ARMY RESEARCH LABORATORY

Building and Vegetation Rasterization for the Three-dimensional Wind Field (3DWF) Model

by Giap Huynh, Yansen Wang, and Chatt Williamson

ARL-TR-5419 **December 2010**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Adelphi, MD 20783-1197

ARL-TR-5419 **December 2010**

Building and Vegetation Rasterization for the Three-dimensional Wind Field (3DWF) Model

Giap Huynh, Yansen Wang, and Chatt Williamson
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
December 2010	Final	

4. TITLE AND SUBTITLE
Building and Vegetation Rasterization for the Three-dimensional Wind Field (3DWF) Model

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Giap Huynh, Yansen Wang, and Chatt Williamson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-CIE-D
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-5419

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Land surface morphology information, such as vegetation and buildings, are key inputs for the parameterizations used in the Three-dimensional Wind Field (3DWF). The microscale land morphology data are often not available and/or are not in a format that 3DWF can readily use. We have developed a novel method to create the land morphology data set and rasterize it to the model grid using Google Maps™ and the accompanying application programming interface (API). This report documents the algorithm, the development of a graphical user interface (GUI) prototype, the process of rasterizing morphological features for a given model domain and terrain map, and step by step procedures for producing necessary input data for the 3DWF model.

15. SUBJECT TERMS
Rasterization, building, vegetation, 3DWF, GUI

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Giap Huynh
Unclassified	Unclassified	Unclassified	UU	38	19b. TELEPHONE NUMBER (*Include area code*) (301) 394-1156

Contents

List of Figures

Acknowledgments

We sincerely thank Dr. Dennis Garvey for his timely and extensively review of this report to catch any errors and unclear issues. His effort is greatly appreciated. We also extend our appreciation to Dr. Donald Hoock for his general guidance and support.

INTENTIONALLY LEFT BLANK.

1. Introduction

The Three-dimensional Wind Field (3DWF) model is a microscale mass consistent diagnostic model (Wang et al., 2005, 2010). Land surface morphology information, such as vegetation and buildings, are key inputs for the parameterizations used in 3DWF. Microscale land morphology data are often not available and/or are not in a format that 3DWF can readily use. We have developed a novel method to create the land morphology data set and rasterize it to the model grid using Google Maps™ and the accompanying application programming interface (API).

Google Maps™ provides users a convenient overview of any region in the world using relatively recent satellite imagery. This imagery is also geo-referenced, providing relatively accurate latitude and longitude information. Leveraging these features and the Google Maps™ API, a simple application can be developed that provides a user with the capability to geo-locate a building or vegetative canopy within a particular 3DWF model domain. However, there is still a need to rasterize these objects (buildings and vegetative canopies) and combine them with the terrain elevation information before the whole domain can be used by the 3DWF model for an assessment of the wind flow over complex or urban terrain.

This report documents the process of rasterizing morphological features such as buildings and vegetative canopies for a given model domain and terrain map, and a procedure to produce morphology data and maps necessary for 3DWF. The main rasterization algorithm was first prototyped in MatLab™ and then written in Fortran90 for quicker execution. The prototyped algorithm in MatLab has proven useful for cross checking and testing purposes. Also a Google Maps-based JavaScript graphical user interface (GUI) program was designed and implemented to generate the data file necessary for the main rasterization program.

2. Rasterization Algorithm

We employ one of the most popular rasterizing approaches (Sheffer, 2005) based on line sweeping of a polygon decomposed into triangles. The advantage of this process has been widely documented, as most current graphics systems support hardware rasterization of triangles and display them efficiently. We first address the rasterization of a triangle and then the decomposition of a polygon into triangles.

2.1 Triangle Rasterization

The approach we take for rasterizing a triangle starts with the concept of an edge equation, which is characterized by the general form of a linear equation $Ax+By+C=0$. Two vertices of a triangle define a unique line in the coordinate plane. This line divides the coordinate plane into two half

planes, a positive or upper half-plane (Ax+By+C>0) and a negative or lower half-plane (Ax+By+C<0), as illustrated in figure 1. Now, for each of the three sides of a triangle, we can define an edge equation. Further, we can define the interior of the triangle as the intersection of the appropriate half-planes. By appropriate we mean given an ordering of the vertices, the interior of the triangle is given by the intersection of the positive or negative half-planes defined by the triangle edges. For example, if the vertices are ordered clockwise, then the interior of the triangle is given by the intersection of the positive half-planes.

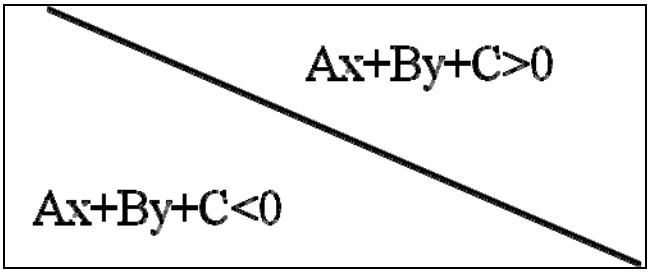


Figure 1. Illustration of the edge equation.

Next, we need to determine the coefficients (A, B, and C) for each of the edge equations. To determine these coefficients, we first consider two vertices of a given triangle $v(x_1, y_1)$ and $v(x_2, y_2)$. Since these vertices lie on the same line we have the following:

$$Ax_1 + By_1 + C = 0 \quad (1)$$
$$Ax_2 + By_2 + C = 0, \quad (2)$$

which when rewritten using matrix algebra takes the form

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = -C \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \quad (3)$$

Solving this system given

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}^{-1} = \frac{1}{x_1 y_2 - x_2 y_1} \begin{bmatrix} y_2 & -y_1 \\ -x_2 & x_1 \end{bmatrix}, \quad (4)$$

we have

$$\begin{bmatrix} A \\ B \end{bmatrix} = \frac{-C}{x_1 y_2 - x_2 y_1} \begin{bmatrix} y_2 - y_1 \\ x_1 - x_2 \end{bmatrix}. \quad (5)$$

Next for convenience set $= x_2 y_1 - x_1 y_2$, which then gives us

$$A = y_2 - y_1 \quad (6)$$

$$B = x_1 - x_2. \quad (7)$$

It should be noted that vertex order was not considered in the previous derivation. While ordering is not essential for the general derivation of the coefficients, it is important in the rasterization algorithm as discussed earlier.

Now that we have the general form of the coefficients we can define the edge equations as follows

$$E_1 = A_1x + B_1y + C_1 \quad (8)$$
$$E_2 = A_2x + B_2y + C_2 \quad (9)$$
$$E_3 = A_3x + B_3y + C_3. \quad (10)$$

Here for clockwise ordering of the triangle's vertices the coefficients are

$$\begin{array}{lll} A_1 = y_2 - y_1 & B_1 = x_1 - x_2 & C_1 = x_2y_1 - x_1y_2 \\ A_2 = y_3 - y_2 & B_2 = x_2 - x_3 & C_2 = x_3y_2 - x_2y_3. \\ A_3 = y_1 - y_3 & B_3 = x_3 - x_1 & C_3 = x_1y_3 - x_3y_1 \end{array} \quad (11)$$

Finally, since for clockwise ordering the interior of a triangle is defined as the intersection of the positive half-planes, we have $E_1, E_2, E_3 > 0$ when a grid point is in the interior of the triangle.

One last consideration is associated with optimization. We would like to limit the line sweep to only those grid points that are near the triangle. This is easily done by defining a bounding box with coordinate pairs (x_{min},y_{min}) and (x_{max},y_{max}), as illustrated in figure 2. This limits the algorithm's test space to the set$\{x, y: x_{min} \le x \le x_{max}, y_{min} \le y \le y_{max}\}$. Note that the graphics coordinate system is defined such that *x* increases left to right and *y* increases top to bottom. For such a system, *z* increases into the page and "clockwise" appears "counterclockwise" when viewed from above the page.

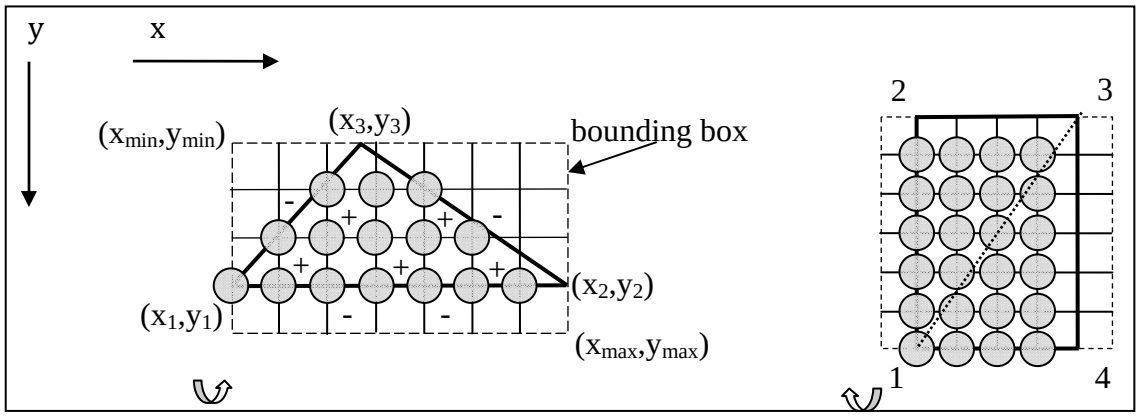


Figure 2. Triangle rasterization from implicit edge equations (left) and of a rectangle 1234 (right) with left and bottom edges included.

2.2 Triangular Decomposition of Polygons

A key geometric property of a triangle is that it is a convex polygon. This fact about triangles is what is exploited by the rasterization algorithm. Another nice geometric property is that all

polygons of higher order can be decomposed into the union of a set of triangles such that the intersection of any two triangles of the set is either the empty set, a singleton (i.e., common vertex), or an edge. These two facts imply that both convex and concave polygons can be rasterized given a proper triangulation. Figure 3 illustrates various types of convex and concave polygons.

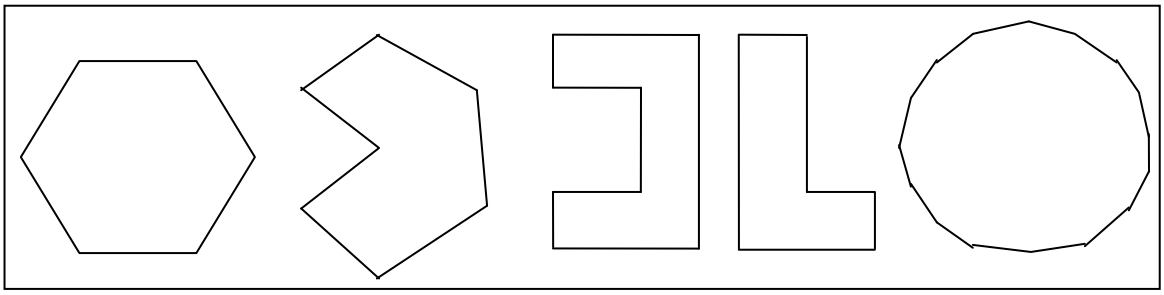


Figure 3. Types of polygons from left to right: convex, concave, "U-shaped," "L-shaped," and a 16-sided estimate of a circle.

For convex polygons, the decomposition is trivial. A simple algorithm for triangulation is to pick a vertex of the polygon as a starting point, call this point P_1 (figure 4), then traverse the polygon clockwise to P_2 and then P_3 to define the first triangle $P_1P_2P_3$. Next, take the starting point P_1 and the last vertex of the previous triangle P_3 and advance clockwise to the fourth point of the polygon P_4. This would then define the second triangle $P_1P_3P_4$. Repeat this process with the next point in the polygon until the polygon has been completely traversed. A second algorithm, which is just as simple, involves adding a point that is in the interior of the polygon and then traversing the vertices of the polygon to create the triangulation using the interior point as a common vertex. For our purposes, we implemented the first of these two algorithms.

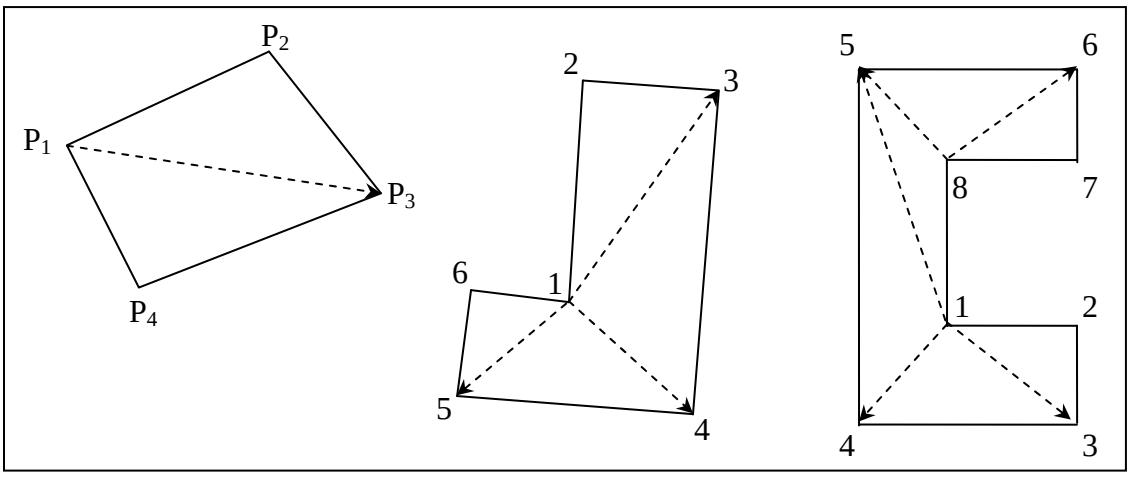


Figure 4. Triangle decomposition process.

Triangulation of concave polygons is not so trivial. Perhaps the easiest case is a concave quadrilateral, where the first algorithm works all the time but the second algorithm is a little bit tricky. The issue with the second algorithm is finding a suitable interior point. The second algorithm is also less efficient since it would result in four triangles rather than two. In general, the interior point algorithm will always have two more triangles in its set than the edge algorithm.

Since triangulation of a concave polygon is a little bit more complicated, we limit our cases to two different shapes. The first shape we consider is the "L" shaped polygon. The triangulation for this polygonal shape is trivial if one chooses the right vertex as the starting point of the traversal (see figure 4). If we select the point where the polygon is concave then we can traverse the polygon just as we did with the first algorithm for convex polygons. This is the method that we have implemented. The second shape that we consider is the "U" or "C" shaped polygon. Once again, if we order the vertices starting from one of the vertices where it is concave, we can traverse the polygon to produce the triangulation. However, eventually the concavity of the polygon will spoil our triangulation. This is easily fixed by moving to the next vertex that is part of the concave shape (i.e., in figure 4, we move to vertex 8). From here, we complete the triangulation. A second method for the "U" shaped polygons is to first decompose the polygon into a set of quadrilaterals that can then be trivially decomposed into triangles. We implemented the first method of triangulation for "U" shaped polygons.

Finally, while it seems that we are fairly restrictive in our use case for defining polygons, this is not the case. While it may take defining a few more polygons, almost any polygon can be decomposed into the union of a set of triangles and quadrilaterals.

In light of the selections of the methods of triangulation described previously, it is imperative that the user of this tool follow the conventions discussed in this section when defining polygons to be rasterized. First and foremost, one must always traverse and/or define the vertices of the polygon using clockwise motion. Second, for concave polygons, one must always start the polygon at a vertex that is part of the concave section of the boundary.

3. Implementation

Through the initial design process of the building and vegetation tool, we decided to implement two separate modules, one module for the GUI for the preparation and production of the geographical information using Google Maps and a second module for the implementation of the line sweep with implicit edge equations.

In this section, we discuss the implementation of the two modules. We start by discussing the implementation of the Google Maps-based GUI and some simple use cases. This is followed by a discussion of the implementation of the algorithm using Fortran90.

3.1 GUI Module

3.1.1 GUI Module Implementation

JavaScript provides a simple yet powerful means to develop GUIs for Web-based applications. A number of toolkits exist that provide for a simple foundation to build rich user interfaces. One such toolkit for geographic information is provided by Google through the Google Maps API.

By design, JavaScript limits access to local resources. This is done to protect against the execution of malicious code. However, ActiveX components or XPCOM extensions can be used by JavaScript to write data to the local file system. Since there is an inherent risk, it is very important to only use these types of objects (ActiveX or XPCOM) from a trusted source in order to minimize the exposure of a computer system to malware. Without the availability of these trusted components, the use of JavaScript for the implementation of a GUI for the tool would be less than ideal.

While HTML is an open standard, its implementation by the various vendors is often muddled with proprietary extensions. While we have tried to maintain compatibility with the various Web browsers available on the market today, the requirement to write files on the client side limited our initial implementation to Microsoft's Internet Explorer (IE) Web browser. The Web application renders fine in Mozilla Firefox and is fully functional sans the capability to save the geographic data through the GUI. Currently, a PC Vista user who prefers to run on Mozilla Firefox can also save data to a local file by downloading an add-on named IE Tab2 (FF3.6+), which is freely available on the Internet. This add-on allows switching to IE from inside Firefox in order to run the GUI. The procedure to obtain the add-on is provided at the beginning of appendix A.

Figure 5 is a screen capture of the GUI module using the most recent version, Google Maps API v.3, rendered in Mozilla Firefox. It has been implemented as a Web page using HTML and JavaScript. The initial interface discussed here was to demonstrate a proof of concept and the capability to leverage the Geographic Information System (GIS) information available through the Google Maps API. It provides the user an interactive way to create polygonal outlines of vegetative canopies and buildings that can later be rasterized by the rasterization module. The GUI consists of two panels, a map panel and an input panel. The map panel is an embedded Google Maps display. Its display control is self-explanatory: this component contains the embedded Google Map that the user interacts with when the control is in "satellite" display mode to mark and/or define the polygonal boundaries of vegetative canopies and buildings.

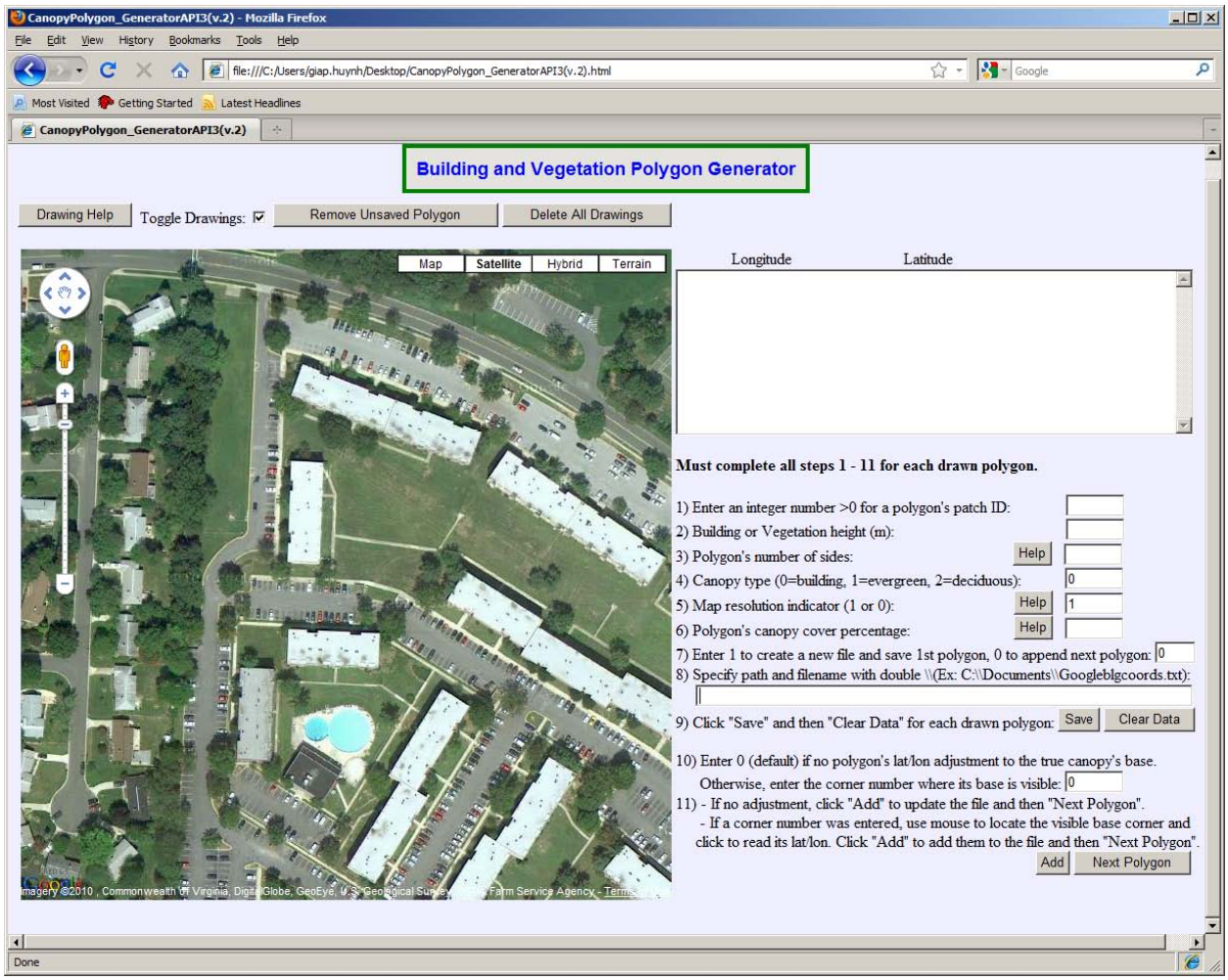


Figure 5. Google Maps based GUI Web page.

The input panel provides two functions. The first function is to display a list of coordinates that the user has marked, defining the vertices of a polygon that indicate the boundaries of a canopy or building. These coordinates are displayed at the top of the input panel. The second function is to gather information about the polygon that is currently being digitized so that it can be properly rasterized later by the rasterization program. These inputs are explained in depth in appendix A.

While the satellite imagery that Google Maps uses is geo-referenced and orthorectified, there are still instances in the imagery where the off nadir angle of the image leads to significant photogrammetric parallax. To correct for this error, we have implemented a simple base point correction algorithm, which is discussed in section 4. Since the satellite imagery is orthorectified, the user can select a corner of the building that is at the surface or ground level, which can be used to make a simple shift correction of a defined polygon. The input to make this correction, if desired, is at the bottom of the input panel.

3.1.2 GUI Module Use Example

Following the convention discussed in section 2, the user uses the mouse to click on a corner point creating the first vertex of the polygon defining the canopy area or building to be rasterized. Continuing clockwise, the user marks each successive vertex of the polygon, which the user interface labels sequentially starting from 1 (1 to 4 in figure 6). The resulting polygon is represented by the blue shaded area. To simplify the rasterization of a complex canopy footprint, it is recommended that the user decompose the canopy footprint into overlapping quadrilateral (or triangular, if quadrilaterals are not feasible) blocks with the lowest canopy height block being drawn first and then going to the next taller, etc. To maintain continuous coverage, the canopy blocks with lower canopy heights should be drawn with a little overlap into the higher canopy height blocks. This is done to avoid the creation of a gap between two adjacent blocks due to the adjusting and shifting of the edge grids during the rasterization process (see the right side of rectangle in figure 2). The user can also define a canopy polygon without decomposition as long as the polygon is convex or is one of the special cases discussed earlier for concave polygons.

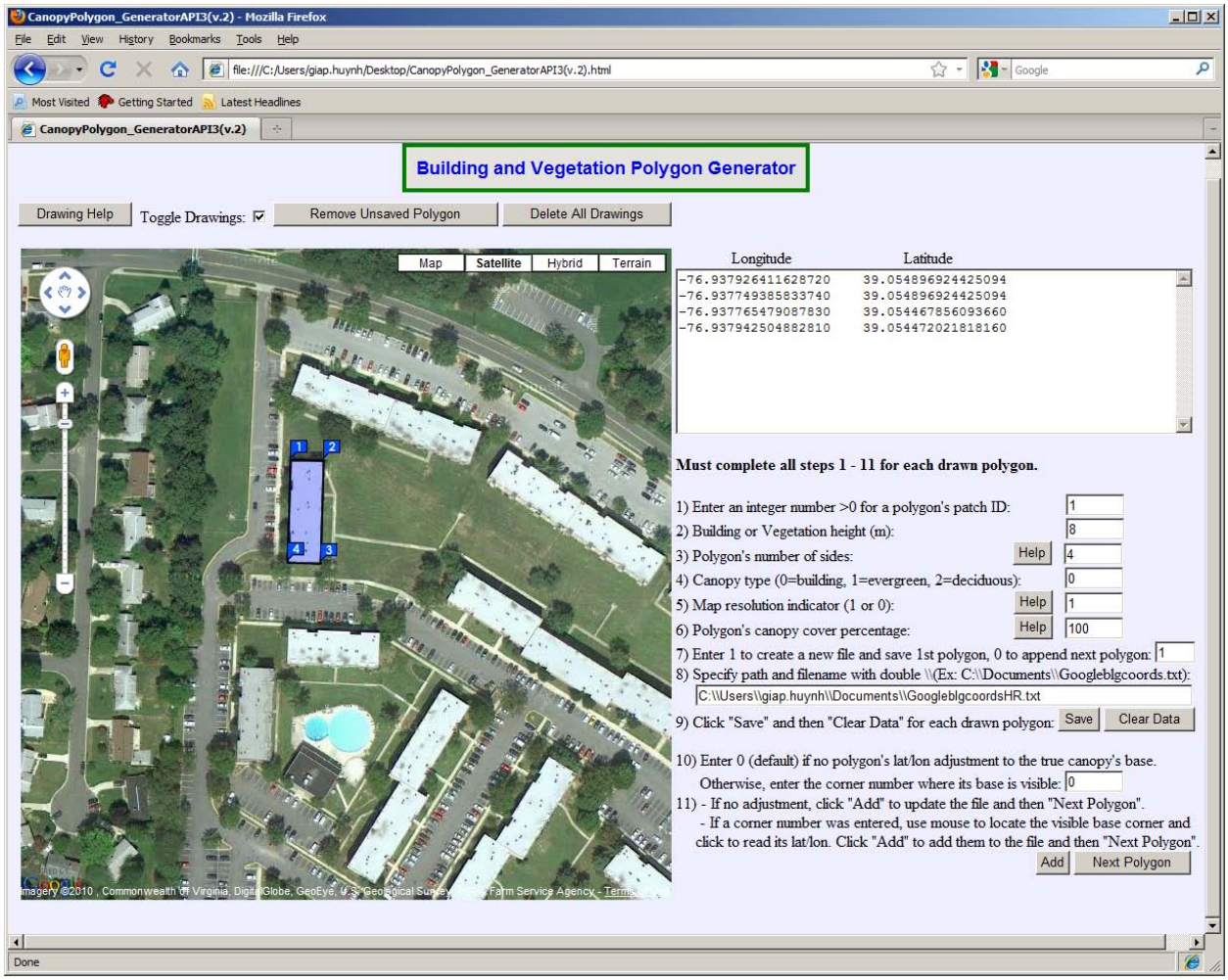


Figure 6. Polygon drawing of a rectangular building's roof.

With each click on the Google Maps interface, a new longitude and latitude pair is added to the "Longitude Latitude" text box. The remaining input controls are awaiting user input about the properties of the digitized object, such as the canopy type (whether building or vegetation), canopy patch identification (ID), building or average canopy height, the polygon's number of sides, vegetative canopy type, high or low resolution map, and estimated canopy cover density. The user must also set the path and filename where the data should be saved.

As stated previously, it is important that the user follow the instructions provided in appendix A to complete all the steps in digitizing each polygon. It is advisable to digitize only one canopy or building object at a time to avoid confusion. After each polygon is done, the user needs to reset the input controls to their default values. This is done by clicking the "Next Polygon" button. Figure 7 displays the drawing after a second building has been digitized.

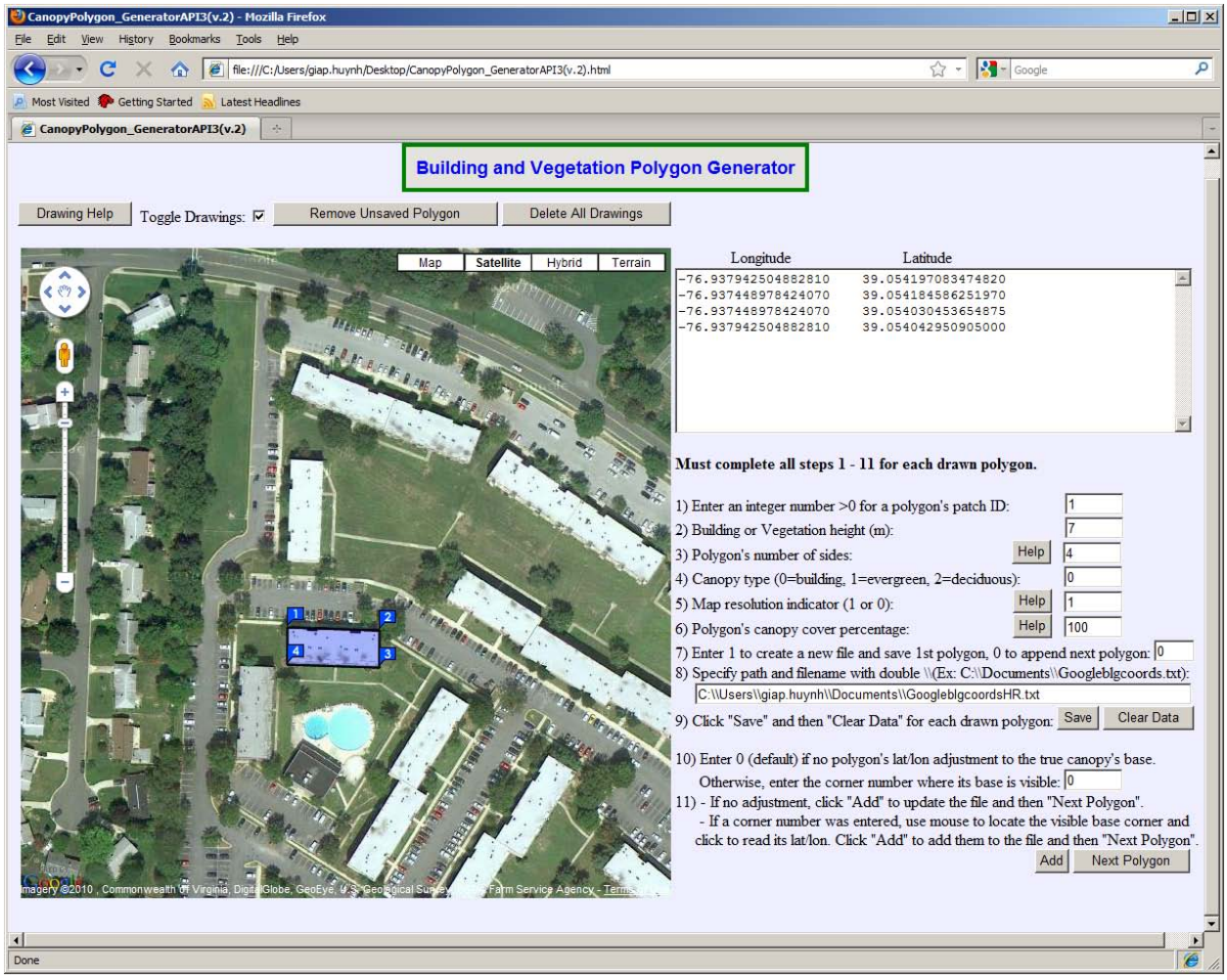


Figure 7. Polygon drawing of a nearby building from the previous map.

From the previous example, the saved file name, “GoogleblgcoordsHR.txt,” is stored in specified subdirectory and its contents are shown in figure 8. This file stores one or more records of polygonal data derived from the user’s input. The polygonal data record has the following format:

- The first row is the patch ID of one enclosed area. The user is free to assign an ID according to their own needs. Many four-sided polygonal blocks within a big complex patch or canopy can have the same ID and the same height. A separated or disjoint canopy or patch can also have the same ID. For a low-resolution or canopy-resolving map, the patch is assumed to have one average height for every grid point.
- The second row lists the canopy’s height in meters.
- The third row gives the number of vertices.
- The fourth row is the coded number indicating canopy type (0 = building, 1 = evergreen, 2 = deciduous).
- The fifth row indicates whether the map is high (building resolving, set to 1) or low (canopy resolving, set to 0) resolution.
- The sixth row lists the density in percentage of the canopy’s cover patch area. Therefore, a canopy only patch or polygon has a density value of 100.
- The seventh row lists the total number n of the longitude and latitude pairs of vertices.
- The eighth through 8+n-1 rows list the longitude and latitude pairs as they appeared in the “Long and Lat” text box for each polygon.
- The (8+n)th row lists an integer number indicating which vertex (labeled corner number) should be used for correcting the location of the base of a building. We call this the true coordinate adjustment and it is used to correct photogrammetric parallax. This is an integer ranging from 0 (indicating no adjustment) to 16 (corner numbered 16 for up to 16 sides). The chosen vertex must have its base visible to be located by the crosshair cursor and by a mouse’s click. The default value for this input is 0, indicating no adjustment required or requested.
- The (8+n+1)th row is the longitude and latitude pair of the visible chosen vertex to be used in the building location correction. Two zeros indicate no correction selected.
- Finally, the last row of the polygon data record displays the polygon’s sequential order number in the file starting from 1. Hence, the last number at the bottom of the file indicates the total number of polygon records and is important for the Fortran90 main program to identify the maximum value during looping.

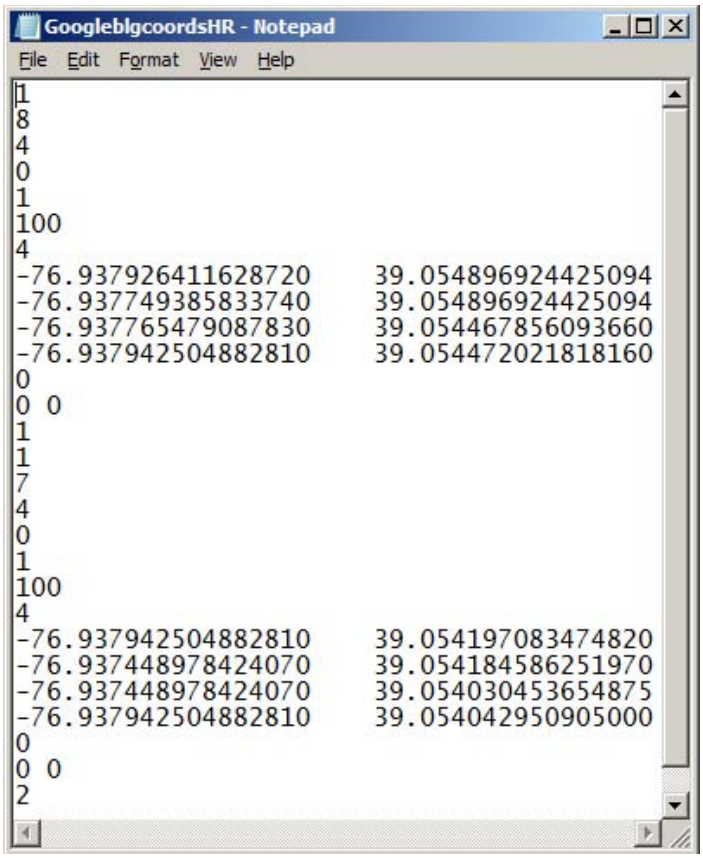


Figure 8. Contents of data file “GoogleblgcoordsHR.txt” for the polygons illustrated in figures 6 and 7.

After the filename and path are typed in, the user clicks on the “Save” button each time to save each polygon record to the named file. The user must also click “Clear Data” button to the right to clear old displayed latitude and longitude values before entering any new information for a new polygon. The “Save” button saves whatever displayed in the “Longitude Latitude” text box; therefore, if old values are still there, they will be saved as well in the new polygon’s information and this will cause error in the reading of a polygon record format. The filename input is persistent and is not cleared. An old saved data file from a previous session can be used to add more data records too, but the user would have to manually edit the polygon record sequence number since it will start again at 1. Later on, the user needs to copy this data file to the same directory where the main Fortran90 module is located for the rasterization process.

3.2 Fortran90 Program Module

The rasterization code was initially prototyped using MatLab. However, since execution speed was of greater importance to our application, the algorithm has since been implemented using Fortran90.

The main function of this module is to rasterize the digitized canopy polygons that the user created using the GUI module and produce the morphology input files for the 3DWF model. There were two versions implemented in Fortran90. The first named “Rasterize_4sides.f90” was designed to rasterize triangles and quadrilaterals. Since it is difficult to accurately represent a shape such as a circle with a limited number of quadrilaterals, a second version was implemented named “Rasterize_16sides.f90,” which could rasterize up to a hexadecagon (16-sided polygon).

For development purposes, we compile and run the rasterizer in a Linux environment. Upon execution, the rasterizing application prompts the user with a series of questions. The user’s responses are then used to create the output filenames. There are two input files that the program reads, the text file containing the polygon records created by the GUI module (“GoogleblgcoordsHR.txt” for this example) and the terrain map file. Here the user is directed to the detailed instructions in part II of the appendix A to run this module. The program then extracts the terrain height, latitude, and longitude data from the terrain data file of a gridded model domain. It then separates each data type and also creates a map of relative terrain heights. Here relative terrain height is defined as the height relative to the terrain’s lowest point. The lowest point is mapped to the zero height value. The program handles both high resolution (building resolving) and low resolution (canopy resolving) terrain maps and generates the needed data files accordingly. The map file that is produced includes the rasterized canopy polygons and can be displayed in two or three dimensions for verification purposes by using a simple MatLab program.

4. Photogrammetric Parallax Correction

When the captured satellite or aerial image is tilted or significantly off nadir, an error is induced in the location of the top of the building that is used to digitize or define the polygon. The error is caused by a well-known phenomenon called photogrammetric parallax. The apparent location of the top of the building could be significantly shifted from its true location. The polygon associated with the roof of a building must be adjusted to its base to get a better estimate of its true location. In order to accomplish this correction, the user needs to select one visible building’s base corner. This base corner ID number (3B from figure 9) is then entered into the building base input box. This vertex/corner is then used by the main Fortran90 program to make a simple correction described below. The latitude and longitude coordinates of this selected base corner allows computations for adjusting all the vertices of the polygon representing the top of the building.

The adjustment process is done by calculating the differences of latitude and longitude (dLat and dLon) between the visible building base corner selected by the user and the corresponding vertex of the polygon representing the top of the building. For example, 3B and its corresponding top corner 3T in figure 9.

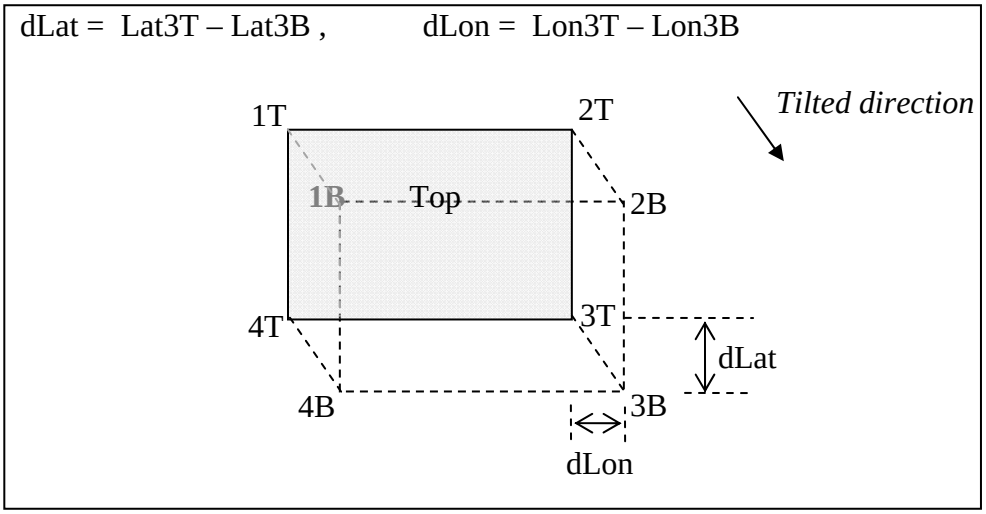


Figure 9. Photogrammetric parallax correction (true coordinate adjustment) illustration.

The adjustments for each of the four building base corners are defined as follow:

- At point 3B: Lat3B and Lon3B are defined by the click and given on the GUI's window.
- At point 4B: Lat4B = Lat4T – dLat, Lon4B = Lon4T – dLon
- At point 1B: Lat1B = Lat1T – dLat, Lon1B = Lon1T – dLon
- At point 2B: Lat2B = Lat2T – dLat, Lon2B = Lon2T – dLon

5. Mapping Geo-coordinates to 3DWF Model Grid

The Fortran90 main program maps the latitude and longitude pair position of each polygon vertex to the nearest model grid position (x, y) on the model grid domain by calling subroutine "loctempgrid." The reason is because the rasterization process is done in the model grid space and not the geo-coordinate space. This process to identify the correct grid position is illustrated in figure 10. The initial grid (labeled as 1) coordinate pixt or piyt position is assigned when the scanned grid's longitude or latitude is greater than the clicked point's longitude (lon) or latitude (lat), or pixt>lon and piyt>lat, respectively. However, this initial guess on the model grid may not necessarily be the model grid nearest the current vertex. For example, from figure 10, the x coordinate distance from the clicked point's lon to grid 2 is shorter than its distance to the initial grid point 1. Therefore, grid point 2 is selected rather than grid point 1. Performing this test in both x and y coordinates results in the nearest model grid point being set as the polygon vertex in the model grid space.

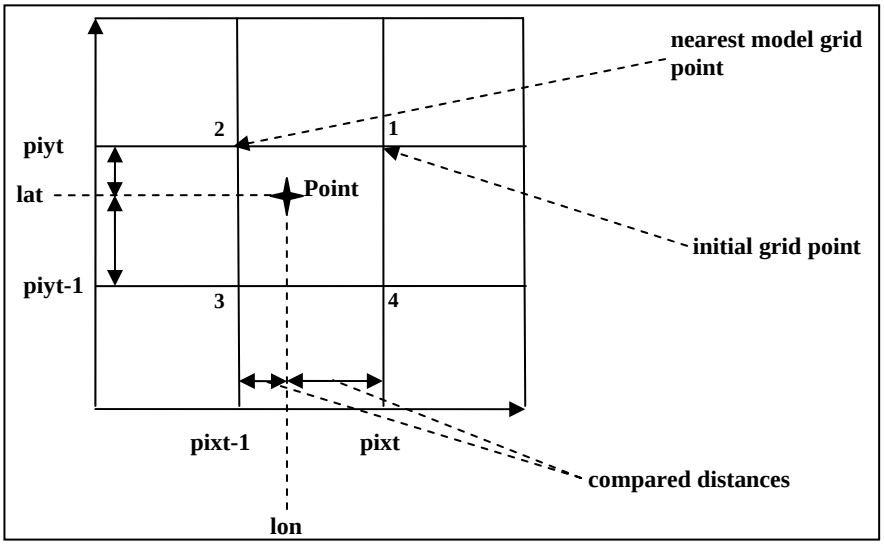


Figure 10. Mapping polygon vertex to model grid process.

6. Building and Vegetative Canopy Representations

The 3DWF wind model uses two different flow parameterizations. The first flow parameterization is associated with the vertical wind profile. The vertical wind profile varies depending on what the underlying canopy type is. The second flow parameterization is a wake flow parameterization for complex terrain and buildings. This second parameterization requires an explicit representation of buildings.

To explicitly resolve a building on the model grid, the horizontal resolution should be no greater than 10 m. We, therefore, categorize the types of binary maps that the rasterization program creates as canopy-resolving for grids with horizontal resolution greater than 10 m and building-resolving when the grid resolution is less than 10 m. For a building-resolving binary map, the second parameterization is activated.

We should note that the vegetative canopy representation is identical for either type of map as it is not resolution dependent. This representation requires rasterizing the vegetative canopies to the model grid so that the type and average height of the canopy can be known at every grid point in order to apply the appropriate vertical wind profile. At coarser resolutions, canopy types include coniferous, deciduous, suburban, and bare soil.

At finer resolutions, when buildings are resolved, we require a slightly different treatment. The vegetative canopies are handled in exactly the same manner; it is our treatment of buildings and/or urban areas that changes. For polygons that represent buildings, there are two things that we do. First, we produce an augmented terrain file and, second, we create a file that indicates

where the building is in the 3-D model grid (see figure 12, left panel). To produce the augmented terrain file, we adjust the terrain elevation where buildings have been explicitly resolved in the following manner. After the rasterization routine has run and we know where a building is located in the two-dimensional (2-D) grid map, we locate the corner of the building that has the lowest elevation and add the building height to get the grid represented building height (figure 11). We then apply the buildings mask on the terrain elevation map and assigning this new height value to produce the augmented terrain elevation map (see examples in sections 7 and 8).

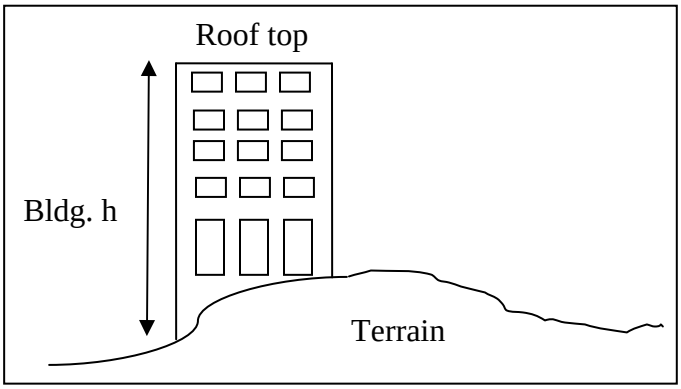


Figure 11. Illustration of building augmentation of terrain elevation map for the building resolving map.

After the rasterization process is completed two binary rasterized maps are produced. One of the map files contains the latitude, longitude, terrain elevation (possibly augmented with building heights), and canopy height. The second map file contains the canopy patch ID that the user defined when they were digitizing the canopies as polygons.

7. Outputs

After execution, the rasterization program produces two ASCII files. For reference, we typically name these files as “CanopygridslayersHR.txt” and “CanopytypeHR.txt” (figure 12). Figures 12 to 14 are results from a different example and are for illustration purposes only. For the file “CanopygridslayersHR.txt,” as illustrated in figure 12, we have the following format for records:

- The first row is the total number of polygons and it appears only once. After this row, repeating sequences appear as the following format.
- The second row is the number of polygon vertices (n),
- The third through (n+3) rows are the *x*, *y* grid positions of the polygonal vertices,

- The (n+4)th row is the lower bound on the vertical grid layer where the building is defined, and
- The (n+5)th row is upper bound on the vertical model grid where the building is defined.

The file “CanopytypeHR.txt” as illustrated in the right panel of figure 12 contains the patch ID, the canopy type (0 = building), the canopy height in meters, and the canopy cover density (1 = 100%).

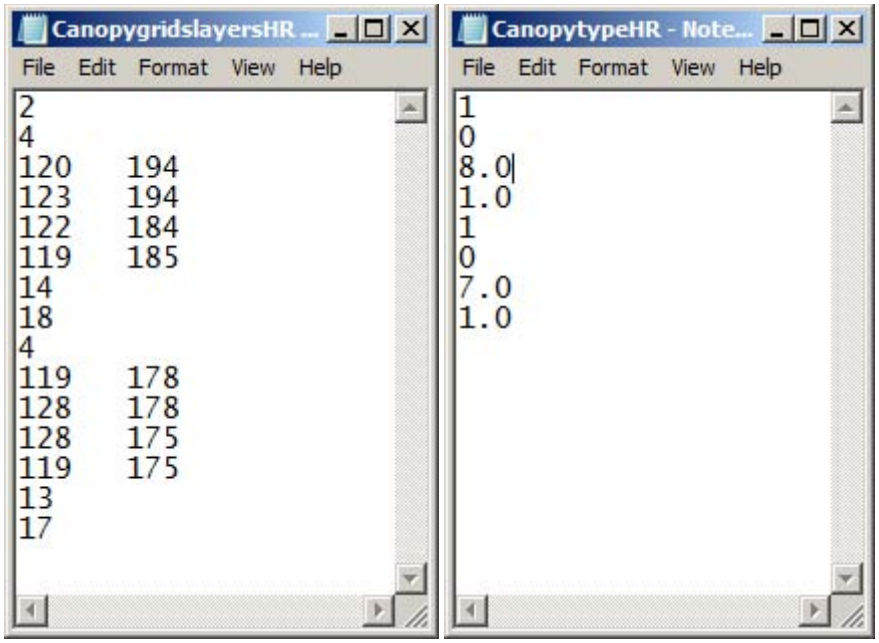


Figure 12. Content of files “CanopygridslayersHR.txt” (left) and “CanopytypeHR.txt” (right).

As discussed in the previous section, the program also produces two binary domain map files, named “ht_lat_long_tercanopy_fileHR.dat” and “ht_lat_long_patchcanopy_fileHR.dat,” respectively. These two files are used by the 3DWF model. These binary files contain the rasterized polygons, as illustrated in figure 13. The 2-D and 3-D graphical representations are also displayed in appendix B using MatLab. The top panel of figure 13 displays an area zoomed in to the two drawn buildings of the patch ID map, and the bottom panel of the same figure shows the buildings as represented in the augmented terrain elevation map.

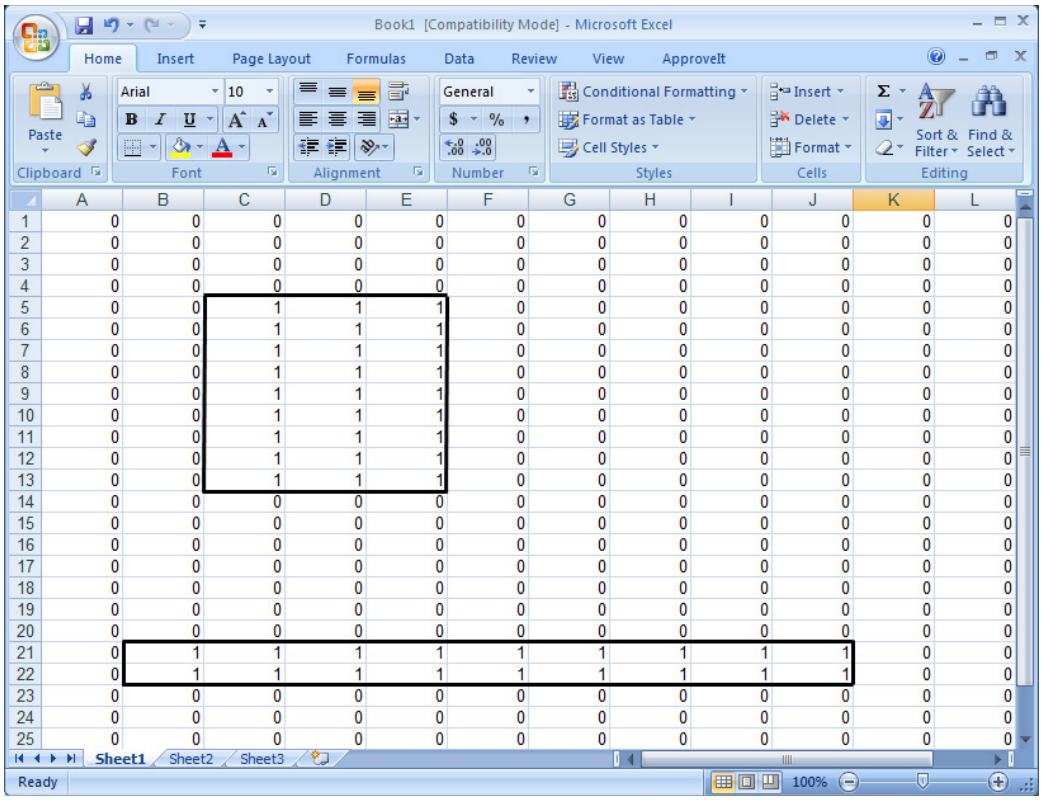
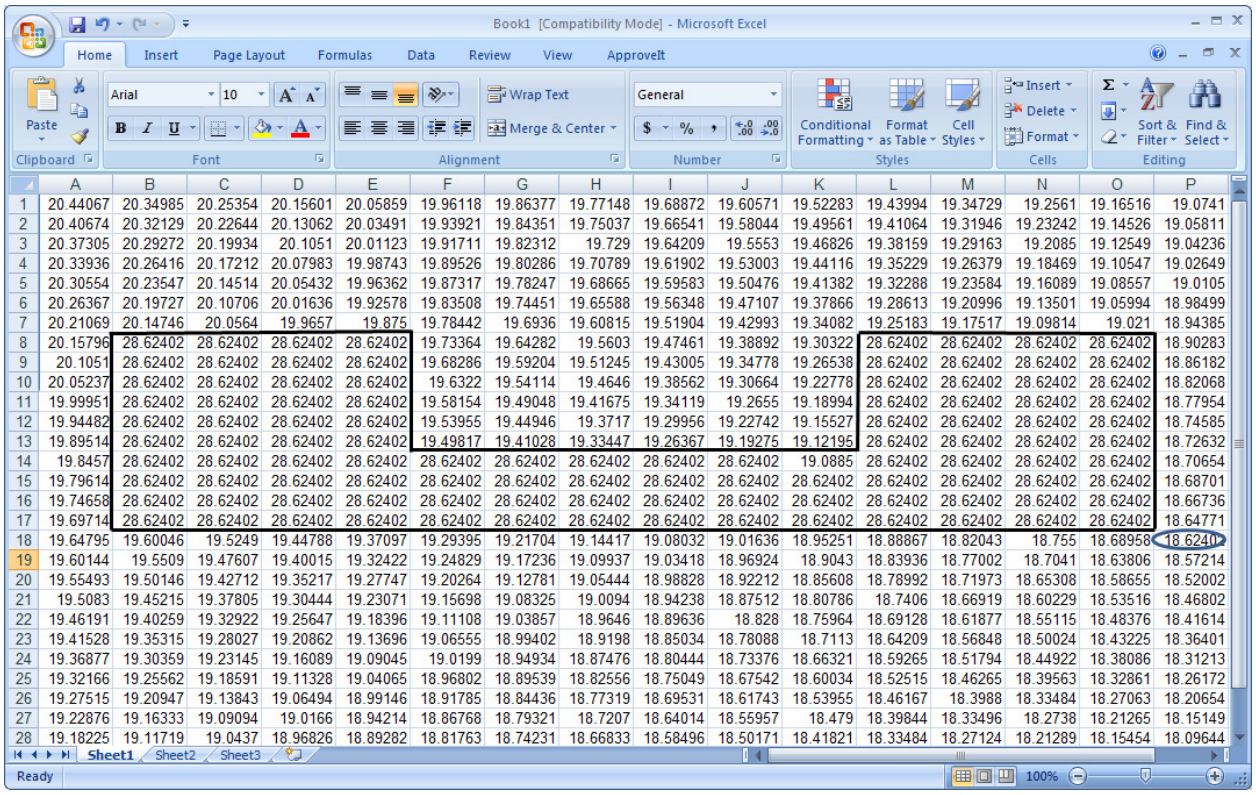


Figure 13. Contents of the canopy patch ID map file (top) and the terrain elevation (m) map file (bottom).

8. Example Rasterization of “U-shaped” Polygon

An example of the rasterization of a “U-shaped” building is shown in figure 14. This zoomed-in figure shows only a small window where the rasterization of the “U-shaped” building took place. For this building, it was assumed that the roof was flat and level with a total building height of 10 m; therefore, all grids within the rasterized area have the same height above sea level of 28.62402 m (= the lowest corner’s terrain height of 18.62402 m at the edge of the building (oval circle) + 10 m). Note that there is one grid with value of 19.0885 m lies inside the U outline as a result of imperfect drawing during the outlining of the U-shaped building.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	20.44067	20.34985	20.25354	20.15601	20.05859	19.96118	19.86377	19.77148	19.68872	19.60571	19.52283	19.43994	19.34729	19.2561	19.16516	19.0741
2	20.40674	20.32129	20.22644	20.13062	20.03491	19.93921	19.84351	19.75037	19.66541	19.58044	19.49561	19.41064	19.31946	19.23242	19.14526	19.05811
3	20.37305	20.29272	20.19934	20.1051	20.01123	19.91711	19.82312	19.729	19.64209	19.5553	19.46826	19.38159	19.29163	19.2085	19.12549	19.04236
4	20.33936	20.26416	20.17212	20.07983	19.98743	19.89526	19.80286	19.70789	19.61902	19.53003	19.44116	19.35229	19.26379	19.18469	19.10547	19.02649
5	20.30554	20.23547	20.14514	20.05432	19.96362	19.87317	19.78247	19.68665	19.59583	19.50476	19.41382	19.32288	19.23584	19.16089	19.08557	19.0105
6	20.26367	20.19727	20.10706	20.01636	19.92578	19.83508	19.74451	19.65588	19.56348	19.47107	19.37866	19.28613	19.20996	19.13501	19.05994	18.98499
7	20.21069	20.14746	20.0564	19.9657	19.875	19.78442	19.6936	19.60815	19.51904	19.42993	19.34082	19.25183	19.17517	19.09814	19.021	18.94385
8	20.15796	28.62402	28.62402	28.62402	28.62402	19.73364	19.64282	19.5603	19.47461	19.38892	19.30322	28.62402	28.62402	28.62402	28.62402	18.90283
9	20.1051	28.62402	28.62402	28.62402	28.62402	19.68286	19.59204	19.51245	19.43005	19.34778	19.26538	28.62402	28.62402	28.62402	28.62402	18.86182
10	20.05237	28.62402	28.62402	28.62402	28.62402	19.6322	19.54114	19.4646	19.38562	19.30664	19.22778	28.62402	28.62402	28.62402	28.62402	18.82068
11	19.99951	28.62402	28.62402	28.62402	28.62402	19.58154	19.49048	19.41675	19.34119	19.2655	19.18994	28.62402	28.62402	28.62402	28.62402	18.77954
12	19.94482	28.62402	28.62402	28.62402	28.62402	19.53955	19.44946	19.3717	19.29956	19.22742	19.15527	28.62402	28.62402	28.62402	28.62402	18.74585
13	19.89514	28.62402	28.62402	28.62402	28.62402	19.49817	19.41028	19.33447	19.26367	19.19275	19.12195	28.62402	28.62402	28.62402	28.62402	18.72632
14	19.8457	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	19.0885	28.62402	28.62402	28.62402	28.62402	18.70654
15	19.79614	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	18.68701
16	19.74658	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	18.66736
17	19.69714	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	28.62402	18.64771
18	19.64795	19.60046	19.5249	19.44788	19.37097	19.29395	19.21704	19.14417	19.08032	19.01636	18.95251	18.88867	18.82043	18.755	18.68958	18.62402
19	19.60144	19.5509	19.47607	19.40015	19.32422	19.24829	19.17236	19.09937	19.03418	18.96924	18.9043	18.83936	18.77002	18.7041	18.63806	18.57214
20	19.55493	19.50146	19.42712	19.35217	19.27747	19.20264	19.12781	19.05444	18.98828	18.92212	18.85608	18.78992	18.71973	18.65308	18.58655	18.52002
21	19.5083	19.45215	19.37805	19.30444	19.23071	19.15698	19.08325	19.0094	18.94238	18.87512	18.80786	18.7406	18.66919	18.60229	18.53516	18.46802
22	19.46191	19.40259	19.32922	19.25647	19.18396	19.11108	19.03857	18.9646	18.89636	18.828	18.75964	18.69128	18.61877	18.55115	18.48376	18.41614
23	19.41528	19.35315	19.28027	19.20862	19.13696	19.06555	18.99402	18.9198	18.85034	18.78088	18.7113	18.64209	18.56848	18.50024	18.43225	18.36401
24	19.36877	19.30359	19.23145	19.16089	19.09045	19.0199	18.94934	18.87476	18.80444	18.73376	18.66321	18.59265	18.51794	18.44922	18.38086	18.31213
25	19.32166	19.25562	19.18591	19.11328	19.04065	18.96802	18.89539	18.82556	18.75049	18.67542	18.60034	18.52515	18.46265	18.39563	18.32861	18.26172
26	19.27515	19.20947	19.13843	19.06494	18.99146	18.91785	18.84436	18.77319	18.69531	18.61743	18.53955	18.46167	18.3988	18.33484	18.27063	18.20654
27	19.22876	19.16333	19.09094	19.0166	18.94214	18.86768	18.79321	18.7207	18.64014	18.55957	18.479	18.39844	18.33496	18.2738	18.21265	18.15149
28	19.18225	19.11719	19.0437	18.96826	18.89282	18.81763	18.74231	18.66833	18.58496	18.50171	18.41821	18.33484	18.27124	18.21289	18.15454	18.09644

Figure 14. Example of an augmented terrain elevation (m) map augmented by a rasterized “U-shaped” building.

9. Conclusion

In this report, we have documented initial efforts to leverage the GIS data that is made available through the Google Maps API. We have implemented a simple GUI that uses the Google Maps interface to assist a user in defining vegetative canopy coverage areas as well as the location of buildings. We have also implemented a simple rasterization program using Fortran90 that has been used to produce building resolving morphology maps for use by the 3DWF model.

As this has been an initial effort, there are a number of improvements that can be made. The current implementation demonstrates a proof of concept, and as such, is quite rough in its implementation. We initially implemented the GUI using HTML version 4 and the Google Maps API version 2. Google has since released version 3 of their API, which is more flexible and expands on the feature set. Also the current version of the HTML standard is version 5. The HTML version 5 specification defines and implements the concept of local storage. This could be leveraged to create a Web application that is cross-platform and not dependent upon proprietary extensions such as Microsoft ActiveX.

There is also a need to restructure and rethink the implementation of the JavaScript code. It would be desirable to save the digitized polygons in an object array so that the polygons could be accessed after their creation and modified as necessary.

The GUI could also be redesigned to leverage Web technologies such as cascading style sheets. This could be used to give the Web application a more polished finish.

It is also our desire and intent to integrate this tool into a larger suite that is used to control the execution of the 3DWF model. In light of this desire, we have considered implementing the rasterization routine using the Java computer language to help simplify some of the aspects of integration.

10. References

Google API Web site. http://code.google.com/apis/maps/ (accessed November 2010).

Sheffer, A. Scan Conversion (part 2) Drawing Polygons on Raster Display; Chapter 8 lecture viewgraphs, University of British Columbia. 2005, 13–25.

Wang, Y.; Williamson, C.; Garvey, D.; Chang, S.; Cogan, J. Application of a Multigrid Method to a Mass Consistent Diagnostic Wind Model. *J. Appl. Meteorol*. **2005**, *44*, 1078–1089.

Wang, Y.; Williamson, C.; Huynh, G.; Emmitt, D.; Greco, S. 2010, Diagnostic Wind Model Initialization over a Complex Terrain Using the Airborne Doppler Wind Lidar Data. *Open Remote Sensing Journal* **2010**, *3*, 17–27.

Appendix A. Procedure to Prepare Input Files for the 3DWF Model

There are two separate procedures for the GUI and the Fortran90 program:

I. Procedure for the GUI module

1. Activate Mozilla Firefox (or Internet Explorer, if using computer with Windows operation other than Vista). For Vista users, an add-on named IE Tab2 (FF3.6+) freely available from the Internet must be installed. Follow steps a to g to download:

   a. Open Firefox by clicking "Getting Started" and then click on the orange arrow next to "Need helps" at the bottom right of the Firefox window.

   b. A new window appears with some lists. Click on "Add-ons" at the list under "Community."

   c. An "Add-ons for Firefox" window appears. Type "IE" inside the slot "search for add-ons" and click the green arrow at the end to start searching.

   d. A list of all current available add-ons developed by the Firefox community will appear, ready for download. Select the add-on named "IE Tab 2 (FF 3.6+)" (developed by ietab.net) by clicking on the green button "Add to Firefox."

   e. A small window will appear with a warning message and with the address link to the add-on. Click on the address link to activate the "Install Now" button.

   f. Click on the "Install Now" button to start installation. A new small window will appear and when done, click on "Restart Firefox" button to activate the new feature. Notice a small Firefox logo will appear at the bottom right corner next to the key lock logo.

   g. Click on that Firefox logo will switch to an IE logo. It will then enable the user to work inside IE environment and allow saving data file to the local disk. To switch back to Firefox, click on the IE logo to change.

2. Click "File" and choose "Open File." A new window appears to allow user to select the HTML file named "CanopyPolygon_GeneratorAPI3.html."

3. Left mouse click on the Firefox's logo at the lower-right corner of the Web page to change to IE's logo so that the Google's satellite map can be displayed. Click on the "Satellite" option if not default to see the real map.

4. For simplicity, only polygon mode or surface painted mode is available in the current version. Choose the target canopy (building or vegetation area) with the help of zoom in/out scale on the map. For more accuracy, the user should zoom in as close as possible. Use the left mouse button to move the crosshair cursor and click on the first corner of the canopy's polygon. An Integer ID will label that first corner as "1" and its longitude and latitude pair will appear on the top of the input panel. The user then continues onto the next corner following clockwise direction until all interested corners are clicked.

   Although the Fortran90 codes allow rasterization of a complete convex polygon up to 16 sides and of concave shapes such as L or U shape, it is advisable to divide the concave shape into 3-sided or 4-sided blocks and then draw and save each block at a time:

   a. For a same height patch, such as in a low-resolution case, each block should overlap into the other as much as possible to prevent a gap in between due to the adjusting and shifting of the edge's grids (see the right side of rectangle in figure 2 in the main report) during translation from lat/long to grid (pixel) when the plot point's lat/long is not at the exact center of the grid.

   b. For different heights patch, such as in the case of a building complex with different heights, the lowest height block should be drawn first and should overlap or extend into the neighbored higher height block as much as possible to avoid the gap mentioned previously. The rasterization of the higher height block will erase the lower height block's overlapped area.

5. After plotting all corners of a polygon, the user will start to enter input or change the default 0 or 1 in each of the following slots from top to bottom of the input panel:

   a. **Enter an integer number >0 for a polygon's patch ID**: Enter any integer number to label a single polygon drawn on the map.

   b. **Building or vegetation height (m)**: Enter the known or approximated height above ground in meters. For vegetation patch, its height is the averaged tree height.

   c. **Polygon's number of sides**: Enter the number of sides for each drawn polygon. For a complex or a concave shape of canopy where it must be divided into three- or four-sided blocks/polygons, then the number enter will be 3 or 4 accordingly and not the number of sides of the complex or concave shape. For more explanation, click the "Help" button.

   d. **Canopy type (0=building, 1=evergreen, 2=deciduous)**: Default 0 is for building, change to 1 for evergreen or 2 for deciduous type of vegetation.

e. **Map resolution indicator (Enter 1 for ≤10m (HR) and 0 for others (LR)):** Enter 1 or 0 to signal whether the binary terrain map to be used in the Fortran90 program is for building-resolving (high) or canopy-resolving (low resolution), respectively.

f. **Polygon's canopy cover percentage:** Enter approximated percentage from 1 to 100 of canopy coverage over each drawn polygon area. Therefore, 100 will be entered for a polygon of building only. For multi-sided patch/area, each four-sided polygon or block within a patch is treated to have the same percentage of canopy coverage as the whole patch area. Click the "Help" button for more hints.

g. **Save data: Enter 1 to create a new file, 0 to append subsequent polygons:** Default 1 is to create a new file, or enter 1 to redo all over again from the scratch. It will automatically change to 0 after the first polygon saving to signal appending of the next polygon's data into the same file.

   However, if new data is required to be saved in a previously created file from a previous session, then the default 0 should be left alone to signal appending at the end of that old file. The user also needs to edit this file manually later to correct the order of the polygon sequential count number since the new session always starts at polygon count number 1.

   To avoid accidently appending to an old but same name file, the user should make sure to name a new file differently.

h. **Specify path and filename with double \\(Ex: C:\\Documents\\Googleblgcoords.txt):** Enter a filename and its directory path where it will be saved. Remember to separate subdirectories with a double forward slashes \\ to save the data in a text file type (including ".txt").

i. **Click "Save"and then "Clear data" for each drawn polygon:** To save data into the file, click the "Save" button, and then click "Clear data" to clear old values from the input panel. These two buttons must be clicked for each drawn polygon or divided block.

j. **Enter 0 (default) if no polygon's lat/lon adjustment to the true canopy's base; otherwise, enter the corner number where its base is visible:** Default value 0 means there is no adjustment to be made to the building corners' coordinates (satellite image is not tilted or significant). Otherwise, if the building is tilted, its top corners' longitude and latitude coordinates need to be adjusted to its true coordinates at the base. In such case, the user needs to locate a base corner where it is visible to the naked eye, and enter into the slot the number identical to its indicated top corner number. Next, the user clicks at that base corner on the map and the same number will appear. Figure A-1 illustrates polygon drawing and the content of the data file for an assuming case with building coordinate adjustment.

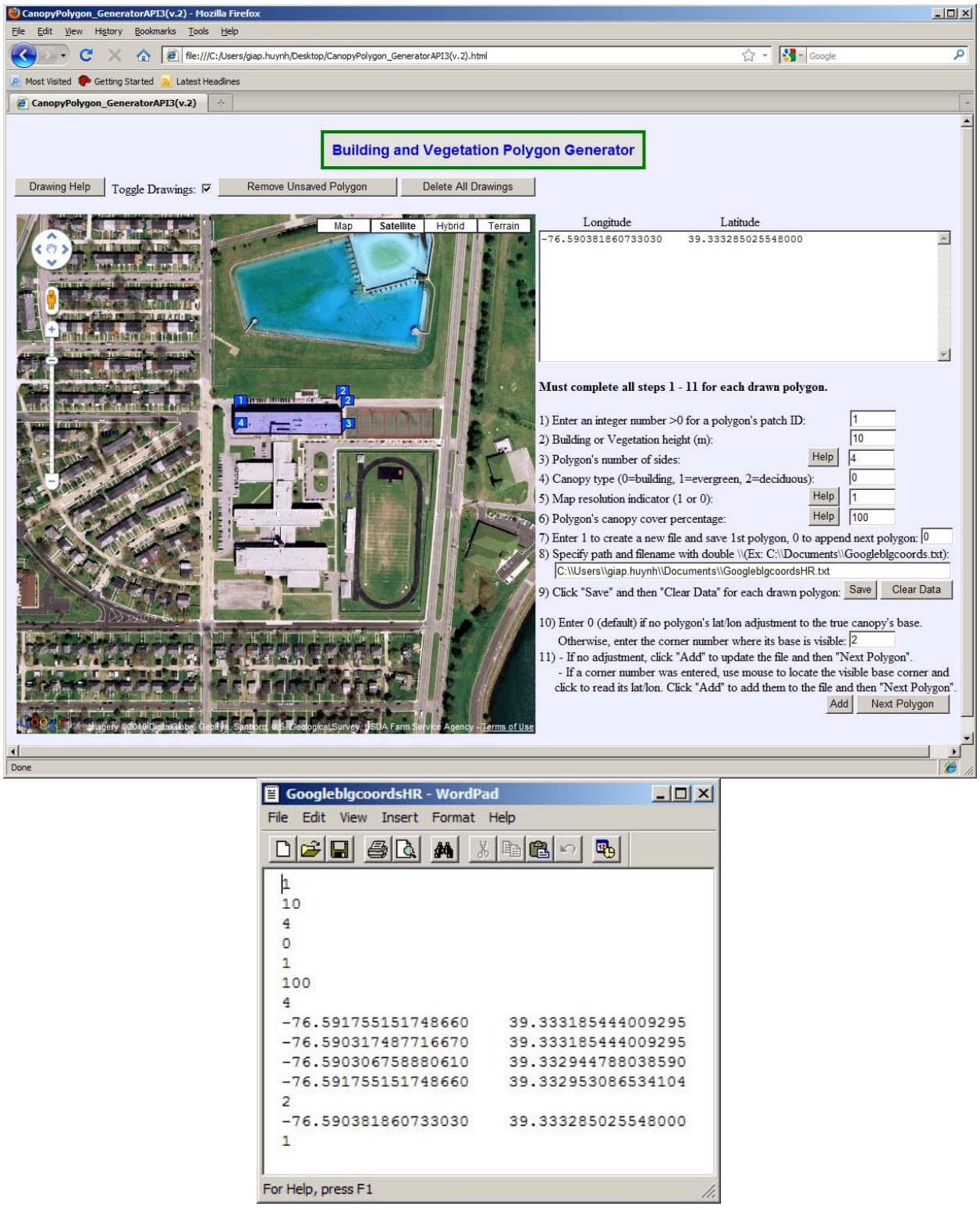


Figure A-1. Polygon drawing of a building with coordinate adjustment case (top) and its data file (bottom).

k. **If no adjustment, click "Add" to update the file and then "Next Polygon" to continue or to end:** If a corner number entered, use mouse to locate (click) the visible base corner's lat/lon, then click "Add" to add its coordinates to the file, and then "Next Polygon" to continue or to end: Even in the case of no adjustment, the user still needs to click the "Add" button to add two zeros to the file indicating no longitude and latitude recorded, and click the "Next Polygon" button to clear the input panel and to continue or to end. In the case of adjustment, click the "Add" button to add the chosen point's longitude and latitude pair to the file, and then click "Next Polygon" to clear old inputs from the input panel and to continue on or to stop.

Other convenient features and important notes:

- Repeat the same procedure from the beginning for each drawn polygon or block and do not skip any step.
- The "Toggle All Drawings:" check box allows user to toggle back and forth the display of all correctly drawn polygons/blocks during each session.
- The "Remove Bad Drawing" button will help to remove a bad polygon and its corners' markers in case the user makes a mistake during drawing.
- The "Delete All Drawings" button will delete permanently all correctly drawn polygons and their corners' markers from the map, but not the saved data. After this button is clicked, the "Toggle All Drawings:" check box will not display any past drawn polygons on the map.

The content of the saved text file is displayed and explained in details in this report.

II. Procedure for the Fortran90 program

1. In a Linux machine where the main Fortran90 program is stored, type **./RunRaster** or whatever the executable Fortran file is named to perform rasterization of the drawn polygons prepared by the GUI. The program will prompt with a series of following prompts:

   a. **Enter 1 for a higher-resolution map (grid's width dx, dy<=10 m), 0 for others:** This information to signal building or canopy resolve as mentioned in this report.

   b. **Enter the number of grid size on *x* coordinate:** This prompt allows the user to flexibly enter the terrain map's horizontal grid size. For example: 401 or 501.

   c. **Enter the number of grid size on *y* coordinate:** This prompt allows the user to flexibly enter the terrain map's vertical grid size.

   d. **Enter a vertical layer's increment value dz (m)**: This is the increment value between vertical layers. Layer number 1 is dz above the ground.

e. **Enter the path and the filename created by the GUI containing lat/longitude data (Ex: C://Users//Rasterize//GoogleblgcoordsHR.txt.txt)**: An example of this filename is the file "GoogleblgcoordsHR.txt" in this report. Its name is given by the user during saving process in the GUI module.

f. **Enter the path and the binary terrain map's filename to be read (Ex: C://Users//Rasterize//ht_lat_long_HR.dat)**: This is the original input file of a high or low resolution terrain map.

g. **Enter a path and a new filename including .txt type to store grids and layers information (Ex: C://Users//Rasterize//Canopygridslayers.txt)**: The user can name any names to save plotted corners' grids and the vertical layer position (integer number) of canopy's base and top.

h. **Enter a path and a new filename including .txt type to store canopy type information (Ex: C://Users//Rasterize//Canopytype.txt)**: The user can name any names to save canopy's type, patch ID, and density information. This file is necessary for the 3DWF model's input.

i. **Enter a path and a new filename including .dat type to store output map with rasterized polygons indicated by terrain+canopy's heights (Ex: C://Users//Rasterize//Outter.dat)**: The user can name any names to produce a binary output map with terrain and canopy's heights incorporated.

j. **Enter a path and a new filename including .dat type to store output map with rasterized polygons indicated by patches' ID and ground cover as 0 (Ex: C://Users//Rasterize//Outpat.dat)**: The user can name any names to produce an identical binary output map but with patch ID (1, 2, 3, etc.) and ground (as 0) presentation designed for 3DWF's purpose.

Hint: The user should examine the output files to make sure no errors occurred. For the terrain maps, a simple MatLab program will allow viewing in 3D. This viewing is actually very helpful to double check if there are any unintended gaps between blocks during rasterizing of a complex building or canopy patch of more than four sides (except for the special case of rasterizing a U or L shape). If a gap occurred, the user needs to go back and extend the lower-or-equal-height block's latitude or longitude (depending on the directions) of the two corners involved. This can also be accomplished simply by changing the latitude or longitude a little in the text file produced by GUI. Another option is the user can choose to re-plot for those particular two corner points using the GUI page. The user then uses the mouse to copy the new longitude or latitude directly from the input panel and paste to replace the old value in the text file.

Appendix B. Visualization of an Augmented Terrain Elevation File

Figure B-1 shows a visualization of an augmented terrain elevation file.

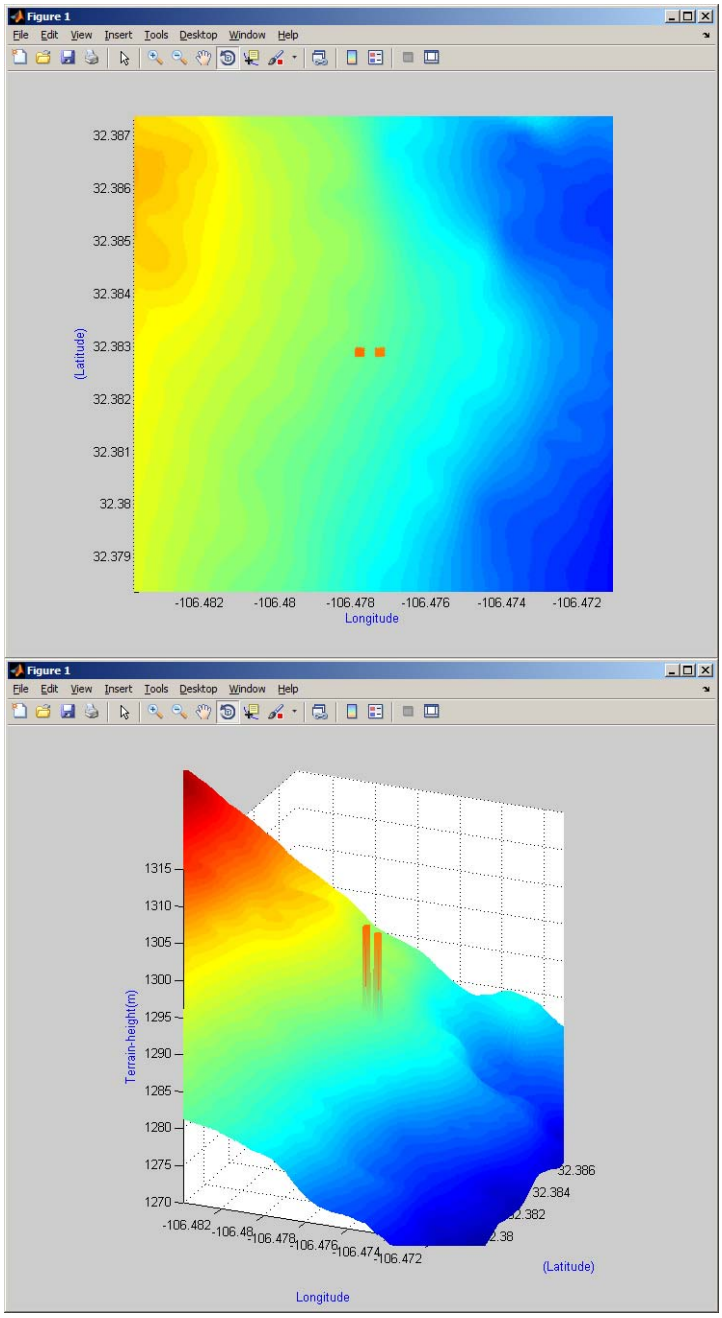


Figure B-1. Two-dimensional (top) and 3-D (bottom) MatLab visualizations of an augmented terrain map illustrating two 10-m-high buildings fused with the terrain.

INTENTIONALLY LEFT BLANK.

List of Symbols, Abbreviations, and Acronyms

2-D	two-dimensional
3-D	three-dimensional
3DWF	Three-Dimensional Wind Field
API	application programming interface
GIS	Geographic Information System
GUI	graphical user interface
HR	high resolution
ID	identification
IE	Internet Explorer
Lat	latitude
Long	longitude
LR	low resolution

No. of
Copies Organization

1 ADMNSTR
ELEC DEFNS TECHL INFO CTR
ATTN DTIC OCP
8725 JOHN J KINGMAN RD STE 0944
FT BELVOIR VA 22060-6218

1 US ARMY RSRCH LAB
ATTN RDRL CIM G T LANDFRIED
BLDG 4600
ABERDEEN PROVING GROUND MD
21005-5066

3 US ARMY RSRCH LAB
ATTN IMNE ALC HRR
MAIL & RECORDS MGMT
ATTN RDRL CIM L TECHL LIB
ATTN RDRL CIM P TECHL PUB
ADELPHI MD 20783-1197

1 ELEC US ARMY RSRCH LAB
ATTN RDRL WML B J CIEZAK-JENKINS
BLDG. 390
ABERDEEN PROVING GROUND MD
21005-5069

7 US ARMY RESEARCH LAB
ATTN RDRL CIE D
G HUYNH (5 COPIES)
Y WANG
C WILLIAMSON
ATTN RDRL CIE P CLARK
ADELPHI, MD

6 US ARMY RESEARCH LAB
ATTN RDRL CIE D
D HOOCK (5 COPIES)
G VAUCHER
BLDG 1622
WHITE SANDS MISSILE RANGE
NM 88002-5501

TOTAL: 19 (2 ELEC, 7 HCS)