Estimating Survey Efficacy - PES 2020

Center for Invasive Biology, Stellenbosch University
Nicholas Salonen
Version 1

January 27, 2021

Abstract

This document serves as a user manual on how to use the functions to estimate survey efficacy for the PES 2020 project being conducted by the Center for Invasive Biology (CIB) and the South African National Biodiversity Institute (SANBI). Four functions have been created to work together, namely, *importSurvey*, *surveyDim*, *plotSurvey* and *surveyEfficacy*. Currently, the functions are written for Matlab, but they will be translated to operate in Python and/or R.

Contents

1 Introduction **3**

1.1 Survey Efficacy 3

2 Function Usage to Estimate Survey Efficacy **3**

2.1 importSurvey 3

2.2 surveyDim 4

2.3 plotSurvey 5

2.4 surveyEfficacy 6

3 Function Results **6**

3.1 plotSurvey 6

3.2 surveyEfficacy 7

4 World File **7**

5 Further Development **8**

A Function Scripts **9**

A.1 importSurvey 9

A.2 surveyDim 11

A.3 plotSurvey 13

A.4 surveyEfficacy 17

1 Introduction

Surveys are conducted in locations where the presence of invasive Australian Acacias are known to exist. The purpose of the survey is to eradicate and monitor these populations. Surveys are done by contractor teams or by individuals who are involved in the study (and will be referred to as 'surveyors'). A surveyor will typically have at least one GPS device per group (in the case of a contracted team), whereas if a group of researchers perform a survey there may be one GPS per surveyor.

Surveyors will walk up and down, this allows the surveyor to cover as much ground as possible with the aim to cover the entire area of interest. It is important that the surveyor who is closest to the previous track leg does not stray too far from this segment. This reduces the area which could be missed.

1.1 Survey Efficacy

Understanding how well an area has been surveyed allows researchers to draw more telling conclusions. If 50% of the survey area is not visited, one could assume that there are specimens which have been missed and would indicate that the site needs to be revisited. To quantify the survey efficacy, a model was created which estimates the percentage of area that was 'visited', 'viewed' or 'missed' by a surveyor. The total area which is considered to be surveyed is the sum of the visited and viewed percentages. Observations are also plotted onto the map.

- Visited: The area, denoted by a grid cell, was occupied by a surveyor.
- Viewed: The area surrounding a visited grid cell is considered to be in viewing range.
- Missed: The area was not visited and falls outside of a surveyors viewing range.
- Observation: Specimen was found at this location.

2 Function Usage to Estimate Survey Efficacy

Estimating survey efficacy requires programming software. To obtain the desired output, four functions were created, namely, *importSurvey*, *surveyDim*, *plotSurvey* and *surveyEfficacy*. These need to be run in the correct sequence:

$$importSurvey \rightarrow surveyDim \rightarrow plotSurvey \rightarrow surveyEfficacy$$

Initial versions have been created for Matlab and will be extended to Python and R.

2.1 importSurvey

! → GPS track data from a survey is used to estimate survey efficacy. The GPS data consists of multiple latitude (lat) and longitude (lon) points. These lat/lon pairs indicate the position of a surveyor. The function *importSurvey* will import the lat/lon data as an m x 2 array. It can import csv or gpx file types. *importSurvey* can also import waypoints, in this case, the filetype must be specified as 'waypoint'.

importSurvey has three input arguments and creates two output variables:

- Input Arguments
  - filepath: path to folder in which the gps file is located.
  - filename: name of file located in filepath.
  - numberOfLogs (optional): specify number of track logs stored within a **gpx** file. If not specified, numberOfLogs defaults to 1. The upper limit of this variable can exceed the actual number of track logs.
  - Filetype: 'track' (default) or 'waypoint'. Specifies the type of gps data. If left empty, it defaults to track.
- Output Variables
  - track: m x 2 lat/lon array
  - filepath: Must be the directory where the functions are kept. GPS data can be in sub-directories.

Example for CSV:
$[pathcsv, trackcsv] = importSurvey("F:\PES2020",\ "Track.csv");$

Example for GPX track with 9 track logs:
$[pathgpx, trackgpx] = importSurvey("F:\PES2020",\ "Track.gpx",\ 1:9);$

Example for GPX waypoint with 195 waypoints:
$[path, track] = importSurvey("F:\PES2020",\ "wypts.gpx",\ 1:195,\ 'waypoint');$

2.2 surveyDim

Once the data is correctly imported, *surveyDim* is used to concatenate multiple tracks from a survey and calculates the dimensions of the survey area. The function is based on the haversine equation which calculates distances from lat/lon coordinates. These distances are the latitudinal and longitudinal distances of a survey area. This assists with correctly setting the grid cell size (bins) and also allows for an accurate calculation of the areas which are visited, viewed or missed. Do not pass waypoint data through this function.

! →

surveyDim only accepts m x 2 lat/lon arrays. It can accept any number of lat/lon arrays and creates three output variables:

- Input Arguments
  - varargin: where varargin is any number of m x 2 lat/lon arrays separated by commas.
- Output Variables
  - track: concatenated (if multiple tracks) m x 2 lon/lat array. The reason for the lat/lon to lon/lat conversion is due to the order in which *pcolor* in *plotSurvey* require the longitude and latitude (X and Y) variables.
  - lat_dist: latitudinal distance of survey area in meters.
  - lon_dist: longitudinal distance of survey area in meters.

Example for one imported track:
$[track, lat\_dist, lon\_dist] = surveyDim(track1);$

Example for multiple imported tracks:
$[track, lat\_dist, lon\_dist] = surveyDim(track1, track2, track3, track4);$

2.3 plotSurvey

This function plots the survey track onto a gridded surface. The first step of *plotSurvey* is to create a histogram which counts the number of times a grid cell is occupied by a surveyor. Due to the grid cell size, a grid cell could count multiple occurrences and for the sake of simplicity, all grid cells with a count of one or more were set to one and is stored as a matrix. The second step finds grid cells with a value of one and assesses if the neighbouring grid cell has a value of zero. If this is the case, it is given a new value of 0.5. Currently, the function sets the neighbouring three grid cells to 0.5, however, the idea is to set an input argument to set this parameter as it can be used to control the viewing range of the surveyor based on the terrain/vegetation. The final step is to remove all points which fall outside of the survey area. It is assumed that the outer most survey track is the border of the survey area. From the four borders of the grid, values are made NaN until the first value of 0.5 is found (as this dictates the edge of the survey area). This creates a data set that is simple to plot as the NaN values do not get assigned any color and do not form a part of the GPS data. After the matrix has been transposed, it is plotted with the *pcolor* function. This plots a matrix onto a grid. Each value is assigned a unique color. Blue indicates a visited grid cell, orange indicates a viewed grid cell and red indicates a missed grid cell.

plotSurvey accepts four input arguments and creates two output variables:

- Input Arguments
  - bin_width: sets the width of the square grid cells.
  - lat_dist: latitudinal distance of survey area in meters from *surveyDim*.
  - lon_dist: longitudinal distance of survey area in meters from *surveyDim*.
  - track: concatenated m x 2 lon/lat array created by *surveyDim*.
  - waypoint: waypoint data
  - image_file_p: image from google earth (optional)
  - world_file_p: sets image from google earth correctly into lat lon coordinates (optional). More information in section 4.
- Output Variables
  - surveymap: the matrix used to create the plot. This matrix is used to estimate survey efficacy.
  - bins: 1 x 2 array stating the number of bins along the longitudinal and latitudinal plane. This will be used to accurately calculate the grid cell sizes.

- Shortest: Shortest distance (meters) from an observation to a missed grid cell. Indicates the distance searched around each observation.

Example:

$$[surveymap, bins] = plotSurvey(1,\ lat\_dist,\ lon\_dist,\ track,\ ...$$
$$waypoint,\ image_{file_p},\ world_{file_p});$$

2.4 surveyEfficacy

The final function computes percentages based on the area covered by grid cells with unique values. The number of cells with each value (1, 0.5, 0) are multiplied by the area of a cell. i.e. The area that was visited is the cell area multiplied by the number of cells with a value of one. The sum of the visited, viewed and missed areas gives the total area. The percentages are then obtained and the surveyed percentage is the sum of the visited and viewed percentages. These values are stored in a table with headings.

surveyEfficacy accepts four input arguments and has one output variable:

- Input Arguments
  - track: sets the width of the square grid cells.
  - surveymap: matrix with information regarding how that cell was surveyed.
  - bins: number of bins in the longitudinal and latitudinal plane.
  - shortestDist: shortest distance estimated from *plotSurvey*.
- Output Variable
  - efficacy: 1 x 5 table with headings.

Example:

$$[efficacy] = surveyEfficacy(track,\ surveymap,\ bins,\ shortestdistance);$$

3 Function Results

When the functions are run correctly, the following output results will be obtained from *plotSurvey* and *surveryEfficacy*. *importSurvey* and *surveyDim* do not produce any final results, but produce variables which are used to obtain them. Please see section 2.1 or 2.2 to see which variables are generated respectively.

3.1 plotSurvey

plotSurvey produces a figure which indicates which areas are visited, viewed or missed. It gives a visual indication as to which areas of the survey site were surveyed more intensively. Below is an example of what a typical figure produced by *plotSurvey* should look like.

Figure 1: An example of the output generated by *plotSurvey*. The survey track is indicated by the blue coloring which represents a visited track. The orange coloring indicates the area around the track which is in the surveyors viewing range. The red indicates areas which are not considered to be surveyed at all and are labelled as missed. Green represents an observation of a specimen.

3.2 surveyEfficacy

surveyEfficacy indicates numerically what area of the survey is surveyed. The data used to create figure 1 is used to estimate the area for each survey category. This data is then stored as a table (shown in table 1) of percentages which belong to each category. Note that the category 'Surveyed' is the sum of the percentages belonging to visited and viewed.

Surveyed	Visited	Viewed	Missed	Shortest Dist (m)
66.22	44.30	21.92	33.78	5.1

Table 1: An example of the tabular output generated by *surveyEfficacy*. The values in each catergory represent the percentage of the survey area that is surveyed, visited, viewed and missed. For this example, only 66.22% of the survey site was surveyed which was 44.30% visited and 21.92% viewed. 33.78% of the site was missed, indicating that this site may need to be surveyed again. The table also shows the nearest missed area to an observation (in meters).

4 World File

! → A world file supplies information regarding the positioning of a map image in lat/lon coordinates. A world file for a jpg image and png image will have an extension of jgw and pgw respectively. The software which has been used during the creation of the functions in this manual is called 'SMART GIS 2020'. The software is free. The user guide (Generating World JPW file using Google Earth.pdf) and installer (Smart GIS Map Editor 21_01.zip) can be found in the supplementary folder. When saving an image from Google Earth Pro (must be v7.3.2), do not change the resolution of the image. The software does not work with later versions of Google Earth Pro. A world file has 6 rows of information, these are:

1. pixel size in the x-direction in map units/pixel
2. rotation about y-axis
3. rotation about x-axis
4. pixel size in the y-direction in map units, almost always negative
5. x-coordinate of the center of the upper left pixel
6. y-coordinate of the center of the upper left pixel

If you do know how to create a world file, or have your own method of doing so, feel free to do what works.

5 Further Development

The function does not handle long and extended circular tracks well. This functionality needs to be included for other styles of surveys, such as car surveys or surveys around dams or buildings where plants cannot grow. The function also struggles to remove some data points which do not form a part of the survey area. The technique by which the model removes points may need to be revised to ensure that all points which fall outside of the most exterior track are removed from the equations. This will ensure a more accurate result.

Currently, the model will produce good results for intensively surveyed sites which are characterized by zig-zagging survey tracks.

Appendix A Function Scripts

A.1 importSurvey

```
function [track, filepath] = importSurvey(filepath, filename, numberOfLogs,
    Filetype)
%IMPORTSURVEY Imports lat/lon data from a CSV/GPX file of GPS track data. The user
% specifies the directory FILEPATH and file FILENAME as well as being able
% to specifiy the number of track logs NUMBEROFLOGS which need to be
% imported if the file is a .gpx file. It is important to know the details
% of the file prior to importing, these can be found using mapping software
% such as the open source tool Garmin BaseCamp.
%
% Input arguments
% ---------------
% filepath = path to working directory
%            Working directory should contain functions and data unless
%            function filepaths' are added manually.
% filename = name of CSV/GPX file
%            Must include the file extension (.csv or .gpx).
% numberOfLogs = number of track logs/waypoints in a .gpx file
%                Imports the specified range of entries which can be
%                stored in a .gpx file. To import all track logs/waypoints,
%                numberOfLogs must cover the entire index range. The upper
%                limit can exceed the range. The default = 1 and will only
%                import the first track log or first waypoint.
%
% Filetype = 'track' (default) or 'waypoint'
%            Specifies if .gpx file is a waypoint file. Default file type
%            is track.
%
% Output variables
% ----------------
% track =  m x 2 [lat, lon] array
% filepath = the input filepath
%
% Examples
% --------
% Output track and filepath
% -------------------------
% [trackcsv, pathcsv] = importSurvey("F:\PES2020", "Track Acacia
    cultriformis and fimbriata Makhanda Grey Dam 20201026-28.csv");
% [trackgpx, pathgpx] = importSurvey("F:\PES2020", "Track Acacia
    cultriformis and fimbriata Makhanda Grey Dam 20201026-28.gpx", 1:9);
% [trackwpt, pathwpt] = importSurvey("F:\PES2020", "Waypoint Acacia
    cultriformis and fimbriata Makhanda Grey Dam 20201026-28.gpx", 1:500, '
    waypoint');
%
% Output track only
% ------------------
% trackcsv = importSurvey("F:\PES2020", "Track Acacia cultriformis and
    fimbriata Makhanda Grey Dam 20201026-28.csv");
% trackgpx = importSurvey("F:\PES2020", "Track Acacia cultriformis and
    fimbriata Makhanda Grey Dam 20201026-28.gpx", 1:9);
```

```
% trackwpt = importSurvey("F:\PES2020", "Waypoint Acacia cultriformis and fimbriata Makhanda Grey Dam 20201026-28.gpx", 1:500, 'waypoint');
%% Addpath to folder and subfolders
filepath = char(filepath);
addpath(genpath(filepath));
cd(filepath);

%% Import lat/lon from CSV/GPX
% Defaults
if ~exist('Filetype', 'var')
    Filetype = 'track';
    warning('goodnessOfSurvey:importSurvey:variableDefaultUsed', 'Filetype defaulted to track, if this is a waypoint file, set Filetype to waypoint.')
end

if ~exist('numberOfLogs', 'var') && contains(filename, '.gpx')
    numberOfLogs = 1;
    warning('goodnessOfSurvey:importSurvey:variableDefaultUsed', 'GPX files may contain multiple track logs. numberOfLogs defaulted to 1, some track logs from input variable filename may be missing. numberOfLogs can be set from 1:n to obtain all track logs.')
end
%
filename = char(filename);

if contains(filename, '.csv')
    trackcsv = readtable(filename);
    track = [trackcsv.lat, trackcsv.lon];
elseif contains(filename, '.gpx')
    trackgpx = gpxread(filename, 'FeatureType', Filetype, 'Index', numberOfLogs);
    track = [trackgpx.Latitude; trackgpx.Longitude];
    track = track';
    track = track(all(~isnan(track),2),:);

else
    error('goodnessOfSurvey:importSurvey:fileExtensionError', 'importSurvey only accepts .csv or .gpx file formats')
end
end
```

A.2 surveyDim

```
function [track, lat_dist, lon_dist] = surveyDim(varargin)
%SURVEYDIM Calculates the latitudinal and longitudinal distances of a
% survey area given the latitudes and longitudes as input arguments. This
% function is modeled on the haversine equation which uses latitude and
% longitude (as radians) to calculate distances in meters. The purpose of
% this calculation is so the bin widths can be correctly sized for plotting
    .
% If multiple files are imported using import_File, this function will also
% concatenate them into a single m x 2 [lon, lat] array.
%
% Input arguments
% ---------------
% varargin = any number of m x 2 [lat, lon] arrays
%            Most surveys will consist of multiple files, list all imported
%            [lat, lon] arrays which were imported using 'import_File'.
%
% Output variables
% ----------------
% track = m x 2 [lon, lat] array
% lat_dist = latitudinal distance
% lon_dist = longitudinal distance
%
% Example 1
% ---------
% [track_csv, lat_dist_csv, lon_dist_csv] = surveyDim(trackcsv);
% 'track_csv' [lon, lat] and 'trackcsv' [lat, lon] will be the same size.
%
% Example 2
% ---------
% [track_gpx, lat_dist_gpx, lon_dist_gpx] = surveyDim(trackgpx1, trackgpx2,
    ..., trackgpxN);
% 'track_gpx' [lon, lat] will be size of the sum of the lengths of
% 'trackgpx1', 'trackgpx2', ..., 'trackgpxN' [lat, lon].

%% Compile all imported tracks into one m x 2 [lon, lat] array
%nargin
if nargin < 1
    error("goodnessOfSurvey:surveyDim:notEnoughInputArguments", "Atleast
        one set of lat/lon coordinates are reauired.")
end

if nargin == 1
    track = [varargin{1}(:,2), varargin{1}(:,1)];
elseif nargin > 1
    arrlen = zeros(1, nargin);
    for x = 1:nargin
        arrlen(x) = length(varargin{x});
    end
    track = zeros(sum(arrlen), 2);
    for n = 1:nargin
        track(1:length(varargin{n}), :) = [varargin{n}(:,2), varargin{n
            }(:,1)];
    end
end
```

```
51
52 %% Haversine Calculation
53 % Convert to radians
54 lat_rad = track(:,2).*(pi/180);
55 lon_rad = track(:,1).*(pi/180);
56
57 % Delta lat and delta lon
58 dlon_rad = max(lon_rad) - min(lon_rad);
59 dlat_rad = max(lat_rad) - min(lat_rad);
60
61 % Haversine Formula
62 % Latitudinal distance
63 Alat = sin(dlat_rad/2).^2 + cos(max(lat_rad)) * cos(min(lat_rad)) * sin
       (0/2).^2;
64 Clat = 2 * atan2(sqrt(Alat), sqrt(1-Alat));
65 Dlat = (6371 * Clat) * 1000;
66 lat_dist = int16(Dlat);
67
68 % Longitudinal distance
69 Alon = sin(0/2).^2 + cos(max(lat_rad)) * cos(min(lat_rad)) * sin(dlon_rad
       /2).^2;
70 Clon = 2 * atan2(sqrt(Alon), sqrt(1-Alon));
71 Dlon = (6371 * Clon) * 1000;
72 lon_dist = int16(Dlon);
73
74 end
```

A.3 plotSurvey

```
function [surveymap, bins] = plotSurvey(bin_width, lat_dist, lon_dist,
    track)
%PLOTSURVEY creates a matrix using the gps track which was imported using
% import_File.m and computed using haversine.m. PLOTSURVEY scores each grid
% cell based on whether the cell was visited, viewed or missed. A visited
% cell gets given a value of 1, a viewed cell 0.5 and a missed cell 0.
% Based on the cell sizes, this gives an indication of how well an area was
% surveyed with the aim being to have no missed cells. The figure is
% plotted using the function PCOLOR and has axes values in meters according
% to the UTM coordinate system.
%
% Input arguments
% ---------------
% bin_width = size of each grid cell
%             A bin_width of 2 creates ~2x2 m cells (~4 m^{2}).
% lat_dist = latitudinal distance obtained from haversine.m
% lon_dist = longitudinal distance obtained from haversine.m
%            Both lat_dist and lon_dist, together with bin_width, are used
%            to set the size of the cells.
%            i.e. lon_dist = 397; bin_width = 2;
%                 lon_bin = lon_dist/bin_width;
%                 lon_bin = 199;
%                 Therefore there are 199, 2 m bins along the x-axis.
% track = m x 2 [lon, lat] array
%         Fills the grid cells with data. Each lat/lon pair which fall in a
%         grid cell give that cell a count. If a cell has at least one
%         lat/lon pair, that cell has been 'visited' by a surveyor.
%
% Output variables
% ----------------
% surveymap = gridded dataset which represents visited, viewed and missed
%             grid cells with 1, 0.5 and 0 resepctivley.
% bins = [lon_bins, lat_bins]
%        Indicates the size of the matrix and the scaling of the bins, also
%        used to calculate efficacy coefficients with surveyEfficacy.m.
%
% Example 1: Small or overgrown survey area - bin_width = 1
% ---------
% [surveymapcsv, binscsv] = plotSurvey(1, lat_dist_csv, lon_dist_csv,
    track_csv);
%
% Example 2: large or open survey area - bin_width = 5
% ---------
% [surveymapcsv, binscsv] = plotSurvey(5, lat_dist_csv, lon_dist_csv,
    track_csv);

%% Set bin size

lon_bin = double(lon_dist/bin_width);
lat_bin = double(lat_dist/bin_width);
bins = [lon_bin, lat_bin];

% Create gridded GPS track: indicates which Xm^2 areas are covered by
% a surveyor. Data stored as a matrix.
```

```
figure('Visible', 'off');
%figure(1);
[H] = hist3(track, bins);
hist3(track, bins)
caxis([0, 1])
cmap = jet(256);
cmap(1,:) = 1;
colormap(cmap)
colorbar
xlabel('Longitude');
ylabel('Latitude');
set(gcf,'renderer','opengl');
set(get(gca,'child'),'FaceColor','interp','CDataMode','auto');
view(2)

% Make all values greater than 1 equal 1
H(H >= 1) = 1;

for xx = 4:(lon_bin-3)
    for yy = 4:(lat_bin-3)
        if H(xx,yy) >= 1
            if H(xx+1,yy) == 0
                H(xx+1,yy) = 0.5;
            end
            if H(xx+2,yy) == 0
                H(xx+2,yy) = 0.5;
            end
            if H(xx+3,yy) == 0
                H(xx+3,yy) = 0.5;
            end
            if H(xx-1,yy) == 0
                H(xx-1,yy) = 0.5;
            end
            if H(xx-2,yy) == 0
                H(xx-2,yy) = 0.5;
            end
            if H(xx-3,yy) == 0
                H(xx-3,yy) = 0.5;
            end
            if H(xx,yy+1) == 0
                H(xx,yy+1) = 0.5;
            end
            if H(xx,yy+2) == 0
                H(xx,yy+2) = 0.5;
            end
            if H(xx,yy+3) == 0
                H(xx,yy+3) = 0.5;
            end
            if H(xx,yy-1) == 0
                H(xx,yy-1) = 0.5;
            end
            if H(xx,yy-2) == 0
                H(xx,yy-2) = 0.5;
            end
            if H(xx,yy-3) == 0
                H(xx,yy-3) = 0.5;
```

```
108                 end
109             end
110         end
111     end
112
113     % Make 0 values outside of survey area NaN (L to R: essentially top to
        bottom as before transpose)
114     for xa = 1:lon_bin
115         for xb = 1:lat_bin
116             if H(xa,xb) == 0
117                 H(xa,xb) = NaN;
118             elseif H(xa,xb) > 0
119                 break
120             end
121         end
122     end
123     % Make 0 values outside of survey area NaN (R to L: essentially bottom to
        top)
124     for ya = lon_bin:-1:1
125         for yb = lat_bin:-1:1
126             if H(ya,yb) == 0
127                 H(ya,yb) = NaN;
128             elseif H(ya,yb) > 0
129                 break
130             end
131         end
132     end
133
134     % Transpose so orientation of matrix is correct
135     surveymap = H';
136
137     % Make 0 values outside of survey area NaN (L to R)
138     for xc = 1:lat_bin
139         for xd = 1:lon_bin
140             if surveymap(xc,xd) == 0
141                 surveymap(xc,xd) = NaN;
142             elseif surveymap(xc,xd) > 0
143                 break
144             end
145         end
146     end
147     % Make 0 values outside of survey area NaN (R to L)
148     for yc = lat_bin:-1:1
149         for yd = lon_bin:-1:1
150             if surveymap(yc,yd) == 0
151                 surveymap(yc,yd) = NaN;
152             elseif surveymap(yc,yd) > 0
153                 break
154             end
155         end
156     end
157
158     % Create X and Y axis: latitude and longitude need to be the same length
159     % as the dimensions of matrix H1_corrected.
160     %lat_ax = lat_min:((lat_max - lat_min)/(lat_bin-1)):lat_max;
161     %lon_ax = lon_min:((lon_max - lon_min)/(lon_bin-1)):lon_max;
```

```
162
163 % Or in UTM
164
165 tracks_utm = ll2utm(track(:,2), track(:,1));
166 lat_utm_max = max(tracks_utm(:,2));
167 lat_utm_min = min(tracks_utm(:,2));
168 lon_utm_max = max(tracks_utm(:,1));
169 lon_utm_min = min(tracks_utm(:,1));
170 lat_ax = lat_utm_min:((lat_utm_max - lat_utm_min)/(lat_bin-1)):lat_utm_max;
171 lon_ax = lon_utm_min:((lon_utm_max - lon_utm_min)/(lon_bin-1)):lon_utm_max;
172
173 % Plot matrix as a surface plot. Shows grids where a surveyor walked
174 % as equivalent to 1, and neighbouring grids which were in viewing range
175 % as 0.5.
176 figure('Visible', 'On');
177 pc = pcolor(lon_ax, lat_ax, surveymap);
178 caxis([0, 1])
179 cmap = bluewhitered(3);
180 cmap(1,:) = [1 0 0];
181 cmap(2,:) = [1 0.5 0];
182 cmap(3,:) = [0 0 1];
183 colormap(cmap)
184 cb = colorbar;
185 cb.YTick = [1/6 3/6 5/6];
186 cb.YTickLabel = {'Missed', 'In Range', 'Surveyed'};
187 xlabel('UTM_{x} [m]');
188 ylabel('UTM_{y} [m]');
189 hAx = gca;
190 hAx.YAxis.Exponent=0;
191 hAx.XAxis.Exponent=0;
192 ytickformat('%d')
193 xtickformat('%d')
194 set(hAx,'XMinorTick','on','YMinorTick','on')
195 set(pc, 'EdgeColor', 'none');
196 xlim([min(lon_ax)-5, max(lon_ax)+5])
197 ylim([min(lat_ax)-5, max(lat_ax)+5])
198 str = strcat('Grid cell area = ~', num2str(bin_width), 'm^{2}');
199 annotation('textbox',[0.754 0.48 0.5 0.5],'String',str,'FitBoxToText','on')
    ;
200
201 end
```

A.4 surveyEfficacy

```
function [efficacy] = surveyEfficacy(track, surveymap, bins)
%SURVEYEFFICACY Computes the efficacy of the survey.
% The efficacy coefficients are based on the areas which are visited,
% viewed and missed. The total area which is considered to be surveyed is
% the sum of the visited and viewed area. The percentage of area which is
% surveyed is computed using the bin_area which is then multiplied by the
% count of cells for each unique value (1, 0.5 and 0).
%
% Input arguments
% ---------------
% track = m x 2 [lon, lat] array
%         Used to obtain the latidudinal and longitudinal distances.
% surveymap = data grid
%             Contains information regarding surveyed status of each cell.
% bins = [lon_bin, lat_bin]
%        The number of longitudinal and latitudinal bins. The latitudinal
%        and longitudinal distances are divided by lon_bin and lat_bin
%        respectively to give an accurate estimation of a single cell's
    area.
%
% Output variables
% ----------------
% efficacy = 1 x 4 table
%            Contains information about the percentage of area which were
%            surveyed, visited, viewed and missed. The percentage of area
%            which was surveyed is the sum of the areas which were visited
%            and viewed.
%
% Example
% -------
% efficacycsv = surveyEfficacy(track_csv, surveymapcsv, binscsv);
% Output: 66.2241806664831 | 44.3018452217020 | 21.9223354447811 |
    33.7758193335169

%% Generate Survey Efficacy coefficient
% Area of each grid cell/bin = bin_width^2
% Calculate precise bin area

utm_co = ll2utm(track(:,2), track(:,1));

bin_height = ((max(utm_co(:,2)) - (min(utm_co(:,2))))/bins(2));
bin_length = ((max(utm_co(:,1)) - (min(utm_co(:,1))))/bins(1));
bin_area = bin_length*bin_height;

area_missed = length(surveymap(surveymap == 0)).*bin_area;
area_viewed = length(surveymap(surveymap == 0.5)).*bin_area;
area_visited = length(surveymap(surveymap == 1)).*bin_area;
area_total = area_missed + area_viewed + area_visited;

% Percentage of area surveyed/viewed/missed
perc_missed = (area_missed/area_total)*100;
perc_visited = (area_visited/area_total)*100;
perc_viewed = (area_viewed/area_total)*100;
perc_surveyed = ((area_viewed+area_visited)/area_total)*100;
```

```
53
54 efficacy = table(perc_surveyed, perc_viewed, perc_visited, perc_missed, '
       VariableNames', {'Surveyed', 'Viewed', 'Visited', 'Missed'});
55 end
```