

NASA Droplet Image Analysis Software
User Guide
May 2015

About this Document

This document describes the functionalities and use of the NASA Droplet Image Analysis Software along with installation and troubleshooting procedures.

Table of Contents

1 Preface	3
2 Overview	3
2.1 MATLAB	3
2.1.1 NASA_Droplet_Image_Analysis.m	3
2.1.2 CSV_Saver.m	3
2.1.3 Save_Images.m	3
2.1.4 MSE.m	4
2.1.5 Make_Video.m	4
2.1.6 Measure_Circle.m	4
2.2 OpenCV	5
3 Installation and Distribution	6
3.1 Github	6
3.1.1 MATLAB Files	6
3.1.2 OpenCV Files	6
3.2 Downloading MATLAB	6
3.3 Downloading the MATLAB Image Processing Toolbox	6
3.4 Downloading OpenCV	6
4 Functionalities	7
4.1 MATLAB	7
4.1.1 NASA_Droplet_Image_Analysis.m	7
4.1.2 CSV_Saver.m	9
4.1.3 Save_Images.m	9
4.1.4 MSE.m	10
4.1.5 Make_Video.m	10
4.1.6 Measure_Circle.m	10
4.2 OpenCV	11
5 Troubleshooting	12
6 FAQ	12
7 Contact Information	13
8 Appendix A: Design Document	13
9 Appendix B: Testing Document	15

1 Preface

The NASA Droplet Image Analysis Software is a package of MATLAB and OpenCV programs for determining the diameter of a moving droplets in NASA's Flame Extinguishment Experiment (FLEX) and Flame Extinguishment Experiment - 2 (FLEX-2). FLEX and FLEX-2 are both experiments conducted on the International Space Station to study fuel droplet combustion.

2 Overview

This section describes what each program, classified by platform, in the NASA Droplet Image Analysis Software package do.

2.1 MATLAB

2.1.1 NASA_Droplet_Image_Analysis.m

This program runs on a sequence of images using Circular Hough Transform and edge detection to detect the droplet in each image. Initially it prompts the user for the directory where the image sequence is located and image in the sequence to start analysis at. It then opens the first image to analyze and prompts the user to draw a circle around the droplet. The user will be able to draw a circle around the droplet, move the drawn circle, and resize it. Once the user has drawn a circle as accurate as possible, the program will start processing the image sequence.

The program processes the images by using the Circular Hough Transform (`imfindcircle()`) on the original image and the original image with a binary gradient mask, resulting in two possible detected circles. Using measurements from the user drawn circle, it looks at both possible detected circles, finds the one of best fit and records its center coordinates and radius. After the first image, it repeats this process using the measurements of the previous image instead of those from the user drawn circle. If it is unable to detect any circles in an image, it will skip that image and move on to the next image. The program will continue until all the images in the sequence have been processed.

2.1.2 CSV_Saver.m

This program takes the results from `NASA_Droplet_Image_Analysis` and saves them to a CSV file.

2.1.3 Save_Images.m

This program takes the results from `NASA_Droplet_Image_Analysis`, draws the detected circles onto the original image, and saves it as a new image.

2.1.4 MSE.m

This program compares the results from NASA_Droplet_Image_Analysis with human measured data and calculates the mean squared error.

2.1.5 Make_Video.m

This program makes a grayscale video clip (.avi) from a sequence of images (.tiff). Color images will be converted to grayscale. Program assumes that images are ordered by filename.

2.1.6 Measure_Circle.m

This program allows you to measure an image sequence manually by drawing a circle around the droplet in each image and obtaining the measurements.

2.2 OpenCV

The dropletFinder program's interface allows the user to choose from several options of operation. The program first asks the user to input a filename range to operate on. The user then is given directions on how to create the input directory to which they would like to store the analyzed images on, and where to create them.

The user can also request the program to create a video file of the analyzed images, generate MSE statistics on the analyzed images, output files of the analyzed image sequence's radii, or generate ROC curves for the sequence. These are all done at the command line prompt, when interfacing with the final compiled executable.

The program can tolerate black and white or color images, TIFF files, JPG's or JPEG's. The program does assume that the files within the file range are in the correct order to be analyzed in. Therefore, the user must be certain that all files within the file range are the correct files for dropletFinder to act on.

The program works by detecting regular polygons, specifically the machinery which initially inject fuel into the droplet. From there, the program tracks the droplet's initial location until it is released from the machinery's hold. Once released, the droplet is now in a recognizable circular form.

A backtracking algorithm is implemented in order to properly identify the droplet's location frame per frame. Based on the droplet's prior location in the image, in the previous frame, the algorithm zooms in on the same region to locate the droplet again in the next frame in the image sequence.

After the machinery leaves the camera's vision field, the Circular Hough Transformation algorithm is implemented to locate and track the droplet's path across the camera's vision space, while it is combusting. Combined with the backtracking algorithm, we are able to follow the general path of the droplet through its combustion process relatively closely and accurately.

Since the ash from the combustion can sometimes swirl into circular shapes, the sensitive Hough algorithm can be falsely triggered, and zero in on those objects as droplet potentials as well. Therefore another backup check is reinforced: no matter how many potential circles are returned, the program runs around the circumference of the detected circle, ensuring that the pixel values within the circle and outside of the circle are of significantly different values. Specifically, we check that the pixel values within the Hough returned circle are close to black, and that the pixel values outside of the returned circle are close to white or light grey.

3 Installation and Distribution

This section describes how to access the NASA Droplet Image Analysis Software package along with where to find the platforms and toolboxes needed to run the programs.

3.1 GitHub

The NASA Droplet Image Analysis Software package can be downloaded from:
<https://github.com/BamyaRhaskar/-15SD>

3.1.1 MATLAB Files

Download the package and move the MATLAB files (.m) to your MATLAB directory. For Windows users, this is C:\Users**User Name**\Documents\MATLAB. You will also want to move your image sequences to this directory. These programs require MATLAB and the MATLAB Image Processing Toolbox to run.

3.1.2 OpenCV Files

Download and install OpenCV. Download Github repo -15SD onto local machine. Download and place the relevant image files you wish to have analyzed, in the subdirectory 'currentJob' within -15SD. This directory contains the main program which can be edited depending on the situation, and all the dependencies required to build the executable dropletFinder, which will execute on the desired image range within the directory currentJob. The current format assumes that file order has been taken into account, when assigning the image range to analyze.

3.2 Downloading MATLAB

MATLAB can be downloaded from:
http://www.mathworks.com/products/matlab/whatsnew.html?s_tid=tb_15a

3.3 Downloading the MATLAB Image Processing Toolbox

The MATLAB Image Processing Toolbox can be downloaded from:
<http://www.mathworks.com/products/image/index-b.html>

3.4 Downloading OpenCV

OpenCV can be downloaded from:
<http://opencv.org/downloads.html>

4 Functionalities

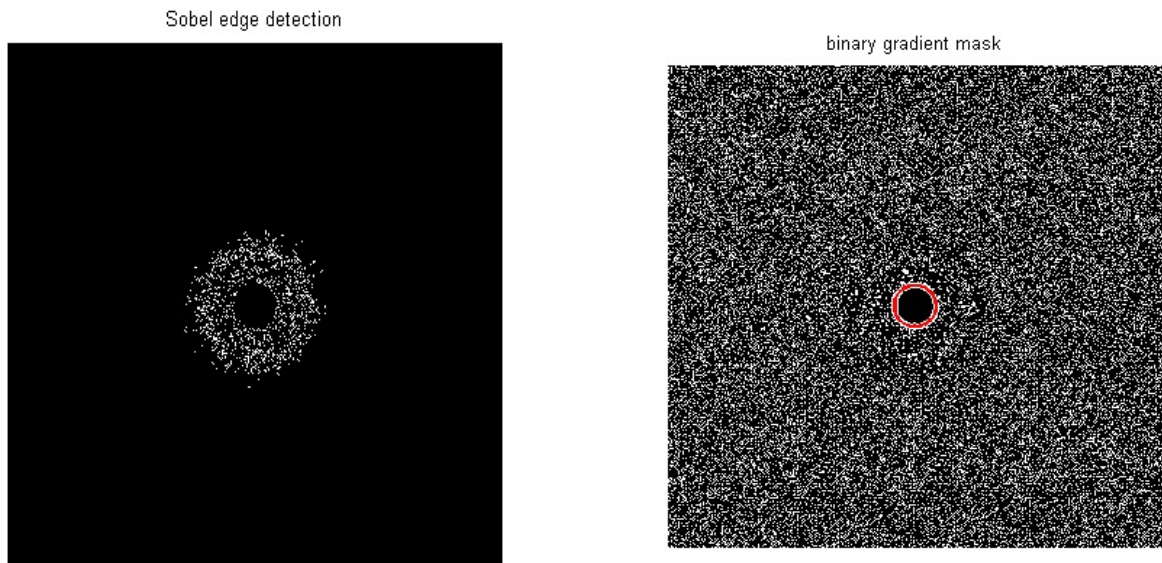
This section describes how to run each program, classified by platform, in the NASA Droplet Image Analysis Software package.

4.1 MATLAB

4.1.1 NASA_Droplet_Image_Analysis.m

1. Keep the image sequences in separate folder within the Matlab folder.
2. While script is open, press the Run button found in the editor tab or F5 to start the program.
3. Initially the program prompts the user for a directory and then the first image in the sequence that the user wants to analyze. If this is not 1, then offset it by 1, so 100 is 101.
4. Then the program prompts user for the last image in the sequence that the user wants to analyze.
5. It then opens the first image that the user wants analyzed and prompts the user to draw a circle around the droplet. It is recommended that the user zooms in to increase accuracy of the drawn circle by selecting the magnifying glass. Once the magnifying glass is selected, clicking on the image zooms in on that region, but it is recommended that you scroll forward on a mousewheel as clicking may not always zoom. If the droplet is not fully in the image, the user may select the hand tool and use that to drag the image to position it in frame. After the user is done zooming in or dragging the image, they need to click the magnifying glass or hand tool again to deselect it. They will then be able to drag a circle around the droplet, move the drawn circle, and resize it by clicking on one of the squares on the edge of the drawn circle and dragging the mouse. Once the user has drawn a circle that is as accurate as possible, the user needs to double click so the program will start processing the images.

6. The program processes the images by using the Circular Hough Transform (`imfindcircle()`) on the original image and a binary gradient mask of the original image. It applies a binary gradient mask by using Sobel edge detection on the image to obtain a threshold and then uses this threshold to do Sobel edge detection again. This returns two sets of possible detected circles found in the variables: `centersDark`, `radiiDark`, `centersDark2`, `radiiDark2`). `centersDark` and `radiiDark` represent the measurements of `imfindcircles()` on the original image and `centersDark2` and `radiiDark2` represent the measurements of `imfindcircles()` on the binary gradient mask of the image. This is what Sobel edge detection and a binary gradient mask do to the image:



7. For the first image, the program uses the measurements of the drawn image from user in step 3 and searches through both sets of the possible detected circles and finds the one of best fit and records the center coordinates and radius of the circle of best fit in the vector `Circle_Estimation`. The columns of this vector correspond to x-coordinate of center, y-coordinate of center, and radius. If `imfindcircles()` cannot find a circle in the original image, it takes the measurements from `imfindcircles()` of the binary gradient mask and stores them in `Circle_Estimation`. If `imfindcircles()` cannot find measurements on either the original image or the binary gradient mask, then it skips the image and continues to the next.
8. After the first image it repeats steps 4 and 5, but instead of using the measurements of the first image, it uses the measurements of the previous image which is stored in `Circle_Estimation`. It will continue till all the images have been processed. You may

see some warnings in the command line as the radius of the circle becomes so small (10 pixels) that it loses accuracy.

9. After the program is done, you may want to save the workspace, so click on the home tab of MATLAB and click save workspace. Doing this will allow to you import all the data used by the program when it has been run because after MATLAB is closed the workspace is cleared.

4.1.2 CSV_Saver.m

1. This script must be run after NASA Droplet Image Analysis.m or you must import the workspace that was saved after NASA Droplet Image Analysis.m was run.
2. Press the Run button found in the editor tab of matlab or press F5 when the script is open and selected.
3. Prompts the user for the total number of the images in the sequence. Enter the total number of images in the sequence. Make sure if the sequence starts at 0 to include that into the count.
4. Creates a total images by 5 matrix to save results.
5. Stores the Sequence number in the first column.
6. Takes the radiuses in pixels from the Circle_Estimation matrix and stores them in column 2 of the results.
7. Multiplies the radiuses in pixels by 2 to obtain the diameter and stores this in the third column of the results.
8. Multiplies the radiuses in pixels by the scaling factor ($30.25/1024$) to get the radius in millimeters and stores it in column 4 of results.
9. Multiplies the radiuses in millimeters by 2 to obtain the diameter in millimeters and saves this in column 5 of results.
10. Prompts the user asking where to save the CSV file.
11. Write the results to a CSV file and prompts the user that the results have successfully been saved.
12. The CSV file will appear in the MATLAB folder.

4.1.3 Save_Images.m

1. This script must be run after NASA Droplet Image Analysis.m or you must import the workspace that was saved after NASA Droplet Image Analysis.m was run.
2. Press the Run button found in the editor tab of matlab or press f5 when the script is open and selected.
3. Prompt the user for the first image in the sequence.
4. The user needs to enter the first image in the sequence, if it is zero, then enter 1.
5. Prompt the user for the last image in the sequence.
6. The user needs to enter the last image in the sequence.
7. Show the original image.
8. Project the estimated circle onto the original image.
9. Save the image. Warning: This will overwrite files with the same name.

4.1.4 MSE.m

1. In the home tab of MATLAB, click import data button. Select the file with the actual data you want imported.
2. Click import selection and it will tell you the name of the vectors that were imported.
3. Locate the imported variable name of the vector in the workspace that corresponds to the diameters.
4. Double click on it in the workspace to change its name to dpixels.
5. Go to the editor tab and hit Run or press F5.
6. Important check to make sure the number of values do not include NaN or you will get Nan for your result.
7. Prompt the user for the number of actual measured values not including NaN.
8. Enter the number of actual measured values.
9. Prompt the user for where the predicted measurements start in Circle_Estimations.
10. Enter where the predicted measurements start. If measurements do not start at 1, then add 1 to the starting point.
11. Prompt the user for where the predicted measurements end in Circle_Estimations.
12. Enter where the predicted measurements end.
13. Mean Square Error is calculated and result is show in the variable MSE1.

4.1.5 Make_Video.m

1. Run the program in MATLAB.
2. Program will prompt user for the folder containing the image sequence.
3. Video clip will be saved in the folder containing the image sequence.

4.1.6 Measure_Circle.m

1. Since this code is used to measure droplets by manually drawing circles, it will save progress automatically if you decide to close matlab and come back later. Only works on actual images.
2. First it tries to load a workspace "measured.mat" so that it can resume work. If the workspace does not exist, then it creates one.
3. Prompt user for the directory of the image sequence.
4. Prompt user for the first image in the sequence they want to analyze.
5. User enters the first image number in the sequence. If the first image number is 0, then all results will be offset by 1.
6. Prompt user for the last image in the sequence they want to analyze.
7. Check if the matrix to hold results exists, if not create the matrix.
8. Show the image to the user and ask them to draw a circle around the droplet. Double click when finished.
9. Obtain width, height, x-coordinate for center, y-coordinate for center, and radius.
10. Save results to the matrix Circle_Estimation.
11. Repeat steps 8-10 until finished. You double click on Circle_Estimation to view results.

4.2 OpenCV

The script `dropletFinder.cpp` combines various functionalities into one program, and direct control of the program is transferred to user once the executable has been built. To build the executable, first navigate to directory `currentJob` in Terminal on Mac OSX. Then, while within Terminal, type the command:

```
/Applications/CMake.app/Contents/bin/cmake .
```

Then press the 'Enter' key, to rebuild all the binary files for the OpenCV libraries, if they were recently changed or altered.

Then type `'make'`, and then press the 'Enter' key to remake the executable `dropletFinder`, to account for any changes made to the main file `dropletFinder.cpp`.

To generate MSE statistics, videos of desired images, drawn images, and various other statistics as outlined in the program interface, simply follow the commands while navigate through the program's user interface.

To create a video file from the command line, simply follow the directions outlined in the user interface of `dropletFinder`, or simply:

Navigate (`cd` into) to directory with all of the desired images for which you would like to make a video file of. The directory should not contain any other files other than these. Once there, type the command:

```
convert -set delay 20 -colorspace GRAY -colors 256 -dispose 1  
-loop 0 -scale 50% *.jpg Output.gif
```

to create a GIF of the analyzed images for performance evaluation.

Since OpenCV is assumed to be running on either Mac OSX Yosemite 10.10 systems or later, it is also assumed that the Unix command tool pack ImageMagik is also installed as well. However, since ImageMagik is generally preinstalled on Mac's, if it is not present on the machine in question, the user is advised to install macports, and then grab the ImageMagik package via that. To install macports, if it is not already installed visit: <https://guide.macports.org/chunked/installing.macports.html>

5 Troubleshooting

This section describes basic troubleshooting steps. If you have further questions, please refer to FAQ or feel free to contact us.

Tips for Debugging:

1. Create breakpoints in code (Place where code should stop by click the - next to the line number of the code)
2. Check the value of variables by clicking on the variable in the workspace.
3. If an error occurs check for the error referenced in the script and click on the line number to be brought to the error.
4. If you run into dimensions or index errors check the dimensions of the variables.

6 FAQ

1. **Q:** Where do I find my MATLAB folder?
A: Please refer to this page for help:
http://www.mathworks.com/help/matlab/matlab_env/matlab-startup-folder.html
2. **Q:** Where should I put the image sequences and MATLAB programs?
A: They should be put in your Matlab folder.
3. **Q:** Where do I find the images and CSV files I saved?
A: They can be found in your Matlab folder.
4. **Q:** What are the commands to save and load a workspace?
A: save('FileName.mat') and load('FileName.mat')
5. **Q:** How do I kill a running program?
A: Ctrl + C
6. **Q:** How to clear the workspace?
A: Type clear in the command window.
7. **Q:** How to clear the command window?
A: Type clc in the command window.
8. **Q:** I'm still having trouble installing OpenCV. Is there another resource?
A: <http://blogs.wcode.org/2014/10/howto-install-build-and-use-opencv-macosx-10-10/>

7 Contact Information

Ramya Bhaskar	rbhaskar@ucdavis.edu
Amanda Ho	ayjho@ucdavis.edu
Willie Huey	whuey@ucdavis.edu
Rylan Schaeffer	ryschaegger@ucdavis.edu

8 Appendix A: Design Document

Technology Background

Fuel combustion in space is of interest to NASA. It is relatively easy to burn a droplet of fuel, but it is time consuming for a human to measure how that droplet's diameter changes over time as the combustion progresses. As a result, an undergraduate student, Vincent Tam, started developing methods to automate this process. We will be reworking/building upon MATLAB code previously written by Vincent Tam and modified to allow for diffraction analysis by Fei Yu, a UCD graduate student. The aforementioned code finds the edge of a droplet in a non-noisy JPEG image and calculates the diameter. The diffraction analysis is not complete so it is difficult to run on a sequence of images. We will be testing the software using images previously collected through NASA.

Design Goal

Our primary goal is to create a program that takes a TIFF image as input, identifies the droplet and calculates the radius of the droplet even in the presence of heavy sooting. We need to automate this process so that analysis on hundreds of thousands of images will require minimal work on the user's part. Secondary goals include diffraction analysis, measuring the flame around the droplet and, calculating the uncertainty of the droplet radius and flame measurements.

Architectural choices and corresponding pros and cons

Matlab

Matlab is a high-level language with an assortment of functionality such as signal and image processing, communications, control systems, and computational finance.

Matlab Pros

Provided code is written in Matlab. We are all very familiar with Matlab and have used it for statistics and machine learning. It has a image processing toolbox that provides a

comprehensive set of reference-standard algorithms, functions, and apps for image processing, analysis, visualization, and algorithm development. The IDE for Matlab has a very nice layout that allows for easy access to the results of our program.

The fact that all the functions that are utilized in the build process are Matlab standard functions guarantees that the functions implemented are extremely high quality, since Matlab is an industry standard, at the upper limit of accuracy.

Matlab Cons

Doing the image analysis on matlab can take a long time to execute. Since time efficiency is of utmost concern, this could potentially be a significant downside.

C++ and OpenCV

OpenCV is an open source computer vision and machine learning software library. Built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception. Written in optimized C/C++, the library can take advantage of multi-core processing.

C++ and OpenCV Pros

It was designed for computational efficiency. Has more than 2500 optimized algorithms, which includes a comprehensive set of computer vision and machine learning algorithms. Algorithms can detect and recognize faces, identify objects, track moving objects etc. Since C++ is a compiled language, does not require an interpreter, and deals directly with the memory addresses, code can run orders of magnitude faster.

C++ and OpenCV Cons

We have never really used this library before so we would have to spend time learning it in order to use it effectively. The code would have to be built entirely from scratch.

Selected architecture

We selected Matlab because the code we were given was already developed in Matlab. It is also more familiar to us than openCV. If we were to switch it would be very time consuming to have to redo all the previous work that was given. Instead of spending time redoing everything we would rather spend that extra time improving the product for our client. However, we are strongly considering using OpenCV/C++, and have been experimenting with it as well. If we were to start the transition early, we could still be on track for completing our base goals.

Implementation notes

We will be implementing methods for the program to handle TIFF files. We need to ensure that the user will be able to do image analysis on one or many images in a sequence. We need to have an effective way to display the output, possibly in different formats. We will also

be implement methods to deal with heavy sooting so that we can accurately find the edge of a droplet and calculate its diameter. One possible way to tell soot from the droplet might be to find the grayscale threshold to distinguish the two. If we have time, we would definitely like to be able to implement previously developed algorithms for diffraction analysis and flame diameter to add additional functionality to our product. Finally, if time permits, we would also like to be able to implement methods to calculate the uncertainty of all the measurements we produce.

9 Appendix B: Testing Document

Summary of Product to be Tested

The product will be used to identify and calculate the change in fuel droplet diameter as a function of time. Droplet images are in the form of TIFF image sequences. The sequences of interest are provided by NASA through Professor Benjamin Shaw. Each sequence contains approximately one thousand individual images, each exhibiting minimal to heavy sooting. Sooting is a phenomenon in which the fuel droplet becomes obscured because of the ongoing combustion reaction. Not all images will be grayscale—some are taken with other types of cameras and other methods will need to be implemented to detect the droplets captured with different cameras.

Resources Required for Testing

MATLAB with image processing toolbox <http://www.mathworks.com/products/matlab/>

NASA image sequences must be acquired from Benjamin Shaw or NASA

Note: The computers we tested our program on had 8GB of RAM. Using a computer with less RAM may impact time to complete.

OpenCV assuming all dependencies (CMake, ImageMagik) are properly included after installation of OpenCV from <http://opencv.org/> then there are no further software packages required.

Estimated Hours for Testing


Since our project is based on comparing the circle of best fit to the droplet, the testing process is not independent of the development process. As a consequence, we expect to spend over 30 hours of testing, as a rough estimate.

Packaging, Building, Configuration and Options

1. Github Repo: <https://github.com/BamyaRhaskar/-15SD>

The public availability of the code has yet to be determined. Professor Shaw will let us know whether the code should be publicly released.

To copy the repo onto your home machine, please follow the steps outlined below:

1. On GitHub, navigate to **your fork** of the Spoon-Knife repository.
2. In the right sidebar of your fork's repository page, click  to copy the clone URL for your fork.
3. Open Terminal (for Mac users) or the command prompt (for Windows and Linux users).
4. Type `git clone`, and then paste the URL you copied in Step 2.
5. Press **Enter**. Your local clone will be created.
6. To keep your repo up-to-date, cd into the repo on your home machine (the directory -15SD). To properly do so on Mac's, the command to cd into -15SD is: `cd -- -15SD`.
7. Once you are in -15SD's directory, type: `git pull`. Try to do this over a stable internet connection-the changes can be large and may take some time to fully catch up with what is currently on your machine.

2. Checkout, Build on the Major Elements

Matlab: The repository for the code needs to be downloaded. After they have been downloaded, the folders containing the image sequences must be placed in the Matlab folder (For Windows, this is located in the Documents folder). Run the `visualize_circle.m` script and enter the images you want to analyze.

OpenCV: The repository for the code also needs to be downloaded. Once that is done, we focus on the process for installing OpenCV (for Macs only at the moment).

Note, that this will require installation of several other helper applications.

1. First, make sure that the Mac OS X operating system is up to date, with Yosemite installed.
2. Install XCode if it hasn't been installed already.
3. Make sure that Command Line Tools has also been installed (can be completed from within the XCode Application).

Once the above is done, the application CMake will have to be installed. In order to do this, simply visit the website: <http://www.cmake.org/download/> and follow the site's instructions.

With CMake installed, we can then proceed to install OpenCV.

Download OpenCV somewhere on computer. Create two new folders in the OpenCV directory, called SharedLibs and StaticLibs. Open CMake application, click browse source and go to the OpenCV folder. Click browse build and go to StaticLibs folder. Click configure

button, click unix makefile (assuming working with a Unix environment). After tests are performed, uncheck the following boxes in red.

1. BUILD SHARED LIBS
2. BUILD TESTS
3. WITH 1394
4. WITH FFMPEG

5. Add an SDK path to CMAKE OSX SYSROOT :
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.9.sdk

6. Add x86 64 to CMAKE OSX ARCHITECTURES

Click Configure again, then Click Generate. Now, open Terminal and type:

1. cd <path/to/your/opencv/staticlibs/folder/>
2. make
3. sudo make install

Type in password and StaticLibs libraries should then be installed. Repeat same above process for SharedLibs, except when you first click browse build, go to SharedLib folder instead, and in the CMake phase, recheck BUILD SHARED LIBS.

If you are still having trouble with installation please refer here for additional support:

<http://blogs.wcode.org/2014/10/howto-install-build-and-use-opencv-macosx-10-10/>

With OpenCV now successfully installed, the code can be executed by the following steps:

1. navigate to directory with appropriate code
2. g++ hough.cpp
3. then you get an executable called hough
4. and then you get an executable called: ./hough
5. and you run as: ./hough *file name that we want to analyze*

3. How to Install and Configure the Elements

Matlab: Currently, the path names are hardcoded to make testing easier. The final program should allow the user to input the directory they want to work on. Directory should be included in the Matlab directory for ease.

OpenCV: Covered in above section.

4. Sanity Check Basic Function to Ensure Its Built & Installed Properly.

The code is well documented, with a number of user prompts. Code should execute without error if built properly.

Process for Defect Reporting & Repair:

Matlab: To ensure that the program runs smoothly make sure there are no errors when running the script. If there are errors, ensure that the directory with the image sequences are within the Matlab path, if not, add the directory to the Matlab path. When running the program, ensure at the beginning that you input image numbers that are within the bounds of the images in the directory.

OpenCV: If there are issues configuring OpenCV please go through the instructions again to make sure you haven't missed anything. If you have further issues please contact Ramya Bhaskar.

Quality Control:

There are two types of bugs: false positives and failures. False positives are difficult to detect because the algorithm reports success (a circle of best fit has been found) and can only be caught by manually inspecting the image or checking the position of the circle and its radius. If a false positive is reported, that specific image sequence will need to be examined to understand why the algorithm produced a false positive.

Failures are also difficult to decide what to do with. In some cases, the image signal isn't present and nothing can be said about the placement of the circle of best fit. But in other cases, the algorithm might report that a circle of best fit does not exist, despite the droplet being clearly identifiable. In the latter case, we will approach the error similarly to false positives.

Developer Repair:

When a bug report is fixed, and it passes regression tests, we will be using Github to commit and push the updates to the testing team and leave a comment on facebook to ensure the developers see the changes.

Functional Test Plan

We will be testing our code by checking our results by eye. In most cases, we will be able to confirm if the area of interest is correctly identified. Furthermore, we will manually run the `imellipse` function in Matlab on an image series and compare the results to the corresponding automated results from our code. By checking the coordinates and measurements of the result, we will be able to determine the accuracy of our solution. Our client will also provide us with a series of simulated images for evaluating the accuracy of our code in the presence of extreme sooting. We will consider the results of both Matlab and OpenCV approaches when deciding whether we will continue with one or both solutions.