances are monitored by active IR sensors which observe multiple terrain faces resulting in a cumulative Pd of ~1.0. The minimum time path (highlighted in black), which does not consider Pd, routes the intruder directly through the main entrance. This is the quickest route (862 seconds) but results in certain detection Pd = 1.0) with an expected reaction time of 235 sec. The minimum Pd path (highlighted in yellow), avoids the main entrance, reducing Pd to 0.85. The minimum Pd algorithm attempts to minimize travel time as a secondary objective and routes the intruder across the interior of the camp, resulting in a total path time of 891 seconds and expected reaction time of 534 seconds. The minimum reaction time route (highlighted in green) achieves the same Pd (0.85) as the minimum Pd algorithm, but by penetrating the camp at a point near the objective reduces expected reaction time to 199 seconds, at the expense of increasing total route time to 1342 seconds.

Each of the path algorithms discussed above calculates a path from a specified starting point (the intruder source) to a specified objective (the defended point.) These points are selected using the **Int Source** and **Def Point** pushbuttons. To select a point left-clicking on the applicable button then left-click the desired terrain face. The source and defended point are placed at the face centroid. The intruder source is displayed as a red diamond and the defended point is displayed as a green dot.

Intruder movement times depend on intruder type as well as terrain and obstacles. In future SSES versions intruder type will also affect sensor performance. Once an intruder database has been loaded, the user may select an intruder type using the intruder selection pull down menu. The Environment selection menu is not enabled in the SSES prototype version.

# SSES data structures and algorithms

SSES data is maintained as a set of data structures that are saved in the SSES GUI figure (“appdata”). These structures contain data describing the site, terrain and mobility networks, obstacles, sensor plan, and “housekeeping” data that supports GUI operation. Experienced MATLAB users may find it useful to examine or manipulate these data structures directly in order to perform operations that are not currently supported by the SSES prototype version (e.g. find and delete all sensors of a particular type, manually edit site elevation data, etc.)

## Site data

The “site” data structure defines the site coordinate system and contains site imagery and elevation data and is shown in Figure 16. The data can be accessed using the commands:

site = getappdata(gcf,'siteData')

setappdata(gcf,'siteData',site)

## Terrain data

The “ter” data structure contains the data which form the terrain partition network including the location of the vertices, vertex to edge and vertex to face adjacency, and the terrain type associated with each terrain face. It also contains temporary data used to construct and adjust the terrain network. The ter.terrainList field contains the list of available terrain types. The terrain data can be accessed using the commands:

ter = getappdata(gcf,'terrainData')

setappdata(gcf,'terrainData',ter)

site.label = ''; % Site label

site.xGrid = [0 100]; % X axis tick cooridnates

site.xUnits = 'meters'; % X units

site.xMesh = [0 100;0 100]; % X coordinate mesh grid data

site.xLimit = [0 100]; % X axis limits

site.xLabel = []; % X axis display label

site.yGrid = [0 100]; % X axis tick cooridnates

site.yUnits = 'meters'; % Y units

site.yMesh = [0 0;100 100]; % Y coordinate mesh grid data

site.yLimit = [0 100]; % Y axis limits

site.yLabel = []; % Y axis display label

site.zUnits = 'meters'; % Z units

site.zElev = [0 0;0 0]; % Elevation mesh data

site.zLimit = [0 10]; % X axis limits

site.zLabel = []; % Z axis display label

site.xGridLines = []; % X axis grid lines

site.yGridLines = []; % Y axis grid lines

site.zGridLines = []; % Z axis grid lines

site.textureMaps = []; % Image structure of indexed images with associated colormap

site.startTM = []; % Label of image to display at startup

Figure 16. Site data structure.

ter.hMasterFigure = handles.MasterFigure; % Handle of master figure

ter.vertexXYZ = []; % Terrain mesh vertex position

ter.vertex = struct(); % Terrain mesh vertex amplifying data

ter.nVertex = 0; % Current number of terrain mesh vertices

ter.deletedVertices = []; % List of deleted vertices for reuse

ter.adj = sparse([]); % Terrain mesh adjacency matrix

ter.edge2vert = zeros(0,2); % List of vertices connected by each edge

ter.edge2face = zeros(0,2); % List of tiles adjacent to each edge

ter.edge = struct(); % Terrain mesh edge amplifying data

ter.nEdge = 0; % Current number of terrain mesh edges

ter.faceTerIndex = []; % Terrain mesh terrain type index

ter.face = terrainList(1); % Terrain mesh face amplifying data

ter.facet2vert = {}; % List of vertices that form each facet

ter.face2vert = []; % nFace x 3 array of vertices that form each face

ter.faceCenterXYZ = []; % Coordinates of face centroid

ter.nFace = 0; % Current number of terrain facets

ter.deletedFacets = []; % List of deleted facets for reuse

ter.altOffset = 0.1; % Offset between actual DEM and terrain facet alt

ter.terrainList = terrainList; % Structure of terrain types

ter.currTerrainIndex = 1; % Index of currently selected terrain type

ter.hTerSel = []; % Handle to terrain select pushbutton

ter.meshEditMode = 0; % Terrain mesh editing mode

ter.currVertex = []; % Currently active terrain mesh vertex

ter.currEdge = []; % Currently active terrain mesh edge

ter.currTile = []; % Currently active terrain mesh facet

ter.newVertexXYZ = []; % Position of prospective new vertices

ter.hNewVertex = []; % Graphic handles to new vertices

ter.selectedEdge = []; % Currently selected edge

ter.hSelectedEdge = []; % Graphic handle to selected edge

ter.dragVertexSel = []; % Current vertex for dragging

ter.dragVertexRestoreXYZ = []; % Saved location of vertex prior to dragging

Figure 17. Terrain data structure.

## Mobility data

The “mob” data structure contains the distance and travel time data for the mobility network. The structure of the mobility network is determined by the terrain network. Mobility data can be accessed using the commands:

mob = getappdata(gcf,'mobilityData')

setappdata(gcf,'mobilityData',mob)

mob.nEdge = []; % Number of mobility map edges

mob.edge2face = []; % list of edge to face adjacency

mob.eDist = sparse([]); % Mobility map Euclidean distance

mob.edgeCost = []; % Mobility map edge cost

mob.adjCost = sparse([]); % Threat specific mobility edge cost (adjacency matrix)

Figure 18. Mobility data structure.

## Obstacle data

The “obs” data structure contains the list of available obstacles types and the obstacles that have been placed on each mobility network edge. Face obstacles are not currently used. Obstacle data can be accessed using the commands:

obs = getappdata(gcf,'obstacleData')

setappdata(gcf,'obstacleData',obs)

obs.hMasterFigure = handles.MasterFigure; % Handle of master figure

obs.obstacleList = []; % List of available obstacle types

obs.edgeObstacles = {}; % Cell array of emplaced edge obstacles

obs.faceObstacles = {}; % Cell array of emplaced face obstacles -- not currently used

Figure 19. Obstacle data structure.

## Sensor data

The “ess” data structure shown in Figure 20 contains the data used to represent and assess the electronic security system design. The ess.sensorList field contains the list of available sensor types and is shown in Figure 21. The ess.lineSensors and ess.volumeSensors fields specify the sensors used in the current plan. The ess.savedPlanData field contains a structure array that specifies the type and locations of all sensors in a plan. Sensor data can be accessed using the commands:

ess = getappdata(gcf,'mobilityData')

setappdata(gcf,'sensorData',ess)

ess.hMasterFigure = handles.MasterFigure; % Handle of master figure

ess.sensorList = sensorList; % List of available sensor types

ess.currentSensor = 0; % Pointer to current sensor

ess.currentSensorParam = sensorList;% Current sensor template

ess.lineSensors = {}; % Cell array of emplaced line (edge) sensors

ess.volumeSensors = {}; % Cell array of emplaced volume (face) sensors

ess.EdgePd = []; % Probability of threat detection when crossing edge

ess.FacePd = []; % Probability of threat detection when transiting face

ess.editMode = 0; % Sensor editing mode 0=none, 1=add, 2=delete, 3=edit

ess.cumCost = 0; % Cumulative cost of sensors and support gear

ess.sensorCounter = 0; % Counter of sensors assigned

ess.currPdValid = 0; % Flag indicating whether current Pd calculation is valid

ess.savedPlanData = {}; % Data for saved ESS plans

ess.savedPlanList = {}; % List of labels for saved ESS plans

Figure 20. Sensor plan data structure.

sensorList.model = ''; % Sensor model

sensorList.class = []; % Sensor class {'line' or 'volume'}

sensorList.type = []; % Sensor type

sensorList.sensorLabel = []; % Discrete sensor label

sensorList.emission = []; % Sensor emission mechanism {'active' or 'passive'}

sensorList.PdMode = []; % Sensor Pd calculation mode -- only constant Pd is implemented

sensorList.Pd\_const = []; % Constant Pd value

sensorList.Pd\_lookup = []; % Pd lookup table value -- not currently implemented

sensorList.Pd\_model = []; % Pd calculation function handle

sensorList.effRange = []; % Sensor maximum effective range

sensorList.Xlocation = []; % Sensor X

sensorList.Ylocation = []; % Sensor Y

sensorList.Zlocation = []; % Sensor Z location

sensorList.mountHeight = 0; % Sensor mounting height -- zero for line sensors

sensorList.AZlimit = []; % Volume sensor azimuth left and right

sensorList.ELlimit = []; % Volume sensor elevation lower and upper

sensorList.AZmax = []; % Volume sensor maximum allowed azimuth extent

sensorList.ELmax = []; % Volume sensor maximum allowed elevation extent

sensorList.unitCost = []; % Sensor unit cost

sensorList.elementCost = []; % Sensor element cost

sensorList.supportCost = []; % Sensor support cost -- manually input

sensorList.totalCost = []; % Sensor total cost

sensorList.edgeRGB = [0 0 0]; % Specifies sensor display color

sensorList.edgeWidth = 2; % Specifies sensor display line width

sensorList.edgeStyle = '-'; % Specifies sensor display line style

sensorList.faceRGB = [1 1 1]; % Volume sensor face color -- not used for line sensors

sensorList.faceAlpha = 0.7; % Volume sensor face density -- not used for line sensors

Figure 21. Sensor element structure.

## Display data

The “dsp” data structure shown in Figure 22 contains various data used to manipulate the GUI display. Fields beginning with “h” or “mob\_h” contain handles to the displayed objects and can be used to manipulate the plot directly using MATLAB handle graphic commands, (e.g. entering: set(dsp.hVertex,’FaceColor’,’w’) would change the color of all the terrain map vertices white). The “dsp” object also stores the results and predecessor map of the current “best path” calculation. Sensor data can be accessed using the commands:

dsp = getappdata(gcf,'displayData')

setappdata(gcf,'displayData',ess)

dsp.hMasterFigure = handles.MasterFigure;

dsp.hVertex = []; % Terrain mesh vertex graphic handles

dsp.hEdge = []; % Terrain mesh edge graphic handles

dsp.hFace = []; % Terrain mesh facet graphic handles

dsp.curTerrainHue = ter.terrainList(1).RGB; % Plot color of current terrain type

dsp.mob\_hVertex = []; % List of handles to mobility map vertices / face centers

dsp.mob\_hEdge = []; % List of handles to mobility map edges

dsp.curTerrainAlpha = 0.5; % Terrain map transparency

dsp.curMobNetAlpha = 1; % Mobility network transparency value

dsp.curTextureMap = []; % Handle to current texture map

dsp.curImageMapAlpha = 1; % Image map transparency

dsp.g2rRGB = g2rRGB; % Utility display colormap

dsp.hEdgeObstacles = []; % List of handles to edge obstacles

dsp.hFaceObstacles = []; % List of handles to face obstacles -- not currently implemented

dsp.nLineSensors = 0; % Number of line sensors emplaced -- not currently used

dsp.nVolumeSensors = 0; % Number of line sensors emplaced -- not currently used

dsp.hLineSensors = []; % List of handles to line sensors

dsp.hVolSensorPos = []; % List of handles to volume sensor location markers

dsp.hVolSensorCoverage = []; % List of handles to volume sensor coverage patches

dsp.defendedPoint = []; % Face index of defended point (intruder goal)

dsp.hDefendedPoint = []; % Handle to defended point marker

dsp.intruderSource = []; % Face index of intruder starting point

dsp.hIntruderSource = []; % Handle to intruder starting point marker

dsp.spMoveTime = []; % Intruder quickest path node data (min movement time)

dsp.spPred = []; % Intruder quickest path predecessor tree data

dsp.minPd = []; % Intruder minimum Pd path node data (probability of detection)

dsp.minPdPred = []; % Intruder minimum Pd path predecessor tree data

dsp.minPdRouteTime = []; % Minimum Pd route final transit time to goal

dsp.minRxTpd = []; % Intruder minimum reaction time path node data

dsp.minRxTwrt = []; % Intruder minimum reaction time path node data (weighted

dsp.minRxTpred = []; % Intruder minimum reaction time path predecessor tree data

dsp.minRxT = []; % Minimum reaction time path final expected reaction time

dsp.hDisplyOptionsPB = []; % Handle of (hijacked) display options pushbutton control

dsp.hEditTerrainPB = []; % Handle of (hijacked) terrain editing pushbutton control

dsp.hEditSensorsPB = []; % Handle of (hijacked) sensor editing pushbutton control

Figure 22. Sensor plan data structure.

## Detection calculation

The SSES prototype version represents Pd as a constant value for a given sensor type. The application includes “hooks” to install threat, environment, and terrain specific detection models in the future. Detection probabilities are calculated as the intruder moves from vertex to vertex along the mobility network. Detection is assumed to occur on arrival at the next vertex. During each transition the threat may be subject to detection by one line sensor and multiple volume sensors. The Pd due to line sensors is stored in the **ess.EdgePd** field which contains a Pd value for each mobility/terrain network edge. The total Pd due to all volume sensors that observe a given terrain face / mobility network vertex is stored in the **ess.FacePd** field. Face Pd is calculated as:

where Pi is the probability of detection by the ith volume sensor that observes the terrain face. Volume sensor detection probabilities are only applied if a clear line-of-sight exists between the sensor and the target. The cumulative detection probability along a path is calculated as where Pf are the detection probabilities of the faces/vertices visited and Pe are the detection probabilitis of the edges traversed.

## Line-of-sight calculation

Sensor to target line-of-sight is determined by checking the line-of sight-against the site terrain network rather than the underlying elevation grid data. The line of site is blocked if the elevation of a terrain edge that is intersected by the line of sight is higher at the intersection point than the elevation of the line of sight vector at that point. Edges are checked by starting at the sensor and walking along the terrain network from face-to-edge-to-face until the target is reached or the line-of-sight is obstructed. This approach minimizes the number of terrain edges that need to be checked. The algorithm is shown in Figure 23.

Figure 23. Line-of-sight algorithm.

## Intruder path calculations

The minimum time, minimum Pd and minimum reaction time paths are calculated using Dijkstra’s shortest path algorithm or a modified version of it. Figure 24 shows a version of the shortest path algorithm derived from Figure 4.6 of Ahuja, Magnanti, and Orlin, 1993.

For the modified algorithm the minimization term is modified to allow the “distance” of the new node to be an arbitrary non-decreasing function of the head node and the “distance” of its predecessor.

For the minimum time path, the distance function is:

and the result is Dijkstra’s algorithm expressed in terms of travel time.

For the minimum Pd path, the distance function is:

where *P(j)* is the minimum cumulative probability of being detected on the path to node *j* and *Pij* is the probability of being detected while moving from node *j* to node *i*.

For the minimum reaction time path the distance function is:

where and are calculated as for the minimum Pd path, is the incremental probability of being detected while traveling from *j* to *i*, and is the time required to reach the target along the quickest path from node *i*. The result is expected of the time for the intruder to reach his objective after detection, which is the amount of time available for the defender to respond to the intrusion.

Figure 24. Generic shortest path algorithm.

# Reference

Ahuja,R. K., Magnanti, T. L., and Orlin, J. B. 1993. Network Flows, Theory, Algorithms, and Applications. Prentice Hall, Upper Saddle River New Jersey