**CS496 Senior Design**

**Salient Feature Extraction from Planetary Images**

**Project Documentation**

Document Prepared by:

Adam Berman

Andy Tong

Benjamin Evans

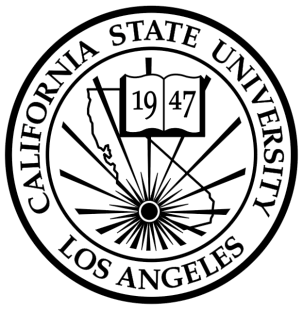
Bruno Recillas

Kevin Casado

Lineli Hairapetian

27 May 2016

CALIFORNIA STATE UNIVERSITY LOS ANGELES



Los Angeles, California

**Table of Contents**

1.0 [Executive Summary](#id.ips8xotqov3c)

2.0 [Introduction](#id.p02j2p7igiy7)

3.0 [Installation/User Guide](#id.k0aeufih2kyh)

1. [Installation Manual](#id.2o5i89wlhkrk)
2. [User Guide](#id.6syj8numq1kt)

4.0 [Architecture and Design](#id.hbd9qrvebwqj)

1. [Ringtoss Pipeline](#id.7xd6mu7pfmil)
2. [Modifications to Ringtoss](#id.46ufj0v8ueex)

5.0 [Conclusions](#id.qna3mss0upev)

# EXECUTIVE SUMMARY

The Salient Feature Detection from Planetary Images project’s goal was to extend an existing system called Ringtoss. Ringtoss was designed last year to detect craters in images of lunar terrain. Our goal was to improve Ringtoss to allow it to function on images of Martian terrain.

The first quarter was spent getting up to speed with the existing implementation, as well as setting up the (rather complex) system on our test machines. We ran into several difficulties in the process, and spent a significant amount of time presenting to each other on the specifics of each algorithm. Team members were paired off and sent away to study in depth each of the three detection algorithms used by the previous year. In addition, time was spent reworking the image-loading code, so it would load appropriate images on request, pulled from the Mars HiRISE dataset.

Once we have Ringtoss running on Martian images, it was time to begin improving its performance. Each pair worked for several weeks individually on their chosen algorithm, analyzing where and why it was succeeding and failing on a variety of Martian images. Template matching got new templates, Circle Hough tested a wide variety of parameters, as did Ellipse Fitting. The entire team also put together a Ground Truth dataset of hand-selected craters from Martian imagery, to measure the performance of Ringtoss. This consisted of some 14 images containing several hundred craters. This process was painstaking, requiring precision if we were to measure our results accurately. We also began to analyze the neural network used to classify the detected features, looking at how it worked and how it could be improved.

The final three months of the project were spent scrambling to finish the changes we’d decided upon in winter quarter. With the detection algorithms worked on, we turned our focus to the neural network. Training sets, a constant, low-level concern throughout the year, came to the forefront and thousands of crater images were sifted through. In the end, we continued to use training data generated from our Ground Truth, but a variety of other approaches were tested. Other changes included implementing a two-phase recognition process that ran the neural network with both the old, lunar training and our new, Martian training set, hoping to separately capture what could not be directly combined. Bugs were found (some of them very well hidden) and fixes made. Additionally, a new feature was added, late in the process; a Hazard Mapping tool that uses the feature detection results to mark areas as more or less safe for landings.

This Ringtoss pipeline is the culmination of our nine months work. Our results were are not groundbreaking, but they do show some promise. The problem of identifying craters programmatically from images is a tough one; human experts sometimes have trouble doing it. While there was no great leap forward in performance of Ringtoss this year as compared to last year, the forward progress made this year is all in the right direction.

# INTRODUCTION

Crater detection and avoidance is critical for spacecraft landings and mapping of Mars terrain. Resources for crater detection are limited and new approaches are always welcome. The Salient Feature Extraction from Planetary Images team at JPL tasked our team to improve the existing application so it will detect craters on Mars images. Our approach to this problem was to improve each one of three algorithms, so they will detect all craters in the image. After having a great detection, we had plan on improving the recognition phase.

The application begins by having a user select a Mars image, Ringtoss then downloads an image from repository. This image is then processed by our three different detection methods. Each crater is added to a list, which undergoes further processing to remove duplicate detected craters. The list is then provided to a machine learning algorithm for improved detection and recognition and Hazard map. Finally, depth and diameter are calculated for each detected crater. Results are provided in text output, a result image will all craters detected and a hazard map.

# Installation/User Guide

**3.I Installation Manual**

**Software Requirements:**

* Java 8 or greater
* GDAL with python bindings
* OpenCV 2.4.9 or 2.4.10
* MySQL 5.6 or greater
* Python 2.7 (for gdal)
* Octave
* USGS ISIS

In order to install the prerequisite software to run Ringtoss simply run the **rintoss\_install.sh** provided. The script has been successfully tested on Ubuntu 14.0 but may need to be edited for other flavors on linux.

**Problems that may occur when running the script:**

1. Fail to setup mysql database user

If during the install the user enters the wrong password for the mysql root user you will have to create open mysql as the root user and issue the following commands before attempting to running ringtoss:

*CREATE DATABASE hirise;*

*CREATE USER 'lipfd'@'localhost' IDENTIFIED BY 'lipfd';*

*GRANT ALL PRIVILEGES ON hirise.\* to 'lipfd'@'localhost';*

1. USGS ISIS failure

After running the Ringtoss installation script the following command should be recognized : hi2isis

If it is not recognized you must manually set an environment variable (call it ISISROOT) to point to where ISIS is installed on the system as well as run the startup script produced when ISIS in installed (located in the scripts folder on the ). If you continue to run into problems you can simply reinstall ISIS by running the **isisInstall.sh** script provided. Since the ISIS startup script must be run before you can use any of the ISIS commands it’s best to add the environment variable to your .bashrc file as well as run the startup script. These can be placed at the bottom of the file; so the last few lines should look something like the following:

*ISISROOT=~/Isis/isis*

*export ISISROOT*

*. $ISISROOT/scripts/isis3Startup.sh*

Once installation is done, the image metadata must be inputted into the local MySQL database before Ringtoss can start processing images. User must download:

[EDRCUMINDEX.TAB](http://hirise-pds.lpl.arizona.edu/PDS/INDEX/EDRCUMINDEX.TAB) - a file containing metadata of Mars images

This file should be placed in the **image-metadata** directory. After doing so run Ringtoss with the **run\_create\_database** parameter set to yes.

Images that are supported with the above metadata can be found here:

<http://hirise-pds.lpl.arizona.edu/PDS/EDR/>

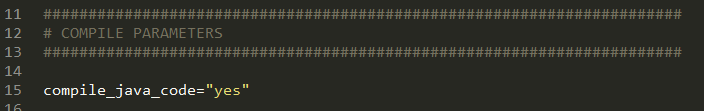
**3.II** **User Guide**

In order to run Ringtoss simply execute the mipfd bash file located in the root directory. Inside the root directory you will find numerous directories.

|  |  |
| --- | --- |
| **Directory** | **Description** |
| cnn | Contains source code for the convolutional neural network used by Ringtoss |
| ground-truth | Contains text files representing the actual craters for a given image |
| src | Contains the source code for Ringtoss (mainly .java files) |
| results | Contains image and text results outputted by Ringtoss |
| libs | Contains mysql connector jar |
| data-files | Contains the .mat files used |
| downloadedImages | Contains the images processed by Ringtoss |
| image-metadata | Contains the .TAB file that holds all image’s metadata |
| negativeImageData | Contains text files representing features within an image that are not craters for the particular image |

**Compile Parameters:**

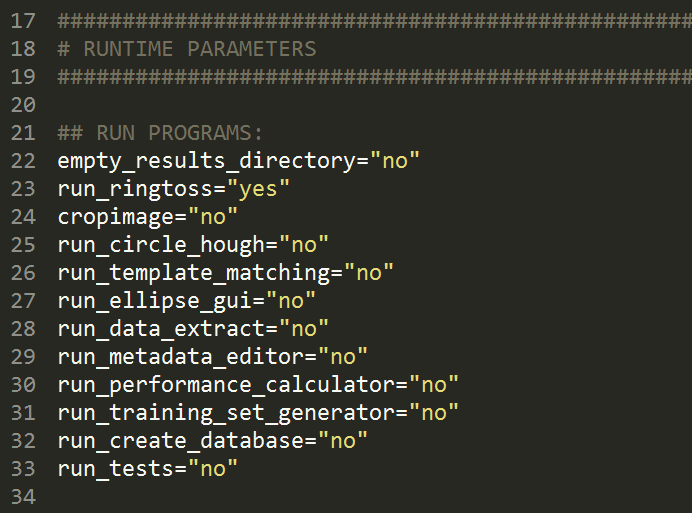
The “mipfd.sh” bash file provides a parameter to compile java code:



Set it to “yes” to compile or “no” to not compile.

**Runtime Parameters:**

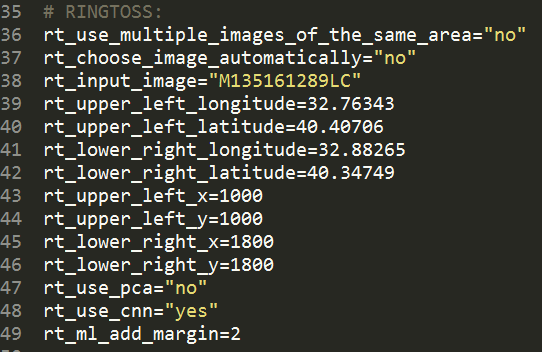
There are various runtime parameters that give an option to run the main Ringtoss program or the individual components of the Ringtoss program, including Circle Hough, Ellipse Fitting, and Template Matching. The following runtime commands take the input “yes” and “no” to be executed.



|  |  |  |
| --- | --- | --- |
| Parameter | Descrition | Values |
| empty\_results\_directory | Will empty results directory before running ringtoss | “Yes”  “No” |
| run\_ringtoss | Runs ringtoss main program | “Yes”  “No” |
| cropimage | Will crop the desire coordinates from the input image | “Yes”  “No” |
| run\_circle\_hough | Runs an independent circle hough GUI | “Yes”  “No” |
| run\_template\_matching | Runs template matching independently | “Yes”  “No” |
| run\_ellipse\_gui | Runs an independent ellipse fitting GUI | “Yes”  “No” |
| run\_data\_extract | Runs a GUI for crater depth and diameter | “Yes”  “No” |
| run\_metadata\_editor | Runs an editor that allows user to highlight craters or update them | “Yes”  “No” |
| run\_performace\_calculator | Computes f1 score | “Yes”  “No” |
| run\_training\_set\_generator | Generates a training set for machine learning portion of ringtoss | “Yes”  “No” |
| run\_create\_database | Creates a database with the Mars images metadata | “Yes”  “No” |
| run\_tests | Runs tests | “Yes”  “No” |

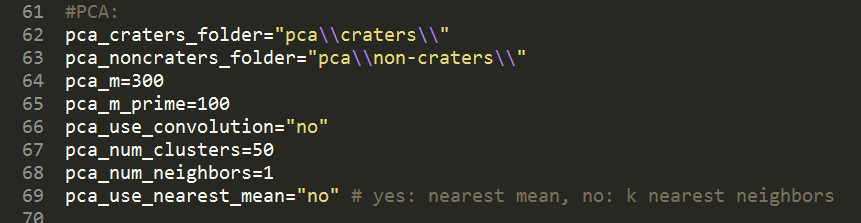
**Note:** Crater detection algorithms must run first and store results in a text file. Data-Extract will work best when running together with Ringtoss, or after a list of detected craters is provided from a crater detection algorithm and stored in the results folder.

**Ringtoss Parameters:**



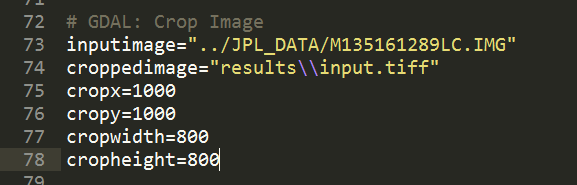
|  |  |  |
| --- | --- | --- |
| Parameter | Description | Values |
| rt\_use\_multiple\_images\_of\_the\_same\_area | Allows user to use multiple images at once. (Not implemented) | “Yes” (string)  “No” |
| rt\_choose\_image\_automatically | Chooses a random image | “Yes” (string)  “No” |
| rt\_input\_image | Image name on the database | String |
| rt\_upper\_left\_longitude | Upper left longitude of the input image | Double |
| rt\_upper\_right\_longitude | Upper right longitude of the input image | Double |
| rt\_lower\_right\_longitude | Lower right longitude of the input image | Double |
| rt\_lower\_right\_latitude | Lower right latitude of the input image | Double |
| rt\_upper\_left\_x | Upper left x pixel value of the input image | Integer |
| rt\_upper\_left\_y | Upper left y pixel value of the input image | Integer |
| rt\_lower\_right\_x | Lower right x pixel value of the input image | Integer |
| rt\_lower\_right\_y | Lower right y pixel value of the input image | Integer |
| rt\_use\_pca | Runs principal component analysis PCA | String |
| rt\_use\_cnn | Runs convolutional neural network | String |
| rt\_ml\_add\_margin | Increase the bounding boxes twice the original size for machine learning | Int |

**PCA Parameters:**



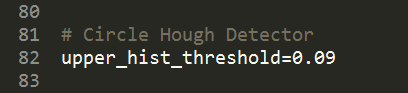
|  |  |  |
| --- | --- | --- |
| Parameter | Description | Type |
| pca\_craters\_folder | Directory for crater training images folder |  |
| pca\_noncraters\_folder | Directory for non-crater training images folder |  |
| pca\_m\_prime | Directory for non-crater training images folder m prime value |  |
| pca\_use\_convolution | Uses convolution | String ‘Yes/No’ |
| pca\_num\_clusters | Number of clusters |  |
| pca\_num\_neighbors | Number of neighbors for nearest neighbors |  |
| pca\_use\_nearest\_mean | mean | “yes/no” |

**GDAL Parameters:**



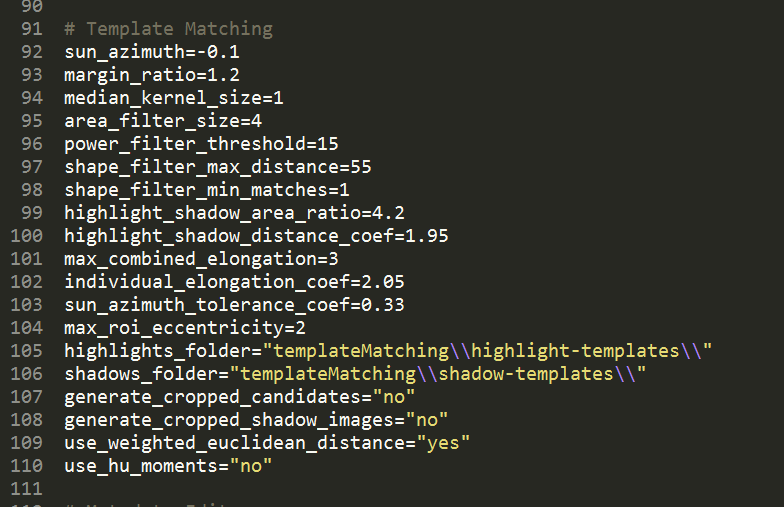
|  |  |
| --- | --- |
| Parameter | Description |
| inputimage | Directory of image to crop |
| croppedimage | Directory and name of resulting cropped image |
| cropx | X pixel offset |
| cropy | Y pixel offset |
| cropwidth | Width of cropped image |
| cropheight | Height of cropped image |

**Circle Hough Parameters:**



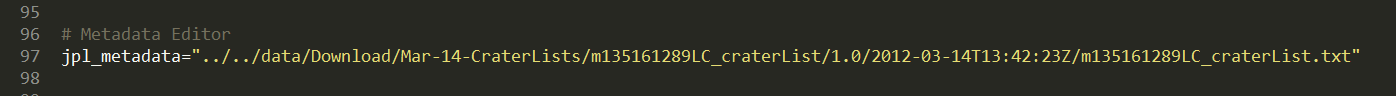
|  |  |  |
| --- | --- | --- |
| Parameter | Description | Type |
| upper\_hist\_threshold | Upper hysteresis threshold percentage ranging from 0.01 to 1.00 |  |

**Template Matching Parameters:**



|  |  |  |
| --- | --- | --- |
| Parameter | Description | Type |
| sun\_azimuth | Sun azimuth angle in radians | Double |
| margin\_ratio | Bounding boxes twice the size | Double |
| median\_kernel\_size | Median kernel filter | Int |
| area\_filter\_size | Remove highlight or shadow < 4 | Int |
| power\_filter\_threshold | A\*(h-hn)^2 > 15 | Int |
| shape\_filter\_max\_distance | Weighted euclidean distance of features from templates < 55 | Int |
| shape\_filter\_min\_matcher | Match at least 1 template | Int |
| highlight \_shadow\_area\_ratio | Area of highlights should be at most 4.2 times the area of the corresponding shadows | Double |
| highlight\_shadow\_distance\_coef | Distance of highlights from shadows is at most 1.95 | Double |
| mac\_combined\_elongation | Max elongation of highlights and shadows | Int |
| individual\_elongation\_coef | Max elongation of individual highlights or shadows | Double |
| sun\_azimuth\_tolerance\_coef | Difference between crater orientation and sun azimuth | Double |
| max\_roi\_eccentricity | Ratio of laterals of resulting rectangular. 2 at most | Int |
| highlights\_folder | Templates folder |  |
| shadows\_folder | Shadow templates folder |  |
| generate\_cropped\_candidates | Crop crater candidates for machine learning | String |
| generate \_cropped\_candidates\_  shadow\_images | Crop shadow images for depth | String |
| Use\_weighted\_euclidean\_  distance | distance | String |
| use\_hu\_moments | Hu moments instead of flusser | String |

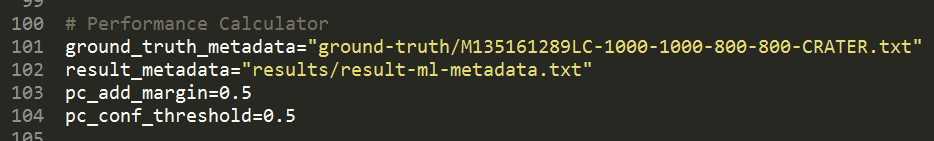
**Metadata Editor Parameters:**



|  |  |
| --- | --- |
| Parameter | Description |
| jpl\_metadata | Directory for JPL metadata. It also needs the image for the corresponding list |

**Performance Calculator Parameters:**

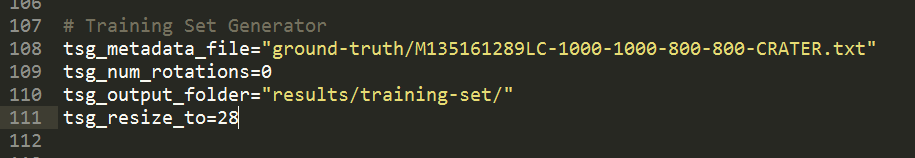
This program compares the output of crater detectors with ground truth files and reports their performance (f1 score, precision, recall etc.) and displays 3 images containing true positives, false positives and false negatives respectively.



|  |  |
| --- | --- |
| Parameter | Description |
| ground\_truth\_metadata | Directory for ground truth |
| result\_metadata | Directory for results file |
| pc\_add\_margin | Margin to add |
| pc\_conf\_threshold | Confidence threshold value |

**Training Set Generator Parameters:**

This program reads a metadata file, crops each roi in the metadata file, rotates it a specified number of times and then saves all those rotated images to a folder.



|  |  |
| --- | --- |
| Parameter | Description |
| tgs\_metadata\_file | Metadata input file |
| tgs\_num\_rotations | Number of rotations |
| tgs\_output\_folder | Output folder for results |
| tgs\_resize\_to | Pixel resize value |

**Create Database Parameters:**

|  |  |
| --- | --- |
| Parameter | Description |
| cd\_metadata\_file | Metadatafile to create database from |
| cd\_username | Username for database |
| cd\_password | Password for database |

**Execute Parameters:**

|  |  |
| --- | --- |
| Parameter | Description |
| delay | Sleep value |

**Hazard Map Parameters:**

|  |  |
| --- | --- |
| Parameter | Description |
| red\_thresh | Value for red pixel value threshold |
| yellow\_thresh | Value for yellow pixel value threshold |

**Two Phase:**

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Type |
| rt\_two\_phase | Two phase | String ‘Yes/No’ |

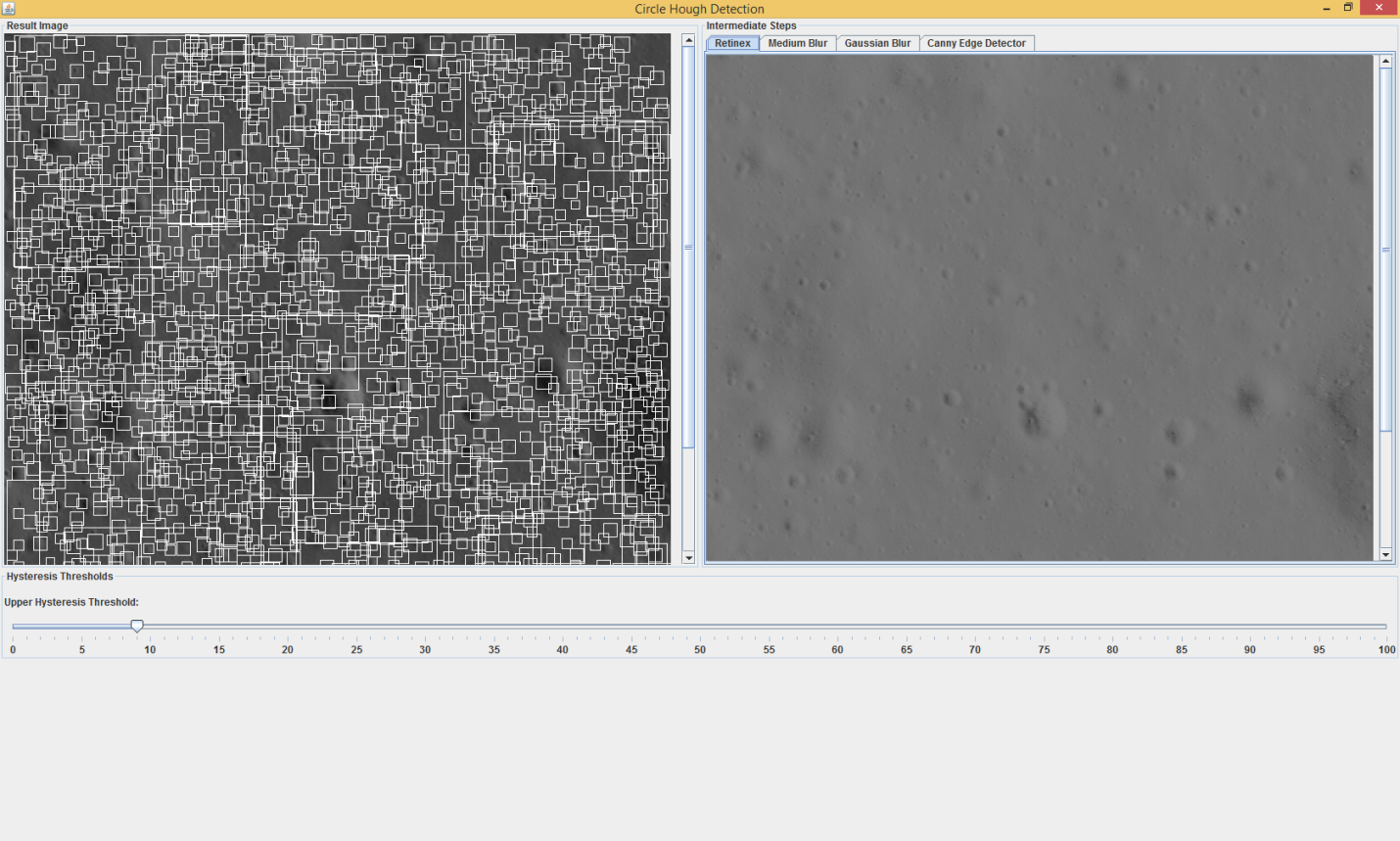
**Running Ringtoss GUI Programs**

If any of the following are set to “yes” in the bash file

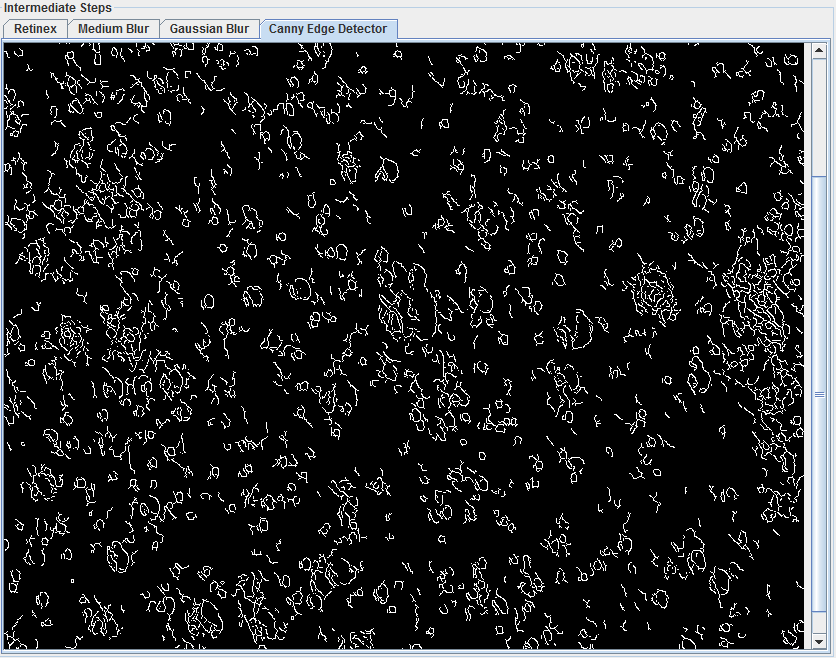
* “run\_circle\_hough”
* “run\_ellipse\_gui"
* “run\_data\_extract"
* “run\_metadata\_editor"
* “run\_performance\_calculator"

then their respective GUI version will be run.

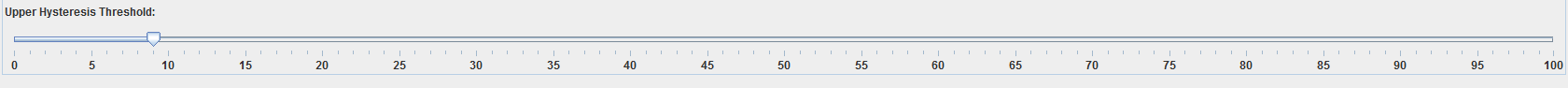
**Circle Hough GUI:**



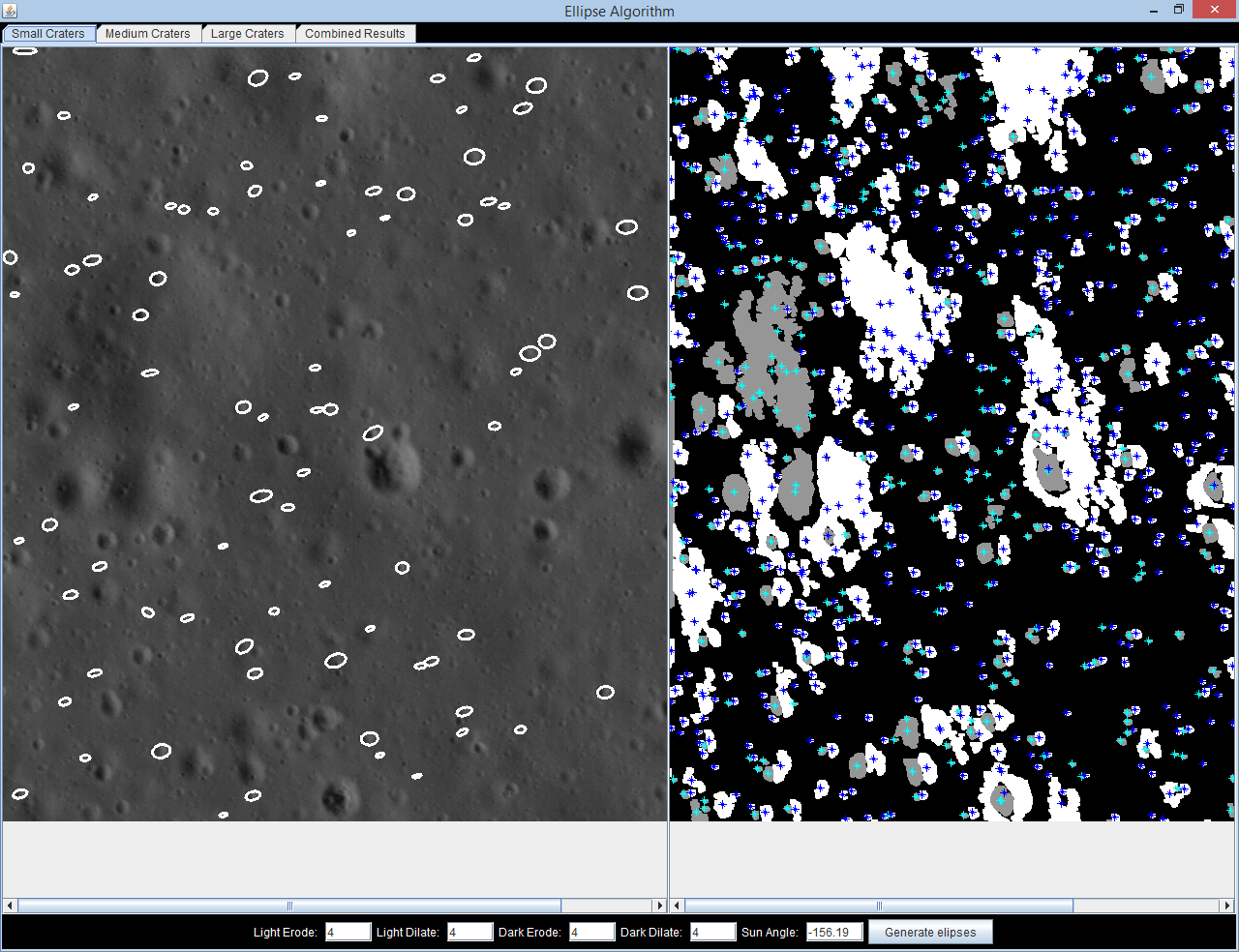
Circle Hough GUI provides two panes with image results and slider that allows the user to change the hysteresis threshold. The left pane is the final result after running the Circle Hough program. The right pane provides image for the intermediate steps of the program including a retinex image, filter images, and an edge image:



The user can change the upper hysteresis threshold using a slider between 1 and 100. Lower values produce more edges.



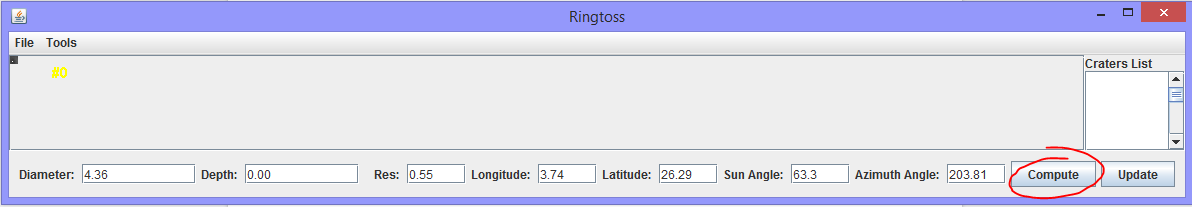
**Ellipse Fitting GUI:**



Ellipse Fitting GUI has 2 windows on the main tab. The left shows the small craters detected and the right shows the centers of the dark & light patches after erosion & dilation. The main tab also has options for entering different erosion and dilation levels to adjust the detection results. The GUI also shows medium and large craters in separate tabs as well as a combined tab shows all three sizes of craters.

**Data Extract GUI:**

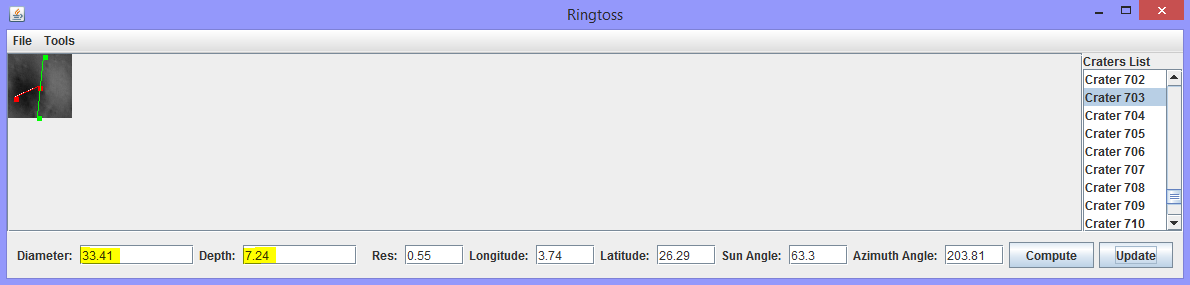
Data Extract GUI is allows you to edit the diameter and/or depth of a crater, after running ringtoss. To view the list of candidate craters, begin by selecting “compute.”



A list of craters will appear on the right hand side. Selecting a crater from the list will display the crater on the upper left corner of the GUI, and populate the fields, depth, diameter, and so on with respect to the selected crater. Resolution, longitude and latitude, and other values are taken from the metadata of the image, and used for computation.



The crater displayed has a white line running through the crater’s shadow region. The endpoints of this line were used to compute the depth by ringtoss. To change the depth select “draw shadows” under tools, and select two points (which will be red) on the crater. Similarly to change the diameter, select “draw diameter” under tools, and select two points on the crater. The points for diameter will be green.



Select update once you’re done with the drawing tools and new depth and diameter values will be displayed.

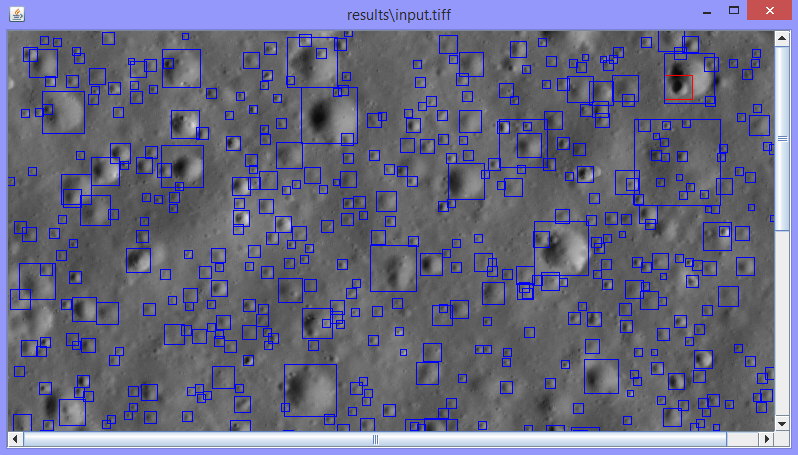
**Metadata Editor GUI:**

A GUI tool to generate ground truth metadata files. Here are some tips on how to use this tool:

|  |  |
| --- | --- |
| Command | Description |
| Ctrl + O | Open metadata file |
| Ctrl + S | Save to metadata file |
| Ctrl + C | Reset to initial state |
| Ctrl + R | Remove all bounding boxes |
| r | Remove the most recently added bounding box |
| d | Toggle display bounding boxes |
| e | Toggle histogram equalization |
| Mouse left | Add a new bounding box |
| Mouse right | Remove a bounding box |

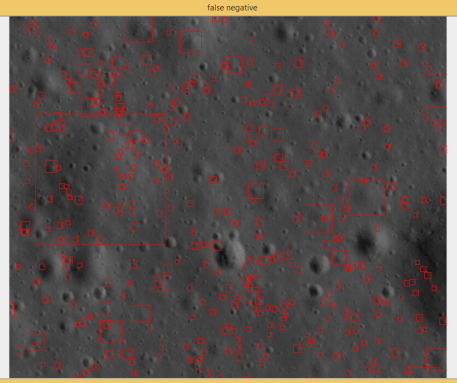
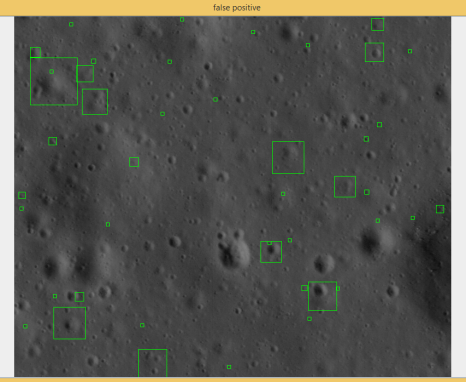
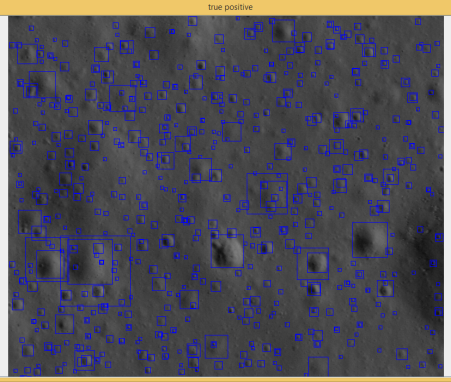
Input parameters in the bash file:

|  |  |
| --- | --- |
| Parameter | Description |
| cropppedimage | Under gdal section |
| jpl\_metadata | Path to JPL metadata file |
| cropx | Under gdal section |
| cropy | Under gdal section |



**Performance Calculator GUI:**

This program compares the output of crater detectors with ground truth files and reports their performance (f1 score, precision, recall etc.) and displays 3 images containing true positives, false positives and false negatives respectively.

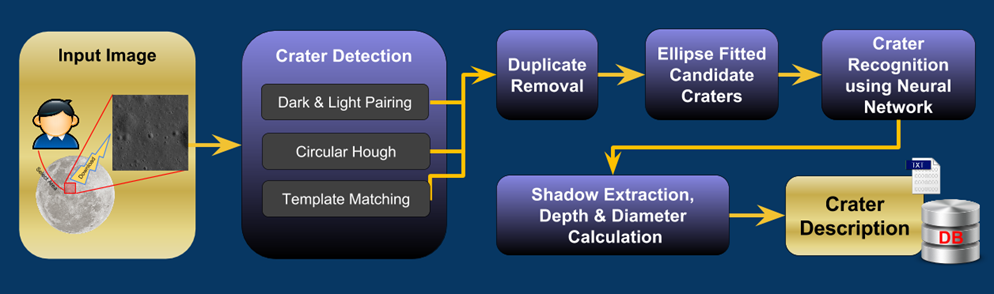


True Positives False Positives False Negatives

**4****.0 ARCHITECTURE AND DESIGN**

**4.1 Ringtoss Pipeline**

Original Ringtoss Pipeline



**Crater Detection**

Ringtoss utilizes three different algorithms for detecting craters within an image:

**Dark and Light Pairing**

The light and dark pairing is based on the approach taken in the research paper "Detection of Craters and Its Orientation on Lunar" by Nur Diyana Kamarudin, Kamaruddin Abd. Ghani, Siti Noormiza Makhtar, Baizura Bohari and Noorlina Zainuddin. This crater detection algorithm focuses on finding the light and dark patches and then matching them up based on sun angle and size restraints. The light and dark patches are removed from the background using Otsu’s method, a thresholding technique. Any background noise around the light and dark sections are morphed using erosion and dilation.

Different size craters are detected using varying levels of erosion and dilation. Thus, crater detection is split up into three levels: small, medium, and large craters. Duplicate craters that are detected between the three levels are removed and saved to a list.

**Circular Hough**

This Algorithm is based on a proposed methodology found in the research paper “LINEAR AND NON-LINEAR FEATURE EXTRACTION ALGORITHMS FOR LUNAR IMAGES” by Tamililakkiya V., Vani K., Lavanya A., and Anto Micheal.

A single scale retinex algorithm is used to remove shadows in the image then filters, canny edge detector and circular hough transform are applied to the image and circular sections are extracted from the image and saved to a list as a crater.

**Template Matching**

Matches light and dark patches with predefined criteria including template images.

**Duplicate Removal**

After the crater detection phase is complete the list of potential craters must be filtered to remove duplicate craters by merging them. For duplicate removal, unique craters are first placed into a quadtree for efficiency for crater searches. Unique craters are determined by bounding box overlap. If there is over 50% overlap in the bounding boxes of two craters, then they are considered to be the same crater, and one is a duplicate of the other.

Once the quadtree is populated with all unique crater candidates, the other craters belonging to the same region as the unique craters are compared with those unique craters. If they are duplicates by our standards (50% bounding box overlap), then they are merged with the unique crater by creating a new averaged bounding box. The quadtree then contains all merged and unique craters which is later used for crater recognition and data extraction.

**Ellipse Fitting**

The Ellipse fitting is used to refine some of the bounding boxes from the detected crater list. This method crops the bounding box of each detected Taking into account the size of the light and dark patches in cropped area, it looks for the largest patches then matches them together. Then the method goes on to draw an ellipse around the matched pairs. Drawing an ellipse will create a new bounding box that may be more refined than it was before. It adds the new bounding box to the list of the detected craters.

**Crater Recognition**

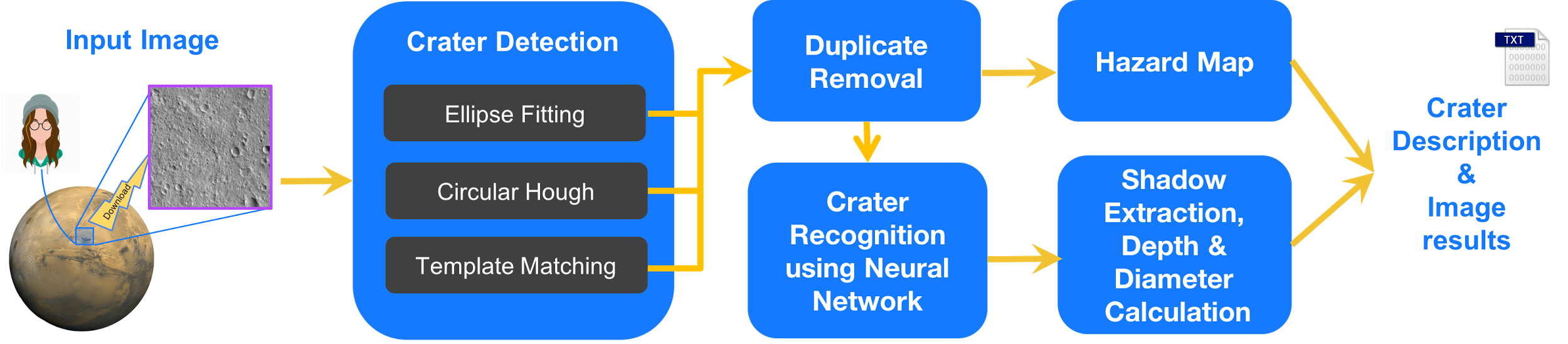
Takes the detected craters for duplicate removal and tries to classify them as a craters or non-craters using an artificial neural network.

**Feature Extraction - Depth & Diameter**

To compute for both diameter and depth, the methods take in a list of candidate craters, along with the corresponding image and metadata of the image. The length and width of crater bounding boxes are used for computing diameter. The average of the width and height of the bounding box are calculated and considered to be the diameter of the crater. The pixel scale determined by the metadata.

**4.2 Modifications to Ringtoss**

Updated Ringtoss Pipeline



The pipeline has been upgraded to include a Hazard Map feature as well as updates to the Duplicate Removal and Crater Recognition Modules

**Hazard Map**

The hazard map takes input from Ringtoss after duplicate removal but isn’t executed until the neural network is done classifying the detected craters. This module splits the input image into a hectare grid and for each hectare calculates the density of the features detected in that region. The features detected are filled with red pixels. Feature density for the hazard map is calculated by scanning through the whole image row by row, pixel by pixel and counting the number of red pixels within each hectare.

Then the percentage used for feature density will be calculated by dividing each hectare’s red pixel count by its respective area. A color will be assigned to that hectare using that percentage. 80% and above is red, 79%-50% is yellow, and anything below 50% is green. These are only the default values and can be changed by changing the values of **red\_thresh** and **yellow\_thresh** in the mipfd.sh file.

**Duplicate Removal**

With the previous implementation of the duplicate removal module, there were a number of actual craters that were being removed from the crater candidate list before the list was sent to the recognition phase. To avoid this, a new system for duplicate removal was proposed and implemented. The updated duplicate removal module now takes into account the following criteria to determine whether two craters are duplicates:

* The distance between the center of the two craters
* The size of the craters
* The area that the craters occupy

To be more specific, a crater is considered a duplicate if:

* The bounding box of the two craters contain each other’s center
* The ratio of the craters area is greater than 80% (similar in size)
* The craters overlap by at least 50%

**Crater Recognition**

The neural network was retrained with mars images in order to get better recognition results for the mars images. However, the neural network that was trained with lunar images still produced good recognition results. To leverage both, a two phase recognition system was implemented in which crater candidates are sent through the lunar trained neural network and then through the mars trained neural network for final classification.

**5.0** **CONCLUSION**

The Salient Feature Detection team succeeded in modifying the Ringtoss system to work with Martian images. Martian craters are very different from lunar craters, and they are generally less well-defined because weather blows sand over them, so it is perhaps not surprising that the performance of Ringtoss on Martian images is still inferior to its performance on lunar images. Still, our work has not been for naught. Ringtoss as it stands is a more versatile product than what we started with, which is all one can really hope for in a project such as this. The hazard mapping provides a new tool both for analyzing the results of the program and for analyzing the surface of Mars itself.

Further improvements can still be made. There is a lot of architectural debt in the current system, so any future team should probably being by looking to clean up the existing systems and migrate the neural network to a more maintainable environment. The detection algorithms were judged “good enough” this year but could still use some tweaking. Template matching, in addition to tweaking, could always use more templates. The entire project is in need of a GUI; it’s not professional to have the official interface of a program be shell script. In terms of larger tasks, we did not implement any process by which multiple images of the same terrain could be stitched together, allowing the system to process a larger area. This process is fairly automated with the Image processing software used, so it would likely be an efficient feature to add.