**CS496 Senior Design**

**Lunar Crater Detection and Recognition**

**Project Documentation**

Document Prepared by:

Natalie Gallegos

Tony Liang

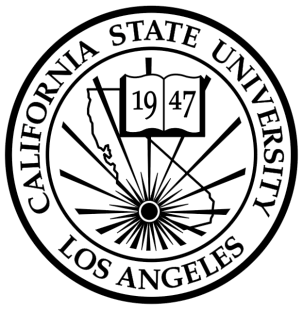
Marvin Mendez

Raul Ornelas

Alberto Serrano

28 May 2015

CALIFORNIA STATE UNIVERSITY LOS ANGELES



Los Angeles, California

**Table of Contents**

1.0 Executive Summary

2.0 Introduction

3.0 Installation/User Guide

1. Installation
2. Bash file
3. GUI

4.0 Lifelong Learning

1. Raul Ornelas
2. Albert Serrano
3. Marvin Mendez
4. Natalie Gallegos
5. Tony Liang

5.0 Architecture and Design

1. Image retrieval
2. Ringtoss
   1. Crater Detection
      1. Light and Dark pairing
      2. Circle Hough
      3. Template Matching
   2. Duplicate Removal
   3. Ellipse Fitting
   4. Machine Learning - Convolutional Neural Network
   5. Feature Extraction - Depth and Diameter
   6. Output

6.0 Conclusions

# 

# 

# 

# EXECUTIVE SUMMARY

The Lunar Crater Detection and Recognition (LCDR) project’s goal was to create an algorithm or system for crater detection. The LCDR team accomplished this goal through a pipeline called Ringtoss. Ringtoss is the culmination of nine months of work between five students and advisors.

The first three months were spent researching and prototyping potential algorithms for crater detection. Each student was tasked with reading and presenting findings on a research paper. Each week the paper currently being read was discussed and voted on to be pursued further into the prototyping stage. We chose a few methodologies to further pursue and move on to the prototyping. Before the prototyping began we were introduced to the OpenCV image processing library and GDAL tool. These were the main tools we worked with for the following six months.

After creating the prototypes in the fall quarter of 2015 the next three months of the winter 2015 quarter were spent refining our prototype algorithms. The team enrolled in a computer vision course which equipped us with the knowledge to refine our algorithms and continue on the task to create crater detection and feature extraction algorithms. The team was split in order to work on these algorithms. Albert, Marvin, and Raul worked on crater detection. Natalie and Tony worked on feature extraction. Albert and Raul worked on shadows and highlights pairings with ellipse fitting as an approach for crater detection. Marvin worked on crater detection by means of a circular Hough transform. One of our graduate advisors, Saman Saeedi, worked on a template matching approach. Natalie and Tony worked on calculating depth and diameter of a crater.

The final three months of the project were spent on integrating our algorithms into one cohesive system we named Ringtoss. This was no easy task and required just about the hours equivalently spent on prototyping and refinement. We used the three aforementioned crater detection algorithms as a hybrid approach to crater detection. The results of the crater detectors were merged and filtered for unique craters in our duplicate removal section of Ringtoss. We refitted ellipse on the unique craters and passed them into a machine learning algorithm. Those results were used in calculating depth and diameter. Relevant information for each detected crater are stored in a MySQL database for future use.

This Ringtoss pipeline is the culmination of our nine months work. We were able to attain results within 5% of the F1 score of JPL results. We obtained a larger number of true positives at the cost of more false positives. There were a few issues when incorporating longitude and latitude calculations that hindered us from implementing multiple image analysis but this is left for future work.

# INTRODUCTION

Crater detection and avoidance is critical for spacecraft landings and mapping of lunar terrain. Resources for crater detection are limited and new approaches are always welcome. The Lunar Mapping and Modeling Portal (LMMP) team at JPL tasked our LCDR team to develop an algorithm that will automatically detect lunar craters using images from the Lunar Reconnaissance Orbiter Camera (LROC). Our approach to crater detection was to combine several algorithms and achieve a high accuracy percentage. Approaches for detecting candidate craters include template matching, circle hough transform, and extraction of highlights and shadows. Accuracy is prioritized over performance, per request by JPL.

The Ringtoss pipeline is our main approach and the culmination of our work. It begins by having a user select a region on the moon. Ringtoss then downloads an image from the LROC repository for the selected region of interest. 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. Finally, depth and diameter are calculated for each detected crater. Results are provided in text output and stored in a DB as well.

# 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)

Please download and setup the requirements listed above. Some helpful links to install these requirements on Ubuntu or a Linux machine are provided below.

* <https://www.java.com/en/download/> - java
* <https://help.ubuntu.com/community/OpenCV> - OpenCV
* <http://www.sarasafavi.com/installing-gdalogr-on-ubuntu.html> - GDAL
* <http://heliumhq.com/docs/installing_python_2.7.5_on_ubuntu> - Python
* <https://help.ubuntu.com/12.04/serverguide/mysql.html> - MySQL

For a windows machine please use the following links to setup the environment for the program:

* <https://www.java.com/en/download/help/windows_manual_download.xml> - java
* <http://docs.opencv.org/doc/tutorials/introduction/windows_install/windows_install.html> - OpenCV
* <http://cartometric.com/blog/2011/10/17/install-gdal-on-windows/> - GDAL
* <https://www.python.org/downloads/windows/> - Python
* <http://dev.mysql.com/downloads/windows/installer/5.6.html> - MySQL

**Additional files to download:**

* [2012-03-14-CraterLists.tgz](https://web.lmmp.nasa.gov/webdav/public/2012-03-14-CraterLists.tgz) - JPL txt files for detected craters on LROC images.
* [CUMINDEX.TAB](http://lroc.sese.asu.edu/data/LRO-L-LROC-2-EDR-V1.0/LROLRC_0021/INDEX/) - A file containing metadata of LROC images.

Links to download additional files:

* 2012-03-14-CraterLists.tgz - <https://web.lmmp.nasa.gov/webdav/public/2012-03-14-CraterLists.tgz>
* CUMINDEX.TAB - Right click on CUMINDEX.TAB and save the file - <http://lroc.sese.asu.edu/data/LRO-L-LROC-2-EDR-V1.0/LROLRC_0021/INDEX/CUMINDEX.TAB>

**Note:** Paths to libraries and jar files will be needed. You will also need to provide the path to the downloaded files, CUMINDEX.TAB and to the text files extracted from 2014-03-14-CraterLists.tgz.

**MySQL setup:**

A database will need to be created in MySQL in order to run the programs. Please create a database named lrocmetadata with username and password “lipfd”. You can use the commands below to set up the new database and user.



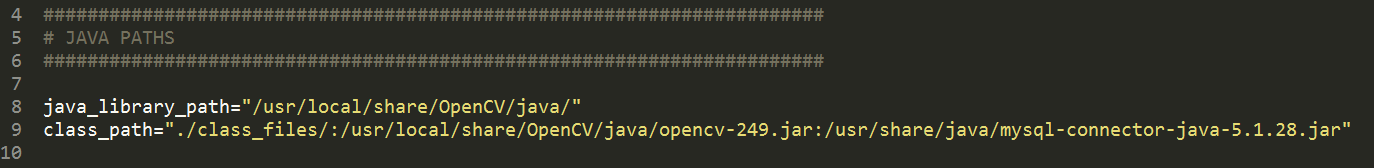
This database will be populated with entries of metadata files for each lroc image from Arizona State University website: <http://lroc.sese.asu.edu>. Further instructions on how to run the program that will insert these entries will be provided in the User Guide, under the “Create Database Parameters” section.

**3.II Bash File Parameters**

**Path Setups:**

In order to begin using Ringtoss you must first navigate to the “lipfd.sh” bash file in the “LIPFD” folder.

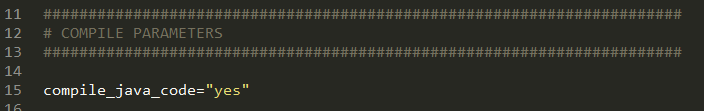
Ensure you provide the correct paths in the following lines:



Jar files for opencv and mysql should be included in class\_path, along with path to all class files. Java\_library\_path should include path to opencv’s java.dll file.

**Compile Parameters:**

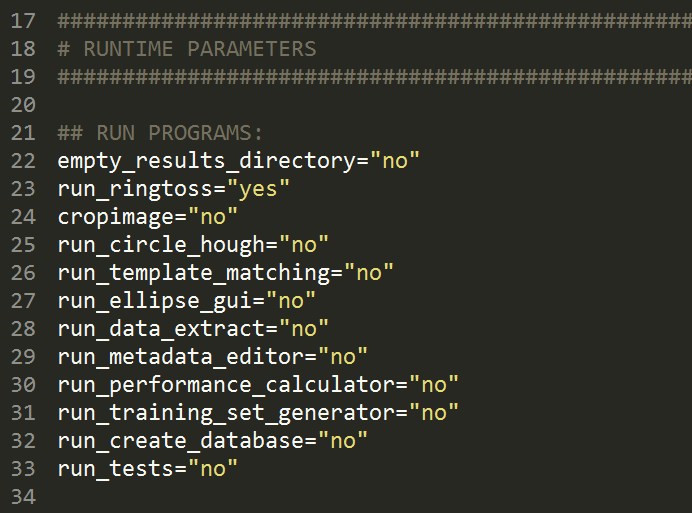
The “lipfd.sh” bash file provides a parameter to compile java code:



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.



empty\_results\_directory:

1. “yes”: will empty results directory before running Ringtoss
2. “no”: will maintain contents in results directory

run\_ringtoss:

1. “yes”: runs main program Ringtoss
2. “no”: does not run main program Ringtoss

cropimage:

1. “yes”: will crop an image using the gdal command below:

gdalcomm.PNG

1. “no”: will not crop an image

run\_circle\_hough:

1. “yes”: runs a GUI version of the Circle Hough program independent of Ringtoss
2. “no”: will not run the GUI version of the Circle Hough

run\_template\_matching:

1. “yes”: runs Template Matching independently of Ringtoss
2. “no”: does not run Template Matching

run\_ellipse\_gui:

1. “yes”: runs a GUI version of the Ellipse Fitting program independent of Ringtoss
2. “no”:does not run Ellipse Fitting program

run\_data\_extract:

1. “yes”: runs a GUI of the crater depth and diameter calculation program
2. “no”: does not run the Data Extract program

**Note:** Crater detection algorithms must run first and store results in a text file. DataExtract 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.

run\_metadata\_editor:

1. “yes”: runs metadata editor to create/update ground truth
2. “no”: does not run metadata editor

run\_performance\_calculator:

1. “yes”: runs performance calculator to compute f1 score
2. “no”: does not run performance calculator

run\_training\_set\_generator:

1. “yes”: generates a training set for the machine learning portion of Ringtoss
2. “no”: does not generate a training set

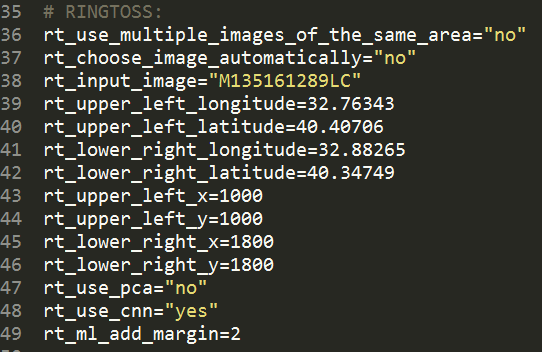
run\_create\_database:

1. “yes”: creates a database with metadata
2. “no”: does not create a database

run\_tests:

1. “yes”: runs tests
2. “no”: does not run tests

**Ringtoss Parameters:**



rt\_use\_multiple\_images\_of\_the\_same\_area:

1. “yes”: non-functional; left for future work
2. “no”: use a single image to run Ringtoss

rt\_choose\_image\_automatically:

1. “yes”: will choose an image to run Ringtoss on automatically
2. “no”: image must be provided as input to Ringtoss

rt\_input\_image: input image id for Ringtoss

rt\_upper\_left\_longitude: upper left longitude of the input image

rt\_upper\_left\_latitude: upper left latitude of the input image

rt\_lower\_right\_longitude: lower right longitude of the input image

rt\_lower\_right\_latitude: lower right latitude of the input image

rt\_upper\_left\_x: upper left x pixel value of the input image

rt\_upper\_left\_y: upper left y pixel value of the input image

rt\_lower\_right\_x: lower right x pixel value of the input image

rt\_lower\_right\_y: lower right y pixel value of the input image

rt\_use\_pca:

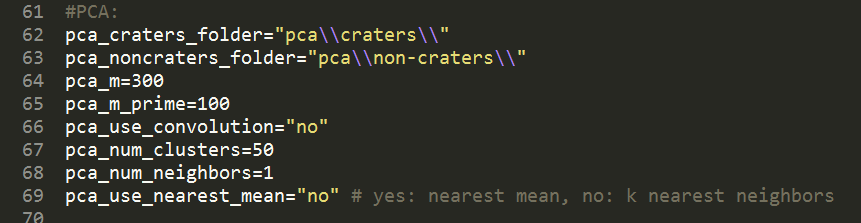
1. “yes”: runs Principal Component Analysis (PCA) with Ringtoss
2. “no”: does not run PCA

rt\_use\_cnn:

1. “yes”: runs Convolutional Neural Network (CNN) with Ringtoss
2. “no”: does not run CNN

rt\_ml\_add\_margin: Increase the bounding boxes to (by default) twice the original size for machine learning

**PCA Parameters:**



pca\_craters\_folder: directory for crater training images folder

pca\_noncraters\_folder: directory for non-crater training images folder

pca\_m\_prime: m prime value

pca\_use\_convolution:

1. “yes”: uses convolution
2. “no”: does not use convolution

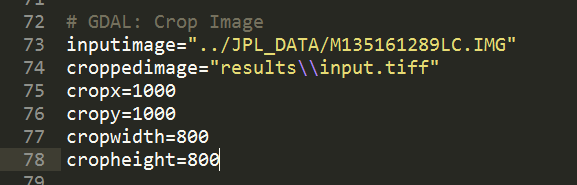
pca\_num\_clusters: number of clusters

pca\_num\_neighbors: number of neighbors for nearest neighbors

pca\_use\_nearest\_mean:

1. “yes”: uses nearest mean
2. “no”: uses k nearest neighbors

**GDAL Parameters:**



inputimage: directory of image to crop

croppedimage: directory and name of resulting crop image

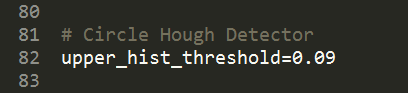
cropx: x pixel offset

cropy: y pixel offset

cropwidth: width of cropped image

cropheight: height of cropped image

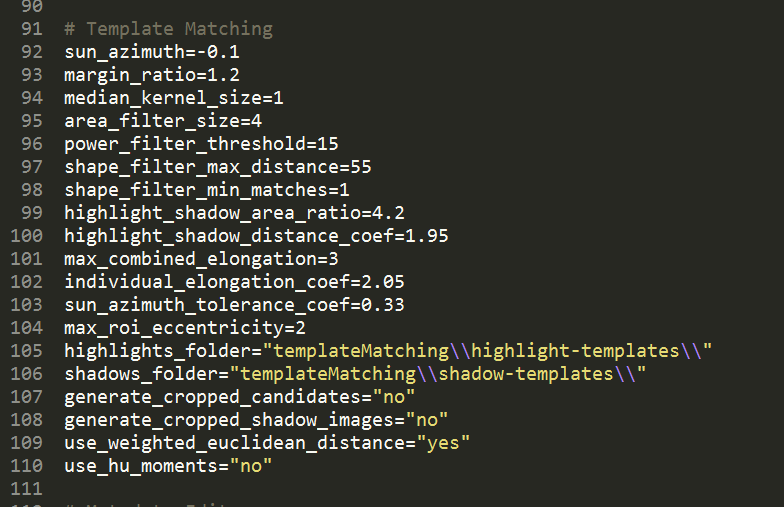
**Circle Hough Parameters:**



upper\_hist\_threshold: upper hysteresis threshold percentage

Ranges from 0.01 to 1.00. Lower values produce more edges in the canny edge detector.

**Template Matching Parameters:**



**1.** sun\_azimuth: sun azimuth angle in radians compared to the east of

the image

**2.** margin\_ratio: the bounding boxes are twice the original size

**3.** median\_kernel\_size: median filter kernel (1=off)

**4.** area\_filter\_size: remove any highlight or shadow smaller than 4 pixels

**5.** power\_filter\_threshold: A\*(h-hn)^2 for any feature should be bigger than 15

**6.** shape\_filter\_max\_distance: weighted euclidean distance of features from

templates should be smaller than 55

**7.** shape\_filter\_min\_matches: match at least one template

**8.** highlight\_shadow\_area\_ratio: area of highlights should be at most 4.2 times the area of the corresponding shadows and vice versa

**9.** highlight\_shadow\_distance\_coef: distance of highlights from shadows over

max(sqrt(their area)) is at most 1.95

**10.** max\_combined\_elongation: max combined elongation of highlights and shadows

**11.** individual\_elongation\_coef: max elongation of individual highlights or shadows

**12.** sun\_azimuth\_tolerance\_coef: difference between crater orientation and sun azimuth

angle can be at most 0.33 \* PI

**13.** max\_roi\_eccentricity: the ratio of the laterals of the resulting rectangular

bounding box can be at most 2

**14.** highlights\_folder: highlight templates folder

**15.** shadows\_folder: shadow templates folder

**16.** generate\_cropped\_candidates: “yes” to crop crater candidates for machine learning

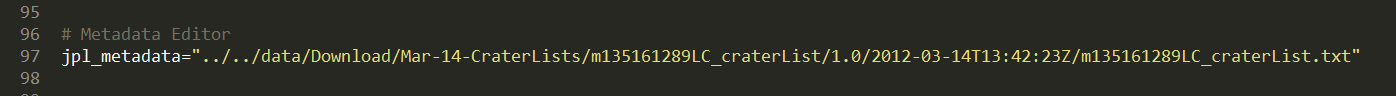
**17.** generate\_cropped\_shadow\_images: “yes” to crop shadow images for depth

detection

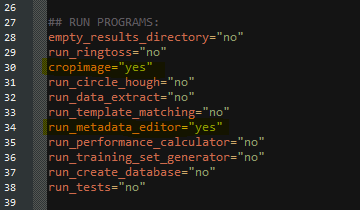
**18.** use\_weighted\_euclidean\_distance: “yes” to use; “no” otherwise

**19.** use\_hu\_moments: “yes” to use hu moment invariants instead of Flusser

**Metadata Editor Parameters:**

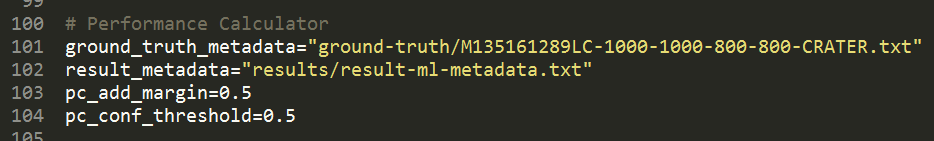


jpl\_metadata: directory for JPL metadata. In addition to supplying path to jpl’s craterLists, metadata also needs the image for the corresponding list. Run GDAL along with Metadata Editor to create ground truth files.



**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.



ground\_truth\_metadata: directory for ground truth

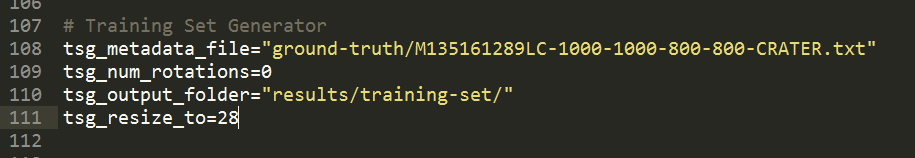
result\_metadata: directory for results file

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.



tsg\_metadata\_file: metadata input file

tsg\_num\_rotations: number of rotations

tsg\_output\_folder: output folder for results

tsg\_resize\_to: pixel resize value

**Create Database Parameters:**

cd\_metadata\_file: metadata file to create database from

cd\_username: username for database

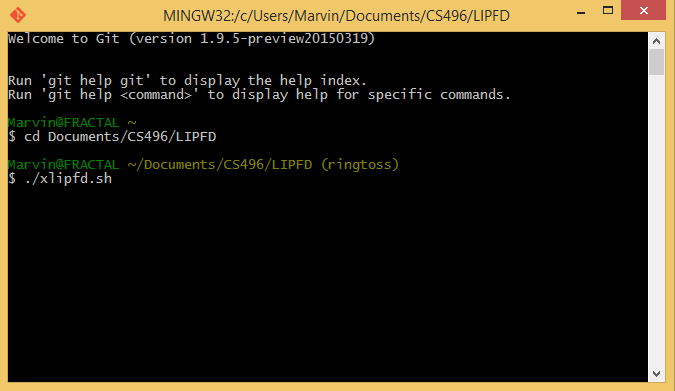
cd\_password: password for database

**Execute Parameters:**

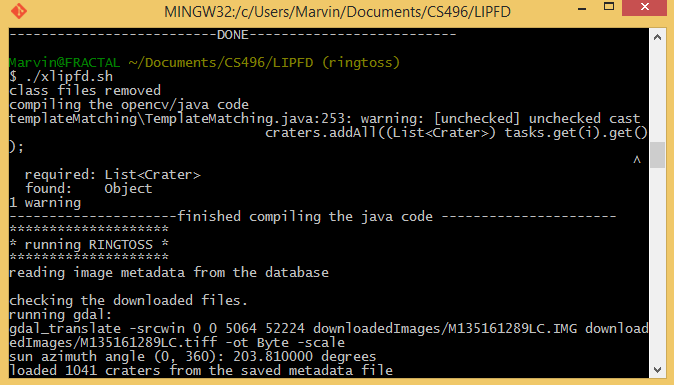
delay: sleep value

**Running Ringtoss**

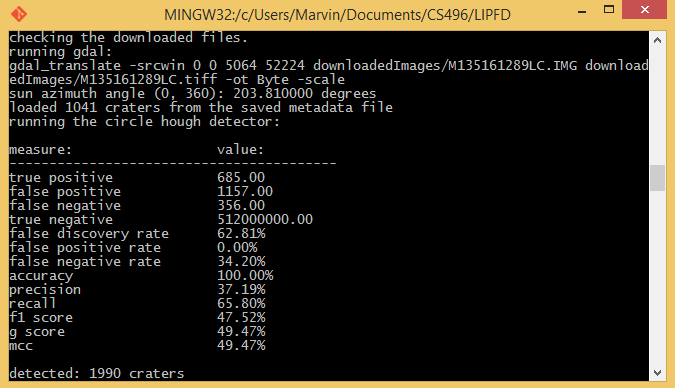
To run the main Ringtoss program navigate to the folder where the “lipfd.sh” bash file is located and run the bash file:



The java code will compile if the parameter “compile\_java\_code” is equal to “yes” and will begin running Ringtoss. Ringtoss will read the input image’s metadata from the database to download and crop the image to a pre-specified area. If the image already exists it will just crop the image. It will also retrieve existing JPL metadata to compare to.



If only “run\_ringtoss” is set to “yes” the non-gui versions of Circle Hough, Ellipse Fitting, and Template Matching will run. Each will be output performance calculations and the total number of craters detected and pass a list of craters to be merged:



After Template Matching the three separate lists are merged into one and the machine learning portion of Ringtoss will begin to run. Depth and diameter will also be calculated on craters on the merged list that pass a confidence threshold value. This will produce a performance report as well:

measure: value:

-----------------------------------------

true positive 599.00

false positive 54.00

false negative 442.00

true negative 512000000.00

false discovery rate 8.27%

false positive rate 0.00%

false negative rate 42.46%

accuracy 100.00%

precision 91.73%

recall 57.54%

f1 score 70.72%

g score 72.65%

mcc 72.65%

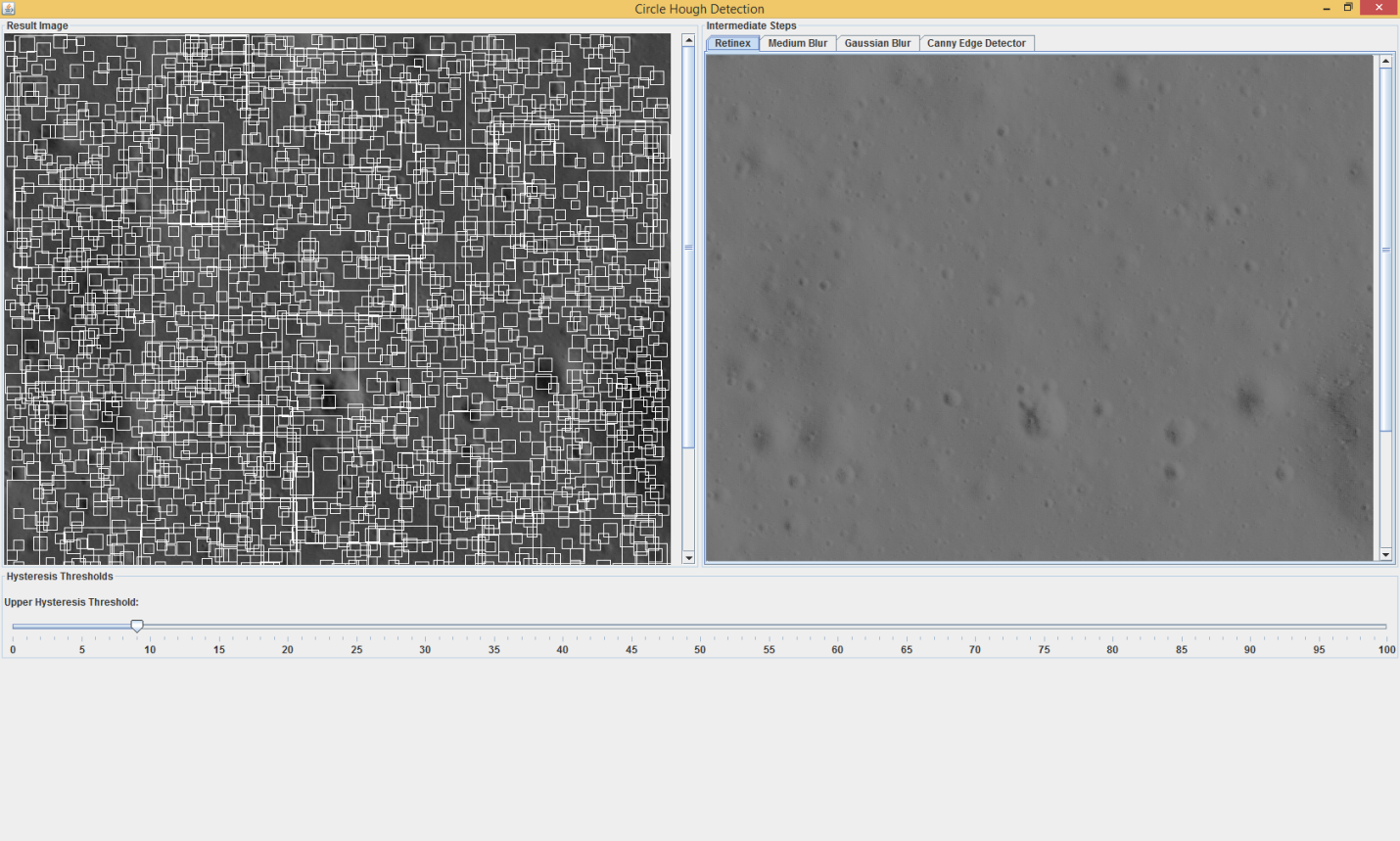
**3. III Individual GUI Programs:**

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

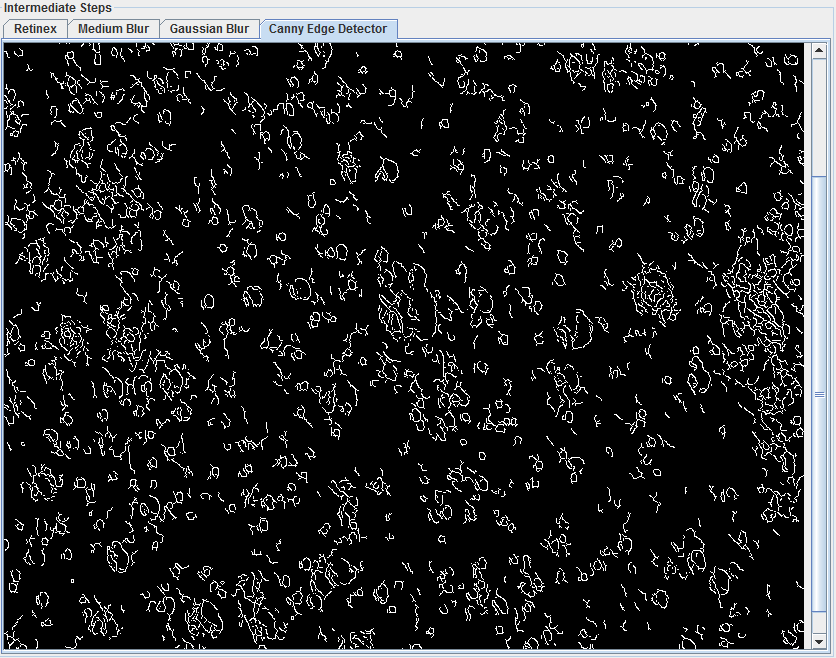
* “run\_circle\_hough”
* “run\_ellipse\_gui"
* “run\_data\_extract"
* “run\_metadata\_editor"
* “run\_performance\_calculator"

then their respective GUI version will be run.

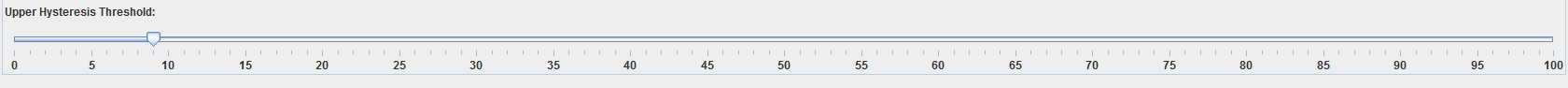
**Circle Hough GUI:**



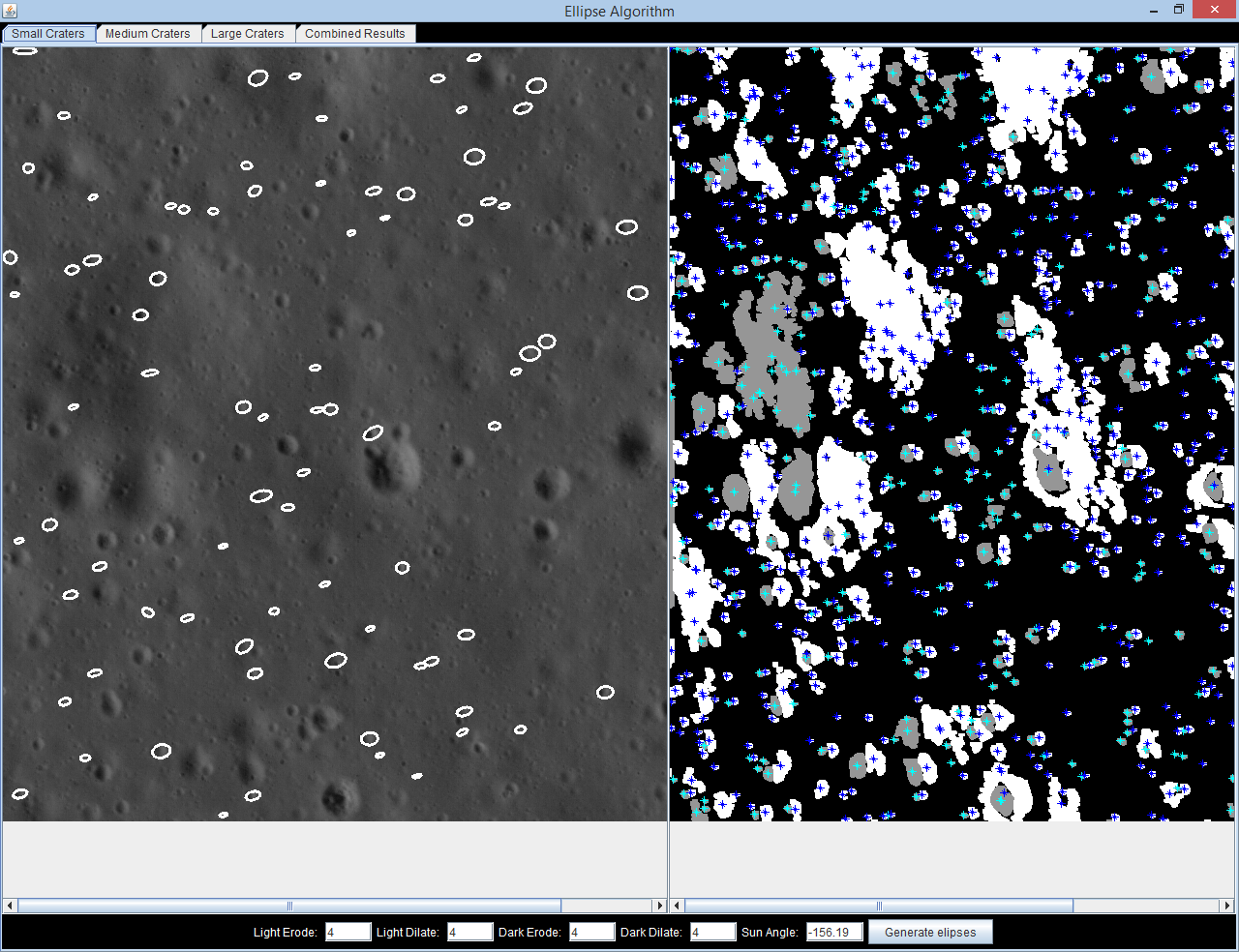
Circle Hough GUI provides two panes with image results and slider that allows the user to change the hysteresis threshold. The left pane is the final result after running the Circle Hough program. The right pane provides image for the intermediate steps of the program including a retinex image, filter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



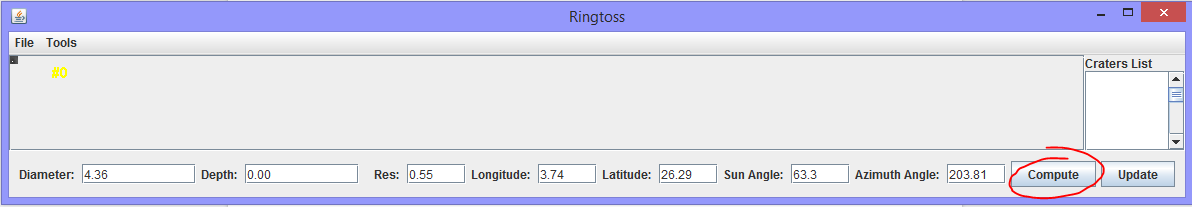
**Ellipse Fitting GUI:**



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:

Ctrl + O - Open metadata file

Ctrl + S - Save to metadata file

Ctrl + C - Reset to initial state (careful!)

Ctrl + R - Remove all bounding boxes (careful!)

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

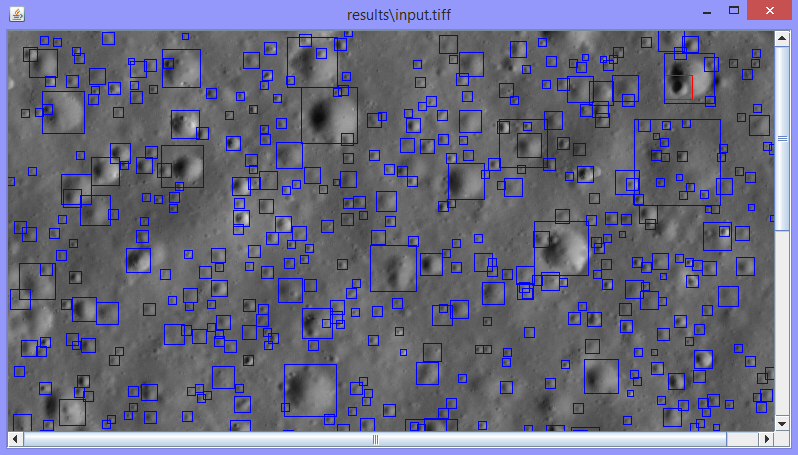
Input parameters in the bash file:

croppedimage (under gdal section)

jpl\_metadata Path to JPL metadata file

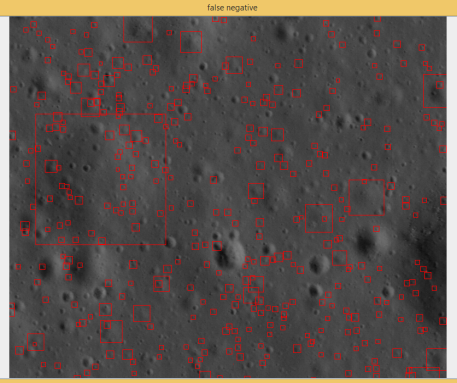
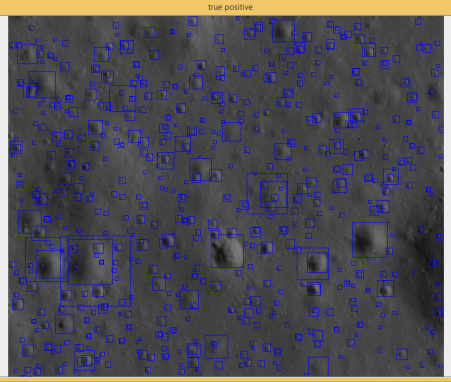
cropx (under gdal section)

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

# LIFELONG LEARNING

**4.I Raul Ornelas:**

This project was challenging in every aspect. From time management, to collaboration and coding; it required every skillset learned as a Computer Science major.

The project began with our team assignments in fall quarter 2014. After learning that I would be part of a project working directly with the talented minds of JPL, I was both honored and daunted. We began by deciding who would be the team leader and I quickly volunteered, not knowing exactly what this would entail. Shortly thereafter we were given a synopsis of the motivation for the project and my team members and I were intrigued to determine how to accomplish this task. Our advisor quickly assigned us research papers to read and get a introductory understanding of crater, rock and feature detection. After reading several papers, we focused our efforts on emulating the approach taken by two papers on crater detection and became the basis for our ellipse bounding algorithm and circular hough detection. Later in the fall quarter we added the depth and diameter as well as the machine learning portion of the pipeline to our development of our final product.

After some extensive discussion of several research papers we were given a crash course in the heart of our project; image processing. Most of our team had little to no experience in the topic and had to resort to tutorials and examples from the internet for a better understanding of our task. Our team was enrolled in a multimedia course during the fall quarter and we gained some valuable insight into the basics of image processing. We then began the process of putting together the framework for our product. We decided, after consulting with our faculty and JPL advisors that we would code our product using Java and employ the tools of the image processing library, OpenCV. After the first quarter, despite our basic skills in image processing, we had some preliminary results.

During winter quarter 2015, our focus was on the refinement of our sub prototypes to improve our results. We ran test on several images and continually looked for ways to increase the accuracy our our results. At the advice of our faculty advisor, our team enrolled in an advanced course on computer vision, which proved to be valuable in our comprehension of our project. The lectures and assignments of this course were based on OpenCv and were closely related to our project. From this course, we were able to further refine our methodology, adding techniques such as Otsu’s method for determining optimal threshold values and a better understanding of image erosion and dilation for removal of background image noise. By the end of winter quarter, we had refined our sub prototypes extensively and determined that each prototype had its strengths in crater detection that could be combined to achieve optimal results. After consulting with our advisors, we determined that a hybrid approach would provide the best results from our final product.

Spring quarter 2015 was the culmination of our project and was the most challenging. We began by planning how to integrate our separate prototypes into one cohesive product. This required input and participation of all our team members to merge our code. After a brief introduction to the Git repository in the fall quarter, we quickly had to get up to speed on the features and benefits of this collaborative tool. After many hours of coding and merging we produced our final product for our project.

**4.II Albert Serrano:**

This project has been a huge learning experience for me. Besides using Java, this project required me to pick up new skills that had not been taught to me in the classes i’ve took at CSULA. Skills in image processing, computer vision, code and time management. A lot of the project required me to teach myself through various sources on the web.

At the beginning of the cs496a we were assigned articles to read to find a way to solve the problem that was presented to us. After we read each article, we pick out the the approaches that looked doable and assigned them to different team members. Raul and I were assigned the approach in the article by Kamarudin. I spent several days reading the article to get a good image on how to implement his approach, then later in the quarter, spent time on figuring out what to change about it to improve the results.

Our code was going to go through a lot of revising. It was going to get big and we needed to make sure we had a good way of saving the different version of our program. Our team decided to use Git to save and store our code. Made it easier for integration and keeping track of revisions. I had not used It before, so I had to look and tutorials online and watch videos on how to perform some Git commands.

Our project involved a lot of image processing. So to make things easier for us, we chose OpenCV. It was a powerful, open source library and it supported java, a language we were all used too. However, the documentation for the java part of OpenCV was very scarce and vague. I’ve had to go online to find many different examples. I’ve watch videos tutorials and looks at a lot of different examples of code to try to stitch together an understanding of OpenCV. I bought a book showing how to use OpenCV with Android. Although we were not using Android for our project. A lot of the examples were in java code, which help me out.In classes a lot of the material that needs to be learned was presented formally. In other to learn about this image processing library. I had to go to many different sources and devote extra time to learning. After the first quarter, our team took a class on image processing using OpenCV, which went on to further expand my knowledge of OpenCV.

In cs496c, most of the work went to integrating our code that we’ve been working on. We ran into a lot of problems where our programs clashed, so time was spent on debugging. I’ve had to go through a lot of revisions in my code to make it easier for other parts of the program to use. This stage help me to communicate with my peers to and work together to combine our code.

I strongly feel that through this project I’ve skills that will help with future projects. I’ve learn how to manage my time, conduct research, how to keep record of updates to my code, work with team members and communication efficiently, topics in computer vision, and using an image processing library. This experience will remind me on how to approach future projects i’ll be involved in.

**4.III Marvin Mendez:**

The entire LCDR project was filled with learning experiences. Most of us entered the project with minimal knowledge on image processing, including myself. We all left with extensive knowledge with the OpenCV image processing library and computer vision in general. My first learning experience, however, began in the 2014 Fall quarter in CS496A.

In CS496A we met our group and immediately began researching solutions to the task provided to us by our client: JPL. I have collaborated with peers before, but this was my first time working first hand with a client. I had to learn to properly manage my time to meet deadlines and work with my fellow peers to tackle the problem of crater detection. I dedicated a set schedule to work on the project and work on my last few courses before graduation. I had to cut back on personal projects and distractions. I learned to balance work and play (to refresh the thinking muscles), so that I would be prepared for the next day on the project.

In the CS496B I learned the technical skills needed to properly handle and solve the problem of crater detection. I took a course (along with most of the team) on computer vision and a course on machine learning. These two courses allowed me to understand the entire pipeline of our project. The knowledge I gained from the computer vision course was easily transferable to my portion of the project. I was able to refine the methodology I was emulating from a proposed research paper thanks to learning the OpenCV library. I learned to use other tools as well, such as GDAL.

CS496C is where everything culminated. I gained more practice with integrating a large system, presenting and communicating with clients, and debugging. I had experienced crunch time before when taking an multiplayer online game course, but this time I was more prepared. I had learn to code in a more modular manner and with easy to understand functions to ensure integration was as painless as possible. I do well presenting for a crowd but I still have plenty to learn when speaking closely with peers about technical details. The practice I gained here helped me learn how to better articulate myself.

Overall, much of what I have learned through progressing through the project and working with our LCDR team I can definitely state as a lifelong learning experience. I learned how to properly research, plan, self-learn, when to ask for help, and above all to communicate and just get things done. This a memorable experience I will be applying to all my future projects.

**4.IV Natalie Gallegos:**

When I first started working on this project, I basically knew the “bare bones” for programming, as I had only taken some the core requirement courses leading up to the graduate program, and not much else in the way of electives. For our project, the goal was to detect craters from images, and to calculate the depth and diameter of each detected crater. I felt overwhelmed at first, but as the year progressed, I learned new techniques, and found myself able to accomplish the tasks at hand.

The biggest takeaway from our project was implementation of the project itself. Our project heavily focused on computer vision techniques and image processing, both of which I had zero experience in along with the rest of my group. We took two courses offered by Dr. Kang as a group, to gain an understanding of the subject matter. Through these courses, I learned about the different techniques used for creating binary images, the importance of threshold values, edge detection, feature extraction, image filtering, smoothing, just to name a few. I even learned about machine learning, algorithms such as k-means clustering, nearest neighbor, which was a relief to me, since I had no knowledge or experience with machine learning, and one of the papers I initially read about used some machine learning for rock detection. Picking up that little bit of machine learning became a valuable asset as I’ve used it on other projects, and I’m sure all I’ll need the other techniques I’ve learned about in the near future as well.

For our project, we used additional tools and libraries to make our implementations easier. For most of our image processing we used OpenCV, which is an open source image processing library. I learned how to install OpenCV with java bindings, and how to use it with java. I learned about the core structures, even picked up a little bit of python in order to understand the documentation. We also had to use LRO images, which when downloaded, came in disk format. We used a tool called GDAL to convert to other image formats and crop them out. I learned how to install GDAL on my computer using python bindings, and learned how to pull some metadata from images, change formats, and crop images.

Aside from the project itself other lessons, such as managing time, prioritizing, and communication were also worthwhile. Making time, not just to work on the project, but adjust my schedule with my colleagues for meetings was not trivial. On top of managing time to work with my teammates for this project, and time to do my own portion of the project, I also needed to divide time with assignments from other courses, and still manage all of this around my part time job. Prioritizing was key. Devoting the most time to projects or milestones that had deadlines sooner rather than later. Communicating effectively was another invaluable lesson. Teamwork depends on communication. Communicating in a friendly and professional manner to my colleagues to express ideas, ask questions, or delegate work wasn’t easy for me, I tend to be more reserved. However the need to get tasks done, and report our progress and our liaisons questions pushed me to speak up, and in time I was no longer nervous or shy about talking during our group meetings.

In the final stages of the project is where I learned a few more key lessons. Over the last two months our work consisted mainly of integration of our individual programs into a single larger program. We worked with GitHub to accomplish this, which I had never used before. I can now clone repositories and add to another’s code and merge new lines of code with old. Along with integration, I had to learn how to read another person’s code, and adapt my own program, or write new methods and modularize our classes to make integration easier. Communication was also the key here, being able to explain what paragraphs of code are used for, and documenting code so that others understand and can use my work as needed. Finally, I learned integration of different programs is not an easy task, and it requires an ample amount of time for debugging and making sure all aspects of the program are functioning together correctly.

I consider myself lucky to have been a part of this project, and to work with this team. In addition to forging new friendships, I have learned that I can perform quite well on challenging projects. I’ve learned new about new tools, libraries, and how to take things in moderation. I am grateful for all I’ve been able to accomplish and learn this year with the help of my teammates, advisors, and liaisons.

**4.V Tony Liang:**

For this project, I have learned about some libraries and software. I have to learn about the OpenCV and GDAL libraries and the Git software. I spent extensively on learning OpenCV and moderately on learning Git. I did not learn much on the GDAL library. The OpenCV and GDAL libraries are necessary for this project in order to preprocess and process crater images before doing crater detection and crater diameter and depth calculations. I spent time on learning Git so that my team members and I can integrate our programs easier and more efficiently. Instead of emailing each other changes to our code or using Google Drive or Dropbox which are slow and inefficient, Git is the better option. Since we all have a great background on Java, we decided to do this project using Java. Therefore, we did not need to learn a new language. We also have to store crater metadata in a database. Since we are also familiar with MySQL, we decided to use it for our project.

For OpenCV, our team took a graduate course about Computer Vision by our project advisor that taught us about image preprocessing and processing as well as programming using the OpenCV library. We have to install the library, configured it correctly, and read its API to learn about the library. In the winter quarter of 2015, we began project code implementation at the same time that we took that course. Understanding how to use the OpenCV library is essential for our project’s completion. We need to use this library for crater detection. The course helped me understand the basics of the library and had me practice so I am comfortable with using the library. I also discuss with my team members and look online to get a better understanding of how OpenCV commands work. The OpenCV library is the driving force that gives us the ability to detect craters.

For GDAL, our team needed to use it to crop crater images to TIFF format before doing any image processing. We used the GDAL library because it was recommended to us for converting IMG format images to TIFF format images. We have to install and configure the GDAL library. At first, we needed to use GDAL so that we could extract metadata from the crater images we decided to use for our project. However, it proved to be not very useful for it. It did not extract the more useful metadata of the crater images. So, we decided to use it only to crop crater images. The crater images we acquired are in IMG format, but we could not use that format for image processing and crater detection. So, we agreed to convert the images to TIFF format instead. We went online to learn how to crop images using the GDAL library.

For Git, our team used it for better and faster project workflow. It is especially important when we have to integrate our individual programs together in the spring quarter of 2015 when we are expected to submit our project to our liaison. We began learning how to use Git during the spring quarter. We all decided to follow tutorials and get assisted from each other to get familiar with Git and its commands like initialization, commitment, pushing, pulling, merging, and so on. Along with Git, we also use GitHub after setting up accounts for it. Many of us are already familiar with GitHub. So, we did not need to be taught about it. We used the command-line version of Git for this project instead of a GUI application of Git.

After learning these new libraries and software, I’ve learned that whenever I start learning something new, it was mostly hard and complicated. However, after setting everything up and with some practice, these new libraries and software were not that difficult after all. With some patience and stepping through the process, everything began to come together and clear. Keeping this in mind, I will have a better time learning new things to come. I’ll not be using more time trying to understand new things as before. I’ll be able to learn new things efficiently and effectively in a timely manner. This is utmost important when working on a job as well as for my everyday life.

**5.0 ARCHITECTURE AND DESIGN**

**5.1 Image Retrieval**

There are two options for image retrieval in Ringtoss. One, you can either select a specific image, and a specific region on the image in pixels, and ringtoss will download, crop, and run crater detection and recognition on the cropped image. The second option allows the user to select a region using longitude and latitude, and all images which contain that region will be downloaded, and appropriate regions cropped out from each image for crater detection and recognition. Once the image(s) are cropped out, they are provided to Ringtoss.

**Note**: Our algorithms rely on metadata provided from each image to crop out the region of interest using longitude and latitude, however the data provided for the LRO images is not in double precision, and thus creates discrepancies when attempting to match crater to crater in the same region. Until we can resolve this issue, ringtoss runs on a single image, and uses pixel coordinates instead of longitude and latitude.

**5.2 Ringtoss:**

Ringtoss is our main program. It is a pipeline that starts by running three crater detection algorithms in sequence: Hough Transform, Light and Dark pairing, and Template Matching. The results produced from these algorithms are combined and then filtered to remove duplicates, then fit ellipses and crater candidates. The results are further filtered out using machine learning with a convolutional neural network, The remaining results are then given to data extract for calculations of depth and diameter for each crater. Results are then finally stored in text form, and in a database. Each of these steps will be explained in detail in the subsections below.

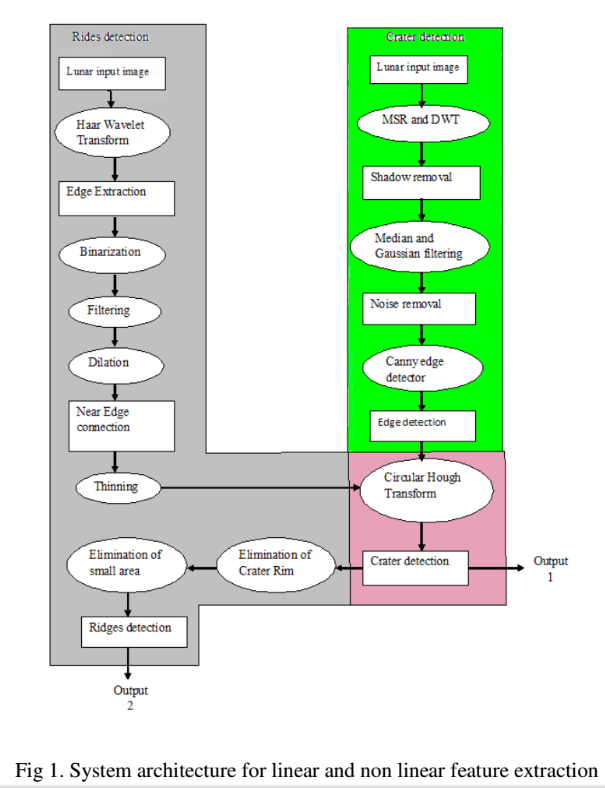
**5.2.A.i Light and Dark Pairing**

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. All these functions are implemented using the OpenCV image processing library.

After testing multiple images, we found that different size craters are detected using varying levels of erosion and dilation. This optimum levels are employed and the crater detection is split up the into three levels; small, medium, and large craters. We eliminate any duplicate craters that are detected between the three levels and save the results to a crater list. The list then moves onto the duplicate removal method, comparing the results with the circular hough and template matching detection results.

**5.2.A.ii Circle Hough**

Crater detection by means of the circular hough transform is based of 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. The methodology used in LCDR is a condensed version of the entire methodology proposed in the paper.



From the crater detection section of the methodology we use a single scale retinex algorithm as our “shadow removal”. The rest of the methodology remains the same: the filters, canny edge detector, and circular Hough transform. This is all done using OpenCV’s image processing library, Imgproc. Imgproc has algorithms for median and gaussian filters, canny edge detector and the circular Hough transform.

Circle Hough will run after image retrieval, alongside the two other forms of crater detectors: dark & light pairing with ellipse fitting and template matching. Once an image is passed into Circle Hough it will run through the preprocessing stages for shadow and noise removal:

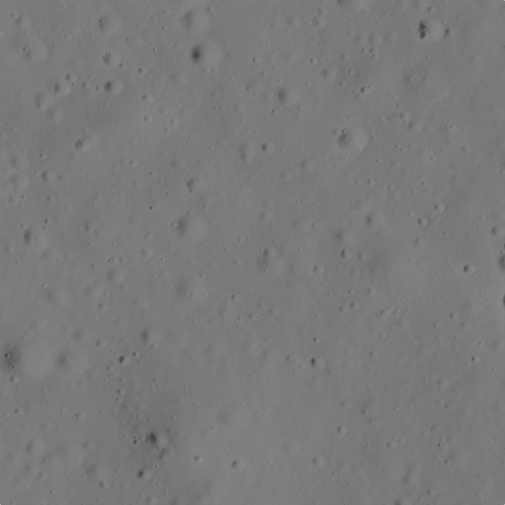
 

Fig 2. Input image Fig 3. Image after preprocessing

Following preprocessing, the edges of the image are extracted through the canny edge detector:

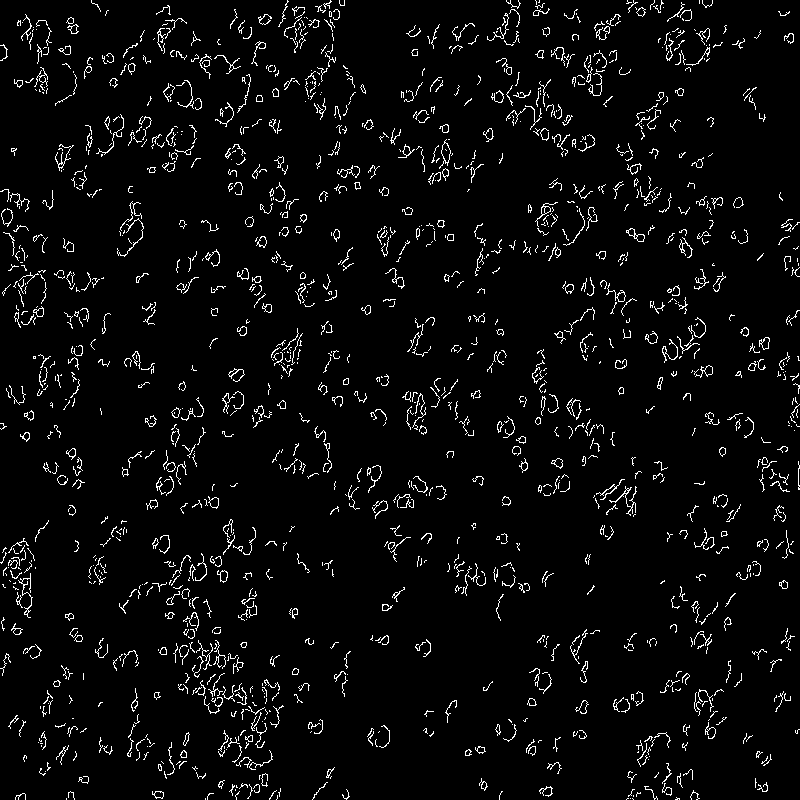


Fig. 4 Image with edges from canny edge detector

The edges are fed into the Circular Hough Transform which iterates through various radii ranges from 5 pixels to 175 pixels and beyond with specific parameters to those ranges, searching for circular shapes. It will overlay a circle or bounding box when it detects circular shapes.

Information such as, center, radius, and bounding box areas are saved into a list of craters. The list of craters is passed to the next step of the Ringtoss pipeline: duplicate removal.

**5.2.A.iii Template Matching**

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

This was part of an adjunct project completed by a graduate student.

**5.2.B Duplicate Removal**

After all three crater detection algorithms have placed crater candidates into a single list, the crater list must be filtered to remove duplicate craters by merging them. For duplicate removal, we first place unique craters 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 we later use for crater recognition and data extraction.

**5.2.C 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.

**5.2.D Machine Learning - Convolutional Neural Network**

Tries to recognize craters from non-craters using an artificial neural network.

This was part of an adjunct project completed by a graduate student.

**5.2.E Feature Extraction - Depth and Diameter**

The methods for extracting the diameter and depth from craters are adapted from a paper by James Andre Clayden, which unfortunately can no longer be viewed Online. However there are still other papers that provide the basic ideas behind James A. Clayden’s approach, one which can be viewed at <http://www.lpi.usra.edu/exploration/training/resources/measuring_meteor_crater>.

Computing for depth and diameter begins after the machine learning portion of Ringtoss. 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. We take the average of the width and height of the bounding box, and consider that to be the diameter of the crater. We use the metadata of the corresponding image for each crater to scale the diameter from pixel length to meters. We use the following formula to compute the diameter.

,

Where is the average from the width and length of the crater’s bounding box and, is the average from the scaled pixel width and scaled pixel height provided from the metadata file.

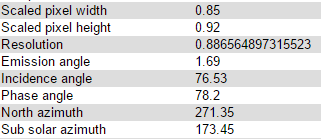


Image of metadata some metadata used calculations

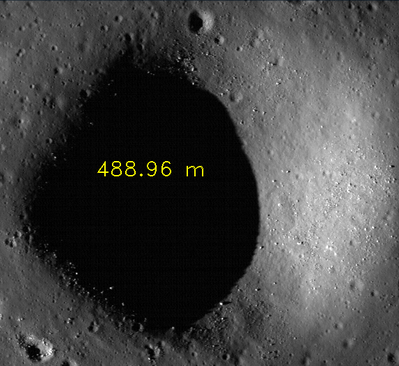
Since the emission angle for LRO images was less than 10°, we did not need to account for curvature of the moon. However if the emission angle for the images used exceeds 10°, then you must account for the curvature of the moon, and will need the longitude and latitude coordinate for each crater center. Your equation will then become:

(2)

where,

(3)

By applying these equations we were able to derive the diameters for each crater with a small difference in value between our calculations and values cited from <https://the-moon.wikispaces.com/Introduction>.



The Yoshi crater is cited to be 500m in diameter. Our results how 488.96m.

To compute the depth of a crater, more metadata is required. We measure the shadow for each crater, and use the sun azimuth and sun elevation angle, coupled with pixel width and pixel height for scaling, to compute the depth. Our approach follows the diagram below:

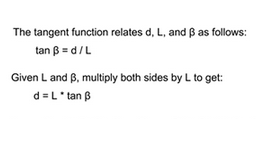
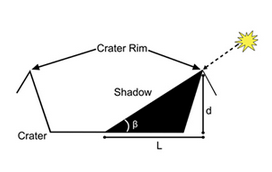
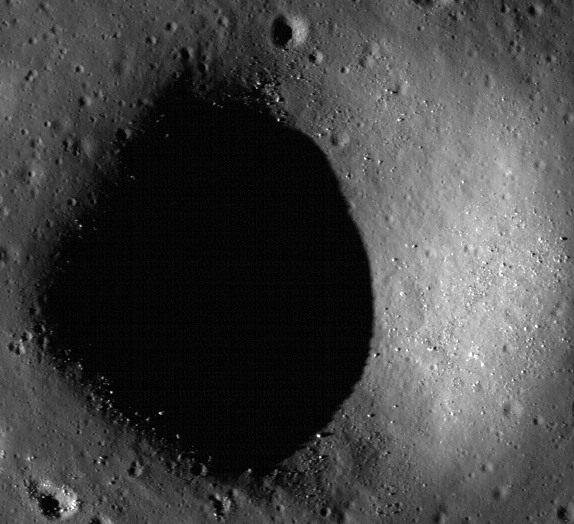


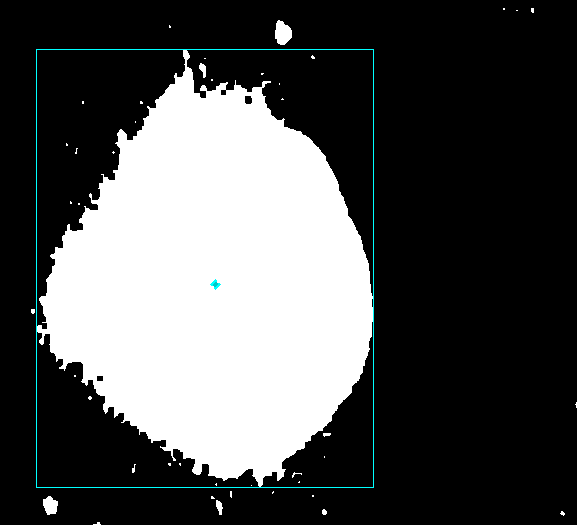
diagram from [www.lpi.usra.edu/exploration/training/resources/measuring\_meteor\_crater](http://www.lpi.usra.edu/exploration/training/resources/measuring_meteor_crater)

To extract shadows for our depth computations, we start with image processing. We first convert the image to grayscale. A median filter is then applied to remove any noise. Next the image is converted into a binary image. The threshold value is obtained by finding the threshold value from Otsu’s method and subtracting the standard deviation, from it.



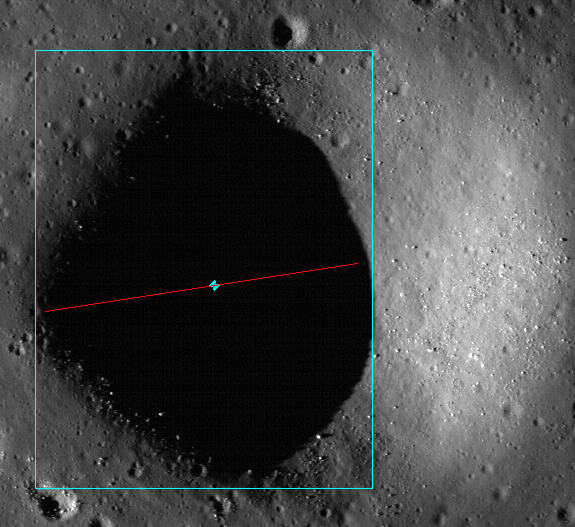
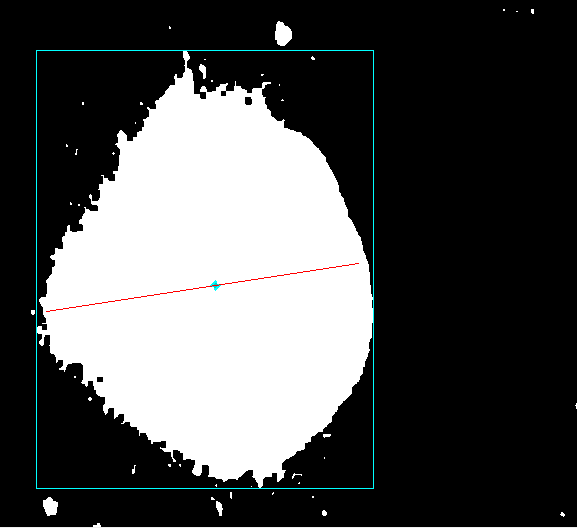
The yoshi crater converted into a binary image using threshold value where . The image on the right shows the shadow of the crater, in white.

After creating a binary image of shadows, and do a bit more image processing by way of erosion and dilation to extract clearer shadow regions, finding additional information about the shadows, such as shadow centers and bounding boxes is the next step. This is done by finding contours of the binary image, and using the contour with the largest area as the target shadow in order avoid selecting shadow centers of small nearby craters.



The center of the the largest shadow and it’s bounding box displayed in cyan.

After extracting the needed shadow information, we proceed to find the two points on the edges of the shadow, which will give us it’s length. We accomplished this by starting from the shadow centroid, and “marching” along the sun angle in both directions away from the center, and kept recorded the position of the two pixels that were furthest from the shadow center, but still a “shadow pixel”, or “white pixel” contained inside the shadow’s bounding box. To march along the sun’s direction we used the subsolar angle provided in the metadata from the images and kept the orientation provided by the metadata, in which 0° is always with respect to an invisible line starting from the center of the image, extending to the right of the image.



Images showing a red line drawn across the length of the shadow, using the two end points obtained from marching away from the center of the shadow using the subsolar angle.

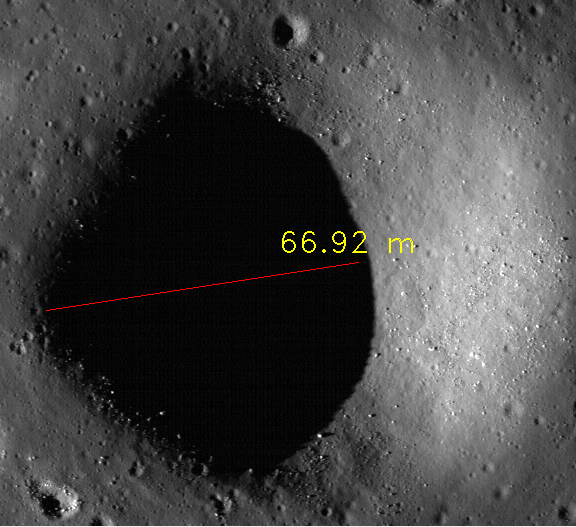
Using the two points obtained from the previous method (endpoints of the red line in the image), we find the euclidean distance between the two points to obtain the length of the shadow We proceed to to calculate the depth by first scaling of the shadow by again using the average of pixel width and height from the metadata provided.

,

If the incidence angle is greater than 10, then equation (3) should be incorporated above as well to reduce error. We then find the elevation angle by subtracting 90 and the incidence angle, , which was provided from the metadata.

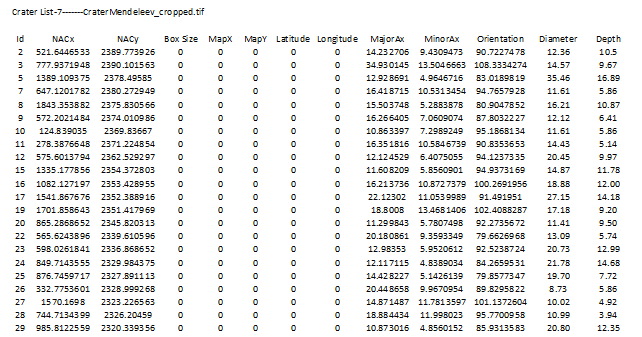
Finally we can compute the depth using the trigonometric function below:

Our results are comparable to those craters we tested on, although they do contain small error. Results can be further improved by using multiple images of the same crater at different lighting angles and combining results at each different angle.



The Yoshi Crater, cited to be 70m in depth

**5.2.F Output**

After computing the diameter and depth of each crater, the results will be outputted into a text file and inserted into a database. Each row represents a crater and information about it. The columns are listed as crater number, Narrow Angle Camera (NAC) x, Narrow Angle Camera (NAC) y, bounding box size, MAP x, MAP y, longitude, latitude, major axis, minor axis, orientation, crater’s diameter, and crater’s depth. The NAC x and y coordinate is the location of the camera that took the image of the crater. The major and minor axis are lengths and the orientation is the angle in degrees for the construction of the ellipse that fits the crater into it.

The results in a text file.

**6.0 CONCLUSION**

The LCDR team succeeded in creating a crater detection algorithm by means of a hybrid approach that is within 5% of JPL results in terms of F1 score. We used three different forms of crater detectors including a Circle Hough, shadows and highlights pairings, and template matching approach. Each have strengths which make up for the weaknesses of others. The shadows and highlights pairing algorithm does well in detecting craters with distinguished shadows and does not necessarily look for a particular shape. The Circle Hough algorithm will detect craters which may have been hidden by shadows (potentially missed by highlights and shadows pairing) and craters which are young with apparent circular rims. Template matching does well in finding older craters with vague rims missed by the Circle Hough. Unfortunately, this does result in over detection of craters including more false positives.

There is room for improvement. More accurate bounding boxes will result in greater true positives and less false positives. The machine learning section of Ringtoss can be improved by employing unsupervised learning techniques. To improve performance the crater detection algorithms can be parallelized. Code for longitude and latitude linear interpolation has been implemented but existing data is inaccurate and results in conversions with large errors. Multiple image processing was unable to be incorporated due to inaccuracies from longitude and latitude linear interpolation, but can be added later using different libraries for longitude latitude to pixel conversions, along with boulder detection by use of sun angles and shadows. Finally, the existing Ringtoss pipeline can be incorporated into the LMMP website.