Beauty Bot

Cara Nix and Olivia Ridge
CSE 5524

Introduction

We sought to use computer vision to create a beauty bot that can be used to change different aspects of an image to enhance its aesthetic appeal to the user. The beauty bot includes the following tools: red eye removal, acne removal, retroactive portrait mode, and lip staining. This idea originally appealed to us due to its simplicity and due to the fact that computer science is a male dominated field. Therefore, most examples of computer vision are video games, virtual reality, robotics, etc. However, there are broader applications for vision and ones that would possibly encourage young women to consider careers in STEM. In the following section we will go through all the features of our beauty bot and discuss the vision techniques used as well as the success of the tool.

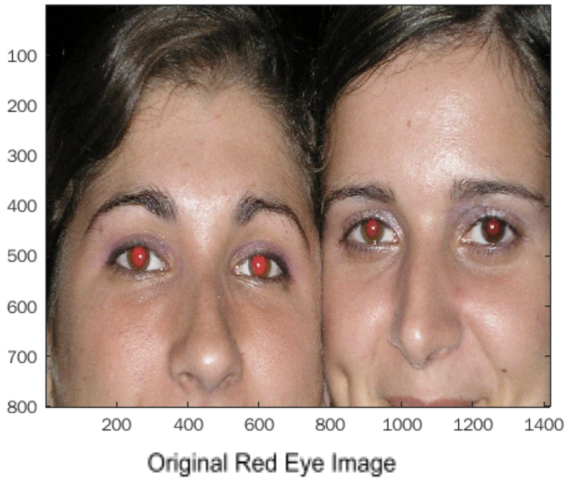
Features

Red Eye Removal - Cara

Most people have taken a photograph of someone with flash and upon inspection the subject has bright red eyes. This red-eye effect can ruin perfectly good images. This effect happens because when the flash goes off one's pupils do not have time to constrict to reduce the amount of light entering the eye (similar to an aperture decreasing in size). This results in a significant amount of light reaching the subject's retina and reflecting back, that reflected light then shows up as red eyes (Rodrigues, 2019) .

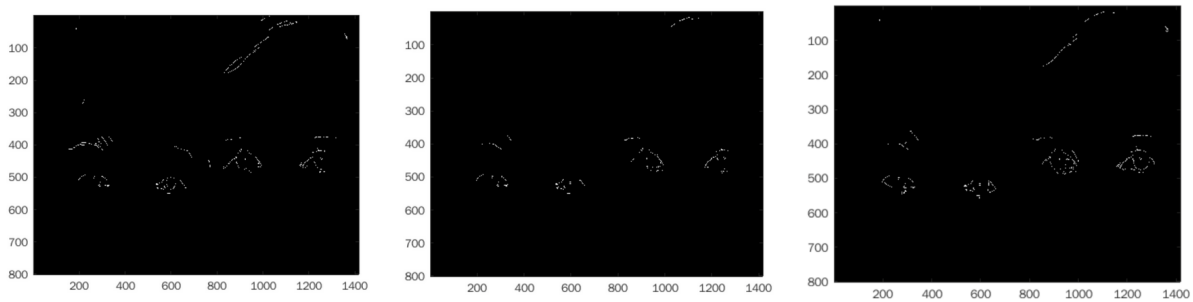
We don't think that this red eye effect should make images unusable. Therefore, one of the features of our beauty bot is isolating and removing the pesky red eyes. In order to do this we used edge detection. We felt edge detection was best suited to the task as it looks for sharp changes in intensity. Similarly, red eyes are a sharp change in intensity, so we hoped implementing edge detection would at least shrink where we are looking for red eyes.

.



Original Red Eye Image

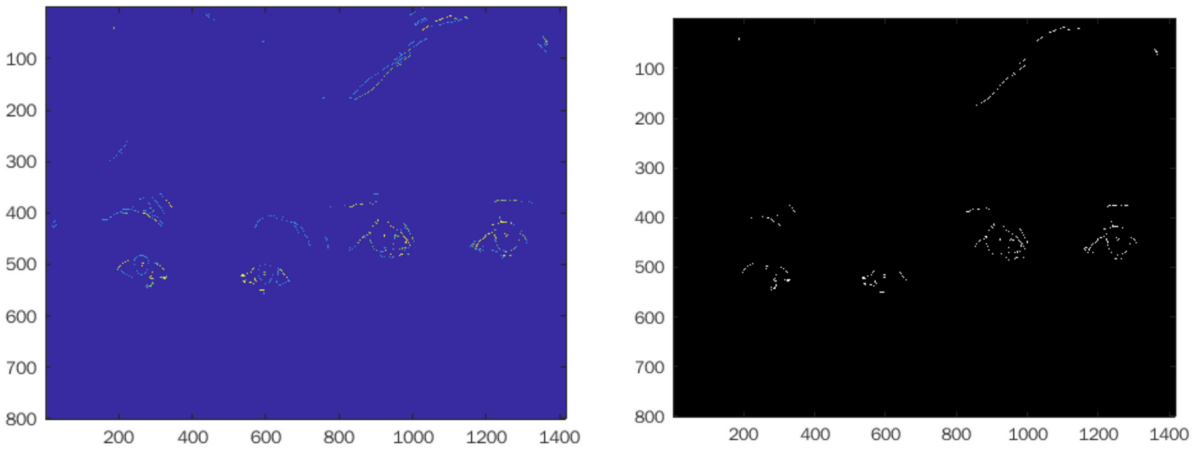
First, we read in the image we sought to remove red eyes from, with different matrices holding the RGB values. Then, Canny Edge Detection was run on each of those three matrices, resulting in the following output:



R G B

The threshold within the edge detection for R used was .66, and for G and B it was .55, these thresholds were found by testing different thresholds to see what edges were identified. Note that this output is not that clean in that it doesn't just isolate the eyes, this is to be expected. As there are multiple places where there is diginficnat shift in intensity, for example, the white skin to the dark brown hair, and similarly for eyebrows. Canny edge detection was chosen as it has the added benefit of smoothing ang using non maximal suppression. However, looking back we could have used alternative edge detection methods that would have been faster computationally (i..e Sobel), though speed wasn't an issue in our case, so Canny was used.

In order to further isolate the edges which were most present in the image, the output of canny for R,G,B were added together and only pixels which contained a 2 or greater were considered to be edges. This was done to ensure rapid change in intensity for at least two colors.



R+ G + B R + G + B >2

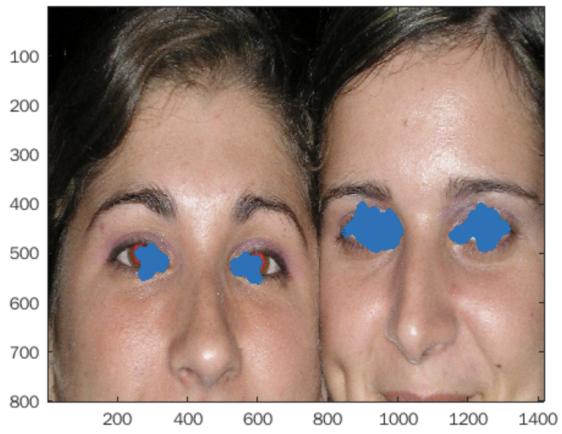
Now that we had reasonable locations to look for the red-eyes, we needed to make the areas found bigger, to fully encompass the eyes in order to search for the red eyes in just that area. This was done via closing (dilation followed by erosion) with a radius of 20 and then connected components

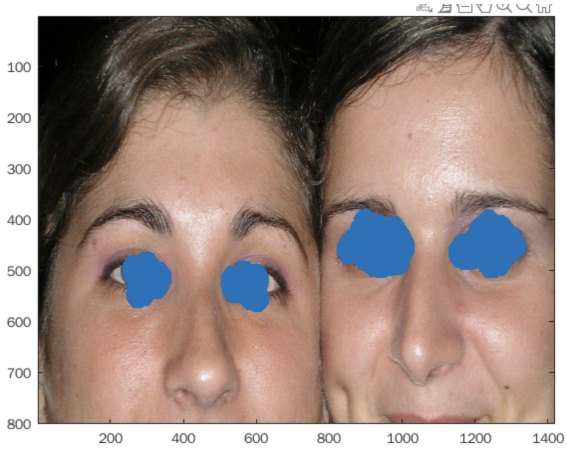
thresholding with a threshold of 600, so only areas which had more than 600 pixels together post-closing would be considered places where the eyes might be. Importantly, these thresholds were all chosen through trial and error and thus may not be extendable to other images or pupil sizes, something future work could consider. The results of the process are below.

Closing (dilation + erosion)

Connected Components
(Threshold 600)

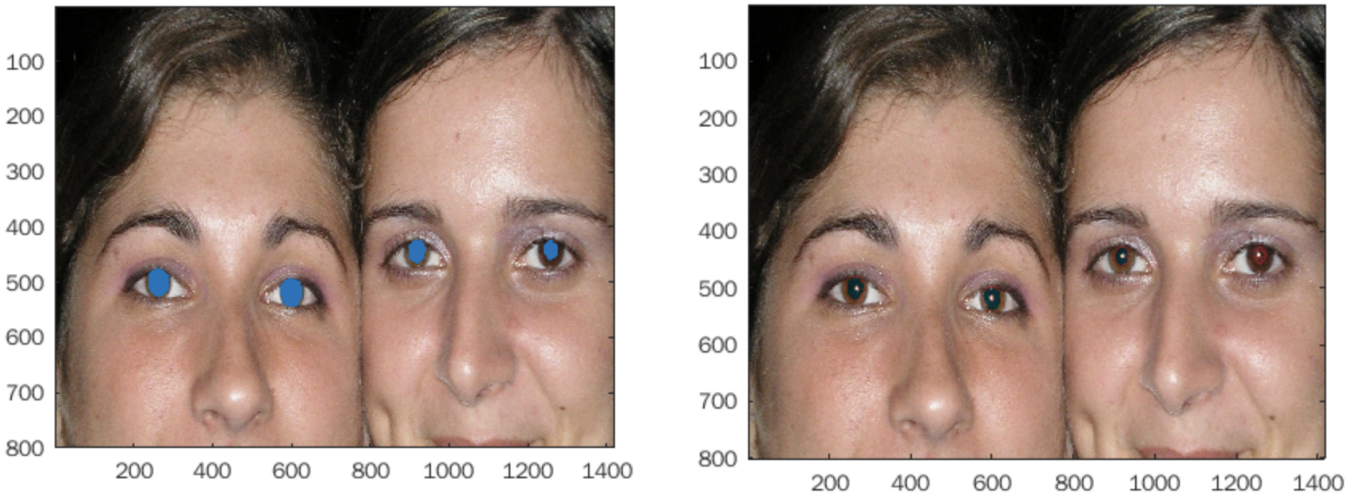
The next issue we ran into was that the areas found did not fully encompass the red eye, so the original idea of searching through the pixels isolated would no longer work. However, there was a simple fix for this: dilation. We ran dilation again and were able to fully cover the red eye effect, as can be seen below (right). The original output (pre-dilation) is also below for comparison (left).





Now, simple thresholding could be applied on the areas isolated where we expect to find the red eye effect. Specifically, values of R>140, G>70 and B>70 were considered, the resulting image demonstrates the impressive isolation of the red-eye effect (left). The red eyes were then “removed” by simply subtracting 120 from the R values which were >140 to make the red value less prominent, the area

was then smoothed via gaussian smoothing to result in the final output image (right). As one can see, the process laid out above does successfully remove red eyes.

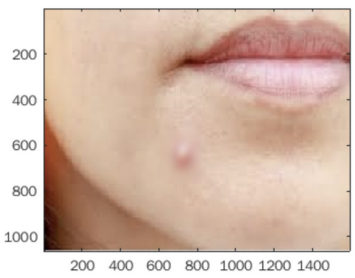


Acne Removal - Cara

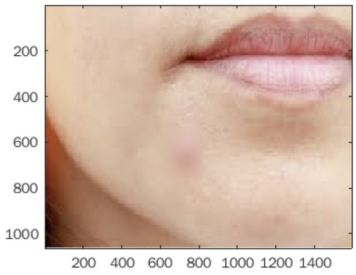
We also implemented a feature which would remove acne from an image. Acne affects 75%- 95% of the adolescent population and acne persists, with 40% to 54% of men and women over 25 still dealing with acne on a regular basis (Lindeberg et al., 2002). Acne also can also have lasting negative impacts on individuals' self-esteem (Gallitano and Berson, 2012). Therefore, the ability to remove acne from photographs is essential in the modern era, where everyone is posting and wants to look and feel their best.

In order to remove acne from images we chose to use template search. This technique requires that a template image be passed in, so a picture of the acne we were trying to remove was inputted and the original image was searched. For each template-sized window NCC was calculated in order to compare the current window to the template. The search took around 16 minutes. After the search was complete the window with the maximum NCC score (indicating the closest match) was extracted. Below, you can see the patch isolated on the original/search image.

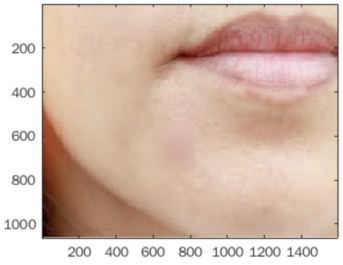
We then blurred that patch with sigma equal to 15, 35, and 50 to visualize different levels of smoothing. One issue we ran into after putting the smoothed window back in the original image was that the smoothed patch looked slightly out of place. In order to fix this, the smoothing window was extended around the patch 50 pixels on each side. This, to us, made the patch fit in a little more, and allowed for more of the natural skin color to be smoothed, rather than the pink/red from the acne. The results are below.



Sigma = 15



Sigma = 35



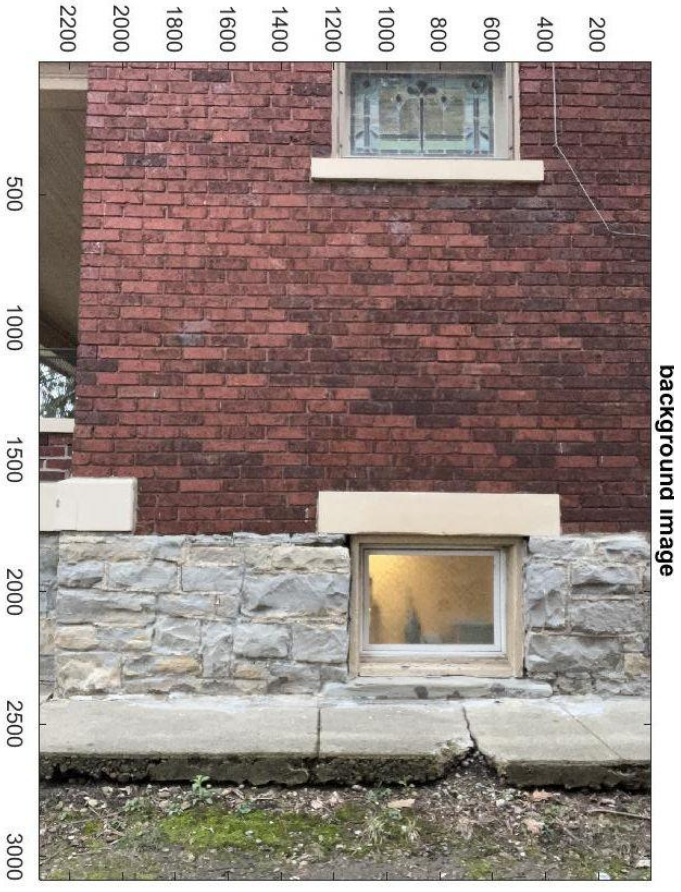
Sigma = 50

As one can see, the results when sigma =50 do remove the acne quite well. Thus the implementation of this feature was a success.

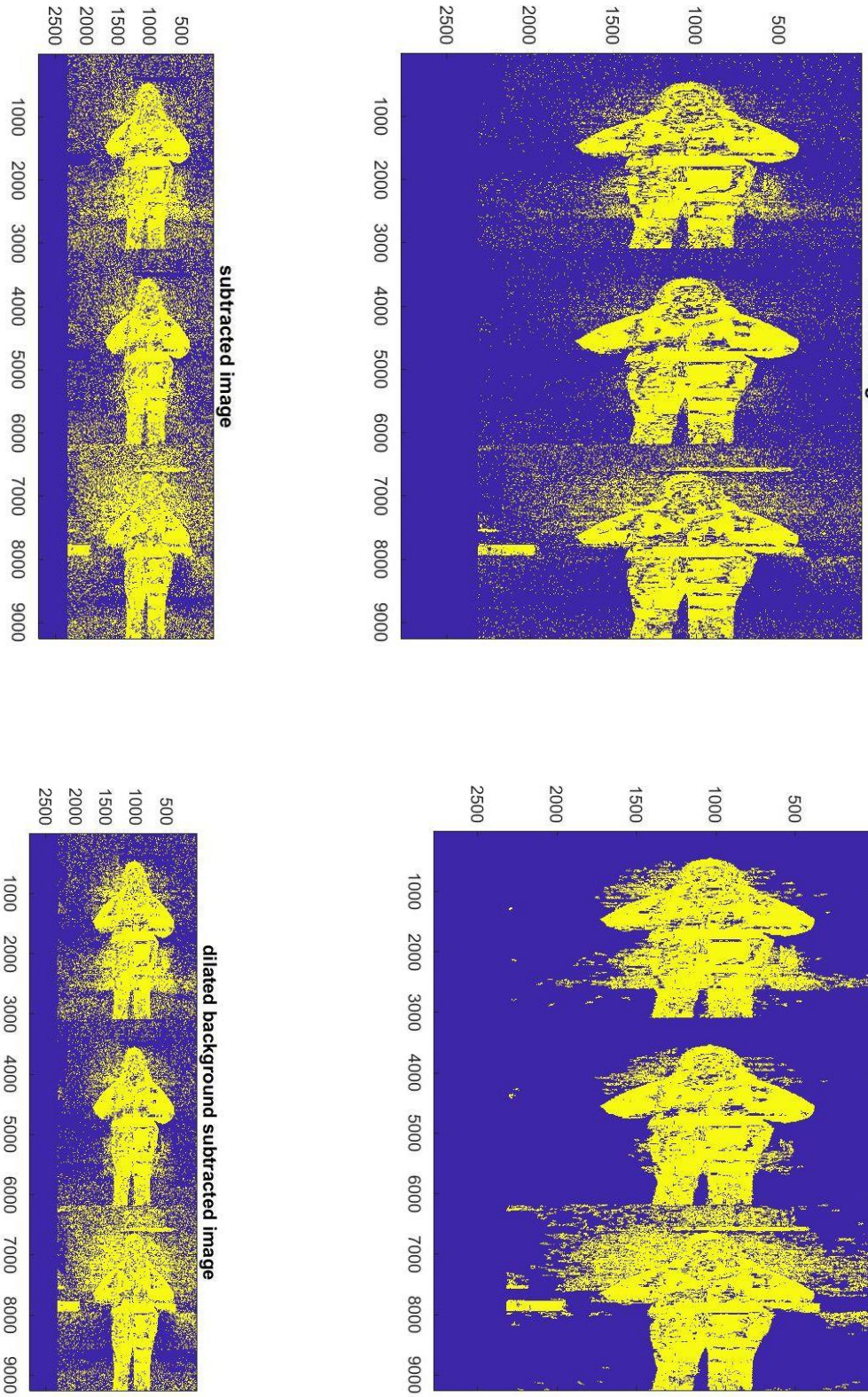
Portrait Mode - Olivia

Another feature we created for the Beauty Bot is a retroactive portrait mode. In portrait mode, the subject is in focus and the background is slightly blurred, and in this version the blurring happens after the photo is taken. On the iPhone, the front facing camera cannot take portrait mode photos due to the way the feature is implemented. Our feature solves this issue by allowing users to use the front camera and requiring that they take two photos, one of only the background they wish to blur and one of them in front of the background.

We implement the feature by first performing background subtraction to isolate the main content of the photo. By using a lower threshold value, such as T=10, we cut out most of the background of the image and keep the features of the subject that we want to keep. Higher T values cut out too much from the subject of the image.



Once we have the subtracted region, we perform erose, dilation, and choose *k* connected components to isolate the subject more accurately.

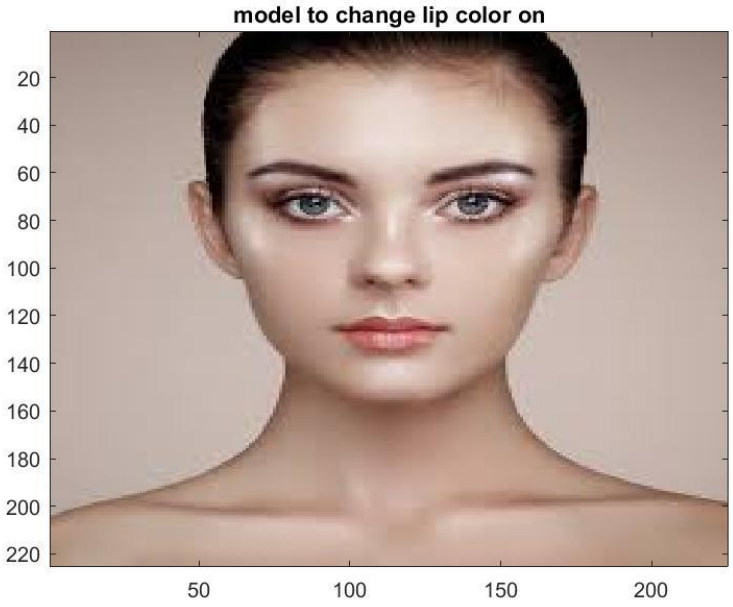


Then, we use gaussian blurring on the background image to create the blur effect. We decided that a sigma value of 15 was best to get the right amount of blur that would be comparable to that used in portrait mode on most cameras.

Lastly, we create the final portrait by looping through the background subtracted (binary) image and overlaying the corresponding main content pixel on top of the blurred background if the corresponding pixel is equal to 1.

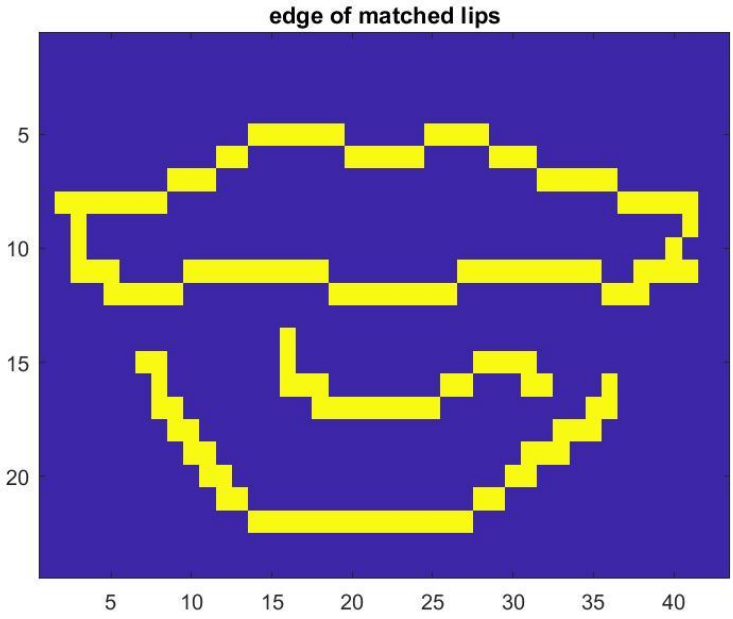
Lip Color - Olivia

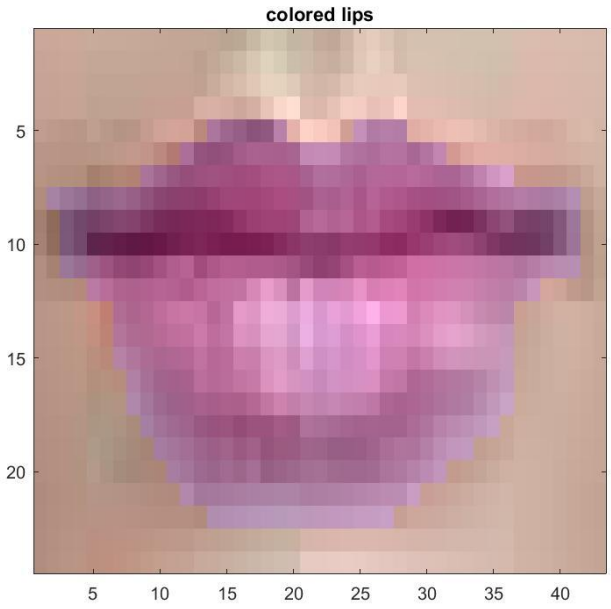
Another feature we implemented was a lip color changer. It can be fun to play around with lip color after taking a picture. You can change it to match an outfit, or try out a color you don't have. To implement this feature we used template matching, using SSD to find the best match. The search image is a picture of the person's face, and the template to find is a cropped window of their face that contains their lips. The idea is that the person would take a photo of their face and then, if they choose to change their lip color, they'd be prompted to draw a box around their lips. We chose to use SSD for our matching equation because it performs well if the area to match has a similar brightness to that of the search image. In our case, they will have the same brightness since they come from the same image. Additionally, we started out by implementing this using NCC, but this made the program take too long as the mean and standard deviation had to be calculated for each color channel in each candidate patch. So in this case, SSD was the better choice.



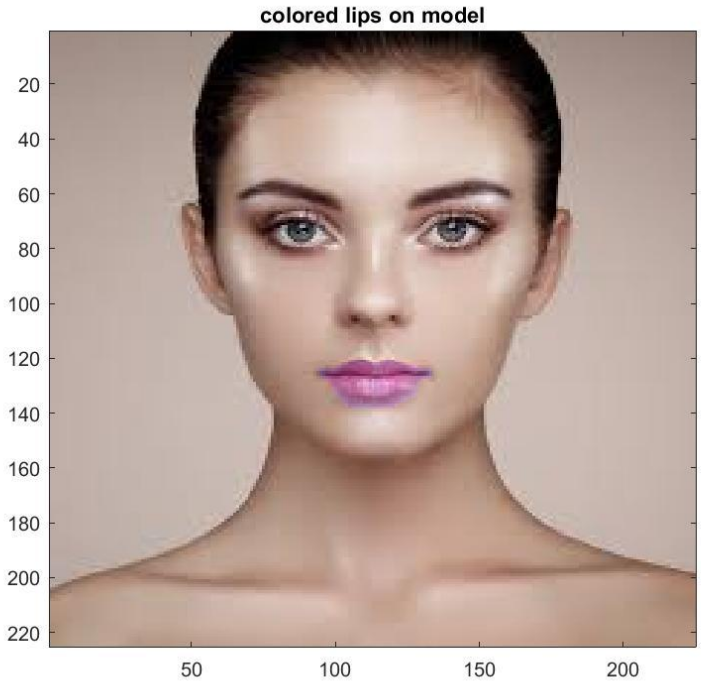
Once we have found the match (lips) in the search image, we take that window and perform edge detection on it using Matlab's canny edge detector. Using the edge boundary, we determine if we are within the lip area or not.

If we are within the lip area, we change the color of the lips by manipulating the color channels of the pixels. By adding or subtracting from their original values, it maintains the depth of the lip color instead of setting all of the pixels to one flat color.





Once we have changed the color of the lips, we can reattach the manipulated window onto the original image of the person's face to obtain the desired output.



Conclusions

Overall, we would consider our project a success. We ran into issues, but were able to overcome them. We were able to implement each feature successfully and tune them to have the best output we could find. For example, when implementing the acne removal feature, we successfully blurred the area but the color was more pink in a more concentrated area than the rest of the skin, so we expanded the area to incorporate more skin color for a more natural look. Another issue we ran into was efficiency and getting the programs to complete within a realistic timeframe so we could run them multiple times and make improvements. In the case of the lip color changing program, we originally started off using NCC as the matching algorithm, but the photos were so big that to perform all the necessary calculations at each individual patch was not allowing the program to finish. So, we resolved the issue by instead using SSD because it was well suited to the task and sped up the program dramatically. There are, of course, still improvements we can make but we were able to work through all the issues that we encountered. If we were to extend this project, we'd consider adding a sharpness feature, where we'd add noise in the image to increase how "sharp" it appears, a common editing feature in most editing applications. We could also attempt to change hair color and eye color. So, rather than just removing red eyes, be able to change eye color altogether. Similarly for hair we could find and change hair color. We also could add an image stitching feature, which would combine two images and make one extra wide image (in case the wide lens wasn't quite wide enough to fit everyone in the frame). We could also implement scale invariant template matching to try to detect features more automatically and generically, rather than having the user isolate the area to match themselves. It would also be a good idea to test our program out on many different inputs, such as varying levels of brightness or different skin tones. Additionally, we could create a graphical user interface (GUI) where the user could use their finger to isolate the areas to smooth, change the color, similar to beauty editing apps. One downside of this is it takes away the "vision" aspect of what we've done previously as now it is the user identifying locations.

For Red Eye Removal, if we were to do it differently we'd want to be more sensitive to changes in red around the pupil. We'd also want to test our implementation to see if it is generalizable to other red eye images. For Acne Removal, we might consider implementing a different search technique, as it did take around 16 minutes using template matching to find the correct match, potentially first isolating areas with sharp changes in red color (similar to red eye removal) and then going and looking in those areas for a matching widow, thus limiting the search window. Another way we could improve this tool in the window located by NCC would be to subtract some from the red color channel of the image, in order to reduce the appearance of the pink/redness post-blurring. For the portrait mode feature, we could play around more with the different parameters, such as the background subtraction threshold value or number of connected components, to give the most precise image. We could also implement background subtraction using statistical distances instead of absolute difference to provide a more accurate output, though this would require many more input photos from the user. Finally, for the lip color changer, we could implement a feature better suited to finding the lip region, such as utilizing clustering instead of edge detection. A clustering method may be better at getting the entire lip region, as the lips will typically all be a similar color and different enough from the skin color.

Sources

Rodrigues, Aimee. “5 Tips for Preventing Red Eyes in Photos.” *All About Vision*, All About Vision, 5 Jan. 2019, https://www.allaboutvision.com/resources/red-eye-photo.htm.

Cordain L, Lindeberg S, Hurtado M, Hill K, Eaton SB, Brand-Miller J. Acne Vulgaris: A Disease of Western Civilization. *Arch Dermatol.* 2002;138(12):1584–1590. doi:10.1001/archderm.138.12.1584

Gallitano, S M, and D S Berson. “How Acne Bumps Cause the Blues: The Influence of Acne Vulgaris on Self-Esteem.” *International journal of women's dermatology* vol. 4,1 12-17. 6 Dec. 2017, doi:10.1016/j.ijwd.2017.10.004