ERDC/CRREL TR-21-11

US Army Corps of Engineers®
Engineer Research and Development Center

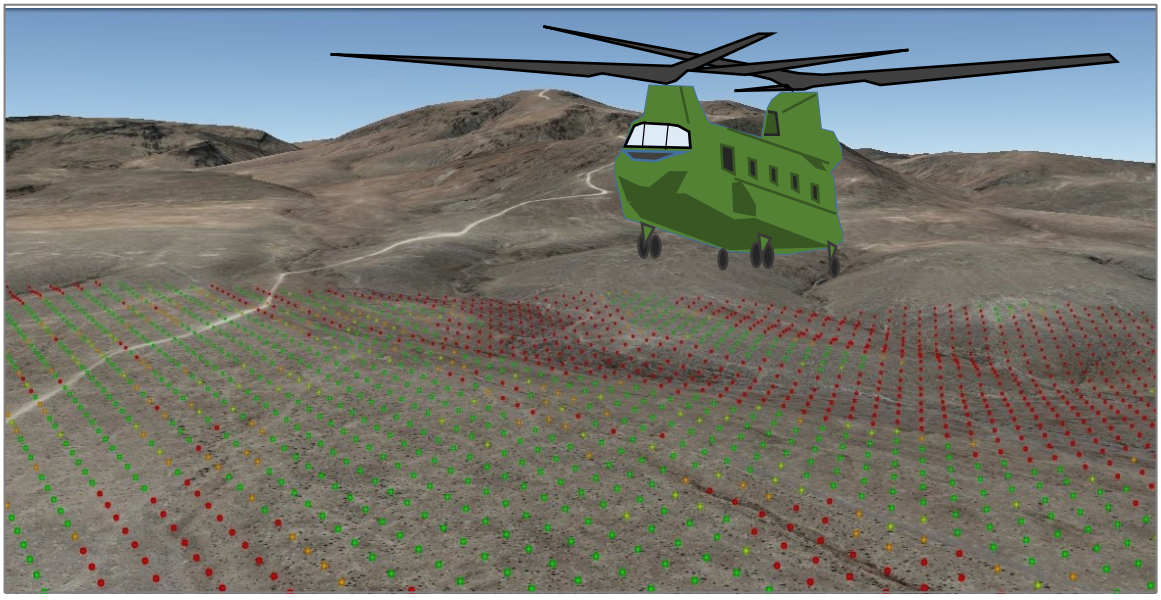
ERDC ENGINEER RESEARCH & DEVELOPMENT CENTER

Entry and Sustainment in Complex, Contested Environments

Incorporating Terrain Roughness into Helicopter Landing Zone Site Selection by Using the Geomorphic Oscillation Assessment Tool (GOAT) v1.0

Michael T. Ekegren and Sandra L. LeGrand

September 2021



Cold Regions Research and Engineering Laboratory

The U.S. Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at https://erdclibrary.on.worldcat.org/discovery.

ERDC/CRREL TR-21-11
September 2021

Incorporating Terrain Roughness into Helicopter Landing Zone Site Selection by Using the Geomorphic Oscillation Assessment Tool (GOAT) v1.0

Michael T. Ekegren and Sandra L. LeGrand

U.S. Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

Final Report

Prepared for Headquarters, U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under PE 0603119A, Project BL8, Task 01, "Entry and Sustainment in Complex, Contested Environments"

Abstract

The Geomorphic Oscillation Assessment Tool (GOAT) quantifies terrain roughness as a mechanism to better explain forward arming and refueling point (FARP) suitability for Army aviation. An empirically driven characteristic of FARP consideration, surface roughness is a key discriminator for site utility in complex terrain. GOAT uses a spatial sampling of high-resolution elevation and land cover data to construct data frames, which enable a relational analysis of component and aggregate site suitability. By incorporating multiple criteria from various doctrinal sources, GOAT produces a composite quality assessment of the areal options available to the aviation commander. This report documents and demonstrates version 1.0 of the GOAT algorithms developed by the U.S. Army Engineer Research and Development Center (ERDC). These details will allow users familiar with R to implement it as a stand-alone program or in R Studio.

Contents

Figures and Tables

Figures

Tables

Preface

This study was conducted for the U.S. Army Corps of Engineers under PE 0603119A, Project BL8, Task 01, "Entry and Sustainment in Complex, Contested Environments."

The work was performed by the Signature Physics Branch and the Terrestrial and Cryospheric Sciences Branch of the Research and Engineering Division, U.S. Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory (ERDC-CRREL). At the time of publication, Dr. Steven Peckham was Acting Chief of the Signature Physics Branch, Dr. John Weatherly was Chief of the Terrestrial and Cryospheric Sciences Branch, and Dr. George Calfas was Division Chief. The Deputy Director of ERDC-CRREL was Mr. David B. Ringelberg, and the Director was Dr. Joseph L. Corriveau.

The authors wish to thank Mr. Justin Putnam, CPT Eoghan Matthews, Dr. John Rushing, Mr. Kyle Elliot, Mr. Anthony Fuentes, Mr. Taylor Hodgdon, and Ms. Samantha Cook for assistance with fieldwork, mapping, and product demonstrations; SFC Ysidro Arredondo, WO1 Todd Facello, the Eagle Team, and the Army National Training Center for inspiring and enabling this research; and Mr. Kyle Elliot and Dr. Sally Shoop for thoughtful discussions, input, and editing.

COL Teresa A. Schlosser was Commander of ERDC, and Dr. David W. Pittman was the Director.

Acronyms and Abbreviations

AGC	U.S. Army Geospatial Center
CRREL	Cold Regions Research and Engineering Laboratory
CSV	Comma Separated Value
DSM	Digital Surface Model
DTM	Digital Terrain Model
DTML	Digital Terrain Model Lagged
ERDC	Engineer Research Development Center
FARP	Forward Arming and Refueling Point
FLOT	Forward Line of Troops
GCS	Geographic Coordinate System
GOAT	Geomorphic Oscillation Assessment Tool
HLZ	Helicopter Landing Zone
lon/lat	Longitude/Latitude
NLCD	National Land Cover Data
NTC	National Training Center
SME	Subject Matter Expert
UTM	Universal Transverse Mercator
WGS84	World Geodetic System 1984

1 Introduction

1.1 Background

A Forward Arming and Refueling Point (FARP) is a temporary location for resupplying rotorcraft with fuel and ordnance closer to their objective location than established permanent landing sites (e.g., U.S. Army 2016). According to the Army Training Publication 3-04.17, *Techniques for Forward Arming and Refueling Points* (U.S. Army 2018), aviation mission success is directly related to the effectiveness of the FARP. Strategically selected FARP locations enable highly mobile and flexible aviation operations, allowing aircraft to spend more time near the front lines and carry loads deeper into contested areas, ultimately increasing the warfighters' operational reach.

FARPs often become high-priority targets in battle since preventing aircraft from arming and refueling can effectively neutralize an aviation force. Thus, planners aim to select FARP locations close to the area of contention, or Forward Line of Own Troops (commonly referred to as the FLOT), that avoid excessive exposure to enemy attacks. The ability to identify and designate viable FARP locations as the battle conditions evolve is critical for force projection; however, currently available Army doctrinal publications for FARP site-selection criteria are rather vague.

Existing aviation resources list several environmental considerations for FARP site selection, such as proximity to terrain folds for concealment and avoiding hazards like snow and dust, but provide few specifics regarding desirable or suitable conditions (e.g., U.S. Army 2018). Official training manuals offer some quantitative recommendations stemming from general physical rotary-wing aircraft restrictions, but even these are interpretable. The most definitive terrain-suitability guidance comes from the Pathfinder Field Manual 3-21.38 (U.S. Army 2006) and the *101st Airborne Division (Air Assault) Gold Book* (U.S. Army 2019), a reference of the doctrine, organization, and capabilities of the 101st Airborne Division (referred to as the *Gold Book* henceforth). These resources suggest that safe helicopter landing zone (HLZ) terrain slopes should not exceed 7 degrees during daytime operations and that terrain slopes exceeding 15 degrees prohibit HLZ use entirely. The *Gold Book* further recommends that nighttime HLZ operations should avoid terrain slopes exceeding 3 degrees.

Also, UH-60 aircraft (i.e., Black Hawk front-line utility helicopters) and CH-47 aircraft (i.e., Chinook tandem-rotor, heavy-lift helicopters) should land at least 30 m* and 35 m from obstructions, respectively. For a comprehensive aggregation of technical doctrine on landing zones, refer to Wieder and Shoop (2017).

The general lack of explicit criteria and apparent dissatisfaction from Army Aviation in existing HLZ and FARP site-selection tools motivated us to conduct "ride-along" fact-finding tours with FARP operation subject matter experts (SMEs) in a tactical training environment. During our first field observation at the National Training Center (NTC) at Fort Irwin, California, we uncovered an undocumented HLZ requirement known to expert ground crews but otherwise ignored in the initial FARP site-selection planning process: surface roughness. Surface roughness is a dominant factor in most of the sites that are unacceptable for use in FARP operations at NTC.

The geomorphic process driving this roughness is fluvial erosion. Much of the landscape at NTC is either an alluvial fan or bajada, which is the confluence of alluvial fans. Alluvial fans are a depositional landform, built up where the water exits mountains and rapidly undergoes a loss of energy, which prompts the dropping of carried eroded material. While alluvial fans tend to have low slopes due to their depositional nature, sequential water flows follow different pathways and create complex drainage features. Aggregated small undulations in the terrain create undesirable conditions for helicopter landing gear and are challenging to detect with existing techniques. In response to this shortfall, we developed the Geomorphic Oscillation Assessment Tool (GOAT) to incorporate surface roughness into terrain-suitability assessment tools used in the FARP site-selection process.

1.2 Objective

The purpose of this report is to present an overview of the GOAT terrain-suitability assessment algorithm.

* For a full list of the spelled-out forms of the units of measure used in this document, please refer to *U.S. Government Publishing Office Style Manual*, 31st ed. (Washington, DC: U.S Government Publishing Office, 2016), 248–252, https://www.govinfo.gov/content/pkg/GPO-STYLEMANUAL-2016/pdf/GPO-STYLEMANUAL-2016.pdf.

Our objectives are twofold:

1. To document the functions that make up the site-selection algorithm
2. To provide a user's guide for code utilization

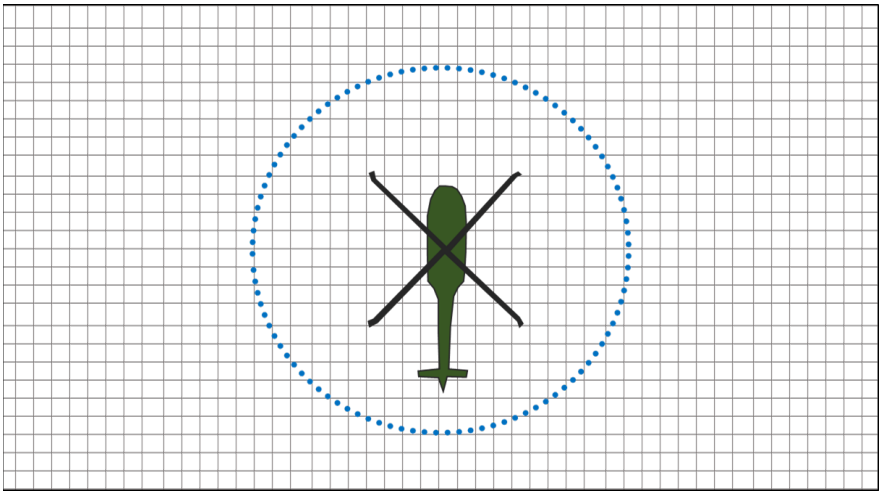
1.3 Approach

GOAT v1.0 uses 1 m grid-resolution data and terrain-suitability thresholds generated from extensive interviews with aviators and FARP SMEs. Here, we demonstrate how to run the software and how to interpret results by using the data from a subset of the training range at NTC. In section 2, we introduce the GOAT concept and provide an overview of important terminology used in the algorithm description. Section 3 describes the required input datasets, and section 4 provides example output files for the NTC area. Finally, section 5 details additional software development and research needed to continue enhancing FARP site-selection efforts.

2 General Overview and Terminology

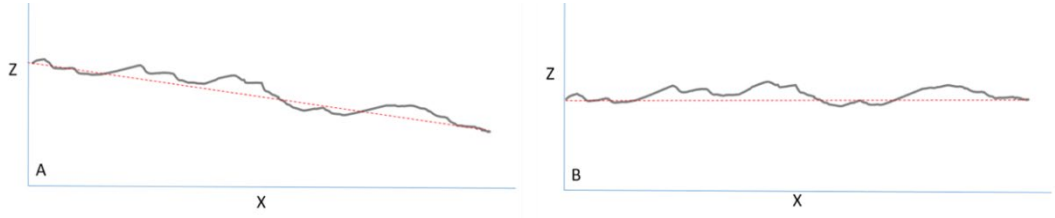
GOAT uses a circular sampling technique to assess the general roughness of a point's surroundings (e.g., Figure 1). A user can modify the circle's radius to suit their particular mission needs as long as the radius is greater than or equal to the required physical footprint of the helicopter (i.e., the area encompassed by the rotor disk and the tail). By assessing the variability in elevation from adjacent points that make up the circle, the tool can draw inferences about the general roughness of the landing zone. This circular sampling approach increases the likelihood that linear features, such as drainages, in the proximity of a potential area of interest are considered. For example, a small channel may not pass directly over a point of interest but is influential to the site-suitability analysis if it is immediately adjacent to the potential HLZ location.

Figure 1. Circular sampling around an aircraft; image intended as a conceptual diagram only and is not to scale.



Surface roughness, also sometimes referred to as surface texture, generally refers to variations in the elevation of a terrain surface relative to a reference plane (e.g., Figure 2). When diagnosing surface-roughness conditions in GOAT, we also include functions to remove the influence of the reference plane slope from the calculation. In this report, we refer to this conceptual reference plane slope as the "macroslope" of the terrain to differentiate it from the "microslope" features of the roughness elements. Note, the HLZ slope limitations outlined in the *Gold Book* cited previously describe macroslope guidance.

Figure 2. Schematic of (*A*) a terrain surface profile, Z(X), and (*B*) the Z(X) profile adjusted to remove the slope of a reference plane. The *solid gray line* represents the terrain profile Z(X), and the *dotted red line* represents the reference plane.



We developed GOAT v1.0 using R v3.5.1 with specialized function library extensions (see Table 1). This section provides a brief overview of important R-related terminology relative to this report to assist readers who may not have extensive experience with the scripting language. R has six basic data types, including

- characters (e.g., text strings like “abc123”),
- numeric (real or decimal numbers),
- integer,
- logical (e.g., TRUE or FALSE),
- complex (i.e., complex numbers with real and imaginary parts, such as 1 + 5i), and
- raw byte data.

R also includes several data structures or means of organizing data. The three primary forms used in GOAT include the atomic vector, the matrix, and the data frame.

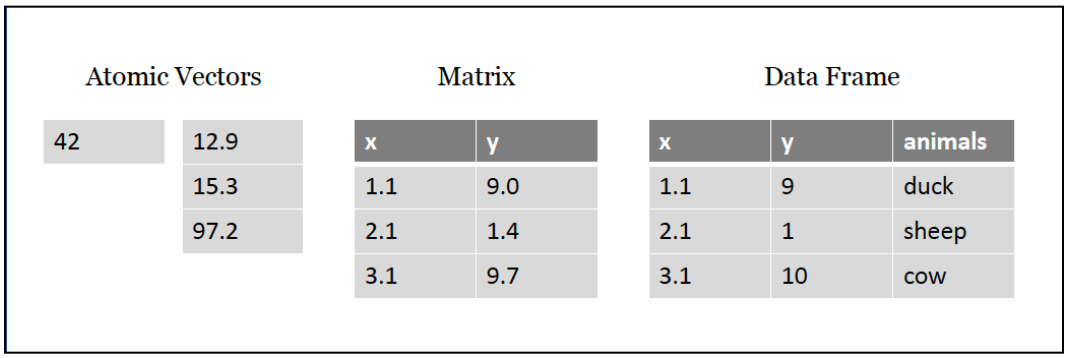
Table 1. Required R library extensions.

Library name	Reason for use in GOAT script
raster	Reads gridded datasets and estimates distance between points
rgdal	Geospatial binding necessary to georeference and project gridded data
geosphere	Performs distance segmentation and bearing calculations
dplyr	Performs lag function to offset column values to the next row
ggplot2	Used to support plotting results
RColorBrewer	Used to support plotting visuals

In R-scripting language, two-dimensional atomic vectors are called matrices (i.e., atomic vectors with rows and columns). As with atomic vectors, all of the elements contained in a matrix must be of the same data type (e.g., all numeric or all integer). Data frames, like matrices, have rows and columns but enable storage of different variable types. For example, a data

frame can include columns representing logical, character, and numeric values; however, all values associated with a particular column must be of consistent type (e.g., Figure 3).

Figure 3. R data structures used in the GOAT script.



For this report, we have adopted a naming convention to help differentiate matrices used in our code. The term *matrix* with a lowercase *m* refers to a two-dimensional matrix dataset. *Matrix* with a capital *M* refers to the two-dimensional array of independent geospatial points (i.e., a class of georeferenced point data in R) associated with the primary analysis domain. Though gridded, the Matrix points are not stored as relative positions to each other as a raster would be, which allows for variable orientation or modifiable spacing if desired. The matrices designated with a lowercase *m* may contain data representing spatial points, but only the Matrix referring to the main analysis grid is capitalized. These naming conventions, however, are unique to this report and should not be considered community standards.

3 Required Input Data

GOAT v1.0 requires three input datasets, including two elevation-based datasets derived from high-precision lidar data and a land cover dataset. In its current configuration, all input fields must be in GeoTIFF format and projected in Geographic Coordinate System World Geodetic System 1984 (GCS WGS84) coordinates. This conversion to GCS WGS84 is necessary to enable the use of R libraries that require decimal-degree coordinates.

3.1 Terrain elevation

We originally designed and evaluated GOAT using 1 m resolution elevation data derived from the U.S. Army Geospatial Center (AGC) Buckeye lidar system (AGC 2013). The Buckeye program collected over 280,000 km² of lidar data in Iraq and Afghanistan, as well as at several U.S. military bases. The overview provided in this report assumes that users have access to Buckeye data sources for their location of interest; however, users can also opt for other high-fidelity data sources to run the code as long as the resolution and vertical precision are sufficient.

GOAT utilizes two elevation datasets processed from Buckeye. The first is a digital surface model (DSM), which is based on the first lidar return, processed to represent the maximum elevation point within a cell regardless of its source. The second dataset is the digital terrain model (DTM), which is a bare-earth model processed to remove all vegetation and structures from the elevation field. Most of the Buckeye data are processed and distributed at 1 m grid spacing; however, some areas have finer resolution. The DTM and DSM datasets are aligned in extent and cell placement and stored as 32 bit float values. For this example, we downloaded each dataset from the AGC Common Map Background Online Interactive Ordering portal (AGC, n.d.) as a series of GeoTIFF tiles projected in Universal Transverse Mercator (UTM) format. The tiles from each layer were mosaicked (i.e., merged) into a single GeoTIFF and output in GCS WGS84 using the "Mosaic to New Raster" tool in ArcGIS v10.4.1.

Users can choose to obtain their DTM and DSM input data from various lidar and stereo-optical sources with similar resolution. The essential nuance is that the two datasets originate from the same collection and have relatively high vertical precision. Both parameters must also be in units of meters.

3.2 Land cover classification

GOAT also requires a land cover input dataset. Although the primary purpose of the code is to assess roughness risk, incorporating land cover helps the tool avoid returning positive results on "flat" water surfaces. Of note, land cover datasets may not incorporate ephemeral lakes or streambeds well. While analysts can usually identify these particular landscape features in topographic maps and imagery, these landforms can be a challenge to represent digitally.

We developed and evaluated GOAT using 30 m National Land Cover Data (NLCD; Wickham et al. 2014; Multi-Resolution Land Characteristics Consortium 2016) downloaded from the Multi-Resolution Land Characteristics Consortium as a single tile. As with the elevation data, we reprojected the NLCD data from their original Albert's Equal Area Conical Projection to GCS WGS84 using ArcGIS v10.4.1.

4 Algorithm Overview

Functions that make up the GOAT code generally fall into one of two categories. Code blocks either identify spatial points that compose the domain Matrix or support terrain attribute calculations relative to a given location. Section 4.1 provides a general overview of the technical approach, and section 4.2 walks through the actual execution of the code.

4.1 Technical approach

4.1.1 Construction of the domain Matrix

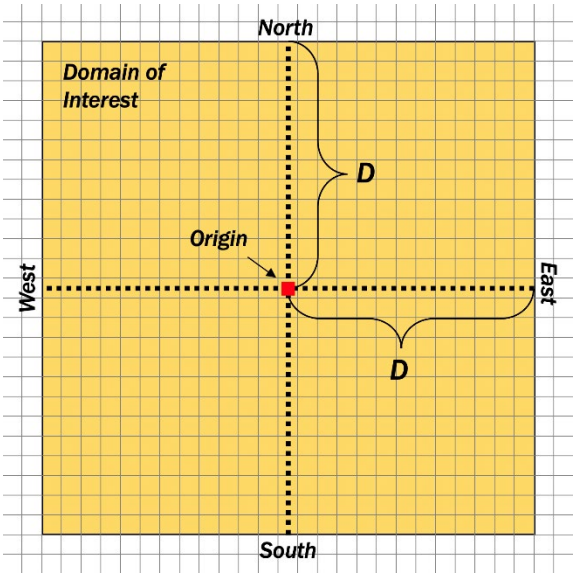
First, GOAT generates a spatial point Matrix for the user's domain of interest by using two user-provided parameters: (1) the location of an origin point and (2) the horizontal distance (in the four cardinal directions) away from the origin point considered for the analysis (e.g., Figure 4). The origin-point argument must be expressed using longitude and latitude (referred to henceforth as lon/lat pairs) in decimal degrees, generally with four-decimal precision. The distance argument should be in meters and is generally an integer on the order of several hundred. GOAT uses these arguments to create a square-shaped grid stored in matrix format with the top edge oriented from west to east. Although the Matrix sampling functions do not explicitly require creating a square oriented in cardinal directions, we chose to limit this version of GOAT to reduce user input complexity.

GOAT also includes a prescribed parameter to set the fixed spacing of points in the domain Matrix. Note, this value is independent of the resolution of the input terrain datasets. For example, a 400 m × 400 m domain Matrix with the 25 m sampling parameter applied includes 256 spatial points rather than the 160,000 points that would exist if the analysis considered every grid cell in the 1 m elevation input datasets. This parameter is tunable depending on user needs, but the value applied should be less than the prescribed sampling radius of the circular sampling function discussed later in section 4.1.2.

Next, GOAT creates a master data frame with one row for each point generated in the domain Matrix and two variable columns: Lon and Lat. These columns are populated using the lon/lat pairs from the domain Matrix. The script then appends columns containing the DTM, DSM, and land cover values from the input data associated with each lon/lat pair to the

master data frame. The DTM and DSM values retain their original numeric form. NLCD data convert to binary integer values, where 1 represents a potentially desirable land cover feature for landing, and 0 represents water hazards. Throughout the script, lon/lat parings are used to associate rows from the master data frame to points in the domain Matrix. GOAT defines the elevation variables contained in the input datasets, as well as a prescribed sampling radius (*r*; in meters), as global attributes accessible to all script functions.

Figure 4. GOAT analysis domain example. Here, *D* represents the user-provided distance from the origin point of interest. GOAT v1.0 builds the analysis domain based on D and the origin-point location. As a result, the default analysis domain is square-shaped with the top edge oriented from west to east.



4.1.2 The analytical point function

Once the domain Matrix build and the data ingest processes are complete, the script runs an analytical function on each point contained in the domain Matrix (referred to in this section as the point function). Note, the script pulls elevation and land cover values directly from the lon/lat pairs in the domain Matrix rather than aligning to the cell centers of the input data layers. In other words, the domain Matrix and all sample points are independent of the alignment of the input GeoTIFFs. This feature of the code enables GOAT to take advantage of the high-fidelity input data without overly redundant processing such as linear interpolation.

The key aspect of determining surface roughness with GOAT is to aggregate the variance of elevation change. This is done with accumulation of sequential changes among a series of radial points that make up a circle around each point in the domain Matrix (e.g., Figure 5). The rest of this subsection describes processes that occur within the point function by referencing a hypothetical point *p* surrounded by a circle of radial points. All calculations taking place in this function are relative to the location of point *p* or the collection of radial points surrounding *p*. The GOAT script independently applies this point function to every point included in the domain Matrix (e.g., the red diamonds in Figure 6). Of note, this code structure lends itself well to future code optimization efforts if processing speed is of concern.

First, the point function identifies a circular sample of 144 corresponding radial points for a given domain point (*p*), starting with the radial point due north of *p* at *r* distance and moving clockwise in 2.5 degree increments (e.g., Figure 5). A temporary data frame stores the 144 radial points along with their associated DTM and DSM values. We set the default *r* value in GOAT to 35 m per guidance from the *Gold Book*, which states that Chinook helicopters (the largest rotorcraft currently used in Army operations) should land at least 35 m from a tree line (meaning any tall obstruction element, not just trees). By using a 35 m prescribed radius in conjunction with the 2.5-degree spacing, the spatial separation between adjacent radial-point samples on the circle is about 1.52 m in length. Given that the 1 m resolution elevation raster grid cells are approximately 1.41 m long in the diagonal direction, the circular sampling method is an adequate means of discerning a large number of relevant points without risking multiple radial point lon/lat pair samples occurring in the same input cell.

Because the 35 m sampling radius exceeds more than half of the distance between individual domain Matrix points, the process produces overlap in the radial-point sampling rings associated with adjacent domain Matrix points. The overlap reduces the independence between the surface-roughness analyses of near-proximity points (e.g., Figure 6). However, this inherent overlap helps smooth the final output of the roughness-hazard analysis, which in turn can be beneficial to end-user analysts since FARP operation footprints usually extend beyond space considerations needed for just the aircraft.

Figure 5. Circular sample using 144 radial points, where *p* represents a point from the domain Matrix and *r* is the sampling radius.

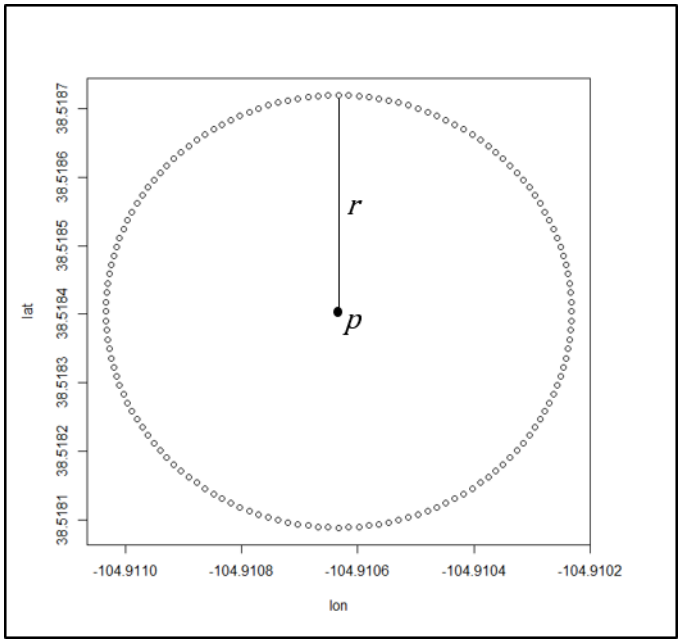
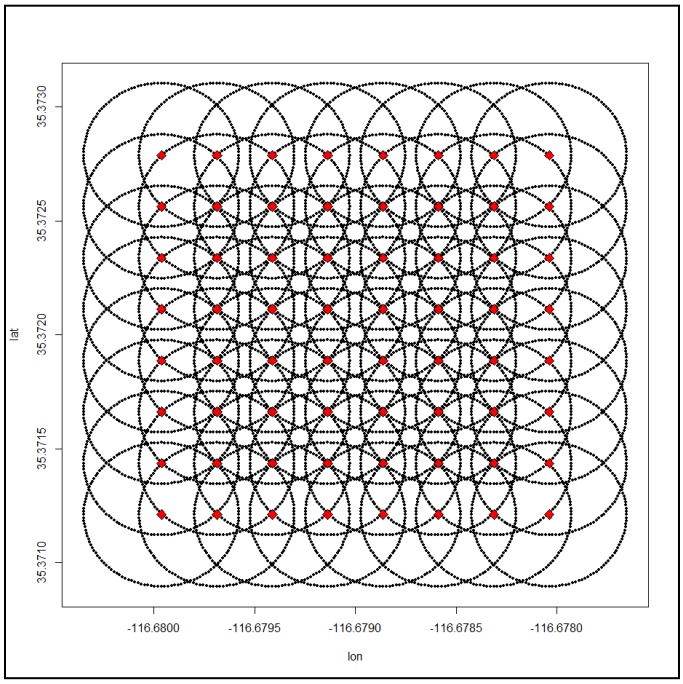


Figure 6. Overlap of radial sample points (*black*) for a hypothetical 8 × 8 domain Matrix (*red diamonds*).



The DTM values in the temporary data frame representing the circle of radial points affiliated with a given point *p* are offset downward by one row and stored as a new variable column, DTML (i.e., DTM Lagged). The DTM

value from the last row in the data frame becomes the value in the first row of DTML, accordingly. Effectively, this process associates the DTM value of a given radial point with the DTM value of an adjacent radial point in the clockwise direction (e.g., Table 2). Next, the point function determines the vertical difference between the DTM and DTML values on each row of the temporary data frame to calculate the elevation change (meters) between adjacent radial points (listed in the Table 2 as ZDelta). The sum of the 144 resultant ZDelta absolute values, R_{op}, provides an initial estimate for the general roughness surrounding domain Matrix point p; however, the influence of the macroslope must still be removed. This adjustment occurs in the main body of the GOAT script, outside of the point function.

The point function continues by calculating the elevation angle (DTM_SLP in Table 2) between the elevation of point p ($z_{DTM,p}$; established from the DTM input dataset) and the DTM value of each radial point in the sampling circle in order to find the steepest elevation angle (referred to as the steepest slope) for the radial-point collection, $\theta_{stp,p}$. Note, the code determines $\theta_{stp,p}$ based on the absolute values of elevation angles from the DTM_SLP column. The function then diagnoses the elevation angle between the radial point associated with $\theta_{stp,p}$ and the radial point directly opposite (180 degrees) from its location. We refer to the resultant value as the influential slope of point p, $\theta_{in,p}$, which GOAT uses later to assess the general consistency of the surface features surrounding p. Next, the point function calculates the average of the DTM_SLP column, $\theta_{avg,p}$, to assess how planar the terrain surface around p is. Section 5 further discusses the influential slope and planar surface aspects.

The point function then adds the DSM values associated with each radial point to the temporary data frame and uses the difference between DSM and DTM values of each respective radial point to identify vertical obstacles taller than 3 m around the circle. If the function identifies any differences greater than 3 m, a binary parameter representing tree line risk for point p, T_p, is set to zero; otherwise, T_p assumes a value of 1. While the use of 3 m is entirely subjective and not driven by Army Doctrine, we based this value on guidance from SMEs at NTC.

The last diagnostic performed in the point function involves in-flight considerations. It determines the angle of climb needed to fly over surrounding obstacles (CLIMB in Table 2) by identifying the elevation angle needed to reach the DSM value of each radial point from $z_{DTM,p}$. Similar to the $\theta_{stp,p}$

calculation, the point function determines the value with the largest magnitude from the CLIMB column to describe the maximum climb (in decimal degrees) for point p, $\theta_{C,p}$. However, unlike the $\theta_{stp,p}$ determination, the θ_{Cp} value is based on the maximum slope value and not the largest absolute value. In other words, the consideration of climb is for upward-sloping features around a point and not downward-sloping conditions. Of note, $\theta_{stp,p}$ is of concern for helicopter landings with firm ground contact, but θ_{Cp} is relevant when aircraft maintain power (e.g., hover) and only need to consider angles above horizontal.

Table 2. Example of a temporary data frame for the sampling and calculation of a hypothetical point *p*. This table has less than 144 rows because it is only intended as a conceptual illustration. The DTM value at *p* is 1060.8278 m, which is the base of the DTM_SLP and CLIMB slope calculations; the DTM and DSM values in this table are affiliated with the radial sample points around the circle.

Lon (deg)	Lat (deg)	DTM (m)	DTML (m)	ZDelta (m)	DTM_SLP (deg)	DSM (m)	CLIMB (deg)
−116.79720	39.39092	1059.0319	1059.0864	−0.0545	−2.9373	1059.0227	−2.9524
−116.79718	39.39092	1058.9792	1059.0319	−0.0527	−3.0234	1058.9798	−3.0224
−116.79716	39.39092	1058.8774	1058.9792	−0.1018	−3.1896	1058.9293	−3.1049
−116.79715	39.39092	1058.9255	1058.8774	0.0481	−3.1111	1058.8581	−3.2210
−116.79713	39.39092	1058.8441	1058.9255	−0.0814	−3.2439	1058.8439	−3.2441
−116.79711	39.39091	1058.7177	1058.8441	−0.1264	−3.4501	1058.7177	−3.4501
−116.79710	39.39091	1058.6059	1058.7177	−0.1118	−3.6324	1058.6507	−3.5594

Finally, the point function returns and appends values for R_{op}, $\theta_{stp,p}$, $\theta_{in,p}$, $\theta_{avg,p}$, T_p, and $\theta_{C,p}$ for point p to the master data frame and continues on to evaluate the attributes of the next point in the domain Matrix.

4.1.3 Diagnosing terrain roughness and HLZ site suitability

Once the point function finishes processing all of the points in the domain Matrix and the six new columns in the master data frame are complete, GOAT performs a surface-roughness evaluation for each row in the master data frame and appends the estimated surface roughness, R_p, for each row to a new column.

The first step in determining R_p involves a conditional check to ensure that $\theta_{stp,p}$ is less than or equal to the user-provided slope threshold (θ_t), which should be no more than 7 degrees based on doctrine. In instances where $\theta_{stp,p}$ exceeds this argument, the script assigns R_p a value of K_{ooc}, an

arbitrary value well above the range of expected results for use in downstream visualization functions to show values out of the range of consideration. If $\theta_{stp,p}$ is at or below θ_t, the script determines R_p (in terms of meters), following

$$R_p = \begin{cases} R_{0p} - C_R\left(\theta_{in,p} - \left|\theta_{avg,p}\right|\right), \theta_{stp,p} \le \theta_t \\ K_{\text{ooc}},\ \theta_{stp,p} > \theta_t, \end{cases} \quad (1)$$

where C_R is an empirical constant set to 2.447438 m deg^{-1} based on the per-degree average of initial roughness summation when tested on artificial planar evaluation datasets up to 7 degrees. The purpose of the C_R scaling factor is to account for the roughness influence that we would expect to exist if the terrain were truly planar. In effect, equation (1) removes the influence of macroslope, slope direction, and larger-scale terrain shape from R_{op}.

After the surface-roughness calculation is complete, GOAT uses additional conditional statements to establish the broader HLZ suitability of each domain Matrix point ($S_{\text{HLZ},p}$). The script sets the R_p value at points identified as having an undesirable land cover (i.e., NLCD binary value [$L_{c,p}$] equals 1), a perimeter obstacle (T_p), or a difference between the point DSM ($z_{\text{DSM},p}$) and $z_{\text{DTM},p}$ exceeding a meter or more to K_{ooc}. The last conditional test is a means to identify potential obstructions at the landing site.

$$S_{\text{HLZ},p} = \begin{cases} R_p,\ L_{\text{c},p} = 0\ \&\ T_p = 0\ \&\ (z_{\text{DSM},p} - z_{\text{DTM},p}) > 1\text{m} \\ K_{\text{ooc}},\ \text{otherwise} \end{cases}. \quad (2)$$

Although there is currently no official guidance on what constitutes a desirable roughness threshold for HLZ suitability, discussion with NTC SMEs suggests conditions with $R_p \le 1.5$ m are ideal for landing, and areas with $R_p > 3.5$ m are unusable. However, further studies are necessary to determine robust recommendations for R_p limits on $S_{\text{HLZ},p}$.

4.2 Code execution

This section provides an in-depth overview of the GOAT code and script execution procedures. The initial GOAT code release is a relatively simple R-based script that can be run for a limited area of interest (e.g., an Army installation site) without the use of high-performance computing resources. When appropriate, we also specify variable data types and their respective units using the following format: (type, units). To run the GOAT

script, a user must first set their working directory and geospatial input data paths. All output datasets generated by the script automatically write to the working directory. Note that the syntax for the file paths shown here uses double backslashes to comply with the format used by the Windows operating system.

```
setwd("C:\\.....") # working directory path
dtm_path <- "C:\\.....DTM_All.tif"    # DTM GeoTiff path
dsm_path <- "C:\\.....DSM_All.tif"    # DSM GeoTiff path
landcover_path <- "C:\\.....NLCD.tif" # NLCD GeoTiff path
```

The user must also set parameter inputs for the central domain origin-point longitude (`BASE_LON`: numeric, decimal degrees) and latitude (`BASE_LAT`: numeric, decimal degrees) and a distance (`subDist`: integer, m) to establish the square-shaped analysis domain. The example below creates an 800 × 800 m square centered on the NTC range.

```
BASE_LON <- -116.7928  # central domain point lon (deg)
BASE_LAT <- 35.3870    # central domain point lat (deg)
subDist <- 400         # cardinal direction distance (m)
```

Additional parameters the user can opt to modify include the domain Matrix sampling resolution (`sampRes`: numeric, m), the circular sampling radius (`sampRad`: numeric, m; r from section 4.1), and the maximum macroslope acceptability (`slopeAcc`: numeric, m; θ_t from section 4.1). As previously mentioned in section 4.1, the domain Matrix sampling resolution should be less than the circular sampling radius.

```
sampRes <- 25  # domain Matrix sampling resolution (m)
sampRad <- 35  # circular sampling radius (m)
slopeAcc <- 7  # maximum macroslope threshold (deg)
```

Other constants that do not require runtime modification include the circular sampling interval (`sampInt`: numeric, decimal degree), the value used to indicate a nonsuitable HLZ site (`OOC`: numeric, unitless; K_{ooc} from section 4.1), and a constant very small number (`vsn`) for use in data-cropping functions.

```
sampInt <- 2.5      # circular sampling interval (deg)
OOC <- 2000         # out of consideration value
vsn <- 0.001        # very small number
```

The script also requires some additional land cover information, including the total number of land cover classes used (`nLCF`: integer, unitless) and the hazardous land cover class identification codes (`LCF_hazard`: integer, unitless). Although we developed and tested GOAT v1.0 using NLCD data, we chose to make this part of the code configurable so users could easily incorporate alternate data sources if needed. Land cover datasets, including NLCD, often use numbering schemes as classification identifiers. These numbers generally serve as codes associated with "lookup" tables containing additional attributes about each land cover class. The NLCD dataset we used to design GOAT includes 95 classes. Classes 11, 90, and 95 represent water, wetlands, and emergent herbaceous wetlands, respectively.

```
nLCF <- 95                # total number of land cover classes
LCF_hazard <- c(11,90,95) # water-related land cover classes
```

Before moving forward with data processing, GOAT sets up an integer vector called `LCF`, which the script uses later to convert the land cover data from its original index format (i.e., values ranging from 1 to 95) to a binary filter of zeros and ones. At first, the script establishes `LCF` as a vector of length `nLCF`, populated with a sequence of repeating ones. A separate function then reassigns the `LCF` vector positions to zero if they match the values identified in the `LCF_hazard` vector.

```
LCF <- c(rep(1,nLCF))
LCF[LCF_hazard] <- 0
```

The script also sets a vector called `sampDir` (numeric, decimal degrees), which is a sequence of directional angles used to select radial sampling points around a circle at intervals equal to `sampInt`. The script utilizes this vector later during the execution of the point function. Note that although the `sampInt` value can be adjusted, we conducted all GOAT software development and testing using a sampling interval of 2.5 degrees, which produces 144 radial points in the circle.

```
sampDir <- c(seq(0,359, sampInt))
```

Once the prescribed constants are set, the script calls the required libraries detailed in Table 1. Only the first four libraries listed below are required for the actual analysis. The RColorBrewer and ggplot2 libraries are exclusively for visualization and plotting purposes.

```
library(raster)
library(rgdal)
library(geosphere)
library(dplyr)
library(ggplot2)
library(RColorBrewer)
```

GOAT uses the raster function to read the DTM, DSM, and NLCD land cover datasets as RasterLayer objects. The code refers to these input parameters as `dtm` (numeric, m), `dsm` (numeric, m), and `landcover` (integer, unitless), respectively. Following this approach enables R to ingest the input data in two-dimensional matrix format and retain essential geospatial attributes like grid resolution, spatial extent, and any relevant coordinate references.

```
dtm <- raster(dtm_path)
dsm <- raster(dsm_path)
landcover <- raster(landcover_path)
```

Next, the script moves on to setting up the domain placement and extent by defining the origin point (`basePoint`) and the location of the northeast and northwest corners.

```
basePoint <- matrix(c( BASE_LON, BASE_LAT ), nrow=1)
northeast <- destPoint(basePoint, 45, sqrt((subDist^2)+(subDist^2)))
northwest <- destPoint(basePoint, 315, sqrt((subDist^2)+(subDist^2)))
```

The domain Matrix stems from two arguments: the FARP zone frontage (`fZf`) and FARP zone depth (`fZd`). The application in this report fixes the frontage and depth to create a cardinally aligned square domain, though the shape is easily modifiable. The `fZf` includes two lon/lat pairs representing the forward edge of the sampling zone, listed in this usage as the northwest and northeast corners. As such, the orientation of the grid is due north. The `fZd` (integer, m) is the horizontal depth of the FARP zone; in this usage it represents the north–south length of the zone. The use of the `subDist` variable fixes the depth to match the frontage length, generating a square-shaped sample.

```
fZf <- matrix(c(northwest, northeast), nrow=2)
fZd <- subDist*2
```

The script further modifies the domain bounds using the `bearing` function from the geosphere library and the `pointDistance` function from the raster library to establish the FARP zone frontage's bearing (`fZfBearing`: numeric, decimal degrees) and distance (`fZfDist`: numeric, m), respectively. The script divides the `fZfDist` value by the `sampRes` with a `floor` function applied to preserve the value as a whole number. In this code block, the script denotes the resultant parameter as `fZSamp` (numeric, m). By using the `gcIntermediate` function from the geosphere library, the code then generates an evenly distributed sample of lon/lat pairs along the frontage in a matrix called `fZfPoint`. Essentially, this step establishes the first row of domain Matrix point locations between the northwest and northeast corners.

```
fZfBearing <- bearing(fZf[,1], fZf[,2])
fZfDist <- pointDistance(fZf[,1], fZf[,2], lonlat=TRUE)
fZSamp <- floor(fZfDist/sampRes)
fZfPoint <- gcIntermediate(fZf[,1], fZf[,2], fZSamp,
addStartEnd=TRUE, sp=FALSE, sepNA=FALSE)
```

Next, the script calculates the FARP zone depth's bearing (`fZdBearing`) by adding 90 degrees to `fZfBearing` with a check to adjust the value down by 360 degrees if the resultant value surpasses due north. Technically, this conditional test and adjustment are unnecessary in this version of GOAT because the frontage is always assumed to be due east at 90 degrees and `fZdBearing` should always be 180 degrees. We chose to include it in this code release to simplify future code modifications that allow for rotated domains.

```
fZdBearing <- fZfBearing + 90
fZdBearing <- ifelse(fZdBearing > 359, fZdBearing - 360, fZdBearing)
```

The `destPoint` function from the geosphere library generates lon/lat pairs along the domain boundary opposite of the frontage (i.e., the southern domain edge) corresponding to the distribution and number of lon/lat pairs in the `fZfPoint` matrix. These values are stored in a matrix called `fZdRear` and run parallel to the values from `fzfPoint`.

```
fZdRear <- destPoint(fZfPoint, fZdBearing, fZd)
```

By using the `fZfPoint` and `fzfRear` matrices, the script can fill in the rest of the lon/lat locations of points that make up the domain Matrix

(sampPoints) needed for the point function, roughness, and final site-suitability analyses. The script first creates an empty matrix, then cycles through each lon/lat pair in fZfPoint to generate a matrix of lon/lat pairs for equally spaced points (based on fZSamp) between the fZfPoint lon/lat values under consideration and its corresponding opposite point location in fzfRear. The new collection of lon/lat pairs associated with the loop cycle append to sampPoints, and the loop moves on to the next lon/lat pair in fZfPoint. Once the loop completes, the script removes the first row of sampPoints because it populates with missing values when the script initially defines the empty matrix shell.

```
sampPoints <- matrix(ncol=2)
for(i in 1:nrow(fZfPoint)){
      newPoints <- gcIntermediate(fZfPoint[i,], fZdRear[i,],
                                  fZSamp, addStartEnd=TRUE,
                                  sp=FALSE, sepNA=FALSE)
      sampPoints <- rbind(sampPoints, newPoints)}
sampPoints <- sampPoints[-1,]
```

The next set of functions crop the input data to reduce the size of datasets held in memory. R 3.5.1 has a limited object size of 1.4 Gb, which can be an issue when processing large, high-fidelity lidar and land cover datasets. To reduce the likelihood of memory issues and slow processing speeds, GOAT reduces the input dtm, dsm, and landcover datasets to bounds that slightly extend past the user's domain of interest. The script uses the vsn constant to extend the cropping bounds (calcExt) slightly beyond the range of the lon/lat values in sampPoints.

```
calcExt <- c(xmin=min(sampPoints[,1])-vsn,
             xmax=max(sampPoints[,1])+vsn,
             ymin=min(sampPoints[,2])-vsn,
             ymax=max(sampPoints[,2])+vsn)
dtm <- crop(dtm, calcExt)
dsm <- crop(dsm, calcExt)
landcover <- crop(landcover, calcExt, snap='out')
```

By using the cropped input datasets and the lon/lat values from the sampPoints Matrix, the script builds the master spatial points data frame (spdf) associated with the domain Matrix. The cellFromXY function from the raster library identifies the grid cell numbers in each of the cropped

datasets associated with the lon/lat pairs. Variable columns in the new `spdf` data frame include `Lon` (numeric, decimal degrees), `Lat` (numeric, decimal degrees), `DTM` (numeric, m), `DSM` (numeric, m), and `LC` (integer, unitless), which represent the `sampPoints` longitudes and latitudes, DTM-based elevation, DSM-based elevation, and a binary land cover–based site-suitability parameter, respectively. The `LC` parameter is derived from the cropped land cover dataset by applying the `LCF` binary reclassification filter from the beginning of the script.

```
spdf <- as.data.frame(sampPoints)
colnames(spdf) <- c("Lon", "Lat")
spdf$DTM <- dtm[cellFromXY(dtm,spdf)]
spdf$DSM <- dsm[cellFromXY(dsm,spdf)]
spdf$LC <- LCF[landcover[cellFromXY(landcover,spdf)]]
```

At this stage, all of the necessary data ingest and domain setup steps are complete, and the script is ready to move on to the analysis phase.

As described in section 4.1, the point function, called `Topo` in the code, handles the circular sampling and terrain-suitability-criteria aggregation for each of the locations identified in `spdf`. Essentially, `Topo` enables GOAT to incorporate characteristics of the environment physically near each `spdf` point in addition to the attributes of each immediate `spdf` point location. The only argument passed to `Topo` is a single row of data from `spdf`, called `evalPoint`, associated with the individual point from the domain Matrix being evaluated. Note, the script stores `evalPoint` in a temporary 1 × 5 matrix, with the five columns representing the point under consideration's longitude, latitude, DTM-based elevation ($z_{DTM,p}$ from section 4.1), DSM-based elevation ($z_{DSM,p}$ from section 4.1), and binary land cover–based site-suitability parameter ($L_{c,p}$ from section 4.1), respectively.

```
Topo <- function(evalPoint){
```

For ease of discussion, we continue to refer to the `spdf` point under consideration as point *p* throughout this section. Note, code in the `Topo` function references column number values associated with the `evalPoint` matrix. For awareness, the only `evalPoint` matrix columns used in this version of GOAT are columns 1, 2, and 3, which represent the longitude, latitude, and DTM-based elevation of point *p,* respectively.

The `destPoint` function from the geosphere library computes the destination of a traveling point when given a starting location, initial bearing, and distance. `Topo` uses the `destPoint` function, the prescribed `sampRad`, and the `sampDir` vector of directions to identify the lon/lat locations of radial sampling points found along the perimeter of the analysis circle. In the code, `Topo` stores the circular sampling lon/lat pairs in a matrix called `localPts`. The lon/lat pairs from `localPts` also serve as the foundation of a temporary data frame called `local_df` with two columns: `Lon` and `Lat`. `Topo` also calls a `pointDistance` function to determine the distance in meters of the circle segments between neighboring radial points (`pointDis`). This task is only done once using two adjacent points since the sampling point spacing interval is consistent.

```
localPts <- destPoint(evalPoint[1,1:2], sampDir, sampRad)
pointDis <-pointDistance(localPts[1,], localPts[2,], lonlat=TRUE)
local_df <- as.data.frame(localPts)
colnames(local_df) <- c("Lon", "Lat")
```

Once `Topo` establishes the sampling circle, the function appends `dtm` values associated with each radial point to `local_df` in a column called `DTM`. These values are then offset downward by one row and stored in a new column called `DTML`. Per the overview in section 4.1, the `DTM` value from the last row in `local_df` becomes the value in the first row of `DTML`; and the difference between the `DTM` and `DTML` elevation values determines `ZDelta`, the elevation change between adjacent radial points. `Topo` then sums the absolute values of `ZDelta` to diagnose `vertChange`, an initial metric of terrain roughness for point p (R_{op} in section 4.1).

```
local_df$DTM <- dtm[cellFromXY(dtm,local_df)]
local_df$DTML <- lag(local_df$DTM)
local_df$DTML[1] <- local_df$DTM[nrow(local_df)]
local_df$ZDelta <- local_df$DTM - local_df$DTML
vertChange <- sum(abs(local_df$ZDelta))
```

The `Topo` function also calculates and appends the slope values from each circular point to point p to `local_df` in a column called `DTM_SLP`. The slope with the maximum absolute value from the `DTM_SLP` column determines the `steepestSlope` parameter ($\theta_{stp,p}$ from section 4.1), and `steepestSlopeNum` records the `local_df` row number associated with `steepestSlope`.

```
Local_df$DTM_SLP <- atan((local_df$DTM - evalPoint[1,3]) /
                    sampRad) * (180/pi)
steepestSlope <- max(abs(local_df$DTM_SLP))
steepestSlopeNum <- which.max(abs(local_df$DTM_SLP))
```

Note, the `DTM_SLP` diagnostic requires the use of the arc-tangent (`atan`) function. In R-scripting language, trigonometric functions return angles in radians by default, so conversion between radians and degrees is necessary.

Next, `Topo` creates a vector called `influenceSlope` that contains `steepestSlopeNum` and the row number associated with the radial point that is directly opposite from the radial point associated with the steepest slope. By using the row numbers from `influenceSlope`, `Topo` determines and applies the affiliated values from the `DTM` column and calculates the slope influence value (`slopeInf`; $\theta_{\text{n},p}$ in section 4.1) for point p.

```
influenceSlope <- c(steepestSlopeNum,
                    steepestSlopeNum-(nrow(local_df)/2))
influenceSlope <- ifelse(influenceSlope < 1,
                         influenceSlope + nrow(local_df),
                         influenceSlope)
slopeInf <- atan((abs(local_df$DTM[influenceSlope[1]] -
                  local_df$DTM[influenceSlope[2]])) /
            (sampRad*2)) * (180/pi)
```

The `Topo` function also finds the mean of the `DTM_SLP` column (`planeDev`; $\theta_{\text{avg},p}$ in section 4.1) to assess the deviation of the area from a planar surface.

```
planeDev <- mean(local_df$DTM_SLP)
```

Once the DTM-related parameters are known, `Topo` appends the elevation values from `dsm` associated with each circle point to `local_df` in a column called `DSM`. The function then calculates and appends the slope values from the `DSM` elevation at each radial point to point p in a column called `CLIMB`. Per the overview from section 4.1, this variable represents the elevation angle needed to clear surrounding obstacles. `Topo` then sets the `maxClimb`

parameter ($\theta_{C,p}$ from section 4.1) using the value with the largest magnitude from the `CLIMB` column to describe the maximum climb angle (in decimal degrees) for point *p*.

```
local_df$DSM <- dsm[cellFromXY(dsm,local_df)]
local_df$CLIMB <- atan((local_df$DSM - evalPoint[1,3]) /
                              sampRad)*(180/pi)
maxClimb <- max(local_df$CLIMB)
```

Lastly, `Topo` generates a binary variable called `treeLine` (T_p in section 4.1) to check for the presence of any radial points with a `DSM` value exceeding its respective `DTM` value by more than 3 m.

```
treeLine <- ifelse(any(local_df$DSM - local_df$DTM > 3), 0, 1)
```

`Topo` returns a single-row matrix containing six variables, including `vertChange`, `steepestSlope`, `slopeInf`, `planeDev`, `treeLine`, and `maxClimb`. Note, the use of the matrix function converts `treeLine` from an integer to a numeric value; however, this formality has no bearing on the functionality of the parameter. Once the return is complete, the `Topo` function closes.

```
outPut <- matrix(c(vertChange, steepestSlope, slopeInf, planeDev,
                   treeLine, maxClimb), ncol=6)
return(outPut)}
```

Next, the script creates a function called `pointValue` to pass rows of data from `spdf` to the `Topo` function and to build a new matrix called `obsMat` using the values returned by the `Topo` function. Each `spdf` row sent to `Topo` as an argument includes values for the point's longitude, latitude, DTM elevation, DSM elevation, and binary NLCD value. Every time `pointValue` runs the `Topo` function, `obsMat` gains a new row of populated data.

```
pointValue <- function(spdfRows){
obsMat <- matrix(nrow=nrow(spdfRows), ncol=6)
for(i in 1:nrow(spdfRows)){
      obsMat[i,1:6] <- Topo(spdfRows[i,])}
return(obsMat)}
```

After the script defines these two functions, it initiates the `pointValue` function and executes the `Topo` function for all points in the domain Matrix. The code temporarily stores the `obsMat` matrix from `pointValue` in another matrix called `localObs`. Accordingly, the script then parses the point-relative `vertChange`, `steepestSlope`, `slopeInf`, `planeDev`, `treeLine`, and `maxClimb` parameters from the temporary `localObs` matrix into new corresponding `spdf` columns labeled `VertDelta`, `SteepSlp`, `InflSlp`, `Plane`, `Obstacle`, and `MaxClimb`, respectively.

```
localObs <- pointValue(spdf)
spdf$VertDelta <- localObs[,1]
spdf$SteepSlp <- localObs[,2]
spdf$InflSlp <- localObs[,3]
spdf$Plane <- localObs[,4]
spdf$Obstacle <- localObs[,5]
spdf$MaxClimb <- localObs[,6]
```

Finally, the script uses the surface-roughness estimate from equation (1) (`Rough`: numeric, unitless) to diagnose terrain roughness hazards around each domain Matrix point and checks to ensure the maximum slopes do not exceed the macroslope tolerance defined by the end user.

```
spdf$Rough <- ifelse((spdf$SteepSlp <= slopeAcc), spdf$VertDelta
                     -((spdf$InflSlp -abs(spdf$Plane))*
                     2.447438), OOC)
```

After the surface-roughness calculation is complete, GOAT uses additional conditional statements to establish general HLZ suitability (`Suitability`: numeric, unitless) of each domain Matrix point ($S_{HLZ,p}$ from section 4.1). The script applies the `OOC` value to any point identified as having an undesirable land cover, landing obstacle, or perimeter obstacle.

```
spdf$Suitability <- ifelse(((spdf$LC==1) &
                            abs(spdf$DSM-spdf$DTM < 1) &
                            (spdf$Obstacle==1)),
                            spdf$Rough, OOC)
```

At this phase of the script, the GOAT analysis is complete, and all of the remaining code supports output visualization or writing. There are four data plots used as a visual check to assess the suitability progression rendered

by GOAT. The first is a maximum slope field that uses a linear spread from 0 degrees to `slopeAcc`. The second is the summation of circular roughness with a scale that ranges from 0 to an arbitrary value of 20 m. The third is the slope-adjusted roughness scaled from 0 to 4 m. While 3.5 m is currently the empirically driven threshold based on NTC SME guidance, using a 4 m range limit allows for visualization of areas slightly outside of the range when they occur. The last plot is for broader terrain suitability and incorporates binary discriminators for land cover, point obstacles, and perimeter obstacles.

```
plot_SteepSlp <- ggplot(spdf, aes(x=Lon, y=Lat,
                         colour=SteepSlp))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "Steepest Slope (decimal degrees)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=(slopeAcc/2),
                       limits=c(0,slopeAcc))

plot_VertDelta <- ggplot(spdf, aes(x=Lon, y=Lat,
                          colour=VertDelta))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
```

```
scale_colour_gradient2(name = "Initial Roughness (m)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=10, limits=c(0,20))

plot_Rough <- ggplot(spdf, aes(x=Lon, y=Lat, colour=Rough))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "Terrain Roughness (m)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=2, limits=c(0,4))

plot_Suitability <- ggplot(spdf, aes(x=Lon, y=Lat,
                           colour=Suitability))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "HLZ Suitability",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=2, limits=c(0,4))
```

GOAT also incorporates a fifth plot for the `MaxClimb` variable. This plot is scaled from 0 to 15 degrees, with the upper limit being the Pathfinder manual's steepest allowed consideration for landing while the rotorcraft is still under power. There is potential for the `MaxClimb` value to be negative

at sharp peaks where all directions are downhill. As written, the plot rendering would not highlight negative locations as viable HLZs.

```
Plot_MaxClimb <- ggplot(spdf, aes(x=Lon, y=Lat,
                         colour=MaxClimb))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "Maximum Climb (decimal degrees)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=7.5, limits=c(0,15))
```

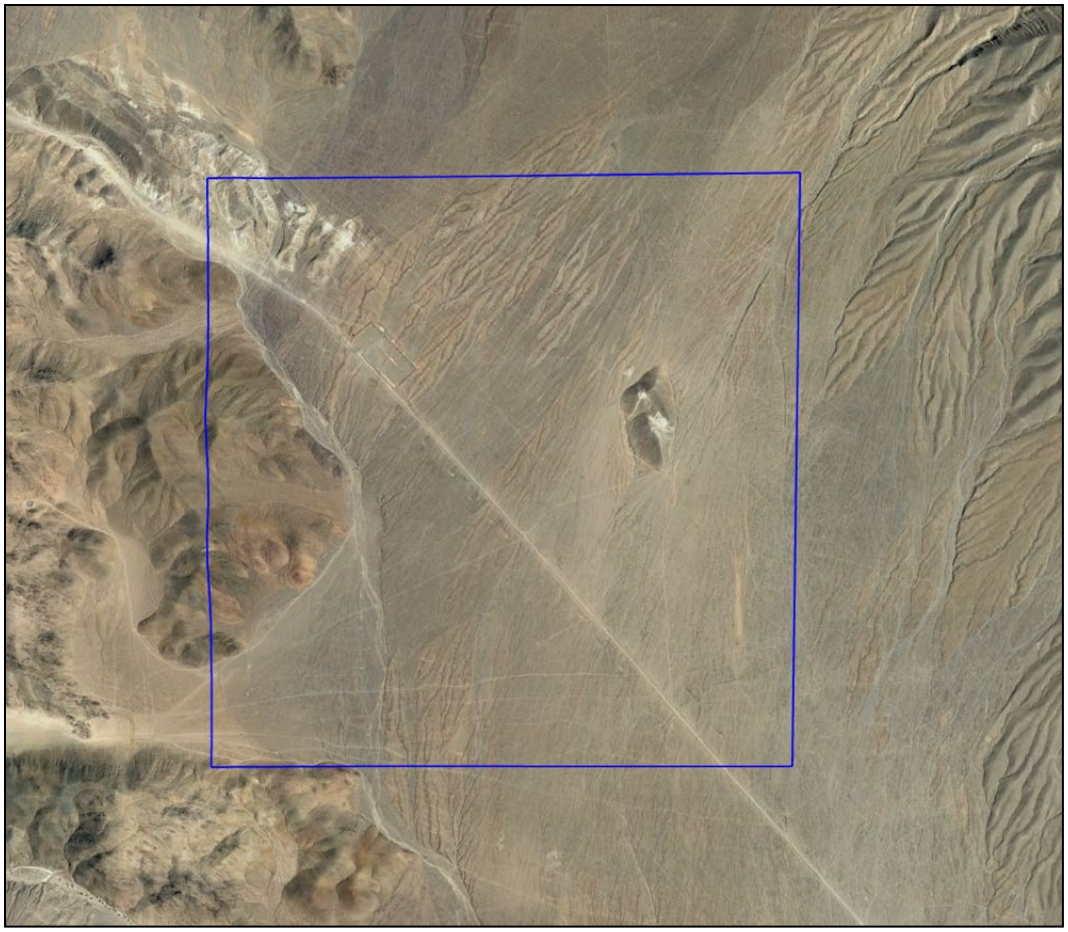
Lastly, GOAT outputs `spdf` as a comma separated value (CSV) file. We chose CSV format because it is easily ingestible by various geospatial analysis tools, including ArcGIS and Google Earth. The default output file name is GOATout.csv; however, users can modify their output file name as desired.

```
write.csv(spdf, "GOATout.csv")
```

4.3 Example output

This section demonstrates GOAT output for a 4 × 4 km area within the NTC training range (Figure 7). Most of the domain consists of wide-open space, with a lone mountain peak east of center and mountainous terrain on the western edge. As seen in Figure 7, alluvial features prone to having erosional channels dominate much of the lower-elevation terrain, and a dirt road cuts diagonally through the domain. Also, this particular region of NTC does not exhibit any of the three water-based land cover classes that restrict landing in the script.

Figure 7. A 4 × 4 km example domain (outlined in *blue*) used to demonstrate GOAT analysis fields showing rills and gully formation on an alluvial plain



Figures 8–12 show the resultant analysis fields, including the steepest slope (θ_{stp}; Figure 8), the initial roughness estimate (R_o; Figure 8), the final roughness assessment (R; Figure 10), the site-suitability assessment (S_{HLZ}; Figure 11), and the maximum climb (θ_c; Figure 12). Note that the suitability and the roughness plots exhibit similarity because the domain has relatively few vertical structures and no water features. Also, the red-yellow-green coloration gradient transitions in these plots are for visualization purposes only. We based these color settings on discussions with NTC SMEs, not doctrine.

The doctrinal criteria used in the suitability assessment (i.e., slope <7 degrees) are all threshold based, which leaves roughness as the only continuous variable. The suitability output shown accounts for multiple criteria; however, roughness based on residual meters of variance controls the color gradation.

Figure 8. Steepest slope (decimal degrees) value derived using the circular point function (i.e., the Topo function). Areas where steepest slope values exceed the maximum allowable slope appear *gray* (no shading).

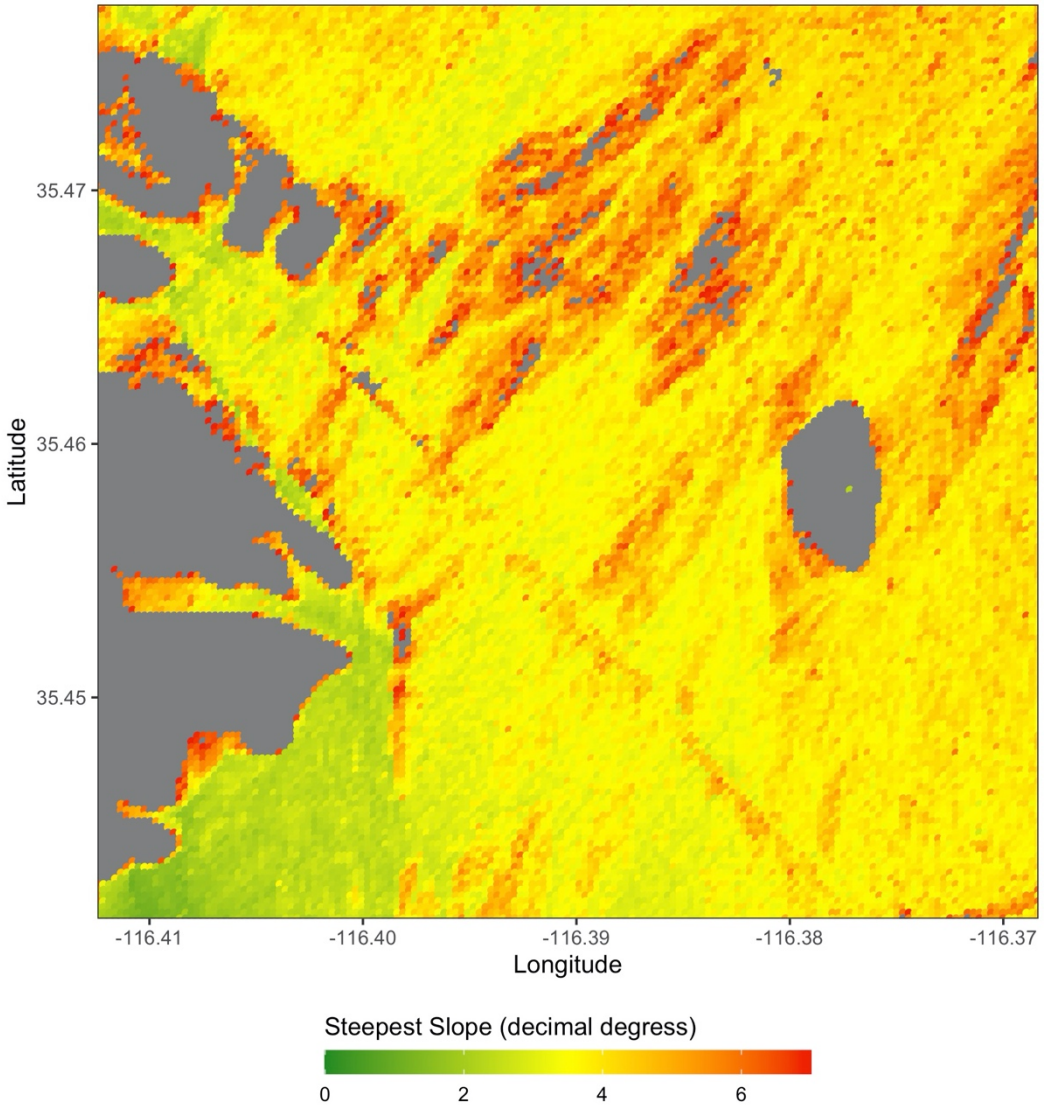


Figure 9. The initial sum of absolute vertical change (i.e., the initial roughness) in meters around the sampling circle at each Matrix domain point. Areas where steepest slope values exceed the maximum allowable slope appear *gray* (no shading).

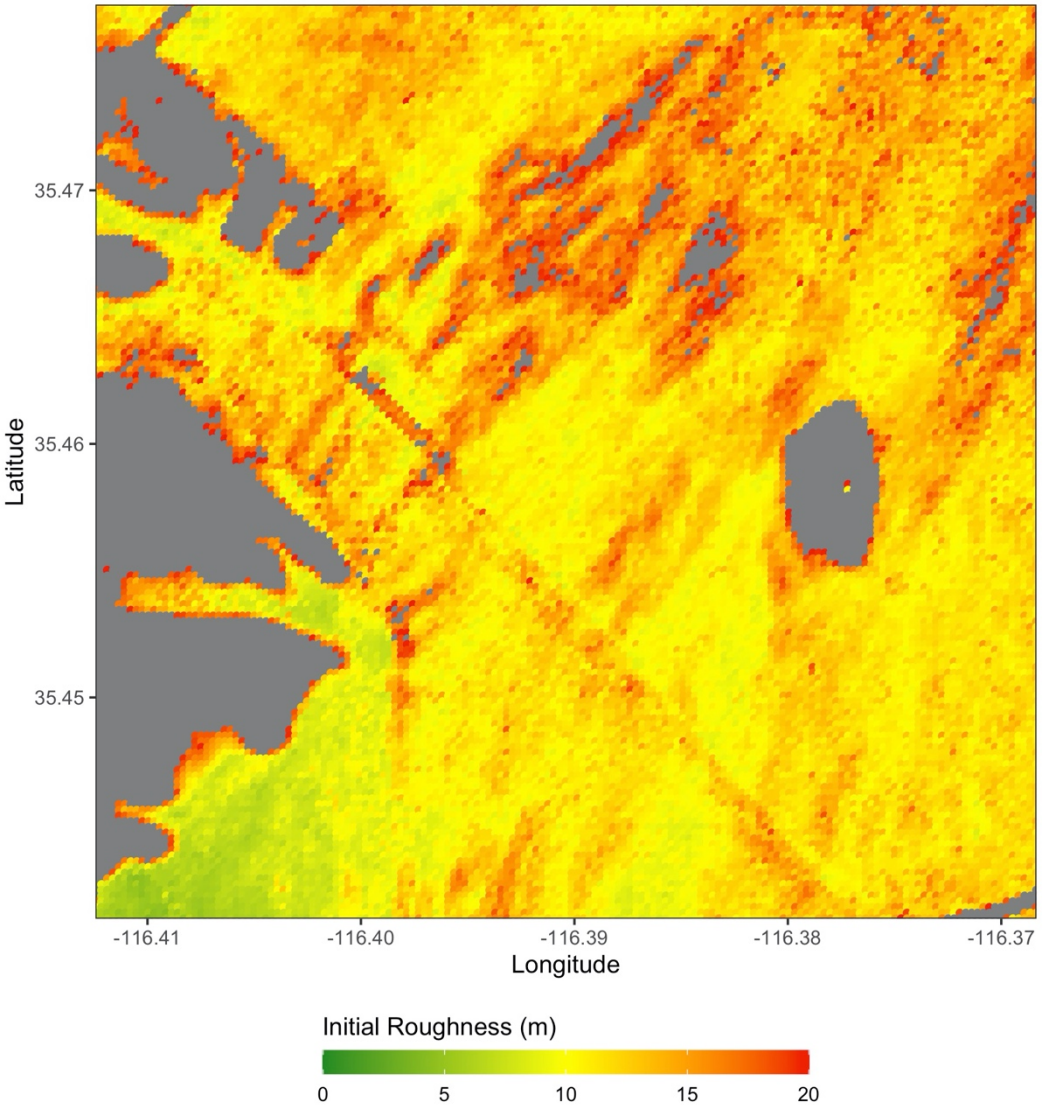


Figure 10. The sum of absolute vertical change (i.e., the roughness) in meters around the sampling circle at each Matrix domain point after the removal of macroslope effects. Areas where roughness values exceed the current empirically derived accepted maximum appear *gray* (no shading).

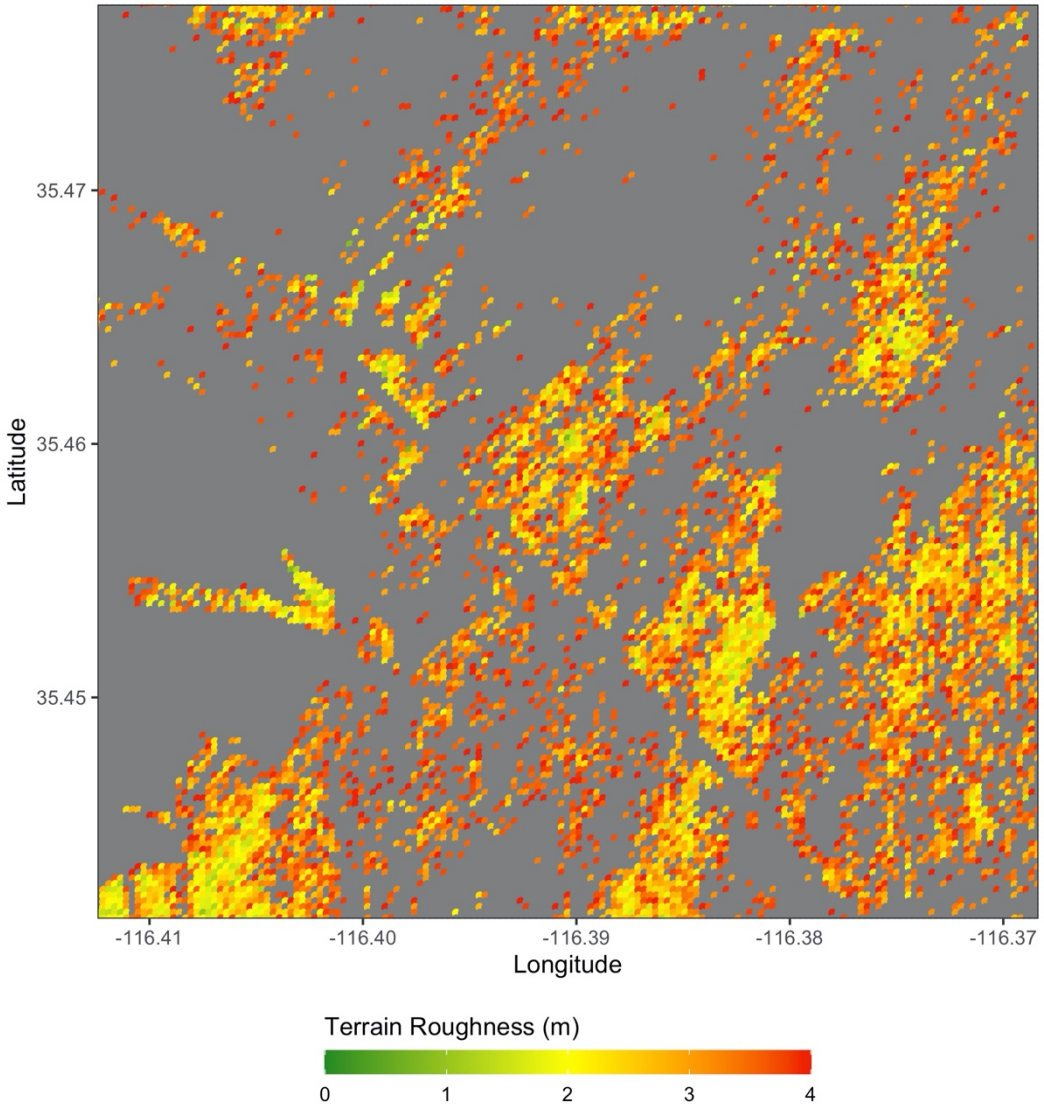


Figure 11. HLZ site suitability (unitless). The dominant criteria shown is roughness, as undesirable land cover and vertical obstructions are rare at this site. Areas deemed unsuitable appear *gray* (no shading).

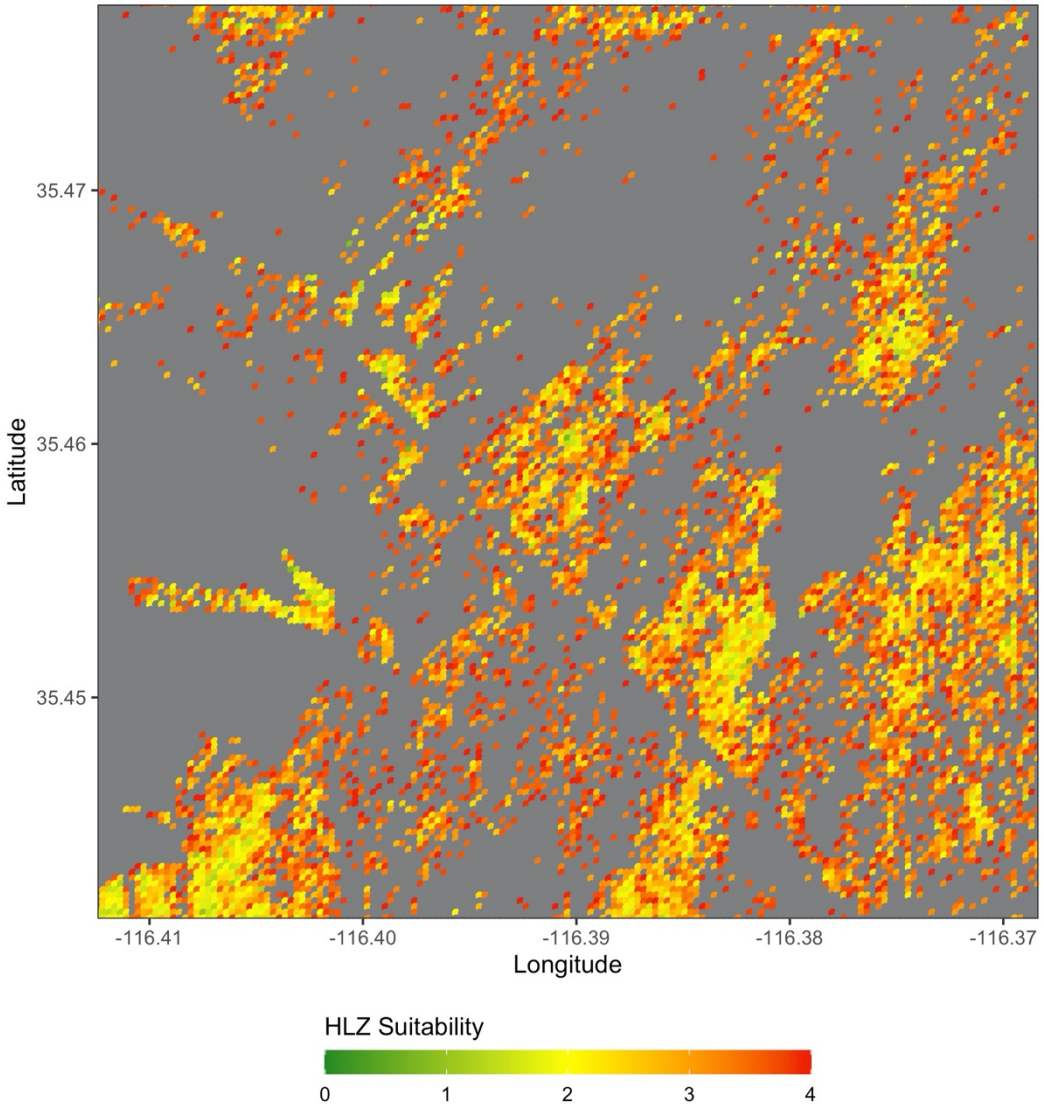
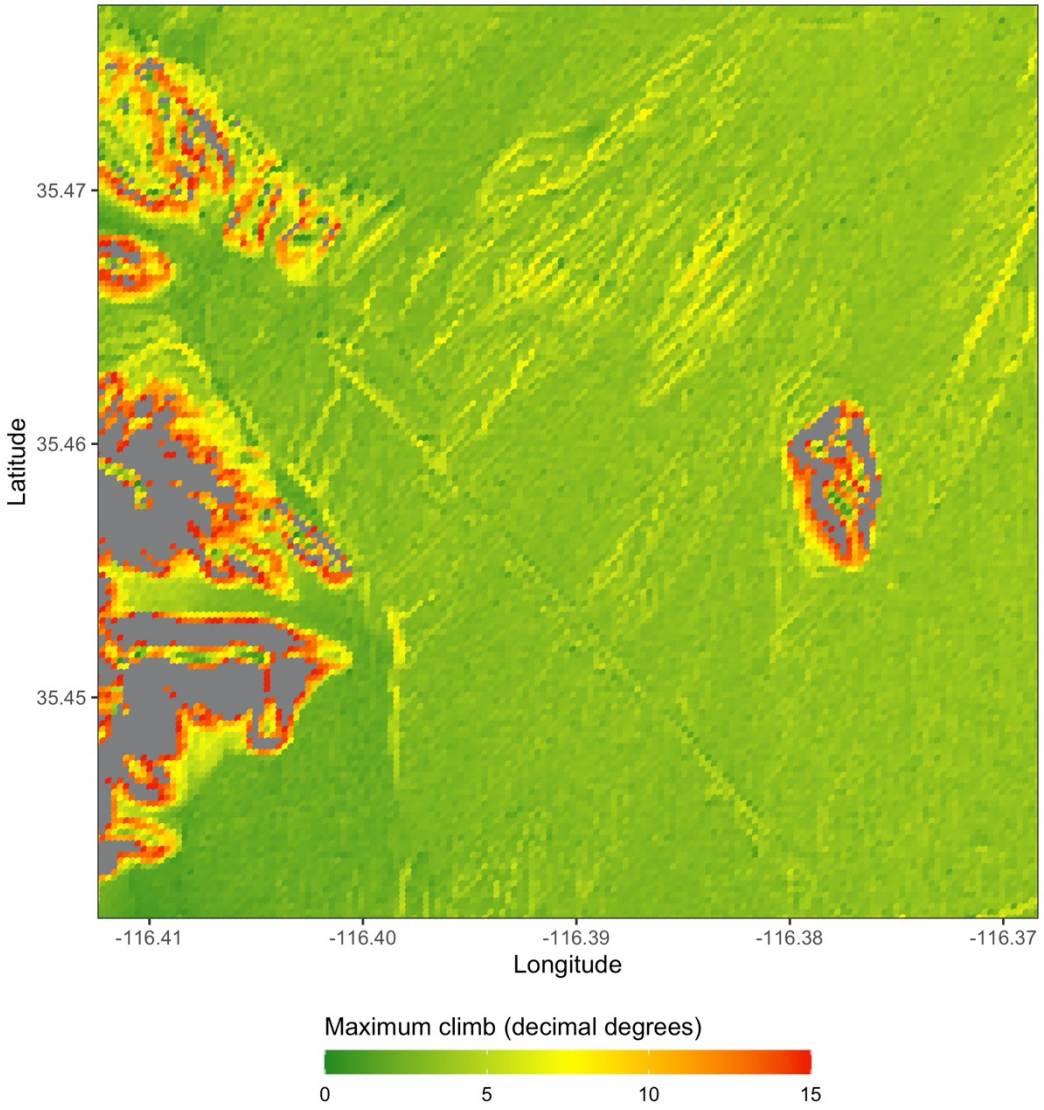


Figure 12. The maximum amount of climb needed to fly over surrounding obstacles at each Matrix domain point in decimal degrees. Areas where values exceed 15 degrees appear *gray* (no shading).



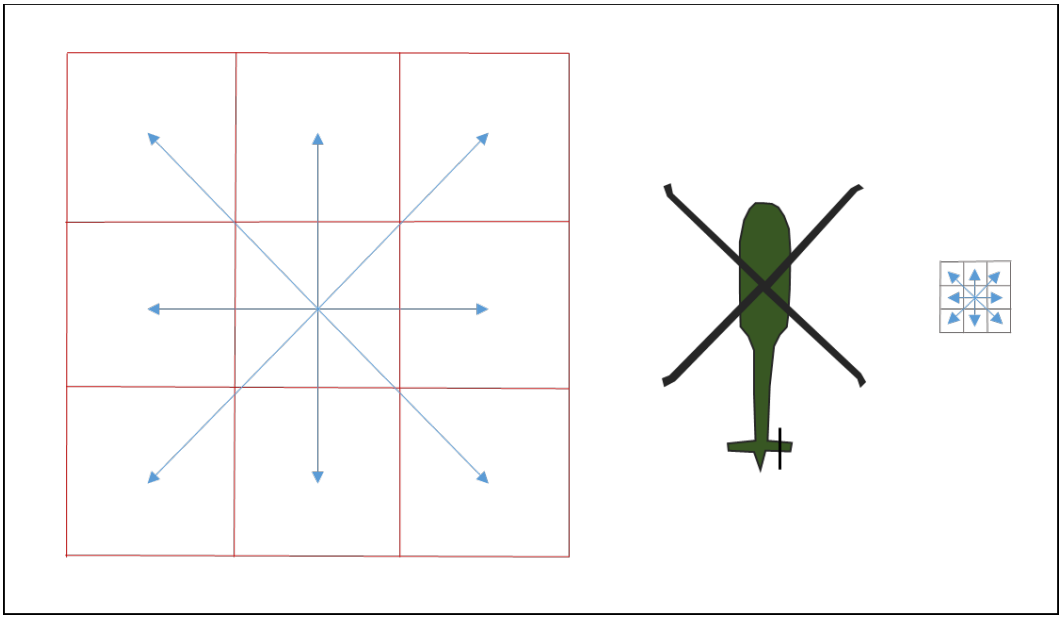
Many of the HLZ site-selection products currently in use would produce results similar to the steepest slope plot rendering in Figure 8. When terrain analysis considers surface roughness, the results can change profoundly (e.g., Figure 11). Also, note the considerable difference in spatial coverage between the HLZ site-suitability parameter plotted in Figure 11 and the maximum climb value plotted in Figure 12. Minor terrain undulations have less effect on missions that do not require helicopters to fully rest on the ground. If the rotorcraft can remain under power (i.e., hover rather than land), even the top of the central hill and a ridgeline on the western edge become usable.

5 Discussion

Terrain slope is the most unambiguous defined criteria for rotorcraft landing. As stated in Field Manual 3-21.38, *Pathfinder Operations*, "Do not land small utility and observation aircraft on slopes exceeding 7 degrees" (U.S. Army 2006). The concept of slope is simple to envision as a rate of change between two points, but it can be challenging to apply to real-world terrain considerations. For example, geographic information systems generally derive slope as a function of adjacent cells from gridded elevation datasets, creating a grid-resolution dependency.

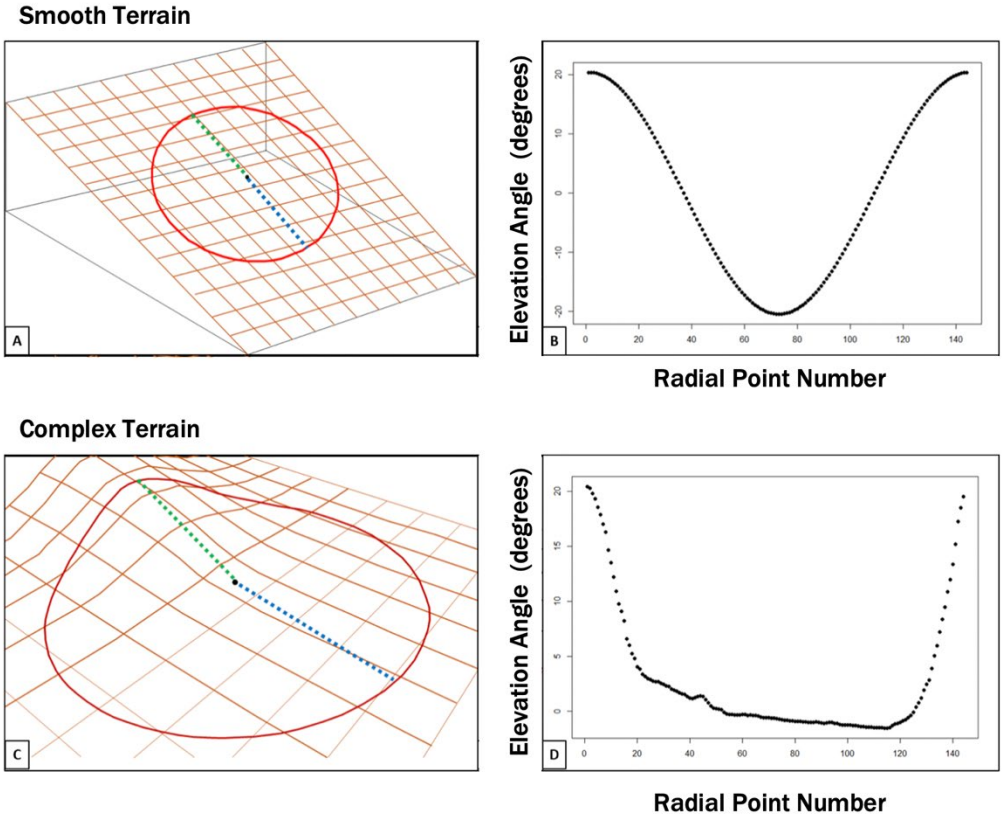
This dependency can cause relevancy issues for gridded-data-based HLZ and FARP site-selection tools. For instance, slope calculated from 30 m resolution terrain elevation data misses microfeatures but generally represents the area relevant to a helicopter's physical footprint (e.g., Figure 13). Conversely, 1 m resolution lidar-derived datasets capture microfeatures but use sub-helicopter-sized cells that have to be assessed further for their cumulative relationship. Running existing, macroslope-based HLZ site-selection tools with higher-fidelity elevation input data is not a reliable solution. We specifically designed the GOAT algorithm to account for these critical considerations of scale.

Figure 13. Conceptual diagram of gridded-input-dataset resolution compared to rotorcraft (not drawn to scale).



Furthermore, slope, a two-dimensional diagnostic, by itself gives a poor representation of essential terrain attributes needed for landing. Slopes derived from gridded data usually assume a smooth, continuous surface between grid cells and represent the steepest angle in two directions relative to a grid cell, one up and one down. Real-world terrain is rarely this smooth, so the steepest angle reported may be a bad approximation of the ground surface.

Figure 14. Conceptual illustration of slope computed from a point of interest to a circular sampling of equidistant points for (*A*) smooth and (*B*) complex terrain. The *red circle* represents the associated radial-point sampling positions. The *green dotted line* represents the steepest slope, and the *blue dotted line* represents slope in the opposite direction. The adjacent plots show the elevation angle from the circle center to the each of the 144 radial points that make up the sampling circle (*B* and *D* for smooth and complex terrain, respectively). The radial-point number sequence (*B* and *D*) begins at the end of the *green line* (*A* and *C*) and increases counterclockwise from 1 to 144, returning toward the *green line*.



As we sample outward with a given radius in GOAT, it becomes important to consider the shape of the terrain in our effort to correct for slope influence (e.g., Figure 14). Figure 14*A* shows a circular sampling on a smooth plane. Under these idealized conditions, the uphill slope and the downhill

slope are the same. Figure 14*B* shows the 144 sampled slope values starting from the steepest slope in Figure 14*A*, which is uphill in this case. Figure 14*C* shows a circular sampling that crosses a spur on the edge of a hill. The uphill slope angle (which is identical to the steepest slope from our smooth plane example) is steeper than the rest of the area, but it is a weak representation for the area in general. The slope in the opposite direction, however, does show strong continuity with the surrounding terrain (e.g., Figure 14*D*).

The algorithm's current technique of averaging the steepest and opposite slopes may still overemphasize the importance of a single sample. We intended the planar deviation correction to compensate for irregular terrain by mitigating the strength of broad slope reduction applied to the averaged slope angle. In essence, the planar deviation variable has a reducing effect on the correction for terrain slope. Future GOAT development efforts should explore methods to better approximate and incorporate terrain shape into landing suitability.

6 Conclusions and Recommendations

We designed GOAT to help terrain analysts incorporate surface-roughness effects into the HLZ site-selection process. This report describes GOAT v1.0 software, a relatively simple R-based script that can be run for a limited area of interest without the use of high-performance computing resources. In this version, GOAT generates a first-order assessment of HLZ terrain suitability based on macroslope, surface roughness, land cover, and tree line exposure.

A critical next step is to test GOAT in a formal user-experience exercise with soldiers and gather feedback to improve user interpretation (visualization rendering) and capability performance. Though technical evaluations of GOAT site-suitability performance are already underway at NTC (a desert landscape), future research efforts should also consider algorithm evaluation in different biomes. Future user-experience reviews focused on output data formatting and ease of use would also be of benefit.

None of the existing aviation decision-support capabilities used operationally incorporate roughness hazards. Without explicit doctrinal guidance or insight from terrain assessment tools, roughness considerations for HLZ and FARP site selection effectively depend on the experience and abilities of individuals responsible for vetting potential sites. Moreover, surface-roughness hazards can be difficult to detect from conventional satellite imagery, making rapid HLZ and FARP site assessment over relatively large spatial footprints a challenge. The GOAT v1.0 algorithm provides a rapid first-order assessment of terrain roughness hazards to rotorcraft. Though opportunity for improvement remains, GOAT received an official endorsement from the Aviation Center of Excellence in February 2020. As such, we advocate for the continued advancement of GOAT software and algorithms as an essential Army research priority.

References

AGC (Army Geospatial Center). 2013. "LIDAR Data." Accessed 25 August 2021. https://www.agc.army.mil/Media/Fact-Sheets/Fact-Sheet-Article-View/Article/480918/lidardata/.

AGC (Army Geospatial Center). n.d. U.S. Army Geospatial Center CMB Online. Accessed 25 August 2021. https://agcwfs.agc.army.mil/CMB_Online/default.html.

Multi-Resolution Land Characteristics Consortium. 2016. National Land Cover Database (NLCD) 2016. https://www.mrlc.gov/national-land-cover-database-nlcd-2016.

U.S. Army. 2006. *Pathfinder Operations*. FM 3-21.38.Washington, DC: Headquarters, Department of the Army. https://armypubs.army.mil/epubs/DR_pubs/DR_a/pdf/web/fm3_21x38.pdf.

U.S. Army. 2018. *Techniques for Forward Arming and Refueling Points*. ATP 3-04.17. Washington, DC: Headquarters, Department of the Army. https://armypubs.army.mil/epubs/DR_pubs/DR_a/pdf/web/ARN8800_ATP%203-04x17%20FINAL%20WEB.pdf.

U.S. Army. 2019. *101st Airborne Division (Air Assault) Gold Book*. Fort Campbell, KY: U.S. Army.

U.S. Army. 2016. *Aviation Tactical Employment*. ATP 3-04.1. Washington, DC: Headquarters, Department of the Army.

Wickham, J., C. Homer, J. Vogelmann, A. McKerrow, R. Mueller, N. Herold, and J. Coulston. 2014. "The Multi-Resolution Land Characteristics (MRLC) Consortium—20 Years of Development and Integration of USA National Land Cover Data." *Remote Sensing* 6 (8): 7424–7441. https://doi.org/10.3390/rs6087424.

Wieder, W. L., and S. A. Shoop. 2017. *Landing Zone and Drop Zone Criteria*. ERDC/CRREL SR-17-1. Hanover, NH: U.S. Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory.

Appendix: Code

The code presented here is also available to users via https://github.com/Sandra-LeGrand/site-suitability-tools. We encourage readers interested in using the GOAT code to check this website for version updates and bug fixes.

```
# Geomorphic Oscillation Assessment Tool version 1.0.
# Authors: Mike Ekegren and Sandra LeGrand (ERDC)

rm(list=ls()) # remove old variables from memory

# set directory paths (required)
setwd("C:\\.....")                     # working directory path
dtm_path <- "C:\\.....DTM_All.tif"     # DTM GeoTIFF path
dsm_path <- "C:\\.....DSM_All.tif"     # DSM GeoTIFF path
landcover_path <- "C:\\.....NLCD.tif"  # NLCD GeoTIFF path

# set domain attributes (required)
BASE_LON <- -116.7928  # domain origin-point longitude (deg)
BASE_LAT <- 35.3870    # domain origin-point latitude (deg)
subDist <- 400         # cardinal direction distance (m)

# tunable parameters
sampRes <- 25  # domain Matrix sampling resolution (m)
sampRad <- 35  # circular sampling radius (m)
slopeAcc <- 7  # maximum macroslope threshold (m)
####### end of user input

# additional prescribed values
sampInt <- 2.5             # circular sampling interval (deg)
OOC <- 2000                # nonsuitable site indicator
vsn <- 0.001               # very small number
nLCF <- 95                 # total number of land cover classes
LCF_hazard <- c(11,90,95)  # water-related NLCD land cover
                           # classes

# set binary land cover filter vector
LCF <- c(rep(1,nLCF))
LCF[LCF_hazard] <- 0
```

```
# set vector of circle sampling direction angles
sampDir <- c(seq(0, 359, sampInt))

# load required libraries
library(raster)       # raster data manipulation
library(rgdal)        # geospatial library
library(geosphere)    # performs distance/direction for sampling
library(dplyr)        # used for lag function
library(ggplot2)      # only needed for R plotting
library(RColorBrewer) # only needed for R plotting

# read input raster files
dtm <- raster(dtm_path)
dsm <- raster(dsm_path)
landcover <- raster(landcover_path)

# establish key domain placement attributes
# includes central point, NE corner, and NW corner of the Matrix
basePoint <- matrix(c( BASE_LON, BASE_LAT ), nrow=1)
northeast <- destPoint(basePoint, 45, sqrt((subDist^2)+
                       (subDist^2)))
northwest <- destPoint(basePoint, 315, sqrt((subDist^2)+
                       (subDist^2)))

# set frontage lon/lat ends and depth of zone
fZf <- matrix(c(northwest, northeast), nrow=2)
fZd <- subDist*2

# determine lon/lat pairs along frontage line
fZfBearing <- bearing(fZf[,1], fZf[,2])
fZfDist    <- pointDistance(fZf[,1], fZf[,2], lonlat=TRUE)
fZSamp     <- floor(fZfDist/sampRes)
fZfPoint   <- gcIntermediate(fZf[,1], fZf[,2], fZSamp,
                             addStartEnd=TRUE, sp=FALSE,
                             sepNA=FALSE)



# determine zone depth bearing
```

```
fZdBearing <- fZfBearing + 90
fZdBearing <- ifelse(fZdBearing > 359, fZdBearing - 360,
                     fZdBearing)

# build points along south boundary, same number as fZfPoint
fZdRear <- destPoint(fZfPoint, fZdBearing, fZd)

# fill in the rest of the domain matrix lon/lat values
sampPoints <- matrix(ncol=2)
for(i in 1:nrow(fZfPoint)){
  newPoints <- gcIntermediate(fZfPoint[i,], fZdRear[i,], fZSamp,
                              addStartEnd=TRUE, sp=FALSE,
                              sepNA=FALSE)
  sampPoints <- rbind(sampPoints, newPoints)}

#remove the first row which only has NA values
sampPoints <- sampPoints[-1,]

# crop the input datasets
calcExt <- c(xmin=min(sampPoints[,1])-vsn,
             xmax=max(sampPoints[,1])+vsn,
             ymin=min(sampPoints[,2])-vsn,
             ymax=max(sampPoints[,2])+vsn)
dtm <- crop(dtm, calcExt)
dsm <- crop(dsm, calcExt)
landcover <- crop(landcover, calcExt, snap='out')

# create master data frame
spdf <- as.data.frame(sampPoints)
colnames(spdf) <- c("Lon", "Lat")
spdf$DTM <- dtm[cellFromXY(dtm,spdf)]
spdf$DSM <- dsm[cellFromXY(dsm,spdf)]
spdf$LC <- LCF[landcover[cellFromXY(landcover,spdf)]]

# establish analytical Point Function
Topo <- function(evalPoint){



  # set radial sample points around circle
```

```
localPts <- destPoint(evalPoint[1,1:2], sampDir, sampRad)
pointDis <- pointDistance(localPts[1,], localPts[2,],
                          lonlat=TRUE)

# establish temporary data frame
local_df <- as.data.frame(localPts)
colnames(local_df) <- c("Lon", "Lat")

# set up dtm values
local_df$DTM <- dtm[cellFromXY(dtm,local_df)]
local_df$DTML <- lag(local_df$DTM)                # lag by 1 row
local_df$DTML[1] <- local_df$DTM[nrow(local_df)] # apply bottom
                                                  # row to top

# calculate adjacent radial point elevation difference
local_df$ZDelta <- local_df$DTM - local_df$DTML

# estimate initial roughness
vertChange <- sum(abs(local_df$ZDelta))

# calculate the slope in degrees to each radial sample point
local_df$DTM_SLP <- atan((local_df$DTM - evalPoint[1,3]) /
                    sampRad) * (180/pi)

# determine steepest slope and location
steepestSlope <- max(abs(local_df$DTM_SLP))
steepestSlopeNum <- which.max(abs(local_df$DTM_SLP))

# determine influential slope
influenceSlope <- c(steepestSlopeNum, steepestSlopeNum -
                  (nrow(local_df)/2))
influenceSlope <- ifelse(influenceSlope < 1, influenceSlope +
                         nrow(local_df), influenceSlope)
slopeInf <- atan((abs(local_df$DTM[influenceSlope[1]] -
                local_df$DTM[influenceSlope[2]])) /
                (sampRad*2)) * (180/pi)

# estimate deviation from planar
planeDev <- mean(local_df$DTM_SLP)
```

```
  # sample dsm at each radial point around the circle
  local_df$DSM <- dsm[cellFromXY(dsm, local_df)]

  # determine degree of climb for each radial point
  local_df$CLIMB <- atan((local_df$DSM - evalPoint[1,3]) /
                         sampRad) * (180/pi)
  maxClimb <- max(local_df$CLIMB) # steepest climb

  # identify any tree line hazards
  treeLine <- ifelse(any(local_df$DSM - local_df$DTM > 3), 0, 1)

  # establish output matrix
  outPut <- matrix(c(vertChange, steepestSlope, slopeInf,
                     planeDev, treeLine, maxClimb), ncol=6)
  return(outPut)}

# establish data passing function
pointValue <- function(spdfRows){
  obsMat <- matrix(nrow=nrow(spdfRows), ncol=6)
  for(i in 1:nrow(spdfRows)){
    obsMat[i,1:6] <- Topo(spdfRows[i,])}
  return(obsMat)}

# run calculation
localObs <- pointValue(spdf)

# add variables from return matrix to master data frame
spdf$VertDelta <- localObs[,1] # initial roughness
spdf$SteepSlp  <- localObs[,2] # steepest slope
spdf$InflSlp   <- localObs[,3] # influential slope
spdf$Plane     <- localObs[,4] # planer deviation
spdf$Obstacle  <- localObs[,5] # tree line obstacle indicator
spdf$MaxClimb  <- localObs[,6] # maximum climb

# calculate roughness diagnostic
spdf$Rough <- ifelse((spdf$SteepSlp <= slopeAcc),
                     spdf$VertDelta - ((spdf$InflSlp -
                     abs(spdf$Plane))* 2.447438), OOC)

# incorporate land cover and vertical obstructions
```

```
spdf$Suitability <- ifelse(((spdf$LC==1) &
                            abs(spdf$DSM-spdf$DTM < 1) &
                            (spdf$Obstacle==1)), spdf$Rough, OOC)

####### plot results
# show macroslope
plot_SteepSlp <- ggplot(spdf, aes(x=Lon, y=Lat,
                        colour=SteepSlp))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "Steepest Slope (decimal degrees)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=(slopeAcc/2),
                       limits=c(0,slopeAcc))

# show initial terrain roughness estimate
plot_VertDelta <- ggplot(spdf, aes(x=Lon, y=Lat,
                         colour=VertDelta))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "Initial Roughness (m)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=10, limits=c(0,20))
```

```
# show adjusted terrain roughness
plot_Rough <- ggplot(spdf, aes(x=Lon, y=Lat, colour=Rough))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "Terrain Roughness (m)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=2, limits=c(0,4))

# show integrated terrain suitability
plot_Suitability <- ggplot(spdf, aes(x=Lon, y=Lat,
                           colour=Suitability))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "HLZ Suitability",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=2, limits=c(0,4))

# show maximum climb
Plot_MaxClimb <- ggplot(spdf, aes(x=Lon, y=Lat,
                         colour=MaxClimb))+
geom_point(shape=16, size=1.5)+
theme_bw()+
theme(legend.position="bottom",
      legend.direction = "horizontal")+
```

```
guides(colour = guide_colourbar(title.position = "top",
                                barwidth = 15,
                                barheight = 0.75))+
labs(x = "Longitude", y = "Latitude")+
scale_x_continuous(expand = c(0, 0))+
scale_y_continuous(expand = c(0, 0))+
scale_colour_gradient2(name = "Maximum Climb (decimal degrees)",
                       low="forestgreen", mid="yellow",
                       high="red2", midpoint=7.5, limits=c(0,15))

####### write results
write.csv(spdf, "GOATout.csv")

####### save plots
ggsave("plot_SteepSlp.jpg", plot = plot_SteepSlp,
       width = 6.5, height = 7)
ggsave("plot_VertDelta.jpg", plot = plot_VertDelta,
       width = 6.5, height = 7)
ggsave("plot_Rough.jpg", plot = plot_Rough,
       width = 6.5, height = 7)
ggsave("plot_Suitability.jpg", plot = plot_Suitability,
       width = 6.5, height = 7)
ggsave("plot_MaxClimb.jpg", plot = plot_MaxClimb,
       width = 6.5, height = 7)

# announce progress
print("Processing complete.")
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2021	Final Report	FY19–FY21

4. TITLE AND SUBTITLE
Incorporating Terrain Roughness into Helicopter Landing Zone Site Selection by Using the Geomorphic Oscillation Assessment Tool (GOAT) v1.0

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT
0603119A

6. AUTHOR(S)
Michael T. Ekegren and Sandra L. LeGrand

5d. PROJECT NUMBER
BL8

5e. TASK NUMBER
01

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
Hanover, NH 03755

8. PERFORMING ORGANIZATION REPORT NUMBER
ERDC/CRREL TR-21-11

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Headquarters, U.S. Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSOR/MONITOR'S ACRONYM(S)
USACE

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

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

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Geomorphic Oscillation Assessment Tool (GOAT) quantifies terrain roughness as a mechanism to better explain forward arming and refueling point (FARP) suitability for Army aviation. An empirically driven characteristic of FARP consideration, surface roughness is a key discriminator for site utility in complex terrain. GOAT uses a spatial sampling of high-resolution elevation and land cover data to construct data frames, which enable a relational analysis of component and aggregate site suitability. By incorporating multiple criteria from various doctrinal sources, GOAT produces a composite quality assessment of the areal options available to the aviation commander. This report documents and demonstrates version 1.0 of the GOAT algorithms developed by the U.S. Army Engineer Research and Development Center (ERDC). These details will allow users familiar with R to implement it as a stand-alone program or in R Studio.

15. SUBJECT TERMS
Army Aviation, Forward arming and refueling point (FARP), Geospatial data, Helicopter landing zone (HLZ), Military geography, R scripting, Rotorcraft, Site selection, Surface roughness, Terrain characterization, Terrain suitability, Trafficability

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)
Unclassified	Unclassified	Unclassified	SAR	58	