Chroma Keying

CS 498 CAPSTONE

By Christopher Pagano

Christopher Pagano

Prof. Chiang

CS 498

6 May 2020

Purpose of project

The purpose of this project is to achieve the effect of chroma keying without the use of a physical green screen. Many different forms of media have used the method of chroma keying to put in something that is not really there into the scene that isn't there in real life. Some of these different forms of media includes newscasting, movies and video games. All of these forms of media use some form of either a physical green or blue screen. Using OpenCV to achieve this goal due to its ability of image processing function of both 2D and 3D images and its background subtraction methods. OpenCV is an Open Source Computer Vision Library which was released in June of 2000.

Introduction

What is chroma keying

Chroma Keying is a technique used in media production take out a certain color in the background of an image or video feed. Many production companies in media use green screens and blue screens to achieve this technique. This technique has been used in many forms of media such as newscasting, movies and video games. This technique is also called color keying, color separation overlay or by the specific color that is used such as green screen or blue screen. The

colors of green and blue are the most common colors used in chroma keying techniques due to the fact they differ the most from human skin colors, but any distinct colors can be used for chroma keying. As movies are becoming more expensive to produce, they move more away from practical effects and move to CGI (Computer Generated Imagery) they rely on green screens to achieve the effects of the movies.

What is OpenCV

OpenCV is an Open Source Computer Vision Library which was released in June of 2000. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. To understand OpenCV we must first understand computer visions. Computer vision is a field of computer science that works to achieve the process that computers process images the same way a human does It is a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware. “OpenCV has the ability to use so that it can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality” (https://opencv.org/about/).

Why OpenCV

OpenCV was chosen to fulfill the need of this project because of its image processing function of both 2D and 3D images and its background subtraction methods. Some of these different methods are MOG2, MOG, and GMG. For this project, the background subtractor that will be used is MOG2 due to its ability that was found in the paper *A Comparison between Background Modelling Methods for Vehicle Segmentation in Highway Traffic Videos* by L. A. Marcomini and A. L. Cunha to be more accurate than GMG and MOG background subtraction. OpenCV offers an accessible open source computer vision library to its users which makes it the ideal choice to be used in the project.

Background Subtraction- the technique used in this project extracts the foreground from the background. “Background subtraction is a way of eliminating the background from image. To achieve this, the moving foreground is extracted from the static background. Background Subtraction has several use cases in everyday life, it has been used for object segmentation, security enhancement, pedestrian tracking, counting the number of visitors, number of vehicles in traffic. It is able to learn and identify the foreground mask” (Gupta, https://www.geeksforgeeks.org/background-subtraction-opencv/). There are three different methods that OpenCV provides for background subtraction such as MOG2, MOG, and GMG.

MOG2- “is a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. One important feature of this algorithm is that it selects the appropriate number of gaussian distribution for each pixel. It provides better adaptability to varying scenes due illumination changes etc.” (Marcomini & Cunha, pg. 4).

MOG- “is a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. It uses a method to model each background pixel by a mixture of Gaussian distributions. The weights of the mixture represent the time proportions that those colors stay in the scene. The probable background colors are the ones which stay longer and more static” (Marcomini & Cunha, pg. 3).

GMG- “This algorithm combines statistical background image estimation and per-pixel Bayesian segmentation. It uses first few frames for background modelling. It employs probabilistic foreground segmentation algorithm that identifies possible foreground objects using Bayesian inference. The estimates are adaptive; newer observations are more heavily weighted than old observations to accommodate variable illumination. Several morphological filtering operations like closing and opening are done to remove unwanted noise. You will get a black window during first few frames” (Marcomini & Cunha, pg. 3).

What is Anaconda

Anaconda is a free and open source python environment that is typically used for data science, machine learning applications, and large-scale data processing. Anaconda aims to simplify the management of packages and the deployment of said packages. Anaconda is perfect to use for this project due to run the packages for OpenCV in a safe environment.

The Code

```
#Webcam
cap = cv2.VideoCapture(0)
#Creating mask for Background Subtraction
fgbg = cv2.createBackgroundSubtractorMOG2(150, 12, False)
(grabbed, frame) = cap.read()
snapshot = np.zeros(frame.shape, dtype=np.uint8)
#Backgrounds
background = cv2.imread('C:/Users/chris/Desktop/capstone/grand.png')
background2 = cv2.imread('C:/Users/chris/Desktop/capstone/pg.png',0)
#Resizing of background
crop_background = background[0:640, 0:480]
crop_background = cv2.resize(crop_background,(640,480))

while(1):
    ret, frame = cap.read()
    #Applying the mask to the webcam capture
    fgmask = fgbg.apply(frame,1.0)
    fgmask2 = fgbg.apply(snapshot)
    #Displays video feed
    cv2.imshow('frame',frame)
    #Mask
    cv2.imshow('fgmask',fgmask)
    replaced_image = cv2.bitwise_and(frame,frame,mask = fgmask)
    cv2.imshow('video', replaced_image)
    replaced_image3 = cv2.bitwise_and(frame,frame,mask = fgmask2)
    cv2.imshow('video3', replaced_image3)
    #Putting in the green screen element
    replaced_image[np.where((replaced_image==[0,0,0]).all(axis=2))] = [0,255,0]
    #Display the green screen element
    cv2.imshow('video4', replaced_image)
    #Singling out the green
    lower_green = np.array([0,254,0])
    upper_green = np.array([0,255,0])
    mask = cv2.inRange(replaced_image, lower_green, upper_green)
    #Replacing the green
    replaced_image3 = cv2.bitwise_and(crop_background,crop_background,mask = mask)
    replaced_image = replaced_image3 + replaced_image
    #Final Display
    cv2.imshow('video5', replaced_image)

```

The image above shows how I implemented OpenCV to achieve the purpose of the project.

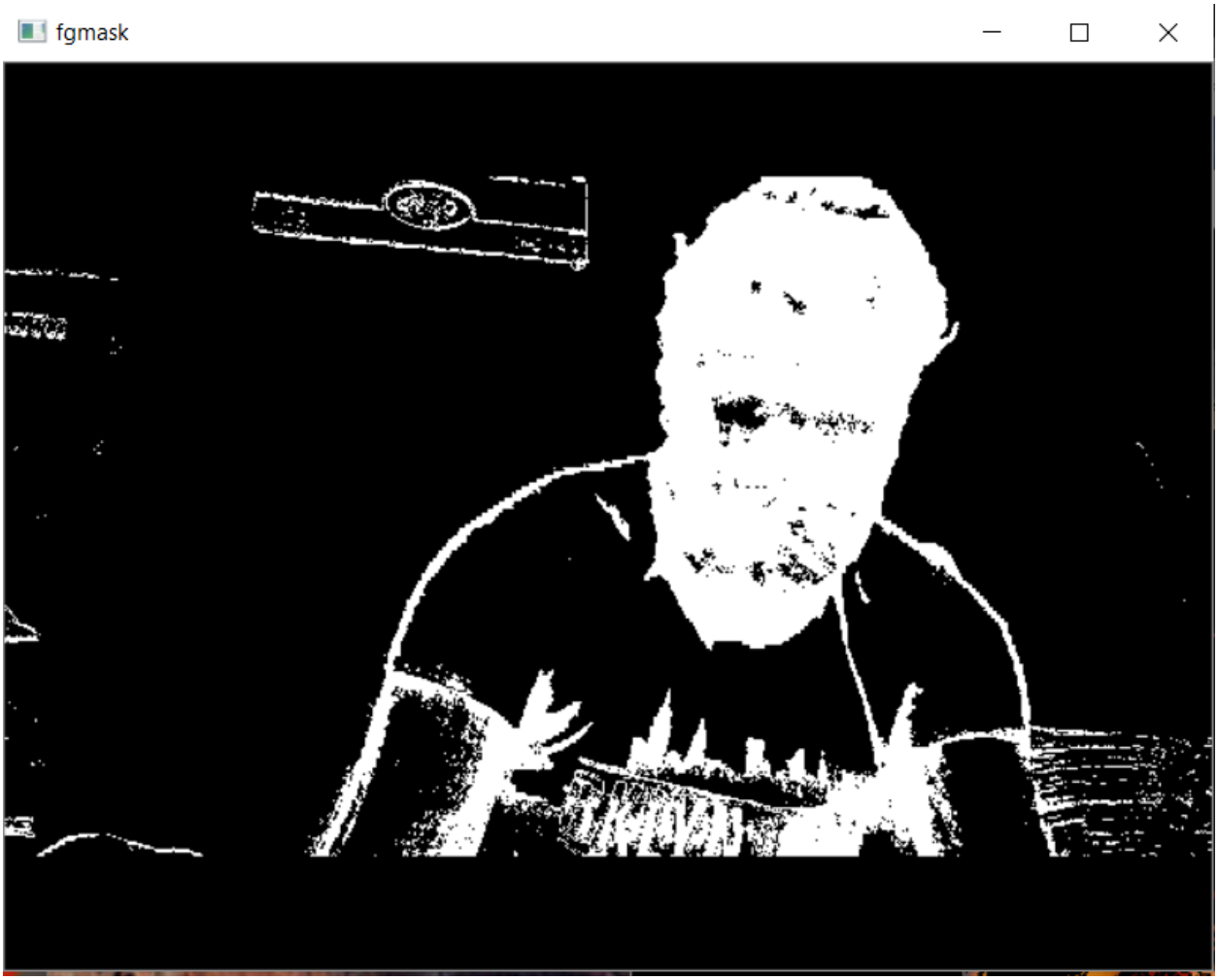
A more in-depth look

```
#Webcam
cap = cv2.VideoCapture(0)
#Creating mask for Background Subtraction
fgbg = cv2.createBackgroundSubtractorMOG2(150, 12, False)
```

This snippet of code shows the creation of the background subtraction method. The reason that the MOG2 method was chosen rather than GMG or MOG methods is due to it having a better accuracy rate, a better precision rate and a lower processing time. The MOG2 method has three parameters, the first one is the history parameter functionality is responsible for the number of frames the method will use to accumulate weights on the model, throughout the entire processing period. The second parameter is varThreshold and it correlates the value of the weight of the pixels on the current frame with values on the model. The third parameter is detectShadows which as the name states enables or disables shadow detection.

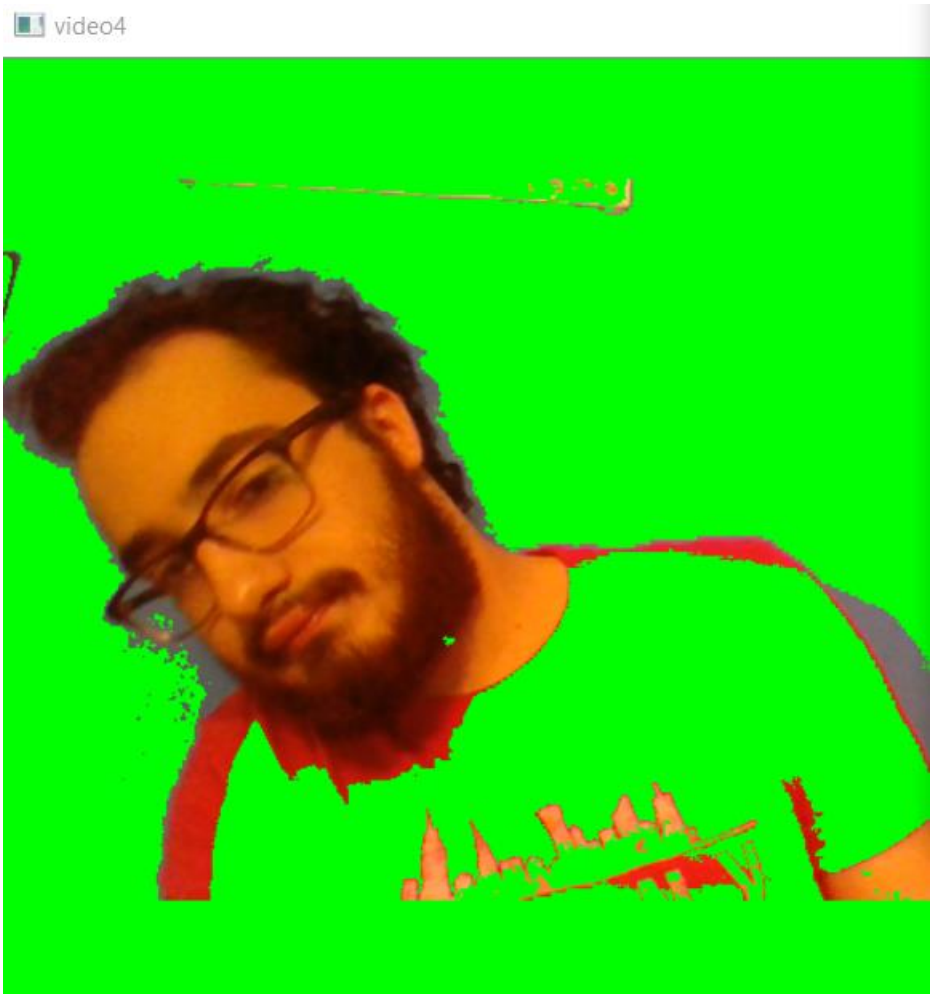
```
#Mask
cv2.imshow('fgmask',fgmask)
replaced_image = cv2.bitwise_and(frame,frame,mask = fgmask)
cv2.imshow('video', replaced_image)
replaced_image3 = cv2.bitwise_and(frame,frame,mask = fgmask2)
cv2.imshow('video3', replaced_image3)
```

This snippet of code performs the task of applying the mask from the MOG2 background subtraction method to the footage from the webcam capture and display the footage with a mask layered on top of the frames to the user. Which leaves us with the image down below.




```
#Putting in the green screen element
replaced_image[np.where((replaced_image==[0,0,0]).all(axis=2))] = [0,255,0]
```

This snippet of code replaces the part of the mask that isn't part of the foreground of the with a green hue that will act as the green screen so that an external image can be inserted into the background. If you look at the image above you will notice how a majority of the face is whited out, that part of the webcam footage will be left alone from this part of the code but the black part will be changed out for green pixels.

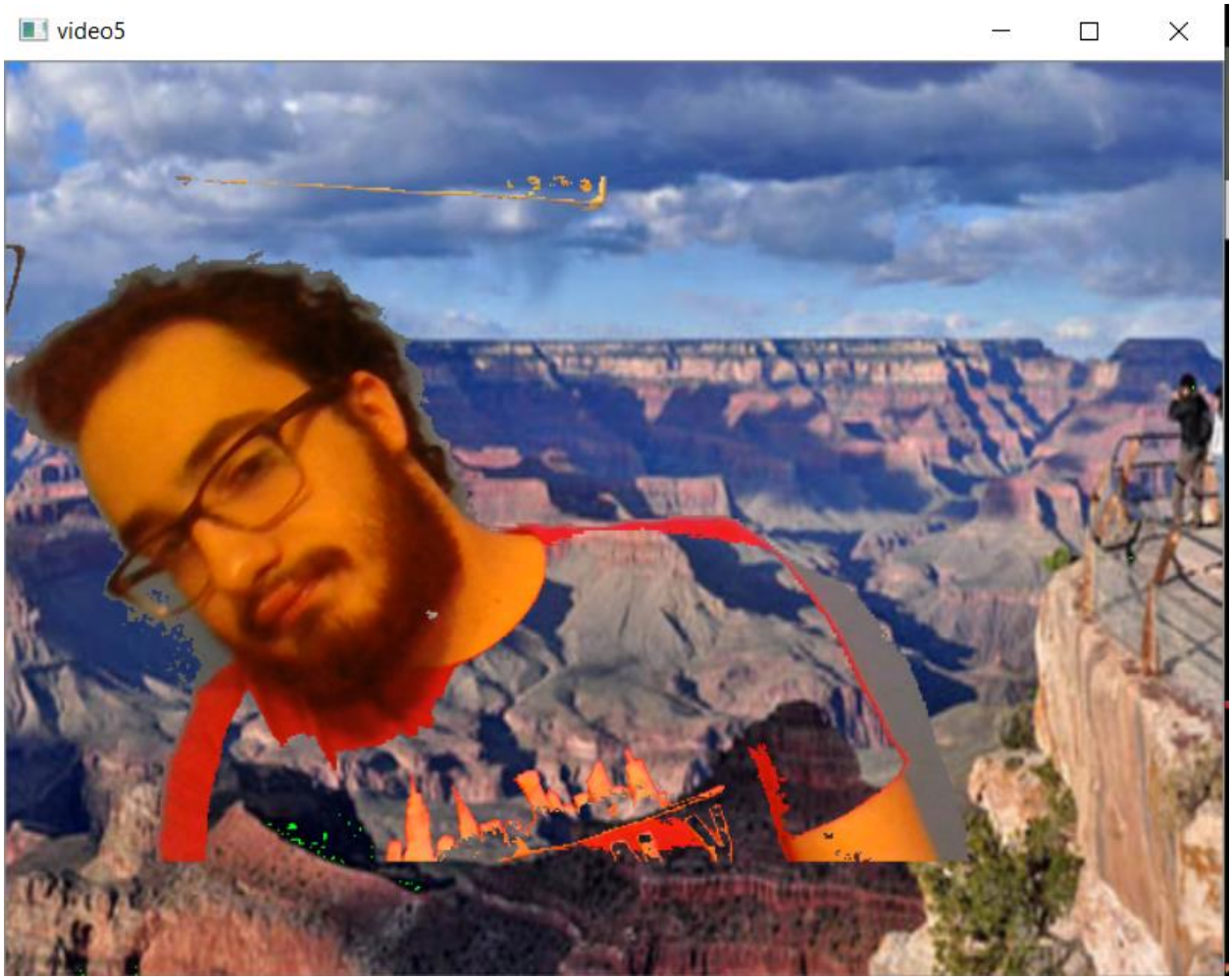



```
#Singling out the green
lower_green = np.array([0,254,0])
upper_green = np.array([0,255,0])
mask = cv2.inRange(replaced_image, lower_green, upper_green)
```

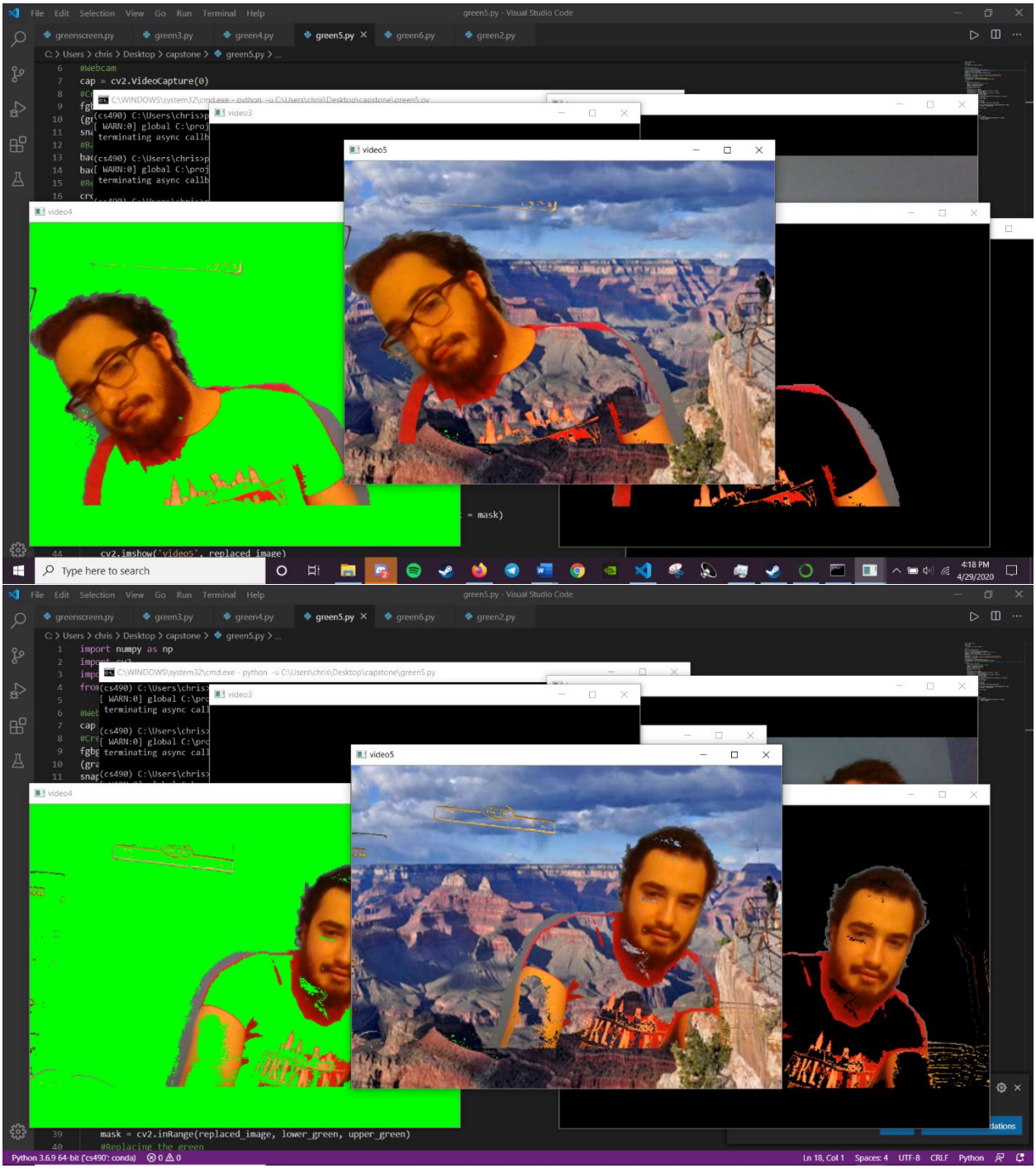
This snippet of code performs the task of looking through the green pixels of the webcam footage that has been masked over and converted to have a green pixel. This will give the ability to pick out the green pixels and replace them with pixels from the external image that is chosen. If this line of code finds a green pixel it will store it to the value mask and will be later used to be replaced by a pixel of the external image.

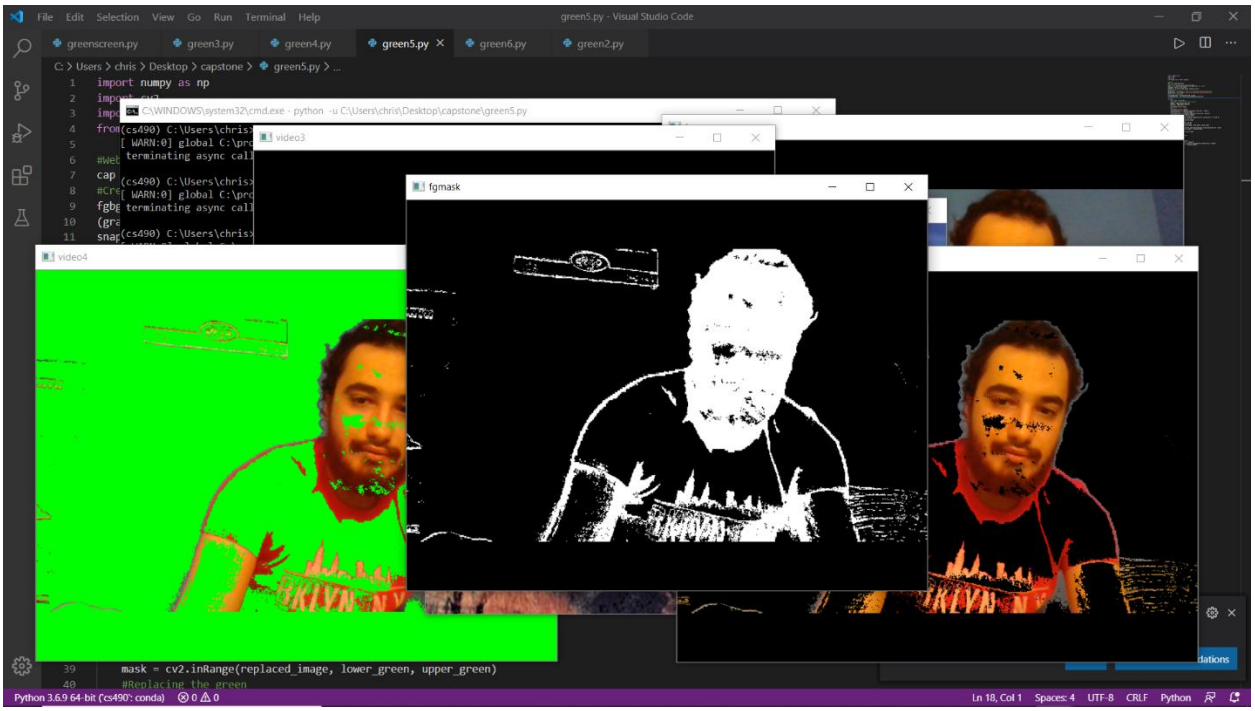
```
#Replacing the green
replaced_image3 = cv2.bitwise_and(crop_background,crop_background,mask = mask)
replaced_image = replaced_image3 + replaced_image
#Final Display
cv2.imshow('video5', replaced_image)
```

This snippet of code performs the actual replacing of the green pixels with the external image file. The replaced_image3 = cv2.bitwise_and(crop_background,crop_background,mask = mask) takes the external image and puts in in pixel by pixel where the green pixels of the mask is. The replaced_image = replaced_image3 + replaced_image takes the green pixels that have been replaced and puts it on top of the webcam footage which gives it a green screen effect and allows the user to see their webcam footage overlaid with the mask from the MOG2 back ground subtractor with the external image.



The Results


green5.py - Visual Studio Code
video3
video5
video4
Type here to search
4:18 PM
4/29/2020
green5.py - Visual Studio Code
video3
video5
video4
Python 3.6.9 64-bit ('cs490': conda)
Ln 18, Col 1
Spaces: 4
UTF-8
CRLF
Python



As you can see from the images above the effect was achieved but not to the desired amount that was hoped for. Sometimes the MOG2 background subtraction method leaves gaps as you can see in the face and it also masked out a good portion of the body. It seems that the closer the subjects face is to the center of the screen; the program would have a difficult time detecting if it was a part of the background or the foreground. Also, another issue that was ran into was that of the lighting. The lighting would affect how well the background subtraction would work. In some situations, a low lighting setting would perform significantly better than a high lighting setting and, in some situations, a high lighting setting would perform significantly better than a low lighting setting.

Future Research

Some future research that would be interesting to take part in is to try different background subtraction methods to see how well they take out the background. As you can see

with the MOG2 method, it sometimes keys out pieces of the face and most of the body and shirt. It would also be interesting to test in conditions with consistent lighting. It would also be interesting to see what other methods besides background subtraction there are to perform the task of this project. Another think that would be interesting to try and find out is if instead of using a still image as the external image to replace the green pixels it would be a video to replace the green pixels from the mask.

Work Cited

About. (n.d.). Retrieved from https://opencv.org/about/

Gupta, R. (2020, February 19). Background subtraction - OpenCV. Retrieved from https://www.geeksforgeeks.org/background-subtraction-opencv/

Individual Edition. (n.d.). Retrieved from https://www.anaconda.com/products/individual

Marcomini, L. A., & Cunha, A. L. (n.d.). A Comparison between Background Modelling Methods for Vehicle Segmentation in Highway Traffic Videos. Retrieved May 4, 2020, from https://arxiv.org/ftp/arxiv/papers/1810/1810.02835.pdf

What is Computer Vision? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/32309/computer-vision

Yeager, C. (2020, February 18). Everything You Need to Know About Chroma Key and Green Screen Footage. Retrieved May 5, 2020, from https://www.premiumbeat.com/blog/chroma-key-green-screen-guide/