Reflectivity data Spike-buster

Pushkar Kumar Jain

February 18 2015

The document deals with the removal of sunspikes from reflectivity data mosaiced from the NEXRAD radar network using an automated Python script. The objective is to develop a proof-of-concept to remove the sunspikes, with the rest of the data intact. The document deals with a brief description of the capability of the Python script and the algorithm followed to achieve the result. The input data is in GeoTIFF format and a suitable library has been used to extract the appropriate data for analyses. The Python script is supported by multiple libraries.

1 Introduction

As given in Ref [1], in addition to precipitating particles, echoes on radar may be due to biological targets such as insects, birds or wind-borne particles, due to anomalous propagation (AP) or ground clutter (GC) or due to test and interference patterns that inadvertently seep into the final products. However, it is possible to identify, and account for, the presence of such contamination via automated weather radar algorithms. Neural networks have been used widely for such applications (Ref [2]).

This document (as given in Section 2) presents an algorithm that was followed to achieve the despiking of the reflectivity data. Python script has been developed for preprocessing, execution and post-processing of data. The image processing technique, using filters and 'erosion' has been used. Since, the knowledge in the field is relatively low, effort has been put to adopt the literature into automated script. Section 3 describes the code structure and will further state the dependencies in Section 3.1. Section 3.2 explains the code exectuion followed by results in 4. Various conclusions are drawn with some ideas that can be implemented in Section 5.

2 Algorithm

WIth minimum prior knowledge in the area of image processing, literature in the field was implemented with basic knowledge from various blogs. The technique can be broken down into following steps -

1. Read GeoTIFF image
2. Find the maxima and the minima value in the array
3. Normalize the data to [-1, 1] by using a linear scaling function.
4. Use percentile mean filter to only use values between percentiles $p0$ and $p1$ (here 10% and 90%). The filter smooths the image background and details so that next process can be specifically implemented on the area of interest, that is, the reflectivity data regions.
5. Use morphological erosion approrpirately.to set a pixel (i, j) to the minimum over all the pixels in the neighborhood centered at (i, j). The structuring element, *selem*, passed to erosion is a boolean array that describes this neighborhood and thus decides the disk radius.
6. Normalize the data back to minima and maxima of the original image, by using linear scaling function.
7. Compare the original and the final image

This method was chosen as it helped to 'farm' the processes to existing modules in scimage ibrary, thereby reaching the goal in the shortest time possible. This obviated the need to handle the data sets manually and implement some specific algorithm like neural networks etc. However, it is to be noted that, the latter is a better approach as it provides flexibility and broader area for new algorithm development.

3 Code structure

The Python code has been packaged, The root directory of the script contains -

- *doc* - Contains the documentation
- *src* - Contains the source code with the main file *pymain.py.* It also contains the plotting tools.
- *input* - Contains the input GeoTIFF file
- *output* - The python execution generates output and plot that is stored in this directoty

3.1 External libraries

The code employs multiple externaml libraries. These are listed as -

- **numpy** - To implement array operations
- **scimage** - Image processing toolbox with various functionalities including application of filters (`http://scikit-image.org/`)

- **docopt** - Command line argument parser (`http://docopt.org/`)
- **matplotlib** - Plotting tool for Python
- **GDAL** - Geoprocessing data library to read in GeoTIFFF file. (`http://www.gdal.org/`)

3.2 Code execution

The *README.rst* states the way to run the Python script. In the source directory, execute

```
$ python pymain.py IN_FILE
```

, where IN_FILE is the path to input TIFFF file. Running status of the script is provided, in addition to robustness in form of system exit in case of any run time error.

Four values have been hard-coded into the script - *ScreenOut*,*selem*, *po* and *p*1(discussed in algorithm in Section 2).

ScreenOut is by default switched off. Different values of *selem* ranging from 10 to 25 were tried inorder to achieve the best result. It is to be noted that, the conclusion of reaching the 'best' result was solely visual aided. No particular algorithm was developed to check the deviation of the result with teh original image. This is open in future. Some technique might be to find the L^p norm of the dataset. Similarly, $p0 = 0.1$ and $p1 = 0.9$ values were taken as they were the default values in the function call.

One possible technique to obtain the best result is to choose a suitable combination of these three values.

HTML pages via Sphinx documentation can be generated by executing the following command in *doc* directory. The HTML page can be accessed in *doc/_ build/index.html*.

```
$ make html
```

4 Results

On executing the code, the reult given in Figure 1 is obtained. The regions of interest, that is, the reflectivity data was extracted from the base image. This was followed by the removal of the spikes via erosion. As stated earlier, the deviation can be quanitifed using appropriate error norms.

Currently, the output is stored as *a.png*. Writing output to TIFFF format was ventured but unsucessful. This can be implemented with further availibility of time.

5 Conclusion

An automated Python script has been developed to perform the despiking of the reflectivity data. Proof of concept approach was used to implement the algoithm via image processing. The result clearly shows the spikes have significantly reduced, but with a

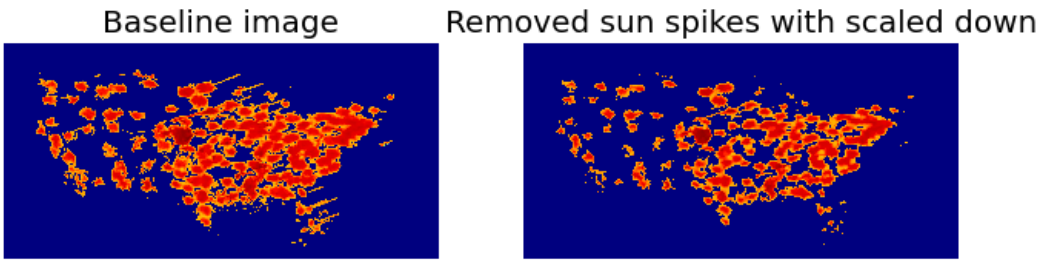

Figure 1: Comparison of the original image with the final image obtained by implementing the stated algorithm

compromise on the reflectivity values in other areas. There is a need to quantify this deviation so that the best combination of $selem$, $p0$ and $p1$ can be chosen. There is a large scope for improvement. Various algorithms from literature can be implemented that handle the array data manually, rather than applying filters. This will provide more flexibility in the form of customization and calibration.

The current code execution takes significant time in the erosion process. This can be reduced by decreasing the dimension of the the input array. One possible technique is to compress the image using single value decomposition technique to remove the 'non-essential' pixels.

The Python package is structured with documentation. Further, various libraries have been used to make the task easier. The plotting utilities can be upgraded to accomodate additional filters.

References

[1] Mazur, Rebecca J., V. Lakshmanan, and Gregory J. Stumpf. "Quality control of radar data to improve mesocyclone detection." Preprints, 20th Int'l Conf. on Inter. Inf. Proc. Sys.(IIPS) for Meteor., Ocean., and Hydr.,(Seattle) P. Vol. 1. 2004. APA

[2] Lakshmanan, Valliappa, et al. "An automated technique to quality control radar reflectivity data." Journal of applied meteorology and climatology 46.3 (2007): 288-305. APA