Layer Analyzing

Team Members: Ken Luo, Christopher Nguyen, Kimberly Reyes, Edwin Ruiz, Elizabeth Sanchez, Alyssa Solon

Faculty Advisors: Dr. Elaine Kang, Dr. Yuqing Zhu

JPL Liaisons: Emily Law, Shan Malhotra, Mike Rueckert, George Chang

Department of Computer Science

College of Engineering, Computer Science, and Technology

California State University, Los Angeles

Table of Contents

Introduction

1.1 Background

Jet Propulsion Laboratory (JPL) is a federally funded research facility and development center managed for NASA by Caltech that carries out robotic space and Earth science missions. JPL has conducted robotic missions to study all the planets in the solar system as well as asteroids, comets and the Earth's moon. Today JPL continues its world-leading innovation, implementing programs in planetary exploration, Earth science, space-based astronomy, and technology development.

There have been various missions to the moon such as Apollo 15, 16, and 17 just to name a few. Through those missions, data about elements and terrain were gathered. Some instruments were more precise in gathering information and thus produced larger data files. Some data was only collected from a landing site while others were gathered from the entire moon. That also contributes to disparity of the data files.

Through the missions involving collecting data from Earth's moon JPL has created Moon Trek, a mapping and modeling portal for the Moon. There are over 2500 layers available on Moontrek, which store information about the moon such as elevation, element concentration, and roughness. Not all of the layers contain raw data which can be analyzed. Moreover, Moontrek does not allow for statistical analysis of layers besides the elevation layer. Visuals of data through graphs is also lacking. To address this issue, JPL is partnering with California State University - Los Angeles, College of Engineering, Computer Science, and Technology to create an analytical tool consisting of functions that will analyze a single or multiple layers. Such calculations will include minimum, maximum, average, median, standard deviation, variance, histogram, etc. for a single layer. Analysis of multiple layers will include correlation, covariance, clustering, and visual representation of the results. Such analysis will provide more insight about the Moon and the elements there to the user and perhaps more importantly to scientists.

1.2 Limitations and Challenges

The JPL-2 Layer Analysis project seeks to provide scientists and users of the Moontrek website an easy to understand way to view data from various scientific data layers of the Moon. Moontrek is a website developed by JPL that allows web users to view and compare different data layers collected from various scientific experiments. The limitation of the website is that currently there is no feasible way to extrapolate deep analysis from the current methods of displaying the layers. As a result of the limited features of Moontrek, JPL has requested that our project creates functions that allow for processing large amounts of data and provides the ability to display this data in a understandable manner in both two and three dimensional graphs. The current system that our project has developed has reached the requirements set by JPL, and as a result can effectively display various graphs from user-selected layers in both two-dimensional and three-dimensional, interactive graphs.

1.3 Objectives and Features

The current test system that we have developed can receive geoTIFF or xyz files, convert them to dataframes, perform functions on these dataframes such as calculating simple statistical information, find correlations, and perform more complicated machine learning algorithms such as clustering, and finally visualize this information in graphical displays that is easy to understand for the user. The project has a number of back-end functions that aid in these tasks, including k-means algorithms, spatial reference conversion functions, and sub-sampling functions. The project design principles are to be able to process large amounts of data without crashing the program, doing it efficiently enough that processing time remains reasonable, and displaying the results in a way that is beneficial to the users. The main benefit from our developed system compared to the older Moontrek display methods is that the user has a variety of graphs that visualize the data in a easy-to-understand way, allowing for the user to see trends and patterns at a glance, while also giving interactivity to some of the graphs allowing for rotation, zoom-in, and detailed point selection. In order for us to test the functions effectively and efficiently, the project shifted over from a non-GUI, purely function testing project to creating an integrated system of functions that utilizes a Command Line Interface. From our testing results, we have been able to create and display graphs in reasonable time without crashing the program using back-end functions such as sub-sampling.

Related works and technologies

2.1 Python and its Libraries

The project used many existing Python libraries as well as Jupyter Notebook for much of the testing before the creation of the integrated system we currently have. Many of the libraries provide much needed functionality and utility that this project's foundation is on. Pandas is a Python library that allows geoTIFF and xyz files to be converted into dataframes which allow data manipulation to be performed on large amounts of data. MatplotLib is a fundamental part of many of the graphing functions that we use to display our data, mainly the two-dimensional data as well as when required to graph large amounts of data points in three-dimensions. However, MatplotLib does not allow the rotation and zooming in of three-dimensional graphs, and for that purpose we use Plotly. Plotly allows the graphing of two and three-dimensional graphs much like MatplotLib, but also allows for the manipulation of the graphs such as zooming functions, rotation, point hiding and a plethora of other functions. The reason we did not switch all our graphs to Plotly is due to how the library utilizes openGL to render each point individually. For small samples of data this leads to acceptable performance and rendering time, but for large amounts of data it can take much longer and could even crash the program. For this reason, we utilize Plotly graphs only where the graph would require the enhanced interactivity, and also run Plotly alongside our sub-sampling function to ensure optimal performance.

2.2 Jupyter Notebook and CLI

Other technology the project utilized was for our testing purposes, such as Jupyter Notebook and later on CLI integration. Jupyter Notebook allowed for the project members to run specific lines of code within the larger files instead of the entire file, leading to effective testing by minimizing runtimes. As the function list grew larger, Jupyter's ability to modularize code became a very important part of testing. As our requirements began requiring CLI interaction, we then began using libraries for Python CLI integration. Unfortunately, we came across issues using libraries readchar and inquirer. As readchar was only supported on UNIX systems, and inquirer required readchar, the Windows project members were unable to provide any testing on the CLI. Virtual Machines were considered to allow Windows project members to help test the CLI, but the issue appeared close to the end of our development and ultimately unused.

2.3 Implementation of UI

Implementation began as hard-coded functionalities that began to modularize as more project members became familiar with Jupyter Notebook. Function modularity became even more important throughout the development cycle as we moved from taking specific inputs to generalizing our inputs, as well as during the transition to using a CLI-based system. As we reached the end of our development period, most of the work was integration of the functions into the CLI wrapper, and a requirement for that was to modularize functions and generalize inputs. The finished project is now nearly 100% dynamic, requiring very minimal input from the user besides what files to input, what functions to run, and other options such as data normalization. Functions that began as hardcoded requirements, such as sub-sampling or graph selection are dynamically selected through statements checking specific parameters and details of the input.

System Architecture

3.1 DFD

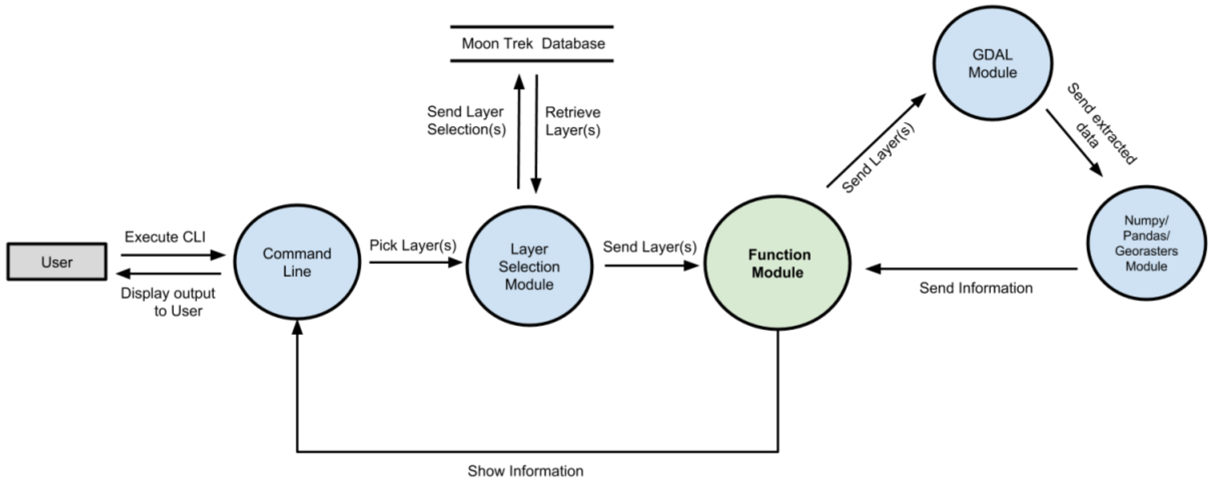

Figure 1Context Diagram (Level 1 DFD)

The system architecture of our project works with command line. It was developed to visualize project results without integrating to Moontrek.The user executes the Command Line Interface and it is prompted with commands throughout execution. The user is presented with a layer selection and function selection. Once the layer is selected it goes into the function module, which reads in Tif files using Gdal, once the data is extracted, the other libraries process the information and send it back to function module and back to the command line to display the output back to the user.

3.2 Opening files/ converting to Pandas dataframe

GDAL was used to obtain the geographical information from the GeoTIFF files. It provided information such as coordinates system, pixel size, origin, and no data values.

Pandas was the library we used to read in the .xyz files and extract data. The GeoTIFF files were a bit different as they required both Pandas as well as Georasters. Georasters was used to read in the file’s content then passed over to Pandas to create the dataframe. Pandas was used to organize our different value types. NumPy was only able to hold one type of value per array while Pandas is able to hold multiple. Being Python's numerical computation library, NumPy's primary advantage is that it contains certain useful functions such as min, max, standard deviation, and mean.

3.3 Preprocessing

3.3.1 File to Dataframe

The system has various functions that preprocess the data to run analytical functions and output their results. The first step is the most important and mandatory regardless of data, and that is the conversion of the geoTIFF or xyz file’s data containers into a Pandas dataframe.

Transferring the data into a dataframe allows the rest of the system to perform additional preprocessing as well as the user selected functions.

3.3.2 Conversion

The second preprocessing step is to ensure a common spatial reference for merging. Each file will be converted into separate dataframes, then merged together along the latitude and longitude values they provide. In the case that the spatial references provided with the files are not in latitude or longitude, a conversion function exists to take the existing spatial reference from the file and convert it into the appropriate one. Once that is complete, then dataframe merging can occur.

3.3.3 Removal of NaN values through Interpolation

Not a Number values, or NaN values, will throw off statistical analysis as well as make visualizations distorted or non-functional, and as such they must be removed from the data. The system uses interpolation to created modified dataframes for use in future functions.

3.3.4 Normalization

An optional step depending on the file and user selection, normalization sets the values of the dataframe to one common scale in order for the data to be read more easily by our functions. The user is allowed to select whether or not they want to normalize each layer selected, and a normalized version of the dataframe will be created for future functions if desired.

3.4 Functions

3.4.1 Layer Selection

The user selects the layer of interest through the interface which is currently provided with : Thorium, Iron, Hydrogen, Elevation (LOLA) and Temperature (spanning 24 hours)

3.4.2 Function Selection

This module contains multiple functions which compute calculations on the selected layers depending on what function the user selects. The available functions for implementation are Stats (Average, Standard Deviation, Minimum and Maximum Value, and Median Value), Covariance, Clustering, Logarithm and Correlation Graphs, Scatter Plot, Plotting x values vs y values, Histogram, 3D Plot. and Visualization Graphs

Results and Conclusions

4.1 Results

Throughout the project we were successfully able to deliver a system through command line that helps the user get results from the layers of the moon and cross examine data from various layers.

4.2 Successes and Challenges

Overall we met our objective, it was a success we were able to deliver our clients requirements. We did encounter some challenges throughout the project. Such as integrating various functions to the command line, converting to decimal degrees since there was no EPSG for converting coordinates. As well as using machine learning algorithms, removing the NaN values and creating a contour plot.

4.3 Follow up

We had future plans to the project, which is being able to select all 2500 layers from Moontrek. As well to select Multiband files, since we currently working on a single band.

References

Visualizing Distributions of Covariance Matrices Document
http://www.stat.columbia.edu/~gelman/research/unpublished/Visualization.pdf

Technologies used

Python - https://www.python.org/download/releases/3.0/

Jupyter Notebook - http://jupyter.org/install

Matplotlib - https://matplotlib.org/

Georasters - http://georaster.readthedocs.io/en/latest/

GDAL - http://www.gdal.org/

Numpy - http://www.numpy.org/

Pandas - https://pandas.pydata.org/

Inquirer - https://pypi.python.org/pypi/inquirer