**Software Design Document for**

**Image Manipulation Project**

**Version: 1.2**

**Savi Medlang**

**4/30/18**

**Table of Contents**

1. **Introduction**
   1. **Purpose**
   2. **Project Scope**
2. **Design Overview**
   1. **Data Structures**
   2. **Sample Output**
3. **Functional Design**
   1. **Data Flow**
   2. **Files**
4. **Introduction**
   1. **Purpose**

The purpose of the design document is to provide detailed information to developer, including the setup, design, functionality, and deployment of our initial concepts and plans. By reading this document, developers should not need to consult the authors of the document in order to start implementing the Image Manipulation Project.

* 1. **Project Scope**

PicFilter provides the users with a way to load a picture and have it filtered. The users can load any jpeg image, and will be returned their image with a filter applied to it. They can choose from 7 different types of filters, indicated by an integer.

* 1. **Overview**

In the following sections, I will outline the entire project. Starting with the design overview, functional design, and a technical overview, including the classes and methods that we plan on implementing. The robustness, constraints, and technical design will be outlined in the sections below. We will also discuss performance requirements and environmental requirements.

1. **Design Overview**
   1. **Data Structures**
      1. Array of threads

This array contains 16 threads, which is constant. 16 threads will be created upon execution of this project.

* + 1. Mat object

This object contains the image and its RGB values. The Mat object comes from OpenCV, a C++ library for image and video data. Each specific RGB value can be accessed by the first, second, and third indices of the specific pixel.

E.g.

image.at<cv::Vec3b>(i,j)[0]

This command would access the **blue** value of the pixel in the ith row and jth column.

* 1. **Sample Output**

Your choice of filter will be printed. Also, the original and the new image will be displayed.

The first image output is: Remove all red from the photo!

The second image output is: Grayscale photo!



1. **Functional Design**
   1. **Data Flow**

The image can be loaded by typing in the filename at the top of the file. When running the executable, the user must specify a number 1-7 to indicate which image manipulation should be done.

e.g. ./im 5

to run the 5th filter

First, the original image is displayed to the user. Then, 16 threads are immediately launched. Each thread has access to the globally stored image. Based on the specific thread ID, each thread manipulates different pixels, which allows this to run concurrently with sixteen threads.

When the original image is displayed, the user must press any keystroke to continue the program. The manipulated image will then appear so that the two images can be compared.

* 1. **Files**

imageManipulator.cpp

The image manipulation happens concurrently by using 16 threads to each work on 1/16th of the image. Also, shared memory is used so that all threads can access this image.

Note: to compile the project, run this command:

g++ $(pkg-config --cflags --libs opencv) imageManipulator.cpp -o im -lpthread