NORTHERN ARIZONA UNIVERSITY
College of Engineering, Forestry & Natural Sciences
Department of Computer Science

Technical Feasibility

October 27, 2017

Jet Propulsion Laboratory Image Analysis
Client: Iona Brockie NASA/JPL-Caltech
Team Hindsight
Charles Beck, Alexanderia Nelson, Adam Paquette, Hunter Rainen
Mentor: Austin Sanders
Capstone Director: Dr. Doerry

Table of Contents

1. Introduction

We are Team Hindsight and we are working on ‘Image Analysis of Abraded Rocks to Determine Dust-Free Area’. Our project sponsor is the Jet Propulsion Laboratory (JPL) at the California Institute of Technology with our main point of contact/client Iona Brockie, a Mechatronics Engineer at JPL. One of JPL’s current projects is the Mars 2020 (M2020) rover, an upcoming mission to Mars that will explore various regions of the Martian surface to search for evidence of past life on Mars. The Mars 2020 rover will use a suite of tools including an onboard drill with a set of drill bits to take measurements of the soil/rock and potentially collect samples from the Martian surface. To identify what samples to take, the rover is also equipped with a Planetary Instrument for X-ray Lithochemistry (PIXL) camera. The PIXL camera looks at a particular region and analyzes it for chemical compounds and elemental makeup. Analyzing these two aspects of a rock give some indication of life, or other interesting features of the martian landscape such as potential signs of water. The significance of finding past life on mars would provide more insight to the development of humans and all species on earth. If we are able to find evidence of martian life, it would create many opportunities for space exploration.

Currently, the team at JPL are running a series of tests on the tools they will use on Mars. For some of these tests, the team at JPL mimics the Martian atmosphere in a vacuum chamber. However, this is a time-consuming process because they have to depressurize the chamber to examine the results and then pressurize the chamber again to run more tests. They also have to review and analyze each test manually, which can be subject to both human error and human bias for what looks more “correct”. Ideally, the team at JPL want to analyze their tests while under pressure in the vacuum chamber using only the cameras.

The goal of this document is to present an outline of the challenges we plan to tackle and solve. The first section is the Technological Challenges section, where we will analyze each major technological challenge we expect to encounter and the solutions we plan to implement. In the Technology Analysis section, we will go through each issue previously described in the technology challenges, look at possible alternatives to addressing this issue, and explain why we chose to go with that solution. The following section, Technology Integration, will bring together all the solutions described in the Technology Analysis section into a coherent overall solution. Finally, we will conclude this document to reiterate why we chose to go with our proposed solution.

2. Technological Challenges

After planning our solution for the image analysis, there are a few challenges we expect to encounter. The most problematic issues we envision running into are the following:

1. Classify dust and dust-free areas of the image
   The main challenge for solving the problem is classifying parts of the abrasion image that have dust or no dust visible in the image.

2. Get measurements of surface area cleared in abrasion using the Gas Dust Removal Tool.
   The second issue is getting the measurements of the surface area that has been cleared of dust in the abrasion using the Gas Dust Removal Tool. The results would be used to help analyze how effective the tool was at removing the dust from the abrasion made in the rock.

3. Our client prefers a platform friendly to non-programmers that the team at JPL will be able to understand and tweak.
   We need to decide on what language to use for our software that will be easy to learn for the JPL team. MATLAB is the preferred language because a majority of the people working at JPL are engineers, however, we need to decide if there is a better language that can solve the other challenges more easily.

4. Create an interface for our client to tweak the image analysis algorithm(s)
   We need to build an interface that will allow the end users in our client's business to modify the algorithms used for our image analysis. The interface needs to be straightforward enough for the users to understand how to interact with the interface.

3. Technology Analysis

This section will examine each problem we plan on solving with this project. For each problem, there will be an introduction to the issue, followed by the possible approaches to the problem, continued with the chosen approach for the problem, and ending with proving feasibility. Each approach will have pros and cons for why it will/won't work for solving the problem.

3.1 Problem 1: Classify Dust Free Areas

The main problem our team needs to address is identifying the total surface area (of the bottom of the abrasion) covered in dust. There are a multitude of approaches to this, most aimed at distinguishing between the texture of dust and non-dust surfaces.

3.1.1 Possible approaches

- Image Subtraction

This method utilizes the before and after images of drilling and dust removal. Using these two images (before/after) we will create a mask of the pixels that changed (eg. dust being blown away).

The decision for using image subtraction will also be determined by the process for gathering statistics on the effectiveness of the dust removal tool. If we need the before and after pictures for that analysis, we should also be willing to use the mask that is created from the subtraction. However, if we do not need both images and the data provided by the subtraction is not significant, then it won't be necessary.

Approach	Pros	Cons
Image Subtraction	- Provides a way to find differences between images. - Supported by most computer vision libraries. - Use a simple mask = img1 - img2 subtracting the arrays resulting in a mask. - Subtracting the two images is fast.	- Results can be inconsistent due to shadows and lighting. - Requires both the before and after image to create mask. This results in an input size of 2n for any set of images. - Having to read in two images at a time is costly.

Table 1. Pros and Cons of Image Subtraction

- Fractal and Stochastic models

These two focus on modeling two different properties of images. Fractals focus on properties of shapes like fractals or other common patterns (geometric) found in nature. Stochastic focuses on modeling regions of textures (with elements of randomness), such as probabilities for how gray level intensities are distributed in regions of an image. Stochastic models are particularly relevant to our project as dust is a stochastic texture with not many clear geometric shapes.

Approach	Pros	Cons
Fractal Model	- Geometric shapes with clear contrast - Textures with hard edges	- Random textures with poor contrast - Unclear edges (gradients) - No clear pattern
Stochastic Model	- Doesn't rely on a clear repeating pattern - Can work within a range of values (eg intensity range) - Textures with clear distinction between intensity histograms	- Geometric shapes with clear contrast - Textures with similar intensities/ histograms

Table 2. Pros and Cons of Fractal and Stochastic Models

- Wavelet analysis and Segmentation

These two types of analysis focus on texture properties of images. Wavelet analysis like Gabor and Wavelet are both used for texture detection. Gabor typically is used for feature extraction and Wavelets are generally used for analyzing

Approach	Pros	Cons
Gabor Transform	- Helps figure out if direction is a major component in the texture (eg wood grain)	- Textures where direction is random and inconsistent (eg dust)
Wavelet	- Frequency and location (intensity values) are major component of texture	- If location and frequency are not a major component of the texture (eg pattern at random locations and frequencies)

Wavelet cont.	- Sudden Transitions (eg Haar) - Complexity is O(n)	- Only theoretical use for identifying dust (this method is typically used for removing noise in an image)
Segmentation	- Learn what features make up an image (eg. AR Model) - If color is major component of picture	- Time consuming to train (eg neural network) - Trial and error for some algorithms to get them to work (this can be solved with basic machine learning however)

Table 3. Pros and Cons of Wavelet Analysis and Segmentation

How well each approach provides a given capability: 1 - 5 (5 = best)

Capabilities	Subtraction	Fractal Modeling	Stochastic Modeling	Wavelets	Segmentation
Fast	5	2	3	3.5	3.5
Find shapes	3	5	1	2	2
Find textures	1	1	5	4	3.5
Divide image into regions	2	1	4	3.5	5

Table 4. Comparison of Identifying Dust Free Area Approaches

3.1.2 Chosen Approach

To find and classify dust we plan to use **Image Subtraction** and **Stochastic Modeling** of dust (Histogram analysis), in conjunction with **Color/Texture Segmentation** on the abrasion images both before and after the blast of air. Through experimentation, and research we have found these methods provide a good base to start with. While further research and experimentation is needed for the processes, what we have found so far is promising.

3.1.3 Proving Feasibility

1. Create demo

   Develop working demo that uses image subtraction and stochastic modeling to find dust in before and after images.
2. Test images

   Input pictures to program in either single images or a batch of images.

3. Compare results
   View images output by the program and compare them to original versions to pinpoint correctness or what the program is identifying as dust.
4. Refactor demo
   Make changes to the sensitivity of the program and the threshold for pixels it is attempting to define as dust.
5. Repeat
   Re-do the steps of development to get a refined version of program.

These three specific methods will each assist in the process for finding dust free areas, each providing some piece of data that will contribute to a larger whole. Image subtraction should allow us to reduce our search space, and provide us a clear indication of exact areas where our other algorithms should look for dust. Stochastic Modeling and color/texture segmentation should then define specific areas of dust and areas where dust is not present. Providing a reasonably accurate map of dust/dust-free regions will give our team a solid foundation to build other data analysis tools, and additional tools to improve dust detection.

3.2 Problem 2: Gathering Measurements of Surface Area Cleared

Generating accurate measurements for the effectiveness of the dust removal tool is a separate challenge that requires accurate dust detection as a prerequisite. We need to analyze how well the dust removal system removes dust from the abrasion. To get an accurate estimate of dust removed, our team needs to find how much of the area (bottom of drilled hole) is dust free and allow the user to adjust the algorithm to get a more accurate analysis if necessary.

3.2.1 Possible approaches

- Regional Segmentation

This approach focuses on the differences in dust pixel counts when comparing before and after images of an abrasion. This involves developing an initial idea of what dust looks like, establishing some threshold for said dust, then comparing the before after images histograms to determine dust free percentage.

- Edge detection

This approach relies on our algorithm's ability to determine the dust free area of an image, and find the abrasion within the image. We will use edge detection in combination with mask from the image subtraction. Edge detection uses a light filter on the image to produce thin lines leaving out pixels that are not considered a part of the edge. We can then use the edges to mark the abrasion on the picture.

- Blob Detection

This approach identifies groups of connected pixels in the image that share some common property. Blob detection takes a threshold of a binarized image and groups connected pixels that meet that threshold. These initial blobs are connected by any pixels within proximity that share the same trait (similar threshold). Using the coordinates of the blobs (making them as we group pixels) and can find the center and radius of the blobs. We then use this information to mark areas that are dust free and apply the marked spots as a mask over the image.

Approach	**Pros**	**Cons**
Regional Segmentation	- More direct approach (If we already can identify dust) we just count how many dust pixels are no longer in picture - Distinguishes dust free areas of the whole image as either dust or dust free	- Does not provide an exact measurement for the percentage of surface area of abrasion that is covered by dust - Less efficient as the image sample sizes for histogram comparisons become smaller (10 x 10 subsets vs 25 x 25 subsets for the whole image)
Edge Detection	- Uses non-maximum suppression to eliminate pixels not considered an edge. - Defines edge pixels using hysteresis(edge pixels > upper bound)	- Applies a gaussian filter, blurring image.
Blob Detection	- Uses grouping to gather pixels that are in a region. - Returns center and radius of blobs.	- Binarizes image, hard to see if dust is distinct or not. - Gets entire picture not just what is in abrasion

Table 5. Pros and Cons of Regional Segmentation and Abrasion Identification

How well each approach provides a given capability: 1 - 5 (5 = best)

Capabilities	Regional Segmentation	Edge Detection	Blob Detection
Find Abrasion	1	3	1
Define Dust Free Regions (whole image)	4	1	3
Define Dust Free Regions (abrasion)	3	2	2
Efficiency	2	2	2

Table 6. Comparison of Identifying Abrasion Approaches

3.2.2 Chosen Approach

Initially, we will use the **histograms** and number of pixels in the dust regions to get a rough estimate of the total dust cleared. If there is enough time, we plan to improve the accuracy by either an algorithm or letting the user manually define the edge/area of the abrasion. We will also be using **Blob Detection** to identify and mark the dust free region in the image.

3.2.3 Proving Feasibility

1. Combine results from previous approaches
   Once the tests for the image subtraction and dust detection are successful, we will combine them to produce a clear estimate of the abrasion and its dimensions
2. Test combination against expected answer
   Use JPL's expected result to test the algorithm to see how it is performing
3. Refractor algorithm
   If algorithm is underperforming or could use improvements, the algorithm will be refactored and we will go back to Step 2 to perform those tests again

3.3 Problem 3: Choosing Platforms and Framework

This issue pertains to what programming languages and frameworks we will be using for this project. We need to take into consideration what the client's business is familiar with/has access to and what the capabilities each has pertaining to the problem we are trying to solve.

3.3.1 Possible approaches

The possible approaches we have considered are MATLAB and Python as our programming languages and OpenCV for third-party computer vision library.

- MATLAB

MATLAB is one possible language because our client is most familiar with this language. It has toolboxes that can be used for computer vision, image processing, and machine learning.

Approach	Pros	Cons
MATLAB	- Everything is there. It contains most of the functional libraries within files. There's no need to load when you want to generate plots or do some specialized processing, even though some imports can be used. - MATLAB allows you to test algorithms immediately without recompilation. Users can type something at the command line or execute a section in the editor and immediately see the results. - The Mathworks (the company that sells the MATLAB software) website has documentation and examples for each function in the toolboxes.	- The algorithms structure are proprietary, which means you cannot see the code of most of the algorithms you are using and have to trust that they were implemented correctly. - Mathworks puts restrictions on code portability. You can run your "compiled" application using the MATLAB Component Runtime (MCR), but your portable app must exactly match the version of the installed MCR, which can be a nuisance considering that Matlab releases a new version every 6 months.

Table 7. Pros and Cons of MATLAB

- Python

Python is another possible language to use for our solution because there are a wide variety of third-party extensions as well as the standard library and it can be applied to many different classes of problems due to being a general-purpose programming language.

Approach	Pros	Cons
Python	- It's free and open-source. - Everything in Python is an object, so each object has a namespace itself. - It was created to be a generic language that is easy to read. - Code can be run everywhere since Python is free and works on Windows, Linux, and OS X.	- It's not as neatly packaged as MATLAB. The language installs fine on all operating systems, but you have to make sure you have all the packages you need installed. - The computers for our client's business may not have Python installed and/or the libraries we use installed.

Table 8. Pros and Cons of Python

- OpenCV

OpenCV is an open source computer vision library that has support for Linux, Windows, and MacOS as well as Python and MATLAB interfaces, both languages we are considering to use.

Approach	Pros	Cons
OpenCV	- Is a well constructed, efficient, and large library for computer vision algorithms - Works in both Python and MATLAB	- Poor documentation in languages it supports. - For MATLAB, it requires the correct compiler for mex functions and requires the OpenCV package to be up to date with the newest version of MATLAB.

Table 9. Pros and Cons of OpenCV

How well each approach provides a given capability: 1 - 5 (5 = best)

Capabilities	Python	MATLAB
Documentation	3	4
Testing Code	4	4
Portability	3	3
Performance (speed)	3	2

Table 10. Comparison of Language Approaches

3.3.2 Chosen Approach

Based on the research and small testing of some of the capabilities of each language's image analysis and computer vision libraries that we ran on the images we received, we have decided to do our prototype in **OpenCV** and incorporate **MATLAB** in our final product where the user can tweak parameters and the **MATLAB** code will call the **Python** code.

3.3.3 Proving Feasibility

1. Prototype in Python
   Create a working prototype written in Python using the Rock Type E, which is classified as having the dust color being a different color than the rock
2. Incorporate MATLAB into Python code
   Add MATLAB code that will call the Python code made in the prototype
3. Refactor
   Make any adjustments based on feedback from client for the Rock Type E
4. Write new code in Python
   Write more code in Python to have it run on the next rock type and repeat the previous 3 steps

3.4 Problem 4: Interface for User Interaction

One issue when dealing with complex algorithms is seeing how exactly some modifications to an algorithm's parameters will change its outcome. As we are developing a backend of computer vision algorithms some way for users to interact with the codebase would be ideal. This interaction will allow users to access the code base and actually use some of the algorithms we will be implementing. Ideally, whatever we choose to do for our user interaction it will be easy to understand, easy to use, and allow our users the ability to effectively use the code base.

3.4.1 Possible approaches

The approaches we have considered for this problem are developing a GUI, using simple sliders/parameter modifications, and allowing users (specifically our client) access to the code base for changing things as they see fit.

- Create a GUI that interfaces with the Image analysis

Creating a GUI is one possible option as it would allow users to easily access the backend exactly how our team would want them to. This would allow us to greatly control the user experience, and give them easy to access tools.

- Simple parameter sliders

Simple tools for tweaking or modifying algorithms is another option as it still provides our client with some form of interactivity.

- Let clients tweak the code manually

Finally, we could give our users access to the code to modify as they see fit. This would allow them complete freedom to change the code base and change algorithms to their liking.

Approach	Pros	Cons
Create a GUI that Interfaces with the Image Analysis	- MATLAB, Python, and other third party libraries (eg. PyQt) make creating a GUI possible for just about any language we do the image analysis in - Great for tweaking or modifying algorithm	- Time consuming, takes away time from programming the image analysis - Limits the user to only the tools we provide.
Simple Parameter Sliders	- Allows minimal tweaking that non-programming users can understand easily	- Not a very robust GUI - Not as much control as directly editing the source code
Let Clients Tweak the Code Manually	- Allows programming team to focus on a more robust image analysis tool	- Non-programmers within JPL may have difficulty understanding code - Potential problems of erroneous modifications.

Table 11. Pros and Cons of User Interaction Approaches

How well each approach provides a given capability: 1 - 5 (5 = best)

Capabilities	GUI	Parameter Sliders	Tweak Code Manually
Interactive	5	3	1
Ease of Using	4	4	1
Customizable	2	3	4

Table 12. Comparisons of User Interaction Approaches

3.4.2 Chosen Approach

Our clients will have access to the code to tweak however they want, but we will give them basic functionality in the form of **sliders** (or something similar) for parameters in our algorithms. We will continue to improve this interface giving users more control over the analysis (so the users have less code to manually tweak) with any left over time. The ultimate goal is to minimize the time it takes for them to process images and by creating an interface this will make the process go smoother for the user.

3.4.3 Proving Feasibility

In future client meetings, we plan to discuss and receive feedback on the GUI prototype from the users of our product. The initial prototypes will be redesigned based on what the users tell us so we can make the interactive part of the software useful for them to make any changes to the algorithm.

1. Prototype
   Develop a working model that can be used by client or users
2. Response
   Get feedback on prototype from other users or client
3. Refactor
   Refactor software to fit user feedback to improve the usability of the program.
4. Improved Prototype
   Develop another prototype and begin proving the concept from the prototype stage above

4. Technology Integration

The challenges with this particular project are finding the right technology to use.The solutions we have proposed in this document will need to come together in small parts in order to create a system for our client to use for analyzing their images.

To start, our application will take in either one set of before/after images of dust being blown out of an abrasion, or a batch of before/ after images.

Next, our application will create a simple mask of pixels that changed from the before image to the after image(s) using image subtraction. In addition to image subtraction, we will also run another set of algorithms to detect regions of the image that have dust textures. These two together will create regions that we are fairly certain have dust (or lack of).

After getting the regions where dust has been removed, we will find the area of the abrasion in the picture (using blob detection and histogram analysis).Combining these will give us the percentage of surface area (of the abrasion) covered in dust.

The client will also want to be able to tweak the algorithms so they can adjust the measurements our applications produces. We will create graphical sliders for users to slide and change the parameters of the algorithms described in the chosen approach sections of each challenge. If the users want to change the algorithms themselves, they will likely have to modify the source code.

Once we have everything up and running, we can take our tool functionality further by optimizing our processing time which may involve parallel processing (running multiple algorithms at the same time) and hardware like graphics cards. However, our primary concern is getting an algorithm which can successfully and accurately find the dust/ dust free surfaces.

5. Conclusion

In conclusion, the Mars 2020 rover creates abrasions covered in dust. The rover will then release a blast of air into these abrasions to remove the dust, then take a picture of the abrasion. The general problem that our application will solve is:

1. Calculate how how much of the surface area of an abrasion is covered in dust
2. Create an application to do the above analysis that is easy for the engineers at JPL to use

Earlier, in section 3, we outlined the technological challenges our team envisions having to face in the process of solving the two major problems just mentioned. Here are our solutions and how confident we are that they will fit JPL's needs.

1 - 3 (1 = needs testing, 2 = reasonable approach, 3 = confident)

Tech Challenge	Proposed Solution	Confidence Level
Dust detection	**Texture detection algorithms**	**2.5**
Area of abrasion cleared	**Texture detection algorithms**	**2**
Choose platform	**Matlab + Python**	**3**
Easy to use Interface	**Sliders for parameters to analyze**	**3**

Table 13. Summary of Confidence for Proposed Solutions for Technology Challenges

5.1 In Summary

Our application will allow the mechanical and electrical engineers working on M2020 to use computer vision analysis on the abrasions done in a vacuum chamber testbed. We are confident that we will produce a useful tool for the team at JPL to help analyze dust cleared from abrasions. We have seen some promising results from our research and limited testing. This will increase the amount of testing JPL can do and save the M2020 team many hours.

We still have some open questions we will be actively working on. How will we optimize? Will we implement a neural network for the more challenging rock types? Once we get working code, optimization will become trivial as the languages we are using can be ported to faster languages, and we can also expect to utilize parallel processing where applicable (eg running multiple image filters at same time).

5.2 Future Work

Two possible routes we may take, depending on time, are parallel processing and machine learning. Parallel processing (running multiple algorithms at the same time) will significantly reduce the processing time of the images. If we decide to implement machine learning, we can talk to experts about implementing search techniques (eg random walk) to help automate the analysis. There are also previous examples of people using machine learning for images analysis (typically feature extraction) such as neural networks (ANN) for face detection. We have a large enough dataset that we could train a ANN with, making that avenue a more realistic approach.

Our team is confident that these challenges are all achievable and expect to save the team at JPL many hours of testing. We believe that this tool will be able to relay to JPL if their air blaster is working correctly and efficiently.