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

Requirements Specification

Version 2

December 8, 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

Accepted as baseline requirements for the project:

For the client:_____ Date:_____
For the team:_____ Date:_____

Table of Contents

Section 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.

Section 2. Problem Statement

JPL's latest Mars rover will have a suite of tools for discovering a variety of scientific evidence of past life. One tool that directly affects Team Hindsight's project is a drill. The rover will drill holes into rocks on Mars so that other instruments like PIXL and SHERLOC (Scanning Habitable Environments with Raman and Luminescence of Organics and Chemicals) can further analyze with finer detail the composition of the Martian surface.

However, the process of drilling into the rocks creates dust in and around the hole. This dust obscures the mineral and chemical makeup of the rocks which the scientific instruments on board the rover are trying to analyze. JPL's current solution is to blow a puff of air into the hole (via compressed gas) to blow out any dust. Our client needs to know how effective this gas dust removal tool is at getting the dust out of any particular hole the rover might drill. Currently, JPL's process for determining the effectiveness of this gas Dust Removal Tool (gDRT) is slow, manual, and prone to human error.

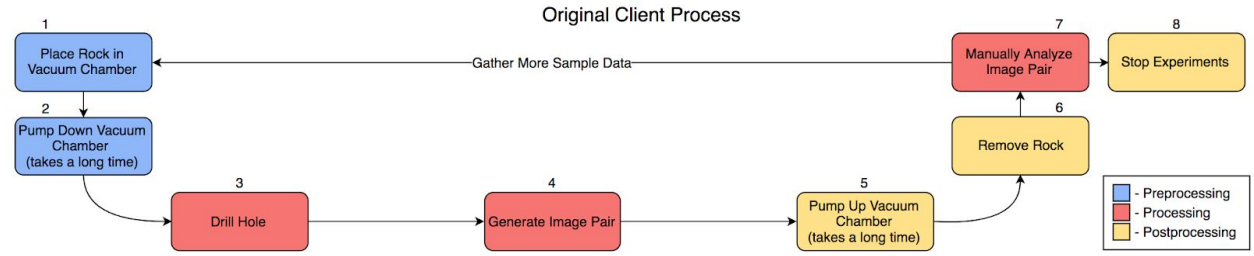

Figure 1. Original Client Process

Specifically, it's slow to re-pressurize the chamber for every test as well as manually analyze every abrasion. Also, moving the test rock (with drilled hole) out of the pressurized chamber runs the risk of disturbing the dust and invalidating the entire test. Furthermore, humans are subjective when defining how much of a given area is covered in dust and may get different answers from other analysis. Figure 1 illustrates the current process that JPL is using to test their gDRT.

Section 3. Solution Vision

With our understanding of the problem, we have determined the best solution to be an image processing pipeline. This pipeline would allow JPL to automatically analyze before and after image pairs (generated by JPLs own testing) using computer vision algorithms to determine the dust coverage of the abrasion area.

Figure 2. Example of Image Analysis Concept

Section 3.1 Specific Features:

- Ability to process multiple pairs of images at once.

JPL normally generates about 10 sets of data per week, meaning that every week they generate a total of 20 images (10 pairs of before and after images) as a minimum. However, some of these sets can include a second after image, as well as a high resolution after image. If this is done for all 10 sets it would generate a total of 40 images. Thus, our solution will be able to process at least 10 sets of data points at a time where we could expect a total of 40 comparisons between before and after images.

- Automatic detection of dust covered areas in an image.

Using various computer vision algorithms, our solution will be able to automatically detect dust covered areas within an image set. Two example methods would be image subtraction, and stochastic modeling. Image subtraction creates a difference between two image (in our client's case, a before and after image), and returns a result image that can be used to find the location of the abrasion. Stochastic modeling uses image data in the form of a histogram to find patterns or specific types of spots in an image and, in our client's case, dust. Both of these methods will be applied to a pair of before and after images as algorithms, where the result will be used to determine dust coverage of the abrasion.

This automation does not drastically change JPL's experiment workflow, but it will allow JPL to focus more of their time on generating data. Currently JPL is spending more time on running the process to gather test data than they are analyzing test data. Thus, by automating JPL's analysis of the image sets, our solution will be able to save them a significant amount of time when gathering test data, allowing more time for analyzing data.

- Application of user defined and programmer defined algorithms.

The team at JPL will need the basic algorithms to analyze the abrasion region and give them an accurate calculation of the dust coverage in the abrasion. Our solution will provide JPL with these algorithms and give JPL the flexibility to apply any algorithm they want on the data. This would allow JPL to run other types of images through the pipeline and give our solution potential to be used on other projects JPL may be working on.

Section 3.2 Process Improvements:

JPL's current process is depicted in figure 3. The current process starts at step 1, and ends at step 8, potentially looping back to step 1 from step 7. The current process is currently slower than our client, JPL, would like it to be. This is due to preprocessing and postprocessing elements being present in the main test gathering loop. This includes steps 1, 2, 5, and 6 all being part of the main test gathering loop, when they should be done before or after the test gathering. The two biggest time sinks for this loop are steps 2, and 6. Both steps require JPL to either pump a vacuum down to the martian atmosphere or up to earth atmosphere, and doing this for every test is costly.

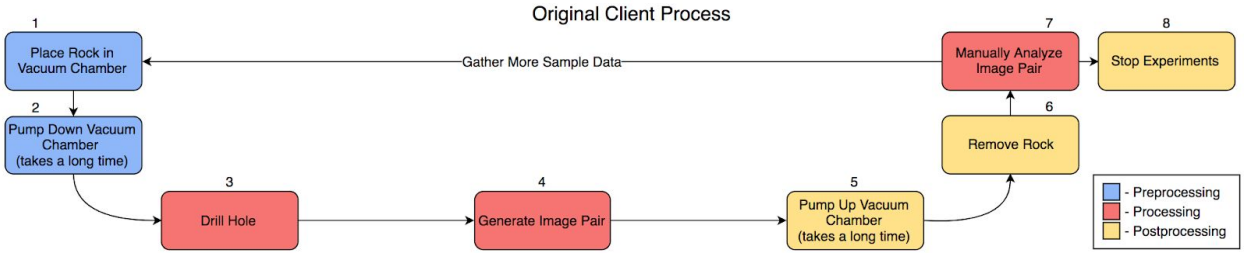

Figure 3. Example of JPL's current process for gather test data

Our tool will give JPL a new process, as depicted in figure 4, that will segment all steps into their appropriate sections. This means that all preprocessing, processing, and postprocessing elements will take place either before, during, or after the process respectively. This gives JPL a streamlined and much faster process than before.

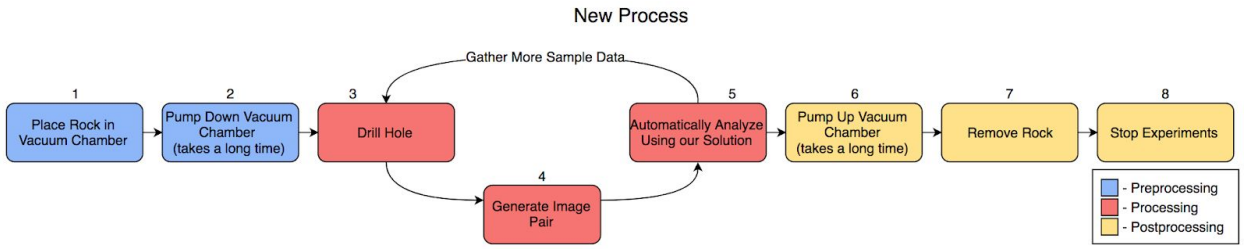

Figure 4. Example of JPL's current process for gather test data

Section 4. Project Requirements

In short, our client, JPL, needs a way to automate analyzing the effectiveness of their dust removal tool on the Mars 2020 rover. The rover will drill a hole (creating dust), take some pictures of the hole with dust in it, use the rover's onboard dust removal tool, and take more pictures of the effects of the dust removal tool on the hole.

Section 4.1 Functional Requirements

1. Take in batches of images of drill/ dust removal testing
    - 1.1. Take in images and analyze naming convention (file name)
        - 1.1.1. The filename for an image pair (or before/ set of multiple after images) should indicate which test-run an image came from
        - 1.1.2. Our application should be able to determine whether an image is before using the gDRT or after using the gDRT (via the filename)
    - 1.2. Take in a batch of single pair of before/ after images (or before/ set of multiple after images)
    - 1.3. Take in a batch of multiple pairs of before/ after images (or before/ set of multiple after images)

2. Detect dust free regions in images
    - 2.1. A GUI will let users select which rock type images belong to (eg. rock type E).
        - 2.1.1. This will help our application use appropriate algorithms
    - 2.2. Given images of the drilled hole, detect areas in images containing dust
        - 2.2.1. Give coordinates of regions covered in any amount of dust
    - 2.3. Given image of drilled hole, identify areas in image that don't contain dust
        - 2.3.1. Give coordinates of regions of rock that are dust free or partially dust free
            - 2.3.1.1. These regions should be inside the abrasion as that is what we are analyzing (regions may be partially outside of the abrasion)
    - 2.4. Regions are any group of pixels in the image that are relevant to what we're analyzing
        - 2.4.1. If we're analyzing the region of pixels that is inside an abrasion, there may be sub-regions which may constitute different groups of pixels inside that abrasion region
            - 2.4.1.1. All the subregions (inside the abrasion) should add up close to the total number of pixels inside the abrasion, or we should be able to

determine what percentage of the abrasion is covered by a certain region (eg. how much of the abrasion is covered in 50% dust covered regions)

2.4.2. Regions may be any shape and may be different sizes

2.5. Determining how much of a region is cleared of dust. For example, given a square region in an image (10x10 pixels), if half of those pixels (50 pixels) are marked as dust, we would mark that region on the image as yellow (see F3)

2.5.1. Completely cleared is less than 30% of a given region covered in dust

2.5.2. Somewhat cleared is between 30% and 70% of a given region covered in dust

2.5.3. Completely covered is more than 70% of a given region covered in dust

2.5.4. These percentages may change slightly in time, but the described percentages will suffice

2.5.4.1. If these percentages change, we will show JPL how to change the percentages within our application (Our team will make the decision to allow users to make changes through either the GUI or in source code).

3. Visually display results of application to users

3.1. Given coordinates of dust free regions in an image, mark the image with a relevant color for users to see.

3.1.1. Colors may be subject to change depending on how visible they are on a given rock type. These colors should be easy to see when placed on top of dust or rock.

3.1.1.1. For example: This means that if the rock/ dust type are bright colored, the program should mark the regions with darker colors. (eg. not yellow)

3.1.2. Given coordinates of completely dust free regions (ie less than 30% of a given region is covered in dust) mark the region with relevant color (eg green)

3.1.3. Given coordinates of dust free regions that are not entirely clear (ie somewhere between 30% and 70% covered in dust) mark with relevant color (eg. yellow).

3.1.4. Given coordinates of regions (of rock) that are completely covered in dust (ie somewhere between 70% and 100% covered in dust), mark image with relevant color (eg. red).

4. Allow users to tweak output of application
    4.1. If users can clearly see incorrect calculations from application or have reason to suspect that it will be inaccurate, they should be able to tweak the input parameters of image analysis algorithms
        4.1.1. Users should be able to see options for different settings (for each algorithm being used) that they are able to change
        4.1.2. Users should be able to go into the code and manually change settings if they think it is necessary
        4.1.3. Details for how we teak algorithms will depend on the specific algorithm

To help describe and rationalize the functional requirements of our application, we have provided below use cases to express these requirements from the user perspective.

Use Case: Submit a batch of images
Actor: Operator
Description: The operator submits images to be analyzed by the system.
Preconditions: The operator has a pair/pairs of images saved.
Post-conditions: The pair/pairs of images are accepted to be analyzed.
Main Flow:
1. The operator selects the pair/pairs of images to be analyzed.
2. The operator submits the images to be analyzed.
3. The system accepts the images.

Alternative Flows:
3. The naming convention of the image(s) is not correct.
1. The system issues a warning.
2. The operator renames the image(s) and the use case is restarted.

Use Case: Detect dust free regions in image
Actor: System
Description: The system determines which regions in the image are dust free, partially dust free, or completely covered in dust.
Preconditions: The system has images submitted to be analyzed.
Post-conditions: The system knows what areas have dust, some dust, or no dust.
Main Flow:
1. The system determines what areas contain dust, some dust, or no dust.
    1a. Identifying the dust areas.
        i. The system analyzes areas where the rock is not visible.
        ii. The system retrieves the coordinates of the regions.
        iii. The system saves the coordinates.

1b. Identifying the dust free areas.
   i. The system analyzes areas where the rock is completely visible.
   ii. The system retrieves the coordinates of the regions.
   iii. The system saves the coordinates.
1c. Identifying the partially dust free areas.
   i. The system analyzes areas that contain some dust and rock is visible.
   ii. The system retrieves the coordinates of the regions.
   iii. The system saves the coordinates.

Use Case: Display results to user
Actor: System
Description: The system displays the results to the operator.
Preconditions: The system has analyzed the images it was given.
Post-conditions: The system displays what regions are dust free, partially dust free, or completely covered in dust.
Main Flow:
1. The system retrieves the coordinates for each type of region.
2. The system checks which type of region it will be marking in the image.
3. The system applies the correct color for the coordinates marked.
4. The system repeats steps 1 to 3 for each region.
5. The system displays the marked image.

Use Case: Change parameters of image analysis algorithm
Actor: Operator
Description: The operator changes an algorithm to obtain a better analysis.
Preconditions: The operator has already run an analysis on images through the application.
Post-conditions: The system displays the new results to the operator based on the new parameters.
Main Flow:
1. The operator selects to change an algorithm.
2. The operator changes a parameter for the selected algorithm.
3. The operator submits the new change.
4. The system accepts the new change.
4. The system re-analyzes the images using the new parameter.
5. The system displays the new results.

Alternative Flows:
3. The new parameter throws an error.
1. The system issues a new notification and the use case is restarted.

Section 4.2 Performance Requirements

Performance requirements:

1. The time our program should take to analyze before/ after images.
    - 1.1. For a single pair of before and after images,
        - 1.1.1. Ideally our application will take under one minute to analyze a single pair of images.
        - 1.1.2. It is still acceptable to analyze a single pair of images in under five minutes.
    - 1.2. For a batch with multiple pairs of before and after images
        - 1.2.1. Ideally, our application will take under thirty minutes to analyze every pair of images.
        - 1.2.2. It is still acceptable to analyze the batch of images in under two hours as this is the benchmark in which how long it takes JPL to analyze the images.

2. How accurate should the application be
    - 2.1. How accurate should dust detection be
        - 2.1.1. Within 10% of what JPL would calculate by hand as dust-covered/free
            - 2.1.1.1. Examples of calculated coverage for a pair of before/ after images for rock type E have been given by the client, and this example will be used as a benchmark for our application for rock type E
            - 2.1.1.2. Our client will provide at least one example of hand calculated coverage for all five rock types by mid to late February 2017
                - 2.1.1.2.1. These examples won't change and will be used as metrics for our program
        - 2.1.2. Identify the edge of the abrasion
            - 2.1.2.1. True edge of the abrasion may be covered in dust (actual edge). With the examples from JPL and our own samples we will identify what the true edge is of the abrasion or as close to it as possible.
            - 2.1.2.2. The edge that is visible after dust has been removed (pseudo edge). This is what our program is viewing as the abrasion edge and returning to the user. Needs to be compatible to what JPL is defining as the edge of the abrasion.
    - 2.2. How many rock types should application handle
        - 2.2.1. Ideally all of the rock types given to us from JPL
        - 2.2.2. Acceptable to handle one type of rock type from the set of five types we are given

3. How large a batch of images our application will expect to analyze
    3.1. For a single before/ after pair
        3.1.1. There should only be one before image for each image set.
        3.1.2. The after images in the set can either be a single after image or multiple after images depending on how many blasts of air were shot out..
    3.2. For a batch of multiple before/ after images
        3.2.1. User expects up to 5 pairs of images and a total of 10 to 12 images in the set.
        3.2.2. Our team will test with 10 pairs of images to ensure scalability.

4. How long should users take to learn the system
    4.1. Understanding the graphical user interface should take less than eight hours
    4.2. Understanding the computer vision algorithms may take more time
        4.2.1. Our team will provide a wiki or document explaining the basics of each algorithm our tool uses and how each parameter will change the output
        4.2.2. The wiki will also explain how to add more functionality to our application
            4.2.2.1. Eg. what adding a new algorithm would entail (what input our current algorithms expect, and how to apply that to a new algorithm)

Section 4.3 Environmental Requirements

Environmental Requirements

1. Platform specifications
    1.1. An interface written in MATLAB
        1.1.1. Specifically, users should be able to run our application from MATLAB
    1.2. An interface with Python
        1.2.1. Source code for first prototypes will be in Python
        1.2.2. Application should be able to be run from within Python

2. Sensor Data
    2.1. Data (eg. pictures) will be in the form of either JPG or PNG or raw files
        2.1.1. Application must be able to handle JPG or PNG
            2.1.1.1. Ideally images will be in PNG format or a format that does not compress information

Section 5. Potential Risks

The purpose of this section is to identify and communicate the potential risks that our team has faced and may face in the further development of our software. We will address our risks through testing and working around these risks to our best ability.

Section 5.1 Inaccurate dust detection

Marking covered areas incorrectly: We believe that it is important to convey that we may not get the perfect identification of dust particles in all of our image sets. For some of the rock types we are working with, the color of the dust is a similar to the color of the surface of dust free regions of rock. Determining the difference between these two materials may prove difficult in testing and we may not get all the rock types analyzed correctly. We are hoping to get the best representation of dust free and dust covered areas to our client and are working on one rock type at a time.

Mitigation: In order to develop a more correct program we will be comparing our results of dust identification from our application to 50x50 pixel samples from rock type images that we collected and marked as “dust”, “no-dust”, or unknown. WIth the sample information, JPL’s hand drawn metrics and what our program returns we can visually compare them to verify the correctness of our implementation. Through testing of our software and refinement we will generate more accurate algorithms and mappings. Furthermore, we will have access to professionally developed computer vision code within Matlab and the OpenCV library. These libraries contain fully implemented algorithms that are standard within the computer vision industry such as image subtraction and color segmentation. We have also asked JPL to store their images in PNG format as it will allow us to view more detailed images and get more accurate results showing dust covered or free regions.

Likelihood: Possible, Moderate

Relevance: This risk is very relevant as it pertains to the exact problem we are trying to solve and that is identify these dust free or covered areas. This allows JPL to be able to analyze and calibrate their tools more efficiently.

Section 5.2 Slow runtimes

Large Images: The images taken by JPL are stored as JPGs and this file type losses some data for the pixels of the images, especially when trying to zoom in and find defined pixels and

make comparisons. These images also may be large when they are using their DSLR camera as it takes massive detailed pictures. It could potentially take longer to analyze the image if it is larger and in more detail.

Mitigation: To avoid this risk we have planned to use image segmentation on the larger images to analyze each area separately (potentially in parallel). Furthermore, we can port our program to a faster/ compiled language (eg C). This will reduce the time it takes to analyze each individual image.

Multiple images in one set: Another possible risk we may face is encountering large sets of images. For example, image_01before.png, image_01after1.png, image_01after2.png, image_01after3.png. This could cause the overall runtime of the program to increase as we will be analyzing more images per pair (one before image, paired with a set of many after images).

Mitigation: We are not concerned about this risk because the test images should take not longer than one minute per-image. We also would like to introduce parallelism to the software so that the images can be analyzed in tandem to improve the speed at which the images are processed. Users will not have to wait for each image to complete its cycle. Furthermore, our client has specified an upper limit on how many pairs of images we expect to analyze in a single batch of pairs. However, given time to optimize our application, (eg parallelism, porting to a fast language), it will be scaleable to larger batches of images.

Likelihood: Not very likely

Relevance: This risk is not very relevant because of the ease of mitigation.
We can use not only image segmentation on the large images, we are also not concerned about the program taking longer than 10 minutes to process a data set of five pairs of images. Our current testing (with 10 pairs of images) has already met the required benchmarks for runtime with a batch of 5 images.

Section 5.3 Comparing the wrong sets of images

When parsing file data coming across variations of naming conventions: If the naming conventions of the image file names are not in similar fashion of each-other, there could be an issue of identifying which images go together. An example of this problem is we have one pair of images named differently “image_10before.png” and “image10after.png”. For the example the parser would either prefer to have the images named with the underscore or without it, not both. There are other possible difference in naming conventions other than the underscore such as using the word hole for the word image.

Mitigation: To mitigate the variance in naming conventions we have asked our client that when they store the images they store them as a png as well as using conformed naming conventions for all sets of images. We have agreed on doing so with JPL and will have conformed naming conventions. For the PNG file form it is important that we use this for image processing as it is a lossless algorithm.

Having large individual sets of images: Most pairs of before and after images in our test data are in either pairs(b1, a1) or sets of images(b1, a1, a2). This is due to the fact that when JPL is removing dust it may take multiple blasts of air. To better calibrate the device they capture an image of the abrasion after each blast of air. (before, after_1, after_2, ….. after_n).

Mitigation: Having a data set of the images passed into the program and storing the multiple image files to the data structure will help us mitigate this problem. This will allow us to access all the names of the images and parse them for matching pairs searching for a naming convention that conforms to the agreed upon structure, such as abraded_010before.png, abraded_010after1.png, abraded_010after2.png. We can then run our tests on the multiple after images and display the result alongside the corresponding before and after images.

Likelihood: Possible, Moderate

Relevance: A relevant issue but one the can be easily avoided or worked around by communication between client and group and agreeing upon naming conventions.

Section 5.4 Costing JPL testing time

Our program not working correctly making JPL use their original method of testing: In order for the Mars 2020 mission to achieve all of its goals all of the rover's tools need to be working correctly and efficiently. If JPL is not able to trust the calibrations of their dust removal tool, then there is a possibility that the information they are getting may not be correct and any findings may be inconclusive. The laser used on their PIXL devices requires the rock to have a clear surface so that it may get the best readings possible of the abraded sediment. To make sure their tools are working properly JPL needs time to run a multitude of tests and calibrate their tools accordingly. In the end, time is money.

Mitigation: Compare JPL's original hand calculated results to our programs results and confirm correctness. During the process we will refine our program's results with improved modeling techniques and prototyping. User testing, will give us feedback on how close our client thinks our application output is to what they would expect. Implementing a validation system for our

selves to identify what is dust and what is not with 50x50px sample sections of each rock type. This allows us to validate not only based off of JPL findings but our own as well.

Likelihood: Not very likely

Relevance: This issue is the most important we would like to give JPL the best product we can for their needs so that they can have increased success on the Mars 2020 mission. Doing this would alleviate the worry of whether or not their dust removal tool will be useful on the Martian surface.

Section 6. Project Plan

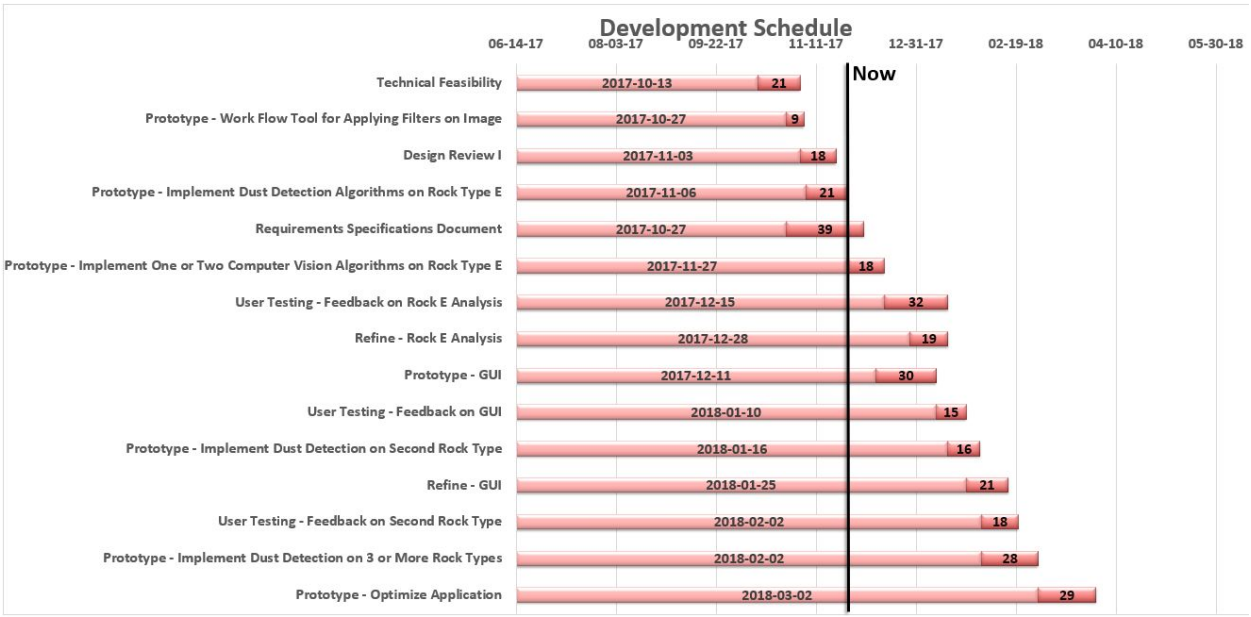

Figure 5. Gantt Chart of Current Development Schedule

Figure 3 shows our current development schedule based on where we stand right now with our project. Some milestones we plan to accomplish for the rest of the semester and into the spring semester are:

1. Finish a prototype of detecting dust on Rock Type E.
   This milestone will help satisfy the requirement of analyzing images for dust. Completing this step is crucial for moving on to the next milestone and laying the groundwork for the other rock types. This would be finished in late-November.

2. Implement one or two Computer Vision algorithms on Rock Type E and identify dust covered areas and see differences in dust coverage. After we have finished the first

milestone, we will begin applying one or two Computer Vision algorithms on Rock Type E images. Doing this will help satisfy the requirement of marking areas for dust coverage. This would be started in late-November.

3. Conduct user testing to get feedback on the Rock E analysis feedback
   This will be started once the first two milestones are completed so we can get feedback on how our program performs for our client's business. Doing this over winter break will allow us more time to work on the other rock types the rover could encounter during its mission in the spring semester. This would be started in mid-December.

4. Refine the Rock E analysis program based on the feedback
   Once we receive the client's feedback, we can make any necessary changes to the Rock E analysis program before we move on to a second rock type. This will cycle back to doing user testing until our client is satisfied with how the analysis performs. This would be started in late December.

5. Start implementing a GUI to allow the algorithms to be tweaked
   While we are waiting for the results of the user testing for our Rock Type E analysis, we will begin working on a GUI; at this stage, we will have an idea of what can be tweaked in the algorithms by a user. This milestone will help satisfy the requirement of allowing the user to adjust the parameters of the algorithms we used. This would ideally be started in mid-December.

6. Start implementing dust detection on a second rock type
   After the client's business is satisfied with how the program analyzes the Rock Type E, we can begin working on implementing dust detection on a second rock type. Ideally, we would like to begin this in mid-January.

7. Conduct user testing to get feedback on the GUI implementation
   After we have a GUI implemented, we will give this to our client to have it tested by the users. The testing will allow us to get a better understanding of how our users will interact with the results of the analysis This would ideally be started in early January.

Section 7. Conclusion

JPL's Mars 2020 rover mission is an upcoming mission to Mars that will explore various regions of the Martian surface to search for possible evidence of past life on Mars. The rover will carry an onboard suite of tools to help with its mission. One of these tools is a drill with a set of drill bits to drill into the rock and collect potential samples. Using the PIXL camera, the rover will then analyze the chemical compounds and makeup of the drilled hole. The results from this mission could be a large step forward for how we may search for life on other planets and space exploration.

The problem that we have been tasked to solve is fixing JPL's current testing process for their gas Dust Removal Tool. Their current process for testing this tool takes a long time, they need to pump down and pump up the vacuum chamber to simulate the martian atmosphere. The test could potentially be invalidated if the dust is disturbed while moving the test rock out of the testing chamber. Currently the end result is subjective, different humans have different opinions for how much of the abrasion is dust free.

Our proposed solution will be creating an image processing pipeline that will allow JPL to analyze multiple image pairs and determine how much of the abrasion is covered in dust. This will allow for multiple tests in one pump down session where our application will be able to quickly analyze the current test, and allow JPL to continue testing in the vacuum without de-pressurizing the chamber. Our application will allow the user to adjust and improve the output our application by tweaking settings for each algorithm in a graphical user interface or by modifying the source code.

This document explored the requirements that our team discussed with our client during meetings, which greatly contributed to understanding what our client expects from the finished product. This document clearly describes the functional requirements of what our application should do when analyzing a batch of images and what the end results of the analysis should output. The performance requirements were also discussed, setting hard benchmarks for how fast our application should analyze each image, how accurate it should be in its analysis, how many images it needs to be able to receive, and how long it should take for new users to learn how to operate the software. It also described environmental requirements that our application must conform to based on our client's background knowledge of programming languages and what format test images should be saved to.

We examined possible risks that we may encounter throughout the development phase and how we plan to mitigate those risks. The risk we are most concerned with is having our software incorrectly detect dust in the abrasion. This is the main problem we will solve by the end of capstone. There are other risks that we may encounter such as slow runtime and inconsistent naming conventions, these by requesting standardization of data, so that we can apply and optimize efficient computer vision algorithms.

Based on our current progress, we are on track to have a prototype for analyzing Rock Type E images by the end of the semester. We are also planning to do testing and refinement of the prototype over the winter break while also working on a GUI prototype once we have a better understanding of what parameters we can alter in the algorithms. This will allow us to be ahead of schedule come spring semester where we will have a working prototype for one of the rock types allowing us to work on analyzing other rock types.