**GEOM 4009 Group Report: SORA**

Nathan Byerley, Alexandra Mullen, Olivia Hobbs, Daniel Nelson

**Introduction**

Our client Brendan Ooi is a recent master’s graduate from Carleton University’s Department of Mechanical and Aerospace Engineering. Brendan is a part of Transport Canada’s research team, Brendan’s research interests lie in the future applications of drones, ensuring that they can safely travel from point A to Z given the environmental factors and other aircraft traffic.

The larger research team working on this project are in the development phase. As a result, Brendan has given our group a reasonable amount of flexibility regarding priorities of individual tasks. Our involvement in the project mostly lays within visualization using GIS. Transport Canada does not have any team members working on the project that have a background in GIS at this stage in development. The core components of the project consisted of filtering the large data set into categories of airspace height to assess risk. Brendan’s research team is familiar with Google Earth, which is what they have been using for visualization. As a result, we chose to work with KML files. We were given a great amount of flexibility regarding software and programming language used, so we decided to use Python. Brendan requested that our Python code have clear, concise comments for the benefit of the research team. Leaving descriptive comments in the final Python script will assist the team in the future to understand what we did, why we did it, and what it entails.

The purpose of this project is to create a specific operational risk assessment (SORA) for Remotely piloted aircraft systems (RPAS) program to assist Transport Canada and their associated research team in assessing risk analysis for drone flights in populated areas. The risks are calculated in 2 ways: air to air collision risk, and air to ground collision risk.

The scope of this project is broken down into four main functions. The first is main which helps the user input the required information. Next we have a filter flights function which selects the flights that meet the input selection parameters to filter the dataframe of flights outputting a geodataframe of appropriate flights. The Air risk function takes output from filtered flights and calculates the air risk based on flights entering the user inputted area of interest and outputs one text file to the user specified folder. This file contains a count showing the amount of flights in the area of interest and the area of the area of interest in km^2. The ground risk function looks at the population density in a census tract and then cross references it with the filtered flights function in order to output a text file with the flight and the census tract it is in with the associated population density.

**Workflow**

* Take user inputs through command line and run them to the right function
* Take user flight inputs regarding time and location and filter required flights, output them in a geodataframe
* Outputs from previous step are used to calculate the air to air risk collison using the user inputted area of interest to output a text file with flight count in area of interest, and the area of the area of interest in km^2
* Uses output of filtered flights geodataframe to cross reference population density from census tracts and outputs a text file of the flights and which census tract its in with the population density to calculate air to ground risk
* Please see Figure 1 in the Appendix for an updated UML diagram demonstrating the workflow of the SORA project

**Documentation**

*Dependencies*

Python 3.x is required for the use of the SORA code. As of April 4, 2021, the newest version of Python for Windows is Python 3.9.4. Python is available for Windows, Linux/UNIX, and Mac OS X.

Required hardware dependencies:

* A Windows, Linux, or OS X machine running Python 3.x

Required software dependencies:

* Anaconda (required on Windows machines, optional for Linux and Mac)
* Conda package manager (required on Windows machines, optional for Linux and Mac)

Required library dependencies:

* os
* argparse
* datetime
* pandas
* geopandas as gpd
* fiona

*User Guide*

1. Make sure Python 3.x is installed, if not head to <https://www.python.org/downloads/> for the latest version.

2. Make sure either Qgis or ArcPro is downloaded. <https://qgis.org/en/site/forusers/download.html>

3. Collect required data. Flight data in CSV format, census tract shape file (.shp) and population density data (CSV). Census tracts available for download from <https://www12.statcan.gc.ca/census-recensement/2011/geo/bound-limit/bound-limit-2016-eng.cfm> and population density data available for download from <https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/hlt-fst/pd-pl/Table.cfm?Lang=Eng&T=1601&SR=1&S=94&O=D&RPP=25&PR=0&CMA=850#tPopDwell>

**\*\*\*Make sure all data is being saved in the same folder! \*\*\*\***

\*\*If the census data is already joined to the census tract shape file, skip to step 5\*\*

4. Join the Census Data to the census tract file, ArcPro or Qgis can be used.

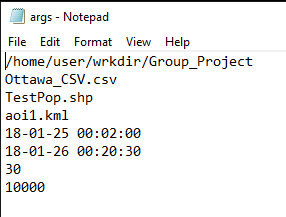
1. Launch QGIS or ArcGIS Pro
2. Load the Census tract shape file
3. Load the Census data
4. Join the census data to the census tract file using the CTUID and Geographic code
5. Export the resultant layer as a shape file
6. While still in the ArcPro or QGIS rename field that corresponds to the population density per square kilometre, PopDen
7. Save the edits made
8. Exit the GIS software.

5. Make a KML of the area of interest. Google Earth pro is the best option

6. Determine the other parameters. Start date and time, end date and time, minimum elevation, maximum elevation

7. Determine the file path where all the required files are saved.

8. Make a .txt file with the required parameters, each parameter has to be on a new line with no additional spaces before or after. See example below.



The first entry is the path, the second entry is the flight data, the third entry is the population density shape file. The fourth entry is the area of interest kml file. The fifth entry is the start date and time. The sixth entry is the end date and time. The seventh entry is the minimum altitude. The eighth entry is the maximum altitude. Please note that the altitude is measured in feet.

9. Save the .txt file in the same folder as the SORA python script.

10. Open command line

11. Navigate to the directory containing the SORA python script (use cd to change directory)

Type in “python” followed by “SORACode.py” followed by the “@” and name of the .txt file containing the arguments. In the case shown below it’s args.txt



12. Hit the enter key

13. To view the results, in the same folder as the rest of the data two new text files have appeared. AirRisk and GroundRisk. The Ground risk file has the census tract ID and population density is printed out for each flight that passes over it between the start time and end time. The AirRisk file is the AirRisk. It gives a number for how many flights are over the area of interest within the desired time period and it also gives the total area in square kilometres for the area of interest.

*Troubleshooting/FAQ*

***Do I really need Conda to run the SORA code on Windows?***

Yes. The Conda package manager is required when using many of the libraries required for running the SORA code, for example GeoPandas. These libraries are not installed on their own in Spyder on a Windows computer.

***How do I install packages on Conda?***

To install packages, the Conda package manager is required. After it is installed, visit <https://geopandas.org/getting_started/install.html>. The website demonstrates how to install the GeoPandas package, but the instructions are applicable to the other packages used in the SORA code.

***What is the difference between QGIS and ArcGIS Pro?***

QGIS is an open source GIS software and is free to use. A download link is available here: <https://qgis.org/en/site/forusers/download.html>. ArcGIS Pro is an ESRI product and has varying costs associated with the type of license desired. Both of these GIS products are able to perform the join as stated in the User Guide.

***Can I use a different metric to determine population density, instead of census data?***

Yes but it will require changing the inputs that all start with OttawaPopDens.(format). This will also require the new data to match either the same variable names used in the GroundRisk function or changing the code to reflect the data in the new chosen population density metrics. This program has only been tested using shapefiles (.shp) to calculate the population density, note this file is a different format than what the user inputs the area of interest in which is KML.

***How can I change a KML to Shapefile?***

Here is a tutorial demonstrating how to do it in QGIS.

Link: <https://www.igismap.com/convert-kml-shapefile-qgis/>

Here is how to do so if you use ArcGIS Pro.

Link:<https://pro.arcgis.com/en/pro-app/latest/tool-reference/conversion/kml-conversion.htm>

***Where can I download Conda?***

Here is the download link for Windows, MacOS, and Linux. Link:<https://conda.io/projects/conda/en/latest/user-guide/install/index.html>

**Discussion**

*Challenges*

One of our challenges was the amount of flexibility we had. Brendan has no prior experience working with Python, only Google Earth, so there was a significant amount of flexibility given to our group. It allowed us to try several different methods to achieve our goals when writing our functions.

Another challenge we had was the creation of the ground risk function. We did not take into account that the function not only required population density, but it also required the flights that are over the census tracts. Originally, our output for this function was a choropleth map with population density of the census tracts, which was later completely revamped and changed.

Fiona has a difficult time working with KML files, which was our goal output due to the experience of the research group we were working with. The solution we came up with was to add an extra line of code to account for the difficulties.

We also had issues with integrating all four of the functions together. Each function was created independently by one group member. While each function worked perfectly fine on its own, the code presented a lot of errors when the functions were run together. There was a significant amount of time spent debugging the entire script containing all four functions.

*Limitations*

Our project had several limitations. One key limitation is the way the program interacts with the user. The file inputs need to contain both the correct information and the correct formatting. This is anticipated to have some level of user error, building off this limitation a key goal would be to work on getting the user better error readings for when their inputs have been put in incorrectly.

Another limitation was regarding the filter flight function. The flights could have multiple locations within the time frame, but just the last position will be kept and processed meaning that some flights may not accurately show all of the census tracts that they flew over.

The outputs of the function may also be challenging for a reader to understand if they do not understand what the job of the air risk and ground risk functions are.

*Client Interactions*

As previously stated in Progress Report 1, our goal was to email our client before each progress report was submitted. Based on the way our team and client interacted we found it more beneficial to send an email after each report was submitted, with a brief summary of what we had been upto and if we had any questions. One team member maintained being our point of content with our client with other team members being CC’d on the emails which resulted in us also seeing our clients response. Our client was wonderful to interact with and typically had less than a 24hr turnaround time for responding to emails. Our client gave us some feedback after progress report 3 on two of our functions, this was the only feedback we received from the client. We think our client had a difficult time giving feedback due to his educational background not being in Python or GIS. Overall he was incredibly pleasant to work with.

*Future Work*

Majority of the future work is regarding things that came from our limitations. With more time and coding experience all these things are simple to accomplish. A key component to having this work well integrated with Transport Canada would be to have a guided user interface as in the current form the inputs will be difficult for those with limited knowledge and exposure to understand and execute this program. Another idea for future work is to integrate the SORA code into the geospatial web and push the user to select an area of interest by creating a polygon on an interactive map. This can be done using online geospatial web services such as Mapbox or Leaflet. However, if the SORA code is used for an internal program at Transport Canada, the research team may not want it hosted on the web.

**Conclusion**

Using the Python programming language and various geospatial libraries (cited in the Digital Appendix), four functions were created to successfully test Remotely Piloted Aircraft Systems flight risk in populated areas in the Ottawa region. The four functions run through a command line that requires an input area of interest from the user, assessing air to air risk and air to ground risk. Despite our project having a reasonable amount of limitations and setbacks, it was completed in working order for use by Brendan and his accompanying research team. In the future, a user-friendly GUI may assist in the utilization of the program, and assist in developing our individual Python skills. Though we had faced challenges during the production of the project, we were able to execute a Python script that meets our client’s wishlist, and will aid in the research performed by Brendan and his accompanying team at Transport Canada.

**Acknowledgements/Sources**

We would like to acknowledge our client, Brendan Ooi, and our professor, Derek Mueller. Completing the project was a challenging task but it helped in developing our Python skills. It also helped us learn the role of programming in the field of geomatics and introduced us to various geospatial libraries. Working with Brendan as our client has given us experience in working for a client, communication and completing a final product based on a client wishlist.

Our task was difficult and somewhat unique, there was not much code on the Internet to source or take inspiration from. The majority of the help we received came from the GEOM4009 course workshops. We also sought various websites, such as Python library pages. Websites Stack Overflow and Real Python assisted Python library pages when we were developing the code. Two main libraries that we used when creating the project were Pandas and GeoPandas. StackOverflow helped us in our efforts to overcome the issues with utilizing Fiona with KML. (Reference link: <https://stackoverflow.com/questions/55496283/reading-kml-file-with-multiple-folders-using-geopandas>)

**Digital Appendix**

Please see the zip file attached with the project submission on CULearn or see it on GitHub. (<https://github.com/Oliolipop11/09repo.git>)

**Appendix**

Figure 1. Updated UML diagram of project workflow