Willie Huey
Amanda Ho
Rylan Schaeffer
Ramya Bhaskar

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 an 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.

Matlab Cons
Doing the image analysis on matlab can take a long time to execute. Since time efficiency is a concern, this could potentially be a significant downside. There are also some issues with how Matlab deals with Tiff images such as some functions not working on Tiff images. This limits the amount of available resources we can use to improve our product.

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 and 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.

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. Getting OpenCV to run on windows might be troublesome since C++ doesn't run natively on windows.

Selected architecture
After developing and testing an algorithm in Matlab, we decided to pursue both in parallel. We don't have enough experience with either to know what problems might arise or which toolbox will ultimately prove to be more accurate or faster. Working with both Matlab and OpenCV will allow us to deliver him the product of highest quality.

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 greyscale 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.

Matlab: Currently, our code uses the circular hough transform in order to find the circle of best fit (This is what we want our droplet to be). To improve prediction accuracy due to detecting false circles, we use previous images in the sequence to predict where the circle of best fit should be next. However, our current methods are still not accurate enough. Because image analysis is relatively new to the four of us, we decided to explore in different directions. One method we are considering is to zoom in on the area of interest before identifying the droplet. Hopefully this will reduce the likelihood of false positives, as the area of interest will have less quasi-circles to confuse the function. Another method we are studying is applying a gradient filter to identify edges more sharply. Given that the circular Hough transform requires a sensitivity and threshold parameter, a third method we are studying is using dynamic error testing to vary the parameters. We would also like to apply machine learning techniques by training our program on simulated images to improve prediction accuracy.

OpenCV: Currently the code also uses the circular hough transform in order to find the circle of best fit, but also selects for any other possible droplet-like shapes that could also be present, and may occasionally present multiple options. However, the threshold for detecting such potential droplets is relatively high, thus it is more conservative than we would like it to be. The circular hough transform method does not execute until the image has been modified a bit to assist in a more accurate performance. The image is first blurred to remove any excess noise, then the contrast is increased by a certain user specified amount to further amplify any potential signals in the image that the hough circle transform can pinpoint, and then sharpened to increase precision. Eventually, we would like the program to become “smarter” in deciding when and how much of each modification to implement, and perhaps try out a range of possibilities before settling in on one modified version of the image to use as the final analysis. This should be far more feasible in C++ since the speed offered by C++ greatly exceeds that of Matlab.