CIS 581: Computer Vision and Computational Photography
Project 3, Part B: Mini-Project
Lightweight Image Manipulator

By Tong Pow, Frank Fan

November 2nd, 2017

Abstract

In this project we utilized the tools we have learned in Computer Vision so far and made a simple user interface to play around with those tools, mostly focusing on filtering images by using convolution and seamlessly carving images by using dynamic programming. By using the Python Tkinter library, we were able to create the GUI necessary to handle the work and control both the inputs and outputs of our program. For the filtering and carving parts of the project, we mostly reused the code we have implemented for previous projects and made certain adjustments and refactoring in order to make the final project work. Overall, the User Interface we have designed is a simple tool for people who want to make basics changes to their images. Going forward, we are excited to bring more features to this user interface such as object detection, facial recognition, or even some more advanced image processing techniques through machine learning.

Introduction

In the modern age of prevalent digital content, image editing has become a common no longer reserved for photo professionals. Whether for social media, professional profiles, or other purposes, people with no experience in photo editing all need to manipulate their images in some way. Professional photo editing tools such as Adobe Photoshop or Gimp provide such a wide variety features that even professionals might not have heard about or haven't used. Those without image editing experience might not need specific functions such as adjusting each individual color hue of an image. As a result of such variety, the barrier to entry in learning these software are high and depending on the user, many functions are often not necessary in practice.

In order to lower the barrier to entry for those without much photo editing experience, we seek to create a lightweight photo editing interface that serve the most fundamental and prevalent functions: seam carving/content-aware resizing and filtering. While cognizant of social media platforms' mobile image editing features such as those in Instagram, our photo editor targets the niche of desktop/laptop users, caters beyond social media, and simplifies the convoluted image manipulation process using the latest technologies. Moreover, with most of the advanced photo editors usually charging a small amount of fees or usually filled with advertisements, our user interface creates a free, commercial-free platform for people who love photos or playing around with photos.

Related Work

Similar lightweight image editing frameworks have been offered in popular social media apps such as Instagram. However, their limitations lie in their mobile-focused base and app-specific bounds. Photoshop have pioneered and popularized the mentioned photo-editing functions, but their comprehensive offerings go in accordance with a higher barrier to entry. The resizing function is based on the work from "Seam Carving for Content-Aware Image Resizing " by Avidan, S. & Shamir in 2007, a paper foundational in most of the resizing functions of image manipulation software used today.

Just recently, software company Adobe launched a series of AI tools in image processing (detailed information: https://www.theverge.com/2017/10/24/16533374/ai-fake-images-videos-edit-adobe-sensei). The company has previously released automatic “selfie tweaker”, and now it is utilizing artificial intelligence to seamlessly make all kinds of edits on not only photos but videos as well. Our image manipulator is surely not at that level of complexity, but it is certainly great to see that there has been a tremendous amount of work done in the related field by such a huge software company.

Proposed Methodology

1. Create a simple user interface that could allow the users to open up an image that he/she wants to make edits to
2. Create 3 buttons for filter selection and one button with two entry fields for content-aware resizing.
3. Implement those two functions to make it applicable for all valid input images, based on what we have learned so far in classes and projects, but also adjust those for better function performance
4. Create two buttons that have the features of redoing and saving; one simply allows the user to redo the action he just did; the other allows the user to save the new image using any name

Implemented Methodology

1. Refactor the code used in previous projects to improve certain functions being used in the user interface such as blurring, sharpening, un-sharpening, and carving; make sure the function takes in reasonable types of inputs so that it will work when integrated into the user interface
2. Create the simple user interface that allows the users to upload an image to make edits with; the user interface contains four buttons: Open, Blur, Sharpen, “Unsharp” Mask, and Carve
3. Have the program follow the instruction given by the user such as Blur the image etc.; allow users to continuously make edits to an image, such as continuously blurring the image.

4. Create two entry boxes to take in inputs from users about the number of rows and columns he/she wants to carve out of the image; the program then takes in those two numbers along with the uploaded image to implement seamless carving through dynamic programming

Comparison between Implemented and Proposed Methodology

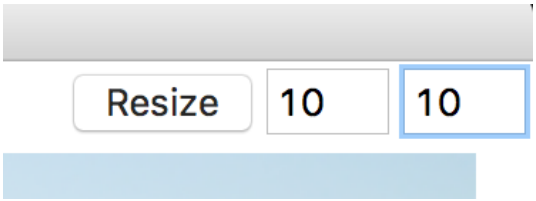
Due to time constraint, we were unable to finish implementing the redoing and saving features as we proposed in our project proposal. Ideally, those two features should be straightforward to implement; in our actual methodology, we spent more time than we expected in refactoring the code we had, mostly on seamless carving. Seamless carving could take a long period of time to perform depending on the algorithm and number of looping happening inside the function. In order to make sure that every feature works relatively fast for the users, we optimized the run-time of carving function so that the user does not need to wait for too long for the carved image.

Future Work

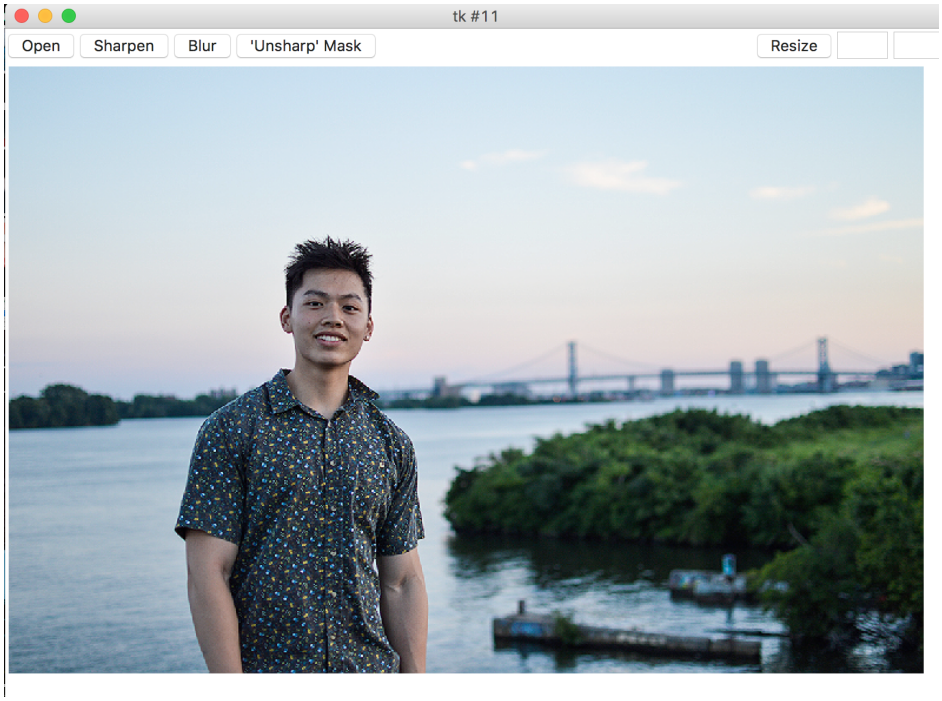
Going forward, we can take this project to the next level by first completing the basic saving feature. We then would polish up the content aware resizing function through further optimizing its run time and possibly incorporating live-time dragging for the resize function. In addition, using other projects that we have done, such as edge detection and image morphing, we can incorporate features into our image editor such as having it take in two pictures, automatically detect their edges, and create output images and videos of the morphed result. Ultimately, we want this editor to be accessible, lightweight, and include all the essential features for an average photographer.

Achieved Result

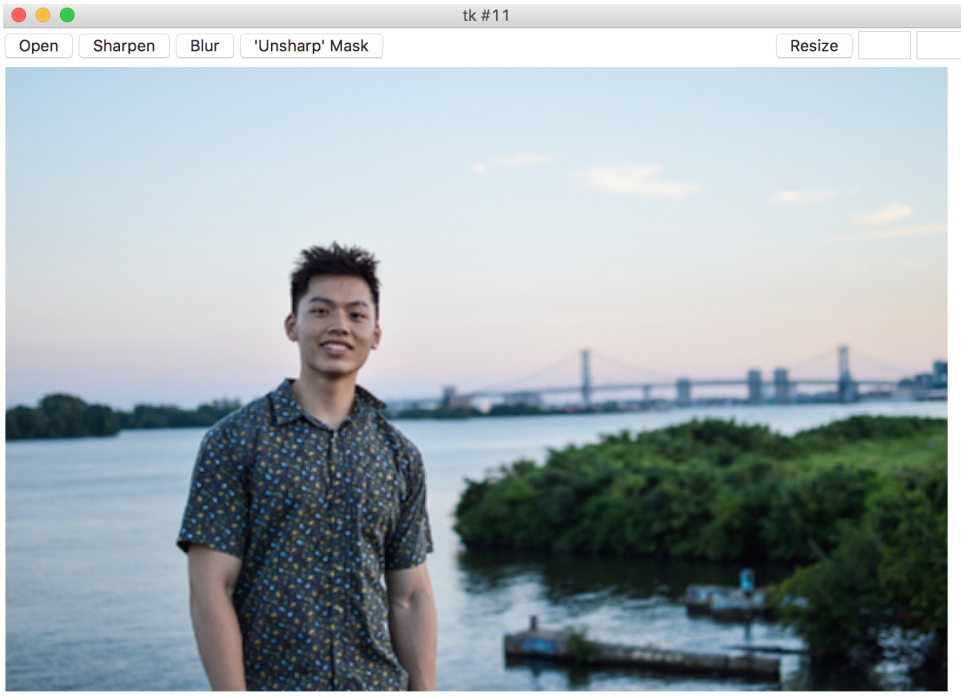
As mentioned, we mostly met what we proposed in our project proposal besides the saving and redoing feature of the user interface. Below is three screenshots demoing our GUI. The first one is the original image uploaded; the second one is the image blurred; the third one is the carved image with 10 rows and 10 columns removed seamlessly. Below we also have the entry boxes for carving inputs.

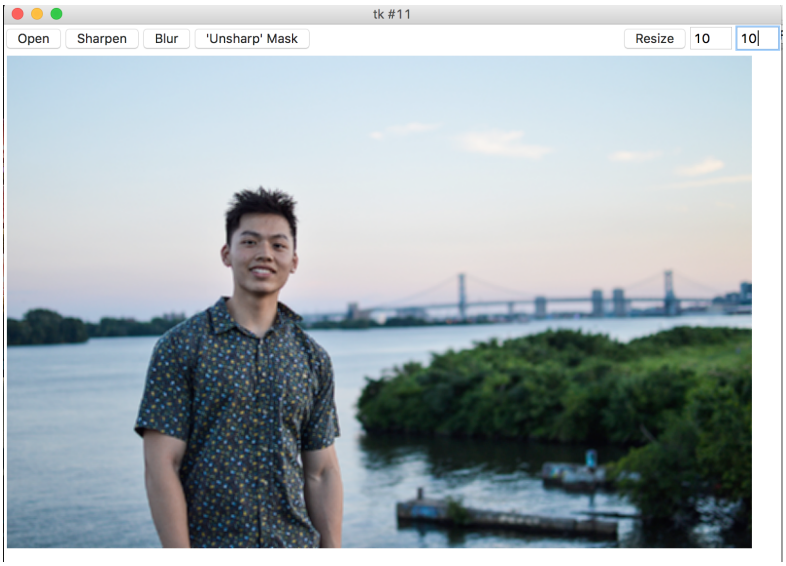


1.


tk #11
Open
Sharpen
Blur
'Unsharp' Mask
Resize

2.


tk #11
Open
Sharpen
Blur
'Unsharp' Mask
Resize

3. 

As can be seen, the user interface is an easy-to-use and self-explanatory tool to play with when making certain edits to images. And our achieved result is largely in line with our expectation.

Contribution from the Team Members:

Frank and Tong worked as a team on this project. Tong consolidated the different filter functions entirely and integrated them with the user interface in a way that the filter functions can be called repeatedly in a feedback loop. Frank consolidated the carve functions entirely and integrated them with the user interface. Tong and Frank both contributed to the implementation of the user interface from the ground up.