

Grace Street

mstreet@rollins.edu

V.Summit

12/ 13/ 17

Final Project: InstaArtist

My project was based on Red Reader, 2010 by Adam Maytak, acrylic and graffiti marker on plywood, which depicts a cartoonish aesthetic of a man reading the newspaper. I was originally drawn to this idea because I adore drawing and I love the look of sketched out cartoony figures. I was also inspired by a photo altering app that I had on a phone in the past that would take images with various artistic filters on them. So I aimed to recreate it in a way with a few touches of my own.



At first the goal was simply to alter the photos and save them. I used many open cv functions and filters in order to find the right aesthetic for creating a pencil sketch filter, a cartoon filter and a black board filter. Open CV already had a pencil sketch filter on hand, so there wasn't much else to do but try to improve the look.(Sketch) Black board is simply the inverse of pencil sketch, all I had to do was bitwise not the pencil sketch function. (Blackboard) However cartoon was a little more difficult as I had to find a way to even out and spread colors out over an area, then provide a thicker out line to them. For this, I combined a bilateral filter with a median blur to reduce noise and even out the colors. I used an edge preserving filter to assist in the detection of images using the adaptiveThresholding method. Then I bitwise anded the edges and blurred image to join them together.(Cartoon)

During this process, I figured it would be a good idea to be able to perform these functions with live video, just like if you were snapping a picture from your phone. So with that I included a video_application method which would open the webcam and show the live video with the filter already applied to it. Since the open cv code was so clean and fast, this made the video filter faster as well. Had I tried to create an effect from scratch using loops, the video would stall(I've tried....didn't go well) I include the option that when the viewer presses 'c' the window will be destroyed and the image will be captured and placed in the directory. (Capture)

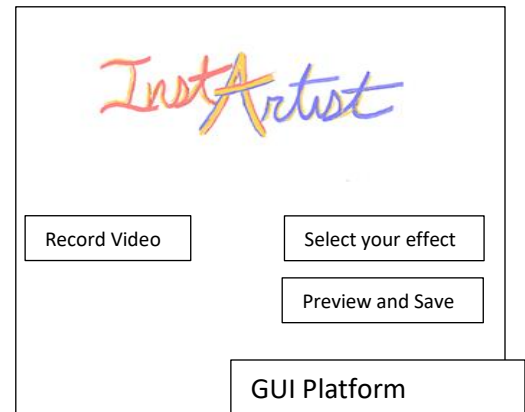
In addition, during this process, I figured, I might as well be able to actually record a video with the effect as well. So I added addition conditions to the video_application method to allow for the writing of videos. This proved to be very challenging as the video writer is very finicky and very dependent on the type of system you are using. But with little persuasion in finding the right file and making sure that it was writing as a grayscale or color when needed. It

was finally compliant, and records the video, then writes it to the same directory after the user presses 'c', it will also capture a still and store it.

The most tricky and time consuming part of this assignment was incorporating a GUI. I decided that I wanted the user to be able to actively choose what they want and not have to go in to write and uncomment code. The GUI it's self is a very sensitive thing, requires a lot of specific attention. The GUI looks like this (GUI platform). On the right, it provides a menu of pre-selected image to choose from, then a menu of effect to select. A preview button which opens windows to see the out put and then writing the image into the directory. On the left holds a button that allows the user to capture from the webcam(Note: The user must select an effect for it to work), like in the last section if the user presses 'c' the image will be captured. There is also a button for recording a video, which does the same; press 'c' and the video will be recorded and stored.

However due to unforeseen circumstances, my GUI stopped working. For some reason, after closing Spyder, the platform just refused to work again, so unfortunately, I have an incomplete GUI to present. However, there were a few technicalities when the program was originally running. For instance, in order to perform a change multiple times you would have to close the GUI and restart it again, otherwise the changes from last time would remain.

To make it seem like a real marketable application I included a logo that I drew myself, (Logo) InstaArtist is the name of my application.



Technical Aspects

Minus the faulty GUI functions, the program at it's most basic level, takes an image and applies a specific filter to it, writes and displays it.

The `apply_cartoon_filter` method takes an image and applies a `cv2.edgePreservingFilter` to it, then it runs through a bilateral filter multiple times in order to have the colors evened out cleanly. Next, it is converted to grayscale and median blurred in order to achieve cleaner edges. Edges are detected and enhanced by `cv2.adaptiveThreshold`. Due to some discrepancies in the code, I must also check and resize the edges picture before then bitwise anding it to the color image, thus joining them.

The `apply_pencil_sketch_filter` method first applies a gaussian blur to the image before calling `cv2.pencilSketch()` which creates the pencil sketch effect for me. In fact, the method actually outputs a colored sketch as well as a black and white sketch. However, for the purposes of this assignment, I only really wanted to use the black and white pencil sketch. But it's there if I ever want to include it.

The `apply_black_board_filter` method was just a matter of theory, in order to achieve a blackboard like aesthetic, wouldn't I just want to reverse the black and white. At first, I attempted to use the pencil sketch method and then use a while loop to go through every image

and turning all that is white to black and vice versa. But the result slowed down the video aspect of my code significantly. After, a little research, the simplest solution was to bit wise not the result of the pencil sketch filter, which makes complete sense, because notting something creates it's inverse.

I also had intended to include an `apply_water_color_filter()` to create a water color aesthetic, I attempted to use the edge preserving filter, followed by a blur but the result looked more like a blemish reducing filter. I had no other ideas for how to create such a filter, so this prospect was pushed to the side.

Had the GUI worked, it would've gone through this process. First it creates the window, then places the InstaArtist logo at the top- center of the window. Then it splits the frame creating a right frame, filled with 2 menu buttons, one for selecting images and one for selecting effects. The menu for selecting images calls the function `image_selection(variable)` which saves the name of an image(the path) as a global string. Then the effect selection menu calls `retrieve_effect(variable, name)`, which takes a number and the name of the image, reads in the image, and then applies the filter to the image based on the number given (1 Cartoon, 2 Sketch, 3 Blackboard, 0 No Effect) it then has the image created. On the right there is also a button Preview and Save, which does as it says given the input from the global variables `in_image` and `filtered`, it writes output image to the current directory and displays the original as well as the altered picture using `imshow()`.

Webcam Capture

Select your image

Next, a left side is created which contains a button to capture an image from the web cam(WebCam Capture), and another that records video(Record Video) from the web cam. When someone selects an effect from the menu on the right, that effect is then applied to the web cam image. When the web cam capture button is pressed, it opens the camera and displays the video with the filter already applied. The user presses 'c' the image will be captured and saved. As for the video capture button the concept is the same, the web camera comes on and is immediately recording, once the user presses 'c' the video will stop and be save to the directory, a still will also be captured.

Image Gallery A : PreSelected Images








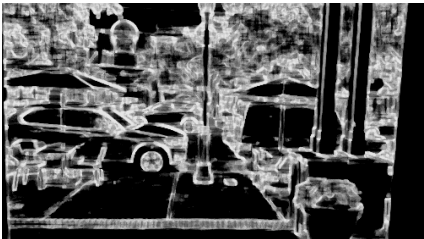
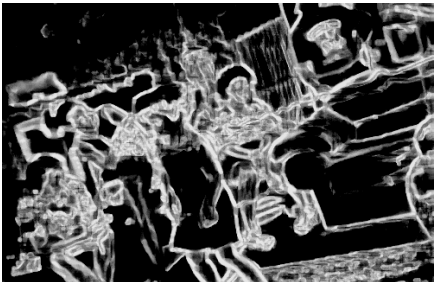
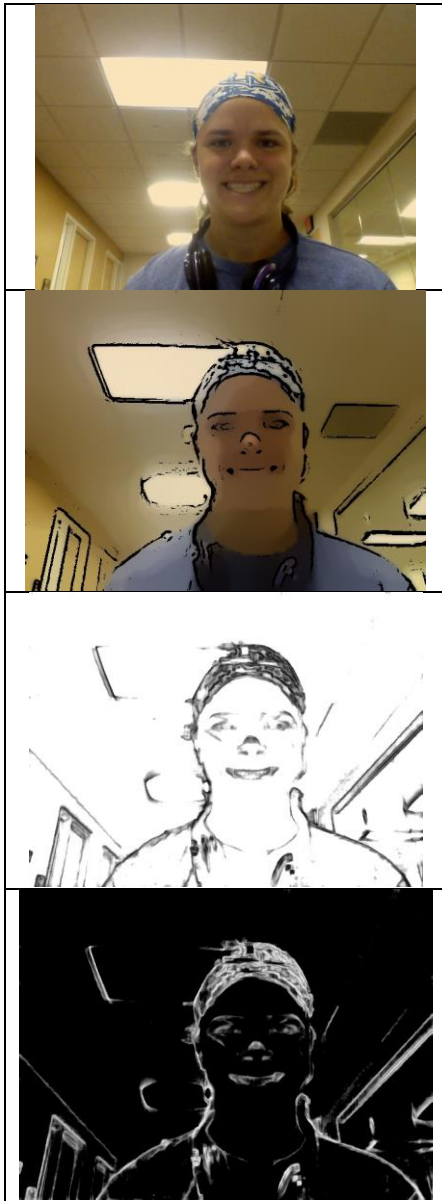
	Bottle	Street Side	Silly String
O r i g i n a l			
C a r t o o n			
S k e t c h			
B l a c k b o a r d			

Image Gallery B: Web Cam



Discussion on Output:

The out put of these methods have several flaws that come with edge detection and contrast. The quality of the pictures is very much so influenced by the lighting, the distance from the camera, the colors of the subject and the background, and even slight shadows throw it off.

Above in Image Gallery A, the bottle is by far the cleanest, with a well-defined subject matter and distinct colors it shows up very cleanly.

By contrast the images of the street side are not quite as defined. There are lots of edges, corners, and a great many subjects. The ones up front seem pretty well defined, but they soon become jumbled together in a bit of a messy haze.

When it comes to humans, this program doesn't do such a good job. There are not a lot of visible corners when it comes to people. Everything has curves and arcs to it which makes detection difficult. The final image above is of me as a child getting shot by silly string, with various relative in the background laughing. The scene is very chaotic and full of noise, which shows in the final processed images, everything is a mess, and hard to make out certain items.

The webcam pictures to the left (Image Gallery B) turned out better than expected but still the quality of the camera has effected it's ability to capture clear subject matter. It has a hard time seeing the curves of my face and jaw. And there are many lines in the background that it is missing completely, but this has more to do with the program itself, not the camera.

Conclusion:

This project was a lot of fun but it took a lot of work and research in order to accomplish this. The hardest part was sorting through all the information that I had, applying the knowledge then debugging; it was a very long rinse and repeat process, that took hours to complete. Even still I felt unsatisfied, mainly because my GUI didn't work, but also because I wish I could've done more with the time that I had. My view of computer science hasn't really changed all that much; from the first day I entered the computer science program, I knew that it was going to be a rough ride. Being a developer means, hours staring at a screen trying to figure out what the hell

is going wrong in the code, hours researching how to tackle these problems. Sometimes it makes you so insane with frustration that you want to set your laptop on fire. But, it is because of these hardships that I find myself enjoying these challenges. I become emotionally invested in these pain-in-the ass programs, because I slaved over them, they're my babies, and I never want to leave them half-done.

However, my view of computational photography has grown. Through this course I have seen how much people have to think about the natural world to create something in the virtual world. There is a lot more math and physics than I thought I'd ever need to know involved with making the photo applications that dominate our modern culture. I've seen these things everyday, but I've never really thought about all the processes behind it. I have also, learned how to tap into and create my own versions of these programs. The knowledge I've gained from this will undoubtedly help me in my future endeavors in both programming and art.

Sources:

https://www.tutorialspoint.com/python/python_gui_programming.htm

<https://www.learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/>

<https://github.com/mbeyeler/opencv-python-blueprints/blob/master/chapter1/filters.py>

<https://www.learnopencv.com/non-photorealistic-rendering-using-opencv-python-c/>

<https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html>

<https://gist.github.com/vscv/f7ef0f688fdf5e888dadfb8440830a3d>

<https://stackoverflow.com/questions/4693120/use-of-global-keyword-in-python>