2013

Statistical Analysis of Simple Martian Impact Crater Morphometry

Lynn M. Geiger
lgeiger@wellesley.edu

Statistical Analysis of Simple Martian Impact Crater Morphometry

Lynn Marie Geiger
Adviser: Wesley Andrés Watters

Submitted in Partial Fulfillment of the Prerequisite

for Honors in Astronomy

April 2013

Abstract

Mars, a planet with a tenuous atmosphere and starved of surface water, is a prime location for studying impact craters. Earth's thick atmosphere stops small craters from forming, and erosion destroys the craters that do. The size range of Martian craters is much greater and craters last much longer than terrestrial craters. The variety of Martian terrain makes it possible to study effects of geology on crater morphology and crater modification, making Mars a much more interesting study location than the Moon. In this project, we investigate the effects of crater size and target geology on crater shape for small ($D < 5$km), simple impact craters. For example, we see morphometry differences between Vastitas Borealis, sedimentary rocks and volcanics. In addition, we investigate changes in morphology between fresh and modified craters. To study craters on a global scale, we adapted computer programs to generate digital elevation models using stereo images from the HiRISE camera on the Mars Reconnaissance Orbiter and developed new software and algorithms to extract crater rim traces and crater shape statistics automatically. We show that moderate crater age has no apparent effect on rim roundness, and that crater flank elevation profiles in the Northern lowlands have lower power-law decay exponents ($\alpha_F \leq -4$) than the global distribution ($\alpha_F \leq -6$). We identify a possible crater shape transition at $D \approx 300$m to increased rim sharpness. Our results show geologic dependence on crater formation, and can be used to test models of crater formation and modification in the future.

1 Acknowledgments

I would like to thank Dr. Wes Watters for his fabulous mentoring and patience, without whom this project would not have been possible. I would like to thank my thesis committee: Dr. Wendy Bauer, Dr. Kim McLeod, Dr. Katrin Monecke, and Dr. Wes Watters, for their guidance, support and their time. Special thanks to Dr. Glenn Stark for being my honors thesis visitor. In addition, I would like to give my deepest gratitude to Dr. Dick French, for his help with my distribution requirements.

I also would like to acknowledge and thank Michaela Fendrock her vast contributions to this project and her help with computer coding. Thanks to Rose Gibson, for spending tireless hours looking at HiRISE images, totaling 5% of the Martian surface, to pick out craters. I would also like to thank my fianceé, Alejandra Ortiz, for her loving support and tireless help editing. Without her, I would have gone crazy months ago.

Many thanks to the Schiff Fellowship for providing funding, to allow this work to be presented at the Lunar and Planetary Science conference. Thanks to the Massachusetts Space Grant for the summer funding that allowed me to get a head start on this research.

Contents

List of Figures

List of Tables

2 Introduction

Impact craters are the most common geologic feature in the Solar System. Cratering happens at all sizes, from the micron-scale impacts of space weathering to impact basins that can encompass half of a planet. Because it is easier to see larger features from far away, most cratering studies have focused on large craters. Statistically, smaller craters should be much more frequent than the large craters, but the majority of crater formation models are based on and tested with the infrequent large craters. Our study is a statistical morphometric investigation of small Martian impact craters, 20m - 5km in diameter, found in HiRISE stereo images. These craters include fresh and slightly modified craters, with characteristic depth to diameter ratios (d/D) ≥ 0.15. We characterize crater morphometry parameters as a function of crater size, freshness, and target geology, in order to discern trends in morphometry and to test existing cratering models.

In this thesis, I will present a background on impact craters: what they are, how they form and an overview of previous studies on craters. I will then describe the steps we took to conduct this study, starting with how we created our digital elevation models (DEMs). I will explain how the computer programs we wrote automatically extract the morphometry of a crater, including crater planform statistics (2-D shape), three dimensional rim traces, and metrics that describe the average crater profile (or transect). Then, I will present the results of the study and discuss the implications of our results in the context of previous studies. Finally, I will present the conclusions drawn from this study and suggestions for future work.

3 Background

3.1 Martian Geology

In our study, we attempt to paint a global picture of Martian craters. The craters we used are displayed on global elevation data from the Mars Orbiter Laser Altimeter (MOLA) in Figure 1 (Smith et al. 2003). The surface of Mars was shaped over the course of 4.5 billion years, which are divided into three geologic epochs: the Noachian (>3.5Gyr), the Hesperian (ending 2.9-3.3 Gyr), and the Amazonian (2.9Gyr to current) (Hartmann and Neukum 2001). The majority of Mars' geologic activity occurred in its first 1.5 billion years (Carr and Head 2010). The Noachian was dominated by heavy crater bombardment and high erosion rates owing to a wetter climate (Golombek et al. 2006). There is also evidence of sustained surface water and lake formation in craters during the period (Fassett and Head 2005).

The Hesperian was an age of high geologic activity, though the high cratering rates had decreased. Volcanism during the epoch resurfaced up to 30% of the planet (Head et al. 2002). It is hypothesized that the northern lowlands of the Vastitas Borealis region were covered by an ocean during this time, as evidenced by relict shorelines and coastal erosion (Parker et al. 1993). This ocean may have been covered in ice, and slowly depleted over time; the remaining traces are stored in the ice deposits found in the northern plains (Clifford and Parker 2001).

Currently, Mars is a land ravaged by wind and ice. In the Early-Amazonian, the exuberant geologic activity of the Noachian and Hesperian was quickly winding down, leaving only the wind and ice we see today to transform the surface (Hartmann and Neukum 2001).

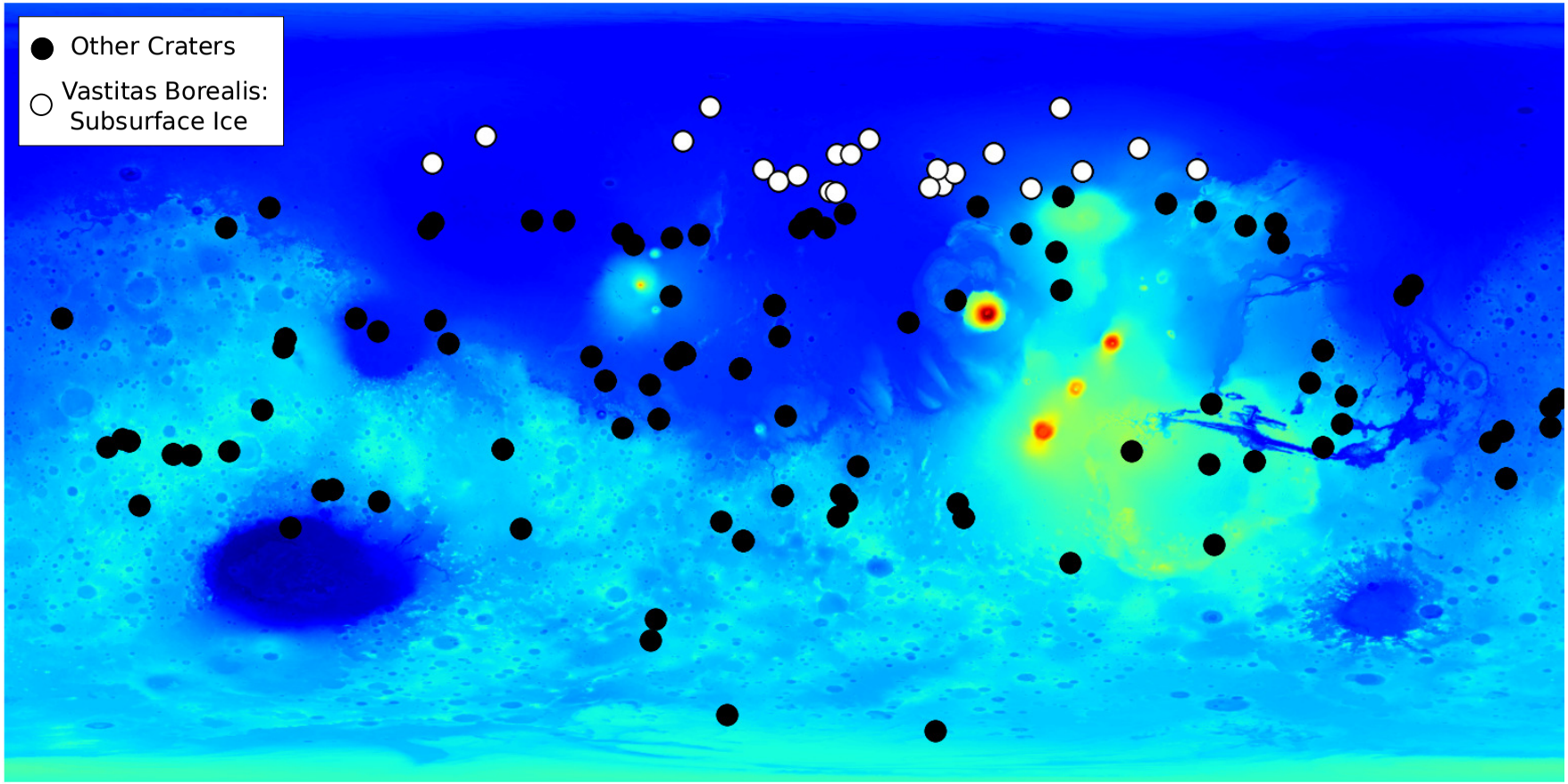

Figure 1: MOLA global elevation map (Smith et al. 2003). The crater locations are shown with circles; white circles represent craters in the Vastitas Borealis Formation, latitude (ϕ)$> 45^\circ$ and elevation (z) < -3km.

3.2 Impact Cratering Basics

3.2.1 Crater Anatomy

The cratering process requires only two things, a projectile traveling at a supersonic velocity and a target. There are two main classes of impact craters: simple and complex. Simple crater cavities are shallow depressions that form in the "strength regime", where the impacting energy is small enough that crater formation is dictated by target strength. Though stereotypically described as bowl shaped, craters with polygonal planform shape are common. Complex craters, which are much larger, have terraced walls and central peaks or rings in the crater floor. Complex craters form well into the "gravity regime", where the effects of gravity surpass the effect of target strength in crater formation. This study focuses on small, simple impact craters.

A simple crater has three main components: the bowl or cavity, the rim, and the flanks (Figure 2). The rim of the crater is the boundary between the cavity and flanks. Theoretically, fresh crater rims will be sharp and become rounded with age from weathering and erosion, as seen with most geologic features, though no quantitative assessment of rim roundness exists currently. The crater rim will be the highest point on a radial profile of the crater, as a combination of uplifted target rock and thickest deposits of ejected material. The flanks of a crater slope away from the rim towards the undisturbed topography, with an exponential decay shape. The flanks are comprised of uplifted target rock and an ejecta blanket, which is the material thrown from the crater cavity during formation, including some material from the impactor. The ejecta will have an inverted stratigraphic sequence, symmetric about a hinge point slightly below the crater rim. In some cases, the ejecta blanket will end with a steep rampart, commonly referred to as a distal toe. Craters with abruptly ending ejecta are called pedestal or rampart craters.

3.2.2 Crater Formation

The process of crater formation lasts seconds to only a few minutes. The cratering process can be split into three stages: contact and compression, excavation, and modification (Melosh 2011). In the first stage, the impacting body makes contact with the target and sends a shock wave through both materials. This pressure shock propagates down and outward through the target material.

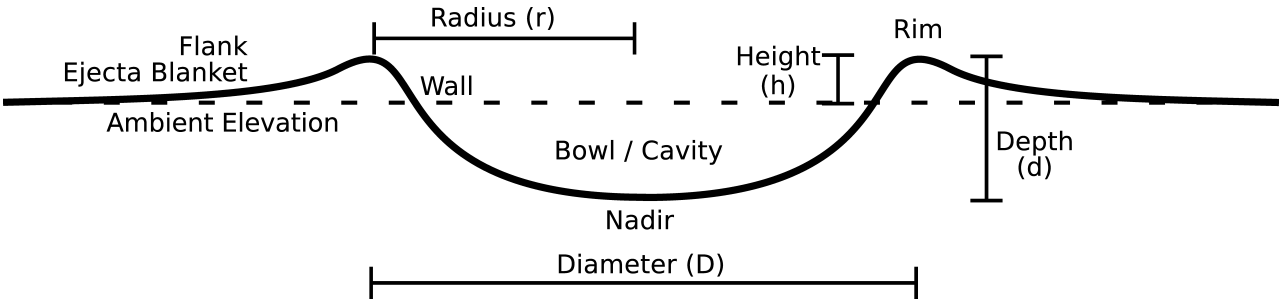

Figure 2: Diagram of a Simple Crater.

Highly shocked material near the surface can be shot out in high-velocity jets, achieving up to five times impactor velocity. The compression stage comes to an end as the shock-wave reaches the back surface of the projectile and a pressure relief wave propagates back down to the projectile-target interface. At this point the majority of the projectile has been vaporized due to the immense heat and pressure. The impactor absorbs up to 50% of the total energy released in the collision. The rest of the energy is transferred into the target and dissipated through heat and crater excavation.

During the excavation stage, the majority of the crater features are created. The initial shock wave, closely followed by the decompression wave, continues to propagate through the target. The weakening shock wave can create a spectrum of minerals through "shock metamorphism." The suite of created minerals are dependent on the composition of preexisting rock and the varying intensity of the shock wave. Under the highest pressure, melting and vaporization of the target rock may occur. The shocked material is ejected from the crater at an angle of approximately 45° from the rim. A small amount of unshocked material may be ejected, known as spallation, due to a pressure of zero at the surface and shock wave pressures directly below. At the end of the excavation stage, the transient crater has a depth to diameter ratio of 1/3 to 1/4. Only material in the first third of the transient crater depth is actually ejected; the rest is merely pushed downward, deforming the original target rock. The thickness of the ejecta ($\delta(r)$) can be approximated using Melosh's equation 6.9 (2011):

$$\delta(r) = f(R)\left[\frac{r}{R}\right]^{-3\pm0.5} \tag{1}$$

where r is distance from crater center and R is crater radius, as seen in Figure 2. $f(R)$ is a function dependent on crater size, which is not well constrained. In some cases, particularly with very large craters, the cratering process ejects surface materials ballistically with high enough velocities that they may create smaller "secondary" craters upon impact. Secondary craters are hard to distinguish from small primary craters, and therefore, can adversely affect surface dating conducted through crater counting. Secondary craters appear to have shorter, more clumped ejecta than primary craters (Calef et al. 2009).

The final stage of crater formation is modification. At this point, the energy of the impact has dissipated and the excavation flow will stop and gravity takes over. Any debris that has not escaped falls downward towards the center of the crater. Slumping walls and the in-falling debris create a brecciated lens of material in the crater bottom. Underneath the lens, melt pockets may be found. The final depth to diameter ratio is approximately 1/5 for all simple impact craters (Melosh 2011).

3.3 Previous Studies

Many studies have been conducted to study various aspects of impact cratering. One such aspect is the polygonality of craters. Polygonal craters have been identified on all of the terrestrial planets and on asteroids and comets (Öhman et al. 2010). Eppler et al. (1983) conducted a study of complex lunar impact craters ($D > 15$km) using Fourier shape analysis. They found that low harmonics (2,3,4 and 6) were predominately controlled by preexisting variations in terrain, and therefore, form during the crater formation process. Higher harmonics appeared to be influenced by post-formation modification by surface processes, which produce only moderate irregularities in shape. Eppler also postulated two models for the formation of polygonal impact craters (Figure 3 A and B). The first theory claims that during the excavation phase, excavation happens parallel to planes of weakness in the underlying geology, such as faults or joints. The second model, derived mostly for understanding complex craters, states that the crater excavation is symmetrical and polygonality is achieved during the modification stage, when crater walls slump along planes of weakness. In the first model, faults bisect corners. This theory is consistent with observations from Meteor Crater, Arizona, where the crater diagonals are within 13° and 6° of jointing found in

the topmost and second geologic unit, respectively (Roddy 1978). The second model is 45° out of phase with the first, and jointing is parallel to the walls of the crater. Öhman et al. (2008) presents a third model for polygonal crater formation, in which the excavation of the crater forms walls parallel to joints through thrusting along faults. The craters in Öhman's study were all complex craters, though he claims the model can predict formation in both regimes.

Watters et al. (2011) propose an alternate model of polygonal crater formation, based on a study of Endurance Crater, Mars. The model states that a "stellate" transient crater planform shape is created during the excavation stage. In this model, the excavation flow intensifies along preexisting fault systems creating a scalloped shape planform with corners bisected by faults. Concave prominences, or "knurls", may be unstable and during modification stages, slump to produce straight sections between corners. One such prominence may be preserved in the northeast side of Endurance Crater. Differences between this model and previous models can be seen in the path of the ejecta hinge and the shape of exposed beds, if they exist. Stellate transient craters will show almond shaped, "lenticular-cresentic" layering patterns, whereas polygons formed during excavation would have linear bedding, and Eppler's wall slump model for polygonality would produce diamond shaped features in the crater corners (Figure 3).

In a study of Meteor Crater, Poelchau et al. (2009) modeled excavation with respect to joint orientation. Poelchau et al. calculated that excavation parallel to jointing requires 1.41 times less force than excavation 45° to the orthogonal joint sets, which supports Eppler's first model and Watters' model. The study also found a strong correlation between increased rim uplift and distance of the rim from the center of the crater. In the case of Meteor Crater, this is attributed to inter-thrust wedging, a mechanism involving consolidated pieces being inserted into weakened bedding planes during the crater excavation stage. Watters et al. (2011) propose the inter-thrust wedging that Poelchau identified may have actually formed all around the crater, but be lost as the prominences of a stellate transient crater collapsed. We test these hypotheses by studying how the rim height correlates to crater radius.

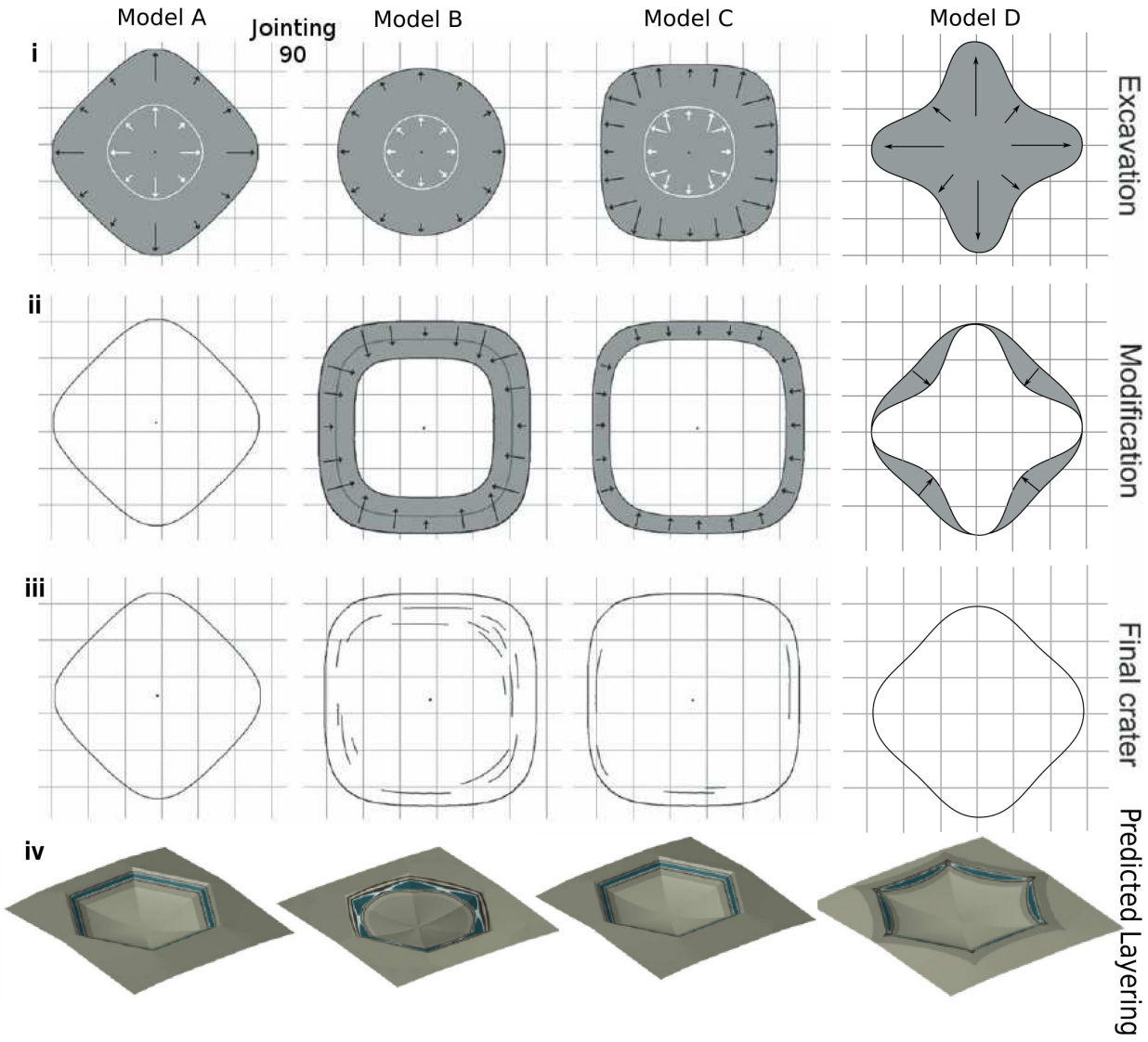

Figure 3: Illustration of the four models for the formation of polygonal craters, showing the excavation stage (i), the modification stage (ii) and the final planform shape (iii) for a square crater. Watters' predicted layering patterns for each model are shown for a hexagonal crater (iv). The models are A) Eppler's Excavation Model; B) Eppler's Slumping Model; C) Öhman's Fault Thrusting Model; D) Watters' Stellate Crater Model. The background grid represents preexisting planes of weakness in the target material. Figure adapted from Eppler et al. 1983, Öhman et al. 2010 and Watters et al. 2011.

Other studies have also found that underlying geology can have drastic implications for impact crater morphologies. Stewart and Valiant (2006) compared crater morphometry in Martian highland plains and lowland plains. They found the effective yield strength of different locations on Mars by studying the simple to complex crater transition, which denotes the shift from target strength dominated to gravity dominated cratering mechanics. Areas in the highlands have strengths as low as soils, and have complex craters as small as 3km, whereas the lowland formations exhibit strengths akin to consolidated rock, with simple craters up to 10km in diameter. Stewart and Valiant used depth-diameter ratios (d/D) and crater cavity volume to infer geologic layer thickness and relative strengths; a crater that forms in a strong target material will retain a larger volume than if it formed in a weaker material. They concluded that the average simple lowland crater would be 1.5-2 times deeper with a 50% larger cavity volume than the highland counterpart.

The modification of craters over time is a topic with great implications for planetary surface age dating. Different places have unique processes that mold crater shapes over time. The moon, for instance, having no atmosphere, thus no weathering in the traditional sense, is subject only to solar wind and diffusion caused by the bombardment of small impactors and thermo-mechanical weathering. On Mars, crater degradation regimes can change spatially, and even temporally. In the Sinus Sabaeus region, Noachian craters appear to have been modified by fluvial processes, but post-Noachian crater modification is predominately aeolian infilling, with very minor contributions from mass wasting (Forsberg-Tayor et al. 2004). The study used models of various modification processes and tested the models using Viking and MOLA data.

The Mars Exploration Rovers have provided us with a great opportunity to characterize Martian surface properties in two locations with great detail. Gusev Crater, and craters in the surrounding region, appear to have a short period of strong mass wasting, followed by slow, sediment-starved aeolian weathering; limited infilling and saltation (Grant et al. 2006). Meridiani Planum, on the other hand, has large quantities of sediment, which increase crater infilling and wind abrasion (Grant et al. 2008). Meridiani is also subjected to more mass wasting than Gusev, which is illustrated in the many promontories and alcoves seen on craters in the region, weakened by increased abrasion. Though the process controlling degradation may be the same, the results can be vastly different.

3.4 Motivation of Study

While impact craters are very common in the Solar System, they are exceedingly rare on Earth, due to highly active weathering processes and a thick atmosphere. Only very large meteorites can maintain supersonic velocities while traveling through the terrestrial atmosphere, making the study of small, natural craters impossible on Earth. The low resolution imagery acquired of other worlds made it possible to study large craters on other planets, while remaining virtually impossible to analyze smaller craters. However, with the launch of the HiRISE camera in 2005, new, high resolution (25cm/pix) imaging data has become available for portions of the Martian surface, making the study of small surface features now possible.

Most relationships between crater shape metrics were derived empirically from large craters and have not been tested on small craters. Because the effects of target strength are poorly understood, laboratory cratering experiments, which utilize ballistics, do not scale well for full sized craters, and these models need to be verified for all size regimes separately. As we test the validity of existing models, we also discuss how morphometric parameters change with crater size for small scale cratering. Many of the metrics that we are investigating have not been used before (such as a robust rim roundness metric to characterize the difference between sharp and muted crater rims quantitatively).

In addition, correlations between morphometry and preexisting geology are poorly documented. Aside from the effect of jointing, what other target properties influence crater formation? We wish to understand how target strength and composition alter morphometric parameters. To answer all of these questions, we collect parameters from a large number of small ($D \leq$5km), simple craters across Mars, to conduct a statistical study of crater morphometry.

4 Methods

Our study of crater morphometry is based on digital elevation models of 200 impact craters, which are spread across Mars to attempt to characterize the global morphometry of Martian craters.

4.1 Digital Elevation Model Synthesis

Because high resolution digital elevation models (DEMs) exist only for a small portion of Mars, we created our own DEMs to analyze crater morphometry across the planet. To do this, I wrote a python script which allows a user to view HiRISE stereo image pairs, and select impact craters in the images. The initial crater selection requires the user to select the crater center and one point along the rim, to store an approximate crater radius. The coordinates are stored in meter space (instead of pixel space), and are passed to a script that prepares the images for the DEM synthesis program.

We used "ISIS" to prepare the HiRISE stereo image pairs for use in generating DEMs. The United States Geological Survey (USGS) created the image processing program ISIS, Integrated Software for Imagers and Spectrometers, for analyzing data from planetary imaging missions. ISIS contains methods that deal specifically with HiRISE imager data. Because the HiRISE camera contains 14 total CCD (Charge Coupled Device) imagers to cover a wider area, image preparation is a large endeavor. Ten of the CCDs are in the red band, which was what we used. To begin, each CCD image had to be projected with standard map coordinates to account for the orientation of the spacecraft, to create a length scale was the same in each CCD frame and across observations. Next, the color of the images was normalized with respect to one another, so that the grey-scale was equal across the images. This is possible because each CCD image overlaps the next one by 48 pixels. Once all the CCD images had the same grey-scale and map-scale, they were mosaicked to make a single image. The ISIS program has individual functions to run these processes, but we created bash scripts to streamline the process to run seamlessly.

Once ISIS preparation was completed, the stereo images of selected craters were cropped in two sizes; the first, designated "small", was cropped at approximately two crater radii from the center, and the second, designated "large", was cropped at five crater radii. Cropped stereo images (Figure

4 A) were then correlated using the NASA Ames Stereo Pipeline (ASP), to create individual DEMs (Figure 4 B). ASP is an open source software program designed by NASA to remove time consuming human dependencies from digital elevation mapping. The ASP stereogrammetry methods have been thoroughly tested for observations obtained by various spacecraft (Moratto et al. 2010; Broxton & Edwards 2008). ASP creates DEMs based on pixel mapping algorithms for stereo image pairs.

The pixel correlation algorithm takes a "pixel neighborhood", or group of continuous pixels, in one image and matches it to the same pixel neighborhood in the other image. The offset between the neighborhoods is recorded, and the correlation is repeated for all pixel neighborhoods in the images. The offsets are combined to create a "disparity map" that records how far each neighborhood moved from image to image, which is called parallax. Parallax is used to calculate the distance from the orbiting space craft to each point in the image. From these varying distances, relative elevation between regions is calculated to create a DEM. Pixel correlation for a single HiRISE stereo image would take over a week to complete using our available computer resources. This is why we chose to select and crop HiRISE images to only correlate areas of interest.

Regions where the stereo-image correlator failed were left blank (NaN, "not a number") in the DEM to minimize introduced error. NaN areas are seen as white space in our plots. The finished ASP DEMs were pasted to our post-processing software to remove artifacts, such as anomalous steps in elevation and areas surrounded by NaN values, called islands (Figure 4 C). Islands were removed because their elevation cannot be tied to the rest of the DEM, and they commonly exhibit aberrant elevations. The post-processed DEM is called the Island Purged DEM, or IP-DEM. Processed craters are stored in a python data structure called a gridmap object[1], which contains the IP-DEM and original, a grey-scale image, and information about resolution, crater location and crater size. Every crater is assigned a unique identification number (i), used in scripts and analysis databases. Individual crater DEM pairs, (large and small) were then fed to the feature selection scripts.

[1]The gridmap object is a data storage structure that was developed in-house to handle multiple layers of 2-D data for an individual map, in addition to information stored in the form of individual variables.

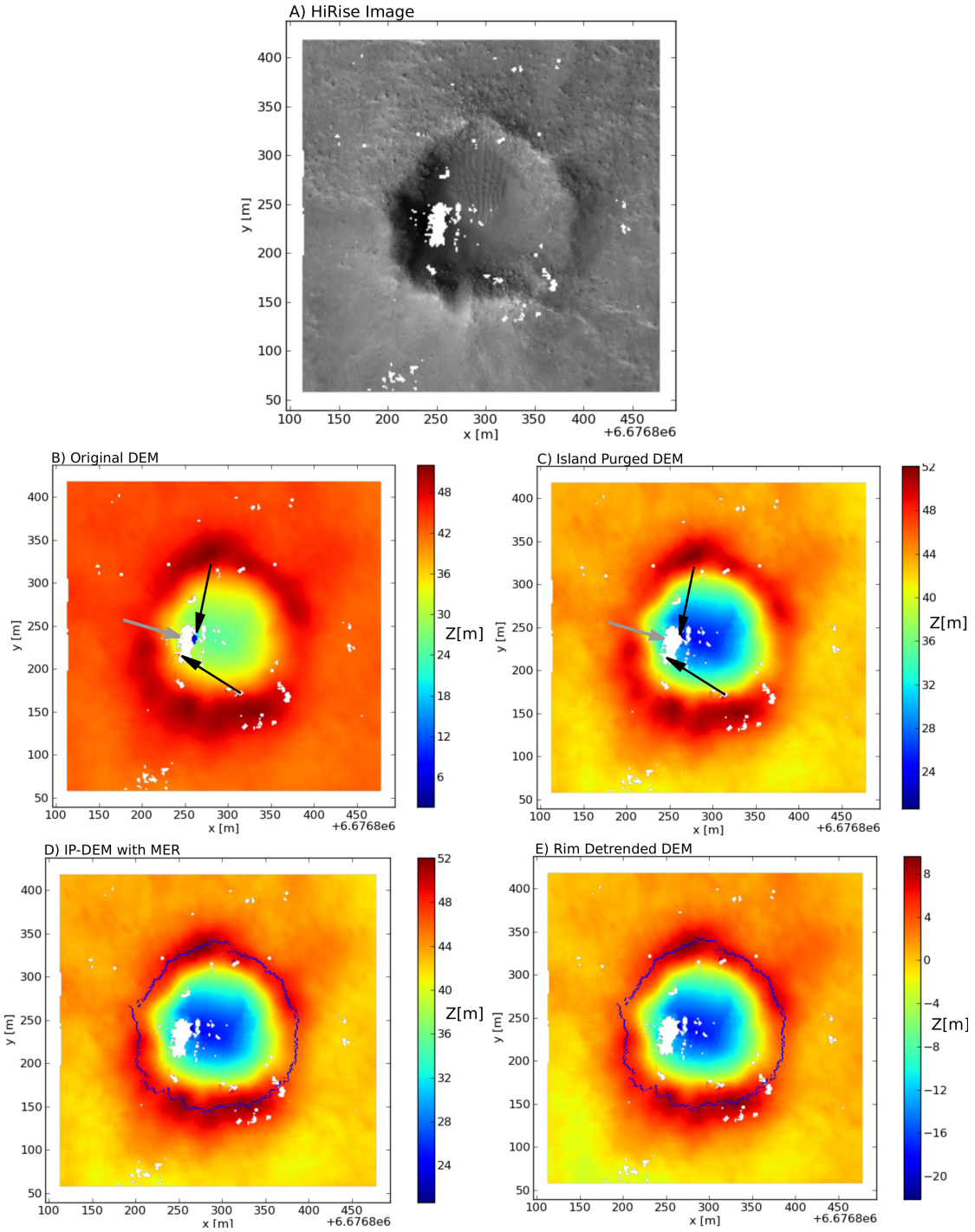

Figure 4: Illustration of the steps required to generate DEMs. First HiRISE images (A) are cropped and input into ASP. The resulting DEM (B) is purged of anomalies, such as steps (black arrows) and islands (gray arrows), resulting in the Island Purged DEM (C). The DEM is then rim-detrended, based on the Maximum Elevation Rim Trace described below (D) to remove background topography, and to set the ambient plane equal to zero (E). The example shown is crater 064, a heavily modified crater with $d/D = 0.151$.

4.2 Crater Rim Extraction

We used python and bash computer scripts to extract crater rims from the ASP DEMs. The rim extraction parameters and their optimal values are given in Table 1. To begin, the large and small gridmap objects of an individual crater are loaded into the rim extracting script. The center of the crater in the small DEM had been selected by hand in the previous crater selecting script, and the coordinates were stored in meter space (instead of pixel space). The crater centers in the large DEMs are slightly offset in meter space from the centers of the small DEMs as a result of map projection uncertainties in the ASP stereo correlation program. The rim selecting script, therefore, recalculates crater centers in the large DEM using the geometric center of a Maximum Elevation Rim Trace (MERT; see below for extraction technique). The newly calculated center is re-input into the MERT script, and the rim is calculated again. The process is repeated until the MERT geometric center deviates by less than a meter between runs. In a few cases, the center of the crater had to be corrected manually. The offset between the large DEM center and initial small DEM center is stored and used to translate coordinates between the two elevation models.

4.2.1 Digital Elevation Model Detrending

To minimize elevation bias of context topography in rim selection, the DEMs were detrended. The first step is to fit a plane through the elevations along two circles, sampled at 3 and 3.5 crater radii in the large DEM to avoid sampling the crater flanks. (If these detrending circles contained greater than 90% NaN points, the large DEM was flagged as unusable and discarded for the remainder of the analysis.) This plane is subtracted from the large and small DEMs, to create a "circle-detrended" DEM, which has been detrended with respect to ambient elevation.

Although the circle-detrending effectively removed large topographic gradients, smaller features would still be picked up by the rim finding script, so a second iteration of detrending took place. First the MERT was found in the circle-detrended small DEM. Next a plane is fit through the MERT, (Figure 4 D), and is subtracted from both DEMs. After this process, all points along the rim of the crater lay at similar elevations in the DEM, increasing the probability of the rim being a global maximum along a radial profile of the crater. The height of the ambient topography

is calculated by averaging elevations sampled from the large DEM at 3 and 3.5 radii, except for flagged craters, in which case ambient elevation is calculated with elevations at 2 radii sampled from the small DEM. Mean ambient elevation is subtracted from the final detrended maps, so that crater rim is positive in elevation and the cavity is negative. In the flagged cases, zero elevation is estimated after the rim selection using $h = 0.036D^{1.014}$, where D is depth of crater from the rim, and h is rim height above the ambient plane (eq. 6.2.1 in Melosh 1989). The resulting DEMs are "rim-detrended" and are used for the rim selection process (Figure 4 E).

4.2.2 Rim Extraction Methods

After rim-detrending, N radial profiles spaced evenly over 360 degrees are extracted from the small DEM using a simple first-order spline interpolation, sampling at half the DEM resolution. The profiles originate at the approximate crater center and end at 2 crater radii, P_R. Three different methods were used to select rim candidate locations on these profiles. One of each kind of "interest point" is shown on a sample profile in Figure 5.

The first method creates Maximum Elevation interest points (ME). These are considered the location of maximum height along a radial profile. A rim containing only ME points is the Maximum Elevation Rim Trace, MERT, that is used to detrend DEMs (see above). If the ME occurs at the last point on the profile, it is not recorded for that profile, to avoid selecting a rim location on a local rise or ridge.

The second method identifies Slope Break interest points (SB). The change in slope is found by fitting linear regression lines, of length λ_{SB}, to the profile before and after each point and calculating the difference in slope at the point. The slope calculations start and end one λ_{SB} from the ends of the profile. Using discrete linear fits avoids fitting a complex function to the entire profile and finding points of inflection, which would smooth the crater profile and slow the computation speed immensely. Once the slope change across every point is found, points with a difference of slope greater than $\Delta\theta$ (moderate downward concavity) are set aside. If these points are in a continuous segment over length χ, the point with the greatest slope change in each segment is selected as an SB point. By requiring a continuous region, we eliminate false positives due to noise, and the over

selection of SB points. Having a minimum grade change removes false positives from slight, near horizontal variations in background topography.

The third method is identifying Local Maxima (LM) along a profile. LM interest points are required to be higher than all elevations within a distance λ_{LM} along the profile. The λ_{LM} interval is used to avoid sampling noise in the profile, which has a shorter spatial scale. These three types of interest points are used together to assemble a superior "composite rim trace"(CRT).

Table 1: List of the rim extraction parameters and their optimal values. SB = Slope Break interest point; LM = Local Maximum interest point; CRT = Composite Rim Trace

Symbol	Parameter	Optimal Value
P_R	Profile length	$2R$
N	Number of radial profiles	512
λ_{SB}	SB linear regression fit length	$0.1R$
$\Delta\theta$	SB minimum slope change	0.05
χ	SB min continuous segment length	$0.05R$
λ_{LM}	LM length	$0.1R$
D_{rad}	CRT max radial discontinuity	$0.1R$
D_{int}	CRT max radial interest point distance	$0.05R$
D_{ang}	CRT max angular discontinuity	2°
T	Total number of traces calculated	17
λ_{mx}	Regression length for max-Slope	$0.1R$
λ_{rr}	Length of profile used to measure rim roundness	$0.25R$

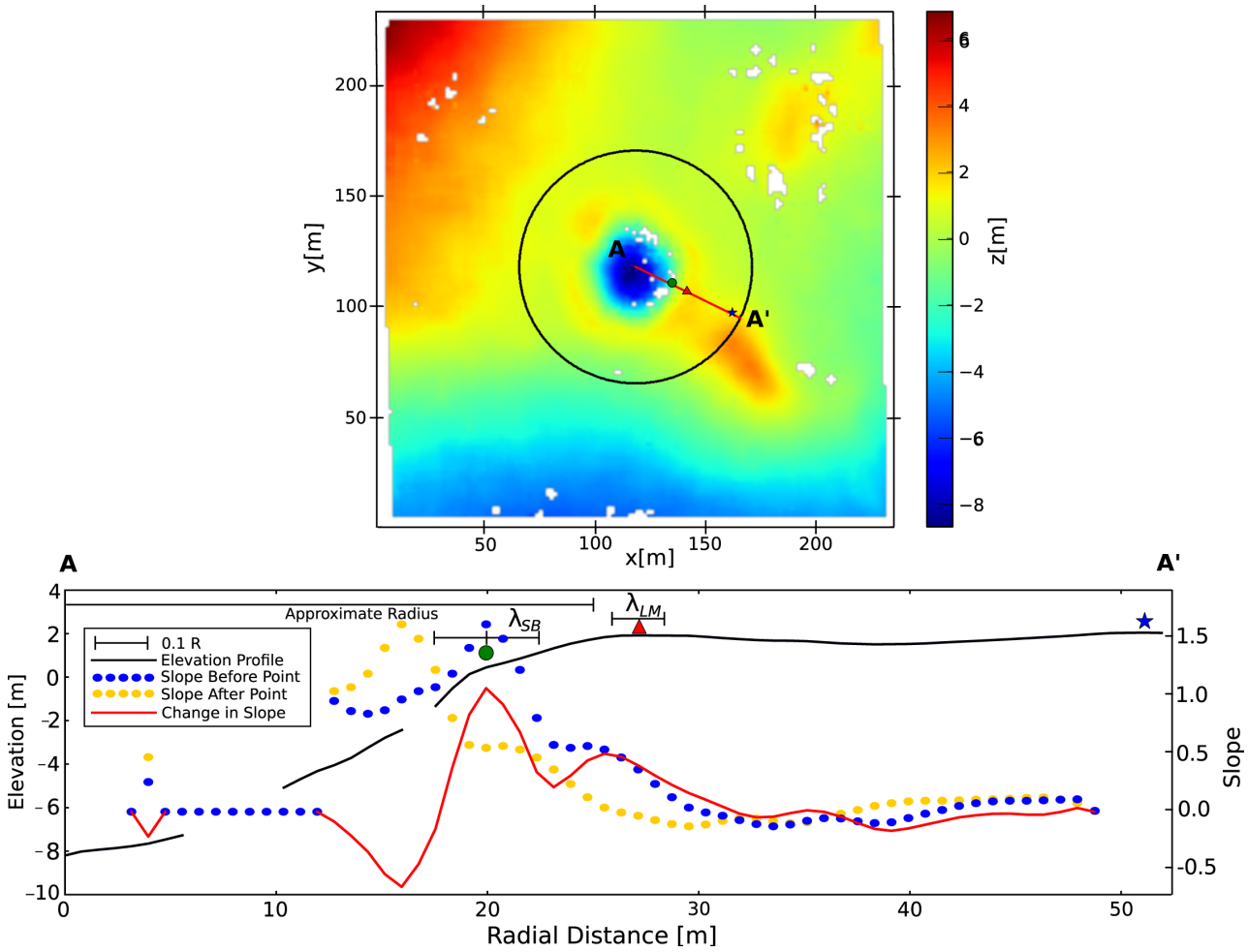

Figure 5: Example of profile interest point selection. Top Panel: DEM of crater 152, with a circle of radius P_R (the length of a profile) plotted in black. A sample profile, 474, is shown in red with the corresponding interest points: Star = Maximum Elevation, Triangle = Local Maximum, Circle = Slope Break. Bottom Panel: The same interest points plotted on elevation profile 474 shown in black with NaN portions missing. The slope of a linear regression fit to the profile over a segment of length λ_{SB} before and after each data point is shown in blue and yellow respectively. The discrete change in slope is plotted in red. Missing data in slope is due to high levels of NaN in the linear regression in the first missing region; the second NaN region was small enough that the slope was still able to be calculated. Note: The primary axis displays elevation, and the secondary axis is slope.

4.2.3 Composite Rim Trace

The composite rim trace (CRT) selection is based on minimizing radial discontinuities between consecutive points along the rim. The method starts on a ME interest point; it then checks the radial distance to the ME of the next profile, and if it is within the maximum allowed radial discontinuity (D_{rad}), it is chosen. If not, all interest points on the next profile within distance D_{int} are identified and one interest point is randomly chosen as the rim location on that profile. If there are no interest points within D_{int}, the profile is skipped, and the next profile is considered. To avoid unnecessarily large gaps in the rim and dead ends, a maximum angular discontinuity (D_{ang}) is enforced. If D_{ang} is exceeded, the program returns to the first skipped profile and assigns the rim location to a ME with discontinuity greater than D_{rad}.

This cycle is repeated for every profile, until a rim point that has been selected previously is selected again. The completed rim is stored and the entire rim selection process begins again. T rims for each crater are calculated in total, the first being the MERT. The remaining $T-1$ traces are CRT; created by advancing both clockwise and counter-clockwise through the profiles. Each CRT starts from one of eight positions: 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°, corresponding to the standard unit circle. Redundant traces are discarded.

The best rim is selected based on two attributes: the average of the five largest radial discontinuities normalized by crater radius (γ) and the fraction of points deviating from the MERT (Ω). The rim trace with the lowest value of $\gamma + \frac{1}{2}\Omega$ is considered the best trace, whose coordinates are stored and used in further analysis.

4.3 Crater and Target Attribute Selection

The target properties of each crater were coarsely determined with geologic maps and papers refering to HiRISE images in each vicinity. The target was further characterized by searching the original HiRISE image for a list of attributes. We saved a list of associated attributes for each crater. The aim of categorizing these attributes was to understand interactions between the geology or surface processes in an area and crater morphology, such as target strength or weathering and erosion. The chosen attributes are listed in Table 2, crater examples are shown in Figure 6, and

context examples are shown in Figures 7 and 8. These attributes can be used to test previous hypotheses regarding the influence of jointing or faulting on crater morphology and formation (Watters et al. 2011, Poelchau et al. 2009, Öhman et al. 2008, Eppler et al. 1983).

We designated the crater's geologic setting using a combination of geologic maps and papers, the MOLA elevation map, and visual characteristics. The lowland craters are situated above 20° North latitude and below -3km in elevation, values that approximate the extent of the Vastitas Borealis formation (Tanaka et al. 2005). Based on the findings of Byrne et al 2009, we assigned all lowland craters ($z < -3$km) above 45° North latitude as forming in ice-rich layers. We separated ice-rich Vastitas Borealis craters from the global distribution to investigate the effect of ice in crater formation, to see if melting ice would influence morphology. Noting other geologic regimes, such as volcanism, can be used to test hypotheses about increased crater depth and volume with target strength, as used by Stewart and Valiant (2006).

Table 2: Attributes of craters and target material

Symbol	Attribute Description	Category
F	Fluidized, Lobate Ejecta	Crater Attribute
R	Rays in Ejecta	Crater Attribute
S	Step, Bench in Wall	Crater Attribute
L	Layering in Wall	Crater Attribute
E	Formed in Preexisting Ejecta	Target Material
e	Edges, Scarps, Jointing	Target Material
f	Fractures, Fissures	Target Material
h	Heavily Cratered	Target Material
l	Layering	Target Material
p	Patterned Ground	Target Material
r	Rubbly, Boulders, Rocks	Target Material
u	Uniform, Uncratered, Flat	Target Material
v	Filled Valleys, Flooded Topography	Target Material
0	Lowlands with Surface Ice	Target Geology
1	Northern Lowlands	Target Geology
2	Volcanics	Target Geology
3	Sedimentary Rocks	Target Geology
4	Ice Rich Soils	Target Geology
5	Lava Plains	Target Geology
6	Debris	Target Geology
@	Amazonian	Target Epoch
$	Hesperian	Target Epoch
^	Noachian	Target Epoch
−	Early	Period
=	Middle	Period
+	Late	Period

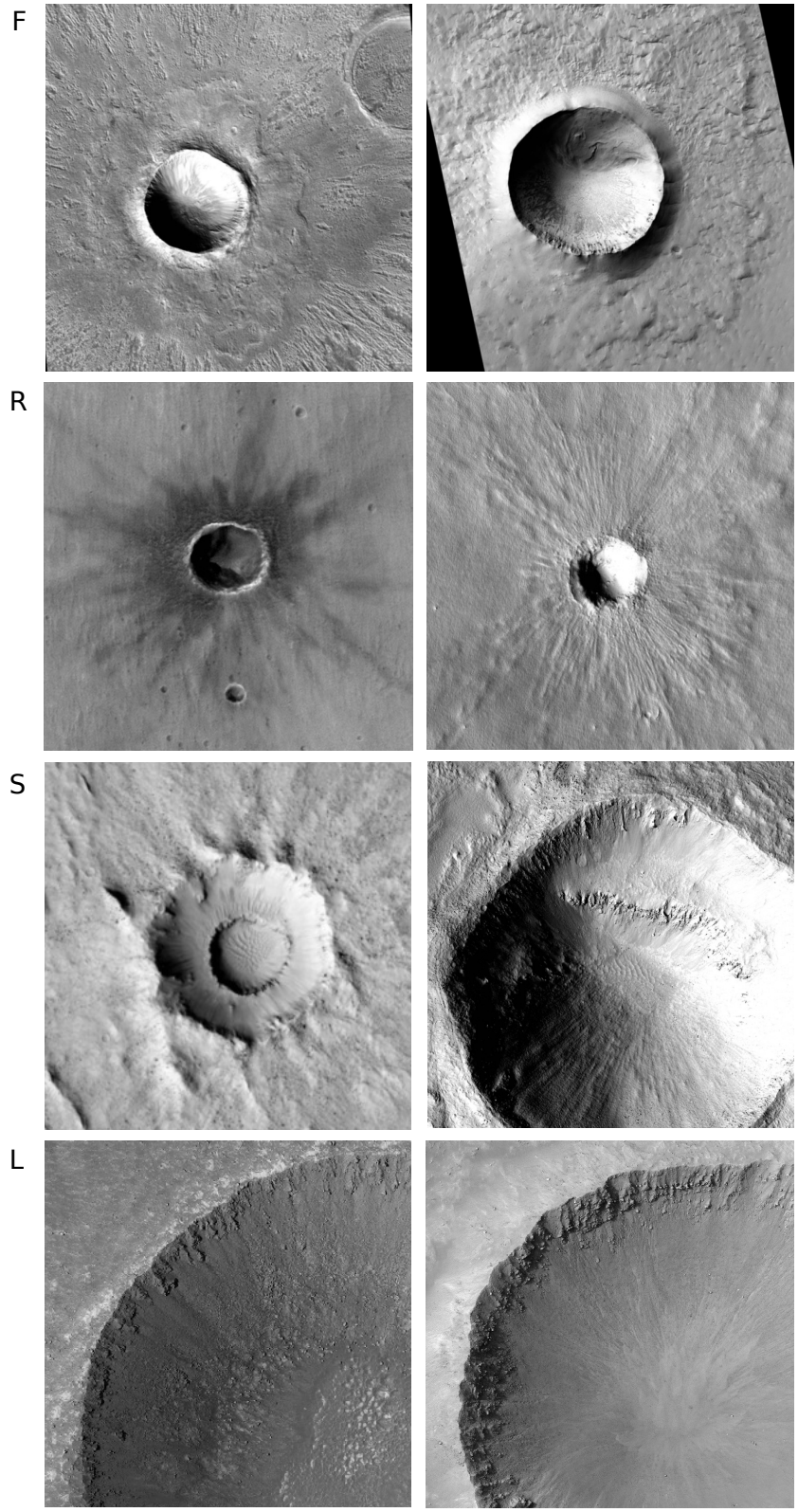

Figure 6: Examples of crater attributes. F = Fluidized, lobate ejecta; R = Rayed ejecta; S = Step or bench in cavity wall; L = Layering in crater cavity.

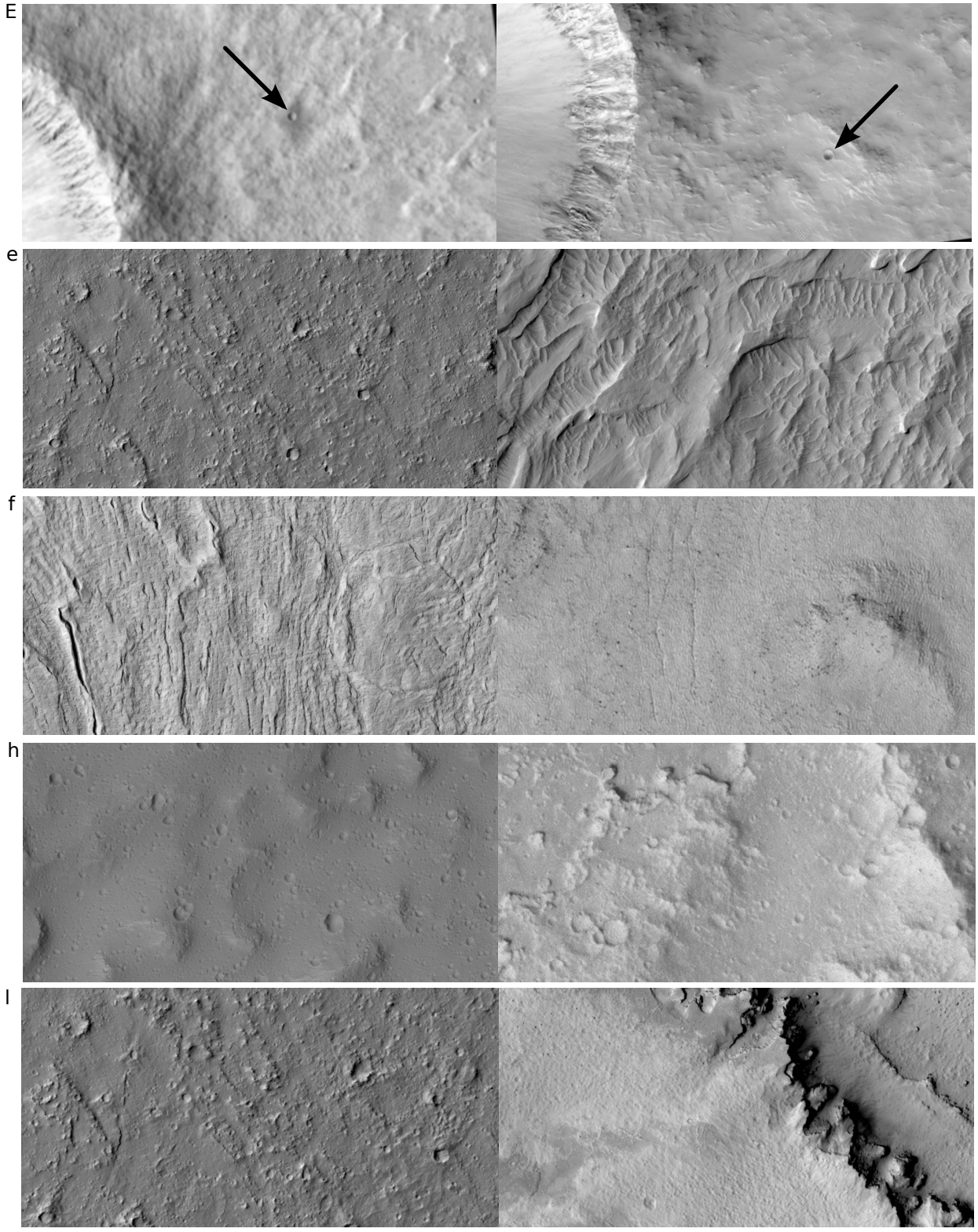

Figure 7: Attributes of target material. E = formed in preexisting Ejecta; e = Edges, scarps or joint controlled mass wasting; f = Fractures or Fissures; h = Heavily cratered terrain; l = visible Layering.

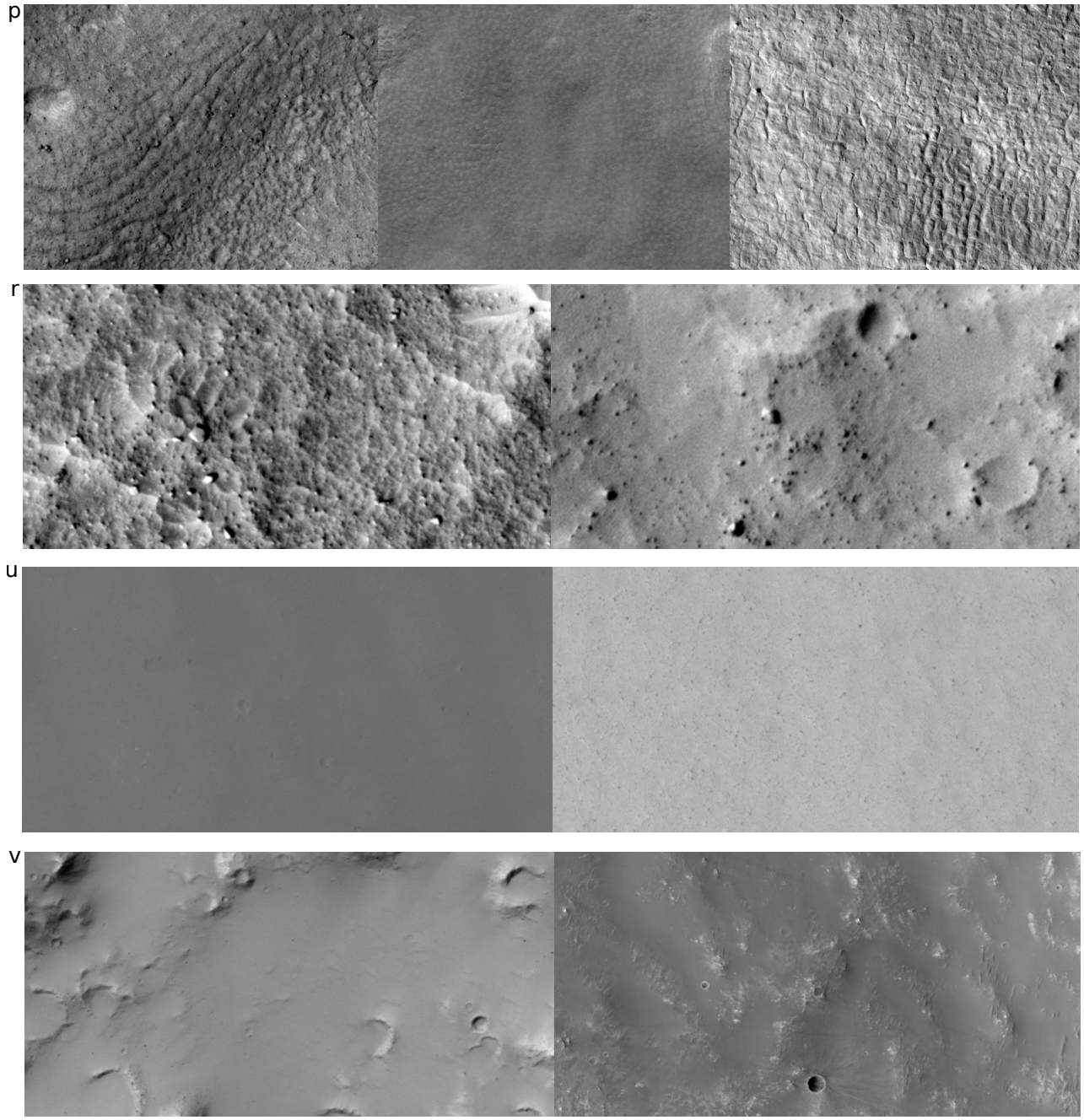

Figure 8: Attribute of target, Continued. p = Patterned ground; r = Rubbly, boulders eroding out of the terrain; u = Uniform and Uncratered; v = filled Valleys or flooded topography, Smooth and featureless low areas with rough highs.

5 Analysis

The crater rims were successfully found for 163 of the 200 craters, with $d/D_{max} > 0.15$ (fresh to moderately modified). 98 of the 200 craters have depth to diameter ratios greater than 0.2, and are considered fresh. The 163 successful craters were used in for our global analysis.

5.1 Planform Metrics

We computed crater rim morphometry metrics to be used in statistical comparison studies; all metrics are listed in Table 3 at the end of this section. The extracted 3-D rim trace was used to calculate many simple metrics: mean crater radius and diameter, range of rim elevation (rim relief), standard deviation in rim height (rim raggedness), radial standard deviation normalized by mean radius (deviation from a perfect circle), and maximum radial deviation.

The nadir, lowest point on the crater floor, was chosen from one of three DEMs: the original small DEM, the small IP-DEM, or the large DEM. The island purged small DEM was preferentially chosen because unprocessed DEMs could contain erroneous data (see Section 4.1), and large DEMs were subject to lower sampling resolution to decrease the stereo-correlation time for large images. In some cases, DEM post processing would remove the crater floor entirely, if there was a ring of NaN near the rim. In 12 of the 199 craters, the original, unprocessed DEM was used to extract the nadir. In two cases, the ASP stereo-correlator failed to find the bottom of the crater in the small DEM but succeeded in correlating the crater floor in the large DEM. Perhaps this is due to the lower resolution in the large DEM, where fewer pixels represent a larger region to correlate and disparities will not be as large. In these cases, the large DEMs were used to extract nadir values. The depth-diameter ratio was calculated from the corresponding nadir value and the best trace's maximum and average diameter.

The circularity ratio was also calculated for each rim. Circularity is a measurement of how circular a shape is. The circularity ratio is defined as $\frac{4\pi A}{C^2}$, where C is circumference and A is area. The ratio will be 1 if and only if the shape is a perfect circle. The further the shape deviates from circular, the lower the ratio becomes. It would be physically impossible for the ratio to reach zero, unless it describes a line.

To better describe polygonal crater shapes, we computed a Fourier transform of crater planform for the first six harmonics. The degree of a harmonic is equivalent to the number of perturbations from circular; for example the third harmonic is triangular. Harmonic amplitudes were calculated by the SciPy Fourier transform method. The harmonic with the highest amplitude is considered the primary harmonic. The primary harmonic is used to describe the overall planform shape. The three most important harmonics were recorded for each crater.

We test the hypothesis, set forth by Poelchau et al. (2009), that rim uplift increases in crater corners, by comparing rim height and radius. To do this, we use the Pearson correlation coefficient, found in SciPy's statistics module. A linear correlation will yield a coefficient of $\pm$ 1, dependent on whether the correlation is positive or negative. A coefficient of 0 indicates no correlation.

Additional shape metrics were also calculated to observe how trends were constrained by geologic setting. Standard deviation of rim height, rim relief, and average rim height were normalized by both crater diameter and crater depth to create six standardized metrics for comparison between craters.

5.2 Radial Profile Analysis

Power law equations can be used to describe a characteristic radial elevation profile for both the crater cavity and the crater flanks. Simple crater cavity profiles are generally thought to be parabolic (Melosh 1989), but we aim to investigate the potential spread in power law exponents. We calculate the power law from approximately 512 cavity cross-sections at different azimuths using the following equation: $z^* = B_C(\frac{r}{R})^{\alpha_c}$, where r is distance from center, R is the crater radius, z^* is height normalized by R, B_C and α_C are quantities we calculate from the profiles; the power-law coefficient and exponent, respectively. We extracted profiles beginning at the geometric center of the crater and ending at each rim coordinate. The lowest elevation of the profile is subtracted from the entire profile, to make everything positive in elevation, to avoid taking the logarithm of negative numbers. Then, the profiles are normalized by crater radius, in both length and height (Figure 9 top). At this point, if a profile contains more than 50% NaN, (blank data points), that profile is excluded from the rest of the analysis.

Next, a logarithmic profile is calculated, starting one point after the minimum elevation, to avoid taking the log of 0. A linear regression is fit to the individual logarithmic profile for each azimuth. Then, all fits are averaged to calculate the power-law exponent and coefficient for the entire cavity (Figure 9 bottom). Uncertainty is calculated by standard deviation. In addition, we calculate the maximum slope, in degrees, of the cavity wall, by finding the slope of every point on the profile over a segment of length λ_{mx}, then averaging the maximum slope, ϕ, found in every profile. The overall maximum slope, ϕ_{max}, of the crater cavity is also found. These are used as proxies for the angle of repose.

The power law describing the cavity shape was used to reconstruct the excavated volume of the crater. Stewart and Valiant (2006) conducted measurements of crater cavity volume as a function of crater diameter in many regions on Mars. Their data show different power-law trends for each region, with exponents ranging from 2.20 to 2.64. We investigate how the global trend compared to Stewart's results. We computed cavity volume as the integral of the cavity equation, with respect to height, from nadir to ambient elevation:

$$r^* = \frac{r}{R} = (B_C \frac{h}{R})^{\frac{1}{\alpha_C}} \tag{2}$$

$$V = \pi \int_0^H r^* (h^*)^2 dh^* \tag{3}$$

$$V = \pi \frac{\bar{R}^{2-2/\alpha_C} h^{2/\alpha_C+1}}{(\frac{2}{\alpha_C}+1) B_C^{2/\alpha_C}} \tag{4}$$

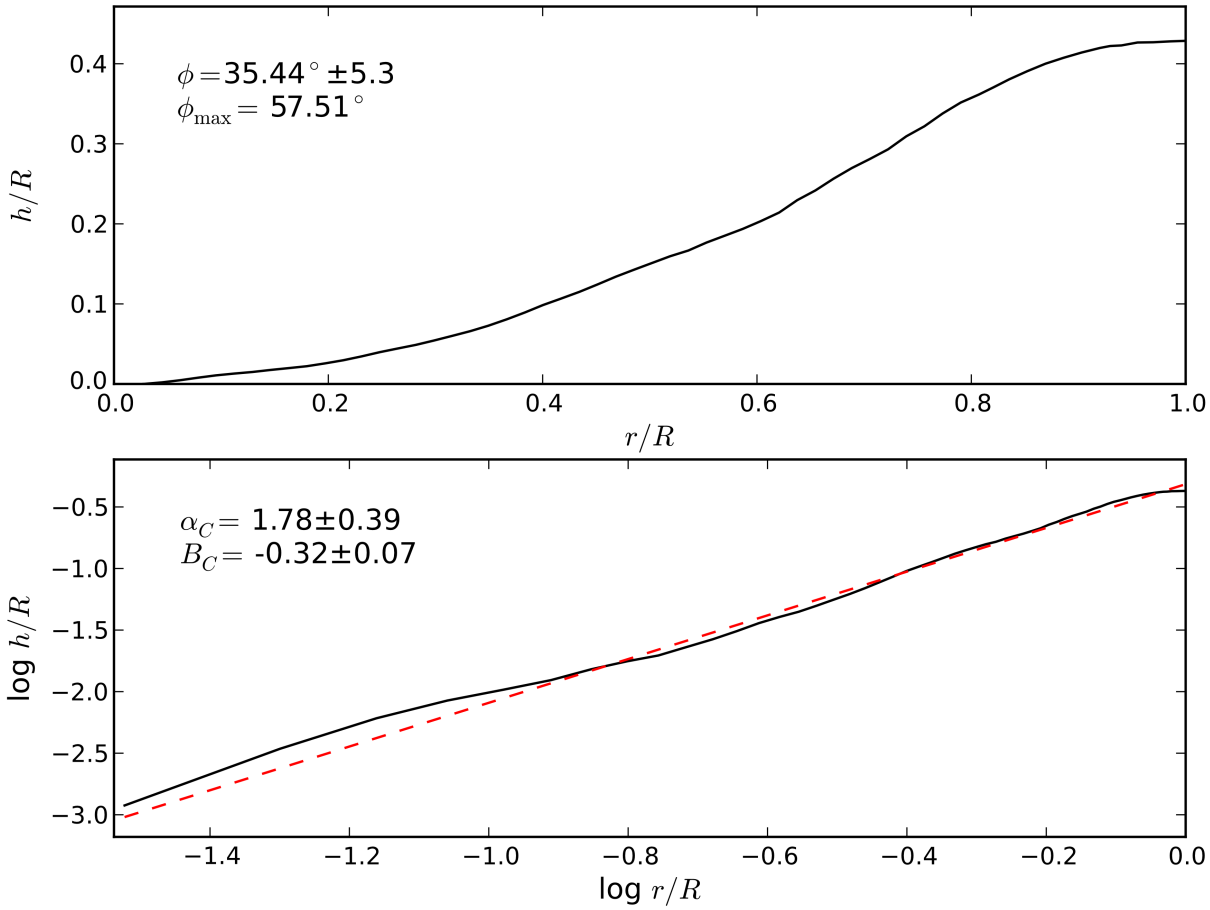

Figure 9: A graphic illustrating the average cavity profiles for Crater-004. The top panel shows the linear profile and the average maximum slope of the crater wall (ϕ) and the absolute maximum slope of the enitre cavity (ϕ_{max}) in degrees. The bottom panel shows a logarithmic scale with the power law fit averaged for all profiles plotted in red. The values α_C and B_C are the fit line slope and intercept, respectively.

The ejecta thickness at a given radius has already been constrained (see Section 3.2.2), but flank elevation also includes a target uplift component, so we calculate a power law decay to understand these effects. The average flank decay equation was calculated in a similar fashion to the cavity power law equation. Radial profiles are extracted beginning at each rim point and ending at three crater radii. The lowest point is subtracted from the profile to ensure we are working with only positive values, before we calculate the logarithm. Next, the program crops the profiles to end at the lowest elevation, in order to avoid sampling background topography, such as a hill or rise. Then, the profiles are normalized by crater radius. To ensure our trend line is fit to the shape crater flank only and not an abrupt distal toe or rampart, a second crop is necessary. According to a preliminary survey, any transition between flank and toe occurs below an elevation of 1% crater radius, so the profiles are cropped again, removing elevations less than $0.01R$. Figure 10 shows the average flank profile for a sample crater.

At this stage, any profile containing less than 0.25R worth of usable data is excluded from analysis. A logarithmic fit is calculated for each remaining flank profile. The fits are averaged to find the exponent (α_F) and coefficient(B_F) corresponding to the elevation equation $z^* = B_F(\frac{r}{R})^{\alpha_F}$. Using the flank decay exponent, we calculated the ejecta run-out length. Run-out length is defined as the radial distance at which elevation drops to 10% of the rim height.

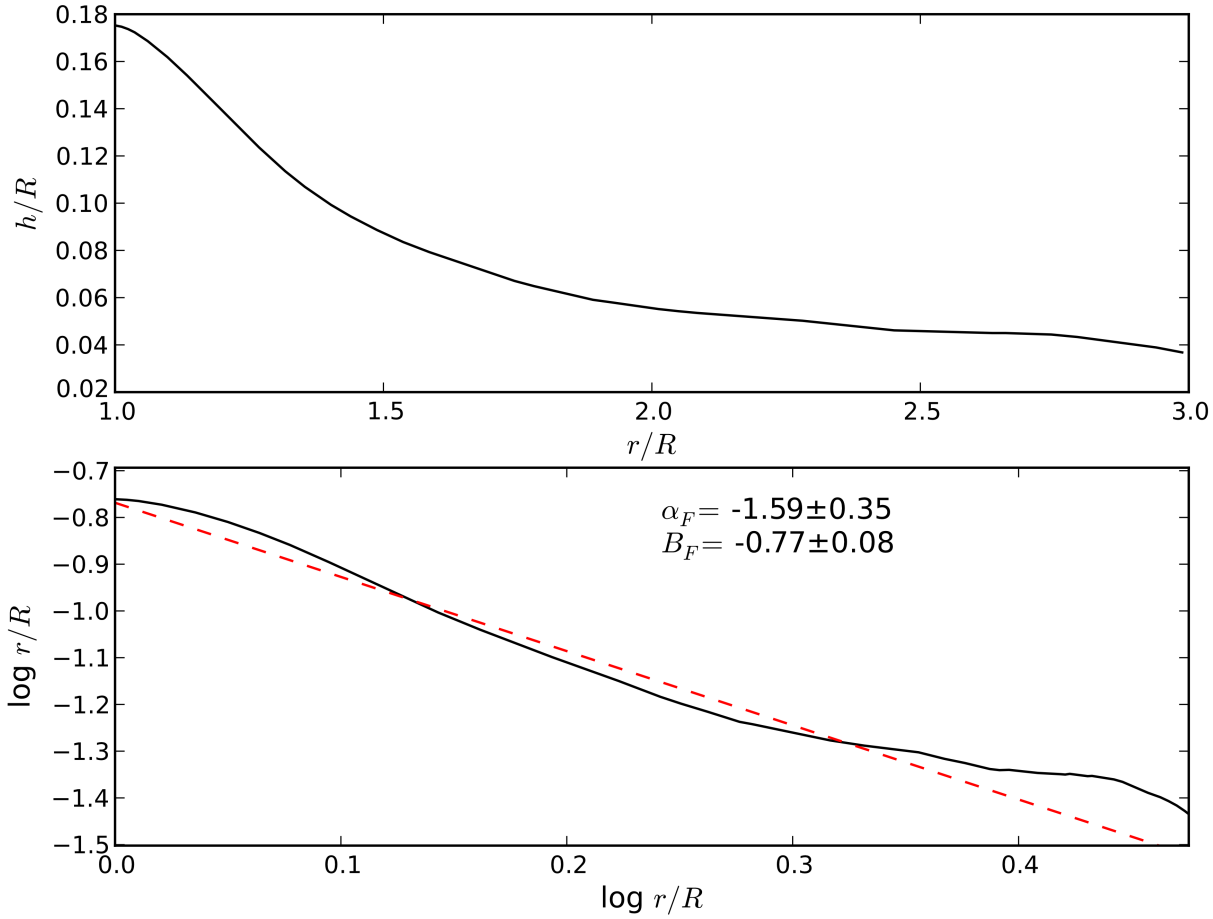

Figure 10: A graphic showing the average flank profile for Crater-004. Linear profiles are carried out to an elevation of $0.01R$ (top). The average flank decay fit is shown in red against the logarithmic profile (bottom). The values α_F and B_F are the fit line slope and intercept, respectively.

5.3 Rim Roundness

Quantitative rim roundness metrics have not been widely adopted in the field. We propose a metric to characterize the length scale at which the slope changes substantially, going away from the rim. If the crater rim is more rounded, then the slope of the rim will change slowly, and the length scale will be longer. We define the rim roundness metric as the length at which the slope reaches fifty percent of the maximum slope, both cavity-ward and flank-ward, ℓ_C and ℓ_F respectively.

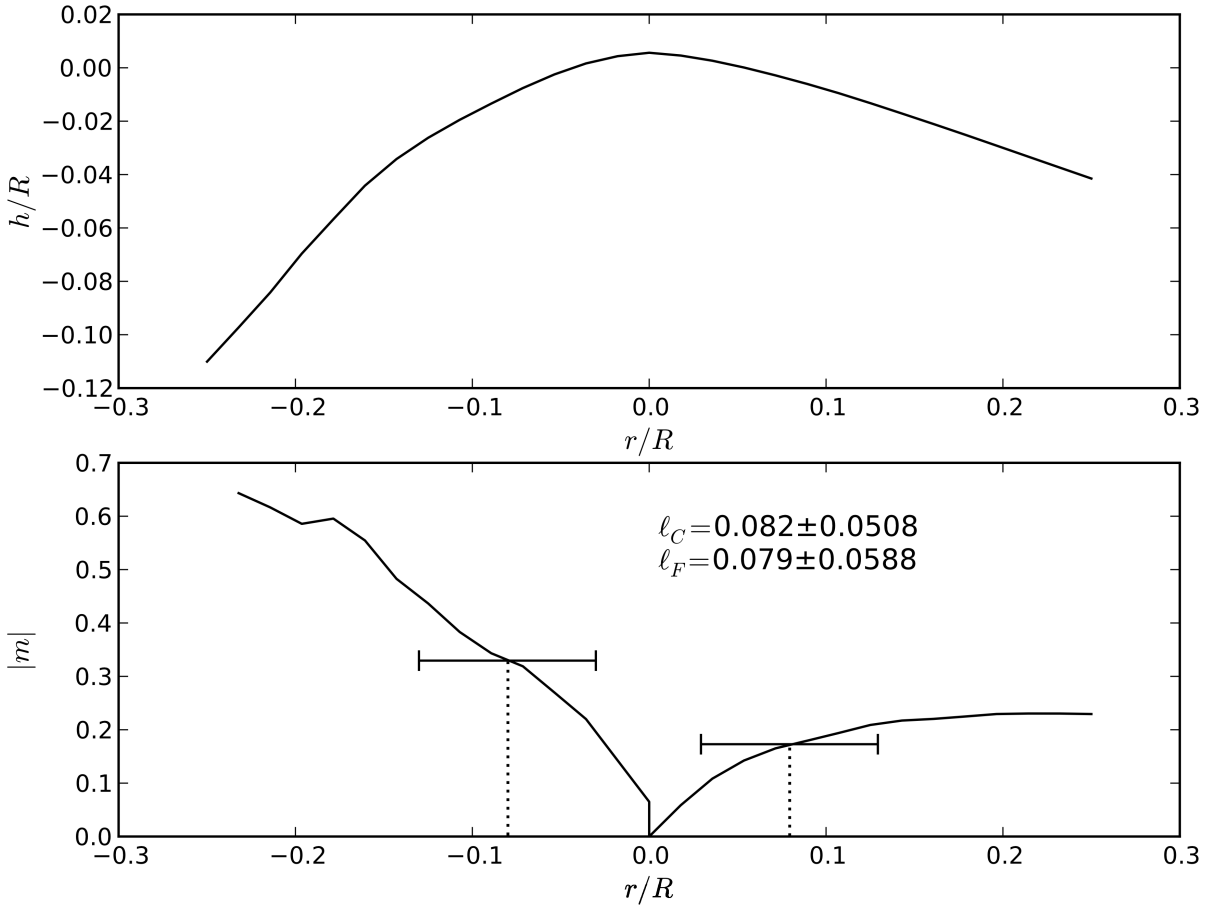

Figure 11: The average rim profiles are carried $0.25R$ cavity and flank-ward; shown for Crater-004 (top). The bottom panel shows the average, absolute slope values of the profile. The rim roundness metric is the distance at which the rim profile slope passes 50% maximum slope on each side of the rim, and is shown with the dotted line with standard deviation.

These values are calculated by taking profiles from each rim point spanning λ_{rr}, inward and outward. The metric is calculated separately for the cavity and flank profiles, as some craters appeared to have one side rounder than the other. The absolute value of the flank profile is taken for easier coding. Because the profiles are a discrete distribution, the roundness length scale for each profile is defined as the first point from the rim that surpasses 50% the maximum slope change (Figure 11). The crater rim roundness metrics ℓ_F and ℓ_C are calculated by averaging all profiles with $< 50\%$ NaN. The cavity-ward and flank-ward rim roundness are added to approximate an overall metric, $\ell = \ell_C + \ell_F$.

We compiled our vast list of crater morphometry metrics into a single database which was passed to a python script for statistical analyses. The results of the analyses are used to test the many hypotheses presented in the study.

Table 3: List of the morphometry metrics that were extracted. The extraction steps are: Digital Elevation Model Synthesis (DEMS); Planform Extraction (PLEX); Profile Extraction (PREX); and Additional Analysis (AANL).

Symbol	Metric Definition	Extraction Step	Error
δ_{DEM}	Small DEM Resolution	DEMS	None
D	Average Diameter	PLEX	δ_{DEM}
ϵ	Maximum Radial Discontinuity in CRT	PLEX	None
Ω	Fraction of CRT Differing from MERT	PLEX	None
Γ	Average of Top Five Radial Discontinuities	PLEX	None
σ_h^*	Rim Raggedness / Rim Height Std Dev	PLEX	None
σ_R/D	Radial Std Dev / Rim Asymmetry	PLEX	None
Δ_R/D	Maximum Radial Deviation	PLEX	None
r_{Rh}	Rim Height-to-Radius Correlation	PLEX	None
C_R	Circularity Ratio	PLEX	None
$d_{\max}/D$	Maximum Depth / Diameter	PLEX	None
d/D	Average Depth / Diameter	PLEX	None
L_0	Primary Harmonic	PLEX	None
L_1	Secondary Harmonic	PLEX	None
A_N^*	N^{th} Harmonic Amplitude	PLEX	None
θ_2	Orientation of Second Harmonic	PLEX	None
α_C	Cavity Power-Law Exponent	PREX	Std Dev
B_C	Cavity Power-Law Coefficient	PREX	Std Dev
α_F	Flank Decay Exponent	PREX	Std Dev
B_F	Flank Decay Coefficient	PREX	Std Dev
ℓ_C	Rim Roundness Metric Cavity-ward	PREX	Std Dev
ℓ_F	Rim Boundedness Metric Flank-ward	PREX	Std Dev
ℓ	Overall Rim Boundedness Metric	PREX	Std Dev
ϕ	Average Maximum Cavity Wall Slope	PREX	Std Dev
$\phi_{\max}$	Maximum Cavity Wall Slope	PREX	None
V_C	Reconstructed Cavity Volume	PREX	None
r_{ej}	Ejecta Runout Length	PREX	None
σ_h/d	StdDev of Rim Height over Depth	AANL	None
σ_h/D	StdDev of Rim Height over Diameter	AANL	None
$\Delta h/d$	Rim Relief over Depth	AANL	None
$\Delta h/D$	Rim Relief over Diameter	AANL	None
$\bar{h}/d$	Average Rim Height over Depth	AANL	None
$\bar{h}/D$	Average Rim Height over Diameter	AANL	None

6 Results

The values of all metrics calculated in this study are shown in tables in the appendix.

6.1 Comparing Distributions

The cumulative distribution function (CDF) is a way to present the distribution of a single variable. The graph shows the fraction of the data that lay below a given value. It is a useful method for comparing distributions if the initial distribution shape is unknown. Though many of the distributions we compare look visibly distinct, we have not yet run analyses to test statistically significant differences between them. A Gaussian distribution will appear as a smooth stretched S-curve (Figure 12). The median value of any distribution will be the location at which the curve passes through 0.5 on the y-axis. A tighter distribution will have a steeper slope.

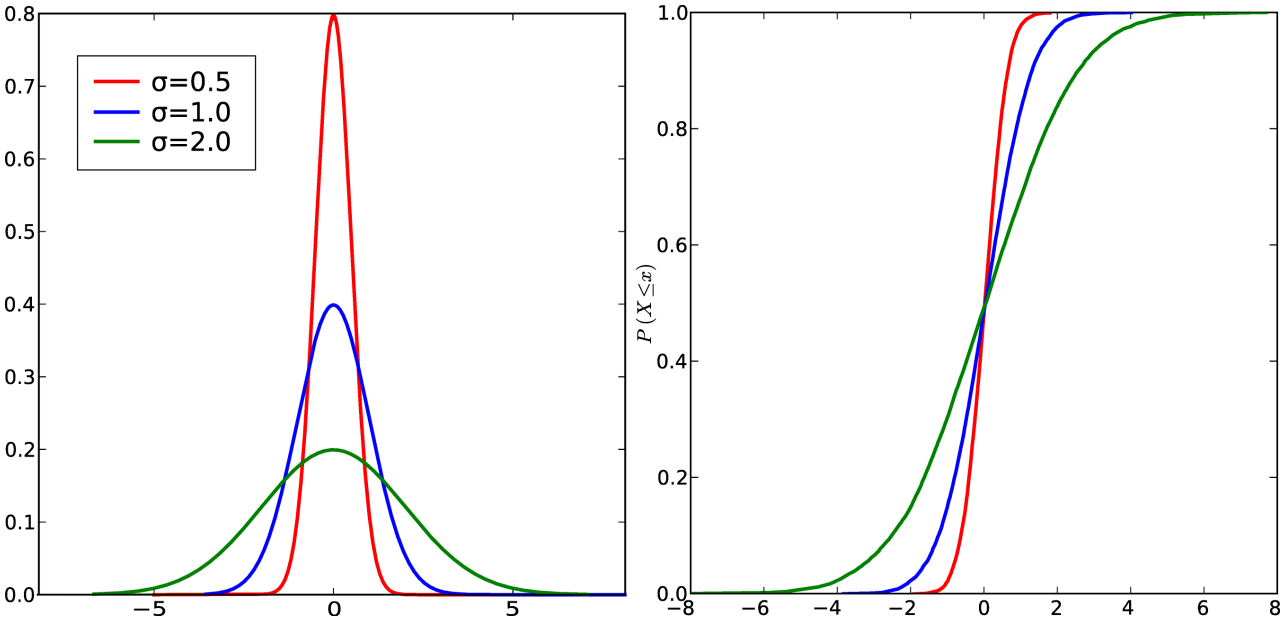

Figure 12: An example of the Cumulative Distribution Function for several Gaussian distributions. The Gaussian probabilities are shown on the left with the corresponding cumulative distribution function potted on the right in the same color. σ shown is one standard deviation.

6.1.1 Fresh vs Modified Craters

By using a CDF to represent distributions, we compare morphometric parameter frequencies between different crater groups. When we separate fresh, unmodified craters ($d_{\max}/D \geq 0.2$) from the rest of the data set, it becomes apparent that fresh craters have a much smoother distribution of flank-decay shapes than the modified craters. Modified craters appear to have smaller decay-exponents in general, an average close to -2 compared to -3 for fresh craters. The older craters also have several steps in the CDF, which corresponds to peaks in the distribution, the most notable at -3.75 (Figure 13).

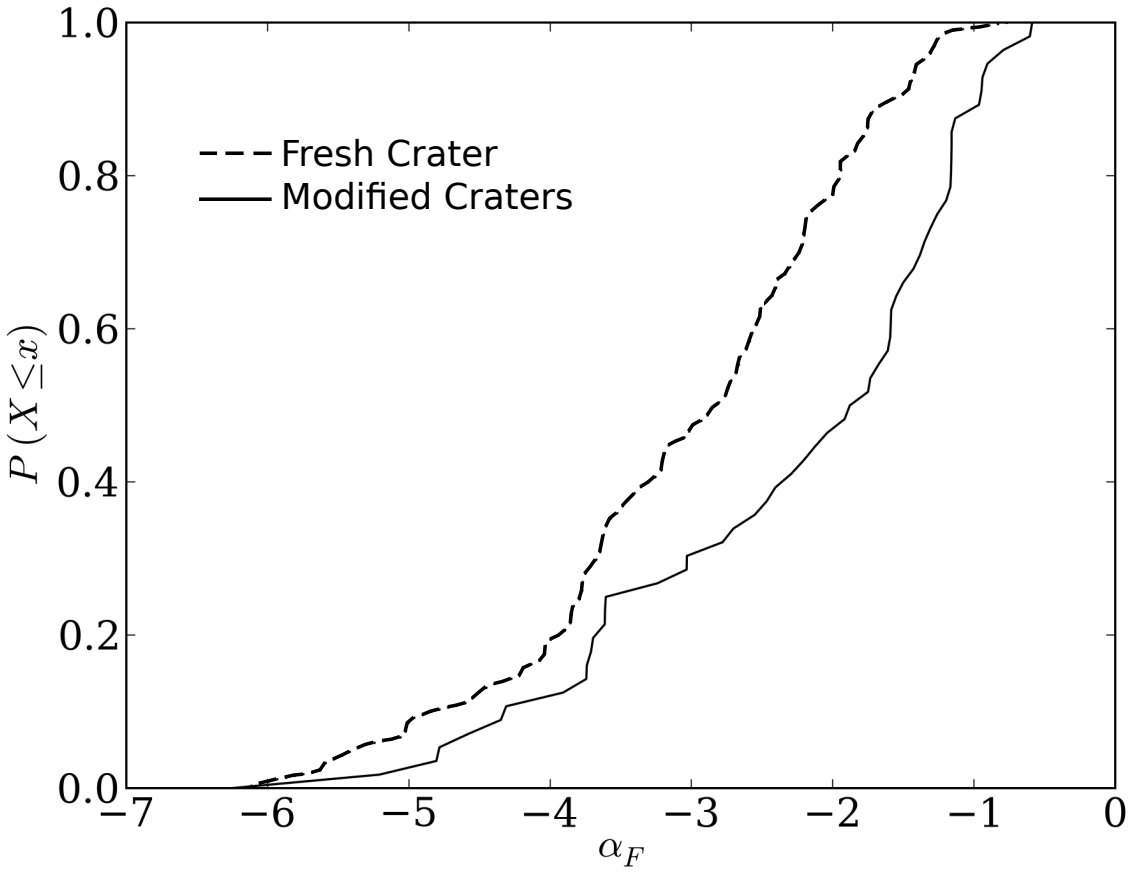

Figure 13: CDF plot of flank decay exponents. The dotted line is the distribution of fresh craters ($d/D \geq 0.2$), solid line is all other craters. Notice the slight step at -3.75, and the higher frequency of relaxed flanks in modified craters.

The crater cavity-decay shape distribution has much less variation between fresh and modified craters, though the fresh craters contain tails more reminiscent of a Gaussian distribution (Figure 14). Rim roundness also does not appear to depend on the age of the crater (Figure 15).

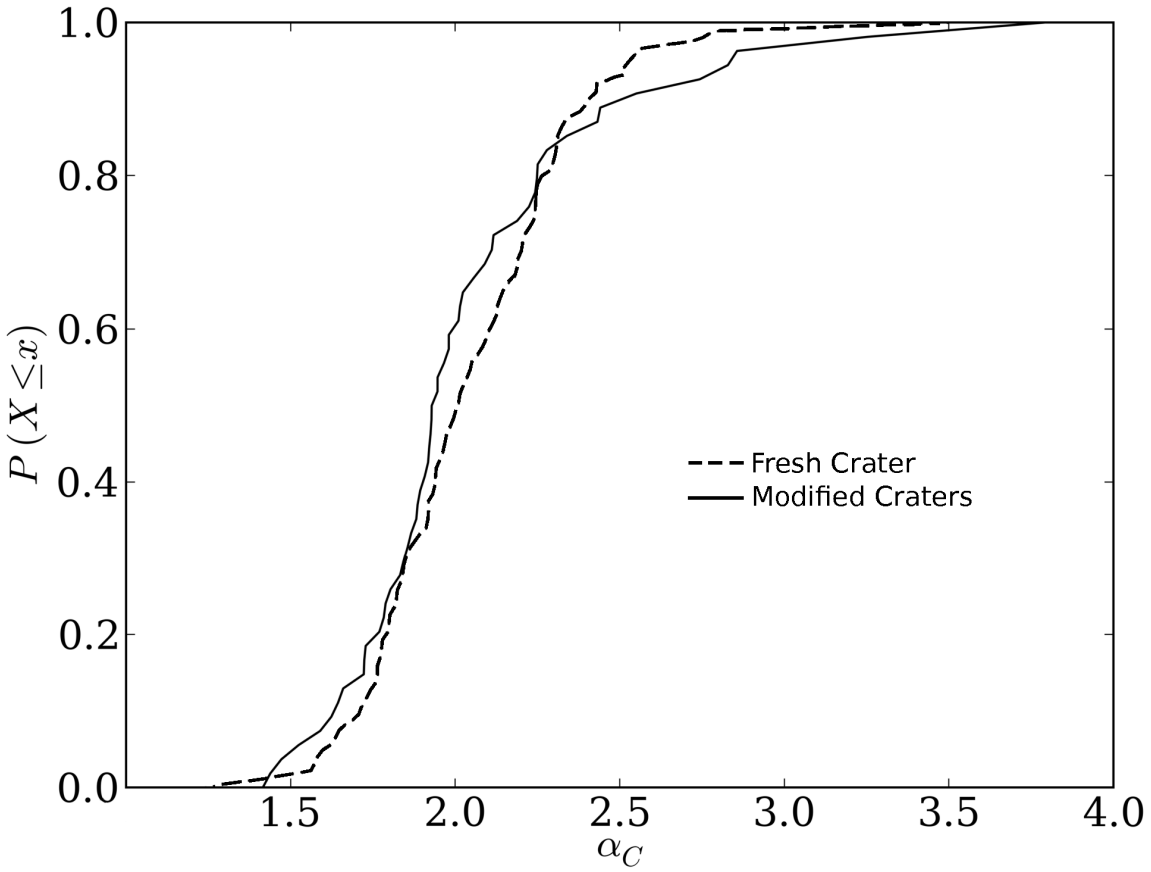

Figure 14: CDF of cavity profile decay shape. The average shape is nearly parabolic, and there is a lack of cone shaped craters ($\alpha_C = 1.0$). The distribution of fresh craters is shown with a dotted line. Note the similarity of fresh and modified craters.

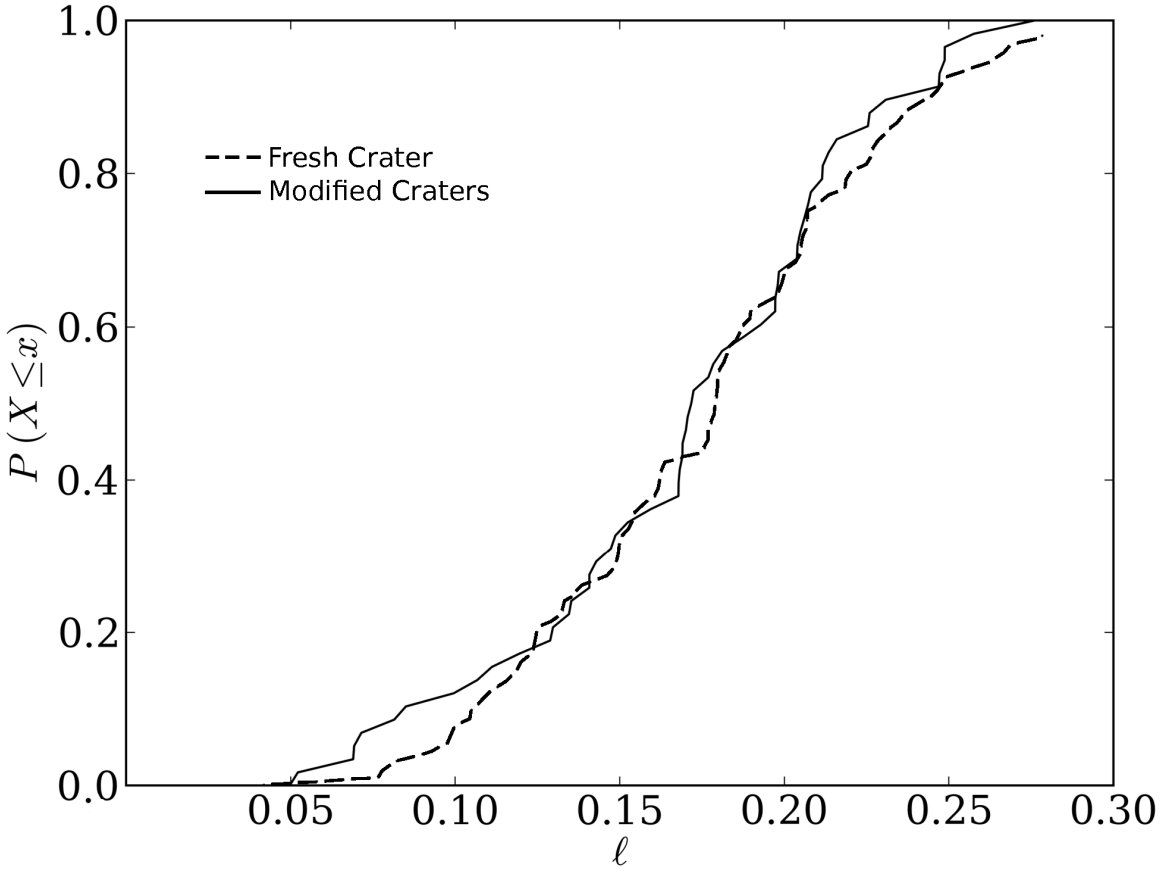

Figure 15: CDF of total rim roundness length-scale; larger numbers are rounder rims. The dotted line denotes fresh craters. Note there is a lack of modification in rim roundness.

6.1.2 Geographic Variation

We isolated the fresh, icy craters of the Vastitas Borealis lowlands from the global fresh crater distribution to compare crater morphology in different geologic regions. The lowland crater rim-roundness metric distribution does not appear to be statistically different from the rest of the craters (Figure 16). Icy crater cavity power-law shape distribution is also very similar to the global distribution, though there are slightly fewer craters with higher order exponents (Figure 17).

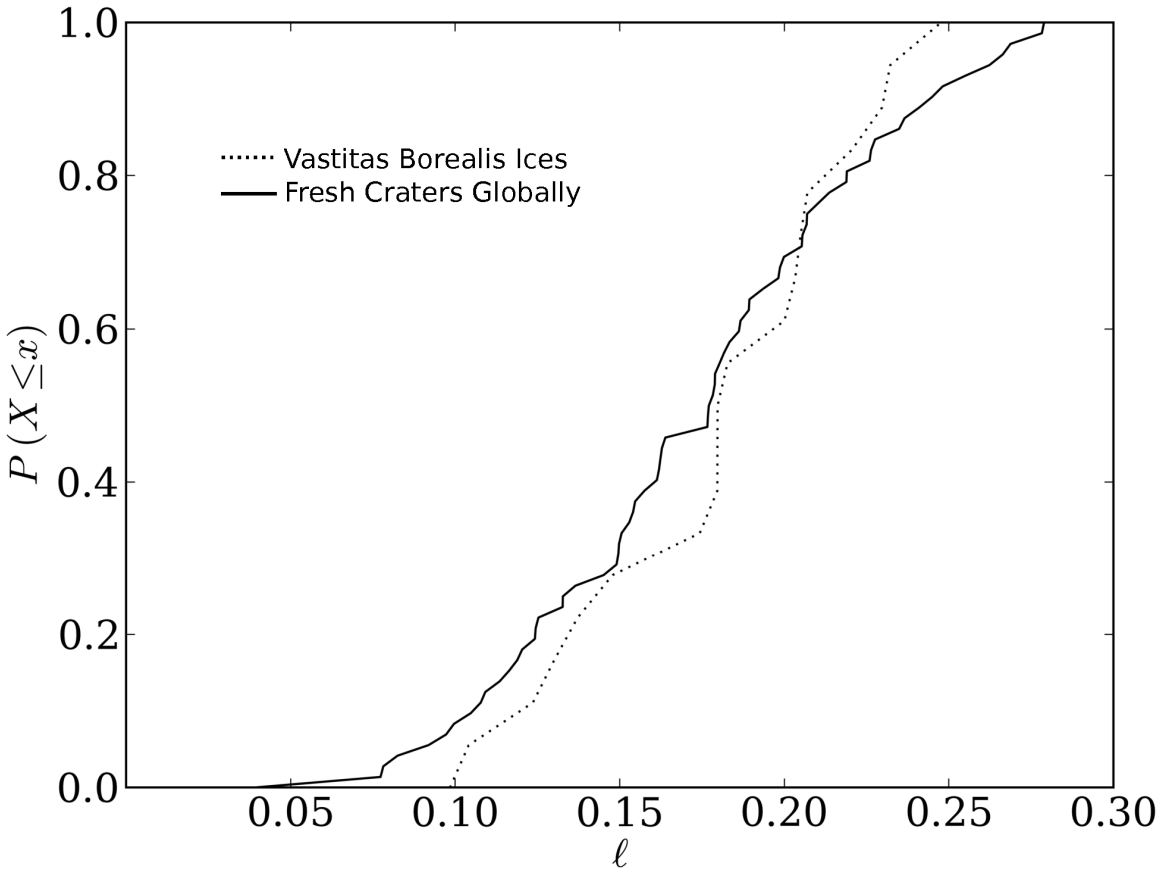

Figure 16: CDF of total rim roundness for the fresh craters; the dotted line shows the distribution in Vastitas Borealis craters and the solid line is all other craters.

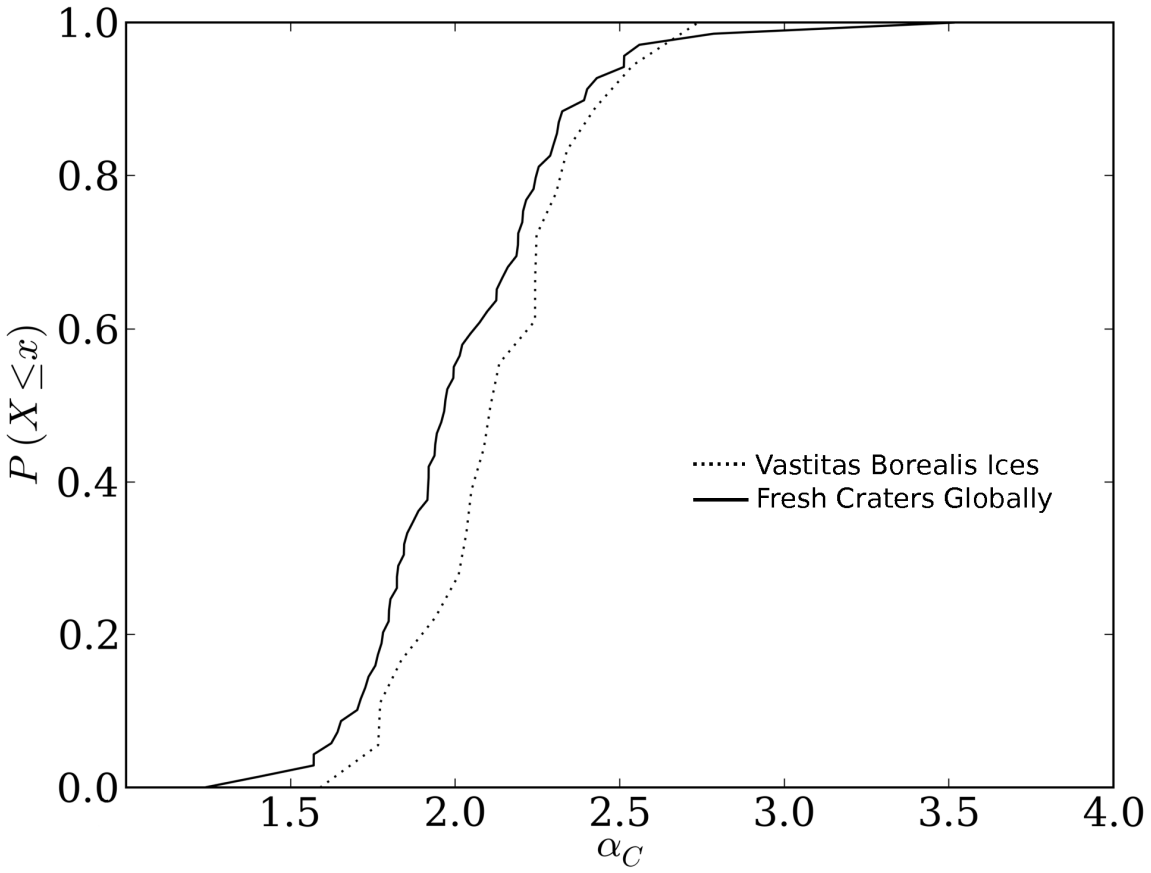

Figure 17: CDF of fresh crater cavity shape. The dotted line denotes icy, lowland craters. The average cavity shape is close to parabolic for both distributions.

The difference between Vastitas Borealis flank profile shape distribution and the global distribution is quite strong. In regions with shallow sub-surface ice, the flank decay only occurs at low orders. Globally, the steepest decay exponents reach up to -6, with an average just over -3. High-latitude ices, on the other hand, have a maximum decay exponent of -4 and average around -2.5 (Figure 18).

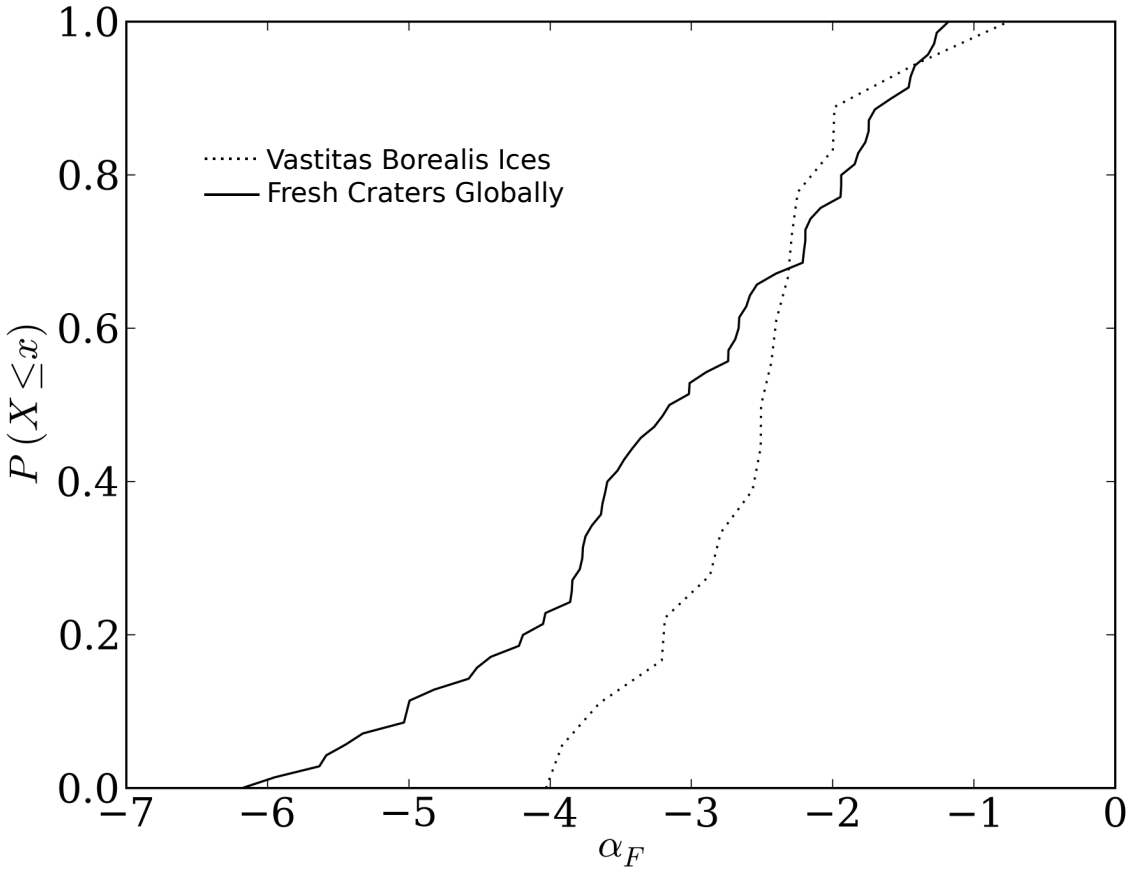

Figure 18: CDF of flank power law decay exponent for fresh craters only. The Vastitas Borealis distribution is shown with a dotted line. The fastest decay rates are absent in icy craters.

Comparing the cavity and flank decay rate, we see clustering around 2,-3 (Figure 19). Fresh craters have nearly parabolic cavities, with $\alpha_c = 1.93 \pm 0.55$, and flank decay exponents of $\alpha_F = -2.68 \pm 1.32$. The fresh lowland craters are plotted with black circles. The lack of high order flank decay seen in Figure 18 is evident in this plot as well.

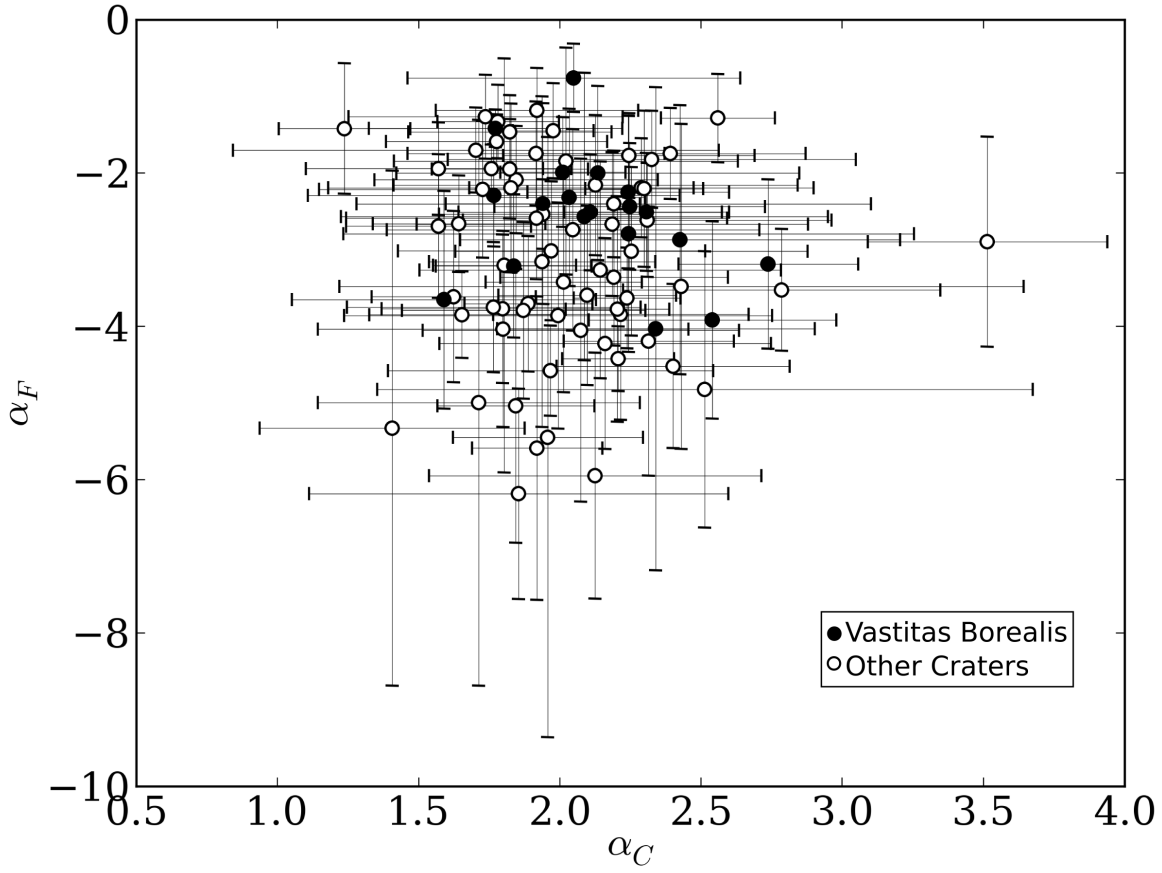

Figure 19: Crater profile flank decay shape versus cavity shape. Filled points represent Vastitas Borealis craters, which have more spread in cavity shape than flank shape.

6.2 Harmonics

The primary Fourier harmonic distribution describes the shape of the crater planform (the first harmonic is excluded because it describes the size of the crater diameter). Figure 20 shows the fraction of craters with a given primary harmonic for the global crater distribution. The distribution appears to fall off exponentially. Figure 21 shows the harmonic distributions of craters forming in Vastitas Borealis ices (29 craters), volcanics (69 craters) and sedimentary deposits (51 craters). The volcanics and ices have similar shape distributions, but the tertiary harmonic frequency is highly developed in sedimentary rocks, and nearly equivalent to the frequency of dominantly oval-shaped craters.

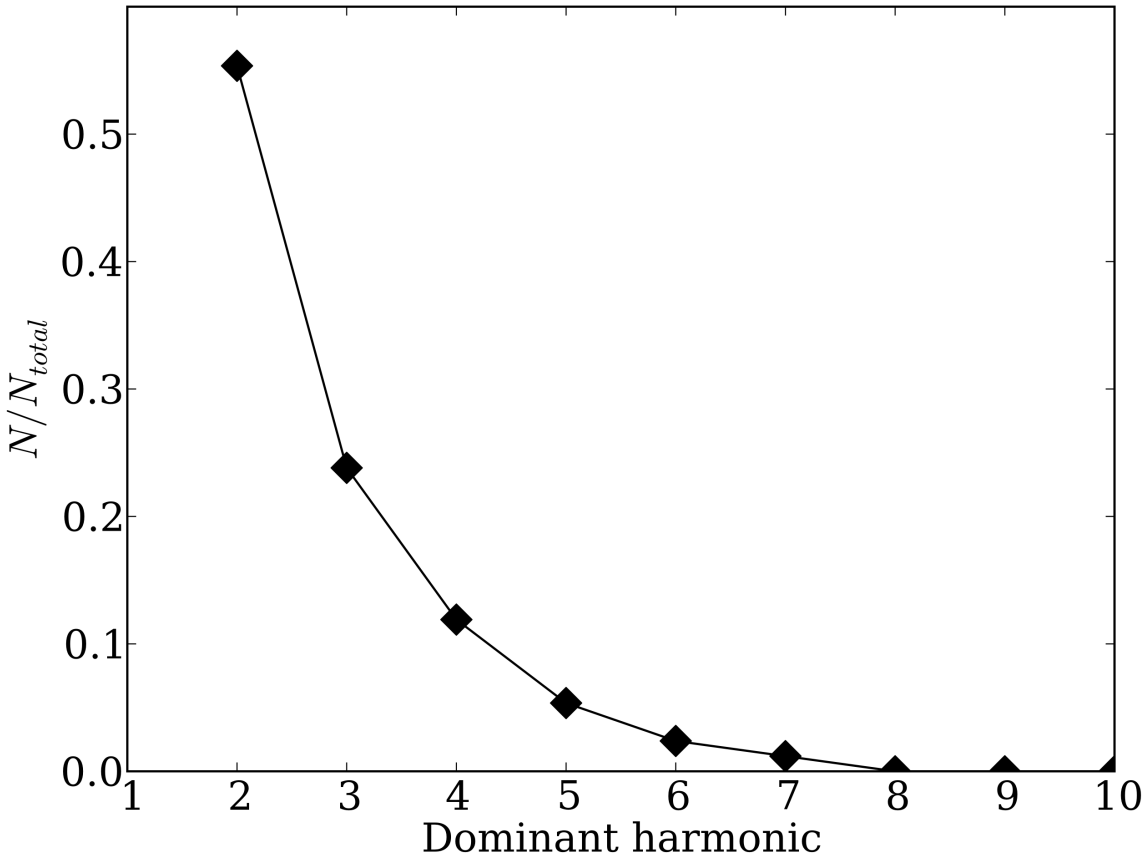

Figure 20: Global Distribution of Primary Harmonics.

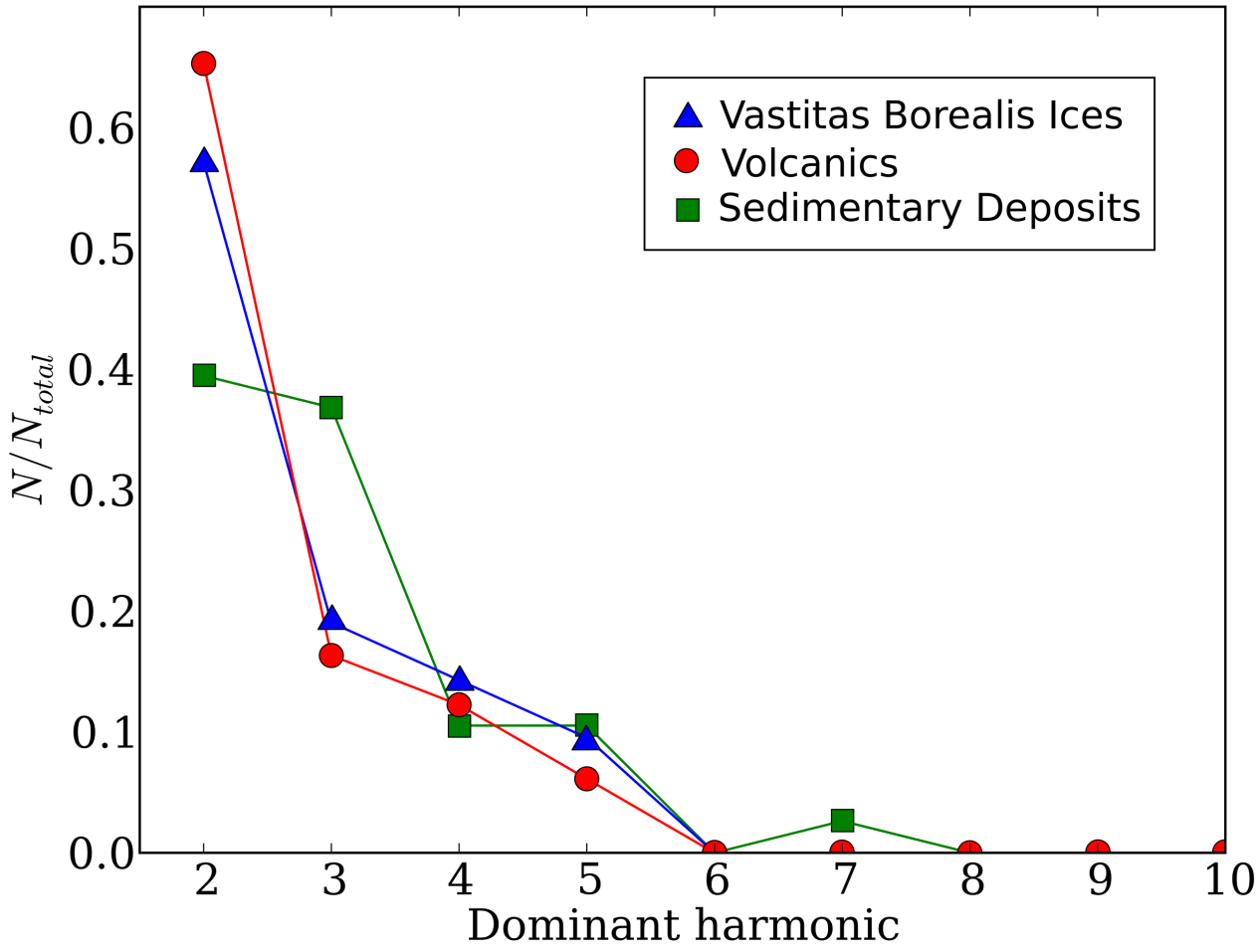

Figure 21: Distribution of primary crater harmonics within the Vastitas Borealis subsurface ices (29 craters in Blue), volcanics (69 craters in Red) and sedimentary rocks (51 craters in Green).

Figure 22 shows the variation in the amplitude of the fourth harmonic, which correlates to the square component of the crater shape. The amplitude of Meteor Crater is shown as a reference. The rim radius-to-height correlation proposed by Poelchau et al. (2009) is plotted against the fourth harmonic amplitude, to test his hypothesis (Figure 23.A). No trends are apparent in our crater sample. We also compare this metric to the radial deviation in planform (δ_R/D) and again, we see no trend (Figure 23.B).

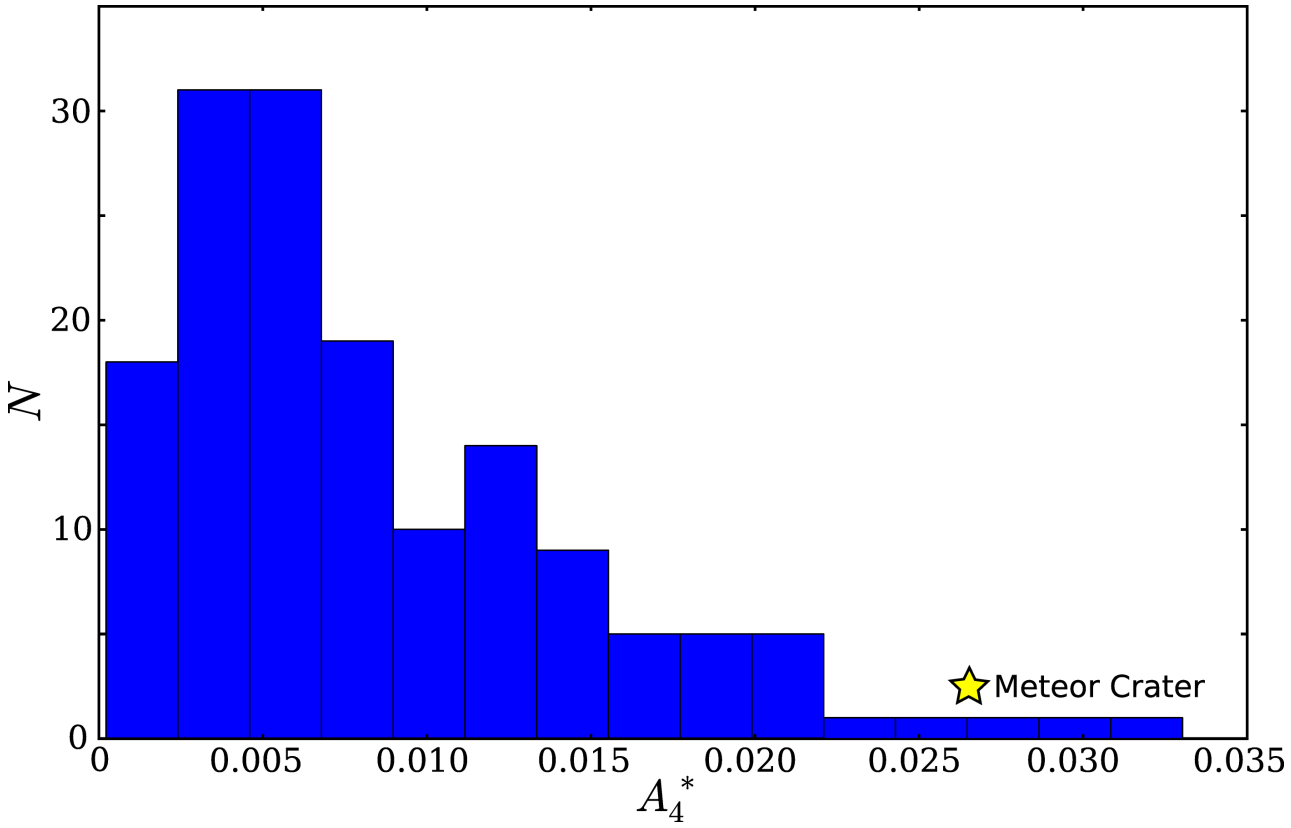

Figure 22: Histogram of fourth harmonic amplitudes. Meteor Crater is denoted with a star.

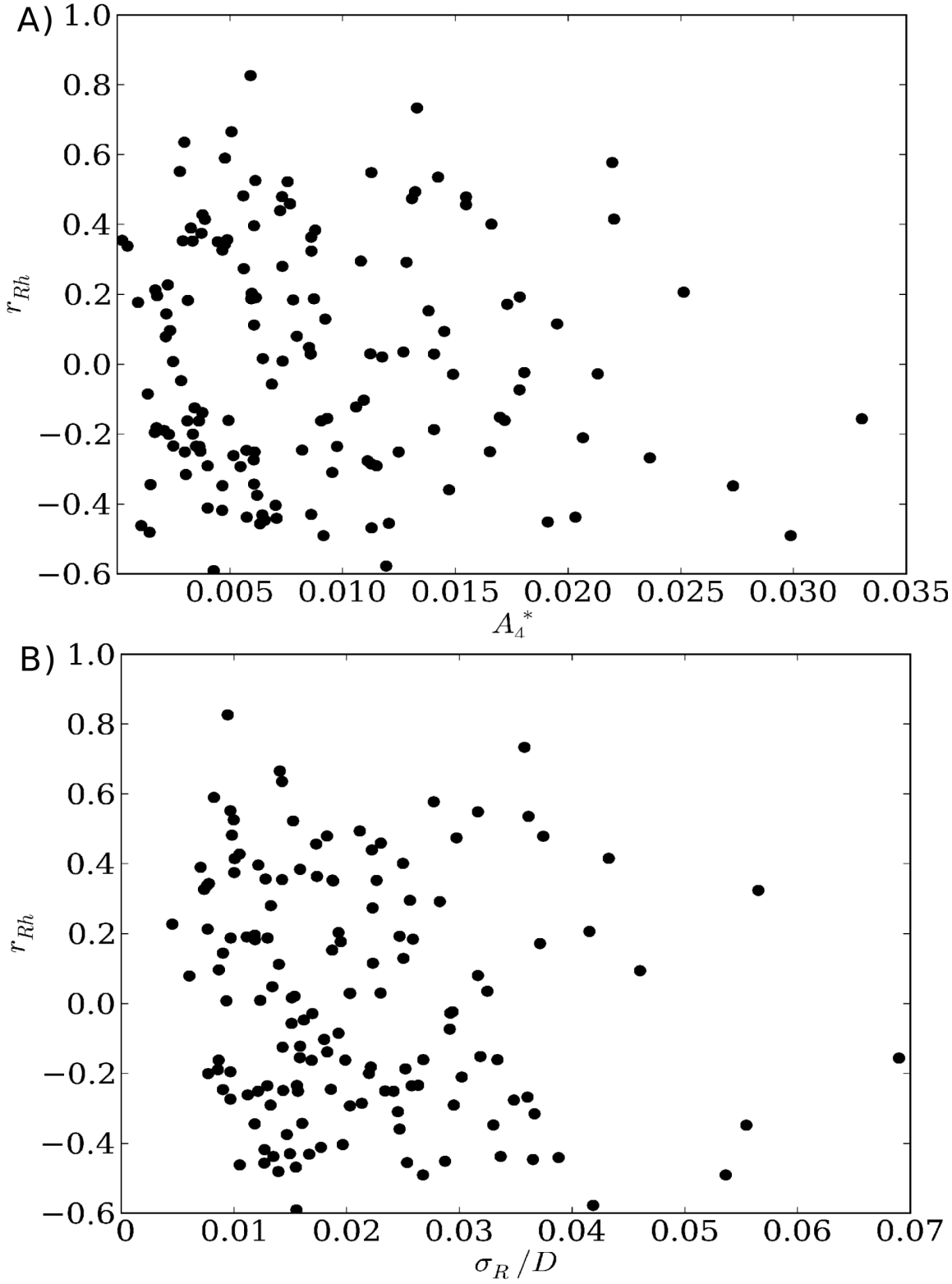

Figure 23: A) Comparison of the rim radius-to-height correlation and the amplitude of the fourth harmonic. No trend is apparent in our dataset. B) Comparison of the rim radius-to-height correlation and the radial deviation in crater planform.

6.3 Metric Correlations

6.3.1 Diameter Correlation

We are interested in the relationships between crater morphometry and crater diameter. Figure 24 shows the logarithmic relationship between crater diameter and the crater rim raggedness (σ_h). The fresh, strong craters ($d/D \geq 0.2, \phi \geq 33°$) are plotted in black and modified craters are white. Rim raggedness in fresh craters is proportional to $D^{-0.25}$, but modified craters appear to have no discernible trend. The fresh craters appear to follow a decreasing trend as crater size increases, but the modified craters are more random, or perhaps follow nearly horizontal trend, suggesting independence from diameter.

Radial deviation has a general downward trend as diameter increases (Figure 25). For all craters, the trend is proportional to D^{-a} for $a \approx 1/4$, but when we only include the fresh craters ($d/D \geq 0.2$) in strong materials (upper wall slope $\geq 33°$), the trend changes to $a \approx 1/3$.

The characteristic rim roundness scale length appears to gradually increase with diameter, until $D \sim 300$m. At this point the scale length quickly drops (Figure 26). This correlates to rims becoming more rounded with diameter then quickly becoming more sharp.

The volume of the crater cavity should increase with target material strength. We plot cavity volume against diameter to compare the global trend with local trends found in Stewart and Valiant (2006). Cavity volume scales very tightly with diameter (Figure 27). The trend line was fit to the log-log plot and translated onto the linear plot. This corresponds to an equation $V = 0.019D^{3.12}$.

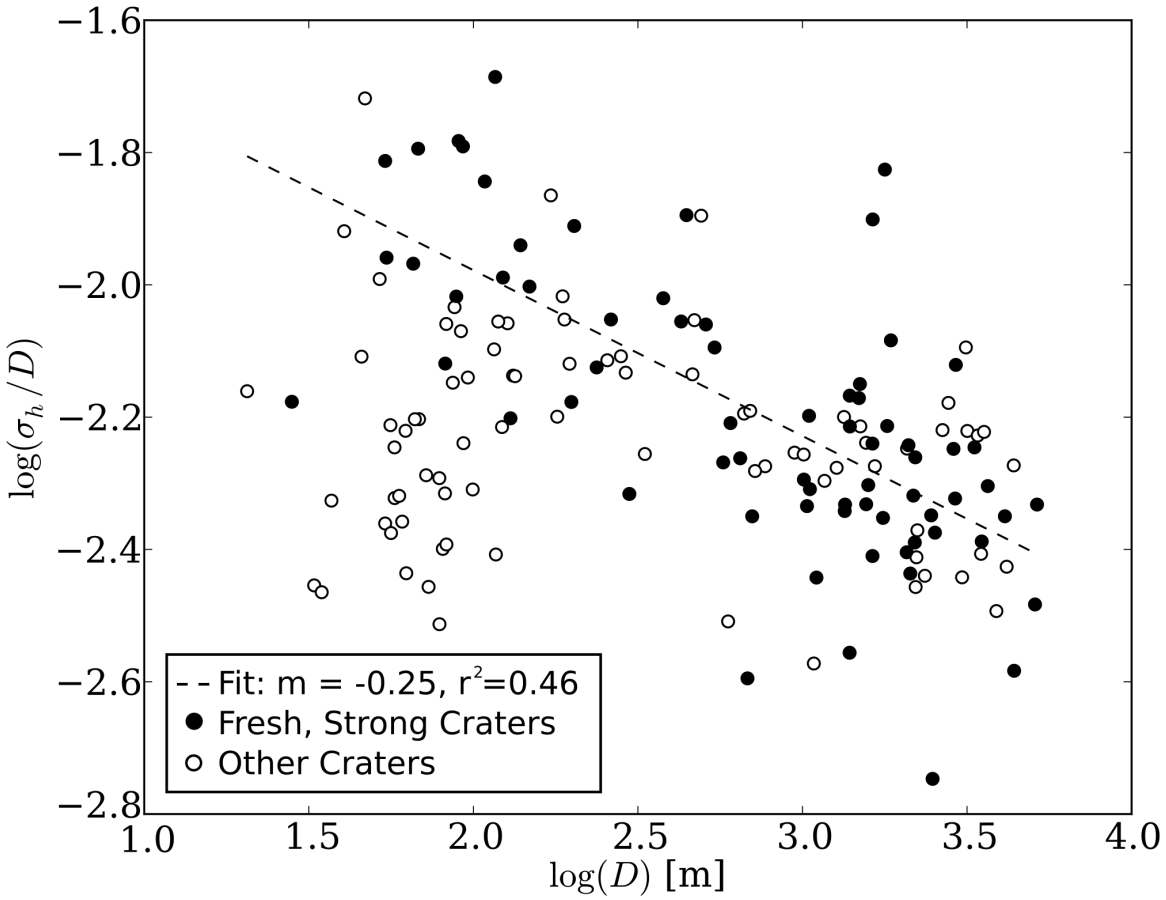

Figure 24: Log plot of diameter against rim raggedness. Black represents fresh ($d/D \geq 0.2$), strong ($\phi \geq 33°$) craters. Fresh crater rim raggedness is inversely correlated to diameter, with a fit of $D^{-.25}$. Modified craters do not fit this trend.

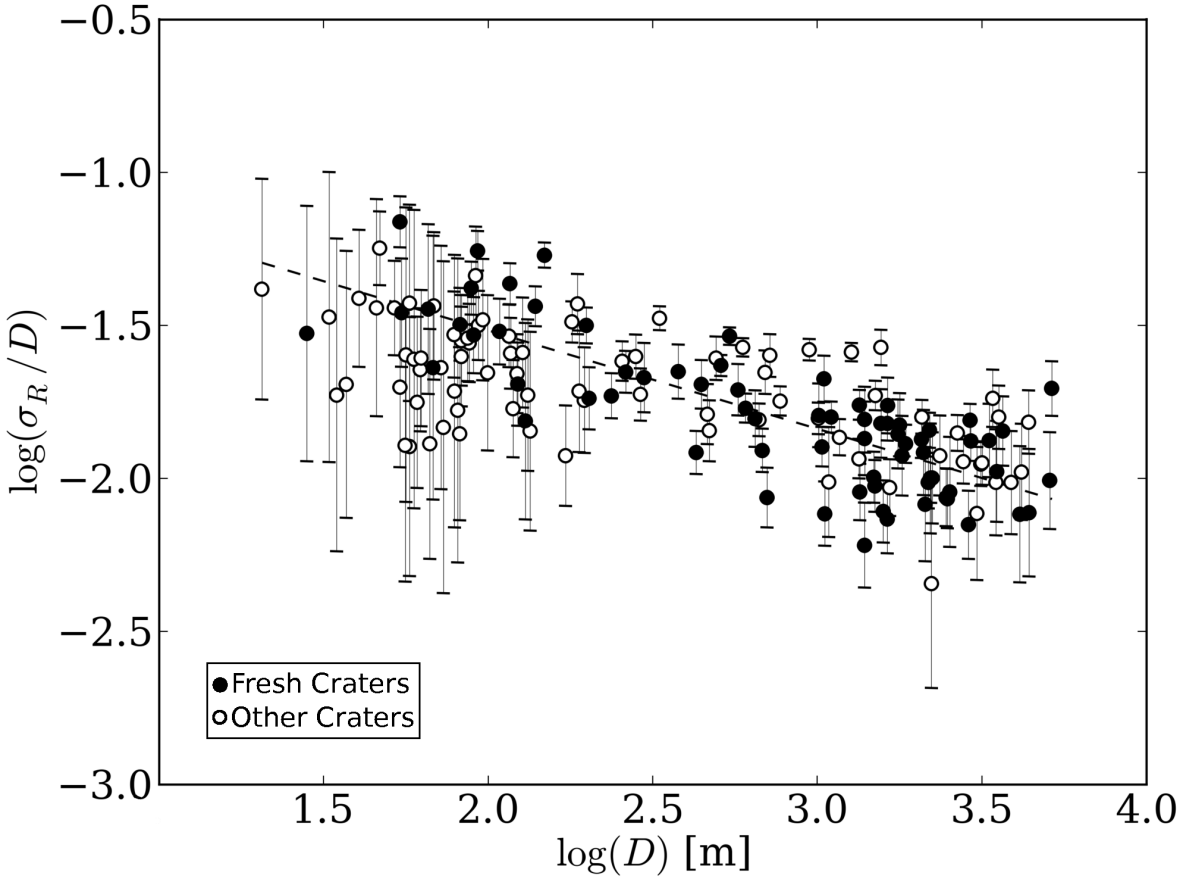

Figure 25: Log plot of planform radial deviation as a function of diameter. Black shows fresh, strong craters, which fit a trend of $D^{-1/3}$ with $r^2 = 0.6$. A trend fit to all craters yields $D^{-1/4}, r^2 = 0.5$.

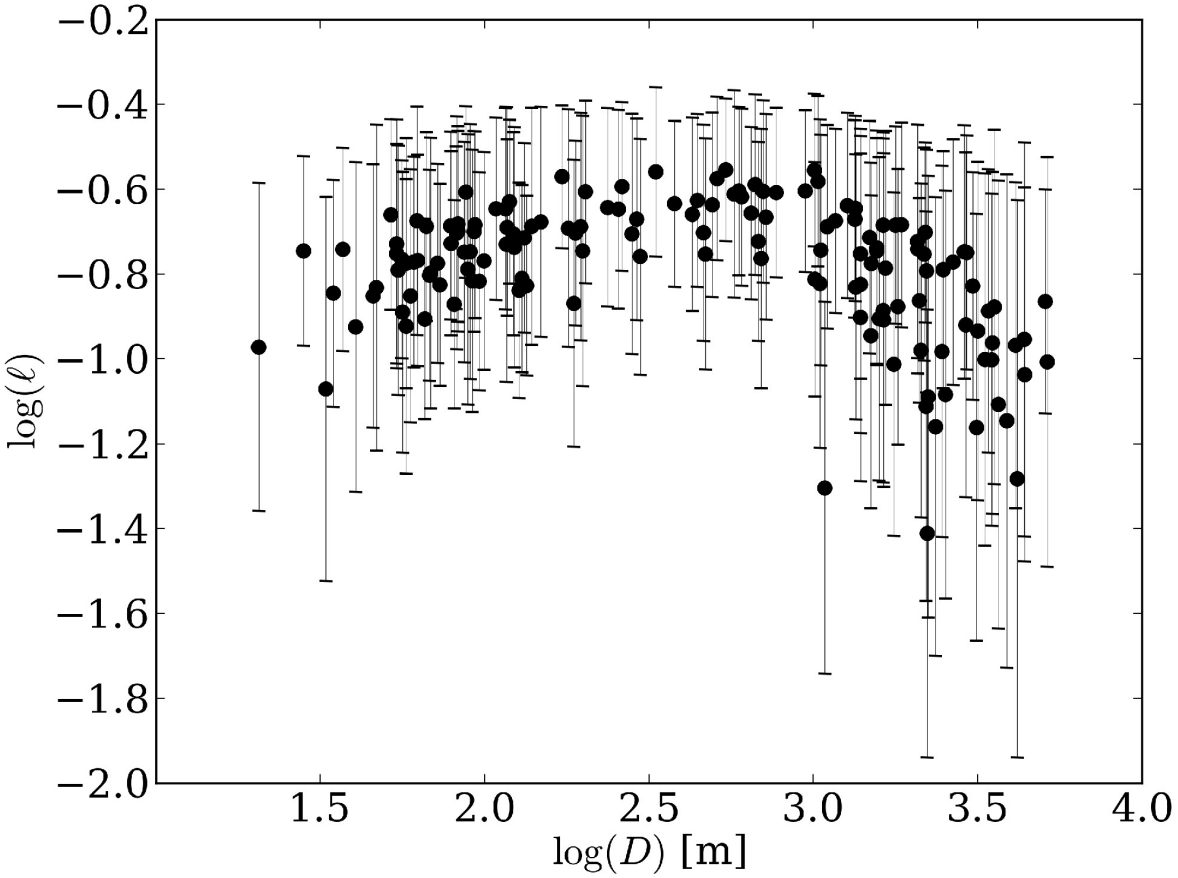

Figure 26: Log plot of rim roundness as a function of diameter. Crater rims appear to become more rounded until a diameter of $\sim$ 300m, where they begin to sharpen. Error bars are one standard deviation of ℓ.

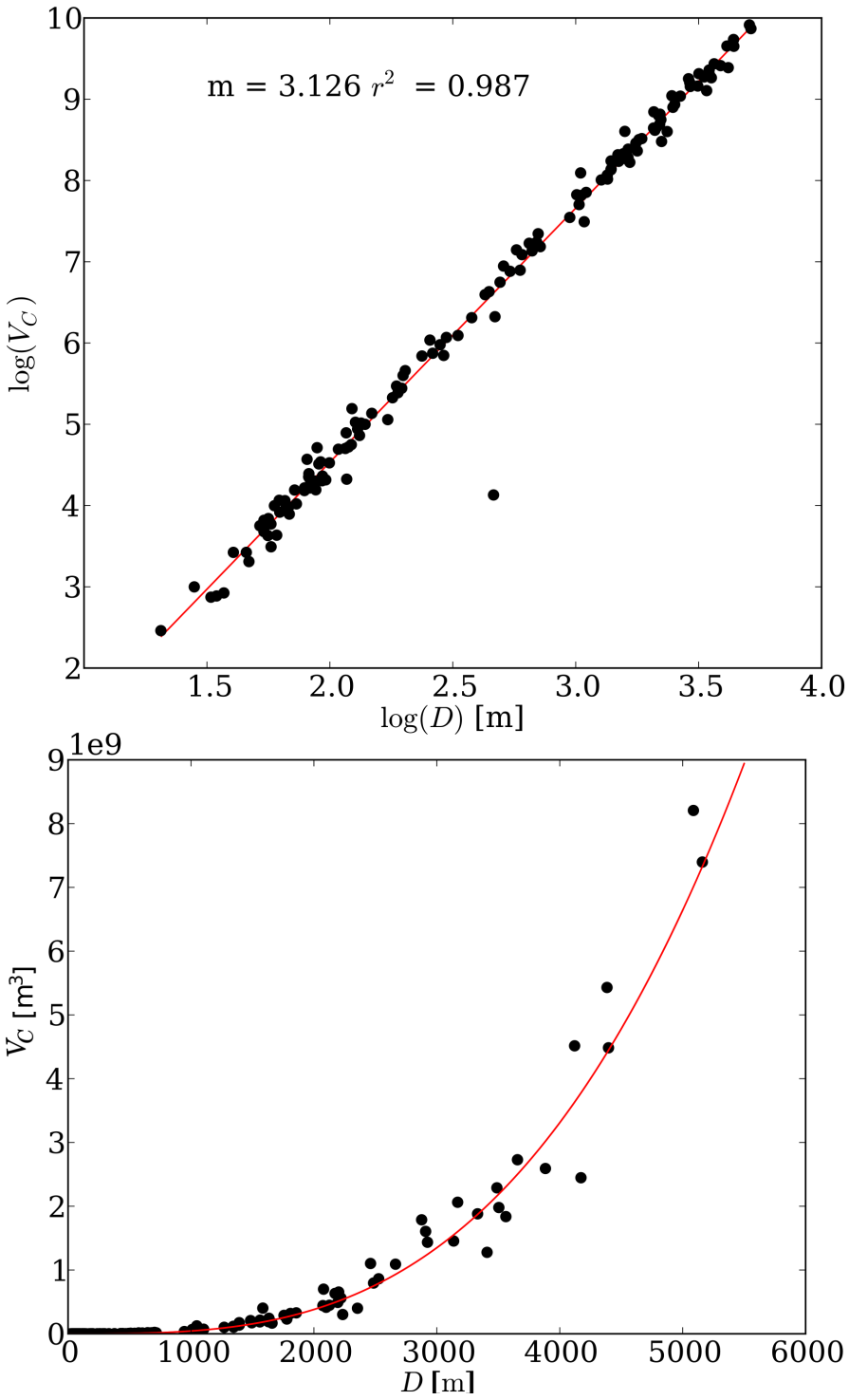

Figure 27: This figure shows the relationship between the cavity volume and crater diameter, in both logarithmic (top) and linear (bottom) scales. Both red trend lines correspond to $V = 0.019D^{3.12}$. The outlier seen in the log-log plot is Crater 129, which is quite modified ($d/D = 0.16$) with a flat crater floor; indicating infilling that accounts for the anomalously low volume.

6.3.2 Other Comparisons

We also looked for relationships between metrics independent of diameter, to examine factors affecting crater morphology aside from crater size. If we use the correlation of radial deviation and diameter ($\sigma_R/D \propto D^{-1/4}$) found in Figure 25, we can attempt to remove dependency on diameter from the metric in order to compare planform asymmetry to other metrics, by dividing the calculated proportionality from the asymmetry metric. Figure 28 displays this diameter-independent radial deviation, ($\sigma_R/D^{3/4}$), against upper-wall slope (target strength). Black represents fresh craters. Fresh craters appear to have larger upper-wall slopes than the modified craters. Significant radial-asymmetry appears to only occur in stronger materials and comparatively fresh craters, though it should be noted our data set contains few highly asymmetric craters.

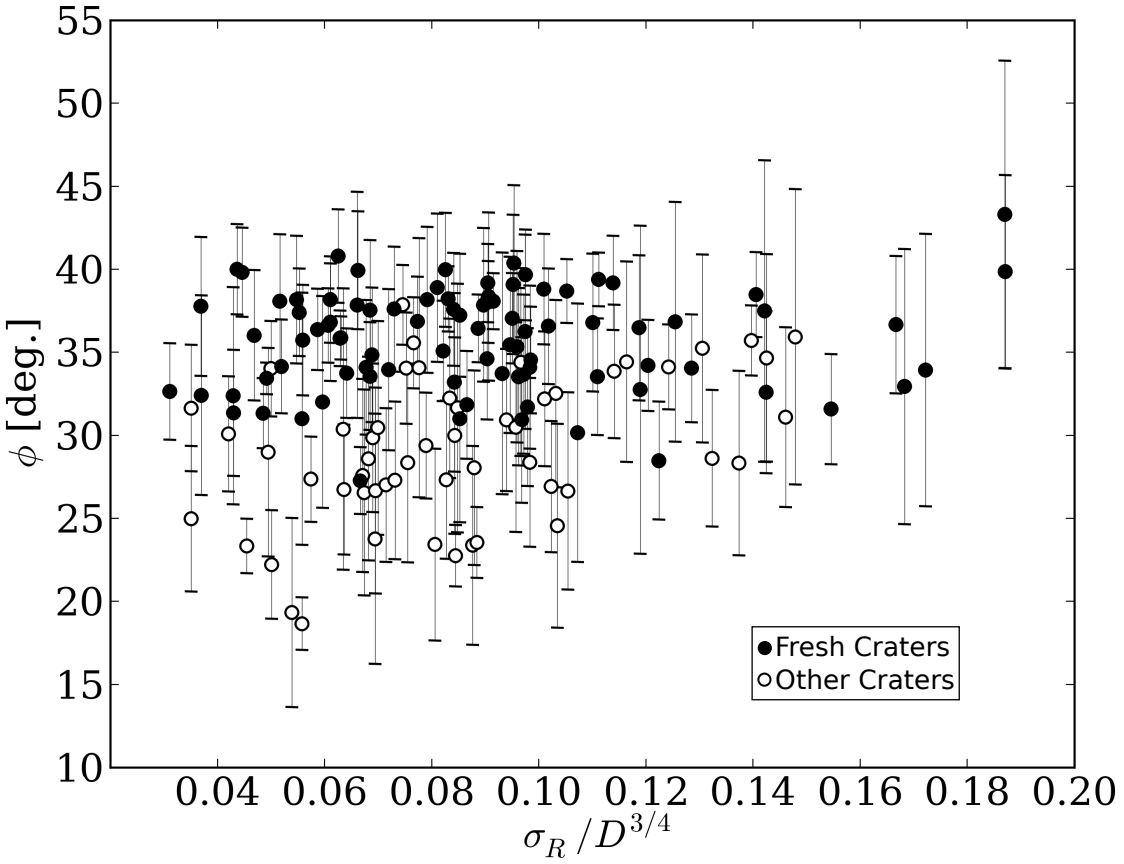

Figure 28: Target strength and crater asymmetry. Black points represent fresh craters ($d/D \geq 0.2$).

7 Discussion

Although we analyzed and compared a wide variety of morphometric parameters, only a handful of significant comparisons have been identified so far. Crater degradation, for the range of $d/D \geq 0.15$, appears to lack any influence on the crater cavity power-law exponent distribution (Figure 14), though the upper wall slope seems to decrease with modification (Figure 28). Modified flanks tend to have a lower decay exponent than fresh flanks (Figure 13). This may be an illustration of the flanks relaxing or being eroded over time, to a more gently decaying profile. This is consistent with other observations, where the ejecta is the first major crater feature to be erased. In fact, most studies use the presence of ejecta as a proxy for freshness.

The roundness of the rim is surprisingly not correlated to the age of the crater (Figure 15). This may indicate a lack of diffusive erosion on Mars, or that diffusion, such as soil creep, is not the most important process modifying Martian landscapes. In a diffusive setting, the entire crater will flatten over time, including the rim, with small craters flattening much faster than large craters (Fassett 2013). Our results show that the roundness of the rim does not depend on modification state (for $d/D \geq 0.15$), but does depend on crater diameter. We found that crater rims tended to become more rounded until a threshold diameter of $\approx$ 300m, at which point the crater rims became more sharp (Figure 26). This could indicate a new type of strength regime. We hypothesize the transition may be indicative of upper rim wall instability. Above the transition diameter, the crater rim may be over steepening during the modification stage. If this is indeed a new strength regime, then we should see a similar transition with lunar craters, though at a larger diameter given the lower gravity.

In general, crater diameter appears to have more influence on fresh crater shape, which could have implications for initial cratering morphometry distributions being dependent on crater diameter. The rim raggedness is decently correlated with diameter with fresh craters, but modified craters seem independent of diameter (Figure 24). This may hint at an initial correlation between raggedness and crater size that is intrinsic to the formation process. The initial rim raggedness distribution may be modified over time to create the random distribution we see in the modified craters. Radial deviation follows a similar trend; fresh craters are more tightly correlated with

diameter, and modified craters are less so (Figure 25). The exact dependence will be compared in future work to model predictions. In this case, the weathered craters have not erased the initial radial deviation distribution as completely. This would imply that the crater planform may be a morphologic feature that is more resistant to modification than the initial variations in rim elevation.

The presence of subsurface ice in the target has little effect on cavity profile shape (Figure 17) or rim roundness (Figure 16). In comparison, crater flank decay is much more influenced by the presence of target ice (Figure 18). This may be caused by the energy of the impact melting the ices and liquidizing the ejecta. Melting would decrease the strength of the deposit, allowing it to behave more similarly to a debris flow than ballistic emplacement and creating a more relaxed decay shape than drier ejecta deposits. This process may not necessarily carry over to the cavity shape, as the energies inside the cavity region will be higher and potentially vaporize water, limiting the mobilization.

The volume of a crater cavity scales by a power of 3.12 with crater diameter (Figure 27); this is higher than the powers found in previous studies. In particular Stewart and Valiant 2006 found that the simple impact crater cavity volume scales with diameter by varying powers based on region. In the lowlands, they found power-law exponents of 2.56 in Acidalia Planitia, 2.59 in Utopia Planitia, and 2.24 in Isidis Planitia. The highlands yielded powers of 2.20 in Solis Planum and 2.64 in Lunae Planum. Our global scaling law is higher than any of the regional powers that Stewart and Valiant had computed. This could mean that either the areas that Stewart and Valiant studied have lower than average crater volumes, or that there is a transition in the scaling of crater volume with diameter around $D \approx 3$km, which is the smallest crater diameter used in their study. Further analysis is needed here.

The global distribution of primary harmonic, computed by Fourier transform, shows an approximately exponential decay of shape order (Figure 20). When split into different geologic settings, we see a very different signal (Figure 21). In sedimentary rocks, the frequency of triangular craters is comparable to the frequency of dominantly oval-shaped craters. This may be due to fracturing patterns in the sedimentary rock that are different or missing in volcanic or icy regions. Sedimentary

rock can develop regional fracture patterns due to local tectonic settings, which can create organized patterns of faults and joints, whereas the fracturing in volcanics is predominately controlled by thermal contraction, which creates fractures with random orientations. Perhaps the triangular craters have developed in sedimentary regions with similar tectonics. Volcanic rocks also tend to be much stronger than sedimentary rock as well, which may hinder the formation of polygonal impact craters.

There is no global correlation between rim height and increased radius. This correlation is neither related to fourth harmonic amplitude nor planform radial deviation (Figure 23). This casts doubt on the height-to-radius correlation hypothesis that Poelchau et al. (2010) suggested to be the result of enhanced uplift in crater corners. The fourth harmonic amplitude of Meteor Crater lies on the tail end of the Martian distribution (Figure 22). Meteor Crater formed in an area with orthogonal fracture sets that were widened by dissolution. Since the frequency of craters with such a high quadradic signature is so low on Mars ($\approx 2.5\%$ of sampled craters), perhaps the conditions in which Meteor Crater formed are extremely uncommon on Mars.

8 Conclusions and Future Work

Using HiRISE stereo-images and a new DEM synthesis program, we were able to successfully extract morphometry statistics for 163 small (20m $\leq D \leq$ 5km), simple impact craters. We found the following morphometric correlations apply to Martian craters:

- Fresh and modified craters have similar cavity power-law exponents, but modified flanks have lower decay rates than fresh flanks.
- Crater flank decay is more relaxed in icy terrain, and relaxes over time globally.
- Rim roundness is independent of crater degradation, but may correlate to crater diameter, with a transition occurring at a crater diameter of $\approx$ 300m.
- In fresh craters, standard deviations in rim radius and height both correlate with crater diameter, $\sigma_R/D \approx D^{-1/3}$ and $\sigma_h/D \approx D^{-2.4}$ respectively. Modified craters lack strong correlations, pointing to increasing homogeneity with age.
- Sedimentary rocks have almost as many triangular craters as oval shaped craters, but triangular craters are not common in other rock types.

Our morphometry extraction programs require only two user inputs for each crater and require little human supervision, making robust data collection for impact craters a nearly automatic process. In the future we will incorporate statistical tests on the different distributions we have found in this study to assess the significance of the variations we have seen. In particular, we plan to apply the Kolmogorov-Smirnov (K-S) test.

We also plan to further investigate the data we have already gathered for this study. We will compare the morphometric distributions more deeply and thoroughly, specifically with regards to the crater and target attributes that we have already identified. Foremost of these comparisons is the distribution of crater harmonics in fractured or layered terrain. We will also use our existing morphometry data to test computer models on crater formation and degradation. We will be able

to analyze how impact crater formation interacts with strength heterogeneities, and whether crater excavation or modification is more important to crater shape. In summary, our results can be used to directly test models of crater growth and modification in a wide range of conditions.

Because our data acquisition and analysis process is highly automatic, future studies will only require feeding more craters to the existing scripts and analyzing the results produced. We will expand our study to include more modified craters, testing the limit on automatic rim extraction, and understanding more completely how crater morphology will change over time. We will use this information to estimate erosion and burial rates across Mars. In addition, we will create computer models to study the effects of viscous relaxation and soil creep mediated modification of impact craters. We also plan to expand the comparison of different geologic settings beyond Vastitas Borealis, and perhaps to continue with the Moon and other planets, and to use these comparisons to test the crater formation models in different geologic settings.

9 References

References

[1] Byrne, S. Dundas, M.C., Kennedy, M.R., Mellon, M.T., McEwen, A.S., Cull, S.C., Daubar, I.J., Shean, D.E., Seelos, K.D., Murchie, S.L., Cantor, B.A., Arvidson, R.E., Edgett, K.S., Reufer, A., Thomas, N., Harrison, T.N., Posiolova, L.V., Seelos, F.P. 2009. *Distribution of Mid-Latitude Ground Ice on Mars from New Impact Craters* Science 325(1674) doi: 10.1126/science.1175307.

[2] Broxton, M. J. and L. J. Edwards. 2008. *The Ames Stereo Pipeline: Automated 3D Surface Reconstruction from Orbital Imagery.* Lunar and Planetary Science Conference 39, abstract 2419.

[3] Calef, F.J., Herrick, R.R, Sharpton, V.L. 2009. *Geomorphic analtsis of small rayed craters on Mars: Examining primary versus secondary impacts.* Journal of Geophysical Research 114 doi:10.1029/2008JE03283.

[4] Carr, M.H., Head, J.W. 2010. *Geologic history of Mars.* Earth and Planetary Science Letters 294 doi: 10.1016/j.epsl.2009.06.042.

[5] Clifford, S.M., Parker, T.J. 2001. *The evolution of the Martian hydrosphere: Implications for the fate of a primordial ocean and the current state of the northern plains.* ICARUS 154(1) doi: 10.1006/icar.2001.6671

[6] Eppler, D.T., Ehrlich, R., Nummedal, D., Schultz, P.H. 1983. *Sources of shape variation in lunar impact craters; Fourier shape analysis* Geological Society of America Bulletin 94(2)

[7] Fassett C. I. 2013. *Crater Degradation of Kilometer-sized craters on the Lunar Maria: Initial observations and modeling.* 44th Lunar and Planetary Science Conference, Conference Proceedings 2026

[8] Fassett, C.I. and Head, J.W. 2005. *Fluvial sedimentary deposits on Mars: Ancient deltas in a crater lake in the Nili Fossae region.* Geophysical Research Letters 32(14) doi:10.1029/2005GL023456

[9] Forsberg-Taylor, N. K., Howard, A. D. 2004. *Crater degradation in the Martian highlands: Morphometric analysis of the Sinus Sabaeus region and simulation modeling suggest fluvial processes.* Journal of Geophysical Research, 109 doi:10.1029/2004JE002242.

[10] Grant, J. A., Wilson, S. A., Cohen, B. A., Golombek, M. P., Geissler, P. E., Sullivan, R.S., Kirk, R. L., Parker, T. J. 2008. *Degradation of Victoria crater, Mars.* Journal of Geophysical Research, 113, doi:10.1029/2008JE003155.

[11] Grant J. A., Arvidson, R. E., Crumpler, L. S., Golombek, M. P., Hahn, B., Haldemann,A. F. C., Li, R., Soderblom, L. A., Squyres, S. W., Wright, S. P., Watters W. A., 2006. *Crater gradation in Gusev crater and Meridiani Planum, Mars.* Journal of Geophysical Research, 111, doi:10.1029/2005JE002465.

[12] Golombek, M. P, Grant, J. A., Crumpler, L. S., Greeley, R., Arvidson, R. E, Bell, J. F., III, Weitz, C. M., Sullivan, R., Christensen, P. R., Soderblom, L. A., Squyres, S. W. 2006. *Erosion rates at the Mars Exploration Rover landing sites and long-term climate change on Mars.* Journal of Geophysical Research-Planets 111(E12) doi: 10.1029/2006JE002754

[13] Hartmann, W.K. and Neukum, G. 2001. *Cratering chronology and the evolution of Mars.* Space Science Reviews, 96(1-4):165-194. 2001

[14] Head, J.W. and Kreslavsky, M.A. 2002. *Northern lowlands of Mars: Evidence for widespread volcanic flooding and tectonic deformation in the Hesperian Period.* Journal of Geophysical Research-Plantes 107(E1).

[15] Melosh, H. J., 2011 *Planetary Surface Processes.* Cambridge University Press, N. Y.

[16] Melosh, H. J., 1989. *Impact Cratering: A Geologic Process.* Oxford University Press, N.Y.

[17] Moratto, Z. M., M. J. Broxton, R. A. Beyer, M. Lundy, and K. Husmann. 2010. *Ames Stereo Pipeline, NASAs Open Source Automated Stereogrammetry Software.* Lunar and Planetary Science Conference 41, abstract 2364.

[18] Ohman, T., Aittola, M., Korteniemi, J., Kostama, V.-P., Raitala, J., 2010. *Polygonal impact craters in the solar system: Observations and implications.* Geological Society of America Special Papers, 465(51-65) doi: 10.1130/2010.2465(04).

[19] Ohman, T., Aittola, M., Kostama, V.P., Raitala, J., Korteniemi, J. 2008. *Polygonal impact craters in Argyre region, Mars: Implications for geology and cratering mechanics.* Meteoritics and Planetary Science 43, Nr 10, 1605162.

[20] Parker, TJ, Gorsline, DS, Saunders, RS, Pieri, DC, Schneeberger, DM 1993. *Coastal Geomorphology of the Martian Northern Plains.* Journal of Geophysical Research-Planets Volume: 98 Issue: E6 Pages: 11061-11078 DOI: 10.1029/93JE00618

[21] Poelchau, M. H., Kenkmann, T., Kring D.A. 2009, *Rim uplift and crater shape in Meteor Crater: Effects of target heterogeneities and trajectory obliquity.* Journal of Geophysical Research 114 doi: 10.1029/2008JE003235.

[22] Roddy, D. J. 1978. *Pre-Impact Conditions and Cratering Processes at Meteor Crater, Arizona, a Typical Bowl-Shaped Crater.* Proceedings of the 9th Lunar and Planetary Science Conference, vol. 3, 3891-3930. Proceedings Paper.

[23] Smith, D., Neumann, G., Arvidson, R. E., Guinness, E. A., Slavney, S. 2003. *Mars Global Surveyor Laser Altimeter Mission Experiment Gridded Data Record*, NASA Planetary Data System, MGS-M-MOLA-5-MEGDR-L3-V1.0.

[24] Stewart, S.T., Valiant, G.J. 2006. *Martian subsurface properties and crater formation processes inferred from fresh impact crater geometries.* Meteorics & Planetary Science 41, Nr. 10, 1509-1537.

[25] Tanaka, K.L., Skinner, J.A., Hare, T.M. *Geologic Map of the Northern Plains of Mars.* U.S. Geological Survey Scientific Investigations Map 2888.

[26] Watters, W.A., Grotzinger, J.P., Bell, JIII, Grant, J., Hayes, A.G, Li, R., Squyres, S.W., Zuber, M.T. 2011. *Origin of the structure and planform of small impact craters in fractured targes: Endurance Crater at Meridiani Planum, Mars* Icarus, doi: 10.1016/j.icarus.2010.08.030

Appendix

Contents

1 Data Tables

The majority of the metrics that we calculated have been normalized, with respect to crater radius, diameter or depth, to produce unitless metrics for comparison. In these cases the units will be listed in the following tables as [R], [D] and [d] respectively.

1.1 Crater Identification Tables

Table 1: Crater Identification Table. i refers to the crater ID. j numbers the feature within individual observations.

i	Observation ID 1	Observation ID 2	j	Latitude	Longitude	Resolution [m/pix]
001	ESP-011425-1775	ESP-012282-1775	0	-2.357	278.261	0.647
002	ESP-011456-2160	PSP-005971-2160	0	35.89	147.783	15.975
003	ESP-011827-1485	ESP-011761-1485	0	-31.271	108.76	1.645
004	ESP-012313-1885	ESP-011891-1885	0	8.223	154.884	0.773
005	ESP-012511-1820	ESP-012234-1820	0	1.991	148.79	2.917
006	ESP-012857-1910	ESP-012791-1910	0	10.678	62.261	2.689
007	ESP-012868-1275	ESP-012947-1275	0	-51.985	131.592	3.315
008	ESP-013149-1540	ESP-012727-1540	0	-25.796	14.708	1.472
010	ESP-013684-1455	ESP-020910-1455	0	-34.064	168.446	1.403
011	ESP-013684-1455	ESP-020910-1455	1	-34.064	168.446	1.387
012	ESP-013684-1455	ESP-020910-1455	2	-34.064	168.446	1.393
013	ESP-013699-1055	ESP-013923-1055	0	-74.198	130.579	7.694
015	ESP-013976-1670	ESP-020279-1670	0	-12.888	113.217	6.355
020	ESP-016134-2285	ESP-016002-2285	0	47.978	234.33	5.827
021	ESP-016226-2235	ESP-016371-2235	0	43.054	240.823	1.665
022	ESP-016478-1935	ESP-016056-1935	1	13.205	204.943	18.278
023	ESP-016502-2355	ESP-016225-2355	0	55.298	264.72	9.119
030	ESP-016681-1575	ESP-017393-1575	0	-22.158	67.319	11.474
031	ESP-016681-1575	ESP-017393-1575	1	-22.158	67.319	0.715
032	ESP-016738-1730	ESP-019296-1730	0	-7.056	309.365	0.695
033	ESP-016797-2170	ESP-016164-2170	0	36.793	133.218	14.633
034	ESP-017153-1830	ESP-017298-1830	0	2.913	138.416	13.485
036	ESP-017968-2355	ESP-017480-2355	0	55.205	200.733	2.746
037	ESP-018058-2275	ESP-018559-2275	0	47.199	263.575	3.896
038	ESP-018272-2185	ESP-017784-2185	1	38.064	184.233	0.903
039	ESP-018398-1715	ESP-025690-1715	0	-8.644	347.396	13.981
041	ESP-018478-2150	ESP-018979-2150	0	34.782	318.609	20.833

Table 2: Crater Identification Table

i	Observation ID 1	Observation ID 2	j	Latitude	Longitude	Resolution [m/pix]
042	ESP-018712-2395	ESP-019147-2395	0	59.361	44.418	13.613
043	ESP-018901-2315	ESP-017490-2315	0	51.248	289.577	1.953
046	ESP-019056-2230	ESP-018423-2230	0	42.754	18.151	2.886
047	ESP-019140-2310	ESP-019641-2310	0	50.823	241.911	1.375
048	ESP-019314-2005	PSP-009121-2005	0	20.315	177.318	1.981
049	ESP-019314-2005	PSP-009121-2005	1	20.315	177.318	0.794
050	ESP-019314-2005	PSP-009121-2005	2	20.315	177.318	0.792
051	ESP-019314-2005	PSP-009121-2005	3	20.315	177.318	0.809
052	ESP-019736-1750	PSP-007394-1750	0	-5.206	180.1	0.846
053	ESP-019784-1900	ESP-020140-1900	0	9.848	305.888	6.447
054	ESP-019784-1900	ESP-020140-1900	1	9.848	305.888	0.735
055	ESP-019784-1900	ESP-020140-1900	2	9.848	305.888	0.764
056	ESP-020078-2265	ESP-019590-2265	0	46.316	197.089	2.528
057	ESP-020133-1720	PSP-010863-1720	0	-7.914	142.063	3.204
058	ESP-020177-1675	ESP-019544-1675	0	-12.32	20.121	12.212
059	ESP-020190-1690	ESP-019346-1690	0	-11.036	25.882	5.854
061	ESP-020240-1970	ESP-020952-1970	0	16.887	95.615	13.428
062	ESP-020245-2190	ESP-020667-2190	0	38.685	316.123	16.122
063	ESP-020337-2195	ESP-019704-2195	0	39.13	325.828	1.727
064	ESP-020337-2195	ESP-019704-2195	1	39.13	325.828	1.667
065	ESP-020875-1490	ESP-020598-1490	0	-30.892	47.18	4.216
066	ESP-021603-1990	ESP-021748-1990	0	18.862	322.648	2.538
068	ESP-022527-1645	ESP-023028-1645	0	-15.579	300.925	0.799
069	ESP-022527-1645	ESP-023028-1645	1	-15.579	300.925	0.793
070	ESP-022527-1645	ESP-023028-1645	2	-15.579	300.925	0.779
071	ESP-022527-1645	ESP-023028-1645	3	-15.579	300.925	0.77
072	ESP-022527-1645	ESP-023028-1645	4	-15.579	300.925	0.754
073	ESP-022657-1775	ESP-023369-1775	0	-2.498	350.394	1.074
074	ESP-023201-1405	ESP-022924-1405	0	-39.116	264.916	5.057
077	ESP-023579-2185	ESP-023302-2185	0	38.238	15.808	3.472
078	ESP-024329-1930	ESP-024474-1930	0	12.65	61.829	3.848
079	ESP-024329-1930	ESP-024474-1930	1	12.65	61.829	0.795
080	ESP-024637-1445	ESP-024927-1445	0	-35.005	300.941	3.928
081	ESP-024835-1640	ESP-025468-1640	0	-15.794	288.793	0.911
083	ESP-025116-2320	ESP-025195-2320	0	51.718	172.057	9.464
084	ESP-025242-2030	ESP-025176-2030	0	22.583	334.685	8.175
085	ESP-025346-1945	ESP-024924-1945	0	14.299	15.652	1.562
086	ESP-025352-2355	ESP-025497-2355	0	55.018	206.624	7.896
088	ESP-025445-1630	PSP-008514-1630	0	-16.831	197.603	0.759

Table 3: Crater Identification Table

i	Observation ID 1	Observation ID 2	j	Latitude	Longitude	Resolution [m/pix]
089	ESP-025664-2365	ESP-025888-2365	0	56.398	327.278	13.467
090	ESP-025775-2290	ESP-026065-2290	0	48.837	177.674	0.995
091	ESP-025788-2200	ESP-025867-2200	0	39.58	185.875	2.279
092	ESP-025801-2185	ESP-025735-2185	0	38.151	191.585	5.91
093	ESP-026316-1560	ESP-025894-1560	0	-23.574	179.268	1.455
094	ESP-026316-1560	ESP-025894-1560	1	-23.574	179.268	1.358
095	ESP-026383-2465	ESP-026673-2465	0	66.008	137.19	4.363
096	ESP-026419-2320	ESP-026076-2320	0	51.665	236.711	1.084
097	ESP-026455-2460	ESP-025901-2460	0	65.768	334.81	15.812
098	ESP-026896-1935	ESP-017191-1935	0	13.229	178.577	1.365
099	ESP-027067-1545	PSP-008145-1545	0	-25.047	195.817	0.746
103	PSP-001432-2015	PSP-001630-2015	0	21.554	222.351	0.848
104	PSP-002118-1510	PSP-003608-1510	0	-28.687	226.937	5.229
105	PSP-002118-1510	PSP-003608-1510	1	-28.687	226.937	0.746
106	PSP-002204-1655	PSP-001782-1655	0	-14.19	38.63	5.226
107	PSP-002461-1545	PSP-002039-1545	0	-25.402	223.917	0.747
109	PSP-003730-1885	PSP-003097-1885	0	8.39	134.927	5.366
110	PSP-005429-1510	PSP-005851-1510	0	-28.493	193.971	0.709
111	PSP-005517-2240	PSP-005807-2240	0	43.81	301.54	6.095
112	PSP-005903-1965	PSP-005837-1965	0	16.419	209.716	4.786
113	PSP-006696-1895	PSP-007408-1895	0	9.347	155.946	0.759
114	PSP-006696-1895	PSP-007408-1895	1	9.347	155.946	0.785
116	PSP-006887-2050	PSP-007823-2050	0	24.86	339.543	0.865
117	PSP-006924-2335	PSP-009864-2335	0	53.111	44.557	7.538
119	PSP-007209-1565	PSP-008290-1565	0	-23.323	194.046	1.126
120	PSP-008304-1860	ESP-012919-1860	0	5.729	169.653	2.174
121	PSP-008304-1860	ESP-012919-1860	1	5.729	169.653	0.731
122	PSP-008304-1860	ESP-012919-1860	2	5.729	169.653	0.779
123	PSP-008304-1860	ESP-012919-1860	3	5.729	169.653	0.717
124	PSP-008304-1860	ESP-012919-1860	4	5.729	169.653	0.736
125	PSP-009205-1665	PSP-009917-1665	0	-13.426	48.188	4.112
126	PSP-009333-2025	PSP-009834-2025	0	22.468	151.449	4.639
127	PSP-009414-1915	PSP-008992-1915	0	11.451	100.751	16.335
128	PSP-009538-2220	PSP-010039-2220	0	41.785	309.756	0.919
129	PSP-009907-1905	PSP-009762-1905	0	10.408	318.624	1.7
130	ESP-027439-2200	ESP-026951-2200	0	39.734	113.858	2.588
131	ESP-027477-2170	ESP-026910-2170	0	36.517	155.4	1.663
132	ESP-027585-1800	ESP-027651-1800	0	0.04	92.421	0.748
133	ESP-027678-1585	ESP-019120-1585	0	-21.217	77.312	0.67
134	ESP-027710-2260	ESP-026365-2260	0	45.416	271.495	2.875

Table 4: Crater Identification Table

i	Observation ID 1	Observation ID 2	j	Latitude	Longitude	Resolution [m/pix]
135	ESP-027775-1675	ESP-028052-1675	0	-12.403	307.033	4.157
136	ESP-027775-1675	ESP-028052-1675	1	-12.403	307.033	0.736
137	ESP-027800-1685	ESP-028789-1685	0	-11.202	345.72	2.874
138	ESP-027847-2145	ESP-026937-2145	0	34.139	137.838	2.045
139	ESP-027866-2320	ESP-026099-2320	0	51.635	333.025	2.63
140	ESP-027949-2280	ESP-028305-2280	0	47.524	229.449	1.958
141	ESP-028113-2195	ESP-028469-2195	0	39.379	75.212	3.103
142	ESP-028206-2310	ESP-028285-2310	0	50.663	56.39	2.004
143	ESP-028207-1660	ESP-025174-1660	0	-14.001	34.65	1.05
144	ESP-028207-1660	ESP-025174-1660	1	-14.001	34.65	0.654
145	ESP-028327-1720	ESP-028604-1720	0	-7.714	358.014	2.308
146	ESP-028410-2170	ESP-028766-2170	0	36.728	247.999	5.495
147	ESP-028629-1660	ESP-029051-1660	0	-13.921	34.438	0.938
148	ESP-028719-1615	ESP-028864-1615	0	-18.477	99.036	1.206
149	ESP-028719-1615	ESP-028864-1615	1	-18.477	99.036	1.35
150	ESP-028990-2130	ESP-029135-2130	0	32.613	254.148	1.474
151	PSP-009240-2055	ESP-017943-2055	0	25.015	167.588	1.581
152	PSP-009240-2055	ESP-017943-2055	1	25.015	167.588	1.597
153	PSP-009244-1575	PSP-009666-1575	0	-22.296	64.874	1.543
154	PSP-009244-1575	PSP-009666-1575	1	-22.296	64.874	1.41
155	PSP-009327-1665	PSP-009960-1665	0	-13.311	317.665	1.573
156	PSP-009606-1665	PSP-008828-1665	0	-13.219	262.115	0.847
157	PSP-009721-1785	PSP-008073-1785	0	-1.277	358.306	0.77
158	PSP-009729-1735	ESP-015953-1735	0	-6.536	141.154	0.715
159	PSP-009768-1880	PSP-010045-1880	0	7.722	154.446	0.812
160	PSP-010423-1720	ESP-016726-1720	0	-8.016	275.895	0.709
161	PSP-010628-1975	PSP-010206-1975	0	17.247	76.39	0.809
162	ESP-020571-1700	ESP-019371-1700	0	-9.825	63.129	1.408
163	ESP-020468-1600	ESP-020046-1600	0	-19.683	356.649	0.769
164	ESP-020468-1600	ESP-020046-1600	1	-19.683	356.649	0.666
165	ESP-020465-1520	ESP-013885-1520	0	-27.971	79.159	1.261
166	ESP-019982-1825	ESP-020404-1825	0	2.462	301.202	1.452
167	ESP-020940-1895	ESP-013582-1895	0	9.135	67.35	1.557
168	ESP-020850-1845	ESP-020705-1845	0	4.214	2.864	0.757
169	ESP-014439-1505	ESP-020742-1505	0	-29.282	74.515	1.391
170	ESP-020167-1640	ESP-020312-1640	0	-15.634	292.455	1.502
171	ESP-020167-1640	ESP-020312-1640	1	-15.634	292.455	1.342
172	ESP-019385-1555	PSP-007109-1555	0	-24.274	43.453	1.291
173	ESP-018550-1890	ESP-019262-1890	0	8.991	156.621	1.543
174	ESP-018550-1890	ESP-019262-1890	1	8.991	156.621	1.432

Table 5: Crater Identification Table

i	Observation ID 1	Observation ID 2	j	Latitude	Longitude	Resolution [m/pix]
175	ESP-018584-1795	ESP-019085-1795	0	-0.641	309.36	1.55
176	ESP-018584-1795	ESP-019085-1795	1	-0.641	309.36	1.52
177	ESP-018474-1970	ESP-017841-1970	0	17.319	71.725	1.639
178	ESP-018814-1740	ESP-018392-1740	0	-5.894	150.778	0.719
179	ESP-018814-1740	ESP-018392-1740	1	-5.894	150.778	0.744
180	ESP-018144-2185	ESP-018856-2185	0	37.982	75.608	0.803
181	ESP-018774-1965	ESP-018141-1965	0	16.086	160.723	1.552
182	PSP-001348-1770	PSP-001678-1770	0	-3.061	356.795	4.128
183	ESP-012991-1335	ESP-013624-1335	0	-45.926	9.54	5.792
184	ESP-019439-1975	ESP-019162-1975	0	17.196	5.493	5.287
185	ESP-025668-2390	ESP-026314-2390	0	58.661	217.254	6.37
186	ESP-026253-1680	ESP-027387-1680	0	-11.866	97.714	11.755
187	ESP-018272-2185	ESP-017784-2185	0	38.064	184.233	1.504
188	ESP-014405-1635	ESP-021776-1635	0	-16.261	281.987	3.108
189	ESP-022613-1230	ESP-022481-1230	0	-56.911	123.191	8.425
190	ESP-021520-1550	ESP-021454-1550	0	-24.949	76.83	1.856
192	PSP-004313-1760	ESP-012356-1760	0	-3.742	59.167	2.156
193	ESP-025115-2220	ESP-025616-2220	0	41.471	198.511	2.169
194	ESP-016083-2205	PSP-010571-2205	0	40.22	187.904	5.571
195	ESP-011563-2200	ESP-011853-2200	0	39.708	104.118	4.422
197	ESP-021596-1500	ESP-020752-1500	0	-29.542	163.122	6.366
198	ESP-014284-2045	ESP-013651-2045	0	24.038	342.109	1.161

1.2 Crater Planform Metrics

Table 6: Data table containing information for crater planform. σ_R/D, $\Delta\sigma_R/D$, Δ_R/D, $d_{\max}/D$, and d/D are ratios and have units of $[D]$. r_{Rh} and C_R are correlation coefficients

i	D [m]	δD [m]	σ_R/D	$\Delta\sigma_R/D$	Δ_R/D	$\sigma_R/D^{3/4}$ [m$^{1/4}$]	r_{Rh}	C_R	$d_{\max}/D$	d/D
001	53.945	0.647	0.02	0.012	0.114	0.054	-0.162	0.549	0.149	0.137
002	4384.215	15.975	0.015	0.004	0.089	0.124	0.522	0.727	0.186	0.171
003	444.185	1.645	0.02	0.004	0.115	0.093	0.029	0.578	0.218	0.182
004	54.489	0.773	0.035	0.014	0.193	0.095	-0.276	0.493	0.266	0.226
005	726.136	2.917	0.039	0.004	0.27	0.201	0.419	0.543	0.175	0.15
006	1390.679	2.689	0.006	0.002	0.045	0.037	0.079	0.832	0.262	0.254
007	1659.379	3.315	0.009	0.002	0.052	0.06	0.008	0.851	0.202	0.191
008	108.874	1.472	0.127	0.014	0.49	0.412	0.586	0.127	0.153	0.131
010	195.998	1.403	0.018	0.007	0.121	0.067	-0.103	0.639	0.169	0.153
011	64.588	1.387	0.061	0.021	0.381	0.173	0.09	0.324	0.189	0.166
012	68.388	1.393	0.037	0.02	0.212	0.105	-0.316	0.48	0.156	0.135
013	2234.113	7.694	0.01	0.003	0.06	0.069	0.375	0.822	0.182	0.171
015	1779.668	6.355	0.015	0.004	0.086	0.097	-0.43	0.789	0.229	0.201
020	3169.187	5.827	0.011	0.002	0.058	0.084	-0.261	0.824	0.218	0.204
021	297.84	1.665	0.021	0.006	0.121	0.089	-0.285	0.659	0.235	0.223
022	5087.183	18.278	0.01	0.004	0.058	0.083	0.482	0.74	0.242	0.231
023	2201.592	9.119	0.01	0.004	0.063	0.068	0.526	0.839	0.225	0.208
030	3140.231	11.474	0.098	0.004	0.574	0.735	-0.005	0.437	0.219	0.205
031	87.659	0.715	0.028	0.008	0.124	0.085	0.577	0.6	0.169	0.152
032	60.775	0.695	0.018	0.011	0.087	0.049	-0.411	0.76	0.179	0.169
033	3883.225	14.633	0.01	0.004	0.06	0.077	-0.195	0.827	0.191	0.178
034	3488.488	13.485	0.01	0.004	0.053	0.075	0.552	0.928	0.195	0.189
036	1494.664	2.746	0.009	0.002	0.049	0.059	0.826	0.874	0.247	0.225
037	1009.716	3.896	0.016	0.004	0.101	0.091	-0.343	0.767	0.236	0.226
038	179.841	0.903	0.032	0.005	0.189	0.119	0.036	0.417	0.202	0.186
039	3504.331	13.981	0.011	0.004	0.063	0.081	-0.462	0.809	0.207	0.199
041	5159.933	20.833	0.02	0.004	0.099	0.167	-0.404	0.648	0.212	0.199
042	3655.906	13.613	0.014	0.004	0.082	0.111	0.635	0.69	0.205	0.186
043	1032.086	1.953	0.013	0.002	0.071	0.072	-0.457	0.747	0.22	0.211

Table 7: Data table containing information for crater planform

i	D [m]	δD [m]	σ_R/D	$\Delta\sigma_R/D$	Δ_R/D	$\sigma_R/D^{3/4}$ [m$^{1/4}$]	r_{Rh}	C_R	$d_{\max}/D$	d/D
046	717.62	2.886	0.025	0.004	0.148	0.131	-0.187	0.523	0.199	0.187
047	703.471	1.375	0.009	0.002	0.051	0.045	-0.162	0.744	0.254	0.244
048	492.261	1.981	0.025	0.004	0.126	0.116	-0.359	0.544	0.179	0.15
049	82.826	0.794	0.028	0.01	0.175	0.085	0.292	0.394	0.209	0.202
050	62.536	0.792	0.025	0.013	0.12	0.069	0.193	0.588	0.138	0.128
051	86.607	0.809	0.029	0.009	0.18	0.088	-0.451	0.626	0.16	0.128
052	176.4	0.846	0.049	0.005	0.287	0.18	0.264	0.495	0.268	0.246
053	1808.669	6.447	0.012	0.004	0.078	0.077	-0.344	0.731	0.244	0.229
054	55.982	0.735	0.013	0.013	0.066	0.035	0.356	0.781	0.199	0.185
055	34.607	0.764	0.019	0.022	0.116	0.045	0.353	0.667	0.128	0.121
056	601.716	2.528	0.056	0.004	0.263	0.278	0.442	0.454	0.183	0.167
057	1631.212	3.204	0.015	0.002	0.082	0.096	0.016	0.65	0.241	0.227
058	3331.346	12.212	0.013	0.004	0.061	0.101	0.28	0.844	0.212	0.195
059	2923.88	5.854	0.013	0.002	0.063	0.097	-0.291	0.732	0.217	0.197
061	3408.577	13.428	0.018	0.004	0.078	0.14	-0.138	0.908	0.19	0.177
062	4120.641	16.122	0.008	0.004	0.06	0.061	0.338	0.721	0.24	0.222
063	377.564	1.727	0.022	0.005	0.109	0.098	0.115	0.681	0.215	0.192
064	189.144	1.667	0.019	0.009	0.089	0.071	-0.085	0.676	0.168	0.151
065	2078.376	4.216	0.016	0.002	0.07	0.107	0.384	0.736	0.243	0.23
066	693.625	2.538	0.022	0.004	0.131	0.114	0.439	0.749	0.192	0.178
068	117.055	0.799	0.026	0.007	0.129	0.084	0.295	0.708	0.167	0.158
069	115.604	0.793	0.029	0.007	0.2	0.096	-0.027	0.607	0.182	0.168
070	82.733	0.779	0.025	0.009	0.127	0.076	0.129	0.622	0.168	0.152
071	93.298	0.77	0.032	0.008	0.157	0.098	0.08	0.741	0.156	0.14
072	78.804	0.754	0.03	0.01	0.193	0.088	-0.29	0.636	0.173	0.163
073	290.408	1.074	0.019	0.004	0.155	0.078	0.351	0.549	0.148	0.13
074	2663.662	5.057	0.014	0.002	0.066	0.101	0.665	0.732	0.181	0.169
077	1856.303	3.472	0.013	0.002	0.069	0.085	0.187	0.814	0.242	0.224
078	1047.026	3.848	0.021	0.004	0.139	0.12	0.493	0.684	0.249	0.238
079	122.216	0.795	0.022	0.007	0.122	0.073	-0.2	0.645	0.165	0.15
080	2073.049	3.928	0.013	0.002	0.078	0.09	0.048	0.635	0.238	0.227
081	255.264	0.911	0.024	0.004	0.108	0.097	-0.251	0.717	0.185	0.17
083	2526.621	9.464	0.009	0.004	0.051	0.064	0.144	0.786	0.21	0.197
084	2099.221	8.175	0.012	0.004	0.082	0.082	-0.25	0.773	0.211	0.2
085	78.898	1.562	0.019	0.02	0.094	0.057	0.203	0.635	0.153	0.144
086	2219.996	7.896	0.005	0.004	0.027	0.031	0.228	0.909	0.217	0.206
088	139.039	0.759	0.037	0.005	0.259	0.125	-0.446	0.285	0.231	0.199
089	3561.512	13.467	0.016	0.004	0.09	0.122	-0.155	0.686	0.2	0.179
090	331.985	0.995	0.033	0.003	0.205	0.142	-0.161	0.395	0.188	0.172
091	1270.052	2.279	0.026	0.002	0.175	0.155	0.184	0.508	0.226	0.216

Table 8: Data table containing information for crater planform

i	D [m]	δD [m]	σ_R/D	$\Delta\sigma_R/D$	Δ_R/D	$\sigma_R/D^{3/4}$ [$m^{1/4}$]	r_{Rh}	C_R	d_{max}/D	d/D
092	1634.165	5.91	0.017	0.004	0.097	0.11	0.456	0.791	0.225	0.207
093	96.229	1.455	0.033	0.015	0.193	0.103	-0.347	0.386	0.126	0.107
094	127.063	1.358	0.026	0.011	0.171	0.087	-0.235	0.64	0.215	0.191
095	1082.652	4.363	0.01	0.004	0.072	0.056	0.188	0.824	0.148	0.136
096	594.27	1.084	0.027	0.002	0.177	0.132	-0.16	0.588	0.186	0.177
097	4172.392	15.812	0.011	0.004	0.051	0.084	0.428	0.897	0.149	0.137
098	680.539	1.365	0.012	0.002	0.057	0.063	0.009	0.761	0.209	0.204
099	236.901	0.746	0.019	0.003	0.114	0.073	-0.246	0.632	0.253	0.228
103	198.521	0.848	0.032	0.004	0.165	0.119	0.549	0.672	0.266	0.253
104	2876.266	5.229	0.007	0.002	0.062	0.052	0.39	0.747	0.221	0.205
105	66.417	0.746	0.013	0.011	0.076	0.037	-0.235	0.693	0.229	0.197
106	1393.127	5.226	0.014	0.004	0.073	0.083	-0.438	0.766	0.245	0.232
107	148.087	0.747	0.054	0.005	0.237	0.187	-0.49	0.379	0.253	0.228
109	1392.244	5.366	0.016	0.004	0.085	0.095	-0.235	0.745	0.247	0.23
110	20.539	0.709	0.042	0.035	0.218	0.088	0.206	0.596	0.158	0.141
111	1561.477	6.095	0.015	0.004	0.074	0.095	-0.057	0.816	0.239	0.226
112	2485.307	4.786	0.009	0.002	0.046	0.061	-0.189	0.888	0.259	0.253
113	90.194	0.759	0.029	0.008	0.199	0.091	-0.023	0.574	0.246	0.214
114	122.997	0.785	0.02	0.006	0.107	0.068	0.029	0.67	0.201	0.183
116	202.51	0.865	0.018	0.004	0.087	0.069	0.479	0.689	0.225	0.199
117	2126.086	7.538	0.008	0.004	0.059	0.056	0.589	0.824	0.226	0.218
119	604.344	1.126	0.017	0.002	0.089	0.084	-0.029	0.649	0.252	0.237
120	574.196	2.174	0.019	0.004	0.112	0.095	0.177	0.313	0.31	0.275
121	58.765	0.731	0.075	0.012	0.358	0.207	-0.25	0.306	0.179	0.143
122	116.498	0.779	0.043	0.007	0.206	0.142	0.415	0.353	0.226	0.176
123	53.955	0.717	0.069	0.013	0.449	0.187	-0.156	0.105	0.256	0.214
124	46.856	0.736	0.057	0.016	0.375	0.148	0.324	0.31	0.188	0.139
125	2171.985	4.112	0.01	0.002	0.066	0.066	-0.273	0.753	0.264	0.254
126	2460.63	4.639	0.009	0.002	0.054	0.061	0.096	0.745	0.241	0.227
127	4396.405	16.335	0.008	0.004	0.061	0.063	-0.2	0.904	0.206	0.197
128	261.879	0.919	0.022	0.004	0.114	0.09	0.274	0.724	0.25	0.22
129	463.141	1.7	0.016	0.004	0.094	0.075	-0.047	0.767	0.177	0.162
130	1346.884	2.588	0.009	0.002	0.051	0.055	-0.246	0.832	0.223	0.212
131	87.505	1.663	0.031	0.019	0.172	0.094	0.262	0.667	0.24	0.227
132	81.988	0.748	0.014	0.009	0.088	0.042	0.113	0.667	0.187	0.174
133	24.067	0.67	0.074	0.028	0.401	0.165	0.013	0.461	0.179	0.158
134	1584.315	2.875	0.008	0.002	0.052	0.049	0.343	0.893	0.28	0.269
135	2207.456	4.157	0.01	0.002	0.047	0.069	0.123	0.824	0.212	0.202
136	119.033	0.736	0.017	0.006	0.093	0.056	-0.163	0.655	0.208	0.191
137	1484.439	2.874	0.01	0.002	0.06	0.063	0.415	0.772	0.279	0.266
138	1102.611	2.045	0.016	0.002	0.081	0.091	-0.122	0.684	0.243	0.231

Table 9: Data table containing information for crater planform

i	D [m]	δD [m]	σ_R/D	$\Delta\sigma_R/D$	Δ_R/D	$\sigma_R/D^{3/4}$ [m$^{1/4}$]	r_{Rh}	C_R	$d_{\max}/D$	d/D
139	1343.762	2.63	0.017	0.002	0.1	0.105	0.364	0.725	0.239	0.226
140	1053.35	1.958	0.008	0.002	0.052	0.044	0.213	0.795	0.264	0.253
141	1631.965	3.103	0.007	0.002	0.038	0.047	0.327	0.914	0.217	0.205
142	945.743	2.004	0.026	0.002	0.154	0.146	-0.234	0.557	0.187	0.171
143	540.652	1.05	0.029	0.002	0.145	0.141	-0.073	0.549	0.268	0.248
144	62.226	0.654	0.023	0.011	0.137	0.064	0.352	0.566	0.133	0.119
145	1337.089	2.308	0.012	0.002	0.072	0.07	0.126	0.702	0.206	0.192
146	2908.784	5.495	0.016	0.002	0.08	0.114	-0.468	0.828	0.244	0.232
147	508.821	0.938	0.023	0.002	0.184	0.111	-0.25	0.623	0.272	0.246
148	32.834	1.206	0.034	0.037	0.198	0.081	-0.438	0.554	0.143	0.136
149	45.728	1.35	0.036	0.03	0.163	0.094	0.535	0.559	0.159	0.142
150	129.663	1.474	0.015	0.011	0.074	0.052	0.021	0.776	0.239	0.226
151	56.213	1.581	0.025	0.028	0.131	0.069	-0.455	0.632	0.172	0.162
152	57.595	1.597	0.037	0.028	0.229	0.103	0.478	0.609	0.185	0.175
153	469.452	1.543	0.014	0.003	0.082	0.067	-0.125	0.831	0.212	0.184
154	183.275	1.41	0.041	0.008	0.172	0.152	0.68	0.673	0.186	0.149
155	186.713	1.573	0.037	0.008	0.239	0.137	0.172	0.557	0.178	0.161
156	427.905	0.847	0.012	0.002	0.069	0.055	0.396	0.854	0.223	0.206
157	92.966	0.77	0.055	0.008	0.24	0.172	-0.348	0.689	0.217	0.181
158	57.649	0.715	0.013	0.012	0.088	0.035	-0.418	0.712	0.136	0.125
159	40.507	0.812	0.039	0.02	0.2	0.098	-0.441	0.53	0.207	0.182
160	81.074	0.709	0.045	0.009	0.193	0.134	-0.655	0.603	0.175	0.159
161	108.33	0.809	0.03	0.007	0.148	0.097	-0.21	0.624	0.224	0.191
162	132.019	1.408	0.019	0.011	0.111	0.063	0.153	0.586	0.184	0.167
163	171.965	0.769	0.012	0.004	0.087	0.043	0.196	0.611	0.234	0.202
164	51.922	0.666	0.036	0.013	0.227	0.097	-0.267	0.441	0.212	0.189
165	664.467	1.261	0.016	0.002	0.078	0.079	-0.59	0.743	0.178	0.158
166	59.427	1.452	0.072	0.024	0.365	0.199	-0.218	0.484	0.215	0.183
167	69.671	1.557	0.115	0.022	0.552	0.333	0.014	0.212	0.16	0.131
168	37.053	0.757	0.02	0.02	0.094	0.05	-0.292	0.675	0.105	0.096
169	47.907	1.391	0.076	0.029	0.269	0.2	0.455	0.136	0.184	0.157
170	65.653	1.502	0.036	0.023	0.205	0.102	0.733	0.579	0.221	0.207
171	73.098	1.342	0.015	0.018	0.067	0.043	-0.375	0.762	0.209	0.201
172	99.532	1.291	0.022	0.013	0.146	0.07	-0.181	0.683	0.174	0.165
173	80.756	1.543	0.017	0.019	0.092	0.05	-0.431	0.713	0.195	0.186
174	133.94	1.432	0.014	0.011	0.08	0.049	0.354	0.778	0.206	0.187
175	91.678	1.55	0.046	0.017	0.274	0.142	0.094	0.498	0.207	0.185
176	71.873	1.52	0.023	0.021	0.096	0.067	0.459	0.652	0.164	0.156

Table 10: Data table containing information for crater planform

i	D [m]	δD [m]	σ_R/D	$\Delta\sigma_R/D$	Δ_R/D	$\sigma_R/D^{3/4}$ [$m^{1/4}$]	r_{Rh}	C_R	d_{max}/D	d/D
177	59.502	1.639	0.025	0.028	0.139	0.068	-0.31	0.584	0.163	0.153
178	82.147	0.719	0.032	0.009	0.191	0.096	-0.152	0.458	0.23	0.216
179	88.74	0.744	0.042	0.008	0.224	0.129	-0.577	0.574	0.239	0.209
180	28.087	0.803	0.03	0.029	0.133	0.068	0.474	0.624	0.218	0.202
181	67.946	1.552	0.023	0.023	0.107	0.066	0.03	0.713	0.216	0.176
182	2194.262	4.128	0.014	0.002	0.083	0.098	-0.249	0.658	0.203	0.192
183	3136.613	5.792	0.011	0.002	0.056	0.083	0.191	0.897	0.195	0.175
184	2776.471	5.287	0.011	0.002	0.065	0.082	-0.313	0.794	0.203	0.185
185	1756.567	6.37	0.014	0.004	0.156	0.09	-0.481	0.627	0.224	0.215
186	3054.81	11.755	0.008	0.004	0.041	0.057	-0.092	0.822	0.191	0.18
187	770.175	1.504	0.018	0.002	0.109	0.094	0.085	0.648	0.095	0.078
188	1498.915	3.108	0.019	0.002	0.082	0.116	-0.561	0.795	0.065	0.05
189	2354.442	8.425	0.012	0.004	0.076	0.083	0.183	0.754	0.174	0.162
190	1008.484	1.856	0.016	0.002	0.102	0.088	0.1	0.561	0.21	0.174
192	1166.839	2.156	0.014	0.002	0.083	0.08	-0.097	0.606	0.06	0.046
193	646.787	2.169	0.016	0.003	0.086	0.079	-0.251	0.714	0.233	0.222
194	1560.306	5.571	0.027	0.004	0.207	0.168	-0.49	0.533	0.214	0.2
195	1254.746	4.422	0.063	0.004	0.345	0.373	-0.687	0.583	0.219	0.208
197	4178.338	6.366	0.085	0.002	0.355	0.685	0.661	0.538	0.184	0.144
198	280.788	1.161	0.025	0.004	0.115	0.102	0.401	0.758	0.174	0.156

1.3 Profile Shape Metrics

Table 11: Profile Information Table

i	α_C	$\delta\alpha_C$	B_C	δB_C	α_F	$\delta\alpha_F$	B_F	δB_F	ℓ_C [R]	$\delta\ell_C$ [R]	ℓ_F [R]	$\delta\ell_F$ [R]	ℓ [R]	$\delta\ell$ [R]
001	3.254	0.608	0.39	0.058	-0.941	0.469	0.075	0.074	0.099	0.051	0.078	0.054	0.177	0.105
002	3.787	1.071	0.447	0.167	-4.807	1.939	0.053	0.181	0.053	0.054	0.059	0.065	0.111	0.119
003	2.142	0.64	0.429	0.127	-3.265	1.414	0.082	0.143	0.14	0.047	0.097	0.064	0.236	0.111
004	1.776	0.392	0.484	0.07	-1.587	0.352	0.17	0.085	0.082	0.051	0.079	0.059	0.162	0.11
005	2.088	0.817	0.367	0.182	-4.136	1.668	0.066	0.134	0.118	0.055	0.069	0.062	0.187	0.117
006	1.653	0.328	0.505	0.109	-3.848	0.566	0.147	0.051	0.118	0.07	0.059	0.05	0.177	0.12
007	2.046	0.66	0.392	0.188	-2.741	1.316	0.065	0.122	0.111	0.077	0.053	0.045	0.164	0.122
008	1.724	0.563	0.375	0.108	-0.876	0.793	0.058	0.19	0.077	0.054	0.085	0.071	0.162	0.125
010	1.98	0.396	0.388	0.089	-2.781	1.272	0.083	0.23	0.108	0.059	0.097	0.068	0.205	0.126
011	2.286	0.759	0.386	0.052	-0.657	0.437	0.06	0.115	0.072	0.064	0.052	0.064	0.124	0.128
012	2.089	0.477	0.278	0.178	-1.161	0.389	0.11	0.099	0.084	0.055	0.075	0.062	0.159	0.117
013	1.946	0.425	0.376	0.087	-3.035	2.197	0.069	0.158	0.046	0.054	0.035	0.044	0.081	0.098
015	1.994	0.758	0.412	0.128	-3.86	1.474	0.156	0.134	0.111	0.055	0.096	0.056	0.207	0.111
020	1.712	0.57	0.324	0.122	-4.997	3.691	0.049	0.309	0.056	0.054	0.06	0.053	0.116	0.107
021	2.247	0.479	0.644	0.095	-2.435	0.826	0.107	0.13	0.096	0.049	0.079	0.062	0.174	0.112
022	2.512	0.32	0.619	0.098	nan	nan	nan	nan	0.072	0.042	0.064	0.041	0.136	0.083
023	1.406	0.469	0.368	0.106	-5.327	3.361	0.045	0.183	0.049	0.051	0.028	0.03	0.077	0.082
030	2.245	0.577	0.436	0.069	-4.608	1.568	0.082	0.232	0.069	0.062	0.06	0.057	0.13	0.12
031	1.438	0.305	0.366	0.053	-1.16	0.564	0.074	0.133	0.155	0.054	0.092	0.061	0.247	0.115
032	1.722	0.436	0.388	0.094	-1.384	0.352	0.148	0.314	0.107	0.049	0.062	0.048	0.169	0.097
033	1.925	0.372	0.316	0.061	-3.745	1.011	0.096	0.066	0.033	0.041	0.038	0.055	0.071	0.096
034	2.856	1.385	0.352	0.255	-4.35	1.059	0.117	0.077	0.066	0.056	0.033	0.033	0.099	0.09
036	2.021	0.609	0.518	0.114	-1.845	1.48	0.045	0.126	0.068	0.065	0.045	0.042	0.113	0.107
037	2.19	0.911	0.506	0.167	-2.403	0.696	0.051	0.094	0.11	0.057	0.044	0.042	0.154	0.098
038	2.01	0.222	0.48	0.064	-1.988	0.716	0.069	0.135	0.118	0.061	0.085	0.07	0.203	0.131
039	1.799	0.656	0.355	0.11	-4.034	1.279	0.119	0.091	0.044	0.044	0.065	0.057	0.109	0.101
041	2.087	0.863	0.421	0.146	-2.568	1.875	0.056	0.163	0.05	0.045	0.048	0.065	0.098	0.109
042	1.944	0.538	0.374	0.097	nan	nan	nan	nan	0.041	0.049	0.037	0.046	0.078	0.095
043	1.888	0.226	0.471	0.102	-3.706	0.881	0.098	0.114	0.142	0.057	0.12	0.065	0.262	0.121
046	2.224	0.949	0.481	0.163	-2.295	0.772	0.084	0.127	0.126	0.054	0.089	0.066	0.216	0.121
047	1.843	0.278	0.54	0.052	-5.036	1.788	0.092	0.108	0.137	0.057	0.111	0.066	0.248	0.123
048	2.278	1.076	0.267	0.133	-3.615	1.782	0.11	0.147	0.135	0.054	0.096	0.062	0.231	0.116
049	1.772	0.449	0.549	0.058	-1.417	0.244	0.118	0.238	0.12	0.053	0.08	0.056	0.2	0.109
050	1.832	0.321	0.304	0.064	-1.501	0.556	0.085	0.117	0.101	0.048	0.07	0.05	0.171	0.098

Table 12: Profile Information Table

i	α_C	$\delta\alpha_C$	B_C	δB_C	α_F	$\delta\alpha_F$	B_F	δB_F	ℓ_C [R]	$\delta\ell_C$ [R]	ℓ_F [R]	$\delta\ell_F$ [R]	ℓ [R]	$\delta\ell$ [R]
051	1.727	0.666	0.235	0.08	-1.308	0.502	0.077	0.179	0.109	0.056	0.069	0.051	0.178	0.107
052	2.37	0.329	0.712	0.052	-2.68	1.418	0.044	0.235	0.119	0.06	0.106	0.067	0.226	0.127
053	2.215	0.452	0.543	0.069	-3.844	1.376	0.107	0.095	0.085	0.06	0.048	0.04	0.133	0.099
054	1.782	0.208	0.478	0.036	-1.43	0.404	0.176	0.164	0.09	0.043	0.081	0.049	0.172	0.092
055	1.788	0.424	0.279	0.068	-0.965	0.272	0.147	0.202	0.088	0.043	0.055	0.045	0.143	0.088
056	1.843	0.143	0.479	0.041	-1.633	0.692	0.032	0.139	0.106	0.066	0.083	0.075	0.189	0.141
057	1.569	0.328	0.452	0.078	-2.691	0.937	0.111	0.176	0.135	0.059	0.072	0.046	0.207	0.104
058	2.43	1.211	0.345	0.214	-3.478	2.12	0.072	0.158	0.043	0.045	0.057	0.056	0.1	0.101
059	2.513	1.161	0.326	0.205	-4.823	1.801	0.17	0.171	0.095	0.057	0.083	0.056	0.178	0.113
061	2.44	0.858	0.396	0.151	-4.312	1.487	0.126	0.098	0.071	0.049	0.059	0.051	0.13	0.1
062	1.995	0.33	0.485	0.051	nan	nan	nan	nan	0.065	0.054	0.043	0.042	0.108	0.095
063	2.306	0.292	0.477	0.055	-2.508	0.77	0.113	0.122	0.124	0.045	0.109	0.06	0.232	0.105
064	1.893	0.453	0.286	0.081	-1.917	0.999	0.081	0.209	0.107	0.043	0.091	0.056	0.198	0.099
065	1.854	0.742	0.229	0.394	-6.184	1.373	0.125	0.111	0.117	0.073	0.065	0.05	0.182	0.123
066	2.828	1.192	0.447	0.182	-3.613	1.239	0.078	0.161	0.083	0.063	0.089	0.059	0.172	0.121
068	1.723	0.513	0.4	0.055	-1.59	0.3	0.122	0.133	0.126	0.051	0.078	0.046	0.204	0.098
069	2.25	0.6	0.465	0.095	-1.735	0.736	0.079	0.19	0.136	0.062	0.09	0.062	0.226	0.124
070	1.842	0.342	0.379	0.087	-1.677	0.612	0.068	0.135	0.123	0.05	0.085	0.06	0.208	0.111
071	1.98	0.516	0.367	0.085	-1.587	0.72	0.069	0.143	0.127	0.052	0.08	0.053	0.207	0.105
072	1.965	0.884	0.379	0.078	-1.594	0.457	0.111	0.081	0.119	0.05	0.087	0.055	0.206	0.105
073	2.023	0.157	0.203	0.082	-3.91	1.282	0.139	0.129	0.11	0.052	0.104	0.065	0.213	0.117
074	2.55	1.518	0.306	0.275	nan	nan	nan	nan	0.091	0.053	0.078	0.06	0.169	0.113
077	2.339	0.563	0.625	0.136	-4.031	3.151	0.07	0.203	0.137	0.059	0.07	0.057	0.207	0.115
078	1.797	0.429	0.592	0.101	-3.769	0.97	0.082	0.24	0.085	0.068	0.065	0.066	0.15	0.134
079	1.946	0.413	0.389	0.049	-1.612	0.428	0.062	0.095	0.117	0.055	0.08	0.054	0.197	0.109
080	2.238	0.189	0.483	0.046	-3.631	0.657	0.138	0.079	0.095	0.055	0.095	0.065	0.189	0.12
081	2.248	1.058	0.352	0.198	-3.607	1.298	0.06	0.194	0.132	0.057	0.093	0.064	0.225	0.122
083	1.803	0.253	0.387	0.062	-3.206	2.702	0.042	0.14	0.032	0.042	0.05	0.049	0.082	0.091
084	2.242	1.01	0.421	0.185	-2.794	1.539	0.112	0.141	0.096	0.053	0.041	0.023	0.137	0.076
085	1.803	0.197	0.365	0.031	-1.552	0.593	0.079	0.152	0.102	0.038	0.085	0.056	0.187	0.094
086	2.073	0.561	0.472	0.084	-4.051	2.237	0.067	0.13	0.014	0.012	0.025	0.035	0.039	0.047
088	1.844	0.502	0.494	0.084	-2.087	0.701	0.087	0.148	0.116	0.065	0.09	0.067	0.205	0.132
089	1.871	0.431	0.41	0.099	-3.791	1.153	0.055	0.125	0.08	0.066	0.053	0.062	0.133	0.128
090	1.769	0.416	0.438	0.109	-2.408	1.029	0.074	0.17	0.169	0.053	0.107	0.074	0.276	0.127

Table 13: Profile Information Table

i	α_C	$\delta\alpha_C$	B_C	δB_C	α_F	$\delta\alpha_F$	B_F	δB_F	ℓ_C [R]	$\delta\ell_C$ [R]	ℓ_F [R]	$\delta\ell_F$ [R]	ℓ [R]	$\delta\ell$ [R]
091	2.242	0.357	0.669	0.095	-2.248	0.996	0.05	0.15	0.144	0.049	0.086	0.067	0.23	0.116
092	2.425	0.779	0.602	0.138	-2.87	1.755	0.043	0.153	0.071	0.05	0.053	0.062	0.124	0.112
093	2.111	0.516	0.277	0.075	-1.163	0.367	0.058	0.126	0.085	0.043	0.067	0.047	0.152	0.09
094	1.726	0.58	0.412	0.048	-2.213	0.89	0.062	0.164	0.087	0.037	0.058	0.048	0.145	0.085
095	1.589	0.45	0.319	0.095	-3.241	0.678	0.058	0.072	0.023	0.019	0.026	0.031	0.05	0.05
096	1.927	0.142	0.505	0.035	-1.88	0.749	0.04	0.166	0.161	0.051	0.087	0.062	0.249	0.113
097	1.907	0.518	0.293	0.088	-6.269	1.402	0.053	0.103	0.027	0.043	0.025	0.036	0.052	0.079
098	2.4	0.413	0.513	0.086	-4.519	1.071	0.099	0.077	0.096	0.051	0.094	0.051	0.189	0.102
099	2.309	0.653	0.558	0.029	-2.613	0.743	0.104	0.106	0.115	0.051	0.113	0.072	0.228	0.123
103	2.033	0.268	0.676	0.04	-2.318	1.157	0.118	0.286	0.108	0.066	0.072	0.066	0.18	0.132
104	1.966	0.575	0.241	0.186	-4.578	0.589	0.133	0.076	0.085	0.059	0.094	0.064	0.179	0.123
105	1.237	0.233	0.47	0.054	-1.419	0.852	0.089	0.148	0.118	0.049	0.087	0.057	0.205	0.105
106	2.013	0.277	0.508	0.065	-3.422	1.44	0.108	0.156	0.077	0.062	0.048	0.05	0.125	0.111
107	1.57	0.47	0.544	0.083	-1.941	0.604	0.085	0.178	0.125	0.065	0.085	0.067	0.21	0.131
109	2.097	0.315	0.406	0.156	-3.596	1.167	0.12	0.108	0.096	0.069	0.054	0.052	0.15	0.121
110	1.623	0.496	0.349	0.071	-0.589	0.381	0.144	0.054	0.043	0.043	0.064	0.052	0.107	0.095
111	1.836	0.275	0.492	0.049	-3.211	0.933	0.082	0.114	0.097	0.06	0.085	0.056	0.183	0.116
112	2.206	0.199	0.753	0.078	-4.421	0.422	0.106	0.038	0.113	0.062	0.049	0.042	0.162	0.105
113	2.244	0.445	0.547	0.096	-1.77	0.553	0.127	0.136	0.1	0.06	0.079	0.063	0.179	0.124
114	2.186	0.693	0.409	0.168	-2.667	0.939	0.065	0.183	0.097	0.056	0.087	0.063	0.183	0.12
116	1.939	0.344	0.416	0.073	-2.4	1.309	0.092	0.15	0.142	0.056	0.106	0.066	0.248	0.123
117	1.957	0.337	0.526	0.077	-5.447	3.914	0.065	0.226	0.062	0.057	0.043	0.037	0.105	0.095
119	2.203	0.183	0.635	0.058	-3.774	1.471	0.088	0.129	0.133	0.051	0.108	0.065	0.241	0.116
120	1.939	0.368	0.539	0.055	-2.537	0.454	0.103	0.072	0.133	0.06	0.112	0.078	0.245	0.138
121	1.816	1.115	0.259	0.127	-0.882	0.749	0.101	0.163	0.097	0.07	0.069	0.073	0.167	0.143
122	1.826	0.647	0.258	0.152	-2.195	1.103	0.107	0.211	0.102	0.07	0.085	0.069	0.186	0.139
123	1.736	0.486	0.44	0.069	-1.264	0.547	0.171	0.155	0.108	0.061	0.079	0.065	0.187	0.126
124	1.415	0.436	0.266	0.094	-1.134	0.598	0.153	0.15	0.089	0.066	0.058	0.065	0.147	0.13
125	2.314	0.302	0.64	0.051	-4.193	1.754	0.107	0.157	0.114	0.057	0.063	0.045	0.177	0.102
126	1.589	0.538	0.311	0.173	-3.652	1.419	0.085	0.12	0.052	0.05	0.052	0.054	0.104	0.105
127	1.765	0.52	0.315	0.102	-3.751	0.85	0.099	0.091	0.053	0.051	0.039	0.042	0.092	0.093
128	1.97	0.544	0.472	0.131	-3.014	0.906	0.151	0.133	0.135	0.054	0.12	0.063	0.255	0.117
129	2.339	0.707	0.345	0.093	-2.206	0.885	0.332	0.466	0.121	0.059	0.077	0.058	0.198	0.117
130	2.107	0.466	0.501	0.098	-2.511	0.758	0.071	0.09	0.092	0.066	0.056	0.04	0.148	0.106

Table 14: Profile Information Table

i	α_C	$\delta\alpha_C$	B_C	δB_C	α_F	$\delta\alpha_F$	B_F	δB_F	ℓ_C [R]	$\delta\ell_C$ [R]	ℓ_F [R]	$\delta\ell_F$ [R]	ℓ [R]	$\delta\ell$ [R]
131	1.755	0.31	0.653	0.069	-1.582	0.812	0.081	0.168	0.085	0.035	0.063	0.057	0.148	0.092
132	1.917	0.489	0.503	0.09	-1.751	0.479	0.073	0.238	0.132	0.037	0.072	0.053	0.204	0.09
133	1.514	0.305	0.327	0.067	-0.851	0.875	0.184	0.147	0.058	0.052	0.051	0.054	0.11	0.105
134	3.514	0.424	0.508	0.02	-2.896	1.369	0.067	0.149	0.082	0.064	0.043	0.045	0.124	0.109
135	0.0	0.0	1.0	0.0	-3.642	0.457	0.145	0.06	0.11	0.058	0.052	0.048	0.161	0.106
136	1.822	0.277	0.457	0.077	-1.946	0.647	0.136	0.207	0.139	0.049	0.096	0.056	0.235	0.105
137	2.159	0.587	0.674	0.1	-4.222	1.375	0.109	0.124	0.115	0.063	0.078	0.06	0.193	0.122
138	2.54	0.439	0.688	0.118	-3.917	1.288	0.083	0.114	0.11	0.057	0.095	0.056	0.205	0.113
139	1.917	0.675	0.487	0.112	-2.587	0.798	0.083	0.108	0.139	0.06	0.087	0.054	0.226	0.114
140	1.937	0.401	0.609	0.091	-3.157	2.156	0.07	0.135	0.107	0.059	0.073	0.054	0.18	0.113
141	2.134	0.714	0.466	0.124	-2.0	1.135	0.064	0.1	0.076	0.066	0.054	0.055	0.13	0.121
142	2.187	0.704	0.45	0.13	-4.576	1.49	0.057	0.133	0.154	0.05	0.095	0.059	0.249	0.109
143	1.642	0.305	0.52	0.082	-2.662	0.629	0.123	0.177	0.154	0.049	0.125	0.059	0.279	0.108
144	2.242	0.725	0.26	0.108	-1.261	1.322	0.063	0.108	0.111	0.055	0.101	0.076	0.212	0.131
145	0.0	0.0	1.0	0.0	-2.739	1.03	0.127	0.123	0.119	0.051	0.095	0.063	0.214	0.115
146	1.623	0.291	0.378	0.106	-3.612	1.12	0.109	0.101	0.073	0.063	0.048	0.05	0.12	0.113
147	2.785	0.562	0.539	0.038	-3.525	0.795	0.135	0.17	0.155	0.054	0.111	0.064	0.266	0.118
148	1.471	0.303	0.369	0.049	-0.605	0.194	0.107	0.077	0.036	0.042	0.049	0.046	0.085	0.089
149	2.01	1.2	0.363	0.16	-1.168	0.471	0.123	0.17	0.07	0.043	0.071	0.058	0.141	0.101
150	1.915	0.456	0.589	0.096	-1.745	0.679	0.123	0.246	0.096	0.039	0.058	0.039	0.155	0.079
151	2.116	0.675	0.381	0.071	-0.906	0.326	0.086	0.208	0.072	0.051	0.057	0.047	0.129	0.098
152	1.644	0.401	0.459	0.074	-0.948	0.393	0.095	0.163	0.061	0.046	0.058	0.05	0.119	0.096
153	2.253	0.624	0.435	0.056	-3.019	1.098	0.245	0.194	0.103	0.064	0.073	0.047	0.177	0.111
154	1.937	1.144	0.357	0.101	-2.308	0.54	0.095	0.478	0.096	0.051	0.044	0.029	0.139	0.08
155	1.885	0.359	0.353	0.057	-2.468	1.315	0.068	0.198	0.078	0.06	0.057	0.045	0.135	0.105
156	2.191	0.403	0.441	0.113	-3.359	0.89	0.097	0.098	0.138	0.054	0.081	0.061	0.219	0.115
157	1.702	0.86	0.369	0.149	-1.702	0.554	0.114	0.122	0.127	0.057	0.073	0.051	0.2	0.109
158	1.881	0.273	0.273	0.08	-1.16	0.519	0.117	0.084	0.085	0.053	0.084	0.061	0.168	0.114
159	2.559	0.202	0.455	0.073	-1.283	0.577	0.129	0.132	0.06	0.052	0.059	0.054	0.119	0.106
160	1.865	1.174	0.318	0.119	-1.154	0.257	0.091	0.071	0.091	0.062	0.083	0.063	0.174	0.125
161	2.288	0.609	0.53	0.096	-2.194	0.646	0.111	0.158	0.124	0.053	0.102	0.059	0.226	0.112
162	1.928	0.179	0.384	0.067	-2.04	0.772	0.095	0.15	0.123	0.053	0.07	0.046	0.193	0.099
163	1.757	0.183	0.454	0.045	-1.941	0.309	0.212	0.195	0.147	0.05	0.122	0.055	0.269	0.104
164	1.919	0.358	0.482	0.055	-1.182	0.55	0.117	0.2	0.116	0.049	0.103	0.064	0.219	0.113

Table 15: Profile Information Table

i	α_C	$\delta\alpha_C$	B_C	δB_C	α_F	$\delta\alpha_F$	B_F	δB_F	ℓ_C [R]	$\delta\ell_C$ [R]	ℓ_F [R]	$\delta\ell_F$ [R]	ℓ [R]	$\delta\ell$ [R]
165	1.855	0.1	0.338	0.047	-3.033	1.302	0.059	0.179	0.144	0.06	0.114	0.067	0.258	0.126
166	1.745	0.353	0.511	0.064	-1.69	0.589	0.274	0.366	0.074	0.04	0.09	0.061	0.164	0.101
167	1.803	0.576	0.293	0.081	-1.444	0.726	0.162	0.234	0.039	0.039	0.075	0.052	0.115	0.091
168	1.92	0.509	0.185	0.141	-0.793	0.282	0.118	0.065	0.085	0.043	0.096	0.057	0.181	0.1
169	1.786	0.572	0.417	0.11	-0.835	0.578	0.076	0.17	0.071	0.053	0.07	0.066	0.141	0.119
170	1.78	0.215	0.571	0.039	-1.327	0.478	0.095	0.255	0.071	0.032	0.053	0.036	0.124	0.068
171	1.822	0.361	0.556	0.065	-1.464	0.481	0.144	0.205	0.084	0.038	0.065	0.044	0.15	0.082
172	1.524	0.077	0.421	0.042	-2.704	1.264	0.076	0.128	0.107	0.053	0.063	0.048	0.17	0.101
173	2.432	0.59	0.551	0.123	-2.126	0.49	0.059	0.086	0.086	0.037	0.048	0.039	0.135	0.076
174	2.298	0.209	0.597	0.043	-2.205	1.017	0.071	0.15	0.096	0.036	0.053	0.037	0.149	0.073
175	2.325	0.722	0.49	0.133	-1.821	0.633	0.105	0.107	0.084	0.048	0.069	0.061	0.153	0.109
176	1.866	0.557	0.387	0.101	-1.198	0.669	0.071	0.142	0.096	0.04	0.071	0.051	0.168	0.091
177	1.659	0.393	0.373	0.061	-1.35	0.394	0.063	0.088	0.077	0.047	0.063	0.05	0.141	0.097
178	1.976	0.143	0.617	0.047	-1.449	0.62	0.099	0.206	0.115	0.055	0.083	0.07	0.198	0.125
179	2.126	0.715	0.592	0.082	-2.158	0.912	0.066	0.182	0.087	0.059	0.076	0.061	0.163	0.12
180	2.049	0.589	0.599	0.086	-0.759	0.444	0.122	0.086	0.087	0.035	0.093	0.058	0.18	0.092
181	2.391	0.479	0.512	0.064	-1.747	0.594	0.133	0.147	0.088	0.045	0.07	0.046	0.157	0.09
182	1.919	0.23	0.484	0.086	-5.586	1.979	0.056	0.125	0.141	0.054	0.058	0.043	0.199	0.097
183	2.055	0.398	0.407	0.1	nan	nan	nan	nan	0.044	0.049	0.025	0.031	0.069	0.08
184	0.0	0.0	1.0	0.0	-5.016	1.424	0.133	0.089	nan	nan	0.076	0.056	nan	nan
185	2.124	0.589	0.545	0.111	-5.949	1.603	0.078	0.147	0.059	0.049	0.039	0.042	0.097	0.091
186	0.0	0.0	1.0	0.0	-3.712	0.871	0.118	0.042	0.118	0.064	0.03	0.028	0.149	0.092
187	0.0	0.0	1.0	0.0	-3.697	1.373	0.1	0.122	0.131	0.051	0.116	0.062	0.247	0.114
188	0.0	0.0	1.0	0.0	-4.785	1.186	0.117	0.082	0.082	0.005	0.086	0.058	0.168	0.062
189	2.014	0.705	0.355	0.143	-3.742	1.606	0.08	0.11	0.031	0.039	0.038	0.047	0.069	0.086
190	0.0	0.0	1.0	0.0	-5.635	1.205	0.051	0.126	0.151	0.052	0.127	0.063	0.278	0.115
192	0.0	0.0	1.0	0.0	-5.21	1.31	0.135	0.09	0.1	0.038	0.111	0.067	0.211	0.106
193	2.738	0.318	0.601	0.036	-3.188	1.102	0.071	0.127	0.141	0.052	0.079	0.052	0.22	0.103
194	1.765	0.658	0.394	0.116	-2.29	0.667	0.073	0.104	0.113	0.055	0.066	0.055	0.18	0.11
195	2.69	0.448	0.588	0.085	-2.135	0.539	0.07	0.133	0.116	0.065	0.075	0.064	0.192	0.128
197	2.138	1.576	0.305	0.166	-2.133	1.612	0.103	0.123	0.067	0.052	0.077	0.066	0.144	0.118
198	2.743	0.602	0.381	0.048	-2.554	1.385	0.067	0.205	0.11	0.066	0.087	0.063	0.197	0.129

1.4 Crater Profile Metrics

Table 16: Data Table of Metrics Extracted from Profiles

i	ϕ [°]	$\delta\phi$ [°]	$\phi_{\max}$ [°]	V_C [m^3]	r_{ej} [m]
001	19.33	5.689	40.778	6584.116	2.336
002	34.11	2.558	41.109	5430441960.69	1357.737
003	33.728	7.284	54.795	4283895.654	109.714
004	35.445	5.303	57.513	5682.867	6.385
005	26.889	6.541	49.92	15170247.76	208.058
006	37.764	4.192	50.842	135532738.752	382.215
007	32.001	6.375	54.927	167318202.095	358.201
008	29.151	4.246	41.725	39495.4	3.932
010	26.551	6.203	53.284	277579.281	42.818
011	30.046	7.255	45.43	15930.807	0.969
012	26.651	5.95	43.142	7838.566	4.707
013	29.851	4.476	52.86	302076936.981	523.077
015	33.683	4.788	55.346	231263666.212	490.037
020	33.194	5.38	52.693	2062389115.63	999.489
021	36.435	2.03	52.695	1170917.329	57.856
022	38.219	1.972	49.751	8206168209.99	nan
023	33.565	3.236	45.949	654318592.691	714.488
030	37.717	4.165	61.4	2386095095.93	952.62
031	31.645	7.49	57.783	15614.807	6.017
032	28.995	6.281	51.429	4316.27	5.755
033	35.569	2.744	44.349	2591646965.53	1049.926
034	37.862	2.399	53.27	2288715889.71	1027.382
036	36.371	2.448	45.959	172032455.205	214.564
037	38.025	2.463	49.644	66766641.882	193.653
038	32.755	9.88	58.593	212107.913	28.244
039	38.895	4.467	64.33	1979832178.24	990.167
041	36.663	4.139	59.117	7397113112.71	1052.639
042	33.527	3.515	49.231	2729120460.12	nan
043	33.95	4.845	44.535	50552291.648	277.217
046	35.238	5.664	52.516	15342921.544	131.538

Table 17: Data Table of Metrics Extracted from Profiles

i	ϕ [°]	$\delta\phi$ [°]	$\phi_{\max}$ [°]	V_C [m^3]	r_{ej} [m]
047	39.822	2.681	49.579	22128501.392	222.661
048	34.429	6.042	52.344	5596109.905	130.17
049	31.012	6.24	53.296	16445.704	8.159
050	26.675	6.21	59.548	8287.341	6.746
051	23.372	5.984	56.447	20048.62	7.449
052	41.536	7.93	56.286	473344.203	37.349
053	36.852	2.705	46.659	318652448.798	496.772
054	31.644	3.799	47.988	4269.396	5.591
055	23.343	1.639	28.232	771.45	1.594
056	29.516	3.555	46.721	7882225.717	73.479
057	33.534	5.347	57.873	189255140.626	346.657
058	38.808	3.329	53.817	1880992067.08	859.212
059	39.668	2.731	47.553	1435211178.72	906.964
061	35.708	2.104	46.917	1277097712.03	999.203
062	38.167	2.608	51.609	4515508870.86	nan
063	34.549	3.21	51.511	2049547.912	75.378
064	27.007	4.626	47.328	244326.652	28.456
065	30.159	7.782	47.596	699616012.232	716.105
066	33.851	4.024	43.072	17422989.975	183.355
068	29.988	5.917	55.649	21100.323	13.757
069	30.495	6.311	53.117	50368.804	15.331
070	28.344	5.985	61.557	21559.993	10.478
071	28.375	5.08	44.93	23042.527	10.93
072	28.047	5.852	50.209	15275.954	9.294
073	34.064	7.802	56.892	703329.391	80.577
074	32.202	4.06	45.744	1091174994.82	nan
077	37.229	3.707	52.568	327958084.202	524.243
078	34.21	2.744	45.728	123950153.205	284.199
079	27.287	4.738	51.821	56163.824	14.643
080	39.175	2.355	47.531	441851092.366	549.718
081	34.409	5.652	63.696	1086122.408	67.415
083	33.744	2.691	55.065	862525972.929	615.952
084	35.083	3.026	52.665	415327163.336	460.317
085	27.356	2.562	33.145	16494.693	8.944
086	32.634	2.909	37.594	560123367.817	628.774
088	36.843	7.216	72.648	99916.067	23.063
089	28.476	3.557	46.064	1837282347.1	970.178
090	34.666	6.252	53.675	1237594.155	63.789

Table 18: Data Table of Metrics Extracted from Profiles

i	ϕ [°]	$\delta\phi$ [°]	$\phi_{\max}$ [°]	V_C [m^3]	r_{ej} [m]
091	31.576	3.314	42.752	101640081.114	228.052
092	36.795	4.143	53.535	243034821.624	366.249
093	24.558	6.142	54.035	20715.131	6.649
094	31.834	3.251	45.374	105947.009	22.448
095	18.654	1.583	29.552	31099917.77	266.002
096	28.615	4.116	65.701	7868593.957	87.316
097	22.747	1.859	56.463	2446898858.93	1444.906
098	35.871	1.301	40.094	15696724.862	204.434
099	37.601	3.765	56.812	691838.77	49.071
103	36.481	4.372	52.146	398316.066	36.757
104	38.089	4.028	50.915	1787257786.6	869.683
105	32.415	6.005	66.748	9456.393	6.557
106	39.971	3.431	58.164	137908721.639	355.383
107	39.868	5.81	58.313	136338.726	22.606
109	39.082	4.194	58.643	174140481.851	366.96
110	23.55	2.131	27.691	288.801	0.206
111	37.051	2.712	48.938	213377587.777	381.151
112	36.617	2.235	42.213	795665515.966	738.183
113	38.36	5.072	59.163	32507.138	12.279
114	34.092	4.045	51.477	155750.581	25.939
116	34.839	4.048	52.736	457287.937	38.793
117	35.724	3.328	52.042	447195405.286	696.562
119	37.588	3.399	48.982	12234773.063	164.156
120	40.369	4.693	58.785	13998837.283	115.857
121	37.671	8.741	65.627	7603.758	2.161
122	37.481	9.086	62.521	78429.042	20.403
123	43.296	9.268	71.234	4768.785	4.366
124	35.931	8.897	56.964	2042.302	3.074
125	39.939	3.559	52.881	632521810.244	627.137
126	36.813	3.53	46.417	1102402629.17	654.902
127	35.84	1.674	45.416	4484325246.48	1189.812
128	37.855	4.631	60.775	747853.476	60.992
129	34.05	3.343	43.908	13492.424	81.535
130	38.174	3.837	56.487	104103210.553	269.178
131	35.812	3.345	57.065	30909.848	10.204
132	30.089	3.466	45.427	22557.958	11.004
133	33.036	4.715	49.004	451.643	0.803
134	33.427	0.939	35.653	402203906.697	357.72
135	0.0	0.0	0.0	nan	586.486

Table 19: Data Table of Metrics Extracted from Profiles

i	ϕ [°]	$\delta\phi$ [°]	$\phi_{\max}$ [°]	V_C [m^3]	r_{ej} [m]
136	31.0	7.585	49.522	52599.651	18.227
137	40.789	2.812	49.911	207244053.048	430.213
138	38.08	1.688	48.113	71369189.286	306.283
139	38.694	1.918	43.86	116492345.151	275.914
140	39.996	2.723	62.25	65602343.179	253.985
141	36.022	3.921	47.62	190767275.009	257.975
142	31.095	5.407	49.785	35104432.729	285.899
143	38.479	2.554	50.48	7630915.759	113.837
144	26.737	3.916	37.619	11603.395	5.011
145	0.0	0.0	0.0	nan	288.422
146	39.171	2.835	55.412	1605515693.27	768.822
147	39.39	1.623	44.201	8849137.952	132.381
148	23.423	5.782	48.357	745.26	0.366
149	30.921	4.285	41.068	2653.747	3.183
150	34.148	2.828	54.007	87488.231	17.333
151	23.769	7.538	45.438	6919.103	2.213
152	32.514	5.638	48.853	5919.136	2.537
153	27.271	2.012	31.624	2107528.797	109.475
154	31.811	4.724	51.099	277292.994	33.79
155	28.328	5.557	51.964	295416.744	36.723
156	37.404	2.638	47.365	3940112.027	107.807
157	33.933	8.198	68.747	20192.453	12.015
158	24.972	4.389	48.866	3106.68	3.957
159	31.713	4.76	44.183	2652.245	3.366
160	29.608	5.003	53.97	18815.549	5.511
161	36.247	5.85	53.936	49317.83	18.967
162	30.374	8.467	55.248	72837.562	21.353
163	32.381	6.529	61.636	114054.351	26.256
164	30.919	4.979	54.92	5634.542	3.702
165	29.387	3.19	53.865	13606310.446	155.503
166	32.138	6.172	58.116	2763.243	7.607
167	21.788	6.535	40.937	3418.673	7.073
168	22.222	3.28	29.48	840.826	1.017
169	38.211	8.286	63.764	6134.943	1.519
170	36.575	3.483	52.18	11482.511	5.79

Table 20: Data Table of Metrics Extracted from Profiles

i	ϕ [°]	$\delta\phi$ [°]	$\phi_{\max}$ [°]	V_C [m^3]	r_{ej} [m]
171	31.348	3.799	66.269	10485.0	7.581
172	30.46	6.433	64.171	33529.28	21.237
173	34.023	2.871	48.837	37048.685	13.672
174	31.332	2.116	36.549	103392.263	23.572
175	32.601	4.892	46.542	34455.567	12.943
176	27.587	5.812	53.771	15514.06	5.255
177	28.595	6.134	59.136	9945.635	5.407
178	35.338	5.772	60.032	24662.875	8.388
179	34.035	3.251	49.199	51507.911	15.269
180	37.536	4.232	49.933	997.699	0.675
181	37.847	6.809	57.882	9620.82	9.093
182	34.105	4.901	43.988	491812659.473	726.522
183	32.236	4.792	49.642	1454381327.83	nan
184	0.0	0.0	0.0	nan	877.232
185	34.615	3.672	50.718	288756049.503	596.415
186	0.0	0.0	0.0	nan	821.386
187	0.0	0.0	0.0	nan	206.574
188	0.0	0.0	0.0	nan	463.18
189	27.313	4.758	37.449	400178783.744	636.189
190	0.0	0.0	0.0	nan	335.097
192	0.0	0.0	0.0	nan	375.004
193	38.161	4.404	50.844	16873936.736	157.069
194	32.934	8.283	69.647	186473349.993	285.45
195	33.944	4.19	45.633	107532069.882	213.363
197	33.68	3.209	43.126	2518955947.9	709.975
198	26.92	3.95	48.885	952990.966	56.983

1.5 Crater Analysis Metrics

Table 21: Data Table of Analysis Metrics

i	σ_h/d $[d]$	σ_h/D $[D]$	$\Delta h/d$ $[d]$	$\Delta h/D$ $[D]$	$\bar{h}/d$ $[d]$	$\bar{h}/D$ $[D]$
001	0.037	0.004	0.187	0.022	0.164	0.019
002	0.035	0.005	0.201	0.031	0.115	0.018
003	0.08	0.013	0.375	0.06	0.143	0.023
004	0.07	0.011	0.372	0.058	0.448	0.07
005	0.049	0.007	0.329	0.044	0.125	0.017
006	0.014	0.003	0.063	0.012	0.308	0.06
007	0.04	0.005	0.173	0.023	0.426	0.057
008	0.074	0.01	0.277	0.036	0.016	0.002
010	0.056	0.008	0.218	0.03	0.13	0.018
011	0.057	0.009	0.308	0.05	0.016	0.003
012	0.069	0.006	0.384	0.035	0.476	0.044
013	0.037	0.004	0.19	0.022	0.48	0.055
015	0.102	0.015	0.341	0.05	0.368	0.054
020	0.035	0.006	0.171	0.029	0.205	0.035
021	0.027	0.005	0.137	0.025	0.244	0.044
022	0.016	0.003	0.133	0.027	0.161	0.032
023	0.029	0.005	0.149	0.028	0.117	0.022
030	0.038	0.008	0.164	0.033	0.025	0.005
031	0.075	0.009	0.262	0.032	0.241	0.03
032	0.041	0.004	0.164	0.017	0.591	0.063
033	0.024	0.003	0.157	0.021	0.317	0.043
034	0.028	0.004	0.129	0.018	0.365	0.05
036	0.039	0.007	0.19	0.035	0.228	0.042
037	0.026	0.005	0.137	0.027	0.143	0.028
038	0.042	0.006	0.191	0.028	0.248	0.037
039	0.027	0.004	0.158	0.024	0.329	0.049
041	0.028	0.005	0.15	0.025	0.187	0.031
042	0.03	0.005	0.181	0.03	0.209	0.034
043	0.027	0.005	0.141	0.024	0.24	0.041
046	0.034	0.005	0.137	0.021	0.235	0.036

Table 22: Data Table of Analysis Metrics

i	σ_h/d [d]	σ_h/D [D]	$\Delta h/d$ [d]	$\Delta h/D$ [D]	$\bar{h}/d$ [d]	$\bar{h}/D$ [D]
047	0.021	0.004	0.091	0.02	0.131	0.028
048	0.105	0.013	0.465	0.057	0.233	0.028
049	0.027	0.004	0.111	0.017	0.322	0.049
050	0.031	0.004	0.156	0.018	0.084	0.01
051	0.07	0.007	0.416	0.042	0.271	0.027
052	0.033	0.009	0.172	0.045	-0.062	-0.016
053	0.033	0.006	0.156	0.029	0.241	0.045
054	0.047	0.006	0.189	0.025	0.418	0.054
055	0.04	0.003	0.162	0.014	0.405	0.035
056	0.05	0.008	0.303	0.047	0.081	0.012
057	0.033	0.006	0.15	0.026	0.295	0.052
058	0.041	0.006	0.221	0.031	0.391	0.055
059	0.052	0.008	0.227	0.033	0.365	0.053
061	0.051	0.006	0.211	0.024	0.53	0.061
062	0.022	0.004	0.143	0.029	0.21	0.042
063	0.067	0.01	0.283	0.04	0.354	0.05
064	0.076	0.009	0.294	0.034	0.29	0.034
065	0.037	0.006	0.183	0.028	0.511	0.078
066	0.043	0.006	0.199	0.03	0.187	0.028
068	0.043	0.004	0.211	0.019	0.744	0.067
069	0.061	0.008	0.263	0.034	0.283	0.037
070	0.062	0.009	0.263	0.037	0.082	0.012
071	0.05	0.006	0.233	0.027	0.213	0.025
072	0.041	0.005	0.199	0.025	0.321	0.04
073	0.086	0.007	0.585	0.05	0.518	0.044
074	0.043	0.006	0.179	0.025	0.168	0.024
077	0.044	0.008	0.184	0.035	0.197	0.037
078	0.022	0.006	0.107	0.031	-0.191	-0.056
079	0.049	0.006	0.209	0.026	0.201	0.025
080	0.024	0.004	0.114	0.019	0.363	0.06
081	0.046	0.008	0.195	0.032	0.023	0.004
083	0.025	0.004	0.137	0.023	0.173	0.029
084	0.039	0.006	0.16	0.024	0.353	0.052
085	0.023	0.003	0.113	0.015	0.105	0.014
086	0.022	0.004	0.115	0.02	0.189	0.033
088	0.072	0.011	0.31	0.049	0.252	0.04
089	0.04	0.006	0.211	0.032	0.195	0.029
090	0.038	0.006	0.182	0.026	0.181	0.026
091	0.027	0.005	0.164	0.032	0.11	0.021
092	0.066	0.013	0.279	0.053	0.09	0.017
093	0.082	0.007	0.322	0.028	0.217	0.019

Table 23: Data Table of Analysis Metrics

i	σ_h/d [d]	σ_h/D [D]	$\Delta h/d$ [d]	$\Delta h/D$ [D]	$\bar{h}/d$ [d]	$\bar{h}/D$ [D]
094	0.05	0.009	0.236	0.041	0.101	0.018
095	0.024	0.003	0.159	0.018	0.227	0.025
096	0.02	0.003	0.092	0.015	0.118	0.019
097	0.033	0.004	0.189	0.022	0.202	0.023
098	0.015	0.003	0.077	0.013	0.225	0.037
099	0.042	0.008	0.23	0.041	0.264	0.048
103	0.032	0.007	0.139	0.029	0.22	0.046
104	0.037	0.006	0.168	0.026	0.345	0.052
105	0.036	0.006	0.139	0.024	0.145	0.025
106	0.034	0.006	0.152	0.028	0.28	0.051
107	0.052	0.01	0.231	0.044	0.196	0.037
109	0.038	0.007	0.174	0.031	0.274	0.049
110	0.052	0.007	0.227	0.03	0.062	0.008
111	0.024	0.005	0.111	0.021	0.17	0.033
112	0.008	0.002	0.051	0.011	0.195	0.041
113	0.099	0.016	0.4	0.066	0.287	0.048
114	0.05	0.01	0.176	0.036	-0.106	-0.022
116	0.071	0.012	0.315	0.054	0.156	0.027
117	0.021	0.004	0.099	0.018	0.222	0.04
119	0.031	0.006	0.135	0.027	0.169	0.034
120	0.024	0.005	0.126	0.029	0.206	0.047
121	0.088	0.01	0.523	0.06	0.245	0.028
122	0.158	0.021	0.761	0.099	0.345	0.045
123	0.11	0.015	0.575	0.081	0.527	0.074
124	0.199	0.019	1.029	0.099	0.444	0.043
125	0.023	0.005	0.102	0.021	0.207	0.044
126	0.025	0.004	0.128	0.023	0.27	0.048
127	0.017	0.003	0.098	0.015	0.309	0.047
128	0.056	0.009	0.271	0.043	0.384	0.061
129	1.258	0.007	5.042	0.029	28.777	0.167
130	0.029	0.005	0.168	0.027	0.314	0.051
131	0.029	0.006	0.122	0.025	0.087	0.018
132	0.029	0.005	0.15	0.025	0.053	0.009
133	0.078	0.01	0.487	0.063	0.215	0.028
134	0.023	0.005	0.12	0.026	0.216	0.047
135	0.025	0.003	0.128	0.018	0.45	0.063
136	0.062	0.009	0.265	0.037	0.354	0.05
137	0.03	0.007	0.114	0.026	0.178	0.04
138	0.019	0.004	0.117	0.023	0.204	0.039
139	0.026	0.005	0.141	0.025	0.285	0.05
140	0.023	0.005	0.107	0.023	0.195	0.041

Table 24: Data Table of Analysis Metrics

i	σ_h/d $[d]$	σ_h/D $[D]$	$\Delta h/d$ $[d]$	$\Delta h/D$ $[D]$	$\bar{h}/d$ $[d]$	$\bar{h}/D$ $[D]$
141	0.025	0.004	0.129	0.02	0.306	0.048
142	0.038	0.006	0.181	0.027	0.152	0.023
143	0.041	0.008	0.182	0.035	0.274	0.053
144	0.049	0.006	0.277	0.034	-0.026	-0.003
145	0.046	0.006	0.203	0.028	0.392	0.054
146	0.025	0.005	0.127	0.024	0.236	0.044
147	0.046	0.009	0.185	0.035	0.3	0.057
148	0.03	0.004	0.138	0.016	0.164	0.019
149	0.069	0.008	0.326	0.037	0.261	0.029
150	0.035	0.006	0.135	0.024	0.273	0.049
151	0.031	0.004	0.132	0.018	0.207	0.028
152	0.038	0.006	0.296	0.044	0.178	0.026
153	0.09	0.009	0.415	0.041	0.868	0.086
154	0.078	0.011	0.389	0.056	0.028	0.004
155	0.066	0.01	0.264	0.038	0.107	0.016
156	0.054	0.009	0.234	0.038	0.269	0.044
157	0.137	0.016	0.524	0.062	0.533	0.063
158	0.061	0.005	0.283	0.022	0.612	0.048
159	0.089	0.012	0.336	0.046	0.338	0.046
160	0.058	0.007	0.26	0.032	0.298	0.037
161	0.095	0.014	0.339	0.051	0.266	0.04
162	0.058	0.007	0.251	0.032	0.318	0.04
163	0.111	0.014	0.404	0.05	0.641	0.079
164	0.064	0.01	0.245	0.039	0.18	0.029
165	0.044	0.006	0.209	0.03	0.092	0.013
166	0.116	0.012	0.426	0.044	0.782	0.08
167	0.126	0.008	0.653	0.043	0.99	0.065
168	0.074	0.005	0.312	0.02	0.51	0.032
169	0.066	0.012	0.263	0.047	-0.12	-0.022
170	0.059	0.011	0.298	0.054	0.139	0.025
171	0.024	0.003	0.112	0.016	0.383	0.056
172	0.032	0.005	0.145	0.022	0.085	0.013
173	0.019	0.004	0.073	0.015	-0.099	-0.02
174	0.043	0.007	0.168	0.028	0.11	0.018
175	0.055	0.009	0.33	0.051	0.183	0.029
176	0.035	0.005	0.13	0.019	0.054	0.008
177	0.03	0.005	0.118	0.019	-0.042	-0.007

Table 25: Data Table of Analysis Metrics

i	σ_h/d $[d]$	σ_h/D $[D]$	$\Delta h/d$ $[d]$	$\Delta h/D$ $[D]$	$\bar{h}/d$ $[d]$	$\bar{h}/D$ $[D]$
178	0.04	0.008	0.174	0.033	0.149	0.028
179	0.042	0.01	0.221	0.051	-0.094	-0.022
180	0.036	0.007	0.139	0.025	0.103	0.019
181	0.125	0.016	0.485	0.062	0.377	0.048
182	0.024	0.004	0.116	0.02	0.113	0.019
183	0.052	0.008	0.21	0.032	0.18	0.028
184	0.042	0.007	0.194	0.03	0.181	0.028
185	0.024	0.004	0.187	0.035	0.15	0.028
186	0.028	0.004	0.147	0.019	0.406	0.052
187	0.152	0.005	0.751	0.026	1.244	0.044
188	2.033	0.006	8.425	0.025	15.563	0.047
189	0.031	0.004	0.169	0.02	0.381	0.045
190	0.031	0.006	0.148	0.027	-0.101	-0.018
192	0.752	0.005	4.132	0.028	5.9	0.04
193	0.029	0.005	0.119	0.023	0.161	0.031
194	0.035	0.006	0.167	0.027	0.219	0.036
195	0.03	0.005	0.141	0.025	0.181	0.032
197	0.09	0.01	0.388	0.044	0.368	0.042
198	0.061	0.008	0.247	0.031	0.243	0.031

1.6 Crater Harmonics Data

Table 26: Harmonics Data table

i	L_0	L_1	L_2	θ_2	A_2^* [R]	A_3^* [R]	A_4^* [R]	A_5^* [R]	A_6^* [R]
001	2	4	10	55.765	0.014	0.007	0.009	0.007	0.005
002	2	4	5	-159.451	0.016	0.004	0.008	0.005	0.004
003	2	6	4	105.877	0.013	0.007	0.009	0.005	0.009
004	2	3	8	-58.731	0.035	0.016	0.011	0.006	0.009
005	6	3	7	-171.087	0.014	0.018	0.014	0.006	0.031
006	2	3	5	175.281	0.004	0.003	0.002	0.003	0.001
007	5	3	2	170.625	0.004	0.005	0.002	0.008	0.003
008	2	3	6	-71.337	0.125	0.07	0.036	0.033	0.037
010	4	3	2	11.28	0.008	0.008	0.011	0.001	0.003
011	4	2	5	147.335	0.034	0.02	0.047	0.027	0.021
012	2	3	5	-112.568	0.028	0.02	0.003	0.019	0.007
013	2	3	4	-156.825	0.009	0.004	0.004	0.003	0.002
015	2	4	3	-35.067	0.015	0.005	0.009	0.004	0.001
020	2	3	4	58.584	0.011	0.007	0.005	0.002	0.002
021	2	8	4	-15.492	0.012	0.006	0.011	0.002	0.008
022	2	4	6	-45.06	0.006	0.004	0.006	0.003	0.005
023	2	4	3	-140.634	0.008	0.004	0.006	0.003	0.001
030	5	2	6	-161.585	0.036	0.03	0.035	0.037	0.035
031	4	3	2	152.234	0.016	0.021	0.022	0.01	0.003
032	3	2	6	6.576	0.011	0.019	0.004	0.003	0.005
033	2	3	8	-28.32	0.008	0.006	0.002	0.003	0.003
034	2	3	6	66.753	0.01	0.007	0.003	0.002	0.003
036	3	4	5	-146.532	0.005	0.007	0.006	0.005	0.001
037	2	5	3	163.764	0.012	0.007	0.006	0.01	0.0
038	2	3	4	-2.551	0.028	0.014	0.013	0.001	0.01
039	2	3	11	104.31	0.01	0.008	0.001	0.001	0.002
041	3	2	7	106.048	0.01	0.014	0.007	0.006	0.005
042	3	2	5	-107.093	0.011	0.012	0.003	0.004	0.002
043	2	5	4	-50.085	0.008	0.004	0.006	0.007	0.005
046	7	4	3	170.306	0.012	0.013	0.014	0.006	0.009

Table 27: Harmonics Data table

i	L_0	L_1	L_2	θ_2	A_2^* [R]	A_3^* [R]	A_4^* [R]	A_5^* [R]	A_6^* [R]
047	5	7	4	49.221	0.001	0.0	0.003	0.005	0.002
048	3	4	2	-177.935	0.013	0.023	0.015	0.006	0.006
049	2	4	3	-155.575	0.025	0.008	0.013	0.005	0.001
050	4	5	2	-52.299	0.011	0.004	0.018	0.015	0.006
051	2	4	5	-2.98	0.02	0.013	0.019	0.014	0.005
052	3	4	5	-33.497	0.017	0.042	0.034	0.027	0.02
053	3	2	15	-132.706	0.008	0.009	0.001	0.003	0.001
054	2	5	6	-0.22	0.011	0.003	0.005	0.007	0.005
055	2	5	9	-13.179	0.017	0.006	0.003	0.009	0.001
056	2	3	6	-157.087	0.053	0.036	0.012	0.014	0.027
057	2	3	4	63.34	0.012	0.01	0.006	0.003	0.005
058	2	4	5	62.852	0.013	0.003	0.007	0.007	0.004
059	2	6	3	-85.157	0.012	0.005	0.004	0.001	0.006
061	2	3	5	-42.773	0.021	0.011	0.004	0.006	0.003
062	3	13	7	92.66	0.001	0.005	0.0	0.001	0.002
063	4	5	6	37.88	0.002	0.01	0.02	0.015	0.011
064	3	9	8	-42.186	0.002	0.017	0.001	0.006	0.007
065	2	4	3	44.226	0.017	0.005	0.009	0.003	0.002
066	2	5	3	-69.778	0.021	0.01	0.007	0.012	0.004
068	3	2	4	-96.918	0.019	0.023	0.011	0.008	0.008
069	4	2	7	91.605	0.017	0.011	0.021	0.005	0.003
070	3	2	5	150.017	0.017	0.024	0.009	0.012	0.008
071	3	2	6	62.785	0.025	0.028	0.008	0.005	0.014
072	2	5	6	86.625	0.021	0.011	0.012	0.017	0.015
073	2	5	11	105.735	0.014	0.003	0.004	0.009	0.005
074	2	3	4	-1.069	0.012	0.011	0.005	0.001	0.002
077	3	4	2	-63.362	0.005	0.012	0.009	0.002	0.002
078	5	4	2	-73.756	0.01	0.01	0.013	0.015	0.01
079	2	3	6	142.921	0.019	0.015	0.003	0.007	0.008
080	2	4	6	65.975	0.01	0.002	0.009	0.001	0.005
081	3	7	4	-103.204	0.01	0.021	0.012	0.005	0.003
083	3	5	2	-131.573	0.004	0.007	0.002	0.005	0.003
084	2	7	9	-47.442	0.01	0.002	0.003	0.004	0.002
085	3	6	2	-111.66	0.007	0.014	0.006	0.004	0.011
086	7	3	4	-92.741	0.002	0.002	0.002	0.001	0.001
088	3	2	8	-164.705	0.017	0.019	0.007	0.014	0.012
089	3	4	7	51.465	0.003	0.011	0.009	0.004	0.005
090	2	4	3	-166.148	0.02	0.016	0.017	0.006	0.006
091	2	9	10	-19.518	0.014	0.008	0.008	0.006	0.008
092	4	2	6	44.422	0.01	0.003	0.015	0.003	0.009
093	3	5	2	38.074	0.017	0.026	0.005	0.019	0.009

Table 28: Harmonics Data table

i	L_0	L_1	L_2	θ_2	A_2^* [R]	A_3^* [R]	A_4^* [R]	A_5^* [R]	A_6^* [R]
094	2	5	7	23.814	0.02	0.009	0.01	0.012	0.004
095	4	2	3	71.224	0.005	0.004	0.006	0.003	0.003
096	6	5	3	-66.04	0.007	0.014	0.005	0.014	0.017
097	2	3	6	-18.997	0.009	0.007	0.004	0.001	0.005
098	2	4	14	-86.145	0.01	0.004	0.007	0.004	0.002
099	3	2	4	63.962	0.01	0.013	0.008	0.005	0.002
103	2	4	3	91.192	0.039	0.009	0.011	0.004	0.003
104	4	5	8	-129.885	0.003	0.002	0.003	0.003	0.0
105	3	2	7	7.266	0.008	0.01	0.004	0.0	0.004
106	2	3	4	-114.624	0.011	0.01	0.006	0.004	0.002
107	3	4	8	-11.719	0.018	0.043	0.03	0.021	0.022
109	2	3	5	34.01	0.015	0.01	0.004	0.006	0.002
110	2	4	3	-30.004	0.045	0.012	0.025	0.008	0.004
111	3	6	4	48.973	0.007	0.012	0.007	0.006	0.007
112	2	3	7	-11.843	0.008	0.006	0.002	0.002	0.002
113	3	4	2	43.385	0.017	0.024	0.018	0.012	0.001
114	4	2	7	-18.725	0.011	0.003	0.014	0.008	0.007
116	2	11	4	-18.667	0.017	0.005	0.007	0.003	0.003
117	4	2	6	-142.793	0.004	0.004	0.005	0.004	0.004
119	4	3	5	-96.464	0.003	0.008	0.015	0.008	0.002
120	6	7	5	-116.068	0.003	0.001	0.001	0.006	0.008
121	2	3	5	-25.118	0.087	0.041	0.022	0.027	0.011
122	3	4	6	-55.331	0.009	0.043	0.022	0.008	0.021
123	5	4	6	-5.626	0.005	0.008	0.033	0.042	0.032
124	2	14	3	79.441	0.054	0.019	0.009	0.012	0.014
125	4	3	2	62.586	0.004	0.006	0.006	0.004	0.002
126	5	3	6	68.241	0.002	0.005	0.002	0.006	0.003
127	2	3	4	55.766	0.008	0.003	0.002	0.002	0.002
128	3	2	5	144.706	0.014	0.015	0.006	0.01	0.006
129	2	7	5	-69.208	0.013	0.005	0.003	0.007	0.002
130	4	3	6	-21.137	0.004	0.005	0.006	0.001	0.004
131	2	3	6	-36.024	0.029	0.017	0.003	0.011	0.015
132	3	7	19.0	-159.12	0.001	0.008	0.006	0.002	0.003
133	3	2	4	-142.993	0.057	0.062	0.041	0.014	0.013
134	3	4	7	3.207	0.001	0.006	0.005	0.002	0.003
135	2	4	3	-37.012	0.009	0.005	0.005	0.004	0.002
136	3	2	10	-175.049	0.009	0.011	0.004	0.003	0.001
137	2	4	5	-28.492	0.008	0.002	0.004	0.003	0.003
138	2	4	6	-1383	0.012	0.006	0.011	0.005	0.007
139	2	3	4	-74.022	0.014	0.012	0.009	0.008	0.003
140	6	3	27	-121.531	0.002	0.003	0.002	0.002	0.003

Table 29: Harmonics Data table

i	L_0	L_1	L_2	θ_2	A_2^* [R]	A_3^* [R]	A_4^* [R]	A_5^* [R]	A_6^* [R]
141	5	4	7	136.64	0.002	0.002	0.005	0.005	0.001
142	5	6	3	-96.857	0.01	0.011	0.002	0.013	0.012
143	3	4	6	47.413	0.013	0.02	0.018	0.01	0.016
144	2	5	7	-101.717	0.022	0.005	0.003	0.011	0.002
145	5	12	6	148.536	0.003	0.004	0.004	0.006	0.004
146	4	3	5	-167.576	0.005	0.011	0.011	0.007	0.001
147	4	6	3	109.668	0.011	0.011	0.017	0.009	0.013
148	4	2	5	-54.716	0.019	0.009	0.02	0.017	0.012
149	2	5	3	-84.029	0.026	0.018	0.014	0.025	0.008
150	4	3	7	55.202	0.006	0.008	0.012	0.005	0.005
151	2	4	6	-11.651	0.025	0.004	0.012	0.009	0.011
152	2	3	4	-6.071	0.031	0.024	0.015	0.004	0.012
153	3	2	5	80.039	0.009	0.013	0.003	0.008	0.003
154	2	3	4	-169.254	0.04	0.038	0.016	0.011	0.003
155	3	4	9	45.661	0.01	0.023	0.017	0.005	0.01
156	2	6	4	-56.061	0.009	0.003	0.006	0.006	0.006
157	2	4	5	5.746	0.067	0.009	0.027	0.018	0.01
158	2	7	5	49.344	0.009	0.003	0.005	0.006	0.001
159	2	3	6	40.868	0.046	0.012	0.007	0.005	0.007
160	2	5	4	146.428	0.056	0.008	0.008	0.017	0.002
161	3	4	2	27.285	0.015	0.021	0.021	0.007	0.009
162	4	2	11	59.466	0.012	0.004	0.014	0.003	0.004
163	2	3	11	94.677	0.009	0.006	0.002	0.002	0.002
164	2	4	5	-36.631	0.028	0.002	0.024	0.018	0.01
165	2	3	5	6.606	0.017	0.007	0.004	0.005	0.002
166	2	3	4	-55.764	0.062	0.05	0.029	0.022	0.022
167	3	4	2	-51.792	0.068	0.093	0.081	0.049	0.015
168	2	5	6	48.56	0.017	0.005	0.005	0.011	0.01
169	2	3	5	71.064	0.058	0.044	0.025	0.036	0.01
170	2	5	3	176.257	0.02	0.014	0.013	0.017	0.013
171	2	3	9	116.738	0.011	0.007	0.006	0.005	0.004
172	2	5	6	34.972	0.018	0.008	0.002	0.011	0.009
173	2	3	4	-54.239	0.013	0.013	0.006	0.002	0.002
174	2	3	5	104.079	0.013	0.008	0.0	0.005	0.004
175	3	2	6	94.497	0.023	0.041	0.015	0.011	0.017

Table 30: Harmonics Data table

i	L_0	L_1	L_2	θ_2	A_2^* [R]	A_3^* [R]	A_4^* [R]	A_5^* [R]	A_6^* [R]
176	2	6	4	-4.232	0.026	0.006	0.008	0.007	0.008
177	2	5	3	-44.688	0.018	0.01	0.01	0.013	0.004
178	4	5	7	1.202	0.014	0.012	0.017	0.016	0.006
179	2	3	5	0.845	0.049	0.018	0.012	0.013	0.007
180	2	4	6	30.386	0.031	0.008	0.013	0.009	0.01
181	3	2	4	-124.15	0.018	0.019	0.011	0.005	0.006
182	2	7	5	-79.583	0.011	0.004	0.004	0.006	0.001
183	2	4	7	-171.212	0.01	0.002	0.006	0.001	0.006
184	2	5	3	-164.576	0.008	0.006	0.004	0.007	0.003
185	2	17	7	-115.394	0.006	0.002	0.001	0.001	0.004
186	2	3	7	-52.012	0.006	0.005	0.002	0.003	0.002
187	2	3	5	-83.046	0.013	0.01	0.004	0.009	0.004
188	2	4	5	28.948	0.024	0.001	0.006	0.004	0.003
189	2	3	8	-169.51	0.007	0.006	0.003	0.004	0.004
190	2	3	6	35.075	0.01	0.007	0.002	0.006	0.006
192	2	3	8	33.437	0.012	0.007	0.003	0.003	0.003
193	3	5	2	171.448	0.008	0.009	0.006	0.009	0.003
194	2	3	4	-88.136	0.019	0.012	0.009	0.008	0.006
195	2	3	4	-109.22	0.047	0.039	0.038	0.029	0.027
197	3	4	2	104.656	0.059	0.072	0.059	0.005	0.032
198	3	2	4	77.522	0.017	0.017	0.017	0.013	0.005

2 Input Parameters

Profile Extractor

```
++ value   # pname                                # psymb
** Value   # Parameter Name                      # Parameter Symbol
./craters/ # Path to crater pickles              # basepath
pltext.txt # savefile                            # svf
plgolb.pkl # golbase save name                   # gbsvname
2.         # Profile Length                      # $P_R$
512        # Number of Radial Profiles           # N
0.1        # SB fit length                       # $\lambda_{SB}$
0.1        # LM Length                           # $\lambda_{LM}$
0.05       # SB minimum slope change             # $\Delta \theta$
0.05       # SB Min Segment Length               # $\chi$
0.1        # CRT Max Radial Discontinuity        # $D_{rad}$
0.05       # CRT Max Radial IP Distance          # $D_{int}$
2.         # CRT Max Angular Discontinuity       # $D_{ang}$
True       # Sift Redundant                      # sift
True       # Continue Looping                    # contloop
1          # Profile Spline order                # prorder
True       # conduct analysis?                   # analysis
eps        # addition atribute flag              # flag
0.25       # flag threshold                      # flcritical
5          # maximum ejecta-bowl center offset   # maxejoff
0.9        # Max %NaN in ejecta circle detrend   # NaNEjMax
3.25       # Min bowl DEM dimensions [R]         # bdim
3.0,3.5    # Ejecta detrending circles           # Ejdtcirc
rim        # Detrending method                   # detrend
Diagnostic # Image type, diagnostics or paper    # images
False      # Find the contour planform           # contour
True       # Run in verbose mode                 # verbose
0,45,90,135,180,\
225,270,315# Start Angles for CRTraces           # $\theta_{st}$
++
```

Profile Extractor

```
++ value    # pname                                  # psymb
** Value    # Parameter Name                         # Parameter Symbol
.           # golbase path                           # gbpath
./craters/  # File path to crater pickles            # pklpath
rim+prf.pkl # Save name of the final golbase         # gbsavename
plgolb      # Save name of Rim Only golbase          # gbsvname
0.1         # Max-Slope fit length                   # $\lambda_{mx}$
0.5         # Rim Roundness Profile Length           # $\lambda_{rr}$
1           # Spline interp order                    # order
3           # Length of ejecta profiles              # ejR
0.125,0.375 # Location to take sample profiles       # samples
50          # Max %NaN before skipping powerlaw fit # fitnan
100         # Max %NaN before skipping profile       # skipnan
0.5         # %Slope to calculate  RimRoundness      #rrmWPer
3.0,3.5     # Ejecta detrending circles              # Ejdtcirc
0.25        # min usable flank data before skipping # ejskp
True        # Use a stretched profile                # stretch
False       # Start profiles at crater nadir         # zNadstart
Diagnostic  # which type of figure to create         # figures
ALL         # which kind of profile to create        # wchprofiles
++
```

3 Source Code

3.1 Select Feature from HiRISE Image

```
# ============================================================================
#
# 20120802-1224-gpucoo - select_features_nemp.py
# Feature dimensions in NEMPed jpgs / Lynn Geiger & W. A. Watters Farfan
# User clicks cater center, rim, ejecta in image, Coords written in text file
#
# <inputs>
#
# Import folder (basepath), and save path is at begining of code.  Also takes a
# "pairings" file: a list of stereo observation IDs formatted as follows: e.g.,
#
# ::
# obsid1, obsid2
# ESP_013639_1995, ESP_012795_1995
# ESP_020875_1490, ESP_020598_1490
# ESP_012511_1820, ESP_012234_1820
# ::
#
# </inputs>
#
# <products>
# Generates one textfile for all features selected in all image </product>
# ============================================================================

import pdb
import pylab as pl
import numpy as np
import os, re
import select_image_features as scfi
import subprocess as sp


# :: User presets ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

# >> Launch Eye of Gnome viewer to explore 25% thumbnail each time?
eogExplore = True

# >> Dir with jpeg version of stereo pair partners:
basepath = './jpgs/'

# >> Path to directory where input file pairings list is stored.
filebase = './'
```

```
# >> File containing list of stereo pairs:
listfile = filebase + 'wasp_tests-00_pairings.txt'

# >> If you are continuing use this stereo list file
#listfile = './continue_stereopairs.txt'

# >> Program output (crater positions) are stored in this file:
savepath = './wasp_tests-00_localize.txt'

# :: Initializations and setup ::::::::::::::::::::::::::::::::::::::::::::::::

# >> Open textfile, a=append, writes at bottom of preexisting text
textfile=open(savepath,'a')

# >> Create column headers only if file is empty

if os.stat(savepath).st_size == 0:
    textfile.write('obsid, fnum, dimx, dimy, x0, y0, r1, r2 \n')

dirlist = np.array(os.listdir(basepath))   # >> must be an array! [120911]
filemid = '_REDmosaic.map.norm.equ.cub_'

pairlist = np.loadtxt(listfile,dtype='str',delimiter=',')

pl.ion()  # >> Run interactive mode (no pl.show() figure blocking) [120911]

# >> Set Up variables

thumb625=[]   # 6.25%  thumb
thumb1250=[]  # 12.5%  thumb
thumb2500=[]  # 25%    thumb

fullim=[]     # actual size jpg
fileID=[]     # name of files
resolution=[] # image resolutions

f0625=[0,0]; f1250=[0,0]; f2500=[0,0]; ffull=[0,0]

ID=[0,0]

th625=[0,0]; th1250=[0,0]; fullds=[0,0]

th625jpg=[0,0]
```

```
# :: Populate the file list ::::::::::::::::::::::::::::::::::::::::::::::::::

# !! Ingests the pairings list, steps through line-by-line.

head = 0

for row in pairlist:

    if row[0] == 'obsid1' : # >> Skip first line if it has column labels
        head = 1
        pass

    else :

        # >> When parsing the observation IDs, first remove leading and trailing
        # >> whitespace characters [120911]

        string_tmp = re.sub(r'^\s*','', row[0], re.M)
        string_tmp = re.sub(r'\s*$','', string_tmp)

        ID[0]=string_tmp

        string_tmp = re.sub(r'^\s*','', row[1], re.M)
        string_tmp = re.sub(r'\s*$','', string_tmp)

        ID[1]=string_tmp

        f0625[0]=ID[0]+filemid+'00625.jpg'
        f0625[1]=ID[1]+filemid+'00625.jpg'

        f1250[0]=ID[0]+filemid+'01250.jpg'
        f1250[1]=ID[1]+filemid+'01250.jpg'

        f2500[0]=ID[0]+filemid+'02500.jpg'
        f2500[1]=ID[1]+filemid+'02500.jpg'

        if ID[0]+filemid+'10000.jpg' in dirlist:
            ffull[0]=ID[0]+filemid+'10000.jpg'
            ffull[1]=ID[1]+filemid+'10000.jpg'
            imres = 1

        elif ID[0]+filemid+'05000.jpg' in dirlist:
            ffull[0]=ID[0]+filemid+'05000.jpg'
            ffull[1]=ID[1]+filemid+'05000.jpg'
            imres = 0.50
```

```
        thumb625.append(f0625[:])
        thumb1250.append(f1250[:])
        thumb2500.append(f2500[:])

        fullim.append(ffull[:])
        fileID.append(ID[:])
        resolution.append(imres)

# >> Go through each set

for i in range(0,len(thumb625),1):

    # >> Set up files

    th625[0],dim,uplf,pdim=scfi.open_image(thumb625[i][0],basepath=basepath)

    th1250[0],dim1250a,uplf,pdim=\
        scfi.open_image(thumb1250[i][0],basepath=basepath)

    fullds[0],fulldim,uplf,pdim=scfi.open_image(fullim[i][0],basepath=basepath)
    th625[1],dim,uplf,pdim=scfi.open_image(thumb625[i][1],basepath=basepath)

    th1250[1],dim1250b,uplf,pdim=scfi.open_image(thumb1250[i][1],\
                                                    basepath=basepath)

    fullds[1],fulldim,uplf,pdim=scfi.open_image(fullim[i][1],basepath=basepath)

    # !! Check that partners are the same size! [waw][130101]
    # >> (otherwise, partners in multiply-acquired pairs can get confused)

    if dim1250a != dim1250b :
        print('!! Partner images are not the same size !! ')
        textfile.close()
        exit(1)

    else: dim1250 = dim1250a

    # >> Correct file dim for the maximum resolution

    fulldim[0] = fulldim[0] / resolution[i]
    fulldim[1] = fulldim[1] / resolution[i]

    # >> Read in both 6.25% jpgs
```

```
    th625jpg[0] = scfi.get_data(th625[0])
    th625jpg[1] = scfi.get_data(th625[1])

    # >> Set up Variables

    num=0  # >> Number of features

# :: Loop over craters picked in image pair ::::::::::::::::::::::::::::::::::


    # >> While loop so you can pick multiple features in each image

    more=True

    # >> Set up a base figure

    figall = scfi.show_image(th625jpg[0],size='asdf',fignum=10)

    while more==True:

# .. Loop over partners ..................................................

        # >> This will open the 25% image in Eye-of-Gnome for "exploration"
        # >> [waw][130101]

        if eogExplore :
            p1 = sp.Popen(["eog", basepath + thumb2500[i][0]])
            p2 = sp.Popen(["eog", basepath + thumb2500[i][1]])

# >> For loop goes through each image in the pair

        for t in range(0,2,1):

        # >> Set up variables here so they clear with new images, but remain
        # >> lists and work with the select_image_feature functions

            featpnts = []
            innerR = []
            outerR = []

            # >> Show 6.25% jpeg

            print('>>>> Currently processing ' + fullim[i][t] + '...')

            pic=scfi.show_image(th625jpg[t],fignum=1)
```

```
# >> If file holds huge feature, say no: pick center rim ejecta from
# >> thumbnail

zoom=raw_input('Zoom In? Continue? or Skip? (z/l, [c/o], s) : ')

cxy625 = None

if zoom=='s' : # >> if image should be skipped, break this loop

    pl.close('all')

    break;

elif zoom =='y' or zoom == 'l':

    pl.figure(pic.number)

    print '>>>> Click on Feature'
    cxy625=pl.ginput(n=1,timeout=1e7)

    #>> Zoom to 12.5% : pick uplft & lwrt corners of zoombox

    pl.close()

    # >> Make sure automatic zoom square doesn't go off the edge

    xoff = int(cxy625[0][0]*2-500)

    if xoff < 0:
        xoff = 0

    if xoff+1000 > dim1250[0]:
        xoff = dim1250[0]-1000

    yoff = int(cxy625[0][1]*2-500)

    if yoff <0:
        yoff =0

    if yoff+1000 > dim1250[1]:
        yoff = dim1250[1]-1000

    # >> Zoom into 12.5% thumb at the feature.  Pick uplf and lwrt
    # >> of desired zoom DOUBLE CLICK TO STOP
```

```
    print '>>>> Click upper-left & lower-right of desired zoombox.'
    print '>>>> DOUBLE CLICK TO STOP'

    th1250jpg=scfi.get_data(th1250[t],xoff,yoff,1000,1000)

    zoom1=scfi.show_image(th1250jpg,fignum=1)

    xy=pl.ginput(n=2,timeout=1e7)
    pl.close(zoom1)

    if xy[0][0]==xy[1][0]:

        more==False
        break

    # >> Zoom into full size image

    UL = xy[0]
    LR = xy[1]

    ULX = int((UL[0] + xoff) /.125*resolution[i])
    LRX = int((LR[0] + xoff) /.125*resolution[i])
    ULY = int((UL[1] + yoff) /.125*resolution[i])
    LRY = int((LR[1] + yoff) /.125*resolution[i])

    cols = abs(ULX - LRX)
    rows = abs(ULY - LRY)

    # >> Show in only desired segment of the image and dtm and pick
    # >> features

    temppoints, tempR1, tempR2 = scfi.pick_features(fullds[t], ULX,\
                  ULY, rows, cols, multi = False)

    if temppoints==[]:
        break

    featpnts.append(temppoints[0] / resolution[i])
    innerR.append(tempR1[0] / resolution[i])
    outerR.append(tempR2[0] / resolution[i])

else: # >> i.e., if "continue w/out zoom is requested"...

    pl.close(pic)
```

```
        temppoints, tempR1, tempR2 = scfi.pick_features(th625[t], \
                   multi = False, pause=True)

        if temppoints==[]:
            break

        featpnts.append(temppoints[0]/.0625)
        innerR.append(tempR1[0]/.0625)
        outerR.append(tempR2[0]/.0625)

    # >> Write data into file

    scfi.write_feature_textfile(featpnts, num, innerR, outerR, \
                                    fileID[i][t], textfile, fulldim)
    if t == 0:

       # >> Show partner to get coords for same feature

        if cxy625 is not None :

            pl.figure(figall.number)
            pl.plot(cxy625[0][0], cxy625[0][1], 'o')
            pl.draw()

            print '>>>> Choose same feature on the stereo partner...'


# >> Option to look for more features in image

if zoom != 's' :  # >> unless we have alread decided to skip it...

    m = raw_input('More in this image pair? ([y]/n) : ')

    if m == 'n' or m == 'o':
        more = False

    # >> increase the feature number of the image

    num += len(featpnts)

else :

    more = False
```

```
    pl.close(figall)

    # >> Continue to next stereopair?

    cont = raw_input('Continue to Next Pair? ([y]/n) : ')

    if cont == 'n' or cont == 'o':

        # >> Option to not start completely over.

        sspot = raw_input('Want to save your place? (y/[n]) : ')

    # >> Create a textfile for the remaining images
        if sspot == 'y' or sspot == 'l':
            pairsleft = pairlist[1+head+i :] # Plus one for header line [lmg]
            contfile = open('./continue_stereopairs.txt','w')
            contfile.write('obsid1, obsid2\n')

            for row in pairsleft:
                contfile.write(row[0]+', '+row[1]+'\n')

            contfile.close()

            print ('>>> Change Pair List File to continue_stereopairs.txt ' +\
                'to pick up where you left off')

        else: print('>>>> Clear positions file if you want to start over!')
        break
    else:
        if eogExplore: p1.terminate(); p2.terminate();

# >> Close File

textfile.close()
```

3.2 Select Image Feature

```
# ============================================================================
#
# 20120802-1224-gpudoo - select_image_features.py
# Select features using GDAL / Lynn Geiger & Wesley Andres Watters Farfan
# User clicks cater center, rim & ejecta from image, pickles or coords returned
#
# <methods>
#
# * open_image(imagefile,basepath='') - Opens dataset using gdal
#
# * get_data(dataset,xoffset,yoffset,columns,rows) - Reads in data by chunk
#
# * show_image(data,datatype='image') - Opens large figure window of gdal data.
#
# * pick_features(ds,xoff,yoff,rows,featurePoint,datatype) - pick features in
#   image Opens image and waits for 3mouse click:center,rim,ejecta.doubleclick
#   to stop
#
# * make_feature_pickle(drape,dtm,basepath,savepath,slices) - makes gdmp pkl
#   Creates a pickle file using the previous functions, shows image in slices
#
# * save_feature_textfile(featurePoints, imagename, file) - Writes in textfile
#
# </methods>
#
# <products>
#
# Generates pickle files or coord text file for features selected. </products>
#
# ============================================================================

import pdb
import pylab as pl
import numpy as np
import golgis as gg
import pickle
from osgeo import gdal, gdalconst

# -- Open image data ---------------------------------------------------------
#
# >> Opens up the specified file into gdal
#
# <inputs>
#
```

```
# image = image file name; basepath = file location
# </inputs>
#
# <products>
#
# ds= data set; dims= dimensions of dataset (x,y); uplf= Upper Left (x,y);
# pixdim= [pixel width, pixel height] in m. </products>
#
# ------------------------------------------------------------------------

def open_image(image,basepath=''):

# >> Retrieve data using gdal
    ds=gdal.Open(basepath+ image)
    band=ds.GetRasterBand(1) #Get image raster data
    bands=ds.RasterCount     #Number of bands
    rows=ds.RasterYSize
    cols=ds.RasterXSize

# >> Retreive Geo spacial data
    transform=ds.GetGeoTransform()
    uplfx=transform[0]
    uplfy=transform[3]
    pixWidth=transform[1]
    pixHeight=transform[5]

# >> Put into useful variables
    dims=[cols,rows]
    uplf=[uplfx, uplfy]
    pixdim=[abs(pixWidth), abs(pixHeight)] # >> GDAL can have negative pixel dim

    return ds, dims, uplf, pixdim



# -- Read in of dataset data ----------------------------------------------
#
# >> Reads in dataset section
#
# <inputs>
#
# ds=dataset (get from 'open_image'); xoff= x offset of desired data from origin
# in pixels; yoff=y offset in pixels; rows=number of rows wanted; cols= columns
# </inputs>
#
```

```
#
# <products>
#
# data= chunk of data read in </products>
#
#
# ------------------------------------------------------------------------

def get_data(ds,xoff=0,yoff=0,rows=[],cols=[]):

    if not rows:
        rows=ds.RasterYSize
    if not cols:
        cols=ds.RasterXSize

# >> Read in desired data

    band=ds.GetRasterBand(1)
    data=band.ReadAsArray(xoff,yoff,cols,rows)
    inds = pl.nonzero(data < -1e3) # >> NaN value= -32768 in some files
    data=pl.asarray(data,float)  # >> Second argument specifies data type
    data[inds] =pl.nan # >> Won't let you do pl.nan unless datatype=float

    return data


# -- Display data as image ------------------------------------------------

# >> Shows file in a large figure window
#
# <inputs>
#
# data= data to be displayed; datatype= how to display the data, for 'image' the
# colormap will be greyscale, for 'dtm' colormap=rainbow and colorbar is added
# Can also specify the figure number to draw to. [120814][lmg]
# Size of figure window = 'Big', 'tiny', or other. Default is 'Big' [120814]
# </inputs>
#
#
# <products>
#
# Image figure window id </products>
# ------------------------------------------------------------------------

def show_image(data,fignum=None,datatype='image',size='Big'):
```

```
# >> Increase the figure size to make feature identification easier
    if size==’Big’:
        fig=pl.figure(num=fignum,figsize=(11,11))
        fig.subplots_adjust(bottom=0.01,left=0.01,top=.99,right=.99)
    elif size==’tiny’:
        fig=pl.figure(num=fignum,figsize=(4,3))
        fig.subplots_adjust(bottom=0.01,left=0.01,top=.99,right=.99)
    else:fig=pl.figure(num=fignum)

    if datatype==’image’:
        colormap=pl.cm.gray

    elif datatype==’dtm’:
        colormap=pl.cm.rainbow

    pl.imshow(data, cmap=colormap)

    if datatype==’dtm’:
        cb = pl.colorbar()      # >> add a colorbar
        cb.set_label(’z (in m)’)

    return fig


# -- Pick features from image ------------------------------------------------

# >> Displays image for clicking on features records 3 mouseclicks.
#
# <inputs>
#
# ds=dataset;xoff=xoffset;yoff=yoffset; rows;cols; dtype=for show_image</inputs>
#
#
# <products>
#
# featurePoint=list of 3x2 mouse click points(Normally: center, rim, ejecta-end)
# featureR= Radius to Rim; ejectaR= Distance Center to End of ejecta; multi=True
# if multi=False only one feature is allowed to be chosen </products>
# ---------------------------------------------------------------------------


def pick_features(ds, xoff =0, yoff = 0, rows = [], cols = [], dtype = ’image’,\
  imsize = ’Big’, multi = True, pause=False):
```

```
    data=get_data(ds,xoff,yoff,rows,cols)

# >> Infinite loop to find as many features as possible. To move on, doubleclick

    next=True
    featurePoint=[]
    featR2=[]
    featR1=[]
    nxy=np.zeros((3,2))
    featx=[]
    featy=[]

    while next :

# >> Plot Image

        f1=show_image(data,datatype=dtype,size=imsize)

        # >> Pause for zooming if desired [waw][130101]

        if pause :
            keep=raw_input(\
                '>>>> Select Zoom, select limits, unselect Zoom, press Enter.')

        print '>>>> Click on feature center, inner radius and outer radius'
        print '>>>> DOUBLE CLICK to STOP'

        xy=pl.ginput(n=3,timeout=1e7)

        xy=pl.asarray(xy)

        if xy[0,:] is xy[1,:]: #Breaks loop if click in same spot.
            pl.close(f1)
            next=False
            break

        nxy[:,0]=xy[:,0]+xoff
        nxy[:,1]=xy[:,1]+yoff

# >> Calculate feature and ejecta radii

        R1=((nxy[0][0]-nxy[1][0])**2+(nxy[0][1]-nxy[1][1])**2)**.5
        R2=((nxy[1][0]-nxy[2][0])**2+(nxy[0][1]-nxy[2][1])**2)**.5
        if R2<R1: #if 3rd point is inside the rim, it means ejecta not visible
            R2=0
```

```
        pl.close(f1)

# >> Show chosen points and give option to delete them

        f2=show_image(data,datatype=dtype, size='iddy-biddy')
        pl.axis('image')
        pl.plot([xy[0][0],xy[1][0]],[xy[0][1],xy[1][1]],'-o')
        pl.plot([xy[0][0],xy[2][0]],[xy[0][1],xy[2][1]],'-o')

        keep=raw_input('Keep points? ([y]/n) : ')
        pl.close(f2)

# >> If points are for keeping, append them to the output variables
        if keep !='n' and keep !='o': #add new set to feature list
            featR1.append(R1)
            featR2.append(R2)
            featurePoint.append(nxy)
            if not multi:
                next=False
                break
            more=raw_input('More features? (y/n) : ')
            if more=='n':
                next=False
            elif more=='y':

# >> Show the previously chosen features, in mini figure window

                mini=show_image(data,datatype=dtype,size='tiny')
                featx.append(xy[0][0])
                featy.append(xy[0][1])
                mini
                pl.plot(featx,featy,'or')
                pl.show() # !! see {ion}

    return featurePoint,featR1,featR2


# -- Write Text File ---------------------------------------------------------
#
# >> Writes a line into an open text file with feature info
#
# <inputs>
#
# featurePnt=list of feature coordinate arrays (output of pick_features); fnum=
# the ID number of the first feature in the feature coordinate array
```

```
# fR1= the distance from the center to the rim; fR2= distance center to
# edge of ejecta; imagename= name of the image the feature was found in; text=
# open text file to write to; extra= anything else to put into text file
# </inputs>
#
#
# <products>
#
# writing in the text file </products>
# ------------------------------------------------------------------------------


def write_feature_textfile(featurePnt,fnum,fR1,fR2,imagename,text,fulldim):

# >> order =ImageID, Feature Num, Center, Rim, Ejecta, FeatureR, EjectaR, Extra

    for i in np.arange(0,len(featurePnt),1):

        # >> in order of id, cn, dx, dy, xc, yc, r1, r2:

        strn = imagename + ', ' + \
            str(i+fnum).zfill(3) + ', ' + \
            str(int(fulldim[0])) + ', ' + \
            str(int(fulldim[1])) + ', ' + \
            str(int(np.round(featurePnt[i][0,0]))) + ', ' + \
            str(int(np.round(featurePnt[i][0,1]))) + ', ' + \
            str(int(np.round(fR1[i]))) + ', ' + \
            str(int(np.round(fR2[i]))) + '\n'

        text.write(strn)

        # >>  id = observation ID (PSP/ESP_012345_0123)
        # >>  cn = feature number
        # >>  dx = host image horizontal dimensions (in pixels)
        # >>  dy = host image vertical dimensions (in pixels)
        # >>  xc = center of feature in the host image, x position
        # >>  yc = center of feature in the host image, y position
        # >>  r1 = feature radius 1 (inner)
        # >>  r2 = feature radius 2 (outer)



# -- Make Gherkins -------------------------------------------------------------
#
# >> completed [120815]
```

```
# >> will make feature pickle starting from opening the image
#
# <inputs>
#
# drape= drape image file name; dtm= dtm image file name; obID2= name of the \
# second stereo image; basepath= file path; savepath= folder path for finished \
# pickles; featnum= feature number can be specified by the user to start at a \
# given number then incremented with every pickle created; crop= array of crop-\
# ing radii for each pickle in a set; crpID= Name of each type of crop, example\
# 'bowl' or 'ejecta';  slices= how many pieces to view
# drape image in, while picking out features </inputs>
#
# <products>
#
# pickled gridmaps </products>
#
# ------------------------------------------------------------------------------

def make_feature_pickle(drape, dtm, obID2, conf, instr, origin, basepath = '', \
         savepath='./pkl/', featnum = 0, crop = [4], crpID = [''], slices = 1):

    drpds, drpdim, uplf, pixdim = open_image(drape, basepath)
    dtmds, dtmdim, uplf, pixdim = open_image(dtm, basepath)
    cnfds, cnfdim, uplf, pixdim = open_image(conf, basepath)

# >> Double Check that dimension of drape and dtm match

    if drpdim != dtmdim or drpdim != cnfdim:
        print 'Dimensions Do Not Match'
        return

    chunk = drpdim[1] / slices

# >> Run through image in chunks

    for t in range(0,drpdim[1]-chunk,chunk):

# >> Reset variables for each chunk

        done=False
        flocations=[]
        ZR=[]

        print 'Cick on feature center and Zoom Radius.'
        print 'DOUBLE CLICK When DONE'
```

```
# >> Set up image to be analyzed

        imdata=get_data(drpds,0,t,chunk,drpdim[0])
        f1=show_image(imdata,fignum=1)
        pl.axis('image') # >> Set axis so pretty dots work :)

        while not done:

            xy = pl.ginput(n=2, timeout=1e7)

# >> Break with Double Click

            if xy[0][0] == xy[1][0] and xy[0][1]==xy[1][1]:
                done = True
                break

# >> Plot previous feature locations

            pl.plot(xy[0][0],xy[0][1],'o')

# >> Save info

            flocations.append(xy)
            ZR.append(((xy[0][0]-xy[1][0])**2+(xy[0][1]-xy[1][1])**2)**.5)

# >> Go through all features flagged

        for i in range(0, len(flocations), 1):

# >> Find Zoom dimensions
            zmxst = int((flocations[i][0][0]-ZR[i]))
            zmyst = int(t + (flocations[i][0][1]-ZR[i]))
            zmsize = int((2 * ZR[i]))

# >> Just incase zoombox goes off image

            if zmxst < 0:
                zmxst = 0
            if zmyst < 0:
                zmyst = 0
            if zmsize+zmxst > drpdim[0]:
                zmsize = drpdim[0] - zmxst
            if zmsize+zmyst > drpdim[1]:
                zmsize = drpdim[1] - zmyst
```

```
# >> Pick feature in each zoom box

            featpoint,fR1,fR2=pick_features(drpds, zmxst, zmyst, zmsize,zmsize,\
                imsize='Bulbasaur', multi=False)

            if featpoint != []:
                centx = featpoint[0][0][0]
                centy = featpoint[0][0][1]

# >> Make N pickles for each feature found

                for N in range(0, len(crop), 1):

# >> Find feature croping dimensions

                    xstart = (centx - crop[N] * fR1[0])
                    ystart = (centy - crop[N] * fR1[0])
                    xstop  = (centx + crop[N] * fR1[0])
                    ystop  = (centy + crop[N] * fR1[0])

# >> If the cropping dimensions are outside the image, crop to image edge. But
# >> keep the crater centered.

                    if xstart < 0:
                        xstart = 0

                    if ystart < 0:
                        ystart = 0

                    if xstop > drpdim[0]:
                        xstop = drpdim[0]

                    if ystop > drpdim[1]:
                        ystop = drpdim[1]

                    cropdim=[[xstart,ystart],[xstop,ystop]]

# >> Create the grid map for the feature.

                    gdmp = gg.cl_gridmap()

                    gdmp.load(map_file = basepath+dtm, map_type = 'gdal', \
                                  crop = cropdim)
                    gdmp.load(map_file = basepath+drape, map_type = 'gdal', \
```

```
                                crop = cropdim, NaNcutoff=-30000)
                    gdmp.load(map_file = basepath+conf, map_type = 'gdal', \
                                crop = cropdim)

                    gdmp.metc['radiiToEdge'] = abs(0.5* (xstart-xstop) / fR1[0])
                    gdmp.metc['FeatureCenters'] = [[int(centx - xstart),\
                                                    int(centy - ystart)]]
                    gdmp.metc['OuterRadiusPix'] = fR2[0]
                    gdmp.metc['InnerRadiusPix'] = fR1[0]

                    gdmp.metc['OuterRadius(m)'] = fR2[0] / pixdim[0]
                    gdmp.metc['InnerRadius(m)'] = fR1[0] / pixdim[0]

# >> Embed gridmaps that go off screen with NaN's

                    gdmp.recenter_gridmap()

# >> SAVE GRIDMAP
# >> Name= "Instrument-DEM origin_observation ID1+ID2 -creaternum_croptype@size"

                    gdmp.pickle(savepath + instr +'-'+ origin+'_'+drape[-17:-4]\
                                         + '+'+ obID2[-17:-4] + '-' \
                                         + str(featnum).zfill(3) +'_' \
                                         + crpID[N] +'@norm.pkl')

                featnum += 1
```

3.3 Detrend Digital Elevation Models

```
# ============================================================================
#
# 20120801-1159-gpudoo - detrend_gridmaps.py
# Methods, detrend gridmaps / W. A. Watters F., Michaela Fendrock & Lynn Geiger
# Methods to detrend gripmap objects
#
# <inputs>
#
# Need to be given a gridmap and a mapnum, and a, the distance in from the edge
# of the image where the profile is taken. </inputs>
#
# <products>
#
# Returns a detrended gridmap. </products>
# ============================================================================

import numpy as np

# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# :::::::::::::::::::::::::::::::: Edge ::::::::::::::::::::::::::::::::::::::
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

# >> Traces rim of image, shifted in n, detrends based on that profile

def edge(gdmp,mapnum,n=100):

    import numpy as np

    # >> How far in from the edge the profile will be taken

    a = 2

    # >> Coordinates of four corners, then the first one again ([x,x][y,y])

    edge = [gdmp.xext[0]+a, gdmp.xext[1]-a, gdmp.xext[1]-a, gdmp.xext[0]+a,\
            gdmp.xext[0]+a],[gdmp.yext[0]+a,gdmp.yext[0]+a,\
            gdmp.yext[1]-a, gdmp.yext[1]-a, gdmp.yext[0]+a]

    # >> xp is x coordinates, yp is y coordinates

    xp = edge[0]
    yp = edge[1]

    # >> profile along those coordinates
```

```
xd, yd, sd, zd = gdmp.profile(x = xp, y = yp, mapnum = 0)

# >> Take n points from the profile, in a list

indss = np.round(np.linspace(0, len(xd)-1, n)).tolist()

# >> Make profiled points into arrays

xd = np.array(xd)
yd = np.array(yd)
zd = np.array(zd)

# >> We want to detrend based on the indices we took above

Xd = xd[indss]
Yd = yd[indss]
Zd = zd[indss]

gdmp.detrend(x = Xd, y = Yd, z = Zd,mapnum = mapnum)

# >> Now to find the nadir
# >> Make a box around the center of the crater that is 2.4 crater radii
# >> on a side (1.2 from center in two ways, gdmp.etc is how many crater
# >> radii from center to edge).

center = [((gdmp.xext[1]+gdmp.xext[0])/2),\
          ((gdmp.yext[1]+gdmp.yext[0])/2)]

r = 1.2*abs(gdmp.xext[1]-gdmp.xext[0])/2

xmin = center[0] - r/gdmp.metc['radiiToEdge']
xmax = center[0] + r/gdmp.metc['radiiToEdge']
ymin = center[1] - r/gdmp.metc['radiiToEdge']
ymax = center[1] + r/gdmp.metc['radiiToEdge']

# >> Only want coordinates that are in the box defined above,
# >> nonzero gives indices

indss = np.nonzero((gdmp.xmsh > xmin)*(gdmp.xmsh < xmax)*\
                   (gdmp.ymsh > ymin)*(gdmp.ymsh < ymax))

# >> Last gridmap is the detrended, detrended gridmap,
# >> want the indss of that
```

```
    mooka = gdmp.maps[-1]
    box = mooka[indss]

    inde = np.nonzero((1 - (np.isnan(box)))*\
                      (1 - (np.isnan(box))))

    boxed = box[inde]

    # >> Minimum of box is bottom of crater

    gdmp.metc['zNad'] = boxed.min()

# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# :::::::::::::::::::::::::::::::: Clip ::::::::::::::::::::::::::::::::::::::
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

# >> Takes a profiled map and profiles along the edge everywhere that the
# >> standard deviation of elevation is crater than the elevation minus the mean
# >> elevation.


def clip(gdmp,mapnum,x,y,z,n=100):

    import numpy as np
    import pdb

    # >> filter where we want to detrend

    # >> Don't want NaNs

    indi = np.nonzero(1-np.isnan(z))

    zs = z[indi]

    indss = np.nonzero(np.abs((zs - zs.mean()) < zs.std()))[0]

    # >> only detrend for filtered points

    Xd = x[indss]
    Yd = y[indss]
    Zd = z[indss]

    gdmp.detrend(x = Xd, y = Yd, z = Zd, mapnum = mapnum)
```

```
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# :::::::::::::::::::::::::::::: Circles ::::::::::::::::::::::::::::::::::::::
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

def circles(gdmp,mapnum,n=100,rs=np.array([4, 4.5, 4.9]),x=None,y=None,z=None,\
                prNaN=False):

    import numpy as np
    import math
    import pdb

    # >> 'nc' is number of circles, n is number of points on each circle

    if x == None:

        # >> Center of image should be center of crater

        center = [((gdmp.xext[1]+gdmp.xext[0])/2),\
                  ((gdmp.yext[1]+gdmp.yext[0])/2)]

        # >> 'radiiToEdge' is from center to edge, so we need half the image

        half = (abs(gdmp.xext[1]-gdmp.xext[0])/2)


        # >> Find radius of crater
        if 'rad' in gdmp.metc: R = gdmp.metc['rad']
        else:
            R = (abs(gdmp.xext[1]-gdmp.xext[0])/2)/abs(gdmp.metc['radiiToEdge'])

        # >> Lists to be filled with x and y coordinates

        x = []
        y = []

        # >> Loop through radii of circles

        for i in rs:

            # >> Sample n, evenly spaced points around the circle, append their
            # >> x and y coordinates to lists


            for j in np.linspace(0,360,n):
```

```
                x.append(center[0] + R*i*math.cos(math.radians(j)))
                y.append(center[1] + R*i*math.sin(math.radians(j)))

        # >> Profile to get z coordinates

        xd, yd, sd, zd = gdmp.profile(x = x, y = y, mapnum = mapnum)

# >> Get rid of 0 zd values [lmg121113](and NaN [121126])
        keepind = np.nonzero((~np.isnan(zd))*(zd!=0))[0]
        if len(keepind)==0:
            print 'Detrending Circles Empty'
            return 0,0,0,1
        x = xd[keepind]
        y = yd[keepind]
        z = zd[keepind]
# >> Detrend based on those points

    gdmp.detrend(x = x, y = y, z = z, mapnum = mapnum)

# >> This is important, need x, y, and z for detrending bowls

    if prNaN:
        # >> Added [121201][lmg]
        pernan = float(len(zd)-len(keepind))/len(zd)

        return x,y,z,pernan

    else: return x, y, z
```

3.4 Planform Extractor

```
#==========================================================================
# 20121112-1326-gpudoo - extract_planforms.py
# Master Crater Rim Extractor / Lynn Geiger, Michaela Fendrock, Wesley Watters
# Crater black-box. All methods, do diagnostics, display images, make golbase
#
# <inputs>
#
# nums      = which master feature numbers to run
#
# basepath  = from where to pull the file list. if input type is a gridmap, it
#             can double as the crater ID name
#
# method    = 'max','slope','detrend'
#
# prorder   = Profile spline order
#
# prlen     = Profile length, in radii
#
# N         = number of profiles for the methods
#
# combo     = if True calls the matrix method, that stitches a "best trace"
#
# matVal    = variables for matrix rim selection [percent of raduis for slope
#             calculation, local max percent, min grade change, min slope
#             cluster size, Max radial discontinuity, Radial Allowed Neighbor
#             Distance, Angular Allowed Neighbor Distance, Sift redundant traces
#             (T/F), continue looping until reaching a previous point (T/F)]
#
# startang  = Angles at which to start best trace 'stitching'
#
# analysis  = True does diagnostic anaysis, and saves golembase
#
# flag      = chooses which property will raise a flag, 'eps' = epsillon
#
# flcritical= threshold to flag objects
#
# maxejoff  = Maximum allowed ejecta-bowl offset, craters above will be flaged
#             offset is in meters.
#
# NaNEjMax  = Max allow NaN in ejcta [%]
#
# bdim      = Minimum Bowl DEM dimensions
#
# Ejdtcirc  = Ejecta detrending circle radii
```

```
#
# detrend   = method to detrend the DEMs, 'circles'
#
# images    = True saves .png composite
#
# contour   = Calculate contours, (T/F)
#
# contfrac  = Only for contours. fraction of rim height for contour, default=.95
#
# verbose   = for debugging prints out each step
#
# savefile  = name of txt file. Uses Epdo syntax:
#               ++ field 1 # field 2 # field 3
#               ** field1 def # field2 def # field3 def
#               3212.3214 # 1234.1234 # .1234
#               3212.3214 # 1234.1234 # .1234
#               3212.3214 # 1234.1234 # .1234
#               3212.3214 # 1234.1234 # .1234
#               ++
#
# gbsvname = golembase save name
#
# giveMat  = Diagnostics term, if true program will return the Matrix and stop
#
# </inputs>
#
# <products>
# DIAGNOSTICS MODE:
# Golembase with following fields;
# 'traces', 3-D coordinates of rim, by chosen method
# 'allTrace', All found rim traces
# 'ejctFile' & 'bowlFile', Name of DEM pickle files
# 'geocnt', Geometeric Center of Crater, Calculated by fitting a circle
# 'eps', Epsilon - Max discontinuity as fraction of diameter
# 'diam', Average Diameter of Crater
# 'desc1','desic2', 'desic3', Old Gamma, Zeta, Eta (Ask Michaela)
# 'ejDem', Usefulness of Ejecta DEM, NaN or Size, (T/F)
# 'dpthdiam', Depth Diameter array [Max, Min, Average]
# 'zNad', Elevation of nadir
# 'Flag', Diagnostic Flag.
#
# </products>
#
# ================================================================================
```

```
import pdb                              # Python Debugger, use pdb.set_trace()
import pylab as pl
import math, os, copy
import numpy as np
import fnmatch as fm
import golbase as gb
import geometria as gm # Do we still want this??
import golrim as gr
from omnigenus import unpickle          # Reading in pickle files' gdmp
import pickle                           # for Saving pickle files
import epdobase as eb
import extract_crater_matrix as ecm
import detrend_gridmaps as dt
import extract_dem_features as edf
import translate_feature_index as tf

def RIM(nums, basepath = '', method = 'max', prorder = 1, prlen =1.5, N =512,\
            combo =True, matVal = [0.10, 0.10, 0.05, 0.05,.1,.05,2,True,True],\
            startang=[0, 45, 90, 135, 180, 225, 270, 315], analysis = True, \
            flag = 'eps', flcritical = 0.25, maxejoff = 5, NaNEjMax = .9, \
            bdim = 3.25, EJdtcirc = np.array([3.0,3.5]), detrend = 'rim', \
            images = 'Diagnostic', contour = True, contfrac = .95,\
            verbose = True, savefile = [], gbsvname = 'golbase.pkl', \
            giveMat = False):
# ****************************************************************************
# >> Preliminary Set up

    if savefile:

        # >> Set up epdobase file

        txtf = open(savefile,'w')
        txtf.write('++ mfnum # obsid1 # obsid2 # fnum # eSz # bSz # MDd #'+\
                       ' mdD # AdD # Flag # FR\n** Crater Name # Observation '+\
                       'ID 1 # Observation ID 2 # Feature Number # Ejecta di'+ \
                       'mensions # Bowl DEM dimensions # Max Depth-Diameter '+ \
                       'Ratio # Minimum Depth-Diameter Ratio # Average Depth'+ \
                       '-Diameter Ratio # Crater Flag # Reason for Flagging \n')


    if analysis:

        # >> Set up golbase

        golb = gb.cl_golbase()
```

```
        fields = ['trace', 'allTrace', 'maxTrace', 'stitches', 'eps', 'omg', \
                  'gam','mfnum', 'OrzNad', 'IPzNad', 'EjzNad', 'ejcnt',\
                  'ejctFile', 'bowlFile', 'dpix', 'ejdim', 'bldim', 'ejoffset',\
                  'geocnt', 'rAvg', 'diam', 'desc1', 'desc2', 'desc3', 'ejDem',\
                  'Ordpdm','Ipdpdm','Ejdpdm', 'proper','Flag','Gaps','manEjCnt']

        sqlType = {'rAvg' :'REAL', 'OrzNad' : 'REAL','IPzNad' : 'REAL','EjzNad'\
                       : 'REAL', 'eps' :'REAL', 'ejctFile':\
                   'TEXT', 'bowlFile' :'TEXT', 'desc1' :'REAL','desc2' :'REAL',\
                   'desc3' : 'REAL', 'proper' : 'BOOLEAN','dpthdiam' : 'REAL', \
                   'ejDem' : 'BOOLEAN', 'dpix' : 'REAL', 'Flag': 'TEXT','diam':\
                   'REAL', 'geocnt':'REAL', 'ejoffset' :'REAL', 'bldim':'REAL',\
                   'ejdim':'REAL', 'mfnum' :'TEXT', 'gam':'REAL', 'omg':'REAl',\
                   'Gaps' : 'INTEGER', 'manEjCnt' : 'BOOLEAN'}

        golb.flds = fields

        golb.sqlType = sqlType


# >> Load in master feature numbers

    mfidx = tf.cl_mfidx()

# >> Get filelist for crater pickles
    filelist = sorted(os.listdir(basepath))

# >> Set up empty variables
    Flaged = []
    Broken = []
    CntFail= []
    Failures=0
    i = 0


#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------


# >> Loop through craters

    for numb in nums:

        if verbose:
```

```
    print str(i * 100 / len(nums)) + '% Complete'

number = str(numb).zfill(3)

# >> Reset anything that could have been flaged on the last crater
# >> If crater is flaged for something and detrending method needs to
# >>  switch for the individual, change back here

dtmeth= detrend
cflag = 'N'
freas = '-'
rimnum= 0
ejsize= True


# >> d-DEM is 3 for ALLLLLLL [130128]
ddem = 3

# >> find file name
crid = mfidx.fname_string_from_mfnum(numb)

# >> Find bowl file
bowlfID = fm.filter(filelist, ('*' + crid + '*bowl*.pkl'))[0]

# >> unpickle Bowl
pbwl = basepath + bowlfID
bowl = unpickle(pbwl)

# >> Create Crater name
cratername = 'mfnum-' + number +'_'+ crid

# >> Let us know where we are
print 'Processing ' + cratername

# >> Get Ejecta file
ejtafID = fm.filter(filelist, ('*' + crid + '*ejct*.pkl'))[0]

if ejtafID:
    ejct = unpickle(basepath + ejtafID)
    ejctpdim = ejct.pdim

# >> If the Ejecta file is not there then flag it as such
else:
    ejsize = False
    freas = 'DetSp'
```

```
        cflag = 'Y'
        print 'Ejecta File Missing'
        ejctpdim = [0, 0]
        coffs = [0, 0]
        Flaged.append(cratername)
        dtmeth = 'special'
        ejtafID = False
        EcrH = np.array([np.nan, np.nan, np.nan])

 # >> Check that the Bowl DEM is reasonably large enough to run
    if 'dimx_rad' in bowl.metc:
        if (bowl.metc['dimx_rad'] < bdim)+(bowl.metc['dimy_rad'] <bdim):
            print 'Bowl Crop Small ' + str(bowl.metc['dimx_rad'])

            if ejtafID[-9:] == '@smal.pkl':
                print 'But Ejecta is  @smal...'
                # >> Flag as bowl & Ej small
                freas = 'BESml'
            else:
 # >> If the bowl is too small, use the ejecta gdmp instead
                bowl = copy.deepcopy(ejct)
                bowlfID = ejtafID
                freas = 'BlSml'

            # >> Still flag being small even if we use it
            cflag = 'Y'
            Flaged.append(cratername)


# >> Important Crater Attributes:

# >> Coordinates of center of image (and hopefully center of crater)
    center = np.array([((bowl.xext[1] + bowl.xext[0]) /2), \
                  ((bowl.yext[1] + bowl.yext[0])/2)])

    print 'Bowl Center: ' + str(center)

    if ejsize:
        ejcenter = np.array([((ejct.xext[1] + ejct.xext[0]) /2), \
                        ((ejct.yext[1] + ejct.yext[0])/2)])

        print 'Ejct Center: ' + str(ejcenter)

# >> Half of the width of the image
    half = (abs(bowl.xext[1]-bowl.xext[0])/2)
```

```
    # >> Crater radii 'Cr', Profile radius, 'Hr'=half radii
        if 'rad' in bowl.metc: Cr = bowl.metc['rad']
        else: Cr = (half/bowl.metc['radiiToEdge'])

        Pr = prlen * Cr # >> profile radius
        Hr = 0.5 * Cr

        if verbose:
            print 'Ejecta Dimensions:   ' + str(ejctpdim)
            print 'Bowl Dimensions:     ' + str(bowl.pdim)
            print 'Detrending....'

# >> Add name to savefile
        if savefile:
            obs1, obs2, fnum = mfidx.obsids_ifnum_from_mfnum(numb)

            txtf.write(number +' # '+ obs1 +' # '+obs2 +' # '+ str(fnum)+' # '+\
                       str(ejctpdim[0]) + ' # ' + str(bowl.pdim[0]) + ' # ')


# >> If the gdmp is too large, the program will choke... Mostly a USGS problem
        if ejctpdim[0] > 10000:

            print 'Warning: Crater DEM too Large,'

            Flaged.append(cratername)
            dtmeth = 'special'
            rimnum = 0
            cflag = 'Y'
            freas = 'DetSp'
            ejsize = False


#-------------------------------------------------------------------------------
#++++++++++++++++++++++++++++ D E T R E N D ++++++++++++++++++++++++++++++++++
#-------------------------------------------------------------------------------

# >> Detrend based on variable

        dtplanes = [] # >> to get correct Z elevations (!!we don't use anymore)


        if dtmeth == 'circles' or dtmeth == 'rim':
```

```
# >> Detrend by ejecta circles
dtx, dty, dtz, prNaN = dt.circles(ejct, mapnum = 0, rs = EJdtcirc, \
                                      prNaN=True)


# >> See if it has been centered manually
if 'ejcent' in ejct.metc:
    recenter = False
    ejcenter = ejct.metc['ejcent']
    manejcnt = True

# >> If this is flaged there is a ej-bl offset in xy meter-space
elif 'dimx_rad' in ejct.metc:
    recenter = True
    manejcnt = False
else:
    recenter = False
    manejcnt = False

# >> Reset counter for recentering
times = 0

# >> Need to assign dfcen, incase grim.proper = False (check this)
dfcen = np.array([np.nan, np.nan])

while recenter:

    # >> Find the geometric center of the ejecta DEM
    coords = simple_method(method, gdmp = ejct, center = ejcenter, \
                               Pr = Pr, N = N, mapnum = -1)
    # >> Remove 0's or NaNs
    inds = np.nonzero((coords[:,0]!=0)*(~np.isnan(coords[:,0])))[0]
    coords = coords[inds]

    if times > 0: # >> Find Avg Radius and Std
        ejrs = ((coords[:,0] - ejcenter[0]) **2 + (coords[:,1] \
                                      -  ejcenter[1]) **2) **0.5
        AejR = sum(ejrs)/len(ejrs)
        stER = sum(ejrs**2)/len(ejrs) - AejR **2

    # >> Remove anything greater than 2std
    # >> (minimized odds of centering on a ridge)
        coords = coords[np.nonzero((ejrs < (AejR + 2*stER)) *(ejrs \
                    < (AejR + 2*stER)))[0]]
```

```
                # >> Use golrim to calculate the geometric center
                grim = gr.cl_golrim()
                grim.cRaw = coords

                if grim.proper:
                    grim.compCentroid()
                if grim.proper:

                # >> Find centroid difference between the runs
                    dfcen = ejcenter - grim.cCen[:2]
                    if verbose:
                        print 'EjCnt before'+ str(ejcenter) + 'EjCnt Aft' + \
                            str(grim.cCen[:2])
                        print dfcen

                    ejcenter = grim.cCen[:2]

                # >> If it's run five times or the difference is less than 1m
                if (times >= 5) + (sum(dfcen**2) < 1):
                    # >> Break the loop
                    recenter = False

                times += 1
                if verbose: print times

     # >> If we are using the ejct as the bowl, we need to use the new center
            if freas == 'BlSml': center = ejcenter

            # >> If the ejct and bowl dems are not centered on same point,
            # >> add the center off set to all ejct calculations:
            coffs = center - ejcenter
# ---------------    Done Recentering ----------------------------

    #  >> Checks the percent NaN of the circles, if over 50% flag
            if prNaN > NaNEjMax:
                print 'Ej NaN Warning: ' +str(round(prNaN*100))+'% Crater '\
                        + cratername
                cflag = 'Y'
                freas = 'DtNaN'
                rimnum=0
                ejsize=False
                Flaged.append(cratername)

            else:
                # >> convert x&y with offset
```

```
            dtx -= coffs[0]; dty -= coffs[1]

            # >> detrend bowl with ejecta information
            dt.circles(bowl,mapnum=ddem,x=dtx,y=dty,z=dtz)
            ddem = len(bowl.maps) -1

            dt.circles(bowl,0,x=dtx,y=dty,z=dtz)
            pfit = gm.fit_plane(dtx,dty,dtz)
            dtplanes.append(pfit)
            rimnum= len(bowl.maps) -1


# ---- Rim Detrending -------------------------------------------------------

        if dtmeth == 'rim' or dtmeth == 'special':

            if verbose:
                print '   Rim Detrending...'

            # >> If ejecta is debunked, skip it
            if ejsize:

             # >> Detrend the ejct with ejct crude rim (MERT)
                coords = simple_method(method, gdmp = ejct, center = ejcenter,\
                                       mapnum = -1, Pr = Pr, N = N)
                inds = np.nonzero((coords[:,0] !=0)*(~np.isnan(coords[:,0])))[0]
                coords = coords[inds,:]

                # >> separate xyz
                dtx = coords[:,0]
                dty = coords[:,1]
                dtz = coords[:,2]

                # >> detrend
                dt.clip(ejct,-1,x=dtx,y=dty,z=dtz)

                # >> Save crater heights for dpDm later
                EcrH = np.array([max(coords[:, 2]), min(coords[:, 2]), \
                                 np.mean(coords[:, 2])])
                # >> Might want to figure better way
            else: EcrH = np.array([np.nan, np.nan,np.nan])

            #>> Find crude rim to detrend with
            coords = simple_method(method, gdmp = bowl, mapnum = rimnum, \
                                   center = center,Pr = Pr, N = N)
```

```
inds = np.nonzero((coords[:,0] != 0)*(~np.isnan(coords[:,0])))[0]
coords = coords[inds]

# >> Turn into array for processing
dtx = coords[:,0]
dty = coords[:,1]
dtz = coords[:,2]

# >> Detrend based on these rim points
try:

    dt.clip(bowl,mapnum=ddem,x=dtx,y=dty,z=dtz)
    ddem = -2

    if contour:
        dt.clip(bowl, mapnum=4, x=dtx, y=dty, z=dtz)
        ddem -= 1

    dt.clip(bowl, rimnum, x=dtx, y=dty, z=dtz)

    rimnum=-1

except:
# >> If it breaks Flag it and move on
    print 'ERROR: Crater ' + cratername
    print 'Failed to RIM Detrend \n'
    cflag = 'Y'
    freas = 'RDetF'
    Broken.append(cratername)
    Failures +=1
    if savefile:
        txtf.write('0 # 0 # 0 # Y # '+freas+'\n')
    continue


x=[];y=[]

# >> If the ejcta is good, sample it for the ambient plane
if ejsize: bksamp= EJdtcirc[-1]*Cr; dtst = ejcenter

# >> Otherwise sample the bowl DEM
else: bksamp= 2.*Cr; dtst = center

for j in np.arange(0,360,1):
    x.append(dtst[0] + bksamp*math.cos(math.radians(j)))
```

```
            y.append(dtst[1] + bksamp*math.sin(math.radians(j)))

        if ejsize: x, y, s, zbk = ejct.profile(x = x, y = y, mapnum = -1, \
                                                order = prorder)
        else:  x, y, s, zbk = bowl.profile(x = x, y = y, mapnum = -1, \
                                            order= prorder)

        # >> Remove NaN and average
        zbk = zbk[np.nonzero(~np.isnan(zbk))[0]]
        backgr= np.average(zbk)

        # >> Subtract ambient elevation from DEM

        bowl.maps[-1]=bowl.maps[-1]-backgr

        if contour: bowl.maps[-2]=bowl.maps[-2]-backgr

        bowl.maps[ddem] = bowl.maps[ddem]- backgr

        if ejsize:
            ejct.maps[-1]=ejct.maps[-1]-backgr

        pfit = gm.fit_plane(dtx,dty,dtz)

        # >> Have to correct for subtracting background elevation
        pfit[3]= pfit[3]-backgr*pfit[2]


        dtplanes.append(pfit) # Planes for all detrending


#_______________________________________________________________________________
#:::::: FIND the RIM TRACE ::::::::::::::::::::::::::::::::::::::::::::::::::
#_______________________________________________________________________________

    if verbose:
        print 'Finding Rim...'

    try:
        # >> Get Lowest point

        Ornad = edf.nadir(bowl,mapnum = ddem)
        IPnad = edf.nadir(bowl,mapnum=rimnum)
        Ejnad = edf.nadir(ejct,mapnum=rimnum)
```

```
        except:
            # >> Used to fail sometimes
            print 'ERROR: Nadir Failure. Crater ' + cratername
            print ''
            cflag = 'Y'
            freas = 'Nadir'
            Broken.append(cratername)
            Failures +=1
            if savefile:
                txtf.write('0 # 0 # 0 # Y # ' + freas + '\n')
            continue


        # >> Find the MaxElevationRimTrace
        coords = simple_method(method, gdmp = bowl, center = center, Pr = Pr, \
                                    N = N, mapnum = rimnum)

# >> Remove the 0s and NaNs
        inds = np.nonzero((coords[:,0] != 0)*(~np.isnan(coords[:,0])))[0]
        coords = coords[inds,:]


#_______________________________________________________________________________
                            # # # MATRIX # # #

        if combo:

#>> Matrices: R,X,Y,Z,SlpB4,DeltaSlope,DSlopeMaxInd,LocalMaxInd,GlobalMax,SlpAft
            try:
# >> Get the Matrix for the crater!
                Matrix = ecm.find_matrix(bowl, mapnum=rimnum, slper=matVal[0], \
                         mxper = matVal[1], mingrade = matVal[2], minslcluster \
                         = matVal[3], N = N, spacing = 0.5 * bowl.dpix[0], \
                                              prlen= prlen, order = prorder)

# >> If we are runing diagnostics, return Matrix and detrended pkl and stop
                if giveMat == True:
                    return Matrix, bowl

            except:
                print 'ERROR: MATRIX FAILED ' + cratername
                Broken.append(cratername)
                cflag = 'Y'
                freas = 'MatPr'
```

```
                if savefile:
                    txtf.write('0 # 0 # 0 # Y # ' + freas + '\n')
                continue

# >> Some of the Smallest Craters are not finding enough GMaxima for next step
# >> If there are too little interest points skip it
            if sum(sum(Matrix[8])) < N/2:

                print 'ERROR: MATRIX FAILURE Not Enough Intrest: ' +cratername
                print 'Missing '+ str((N-sum(sum(Matrix[8])))*100/N) + '%'

                Broken.append(cratername)
                cflag = 'Y'
                freas = 'MatPr'

                if savefile:
                    txtf.write('0 # 0 # 0 # Y # ' + freas + '\n')
                continue

            # >> Calculate the composite rim.  Omega = Points off Max / #Points
            RAltR, Omg, Stitch, Gaps = ecm.composite_rim(Matrix, stangle = \
                            startang,
                            rdis = (Cr*matVal[4]), RAND = (Cr*matVal[5]), \
                            AAND = matVal[6], siftRedundant = matVal[7], \
                            contloop = matVal[8], kpstitch = True)


# >> Identify which is the best trace!
            RAltR.append(coords)
            Omg.append(0)
            Stitch.append([])

# >> set up variables:
# >> Eps = Greatest Discont/Radius, Gam=Top4discont/Radius, Alp=Gam+Omg
            Eps = np.zeros(len(RAltR))
            Gam = np.zeros(len(RAltR))
            Alp = np.zeros(len(RAltR))
            Omg=np.array(Omg)

            AltRim=[]
            t=0

            for trace in RAltR:
            # >> Remove all NaN
                inds = np.nonzero((trace[:,0]!=0)*(~np.isnan(trace[:,2])))[0]
```

```
        trace = trace[inds,:]

    # >> Calculate Eps and Gam for each trace
        grim = gr.cl_golrim()
        grim.cRaw = trace
        grim.compCentroid()
        grim.compPolar()
        grim.compInterp()
        grim.compGaps()

        Eps[t] = grim.eps2d
        Gam[t] = grim.gam2d
        Alp[t] = 0.5*Omg[t]+Gam[t]

        AltRim.append(trace)
        t += 1


    RimTraces=[]

# >> Set up variables for sorting
    epssort= []
    omgsort= []
    gamsort= []
    alpha =  []
    stitches=[]
    f=0
    found = 0

    while f < len(Eps):
    # >> Find smallest Alpha
    # >> Only need one, if Alp is exactly same, trace will be
        inds = np.nonzero(Alp == min(Alp))[0]
        ind = inds[0]

        RimTraces.append(AltRim[ind])

        epssort.append(Eps[ind])
        omgsort.append(Omg[ind])
        gamsort.append(Gam[ind])
        alpha.append(Alp[ind])
        stitches.append(Stitch[ind])
        found += 1
        f += len(inds)
```

```
                if found > 5: break
                Alp[inds] = max(Alp) + 10

            BestT = RimTraces[0]

        # >> If only using one method
        else:
            RAltR=[]
            Omg = []
            RimTraces=[]
            BestT=coords

#_______________________________________________________________________________
#::::::::::::Make Diagnostics::::::::::::::::::::::::::::::::::::::::::::::::::

        if analysis:

            if verbose:
                print 'Begining Rim Analysis'

            diagnostic = {}
            working = True

# >> Store the best
            diagnostic['trace'] = BestT
# >> Store a list of the 3xN arrays
            diagnostic['allTrace']= RimTraces
# >> Store the Max
            diagnostic['maxTrace']= coords
            diagnostic['stitches']= stitches
            diagnostic['eps'] = epssort
            diagnostic['omg'] = omgsort
            diagnostic['gam'] = gamsort
            diagnostic['Gaps']= Gaps

# >> Store pickle name and info
            diagnostic['mfnum'] = number
            diagnostic['ejctFile'] = ejtafID
            diagnostic['bowlFile'] = bowlfID

            diagnostic['dpix'] = bowl.dpix[0]

            diagnostic['ejdim'] = ejctpdim[0]
            diagnostic['bldim'] = bowl.pdim[0]
```

```
# >> Use golrim to calculate other important things
            grim = gr.cl_golrim()
            grim.cRaw = diagnostic['trace']

# >> Compute all rim properties: (See header for catalog)
            if grim.proper:
                grim.compCentroid()

            if grim.proper:
                bGC = grim.cCen[:2]
                diagnostic['geocnt']= bGC

                diagnostic['ejoffset']= bGC - ejcenter
                diagnostic['ejcnt'] = ejcenter
                diagnostic['manEjCnt']= manejcnt

                grim.compPolar()

            if grim.proper:
                grim.compInterp()

            if grim.proper:
                grim.compShape2D()

            if grim.proper:
                grim.compShape3D()

            if grim.proper:
                grim.compGaps()

            if grim.proper:

# >> Find the three different depth to diameter ratios
                IPdD = (np.array([max(BestT[:,2]), min(BestT[:,2]), grim.zAvg])\
                                - IPnad) / (2. * grim.rAvg)
                EjdD = (EcrH - Ejnad) / (2. * grim.rAvg)

                OrdD = (np.array([max(BestT[:, 2]), min(BestT[:, 2]),\
                                          grim.zAvg]) - Ornad) / (2. * grim.rAvg)
                diagnostic['OrzNad'] = Ornad
                diagnostic['Ordpdm'] = OrdD

                diagnostic['diam']  = 2*grim.rAvg
                diagnostic['desc1'] = grim.desc1    # >> gamma
```

```
                diagnostic['desc2'] = grim.desc2    # >> zeta
                diagnostic['desc3'] = grim.desc3    # >> eta
                diagnostic['ejDem'] = ejsize

                diagnostic['IPdpdm']= IPdD
                diagnostic['Ejdpdm']= EjdD

                diagnostic['IPzNad'] = IPnad
                diagnostic['EjzNad'] = Ejnad

# >> Optional flagging based on a metric
            if flag == 'eps':

                if diagnostic['eps'][0] > flcritical:

                    diagnostic['proper'] = False
                    Flaged.append(cratername)
                    # >> If the crater hasn't been flaged yet:
                    if cflag == 'N':
                        cflag = 'Y'
                        freas = 'LgEps'

# >> Save the flagin reasons
            diagnostic['Flag'] = freas
            golb.db.append(diagnostic)

        else: IPdD = [0,0,0] # >> If not using complex analysis

        if savefile:
            txtf.write(str(IPdD[0]) +' # ' + str(IPdD[1]) + ' # '+ str(IPdD[2])\
                           + ' # ' + cflag + ' # ' + freas + ' \n')

#_______________________________________________________________________________
#::::::::::::PLOT Bowl Images::::::::::::::::::::::::::::::::::::::::::::::::::

    # >> Plot first subplot, a dtm that will later have a rim plotted on it

    # 1.grey ejecta    # 2. grey bowl          # 3.grey bowl, rim & cont
    # 4. dem ejecta    # 5.dtm bowl detrend    # 6.dtm bowl w/rim &cont
    # 7.dt dem ejct    # 9.rim points alone    # 8.all interest pnts

        if images:

    # >> Reshape the trace array so that all the x's are in the
    # >> first element, y's second, and z's third
```

```
    coordsa = coords.T
    coordsB = RimTraces[0].T

    pl.figure(1); pl.clf()

    if verbose:
        print 'Plotting Plots'

    # >> Decrease the size of the tick labels
    pl.rc('xtick', labelsize=10)
    pl.rc('ytick', labelsize=10)

    xmin= bowl.xext[0]
    ymin= bowl.yext[0]

    # >> Subtract min for nicer axis
    Mmplx = (coordsa[0]-xmin)
    Mmply = (coordsa[1]-ymin)

    Bplx = (coordsB[0]-xmin)
    Bply = (coordsB[1]-ymin)


    ctx = center[0] - xmin
    cty = center[1] - ymin

    lims = (0, (bowl.xext[1] - xmin), 0, (bowl.yext[1] - ymin))
    clrmap = pl.cm.gray

if images == 'Paper':
    pl.ylabel('y [m]')

    pl.imshow(bowl.maps[-1],cmap=pl.cm.jet,extent=lims)
    pl.xlabel('x [m]')

    pl.colorbar()
    pl.savefig(cratername +'_DEM.png',dpi = 500)
    pl.savefig(cratername +'_DEM.svg',dpi = 500)
    pl.figure(1); pl.clf()

    sp1 = pl.subplot(121)


if images == 'Diagnostic':
```

```
        pl.figure(2); pl.clf()
        pl.figure(3); pl.clf()
        pl.figure(4); pl.clf()
        pl.figure(1)

# >> Greyscale Ejecta (1)

        if ejsize:
            ejlm = (0, ejct.xext[1] - ejct.xext[0],\
                        0, ejct.yext[1] - ejct.yext[0])
            sp1=pl.subplot(331)

            pl.imshow(ejct.maps[1],cmap=clrmap,extent=ejlm)
            pl.ylabel('y [m]')
            pl.xlabel('x [m]')
            pl.axis(ejlm)
            sp1.xaxis.set_label_position('top')
            pl.setp(sp1.get_xticklabels(), rotation='vertical')

# >> Greyscale Bowl (2)
        sp2=pl.subplot(332)
        pl.imshow(bowl.maps[1],cmap=clrmap,extent=lims)
        pl.axis(lims)
        pl.xlabel('x [m]')
        sp2.xaxis.set_label_position('top')
        pl.setp(sp2.get_xticklabels(), rotation='vertical')

# >> Greyscale Bowl And Pnts + Contour(3)
        sp2=pl.subplot(333)

    if images:

        pl.imshow(bowl.maps[1],cmap=clrmap,extent=lims)
        pl.plot(Mmplx, Mmply, 'bo', markersize = 2, markeredgewidth = 0.1,\
                    label = 'Max. Elev.')
        pl.plot(Bplx, Bply, 'ro', markersize = 2, markeredgewidth = 0.1,\
                    label = 'Composite')

        if contour:
            if good != 'fail':
                pl.plot((conttrace[0]-xmin),(conttrace[1]-ymin),'c-')

        pl.plot(ctx,cty, 'c*')
```

```
        pl.axis(lims)

        pl.xlabel('x [m]')

        clrmap = pl.cm.jet

    if images == 'Paper':
        pl.ylabel('y [m]')
        pl.legend(numpoints = 1, loc=1, prop={'size':10})
    else:
        sp2.xaxis.set_label_position('top')
        pl.setp(sp2.get_xticklabels(), rotation='vertical')

        pl.figure(3)
        pl.imshow(bowl.maps[-1],cmap=clrmap,extent=lims)
        pl.plot(Mmplx,Mmply,'k.')
        pl.figure(1)

# >> Ejecta DEM (4)
        if ejsize:
            sp4= pl.subplot(334)
            pl.imshow(ejct.maps[0],cmap=clrmap,extent=ejlm)
            pl.axis(ejlm)
            pl.ylabel('y [m]')
            pl.setp(sp4.get_xticklabels(), rotation='vertical')

# >> Ejecta DEM Detrended (7)
            sp7=pl.subplot(337)
            pl.imshow(ejct.maps[-1],cmap=clrmap,extent=ejlm)
            pl.ylabel('y [m]')
            pl.axis(ejlm)
            pl.setp(sp7.get_xticklabels(), rotation='vertical')


# >> Detrended Bowl (5)
        sp5=pl.subplot(335)
        pl.imshow(bowl.maps[-1],cmap=clrmap,extent=lims)
        pl.axis(lims)
        pl.setp(sp5.get_xticklabels(), rotation='vertical')


# >> Rim on dem (6)
        sp6 = pl.subplot(336)

        pl.imshow(bowl.maps[-1],cmap=clrmap,extent=lims)
```

```
        pl.plot(Mmplx, Mmply, 'k.', markersize = 2)
        if contour:
            if good !='fail':
                pl.plot((conttrace[0]-xmin),(conttrace[1]-ymin),'m-')
        pl.plot(ctx,cty, 'c*')
        pl.axis(lims)
        pl.colorbar()
        pl.setp(sp6.get_xticklabels(), rotation='vertical')


# >> Rims alone (8)
        sp3=pl.subplot(338)
        pl.plot(Mmplx,Mmply,'k.')

        pl.axis('scaled')
        pl.ylim( sp6.get_ylim() )
        pl.xlim( sp6.get_xlim() )

        colors=['r.','y.','g.','b.']
        sub =[231,232,234,235]

        # >> Want to plot 4 best traces, if there are less than 4 plot all
        if len(RimTraces)<4:
            tn = len(RimTraces)-1

        else: tn = 3

        # >> Loop through the 4 figures
        for p in range(4):
            K=0
            # >> Plot the top 4 traces
            for q in np.arange(tn,-1,-1):
                trace = RimTraces[q]
                # >> p2 is the individual trace figure
                if p ==2:
                    pl.figure(4)
                    pl.subplot(sub[tn-K])
                    pl.title( '$\epsilon$: ' + str(round(epssort[q], 3)) + \
                              '  $\Omega$: ' + str(round(omgsort[q], 3)) + \
                              '  $\gamma$: ' + str(round(gamsort[q], 3)),
                               fontsize = 9)

                    pl.plot(Mmplx,Mmply,'k.')
                # >> Is traces on DEM only figure
                if p==3: pl.figure(4);pl.subplot(236)
```

```
                pl.plot((trace[:,0] -xmin), (trace[:,1]-ymin),colors[tn-K],\
                            markersize=2)
                K+=1

                for sti in stitches[q]:
                    # >> If not empty of stitches
                    pl.plot((sti[:2] - xmin), (sti[2:] - ymin), \
                                color = [0.5, 0.5, 0.5])


                pl.axis('scaled')
                pl.ylim( sp6.get_ylim() )
                pl.xlim( sp6.get_xlim() )
                pl.plot(ctx,cty, 'c*')
            pl.figure(3)

        pl.figure(1)
        sp3.xaxis.set_label_position('top')
        pl.setp(sp3.get_xticklabels(), rotation='vertical')


# >> All intrest points (local Max, DeltaSlopeMax) on Bowl (9)
# >> This one get repeated for Big Picture Figure

        sp9=pl.subplot(339)
        Size=[0,5,2]
        pl.setp(sp9.get_xticklabels(), rotation='vertical')
        colors=['ko','g.','w.']
        label = ['Slope Break','Local Maxima', 'Global Maxima']

        if combo:
            sets=[2,1]
        else: sets=[2]
        # >> repeat twice

    if images == 'Paper':
        sets = [0]
        Size = [2]
        colors=['yp','gs','bo']
        label = ['Slope Break','Local Maxima', 'Max. Elev.']
        sp2 = pl.subplot(122)


    if images:
```

```
        for p in sets:
            pl.imshow(bowl.maps[-1],cmap=clrmap,extent=lims)
            c=0

        # >> Find Interest Points, repeat for 3 types of intrest
            if combo:
                for typ in [6,7,8]: #>> Dslope, LocalMax, GlobalMax

                    ind2 = np.nonzero(Matrix[typ])[-1]
                    ind1 = np.nonzero(Matrix[typ])[-2]

                    intXs= Matrix[1][ind1,ind2] - xmin
                    intYs= Matrix[2][ind1,ind2] - ymin
                    pl.plot(intXs,intYs, colors[c],markersize=Size[p],\
                             label=label[c],markeredgewidth = 0.1)
                    c+=1

            pl.plot(ctx,cty, 'c*')
            pl.axis(lims)

            if images == 'Diagnostic':
            #>> Set current fig for next
                pl.figure(2)
                pl.xlabel('x [m]')

    if images == 'Paper':
        pl.plot(Bplx, Bply, 'ko',markersize=2, label = 'Composite')
        pl.legend(numpoints = 1, prop={'size':8},ncol=2,loc=1)

        pl.axis(lims)
        pl.xlabel('x [m]')

        print 'saving figure'
        pl.savefig(cratername +'_Planform.png',dpi=500)
        pl.savefig(cratername +'_Planform.svg',dpi=500)

    else:

# >> Plot PROFILES AND INTEREST POINTS
        if combo:
            pl.figure(5);pl.clf()
            pl.subplot(121)

        # >> Pull out x and y for easier coding
            xM = np.array(Matrix[1][[0, N/2, N/4, 3*N/4]])
```

```
    yM = np.array(Matrix[2][[0, N/2, N/4, 3*N/4]])
    rM = np.array([Matrix[0][0], -Matrix[0][N/2], -Matrix[0][N/4],\
                  Matrix[0][3*N/4]])
    hM = np.array(Matrix[3][[0, N/2, N/4, 3*N/4]])

    pl.imshow(bowl.maps[-1],cmap=clrmap,extent=lims)

# >> Horizontal Profile, from Left to Right
    x1 = np.array([xM[1][-1], xM[0][-1]])
    y1 = np.array([yM[1][-1], yM[0][-1]])

    xn,yn, pls1, plz1 =bowl.profile(x =x1, y=y1, mapnum =-1, spacing\
                                    =bowl.dpix[0]*0.5,order=prorder)
    pls1 = pls1 + rM[1][-1]

# >> Vertical Profile, from Top to Bottom
    x2 = np.array([xM[2][-1], xM[3][-1]])
    y2 = np.array([yM[2][-1], yM[3][-1]])

    xn,yn, pls2, plz2 =bowl.profile(x=x2, y =y2, mapnum =-1, spacing\
                                    =bowl.dpix[0]*0.5, order=prorder)
    pls2 = pls2 + rM[2][-1]

    colors = ['ko','g.','w.']
    procolor=['ko','g^','w*']
    q = 0

# >> Pull and plot the intrest points for each profile
    for typ in [6,7,8]:

        Mplx = xM[np.nonzero(Matrix[typ][[0,N/2,N/4,N*3/4]])]
        Mply = yM[np.nonzero(Matrix[typ][[0,N/2,N/4,N*3/4]])]
        Mplr = rM[np.nonzero(Matrix[typ][[0,N/2,N/4,N*3/4]])]
        Mplh = hM[np.nonzero(Matrix[typ][[0,N/2,N/4,N*3/4]])]

        pl.subplot(121);pl.plot(Mplx-xmin, Mply-ymin, colors[q])

        pl.subplot(122)
        pl.plot(Mplr, Mplh, procolor[q], markersize=8)

        q += 1

    pl.plot(pls1, plz1, 'm-'); pl.plot(pls2, plz2, 'c-')

    pl.subplot(121);pl.plot(x1 -xmin, y1 -ymin, 'm')
```

```
            pl.plot(x2 -xmin, y2 -ymin, 'c')
            pl.axis(lims)
            pl.savefig(cratername + '_intrest+prof.png',dpi = 250)

        pl.figure(1)
# >> Title Plot with important diagnostics


        pl.suptitle('Ejd: '+str(round(diagnostic['Ejdpdm'][0],3))+\
                    ', '+str(round(diagnostic['Ejdpdm'][2],3))+\
                    ', '+str(round(diagnostic['Ejdpdm'][1],3))+\
                    '    Ord: '+str(round(diagnostic['Ordpdm'][0],3))+\
                    ', '+str(round(diagnostic['Ordpdm'][2],3))+\
                    ', '+str(round(diagnostic['Ordpdm'][1],3))+\
                    '    IPd: '+str(round(diagnostic['IPdpdm'][0],3))+\
                    ', '+str(round(diagnostic['IPdpdm'][2],3))+\
                    ', '+str(round(diagnostic['IPdpdm'][1],3)),\
                       fontsize=8)


# >> SAVE FIGURES! then clear them to free space

        print 'Saving Figure'
        pl.figure(1);pl.savefig(cratername +'_diagns.png',dpi=250)
        pl.clf()

        if combo:
            pl.figure(2); pl.colorbar()
            pl.savefig(cratername +'_intpnt.png', dpi=250)
            pl.clf()

            pl.figure(4)
            pl.subplot(233);pl.plot(Mmplx,Mmply,'k.');pl.plot(ctx,cty, 'c*')
            pl.title( '$\epsilon$: ' + str(round(grim.eps2d, 3)))

            pl.axis('scaled'); pl.ylim(sp6.get_ylim());
            pl.xlim(sp6.get_xlim())
            pl.savefig(cratername +'_ind-traces.png', dpi=250)
            pl.clf()

        pl.figure(3)
        pl.savefig(cratername +'_all-traces.png', dpi=250)
        pl.clf()

# >> Increase Crater index
```

```
        i += 1

    # >> Save Golembase object at the end of the run
    if analysis:

        print('Pickling the result...')
        golb.pickle(gbsvname)

    # >> Save the Epdobase text file
    if savefile:
        print('Saving .txt File')
        txtf.write('++')
        txtf.close()

    # >> Return value of skiped and flagged craters
    if verbose:
        print '  *  *  *  *  *  *  *  *  *  *'
        print str(Failures) + '  CRATERS MESSED UP DETRENDING'
        print str(len(Flaged)) + '  CRATERS WERE FLAGGED'
        print str(len(Broken)) + '  CRATERS SKIPPED'
        print str(len(CntFail))+ '  COUNTOURS FAILED'


    return Flaged, Broken

#.....................................................................

#=====================================================================
# simple_method
# Makes the MERT, or a SBRT, or a detrendedRT
#
#
#=====================================================================

def simple_method(method, gdmp, Pr, center, mapnum = 0, N=512, order=0):

        # >> Set up empty trace arrays
    coords = np.zeros((N,3))

        # >> qq hold the location in the array.
    qq = -1

# >> We will take profiles N times through 360 degrees

    for j in np.linspace(0, 360, N):
```

```
        rimspot=[]
        qq +=1
                # >> Find x & y coordinates change for each degree change

        x = [center[0], center[0] + Pr*math.cos(math.radians(j))]
        y = [center[1], center[1] + Pr*math.sin(math.radians(j))]

        xn, yn, s, z = gdmp.profile(x = x, y = y, mapnum = mapnum, \
                                    spacing =  0.5* gdmp.dpix[0],order=order)

                # >> Remove the NaNs
        inds = np.nonzero(~np.isnan(z))[0]

        xn = xn[inds]
        yn = yn[inds]
        s = np.array(s)[inds]
        z = np.array(z)[inds]


# ------- Split for dif methods ------------------------------------------------

#   # # # # # # ### MAX METHOD #### # # # # # # #  #  #  #  #  #   #   #   #   #

        if method == 'max':

                # >> step through profile

            for k in range(0,len(z),1):

            # >> We don't want to plot a point if it is at the endpoint

                if k == len(z)-1:
                    break

            # >> If current z is maximum length, put it in our array
                if z[k] == max(z):

                    rimspot= k

                 # >> Stop computing once we find max
                    break
#_______________________________________________________________________________
```

```
#  #  #  # # # ### SLOPE METHOD ### # # # #  #  #  #  #   #   #   #   #   #   #

        elif method == 'slope':

            # >> Length of slope increments
            l = int(round(r/10))

    # >> start at halfway out to rim
            start= np.nonzero(s >= Hr)[0][0]

            m = 1

            for w in range(start,len(s),1):

                    # >> Find the slope break, where m changes sign

                if w+l < z.shape[0]:
                    m = (z[w+l]-z[w-l])/(s[w+l]-s[w-l])

                    if m < 0:
                        rimspot=k
                        break

                else:  # >> if segment will go pass end of profile
                    # >> i.e. if the slope doesn't change sign
                    # >> Plot at maximum z value
                    rimspot = np.nonzero(z == max(z))[0][0]
                    break
#_______________________________________________________________________________


#  #  #  # # # # ### DETRENDED METHOD #### # # # # # # #  #  #  #  #  #   #   #

        elif method == 'detrend':

## >> Make Sure Every Profile Begins at the Lowest Point (after removing nan)

            begin = np.nonzero(z == min(z))[0][0]
            xn = xn[begin:]
            yn = yn[begin:]
            s =  s[begin:]
            z = z[begin:]

            # >> Calculate the fit
            A = np.array([s, np.ones(len(x))])
```

```
            slope, intercept = np.linalg.lstsq(A.T,z)[0]

            # >> Find detrended profile
            nz = z - s*slope - intercept

            dmax = np.nonzero(nz == max(nz))[0][0]

            # >> Don't want last point, just in case
            if dmax != len(nz)-1:
                rimspot = dmax

#-------------------------------------------------------------------------------


            # >> Assign the Rim Coords

        if rimspot != []:

            coords[qq][0]= xn[rimspot]
            coords[qq][1]= yn[rimspot]
            coords[qq][2]= z[rimspot]


    return coords
```

Launching Script

```
#============================================================================
# 20121220-1100-gpudoo -launch_plex.py
# Wasp Shape Extraction Execution / Lynn Geiger
# Runs the shape extractor on the wasp files
#
# <inputs>
# User Presets:
# basepath = from where to pull the file list
# savefile = Name of epdobase formatted text savefile
# golbfile = Name of golembase save file
#
# to run the combination matrix method, set usematrix = True
# to parallelize the script, set parallelize = True
#
# </inputs>
# <products>
# produces pngs for each crater, an epdo and golembase file for all craters,
# </products>
#
# ============================================================================


import pdb
import sys
import os
import fnmatch as fm
import extract_planforms as plex
import epdobase as eb
import translate_feature_index as tf
import numpy as np
import epdobase as epdo

# -------------------- User Presets: ------------------

parampath = '/home/lynn/Thesis/python/plex_param.dex'

ep = epdo.cl_epdobase()
ep.load(parampath, format='dex')

parallelize = True
parallelize = False

viable_only = True
```

```
basepath = ep.dcty['value'][0]
savefile = ep.dcty['value'][1]
golbfile = ep.dcty['value'][2]

matVal = [float(ep.dcty['value'][5]), float(ep.dcty['value'][6]), \
              float(ep.dcty['value'][7]), float(ep.dcty['value'][8]), \
              float(ep.dcty['value'][9]), float(ep.dcty['value'][10]), \
              float(ep.dcty['value'][11]), ('True' == ep.dcty['value'][12]), \
              ('True' == ep.dcty['value'][13])]

# ------------------------------------------------

# >> Extract the list of craters

craterls = fm.filter(os.listdir(basepath),'*bowl*.pkl')

# >> Load master to get CIDs
mfidx = tf.cl_mfidx()

mfnums = []
for crater in craterls:
    mfnum = mfidx.mfnum_from_filename(crater)
    if mfnum != None:
        if viable_only:
            if mfidx.viable[mfnum] == 'True':
                mfnums.append(mfnum)
        else: mfnums.append(mfnum)

mfnums = np.array(sorted(mfnums))

# >> If you want to run a subsection of the list, use the following command:
#mfnums = np.array([32,110,168])

# >> Parallelizing terms

if parallelize:

    numslices = int(sys.argv[1])
    whichslice = int(sys.argv[2])

    sys.stdout = open(('debug_output'+str(whichslice)+'.txt'), 'w')
    sys.stderr = open(('error_plex' +str(whichslice) +'.txt'), 'w')

    numcrat= len(mfnums)/numslices
    start = whichslice * numcrat
```

```
    # >> give last slice any remaining gdmp

    if numslices == (whichslice + 1):
        mfnums = mfnums[start:]

    else:
        mfnums = mfnums[start:(start+numcrat)]

    savefile = savefile[:-4] + '_' + str(whichslice) + '.txt'
    golbfile = golbfile[:-4] + '_' + str(whichslice) + '.pkl'

# >> Find the Crater Rims using shape extractor

plex.RIM(mfnums, basepath = basepath, savefile = savefile, gbsvname = golbfile,\
     prlen = float(ep.dcty['value'][3]), N = int(ep.dcty['value'][4]), \
     prorder = int(ep.dcty['value'][14]), analysis = ('True' == ep.dcty['value'][15]),
     flag = ep.dcty['value'][16], flcritical = float(ep.dcty['value'][17]),
     maxejoff = float(ep.dcty['value'][18]),\
     NaNEjMax = float(ep.dcty['value'][19]), bdim = float(ep.dcty['value'][20]),
     EJdtcirc = np.array([float(s) for s in ep.dcty['value'][21].split(',')]),
     detrend = ep.dcty['value'][22], images = ep.dcty['value'][23], contour =
     ('True' == ep.dcty['value'][24]), \
     contfrac = float(ep.dcty['value'][25]), verbose = ('True' == ep.dcty['value'][26]), \
     startang = np.array([int(s) for s in ep.dcty['value'][27].split(',')]))
```

3.5 Extract Interest Points and Composite Rim Trace

```
# ============================================================================
# 20121002-1011-gpudoo - extract_crater_matrix.py
# Extract Characterisic Matrix from Crater / Lynn Geiger
# Imports crater gridmap and calculates a Matrix of various atributes
#
# <inputs>
# gdmp; gridmap object,
# mapnum; Which DEM map to use in analysis,
# slper = .10; Percent of Radius to calculate slope with,
# mxper = .10; Percent of Radius local maximum must be higher than,
# mingrade = .05; minimum slope change required for interest point
# minslcluster = .05; Percent of R that the slope must be changing continuously,
#                       to minimize false possitives from noise in the profile
# N = 512; Number of profiles
# spacing = If you want to do 1/2 pdim spampling instead of default [121128]
# prlen = 2.0; Length of each profile in Radii
# order = 0; Order of the spline interpolation for profile extraction
# </inputs>
#
# <products>
# One List of Matrices, including 6 values at each point in each profile,
# and 2 indice matrices, where value of one indicates a point of intrest:
#
# Matrices:
# 0-Radius; 1-X Value; 2-Y Value; 3-Z Value; 4-Profile Slope before point;
# 5-Slope Change at the point;
# 6- Index Matrix where 1= Local Max Slope Change and 0=Not;
# 7- Index Matrix where 1= Local Maxima
# 8- Index GlobalM Maxima
# 9- Profile Slope after point [added 130513]
# </products>
#
# ============================================================================


import pdb
import numpy as np
import math

def find_matrix(gdmp, mapnum, slper = .10, mxper = .10, mingrade = .05, \
                    minslcluster = .05, N = 512, spacing = 1, prlen = 1.5, \
                    order = 0):

# >> Pull out important constants
```

```
    center = [((gdmp.xext[1]+gdmp.xext[0])/2) , ((gdmp.yext[1]+gdmp.yext[0])/2)]
    half = (abs(gdmp.xext[1]-gdmp.xext[0])/2)
    if 'rad' in gdmp.metc: R = gdmp.metc['rad']
    else: R = (half/gdmp.metc['radiiToEdge'])

# >> Find out how many points will be in each slope segment

    rm = np.ceil(R*prlen)
    slopenum= int(rm * slper)
    maxnum = int(rm * mxper)
    clustersz= rm*minslcluster

# >> Create Empty Matrices, (Radius,X,Y,Z, Slope(before point),
# >>  Slope Change, Local Max Slope Change, Local Maxima, Global Max)

    RMat = np.zeros((N,int(rm/spacing+1)))
    XMat = np.zeros((N,int(rm/spacing+1)))
    YMat = np.zeros((N,int(rm/spacing+1)))
    ZMat = np.zeros((N,int(rm/spacing+1)))
    Sb4Mat = np.zeros((N,int(rm/spacing+1)))
    DSMat = np.zeros((N,int(rm/spacing+1)))
    DSIndMat=np.zeros((N,int(rm/spacing+1)))
    MaxMat = np.zeros((N,int(rm/spacing+1)))
    GMaxMat = np.zeros((N,int(rm/spacing+1)))

    SafMat = np.zeros((N,int(rm/spacing+1))) #>>[130513]

# >> Use q, instead of j, since j is in degrees not indices
    q = 0

# >> Loop through each Theta value

    for j in np.linspace(0., 360., N):

# >> Create lines along which to take a profile
        x = [center[0], center[0] + rm*math.cos(math.radians(j))]
        y = [center[1], center[1] + rm*math.sin(math.radians(j))]

# >> Extract the Profile
        xn, yn, s, z = gdmp.profile(x = x, y = y, mapnum = mapnum, spacing = \
                                        spacing, order = order)

# >> Figure out slopes before and after at each point
        # >> Set up empty variables
```

```
        slopebe4 = np.zeros((len(z)))
        slopeaft = np.zeros((len(z)))

        # >> Loop through each point
        for t in np.arange(slopenum, len(s)-slopenum,1):

# >> Find s, and z values to construct a slope
            sbe4 = np.array([s[(t-slopenum) : t], np.ones(slopenum)]).T
            zbe4 = z[(t-slopenum) : t]

            saft = np.array([s[t : (t+slopenum)], np.ones(slopenum)]).T
            zaft = z[t : (t+slopenum)]

# >> Need to Get Rid of NaNs for the least squares to work
            b4in = np.nonzero(~np.isnan(zbe4))[0]
            afin = np.nonzero(~np.isnan(zaft))[0]

# >> Skip if all are NaNs, in EITHER one [120126]
# >> also slope of 1 pnt is silly, need 3 to be reliable [130126]

            if (len(b4in)<=2) + (len(afin) <= 2): continue

            sbe4 = sbe4[b4in]
            zbe4 = zbe4[b4in]

            saft = saft[afin]
            zaft = zaft[afin]

# >> Fit linear Regression to find the slope
            be4stat = np.linalg.lstsq(sbe4, zbe4)[0]
            aftstat = np.linalg.lstsq(saft, zaft)[0]

            slopebe4[t] = be4stat[0]
            slopeaft[t] = aftstat[0]

# >> Subtract to find max slope differences
        difs= (slopebe4 - slopeaft)

# >> Make empty variables to fill in for loop
        cluster=[]
        inds=[]

# >> Find all clusters of possitive Slope change,
        # >> then the index of the local dslope maxima
```

```
# >> Loop through all points
        for val in difs:
            if val > mingrade: #Changed from 0, [12122012]
                cluster.append(val)

# >> Once value goes below zero, cluster ends
            if val <= 0:

# >> If cluster is greater than the cluster minimum length save it
                if len(cluster) > clustersz: #[12122012]
                    cluster=np.array(cluster)

# >> Pull out the Maximum Slope Change in the cluster
                    pind = (np.nonzero(difs==max(cluster))[0][0])

# >> Just double check that the possible index isn't on a NaN [130126]
                    if ~np.isnan(z[pind]):
                        inds.append(pind)

# >> Start a new cluter
                cluster=[]


# >> Find the Global Maximum

        # >> In some cases there are multiple global maxima (aka the same
        # >> elevation twice) an artifact of the profiling, but they are alway
        # >> directly next to eachother, so pick 1st to be consistent.
        gmax=np.nonzero(z[:-1]==max(z[~np.isnan(z)]))[0]
        if len(gmax)>1: gmax=gmax[0]
        if gmax==0: gmax=[]

        # >> If the gmax is the very last in the profile, it's wrong
        if gmax == (len(z)-1): gmax = []

# >> Assign Matrix Values for the azimuth
        RMat[q] = s[:]
        XMat[q] = xn[:]
        YMat[q] = yn[:]
        ZMat[q]= z[:]
        Sb4Mat[q]= slopebe4[:]
        DSMat[q]=difs[:]
        DSIndMat[q][inds]= 1
        GMaxMat[q][gmax] = 1
```

```
        SafMat[q]= slopeaft[:]

# >> Loop through all points starting mxlen/2 in and ending mxlen/2 from end
        for k in np.arange(maxnum,len(z)-maxnum,1):
            chnkb4 = z[(k-maxnum) : k]
            chnkaf = z[k+1 : (k+maxnum+1)]
# >> If the point we're on is greater than those before and after then save it
            if z[k] > max(chnkb4):
                if z[k] > max(chnkaf):
                    MaxMat[q][k]=1

# >> Increase the Index of the Matrix
        q += 1

# >> Combine all Matrices

    Matrix=[]

    Matrix.append(RMat)
    Matrix.append(XMat)
    Matrix.append(YMat)
    Matrix.append(ZMat)
    Matrix.append(Sb4Mat)
    Matrix.append(DSMat)
    Matrix.append(DSIndMat)
    Matrix.append(MaxMat)
    Matrix.append(GMaxMat)

    Matrix.append(SafMat)

# >> END
    return Matrix


#================================================================================
#
# Compile Composite Rim Trace - Lynn Geiger
# Uses Matrix of interest points to find an optimal rim trace
#
# <inputs>
# Matrix   = found in above method
#
# stangle  = Angles at which to begin the traces (in degrees)
#
# rdis     = Radial Discontinuity Allowed for GM trace (m)
```

```
#
# RAND     = Radial Allowed Neighbor Distance (m).  Leave Empty to use the
#            other method
#
# AAND     = Angular Allowed Neighbor Distance (degrees)
#
# siftRedun= Sift out the redundant traces, only return unique candidates
#
# contloop = continue around the crater until the current point has been found
#            before
#
# kpstitch = option to return the 'stitches', where AAND skips (T/F)
#
# </inputs>
# <products> A list of rim candidates, and a list of corresponding omega values
# </products>
#
#==========================================================================

def composite_rim(Matrix, stangle=[0,90,180,270], rdis=1., RAND=[], AAND = 0,\
                      siftRedundant=True, contloop=True, kpstitch = False):

# >> Need to import for this method
    import random, copy

    # >> Create Master Matrix, weighted towards max
    Mmaster= 4*Matrix[8]+2*Matrix[7]+Matrix[6]

    lMat =len(Mmaster)

    # >> Convert from Degrees to Spokes
    AAND = np.ceil(AAND * lMat / 360.)
    stpnts = (lMat*np.array(stangle))/360

    # >> Create empty arrays to fill in

    GMR = np.zeros(lMat)
    IndGM= np.zeros(lMat)

    OptRims=[]
    Omegas =[]
    Lilo = [] # >> Where we keep Stitch ;)

    # >> Find Indexes of the global Maxima
    # >> Put inside for loop in case there is no GM in that profile
```

```
    for i in range(len(Matrix[8])):
        gm =np.nonzero(Matrix[8][i])[0]
        if len(gm)==1:
            IndGM[i] = gm
            GMR[i] = Matrix[0][i][gm]

# >> Start counter:
    pnum=0

    for strt in stpnts:

        # >> Go counter / clockwise for each starting point
        ccw= np.array(range(strt,lMat) + range(0,strt))

        cw = np.array(range(strt,-1,-1) + range(lMat-1,strt,-1))

        loops = [cw, ccw]

        # >> Go each direction
        for path in loops:

            print 'On path ' + str(pnum)

# >> Create empty rim trace for this path
            RIM = np.zeros([lMat,3])

            Stitch= []

# >> Find Last point of loop, to have reference for start
            LastR = GMR[path[-1]]
            Lind  = IndGM[path[-1]] #>>Last index (for AAND stitch)

            # >> If the last profile is empty...
            uhoh = -2
            # >> Go one point infront of that
            while LastR==0:
                LastR = GMR[path[uhoh]]
                Lind = IndGM[path[uhoh]]
                uhoh -=1


            Om = 0
            k = 0
            gap = 0
```

```
            before = False

            # >> Keep going until you hit a place you've been to before
            while before == False:

                # >> What is the index:
                i = path[k]

                # >> Is the radius of the global max at that index within reason
                if (GMR[i] <= np.ceil(LastR +rdis))*\
                        (GMR[i] >= np.floor(LastR -rdis)):

                    choi = IndGM[i]

                # >> If not, look for other intrest-points
                else:
                    # >> Keep track of # off the max method
                    Om +=1

                    # >> Where are possible rims:
                    posI = np.nonzero(Mmaster[i])[0]

                    # >> Apparently there are profiles without intrest...
                    if len(posI)==0:
                        print 'At '+ str(i*360/512)+' degrees, there\'s no in'+\
                            'trest points'

                    # >> Assign gap and move on
                        gap +=1
                        k += 1
                    # >> If we are at the end start over
                        if k>=len(path):
                            k=0
                        continue

                    cand = [] # reset candidates
                    choi = [] # and choice

#   #   #   #  #   # >> If using the RAND AAND Method #  #  #  #   #    #    #
                    if RAND:

                        adis = 0 # reset angular distance
                        ak = k # index starts at i
                        aind= path[ak]
```

```
                    sti = []
                    while adis <= AAND: # Keep below max angle

                # >> Find possible candidates in next spoke
                        posI = np.nonzero(Mmaster[aind])[0]

                # >> Find the Radial distance of the candidates from Lastpnt
                        posR = abs(Matrix[0][aind][posI] - LastR)

                # >> Candidates must be within RAND
                        cand = posI[np.nonzero(posR <= RAND)]

                # >> If candidates is empty, random.py will raise an error
                        try:
                    # >> Pick a choice at random
                            choi = random.choice(cand)
                    # >> Then stop the while loop
                            break

                    # >> If not, go to next spoke
                        except:
                            adis += 1
                            ak += 1
                            if ak == len(path):
                                ak = 0
                            aind = path[ak]


                    if (adis > 0)*(adis <=AAND): # If something was skipped:
                        # >> add to stitch
                        sti  = np.array([Matrix[1][i-1][Lind], \
                                  Matrix[1][aind][choi],Matrix[2][i][Lind],\
                                  Matrix[2][aind][choi]])

                        # >> Sti is [x1,x2,y1,y2]
                     # >> Update i & k if something is skipped to prof we are on
                            i = aind
                            k = ak
                            Stitch.append(sti)


            # >> if doesnt work, just pick the closest (by not assiging choi)
#   #    #   #   #   #   # >> Not RAND AAND Method <<  #  #   #   #    #   #   #
                    if choi == []:
```

```
                    posI = np.nonzero(Mmaster[i])[0]
                # >> Find the Radial distance of the candidates from Lastpnt
                    posR = abs(Matrix[0][i][posI] - LastR)

                    # >> Candidate is closest
                    cand = posI[np.nonzero(min(posR)==posR)[0]]

                    # >> if multiple closests pick optimal one
                    if len(cand)>1:
                        opti = Mmaster[i][cand]
                        choi = cand[np.nonzero(max(opti)==opti)[0]]

                    else: choi = cand
#-------------------------------------------------------------------------------


            # >> Check if we've been here before
            if RIM[i][0] == Matrix[1][i][choi]:
                if RIM[i][1] == Matrix[2][i][choi]:
                    if RIM[i][2] == Matrix[3][i][choi]:
                        # Iff the same, end loop
                        before = True

            # >> Put this rim point in the trace
            RIM[i][0] = Matrix[1][i][choi]
            RIM[i][1] = Matrix[2][i][choi]
            RIM[i][2] = Matrix[3][i][choi]

            # >> Assign New Last R
            LastR = Matrix[0][i][choi]
            Lind = int(choi)

            # >> increment the path loop
            k += 1
            # >> if at the end of the path, start over
            if k>=len(path):
                k=0
                # >> If only going around once, stop the loop
                if contloop == False:
                    before = True

        # >> When finished, divide the points off MaxMethod by total
        Om = Om / float(lMat) # (Omega)

        aprim = True
```

```
        # >>if not the first, check if it matches any previous rim:
        if len(OptRims)>0:
            for j in range(len(OptRims)):
                if sum(RIM[:,0] != OptRims[j][:,0])==0:
                    print 'Trace is Same as Trace ' + str(j)

            # >> added option to print all rims or sift the repeates
                    if siftRedundant:
                        aprim = False
                    break

        # >> Append the Rim Trace
        if aprim:
            OptRims.append(copy.deepcopy(RIM))
            Omegas.append(Om)
            if kpstitch:
                Stitch = np.array(Stitch)
                Lilo.append(Stitch)
        pnum+=1

# >> Return the rim traces and omegas
if kpstitch:
    return OptRims, Omegas, Lilo, gap
else: return OptRims, Omegas, gap
```

3.6 Launch Profile Metric Extractor

```
# ============================================================================
# 20131130-1500-gpudoo - launch_prex.py
# Launch profile extractor / Lynn Geiger
#
#=============================================================================

import pdb                              # Python Debugger, use pdb.set_trace()
import os, sys
import numpy as np
import epdobase as eb
from omnigenus import unpickle          # Reading in pickle files' gdmp
import pickle                           # Saving pickle files
import extract_profile_metrics as prex
import fnmatch as fm
import epdobase as epdo

#.............................................................................

parampath = '/home/lynn/Thesis/python/prex_param.dex'

#.............................................................................

ep = epdo.cl_epdobase()
ep.load(parampath, format='dex')

parallelize = True
parallelize = False

gbname = fm.filter(os.listdir(ep.dcty['value'][0]), ep.dcty['value'][3] + \
                          '*_Comp*.pkl')[0]

gbsavename = ep.dcty['value'][2]

gb = unpickle(gbname)

crls = []

for db in gb.db:
    crls.append(db['mfnum'])

crls = np.array(crls)

if parallelize:
```

```
    numslices = int(sys.argv[1])
    whichslice = int(sys.argv[2])

    sys.stdout = open(('debug_prex_output'+str(whichslice)+'.txt'), 'w')
    sys.stderr = open(('error_prex'+str(whichslice)+'.txt'), 'w')

    numcrat= len(crls)/numslices
    start = whichslice * numcrat

    # >> give last slice any remaining gdmp

    if numslices == (whichslice + 1):
        crls = crls[start:]

    else:
        crls = crls[start:(start+numcrat)]

    gbsavename = gbsave[:-4] + '_' + str(whichslice) + '.pkl'

prex.avgprof(gb, crls, pklpath = ep.dcty['value'][1], gbsavename=gbsavename, \
     mxslp = float(ep.dcty['value'][4]), rpsz = float(ep.dcty['value'][5]), \
     order = int(ep.dcty['value'][6]), ejR = float(ep.dcty['value'][7]), samples\
                 = np.array([float(s) for s in ep.dcty['value'][8].split(',')]),
 fitnan = int(ep.dcty['value'][9]), skipnan = int(ep.dcty['value'][10]), rrmWPer
 = float(ep.dcty['value'][11]), Ejdtcirc = np.array([float(s) for s in ep.dcty['
value'][12].split(',')]), ejskp = float(ep.dcty['value'][13]), stretch = ('True'
 == ep.dcty['value'][14]), zNadstart = ('True' == ep.dcty['value'][15]), figures
 = ep.dcty['value'][16], wchprofiles = ep.dcty['value'][17])
```

3.7 Extract Profile Metrics

```
#==========================================================================
# 20121130-1327-gpudoo -profile_extractor.py
# Crater Profile Extractor / Lynn Geiger
# Computes various radial profiles for a crater.
#
# <inputs>
# gb        = golmbase object
# rpsz      = rim profile size, as fraction of radius
# pklpath   = path to crater pickles
# wchprofiles= 'ALL', 'bowl' or 'ejct'
# order     = Profile spline interpretation order
# ejR       = Radius to calculate Ejecta profiles to
# samples   = where to sample 'Cardinal Profiles'
# stretch   = True=Strech the profiles to all be the same size, False - Add NaNs
# gbsavename= name to save the new golembase as
# rrmet     = Rim Roundness Metric, 'Curvature' 'Slope'
# rrmWPer   = Percent at which the RimRoundnessMetric 'Width' is calculated.
#             5% for K, 50% for Slope change.
# figures   = 'Diagnostic' or 'Paper'
# </inputs>
#
# <products>
# Figure of all profile diagnostics and info
# </products>
#
#==========================================================================

import pdb                                  # Python Debugger, use pdb.set_trace()
import pylab as pl
import math, random
import numpy as np
from omnigenus import unpickle              # Reading in pickle files' gdmp
import detrend_gridmaps as dt
import golbase as gb
import translate_feature_index as tf

def avgprof(golb, crls, pklpath, wchprofiles ='ALL', order =1, ejR =3, samples=\
                [1/8., 3/8.], stretch = True, zNadstart = False, fitnan=50, \
                skipnan=100, rpsz = 1/5., gbsavename ='gb_rim+profiles.pkl',\
                rrmet = 'Curvature', rrmWPer = 0.05, figures = 'Diagnostic',\
                mxslp = .1, Ejdtcirc = np.array([3.0,3.5]), ejskp=0.25):
```

```
# >> find how many craters
Cr= len(crls)

mfidx = tf.cl_mfidx()

print 'Total Craters: ' +  str(Cr)

if gbsavename:
    newGB = gb.cl_golbase()

# >> Loop through each crater in golb

i = 0

for crater in crls:
    # >> Flexible to type of crls
    crater = str(crater).zfill(3)
    numb = int(crater)

    print str(i*100/Cr) +'%'

# >> Initial set up:

    # >> reset incase changed during loop
    kind = wchprofiles

    # >> select the database
    for db in golb.db:
        if db['mfnum'] == crater:
            database= db
            break

    # >> find crater name
    crid = mfidx.fname_string_from_mfnum(numb)
    cname = 'mfnum-' + crater + '_' + crid
    print 'Processing Crater ' + cname

    # >> Make sure ejecta Dem is working
    isej = database['ejDem']
    if not isej:
        print 'Ejecta DEM is unusable'

    # >> NaNs and Zeros already removed
    rim = database['trace']
```

```
if zNadstart: cnt = database['xyNad']
else: cnt = database['geocnt']

# >> Find avg R, max, min, and R of each point
R = database['diam'] / 2.0
radii= ((rim[:,0]-cnt[0])**2 +(rim[:,1]-cnt[1])**2) **0.5
maxR = max(radii)
minR = min(radii)

# >> Set Up Variables to fill
bproS = []; bproZ = []
MaxSl = []

blogstats=[]
blogZ = []; blogR = []

eproS = []; eproZ = []

elogstats=[]
elogZ = []; elogR = []

rimpS = []; rimpZ = []
RRMet = []
RRdk  = []

cardinal= []
ccoord = []

k = 0

# >> Find where the cardinal profiles are:
samppnt= (len(rim)*np.array(samples)).astype(int)

# >> Import the DEMs for retriving Profiles,
# >> Detrend Bowl based on rimm and ejct based on circles

bowl = unpickle(pklpath + database['bowlFile'])
# >> need original z to detrend (cant use saved z's)
xd, yd, sd, zd = bowl.profile(x = rim[:, 0], y = rim[:, 1], mapnum = 0,\
                                  order=order)

dt.clip(bowl, mapnum = 0, x = xd, y = yd, z = zd)

# >> Equivalent to sampling at dpix resolution on smallest R
numsamp = round(2*minR/bowl.dpix[0])
```

```
        slprolen= int(numsamp * mxslp)
        hslp = slprolen / 2
        # >> If don't want to stretch, need to know length of longest profile,
        # >> to be able to tack on NaNs to shorter Profiles
        blprolen = round(2*maxR/bowl.dpix[0])+1

        # >> If Ejecta dem works, load and process it.

        if isej:
            ejct = unpickle(pklpath + database['ejctFile'])
            eoff = database['ejoffset']
            ejcnt= database['ejcnt']

            dt.circles(ejct, mapnum = 0, rs = Ejdtcirc)

            ejprolen = round(ejR*2*maxR/ejct.dpix[0])+1
            ejtsamp = round(2*minR/ejct.dpix[0]*ejR)
            demxmin = ejct.xext[0]
            demymin = ejct.yext[0]
            limit = (0, ejct.xext[1] - demxmin, 0, ejct.yext[1] - demymin)

        # >> if Ejecta is not working only process the bowl
        else:
            kind = 'bowl'
            demxmin = bowl.xext[0]
            demymin = bowl.yext[0]
            limit = (0, bowl.xext[1] - demxmin, 0, bowl.yext[1] - demymin)


#_______________________________________________________________________________
        print 'finding profiles'

        # >> Step through spokes
        for pnt in rim:

#  ____________________________ B O W L  ______________________________________

            if kind == 'bowl' or kind == 'ALL':
                # >> Start & end points:
                bx = [cnt[0], pnt[0]]
                by = [cnt[1], pnt[1]]

                if stretch:
                    # >> Keep all profiles the same length
                    bx = np.linspace(cnt[0], pnt[0], numsamp)
```

```
            by = np.linspace(cnt[1], pnt[1], numsamp)
            xn, yn, bs, bz = bowl.profile(x = bx, y = by, mapnum = -1, \
                                spacing = None,order=order)

        # >> Find lowest point in the profile and start there
            nanper = sum(np.isnan(bz))*100 / len(bz)
            if nanper >= 90:
                print 'Bowl NaN ('+str(nanper)+'%),Skipping profile '\
                    +str(k)
            else:
                strt2 = np.nonzero(min(bz[~np.isnan(bz)])== bz)[-1][-1]

                bs[:(len(bs)-strt2)] = bs[strt2:]
                bs[(len(bs)-strt2):] = np.nan

                bz[:(len(bz)-strt2)] = bz[strt2:]
                bz[(len(bz)-strt2):] = np.nan


            nanper = sum(np.isnan(bz))*100 / len(bz)

        else:
            xn, yn, bs, bz = bowl.profile(x = bx, y = by, mapnum =-1,\
                                spacing = bowl.dpix[0]*0.5,order=order)

            # >> Start at minimum
            bs = bs[np.nonzero(min(bz[~np.isnan(bz)])==bz)[0][0] :]
            bz = bz[np.nonzero(min(bz[~np.isnan(bz)])==bz)[0][0] :]

            nanper = sum(np.isnan(bz))*100 / len(bz)
            # Tack NaN onto end of short profiles, to avg them later
            if len(bz) < blprolen:
                dif = blprolen - len(bz)
                tail= np.zeros(dif)
                tail[:]=np.nan
                bz = np.append(bz,tail)
                bs = np.append(bs,tail)

        if nanper >= fitnan:
            print 'Bowl NaN ('+str(nanper)+'%), Skipping profile ' +str(k)

        else:
        # >> Normalize and start at 0

            bz = (bz - min(bz[~np.isnan(bz)])) / radii[k]
```

```
                bs = bs / radii[k]


            # >> Find the MAX slope
                slopes = []

                for t in np.arange(hslp, (len(bs) - hslp), (2*hslp)):
                    sbs = np.array([bs[(t -hslp) : (t +hslp)], \
                                    np.ones(2*hslp)]).T
                    sbz = bz[(t-hslp) : (t+hslp)]
                    if sum(~np.isnan(sbz))>1:
                        slp = (np.linalg.lstsq(sbs[~np.isnan(sbz)],\
                                               sbz[~np.isnan(sbz)])[0][0])
                    # >> Conver to angle
                        ang = math.degrees(math.atan(slp))
                        slopes.append(ang)

                    else: slopes.append(0)
            # >> Drop First Point to avoid log(0) -> infinity

                logbz=np.log10(bz[1:])
                logbs=np.array([np.log10(bs[1:]), np.ones(len(bs)-1)]).T

                logbz[(abs(logbz) == np.inf)] = np.nan

            # >> Some have NaN, which screw up log fit (Wasp especially)
                ind = np.nonzero(~np.isnan(logbz))

            # >> Only calculate if low NaN levels
            # >> Calculate Log Fit for individual profiles
                bstats= np.linalg.lstsq(logbs[ind], logbz[ind])[0]
                blogstats.append(bstats)

            # >> Add to lists
                bproZ.append(bz)
                bproS.append(bs)

                MaxSl.append(max(slopes))

                blogZ.append(logbz)
                blogR.append(logbs)


#  ____________________________ R I M ____________________________________________
```

```
# >> take Profile Right at Rim to find Curvature
if stretch:
    Brarray = np.linspace((1 -0.5 *rpsz), 1, numsamp *0.5*rpsz)
    Frarray = np.linspace(1, (1 +0.5 *rpsz), numsamp *0.5*rpsz)
    rsp = None
else:
    Brarray = np.array([(1 - 0.5 * rpsz), 1])
    Frarray = np.array([1, (1 + 0.5 * rpsz)])
    rsp = bowl.dpix[0] * 0.5

# >> Find coordinates
Brimx = cnt[0]+(pnt[0]-cnt[0])*Brarray
Brimy = cnt[1]+(pnt[1]-cnt[1])*Brarray

Frimx = cnt[0]+(pnt[0]-cnt[0])*Frarray
Frimy = cnt[1]+(pnt[1]-cnt[1])*Frarray

xn, yn, Brims, Brimz = bowl.profile(x =Brimx, y =Brimy, mapnum =-1,\
                                    spacing = rsp,order=order)

xn, yn, Frims, Frimz = bowl.profile(x =Frimx, y =Frimy, mapnum =-1,\
                                    spacing = rsp,order=order)


Brims = -1*(Brims[::-1]) / radii[k]; Frims = (Frims) / radii[k]
Brimz = (Brimz) / radii[k]; Frimz = (Frimz) / radii[k]

# >> Recalibrate so peak is at 0, in middle, and normalize
rimz = np.append(Brimz, Frimz)
rims = np.append(Brims, Frims)

rimpZ.append(rimz)
rimpS.append(rims)

# >> Save profiles
if not stretch:
    if len(rimz) < np.ceil(blprolen/4):

        dif = np.ceil(blprolen/4) - len(rimz)

        tail= np.zeros(np.ceil(dif/2.0))
        head= np.zeros(np.floor(dif/2.0))

        tail[:]=np.nan
```

```
            head[:]=np.nan

            rimz = np.append(head, rimz)
            rims = np.append(head, rims)

            rimz = np.append(rimz,tail)
            rims = np.append(rims,tail)


# >> If high nan level, skip it, check both bowl and flank-ward sides
# >> plus 1 because diff drops the first point, also if NaN comes before
# >> a point, that also becomes NaN, i.e. if every other is NaN the whole
# >> string becomes NaNs
    nanperBW = (sum(np.isnan(np.diff(Brimz)))+1)*100 / len(Brimz)
    nanperFW = sum(np.isnan(np.diff(Frimz)))*100 / len(Frimz)

    CalcBW = True; CalcFW = True

    if nanperBW >= fitnan:
        print 'BW-Rim NaN ('+ str(nanperBW) +'%) skip profile ' + str(k)
        CalcBW = False
    if nanperFW >= fitnan:
        print 'FW-Rim NaN ('+ str(nanperFW) +'%) skip profile ' + str(k)
        CalcFW = False

    if rrmet == 'Slope':
        dk = []
        rrmetpl = []
        rrs = []
        for K in [0,1]:

            rims = [Brims, Frims][K]
            rimz = [Brimz, (-1)*Frimz][K]
            calc = [CalcBW,CalcFW][K]

            if calc:
                slp = np.array(np.zeros(len(rims)))

            # >> Remove 1st to not be included in Calc
                if K ==1: slp[0] = (Brimz[-1]-Frimz[0])
                else: slp[0] = np.nan

        # >> Calculate decay width Flank and Bowl-wards
                slp[1:]= (np.diff(rimz)) / (np.diff(rims))
                maxind = np.nonzero(max(slp[~np.isnan(slp)])==slp)[0][0]
```

```
                    dkper = slp[maxind] * rrmWPer
                    dwind = np.nonzero((slp <= dkper) * (~np.isnan(slp)) * \
                                       (abs(rims) < (abs(rims[maxind]))))[0]
                    if len(dwind)>= 1:
                        dk.append(max(abs(rims[dwind])))
                    else:

                        dk.append(np.nan)

                    rrmetpl = np.append(rrmetpl, slp)
                    rrs = np.append(rrs, rims)
                else:
                    dk.append(np.nan)
                    nls = np.zeros(len(rims)); nls[:] = np.nan
                    rrmetpl = np.append(rrmetpl, nls)
                    rrs = np.append(rrs, nls)

        # >> Save
            RRMet.append(np.array([rrs,rrmetpl]).T)
            RRdk.append(np.array(dk))


#  ___________________________ E J E C T A ______________________________________

         if kind == 'ejct' or kind == 'ALL':

            # >> Since Flank prof is last can modify pnt here
            pnt[0] = pnt[0] - eoff[0]
            pnt[1] = pnt[1] - eoff[1]

            # >> Start and End Pnts
            ex = [pnt[0], (ejcnt[0] + ejR*(pnt[0]-ejcnt[0]))]
            ey = [pnt[1], (ejcnt[1] + ejR*(pnt[1]-ejcnt[1]))]

            # Find Profile
            if stretch:
                ex = np.linspace(pnt[0], (ejcnt[0] + ejR* (pnt[0] - \
                                           ejcnt[0])), ejtsamp)
                ey = np.linspace(pnt[1], (ejcnt[1] + ejR* (pnt[1] - \
                                           ejcnt[1])), ejtsamp)
                xn, yn, es, ez = ejct.profile(x =ex, y =ey, mapnum = -1, \
                                     spacing = None, order = order)

                nanper = sum(np.isnan(ez)) *100 / len(ez)
```

```
    if nanper >= fitnan:
        print 'Flank NaN '+str(nanper)+'%, skip profile '+str(k)
        k += 1
        continue

    # >> FIND Lowest point of the Ejecta and end there
    end2 = np.nonzero(min(ez[~np.isnan(ez)])==ez)[0][0]

    es[end2:] = np.nan
    ez[end2:] = np.nan

    if sum(~np.isnan(ez)) <= 1:
        print 'Flank end '+str(end2)+' all-nan skip profile '\
            +str(k)
        k +=1
        continue

# >> If Strech is Off:
else:
    xn, yn, es, ez = ejct.profile(x =ex, y =ey, mapnum = -1,\
                        spacing = 0.5*ejct.dpix[0],order=order)

    nanper = sum(np.isnan(ez)) *100 / len(ez)
    if nanper >= skipnan:
        print 'Flank NaN '+str(nanper)+'%, skip profile ' +str(k)
        k += 1
        continue
    # >> Start at lowest

    es= es[: (np.nonzero(min(ez[~np.isnan(ez)])==ez)[0][0]+1)]
    ez= ez[: (np.nonzero(min(ez[~np.isnan(ez)])==ez)[0][0]+1)]

# Make all profiles the same length, by tacking on NaNs

    dif = ejprolen - len(ez)
    tail= np.zeros(dif)
    tail[:]=np.nan
    ez = np.append(ez,tail)
    es = np.append(es,tail)


# >> Normalize Profiles
ez = (ez - min(ez[~np.isnan(ez)]) + 1) / radii[k]
es = (es + radii[k]) / radii[k]
```

```
        # >> According to results, any transition between fits occurs below an
        # >> elevation of 1% R. In order to fit to t flank and not the distal
        # >> toe, we crop the profile of elevations less than 0.01

                cropi = np.nonzero(ez < 0.01)[0]

                if len(cropi) >= 1:
                    ez[cropi[0]:] = np.nan
                    es[cropi[0]:] = np.nan

                # >> Find Log profiles

                ejlen = sum(~np.isnan(ez))/(ejtsamp/ejR)

                if ejskp <= ejlen:
                    logez=np.log10(ez)
                    loges=np.array([np.log10(es), np.ones(len(es))]).T

                # >> Some have NaN, which screw up log fit (Wasp especially)
                    ind = np.nonzero(~np.isnan(logez))[0]

                    if len(ind) > 3:
                        estats= np.linalg.lstsq(loges[ind], logez[ind])[0]

                    elogstats.append(estats)

                # >> Add to lists
                    eproZ.append(ez)
                    eproS.append(es)

                    elogZ.append(logez)
                    elogR.append(loges)

                else:
                    print 'Viable flank: ' + str(ejlen) + '  R'

#  ------------------------------------------------------------------------

            # >> Get the 'Cardinal Profiles'

            if (k in samppnt) * (kind=='ALL'):

                # >> Calc start and end
                x =  np.array([(ejcnt[0] + ejR * (pnt[0] - ejcnt[0])),\
                          (ejcnt[0] - ejR * (pnt[0] - ejcnt[0]))])
```

```
                y =  np.array([(ejcnt[1] + ejR * (pnt[1] - ejcnt[1])),\
                          (ejcnt[1] - ejR * (pnt[1] - ejcnt[1]))])

                # >> Save coords for ploting on map view
                ccoord.append(np.array([x, y]))

                # >> Get profile
                xn, yn, cs, cz = ejct.profile(x = x, y = y, mapnum = -1, \
                                 spacing =  0.5 * ejct.dpix[0],order=order)

                # >> add profile to list
                cardt= np.array([cs,cz])
                cardinal.append(cardt)

    # >> Where in circle
            k+=1

# --------------------------profile loop end-----------------------------------



#  ________________________ C A L C U L A T I O N S ____________________________


        if kind== 'bowl' or kind== 'ALL':


            # >> Turn into Arrays for Processing
            bproZ = np.array(bproZ)
            bproS = np.array(bproS)

            MaxSl = np.array(MaxSl)

            blogstats= np.array(blogstats)
            blogZ = np.array(blogZ)
            blogR = np.array(blogR)

    # >> Find the Averages with Masked arrays to remove NaNs

            if len(blogstats) > (0.05 * len(rim)):
            # >> Log Statistics:
                BStatAvg= sum(blogstats)/len(blogstats)

            # >> Find max of all profiles, and average max + std
                MasMax = max(MaxSl)
```

```
    AvgMax = sum(MaxSl)/len(MaxSl)
    stdMax = ((sum(MaxSl**2)/len(MaxSl)) - AvgMax**2)**0.5

# >> Bowl Log Profile Average

    BLPAS = np.mean(np.ma.masked_array(blogR[:,:,0], \
                            np.isnan(blogR[:,:,0])),axis=0)
    BLPAZ = np.mean(np.ma.masked_array(blogZ, np.isnan(blogZ)),\
                   axis=0)


# >> Calculate Standard Deviation: sig^2 = <x^2> - <x>^2

# sigma bowl log profile

    sigBLPS = (np.mean(np.ma.masked_array(blogR[:,:,0] **2, \
                np.isnan(blogR[:,:, 0])), axis = 0) -BLPAS**2)**0.5

    sigBLPZ = (np.mean(np.ma.masked_array(blogZ **2, \
                np.isnan(blogZ)), axis = 0) - BLPAZ**2)**0.5




    keepind = np.nonzero(~(BLPAS.mask + BLPAZ.mask))
    BLogProAvg = np.array([BLPAS[keepind], BLPAZ[keepind]]).T

    keepind = np.nonzero(~(sigBLPS.mask + sigBLPZ.mask))
    sigBLP = np.array([sigBLPS[keepind], sigBLPZ[keepind]]).T

    # sigma in log fit statistics
    sigB = ((sum(blogstats**2)/len(blogstats)) - BStatAvg**2)**0.5

else:
    BStatAvg = [0,0]; BLogProAvg= np.zeros([2,2]);
    sigBLPS =[0]; sigBLPZ=[0]; sigB = [0,0]
    MasMax = 0; AvgMax = 0; stdMax = 0

# >> Bowl Profile

if len(bproZ) > (0.05 * len(rim)):
    bowlavgS = np.mean(np.ma.masked_array(bproS,np.isnan(bproS)),\
                            axis=0)
    bowlavgZ = np.mean(np.ma.masked_array(bproZ,np.isnan(bproZ)),\
                            axis=0)
```

```
            # >> Get rid of masked elements, (where NaN for all profiles)

                keepind = np.nonzero(~(bowlavgS.mask + bowlavgZ.mask))
                bowlavg = np.array([bowlavgS[keepind], bowlavgZ[keepind]]).T

            else: bowlavg = np.zeros([2,2])

#.....................................................................
            # >> Append Values To the database:
            database['BowlExp'] = BStatAvg[0]
            database['BExpStD'] = sigB[0]

            database['BowlCoef']= BStatAvg[1]
            database['BCoefStD']= sigB[1]

            database['AbsMaxS'] = MasMax
            database['AvgMaxS'] = AvgMax
            database['stdAMxS'] = stdMax
#.....................................................................



#:::::::::::::::::::::::::# >> RIM SHARPNESS << #:::::::::::::::::::::::::::::

        # >> Turn into arrays
        rimpZ= np.array(rimpZ)
        rimpS= np.array(rimpS)
        rimavg = np.zeros([len(rimpZ[0]),2])

        if (np.sum(~np.isnan(RRdk),axis=0) < (0.05*len(rim))).all:
            RRdk = np.ma.masked_array(RRdk,np.isnan(RRdk))

            RRMetAv = np.mean(np.ma.masked_array(RRMet,np.isnan(RRMet)), axis=0)
            DecayW = np.mean(RRdk, axis = 0)
            sdDecay= ((np.sum(RRdk **2, axis =0) / np.sum(~np.isnan(RRdk), \
                                    axis=0)) - DecayW **2) **0.5

            keepind = np.nonzero(~(RRMetAv[:,0].mask + RRMetAv[:,1].mask))
            RRMetAv = RRMetAv[keepind]

        else:
            RRMetAv = np.zeros([2,2])
            DecayW[np.nan,np.nan]; sdDecay[np.nan,np.nan]

        if (len(rimpS)) > (0.05*len(rim)):
```

```
        # >> Calc Averages

            rimavg[:,0] = np.mean(np.ma.masked_array(rimpS,np.isnan(rimpS)),\
                                  axis=0)
            rimavg[:,1] = np.mean(np.ma.masked_array(rimpZ,np.isnan(rimpZ)),\
                                  axis=0)

        # >> Remove Masked Elements
            rimavg = rimavg[np.nonzero(~((rimavg[:,0]==0)*(rimavg[:,1]==0)))[0]]

        else: rimavg = np.zeros([2,2])
        # >> Find Kurv of Avg rim
        RRofAvg = np.array([rimavg[:,0],np.zeros(len(rimavg[:,0]))]).T

#.....................................................................

        # >> Append Values to Database

        database['RimDecayBw'] = DecayW[0]
        database['RDecayBwStD']= sdDecay[0]

        database['RimDecayFw'] = DecayW[1]
        database['RDecayFwStD']= sdDecay[1]


#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
            # >> Ejecta << #


            # >> if at least 5% of the ejecta profiles are valid,
                # >> then calculate stats

        if (kind == 'ejct' or kind == 'ALL')*(len(eproZ) > (0.05*len(rim))):

            # >> Turn in Arrays
            eproZ = np.array(eproZ)
            eproS = np.array(eproS)

            elogstats= np.array(elogstats)
            elogZ = np.array(elogZ)
            elogR = np.array(elogR)

        # >> Calculate Averages
            # >> Ejecta profile avg
            ejctavgS =np.mean(np.ma.masked_array(eproS, np.isnan(eproS)),axis=0)
```

```
            ejctavgZ =np.mean(np.ma.masked_array(eproZ, np.isnan(eproZ)),axis=0)

            # >> Remove Masked Elements:

            keepind = np.nonzero(~(ejctavgS.mask + ejctavgZ.mask))
            ejctavg = np.array([ejctavgS[keepind], ejctavgZ[keepind]]).T

            if len(elogstats) > (0.05*len(rim)):
            # >> Log Fit Stats
                EjStatAvg= sum(elogstats)/len(elogstats)

            # >> avg Log Profile
                ELPAS = np.mean(np.ma.masked_array(elogR[:,:,0], \
                                        np.isnan(elogR[:,:,0])), axis = 0)
                ELPAZ = np.mean(np.ma.masked_array(elogZ, \
                                        np.isnan(elogZ)), axis = 0)

        # >> Standard Deviation
            # >> Log Fit Stats
                sigEj = ((sum(elogstats**2)/len(elogstats)) - EjStatAvg**2)**0.5

            # >> Log Profile
                sigELPS = (np.mean(np.ma.masked_array(elogR[:,:, 0] **2,\
                               np.isnan(elogR[:,:,0])),axis=0) -ELPAS**2)**0.5

                sigELPZ = (np.mean(np.ma.masked_array(elogZ **2, \
                               np.isnan(elogZ)), axis = 0) - ELPAZ**2)**0.5


                keepind = np.nonzero(~(ELPAS.mask + ELPAZ.mask))
                EjLogProAvg = np.array([ELPAS[keepind], ELPAZ[keepind]]).T

                keepind = np.nonzero(~(sigELPS.mask + sigELPZ.mask))
                sigEjLP = np.array([sigELPS[keepind], sigELPZ[keepind]]).T

            else:
                EjStatAvg=[np.nan,np.nan]; sigEj=[np.nan,np.nan]
                EjLogProAvg = np.zeros([2,2])

#.....................................................................

            # >> Append Values To the database:
            database['EjctExp'] = EjStatAvg[0]
            database['EExpStD'] = sigEj[0]
```

```
            database['EjctCoef']= EjStatAvg[1]
            database['ECoefStD']= sigEj[1]

        else:
            kind = 'bowl'
            ejctavg = np.zeros([2,2])
            database['EjctExp'] = None; database['EExpStD'] = None
            database['EjctCoef']= None; database['ECoefStD']= None

        if gbsavename:
            newGB.db.append(database)

#-----------------------------------------------------------------------------
                        ###  PLOTS ### DIAGNOSTIC ###

#  1) Cross & Rim on Img    2) Cross & Rim on Dem    3) Cross Profiles
#  4) Average Bowl Linear   5) Average Rim Linear    6) Average Flank Linear
#  7) Avg Bowl Log & Fit    7) Absolute Rim Slope    9) Avg Flank Log & Fit

#=============================================================================

                    ###  PLOTS ### PAPER FIGURES ###

#          1) Average Bowl  Linear     2) Avg Bowl Log & Fit
#          3) Average Rim   Linear     4) Absolute Rim Slope
#          5) Average Flank Linear     6) Avg Flank Log & Fit

#=============================================================================

        if figures:
            print 'making plots'

            pl.figure(2);pl.clf()
            pl.figure(3);pl.clf()
            pl.figure(1);pl.clf()

        if figures == 'Paper':
            subpl = [211, 211, 211, 212, 212, 212]
            fsz = 12

            pl.rc('xtick', labelsize=10)
            pl.rc('ytick', labelsize=10)

        elif figures == 'Diagnostic':
            subpl = [334, 335, 336, 337, 338, 339]
```

```
    fsz = 5

# >> Change tick size
    pl.rc('xtick', labelsize=5)
    pl.rc('ytick', labelsize=5)

    colors = ['r','g','b','k']
    if kind== 'ALL':

    # >> Set up to Plot Cross & Rim on Image
        pl.subplot(331)
        pl.imshow(ejct.maps[1], cmap = pl.cm.gray, extent = limit)
        pl.xlabel('$\ x [m]$', fontsize = 5)
        pl.ylabel('$\ y [m]$', fontsize = 5)

    # >> Set up to Plot Cross & Rim on DEM
        pl.subplot(332)
        pl.imshow(ejct.maps[-1], cmap = pl.cm.jet, extent =limit)
        pl.xlabel('meters',fontsize=5); pl.ylabel('meters',fontsize=5)

# >> If the ejecta is broken, use the bowl DEM for the pics
    elif kind == 'bowl':

    # >> Set up to Plot Cross & Rim on Image
        pl.subplot(331)
        pl.imshow(bowl.maps[1], cmap = pl.cm.gray, extent = limit)
        pl.xlabel('meters',fontsize=5); pl.ylabel('meters',fontsize=5)

    # >> Set up to Plot Cross & Rim on DEM
        pl.subplot(332)
        pl.imshow(bowl.maps[-1], cmap = pl.cm.jet, extent = limit)
        pl.xlabel('meters',fontsize=5); pl.ylabel('meters',fontsize=5)


# >> 1,2,3) Plot Cardinal Profiles, and Locations on DEM and Image
    for sub in [331,332]:
        qq=0

        for profile in cardinal:
            pl.subplot(333); pl.plot(profile[0],profile[1],colors[qq])
            pl.subplot(sub);
            pl.plot(ccoord[qq][0]-demxmin, ccoord[qq][1]-demymin,\
                            colors[qq])
            qq+=1
```

```
                pl.plot(rim[:,0] - demxmin, rim[:,1] - demymin, 'k.')
                pl.axis(limit)

        # >> 4) Plot Bowl Profiles

        if figures:
            if kind== 'bowl' or kind== 'ALL':
                pl.subplot(subpl[0])
                pl.plot(bowlavg[:,0],bowlavg[:,1],'k')
                pl.ylabel('$h/R$',fontsize=fsz)
                pl.xlabel('$r/R$',fontsize=fsz)
                pl.axis((0, 1, 0, max(bowlavg[:,1])*1.1))
                pl.text(0.05,max(bowlavg[:,1])*0.75, ' $\phi =$' \
+str(round(AvgMax,2)) +'$^\circ$' +u"\u00B1" + str(round(stdMax, 2)) + \
'\n$\phi_{\\rm max} =$ ' + str(round(MasMax,2))+'$^\circ$', fontsize = fsz)



            # >> 7) Plot Log Scale
                pl.subplot(subpl[3])
                pl.plot(BLogProAvg[:,0],BLogProAvg[:,1],'k')

            # >> Plot Fit
                bfit = np.array([BLogProAvg[0][0],0]) * BStatAvg[0] +BStatAvg[1]
                pl.plot([BLogProAvg[0][0],0],bfit,':k')

                pl.xlabel('log $r/R$',fontsize=fsz)
                pl.ylabel('log $h/R$',fontsize=fsz)
                txtx = BLogProAvg[0][0]
                txty = BLogProAvg[-1][1]
                dfy = 0.1*abs(txty-BLogProAvg[0][1])
                pl.axis((txtx*1.01, 0, BLogProAvg[0][1]-dfy, txty+dfy))
            # >> u"\u00B1" Makes the plus or minus sign
                pl.text(txtx, txty-2*dfy, ' $\\alpha_{C}=$ '+ \
                            str(round(BStatAvg[0], 2)) + \
                            u"\u00B1" + str(round(sigB[0], 2)) +'\n $B_{C}=$ '+\
                            str(round(BStatAvg[1], 2)) + u"\u00B1" +  \
                            str(round(sigB[1],2)), fontsize = fsz)

            if figures == 'Paper':
                pl.savefig(cname +'_BowlProfiles.png', dpi=500)
                pl.savefig(cname +'_BowlProfiles.svg', dpi=500)
                pl.figure(2);pl.clf
        # >> 5) Plot near RIM profile
            pl.subplot(subpl[1])
```

```
    pl.axis((-.5*rpsz, .5*rpsz, min(rimavg[:,1])-.01, \
                 max(rimavg[:,1])+.01))
    pl.plot(rimavg[:,0],rimavg[:,1],'k')

    pl.ylabel('$h/R$',fontsize=fsz)
    pl.xlabel('$r/R$',fontsize=fsz)

# >> 8) Plot Rim Roundness Metric

    pl.subplot(subpl[4])
    pl.plot(RRMetAv[:,0], RRMetAv[:,1], 'k')
    pl.xlabel('$r/R$', fontsize=fsz)
    pl.ylabel('$|m|$', fontsize=fsz)

# >> Subplot title

    pl.text(0, max(RRMetAv[:,1])*.75, ' $\ell_{C}=$' + \
       str(round(DecayW[0],3)) +u"\u00B1"+ \
       str(round(sdDecay[0],4)) + '\n $\ell _{F}= $' + \
       str(round(DecayW[1],3)) + u"\u00B1" + str(round(sdDecay[1], 4)),\
             fontsize = fsz)

    if figures == 'Paper':
        pl.savefig(cname +'_RimProfiles.png', dpi=500)
        pl.savefig(cname +'_RimProfiles.svg', dpi=500)
        pl.figure(3);pl.clf

# >> 6) Plot Ejecta Profiles

    if kind== 'ejct' or kind== 'ALL':

        pl.subplot(subpl[2])
        pl.plot(ejctavg[:,0],ejctavg[:,1],'k')
        pl.ylabel('$h/R$',fontsize=fsz)
        pl.xlabel('$r/R$',fontsize=fsz)

    # >> 9) Log Scale Plot
        pl.subplot(subpl[5])

        pl.plot(EjLogProAvg[:,0], EjLogProAvg[:,1],'k')

    # >> Plot fit

        efit = EjStatAvg[0] *np.array([0, EjLogProAvg[-1][0]])\
                            +EjStatAvg[1]
```

```
            pl.plot([0,EjLogProAvg[-1][0]], efit, ':k')

            pl.xlabel('log $r/R$',fontsize=fsz)
            pl.ylabel('log $r/R$',fontsize=fsz)

            txty = (EjLogProAvg[0][1])
            txtx = (EjLogProAvg[0][0] + EjLogProAvg[-1][0])/2
            dfy = 0.1*abs(txty-EjLogProAvg[-1][1])
            pl.axis((0,EjLogProAvg[-1][0], EjLogProAvg[-1][1]-dfy,\
                           txty+dfy))
            pl.text(txtx, txty-2*dfy, ' $\\alpha_{F}=$ ' +
                     str(round(EjStatAvg[0], 2)) +\
                     u"\u00B1" + str(round(sigEj[0], 2)) + '\n $B_{F}=$ ' +\
                     str(round(EjStatAvg[1], 2)) + u"\u00B1" +  \
                     str(round(sigEj[1], 2)), fontsize = fsz)

    # >> Save the Figure
    if figures == 'Paper':

        pl.savefig(cname +'_Flank_Profiles.png', dpi=500)
        pl.savefig(cname +'_Flank_Profiles.svg', dpi=500)

    elif figures == 'Diagnostic':

    # >> Title of Figure

        pl.suptitle('Profile Analysis: Crater ' + cname)
        pl.savefig(cname +'_Profiles.png', dpi=250)
        pl.figure(2); pl.clf()
        leg = []

        for ii in random.sample(range(0,len(RRMet)),5):
            pl.plot(RRMet[ii][:,0],RRMet[ii][:,1])
            leg.append([round(RRdk[ii][0],4),round(RRdk[ii][0],4)])
        pl.legend(leg, prop = {'size':7})

        pl.savefig(cname +'_Decay.png',dpi=250)
    i += 1

# >> Save golembase object

if gbsavename:
    print 'Pickling Result'
    newGB.pickle(gbsavename)
```

3.8 Combine Database

```
#=============================================================================
# 20130123-1200-gpudoo - combine_golb.py
# Combine Golembases / Lynn Geiger
# Unify data that had been split by parallelizing
#
# <inputs>
# When calling the method, use 'python combine_golb.py step' where step is
# either plex or prex depending on which part of the code you are on.
# It is assumed that the code is run in the same directory, but this is
# changable. </inputs>
# <products>
# Single Golbase </products>
#=============================================================================
import os, sys
import fnmatch as fm
from omnigenus import unpickle
import golbase as gb
import epdobase as epdo
#-----------------------------------------------------------------------------
# >> Read in the parameter list containing the name of the golbase
parampath = '/home/lynn/Thesis/python/' + sys.argv[1] + '_param.dex'
ep = epdo.cl_epdobase()
ep.load(parampath, format='dex')

# >> If not in current directoy, change this. Maybe input a sys.argv[2]?
basepath = './'

# >> Extract the name of the golbases
PkBase = ep.dcty['value'][2][:-4]

# >> Find all of the golbases to combine
Pkls = sorted(fm.filter(os.listdir(basepath), PkBase + '*.pkl'))

# >> Start a new / fresh golbase
gb = gb.cl_golbase()

# >> For each golb-pickle, open it and append it onto the new one
for pickle in Pkls:
    golb = unpickle(basepath + pickle)
    gb.db += golb.db

# >> Save the master golbase:
gb.pickle(PkBase + '_Comp.pkl')
```

3.9 Selecting Target Properties

```
#=========================================================================
# 20130211- 1800-gpudoo - prep_for_hiview.py
# Prepare for hiview attribute selection / Lynn Geiger
# Finds mfnums of all craters in an obsid, and returns them
#
#=========================================================================

import sys, os
import translate_feature_index as tf
import numpy as np

mfidx = tf.cl_mfidx()

obs2check = sys.argv[1]

inds1 = np.nonzero(np.array(mfidx.obsid1) == obs2check)[0]
inds2 = np.nonzero(np.array(mfidx.obsid2) == obs2check)[0]

index = np.concatenate((inds1,inds2))

mfnums = mfidx.mfnums[index]

for num in mfnums:
    print str(num).zfill(3)

#=============================================================================
# 20130212-0900-gpudoo - select_attributes.bash
# Select Context attributes in HiView / Lynn Geiger
# Shows crater context images then HiView image for attribute selection
#
# [remarks] If all images load, check permissions.
#
#=============================================================================

# >> Loop through HiRISE images (Can replace with list if you only need few)
for j in $(ls /mnt/d4/picm/context/*.JP2)
do

# >> Show image name and open HiView
    echo contex image: $j
    HiView -Image $j &
    finm=$(basename $j)
    finm=$(echo $finm | cut -d "_" -f1,2,3)
```

```
# >> Show Obs ID just to double check
    echo obsid: $finm

# >> Find all mfnum craters with that obsid:
    for i in $(python prep_for_hiview.py $finm)
    do

# >> Show crater number
        echo crater: $i

# >> Open context image for crater
        eog /mnt/d4/picm/context/attributes/context-zoom/mfnum-$i*

    done
done

#================================================================================
# 20130212- 1000-gpudoo - convert_attrib-pngs_epdo.py
# Convert Attrib-pngs to epdo
#================================================================================
import os

a = sorted(os.listdir('.'))
txtf = open('../idx mars fresh attributes.dex','w')
txtf.write('++ mfnum # attr\n** MasterFeatureNum # Attributes')

lastnum=''
attr = ''

for pic in a:
    mfnum = pic[6:9]
    attrls= pic[10:-4]

    if mfnum == lastnum:
        for i in attrls:
            if i not in attr:
                attr += i
    else:
        txtf.write(lastnum + attr)
        attr = ' # ' + attrls
        lastnum= '\n' + mfnum
txtf.write('\n++')
```

3.10 Selecting Target Geology

```
#=============================================================================
#
# 20190218-1200-gpudoo - search_mola.py
# Search MOLA / Lynn Geiger
# Search MOLA for crater location and elevation and geosetting
#
#=============================================================================

import numpy as np
import pylab as pl
import translate_feature_index as tf
from omnigenus import unpickle
import epdobase as eb
import math

mfidx = tf.cl_mfidx()

# >> Circumference of Mars:
Cm = 21344000
A = Cm/360.

# >> Open Pickle (4,16,32 pix/degree)
ares = unpickle('/home/lgeiger/Mars32.pkl')

ebattrib = eb.cl_epdobase()
eblatlon = eb.cl_epdobase()
ebattrib.load('/home/lgeiger/git/picm/binuab/dbase/idx mars fresh craters' + \
                  'attribs.dex',format='dex')

eblatlon.load('/home/lgeiger/git/picm/binuab/dbase/idx images lat lon.dex',\
                  format='dex')

mfnums = ebattrib.dcty['mfnum']

eblatlon.numerize()
inls = []
xmls = np.array([])
ymls = np.array([])
ltls = []

# >> Everything will be in the same order as the attributes epdobase

for i in np.arange(len(mfnums)):
```

```
    obsid = mfidx.obsid1[int(mfnums[i])]
    llind = eblatlon.matching_records({'obsid':obsid})
    inls  = np.concatenate((inls,llind))

    atstr = ebattrib.dcty['attribs'][i]
# >> Remove any zeros, and if the crater is in lowlands, we'll put it back in
    atstr = atstr.replace('0','')
    ebattrib.dcty['attribs'][i] = atstr

    lat = (eblatlon.ndct['maxlat'][llind] + eblatlon.ndct['minlat'][llind]) / 2.
    lon = (eblatlon.ndct['maxlon'][llind] + eblatlon.ndct['minlon'][llind]) / 2.

    ltls = np.concatenate((ltls,lat))

    xmls = np.concatenate((xmls,(A *(lon -180) * math.cos(math.radians(lat)))))
    ymls = np.concatenate((ymls, (A * lat)))

nx, ny, ns, elvs = ares.profile(x = xmls, y = ymls, spacing = None, mapnum =0,\
                                    order=0)
# >> Figure out Where in the world is Carmen san de crater....

lola2 =  np.nonzero((ltls > 20)*(elvs <-3))[0]
lowlands = np.nonzero((ltls > 45)*(elvs <-3))[0]

# >> Plot and save the image
ares.draw(1,0,'rainbow')
pl.plot(xmls,ymls,'.k')
pl.plot(xmls[lola2],ymls[lola2],'*r')
pl.plot(xmls[lowlands],ymls[lowlands],'*w')
pl.axis('image')
pl.legend(['Crater Location','Vastitas Borealis','Surface Ice'],numpoints =1,\
                        prop={'size':8},loc=2)
pl.savefig('crater_mola32_map.svg')

for ind in lola2:
    if ind in lowlands:
        ebattrib.dcty['attribs'][ind] += '0'
    else: ebattrib.dcty['attribs'][ind] += '1'

ebattrib.save('dex','idtest32.dex')
```