3D Spectrum Sensing Map via Drone Mounted Receiver

Karl Roush, github.com/karoush/droneSpectrumMap
Lincroft, NJ 07738 karlroush@gmail.org

I. Abstract

This project seeks to develop and test a low-cost, easy to use approach to signal mapping. The proposed objective will be accomplished by mounting an Android mobile device on a 3DR Solo drone. Data will be collected via a signal strength measurement app, outputted to a .csv file. The program Octave will be used to place the data points into three dimensional space. At this point, a custom written interpolation algorithm uses the nearest neighbor node signal strength theory to fill in the space where no data is collected. All of this data is then placed onto a 3D graph, for user end visualization. The applications of this project are fourfold. Firstly, this approach will allow for the identification of signal leakage beyond designated broadcasting bands. Secondly, this approach will allow an optimization of signal output based on nearby topography. Thirdly and fourthly, this approach will allow for ease of troubleshooting, as well as general diagnostics.

II. Introduction

In today's world, wireless connections are everywhere. However, there are almost no tools that produce an easy to use graphical display. In the United States, the FCC allows certain areas of the spectrum for certain purposes. Cable companies have one section, while the military has its own section. Signal leakage is when signals spill over from their designated band into adjacent bands. This can cause interference and disruption of other signals [51]. In a world of growing wireless connections, signal leakage is becoming more and more of an issue [6]. The approach and techniques proposed in this paper would assist in determining signal variability, strength, and leakage.

Beyond this, there is a general lack of portable and efficient spectrum mapping tools. This project proposes an approach and several techniques that when coupled together can greatly alleviate this issue.

III. Objective

Create a 3D signal quality map using a drone as a mobile receiver carrier. This will be accomplished through the use of a 3DR Solo drone (receiver carrier), mobile Android device (signal recorder), and Octave (graphical program with custom written interpolation algorithms).

IV. Implementation

A mobile receiver in the form of an android phone will be attached to a 3DR Solo drone. This drone will be released into the area of interest and follow a predetermined, uploaded flight plan. Upon completion of the flight, the signal data (comma separated value file) will be downloaded for offline analysis.

The data analysis program will have two purposes. The first purpose is to use a custom written interpolation algorithm to fill in the space where no data was collected. Since wireless signals follow predetermined mathematical formulae, the interpolated data will have a reasonably high degree of accuracy. The second purpose is to produce a graphical output to display the data at a given point in time.

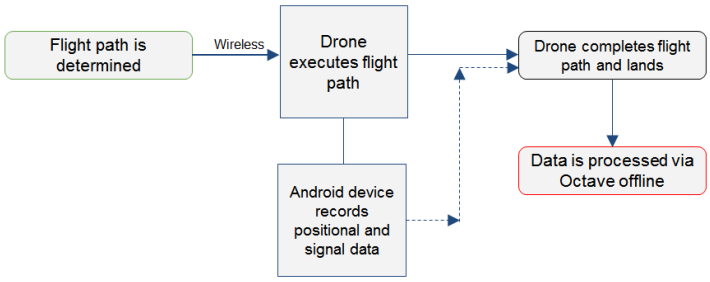

Figure 1: Simplified diagram of the various components and their interactions

V. Materials, Equipment and Programs

- Drone (3DR Solo)
- Mobile receiver (Android device)
- Data collection software (Network Signal Info app)
- Data processing program (Python/Octave)
- Computer with wireless connectivity

VI. Data Collection

Drone Flight Path

1. Determine area and height range where flight/data collection will occur
2. Open MissionPlanner program
3. Place waypoints in a designated path in the area
4. Link drone via wireless connection (UDP 14550; Port 500)
5. Export mission plan to drone

Data Retrieval:

1. Turn on Android device
2. Connect device to network being tested
3. Open "Network Signal Strength Info" app
4. Mount device to gimbal of the drone
5. Retreat to safe distance (~50m, varies by drone)
6. Initiate flight plan via MissionPlanner
7. Once flight is complete, drone will land and be ready for retrieval

VII. Data Availability

The Network Signal Info (NSI) app logs data every 30 seconds; however interval modifications are also supported. The user can change how often data is logged or have the app log data at certain distance intervals. NSI does not support altitude changes as a way of logging data, though it does record the device's position in three dimensional space. Regardless of the interval settings, the large quantity of data makes presentation within this paper difficult. As such, test data will be used in place of actual collected data.

VIII. Flight Pathing

Fight pathing is critical for optimal data collection. This can be accomplished by hand though it is tedious and often ineffective. There is a similar issue in the field of agriculture, specifically when it comes to placing seeds. The seeds must be as close as possible, but still have enough space between them to allow for watering and growth. Similarly, data needs to be logged at certain points in the area of interest- spaced out as far as possible to save time, but close enough to each other so that the interpolation algorithm is accurate.

Using the program MissionPlanner, a grid (called "auto grid") was generated for the area of interest. These way points were then stored in a mission file, which is uploaded to the drone via a wireless. This flight plan is then executed by connecting to the drone (via its own wireless network) and running the appropriate flight sequences.

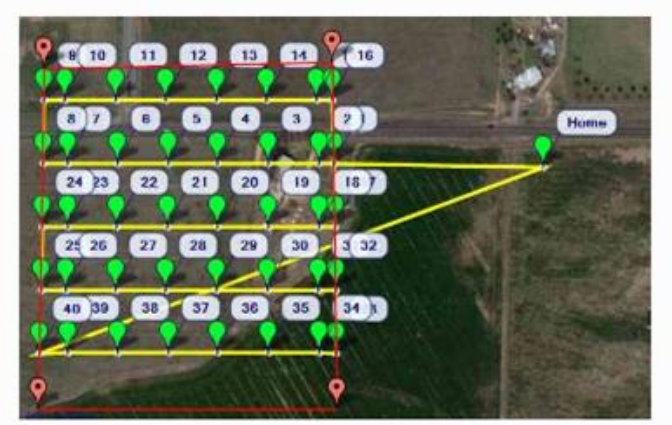

Figure 2: Mission planner auto grid [43]

IX. Data Analysis

Data from the .csv log file is read into Python. This data is then placed onto a three dimensional graph and shown to the user. High signal strength is represented as red and low signal strength is represented as blue. The space between data is filled in by generating additional data points through a custom written interpolation algorithm (see next section). Currently, there is no way to represent time on the graph (five variables on three axes presents an issue), but if developed for commercial use, this could be remedied by continually generating graphs for the user.

In its current state, the program for data analysis is written in Python 3.6 . However, it is in the process of being ported over into Octave. Octave is a free version of MATLAB and is prevalent in many engineering fields. As such, an Octave version of the program is both easier to modify and offers more data manipulation options.

Interpolation Algorithms

In the field of spectrum mapping and networks, there are three major interpolation algorithms: nearest neighbor, inverse distance weighting, and natural neighbor. Each has specific applications, but is usually used for varying degrees of accuracy (previously mentioned in increasing order).

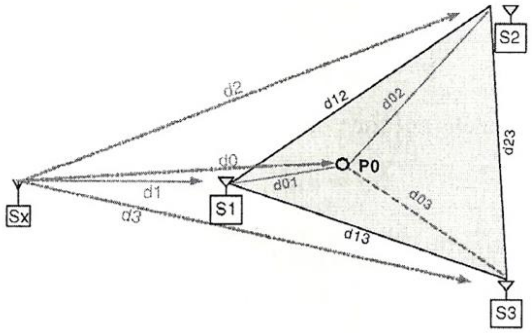

Figure 3: Diagram of sensor triangle (possible configuration of data points in 2D space) [62]

Nearest Neighbor

The point of interest is given the value of the next closest data point. In figure 3, P0 would be given the value of S1, because that is the closest data point.

Inverse Distance Weighting

The point of interest is given a value based on the distance for nearby data points. The value of P0 would be determined using the following equation:

$$P0 = \frac{S1}{d01} + \frac{S2}{d02} + \frac{S3}{d03}$$

Natural Neighbor

The point of interest is given a value based on the area of overlap between nearby data points. It is functionally similar to IDW, but with areas instead of distances.

Algorithm Selection

A modified nearest neighbor algorithm was selected for several reasons. The primary reason was lack of computing power. Python itself offers limited analysis of large quantities of data, but the computer used for processing also had limited processing power. The nearest neighbor algorithm was modified to use the least computing power because of this.

Of note, the points used by the algorithm were randomly generated. These points were then compared to the original data set, and then given the value of the closest point. Both the original data and generated values were then combined into one data set, which was used for the graphical output.

X. Graphical Output

At the most basic level, the output file is a three dimensional graph with different colors representing signal strength. Specifically, red is a stronger signal while blue is weaker.

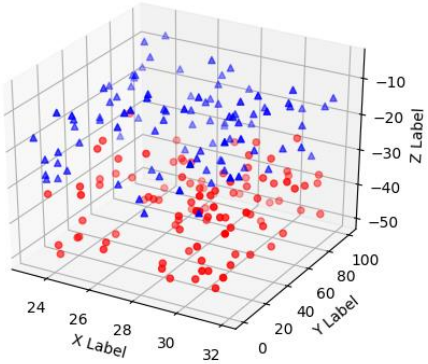

Figure 4: Sample graphical output of raw data with coloring by signal strength. [2]

However, through the usage of the custom written interpolation algorithm (modified nearest neighbor), a smooth three dimensional surface can be created. Given more computing power, it would be extremely easy to generate additional points via the interpolation algorithm, thus creating a smoother shape.

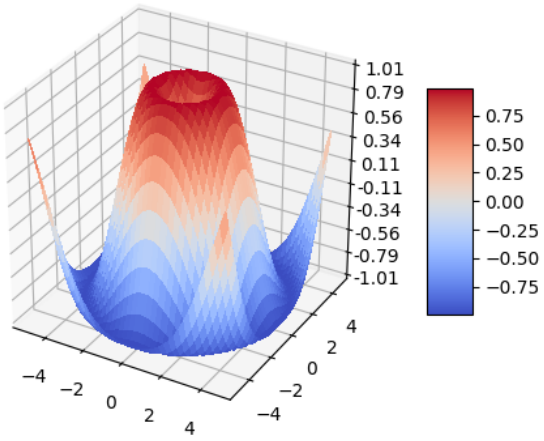

Figure 5: Sample graphical output of the set of both interpolated and actual data [2]

XI. Conclusions

The approach taken in this experiment proved to be feasible. Although limits on computing power produced

a less than optimal result, there is a clear proof of concept.

Of note, the custom written interpolation algorithm (modified nearest neighbor) sacrifices accuracy for speed and computing power. Consequently, the graphical output and any models produced from it would lack significant accuracy. Future iterations should opt to use the optimal algorithm for the appropriate available computing power.

XII. Applications

Aside from generating a visual representation of the spectrum and signal strength in an area of interest, this project has four major applications.

The first and perhaps the most obvious of the possible applications is the identification of signal leakage. In the United States, the FCC allows certain areas of the spectrum for certain purposes. Cable companies have one section, while the military has its own section. Signal leakage is when signals spill over from their designated band into adjacent bands. This can cause interference and disruption of other signals [51]. In a world of growing wireless connections, signal leakage is becoming more and more of an issue [6]. The approach and techniques proposed in this paper would assist in determining signal variability, strength, and leakage.

Secondly, the approach and techniques proposed in this paper would allow for an optimization of signal output. Signal output is not evenly distributed, nor does it need to be. Using the graphical output, one could easily redesign the signal source to be stronger in areas of interest.

Third and fourthly, troubleshooting and general diagnostics could be made easier. A graphical representation of the spectrum would greatly assist users in pointing out issues and consequently fixing them. The entire system is both easy to use and portable, further adding to its usefulness.

XIII. Further Research

Given the limitation of the project in both duration and computational power, only test data was implemented. Consequently, further research would include applying the methods to a larger data set and/or a more varied data set.

Additionally, other interpolation algorithms would be tested against the existing modified nearest neighbor. The benefits and downsides of each should be recorded so that one may more accurately choose one in the future.

References and Works Consulted

1. "14.1. Csv — CSV File Reading And Writing — Python 3.6.1Rc1 Documentation". Docs.python.org. N.p., 2017. Web. 15 Mar. 2017.
2. "3-D Scatter Plot - MATLAB Scatter3 Properties". Mathworks.com. N.p., 2017. Web. 15 Mar. 2017.
3. "3D Surface Plots". Plot.ly. N.p., 2017. Web. 15 Mar. 2017.
4. "Accessing Solo". 3DR Solo Development Guide. N.p., 2017. Web. 27 Feb. 2017.
5. Agbabiaka, Ahmed. "Connecting Mission Planner To 3DR Solo". YouTube. N.p., 2017. Web. 27 Feb. 2017.
6. "Cable Signal Leakage". Federal Communications Commission. N.p., 2017. Web. 15 Mar. 2017.
7. "Calibration". 3dr.com. N.p., 2017. Web. 27 Feb. 2017.
8. "Calibrations Process". 3dr.com. N.p., 2017. Web. 27 Feb. 2017.
9. "Chirp Spread Spectrum". En.wikipedia.org. N.p., 2017. Web. 27 Feb. 2017.
10. Churchward, Budd. "How To Set Up An SDR Radio". YouTube. N.p., 2017. Web. 27 Feb. 2017.
11. Community, The. "Function Reference: Linspace". Octave.sourceforge.io. N.p., 2017. Web. 27 Feb. 2017.
12. "Connecting To A Vehicle". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
13. "Direct-Sequence Spread Spectrum". En.wikipedia.org. N.p., 2017. Web. 27 Feb. 2017.
14. "Dronekit-Python API Reference". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
15. "Dronekit-Python API: Air Speed". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
16. "Dronekit-Python API: Capabilities". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
17. Eaton, John. "GNU Octave: Documentation". GNU. N.p., 2017. Web. 27 Feb. 2017.
18. "Example: Basic Mission". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
19. "Example: Guided Mode Movement And Commands (Copter)". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
20. "Example: Mission Import/Export". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
21. "Example: Simple Go To (Copter)". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
22. "FAQ - Octave". Octave Wiki. N.p., 2017. Web. 27 Feb. 2017.
23. "Frequency-Hopping Spread Spectrum". En.wikipedia.org. N.p., 2017. Web. 27 Feb. 2017.
24. "Function Reference: Scatter3". Octave.sourceforge.io. N.p., 2017. Web. 15 Mar. 2017.
25. "GNU Octave: Multiple Plots On One Page". Gnu.org. N.p., 2017. Web. 27 Feb. 2017.
26. "GNU Octave: Object Sizes". Gnu.org. N.p., 2017. Web. 27 Feb. 2017.
27. "GNU Octave: Three-Dimensional Plots". Gnu.org. N.p., 2017. Web. 27 Feb. 2017.
28. "GNU Octave: Two-Dimensional Plots". Gnu.org. N.p., 2017. Web. 15 Mar. 2017.
29. "Guided Tutorial GRC". Gnuradio.org. N.p., 2017. Web. 27 Feb. 2017.
30. "Guided Tutorial Introduction". Gnuradio. N.p., 2017. Web. 27 Feb. 2017.
31. "Guiding And Controlling Copter". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
32. Honiball, Rob and emilio mendoza. "Can't Connect To Solo With Mission Planner". 3D Robotics Drone Forum. N.p., 2017. Web. 27 Feb. 2017.
33. Hutchinson, Ian. "Octave/MATLAB® For Beginners, Part 1: Starting From Scratch". Ocw.mit.edu. N.p., 2017. Web. 27 Feb. 2017.
34. "Installing Files And Code". 3DR Solo Development Guide. N.p., 2017. Web. 27 Feb. 2017.
35. "Installing GR". Gnuradio.org. N.p., 2017. Web. 27 Feb. 2017.
36. Linge, Svein and Hans Peter Langtangen. "Programming For Computations - A Gentle Introduction To Numerical Simulations With MATLAB/Octave". Github.io. N.p., 2016. Web. 15 Mar. 2017.
37. Long, P.J.G. "Introduction To Octave". University of Cambridge, Department of Engineering. N.p., 2017. Web. 27 Feb. 2017.
38. Martin, Rich. "The Case For Transmit Only Communication". RU Department of Computer Science. N.p., 2017. Web. 27 Feb. 2017.
39. "MATLAB Quick Guide". www.tutorialspoint.com. N.p., 2017. Web. 27 Feb. 2017.
40. "Mplot3d Tutorial". matplotlib. N.p., 2017. Web. 15 Mar. 2017.

41. "Octave Programming Tutorial/Getting Started - Wikibooks, Open Books For An Open World". En.wikibooks.org. N.p., 2017. Web. 27 Feb. 2017.
42. "Octave Support/Help". Octave. N.p., 2017. Web. 27 Feb. 2017.
43. "Octave Tutorial #4 - Plotting Data". YouTube. N.p., 2017. Web. 27 Feb. 2017.
44. "Planning A Mission With Waypoints And Events — Mission Planner Documentation". Ardupilot.org. N.p., 2017. Web. 27 Feb. 2017.
45. "Quick Start". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
46. "Quick Start Guide". rtl-sdr.com. N.p., 2017. Web. 27 Feb. 2017.
47. Redford, Chris. "Displaying Filled Markers With Scatter3 In Octave While Using A Colormap". Stackoverflow.com. N.p., 2017. Web. 15 Mar. 2017.
48. "RTL-SDR Tutorial: Receiving NOAA Weather Satellite Images". rtl-sdr.com. N.p., 2017. Web. 27 Feb. 2017.
49. "Running The Examples". 3DR Solo Development Guide. N.p., 2017. Web. 27 Feb. 2017.
50. "Running The Examples". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
51. "SCTE Technical Report". Society of Telecommunication Engineers. N.p., 2015. Web. 15 Mar. 2017.
52. Seeber, Balint. "Hacking The Wireless World With Software Defined Radio - 2.0". Black Hat Europe 2014. N.p., 2017. Web. 27 Feb. 2017.
53. "Software Defined Radio (SDR) - RTL-SDR". YouTube. N.p., 2017. Web. 27 Feb. 2017.
54. "Software Radio Basics". YouTube. N.p., 2017. Web. 27 Feb. 2017.
55. ""Solo" Command Line Tool". 3DR Solo Development Guide. N.p., 2017. Web. 27 Feb. 2017.
56. "Solo Compass Calibration How To?". 3D Robotics Drone Forum. N.p., 2017. Web. 27 Feb. 2017.
57. "Spectral Flux Density". En.wikipedia.org. N.p., 2017. Web. 27 Feb. 2017.
58. "Spread Spectrum". En.wikipedia.org. N.p., 2017. Web. 27 Feb. 2017.
59. "Summer SDR Spectrum". GitHub. N.p., 2017. Web. 27 Feb. 2017.
60. "Taking Off". Python.dronekit.io. N.p., 2017. Web. 27 Feb. 2017.
61. Talbert, Robert. "Plotting Functions Of Two Variables In MATLAB (Part 2)". YouTube. N.p., 2017. Web. 27 Feb. 2017.
62. "Time-Hopping". En.wikipedia.org. N.p., 2017. Web. 27 Feb. 2017.
63. Zhang, Yanyong. "Transmit Only For Dense Wireless Networks". Rutgers WINLAB. N.p., 2017. Web. 27 Feb. 2017.
64. Zhao, Xuefeng. "Path Loss Estimation Algorthims and Results for RF Sensor Networks". Rutgers WINLAB. N.p., 2017. Web. 27 Feb. 2017.