**A Simple Image Editor**

CAI Wenjing 116010004

LAN Yifan 116010103

YE Chengwei 116010265

Course coordinator: Prof. Rui Huang

CSC3002: Programming Paradigms II

The Chinese University of Hong Kong, Shenzhen

**Introduction**

Nowadays, people are taking dozens of photos every day, whenever they want and wherever they are, yet problems come when people are not satisfied with the photos they take. Very often people would like to make various adjustments to photos, and almost every individual has a demand for a comprehensive and user-friendly photo processing tool.

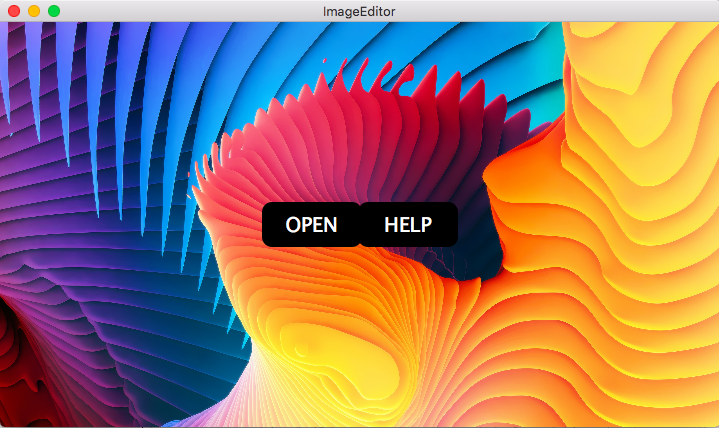
In this project, we have succesfully designed and implemented a powerful and user-friendly image processing software. It is powerful in the sense that it supports a variety of image processing operations, including but not limited to adjusting degrees of lights and colors, adding thematic filters, as well as altering geometric shape. It is user-friendly in the sense that it not only provides the users with a concise operating interface, but also supports convenient operations such as opening an image, undoing, and saving image as output.

The product of our project introduces a great solution to the lack of powerful and free image processing tool on Windows, enabling the users to perform multiple tasks on an imperfect image and eventually obtain a perfect image, with the help of a single software.

**Achievements Descriptions**

* Interface overview

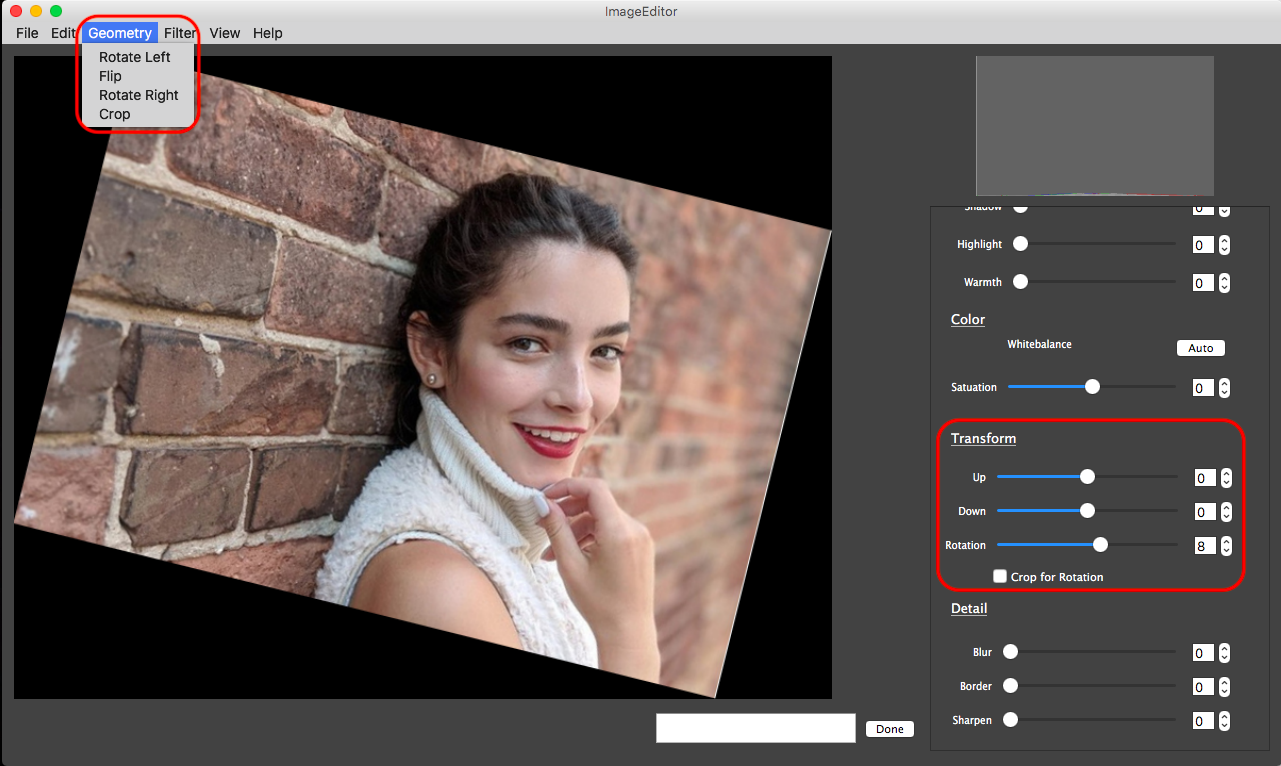
In terms of the graphical user interface, we design two interfaces – the welcome interface and the develop interface. The welcome interface contains an “OPEN” button and a “HELP” button. The develop interface is divided into three parts – the middle part, the top part, and the right part. The image is placed in the middle. On the right part, there is a function plate displaying the basic editing functions as well as the histogram. A tool bar is placed on the top part, containing six buttons for “File”, “Edit”, “Geometry”, “Filter”, “View” and “Help”.

* Welcome interface

The concise welcome interface will show up first upon running the project. A few slices of documentation will be shown if the user clicks the “HELP” button. After clicking the “OPEN” button, the user will be asked to select an image to work on (through a pop-up window).

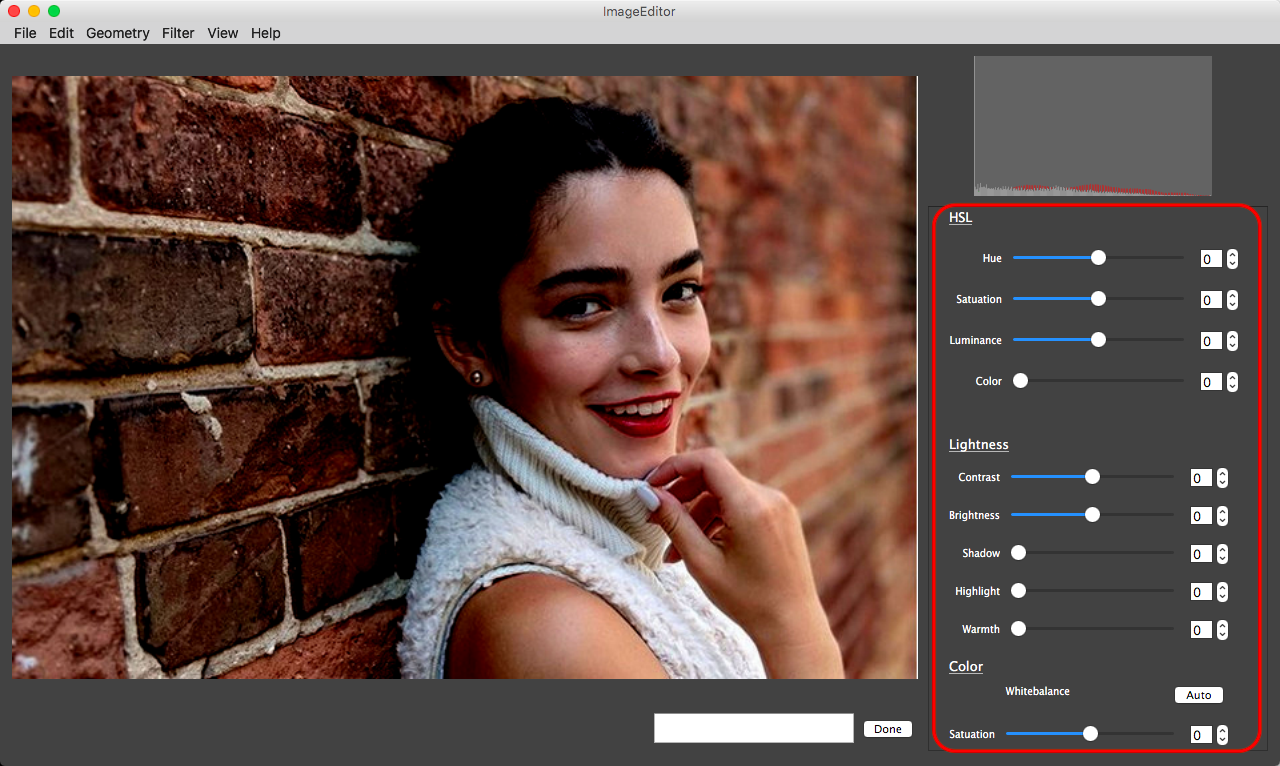
* Geometric transformations
  + Crop;
  + Flip;
  + Rotate: any degrees / clockwise 90 degrees / counter clockwise 90 degrees;
  + Transform: horizontal / vertical.

Functions of Horizontal transform, Vertical transform, and Free rotation appear on the function plate on the right part of the interface, while functions of Rotate right (clockwise 90 degrees), Rotate left (counter clockwise 90 degrees), and Flip appear on the tool bar on the top of the interface. Due to limited space, only the Free rotation function is shown here.

****

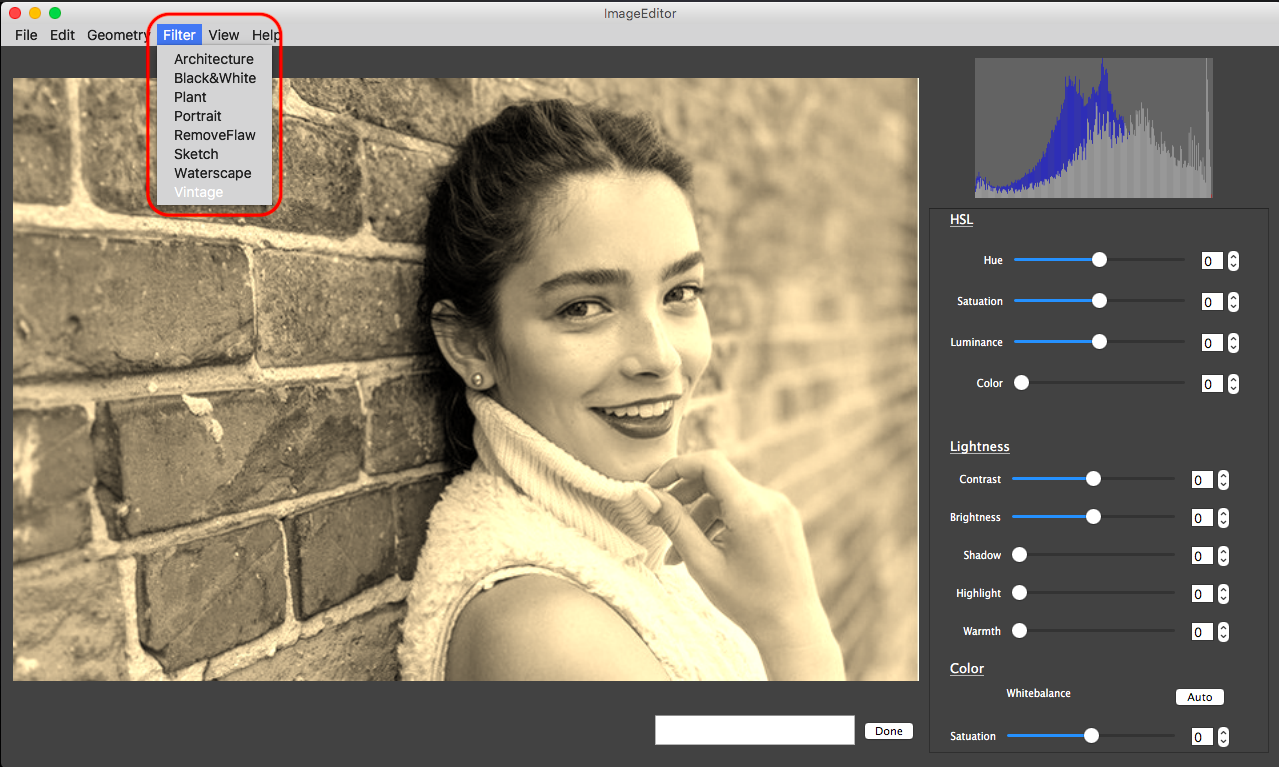
* Image adjustments
  + Color adjustments
    - Class: imgHSB (Hue, Saturation, Brightness)
    - Functions: imgBrightness, whiteBalance, imgWarmth, imgSaturation
  + Light adjustments
    - Class: imgSDHL (Shadow, Highlight)
    - Functions: imgContrast, imgShadow, imgHighlight
  + Other adjustments
    - Sharpen
    - Blur

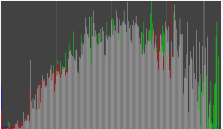
In consideration of encapsulation, we implement two classes. The imgHSB class is for color adjustments, with its key instance variables being Hue, Saturation, and Brightness. The imgSDHL class is for light adjustments, with its key instance variables being Shadow and Highlight. We also implement other functions for possible needs, including Sharpening, Blurring, and Border detection. All these functions appear on the function plate on the right part of the interface, with the name of each slider demonstrating its function clearly, which enable beginners to get started without much difficulty. Due to limited space, only the Shadow function is shown here.



* Filters
  + Thematic: Portrait / Waterscape / Plant / Architecture;
  + Artistic: Vintage / Sketch / Black&white / Border detection.

Based on the image processing functions and the imgHSB class introduced above, we design and implement several image filters. All the filters appear on the tool bar on the top of the interface. Due to limited space, only the Vintage filter is shown here.

****

* Histogram

A histogram is placed at the top-right corner of the interface. It displays the brightness of the red channel, the green channel, the blue channel, and the whole channels. Every time the user applies a filter or adjusts the lights or colors, the histogram will change.

**Difficulties and solutions**

There are two major problems that we have encountered when we design the data structure. The first one is the performance problem. The second one is the problem of keeping the history of changes to images.

* Performance problem

If the performance problem is not solved, when the user uses the slider to perform the adjustment, he will encounter serious slowness, which cannot be tolerated. But immediate preview is an important feature of a modern image editor. However, when the user presses buttons, the performance problem is not so serious. Based on this fact, the software generates a smaller image for adjustment. The size of small image is independent of the original image, which means that the size of original image does not affect the performance of the software. When the user finishes a slider adjustment, he has to press a button to confirm that he has finished. Then the software will apply the changes to the original image. The performance problem is solved.

* History problem

We want to keep all the history of changes. That means the user can regret and go to the step before the last step. To realise this function, we designed a class called “main\_pic”. Its inner structure consists of two vectors containing all the large images and small images. We designed methods that enable the software to store or remove images just like a stack. Because cv::Mat has garbage collection mechanism, we do not worry about the memory storage. Every change we make does not alter the original image. Instead, it generates a new image and puts it in the “main\_pic”. It is similar to the idea of “layer”.

With the help of “main\_pic”, we enjoy the convenience of keeping every history versions of images, while the overall performance is also guaranteed.

**External resources**

1. Text books

*Digital Image Processing\_3ed*, by Rafael C.Gonzalez & Richard E. Woods

*Programming Introduction to Qt5*, by Yafei Huo & Liang Cheng

*OpenCV By Example*, by Prateek Joshi & David Millan Escriva & Vinicius Godoy

2. openCV

We make use of some of the functions and classes in openCV, such as the Mat class, the imread & imwrite function, the cvtColor function, and the mouse event.

3. Codes shared by others

We make use of some codes shared by others on the Internet, such as the HSL class and the colorspace\_trans function implemented by JoStudio. We have made a declaration in every file written by others. Since we are not familiar with all the image processing principles, for some of the processing functions, we learned the principles from the Internet, but finished the implementations on our owns.

**Team members and division of work**

|  |  |
| --- | --- |
| **Division of work** | |
| **Name** | **Division** |
| CAI Wenjing | 1. Design and implement GUI.  2. Implement the connections between processing functions and interface components. |
| LAN Yifan | 1. Implement the data structure to store and display images.  2. Implement operations including undoing, redoing, and confirming change. |
| YE Chengwei | 1. Implement image processing functions that support geometry, light, and color adjustments.  2. Design and implement several filters. |

**Summary**

Through this project, we have fully practised the idea of object-oriented programming by designing and implementing several classes, and we have designed suitable data structures for our data in order to obtain good performance. These are highly related to the knowledge taught in classroom.

Besides, we have finished our first try of designing a concise and elegant graphical user interface. Moreover, we have gained comprehensive knowledge of image processing principles and methods in both image scale and pixel scale.