Computer Vision Assignment0

Computer Vision Basics and Chroma Keying

By- Apoorva Srivastava (2019702014)

Task1. Video ↔ Images: Write a program to convert a given video to its constituent images. Your output should be in a specified folder. Write another program that will merge a set of images in a folder into a single video. You should be able to control the frame rate in the video that is created.

1.1)Code for converting video to images:

```
import cv2
import os
i=0
try:

        # creating a folder named frame_captured
        if not os.path.exists('folder'):
                os.makedirs('folder')


except OSError:                                         # if folder not created
        print ('Error: Creating directory of data')
video=cv2.VideoCapture("C:\\Users\\Asus\\Videos\\Tom_Jerry.mp4");# Creating the object Video for capturing video
# from the first webcam,the same function is used for capturing video already stored as a file by specifying path

if video.isOpened():      # to check if the webcam is found connected to the computer
        ret,frame=video.read() #command to capture video

else:
        print("Error Loading Video")
        ret=False

while(ret==True):
    ret,frame=video.read()
    if ret==False:
            break
    img = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)# The function is for changing colorspace of an image.
                                               # frame is the input image and cv2.COLOR_BGR2RGB is flag telling
                                               # type of conversion
    i+=1
    cv2.imwrite('./folder/img'+str(i)+'.jpg',frame)


    if ret==False:
            break
    #if i==1000:     #if 1000 frames retrieved
        #ret=False
video.release()
```

Link to the Video: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/EcHyuMVD3JRGsXmTbQrTRLwBcG5MyNxKE8a5VHmYqZ9EPg?e=2ncRVv

Link to the Image Frame Folder: https://iiitaphyd-my.sharepoint.com/:f:/g/personal/apoorva_srivastava_research_iiit_ac_in/EjUGCkCnfGdGoHH92ZcylisBBi-XOR7_5kALn1nmyNQJKw?e=1hWnFP

1.2)Code for converting a folder of images into video:

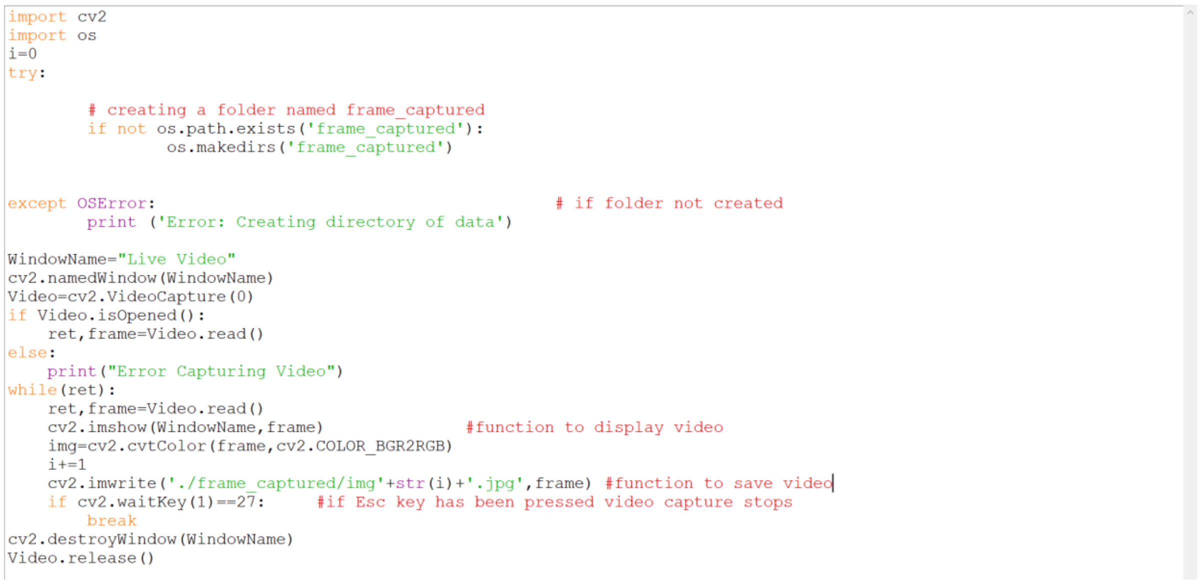
```
import cv2
import os
from time import sleep

def Image2Video(Output, fps,size,image_array):          #function to create video from image

    video = cv2.VideoWriter(Output,cv2.VideoWriter_fourcc(*'MP4V'), fps, size)
    for i in range(len(image_array)):
        # writing to a image array
        video.write(image_array[i])
    video.release()

if __name__=="__main__":
    image_array=[]
    Input= './/folder//img'
    Output = './/folder//video.mp4'
    fps = 25

    for i in range(0,1000,1):     #Creating array of images to be converted into video
        filename= Input +str(i)+'.jpg'
        img=cv2.imread(filename)
        #print("loading"+filename)
        image_array.append(img)
    height,width,layers=img.shape
    size=(width,height)
    Image2Video(Output, fps,size,image_array)
```

Link to the Image Frame Folder: https://iiitaphyd-my.sharepoint.com/:f:/g/personal/apoorva_srivastava_research_iiit_ac_in/EjUGCkCnfGdGoHH92ZcylisBBi-XOR7_5kALn1nmyNQJKw?e=74Fd3g

Link to the Video: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/EcHyuMVD3JRGsXmTbQrTRLwBcG5MyNxKE8a5VHmYqZ9EPg?e=2ncRVv

Task2. Capturing Images: Learn how to capture frames from a webcam connected to your computer and save them as images in a folder. You may use either the built-in camera of your laptop or an external one connected through USB. You should also be able to display the frames (the video) on the screen while capturing.

2) Code for Capturing Video from webcam and storing it in folder also displaying it while capturing:



```
import cv2
import os
i=0
try:

        # creating a folder named frame_captured
        if not os.path.exists('frame_captured'):
                os.makedirs('frame_captured')


except OSError:                                         # if folder not created
        print ('Error: Creating directory of data')

WindowName="Live Video"
cv2.namedWindow(WindowName)
Video=cv2.VideoCapture(0)
if Video.isOpened():
    ret,frame=Video.read()
else:
    print("Error Capturing Video")
while(ret):
    ret,frame=Video.read()
    cv2.imshow(WindowName,frame)              #function to display video
    img=cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
    i+=1
    cv2.imwrite('./frame_captured/img'+str(i)+'.jpg',frame) #function to save video
    if cv2.waitKey(1)==27:     #if Esc key has been pressed video capture stops
        break
cv2.destroyWindow(WindowName)
Video.release()
```

Link to the Image Frame Folder: https://iiitaphyd-my.sharepoint.com/:f:/g/personal/apoorva_srivastava_research_iiit_ac_in/EjuENmJhGVJFhuiYsVXcpnEB7HsRtPYBLtrUM_BX-yuf5A?e=paaD25

Task3. Chroma Keying: Read about the technique of chroma keying. Following are a

few good starting points:

• Introduction: http://en.wikipedia.org/wiki/Chroma key

• Alvy Ray Smith and James F Blinn, ”Blue Screen Matting”, SIGGRAPH’96.

Create an interesting composite of two videos using this technique, possibly with

one video including yourselves.

3.1) What is Chroma Keying:

Chroma key compositing, or chroma keying, is a visual effects/post-production technique for compositing (layering) two images or video streams together based on color hues (chroma range).Here I am using Video shot against green screen to perform the Chroma Keying. The technique has been used in many fields to remove a background from the subject of a photo or video – particularly the news-casting, motion picture, and video game industries.

Chroma Keying is a type of mating problem of separating non-rectangular foreground from a rectangular background for allowing a substitution of different background. There are several approaches to solve a matting problem but Chroma keying uses the technique of using a constant backing(background) color.

Matte: Classically, matte is used as an opaque stripped mask to a transparent color film so that when light is projected on it, the light passes only through the transparent region of the matte and rest is blocked. A holdout matte is the complement of a matte having opaque region for the area of interest.

Alpha Channel: It is Digital Equivalent of holdout matte. It is a grayscale channel having full pixel value for the region to be seen and 0 pixel value for the region not to be seen. The values lie between 0 to 1 representing the transparency of the particular pixel.

Formal Presentation of Matting Problem:

Any image can be considered as $f(C_f, C_b)=\alpha * C_f + (1-\alpha)*C_b$;

where C_f=Colour vector of foreground object , C_b= Colour vector of background image, α=Value of the transparency

In other words, an image can be understood as the combination of foreground and the background image in proportion to the α i.e. transparency.

In Case of an Image shot against a constant colour background: C_b is known and is equal to C_k, hence such an image can be represented as $f(C_f, C_k)= \alpha * C_f + (1-\alpha)*C_k$;

Now if we know α and C_f for this image we can get exactly those pixels locations for which the foreground object is present and then replacing C_k with some other background Vector C_b exactly. That is, New Image formed is $f(C_f, C_k, \alpha)= \alpha * C_f + (1-\alpha)*C_b$. Its an interesting concept but there is an intrinsic problem associated with the solution discussed in next section.

<u>The Intrinsic Difficulty Of Chroma Keying:</u>

We have 3 equations with 4 unknown values, so the problem is underdetermined and hence infinite solutions exist.

The 4 unknowns are R_f,G_f,B_f and α.

The 3 equations are: $R_f =\alpha * R_f + (1-\alpha)* R_b$; $G_f =\alpha * G_f + (1-\alpha)* G_b$; $B_f =\alpha * B_f + (1-\alpha)* B_b$

<u>The Solutions suggested in the paper:</u>

As per the above statements, there are infinite solutions, so finding a correct solution narrows down to the ability of applying the appropriate constraints to shrink the solution space.

In the Blue Screen Matting Paper, the solution to the problem has been stated by 3 methods and one previous work of:

1) Petro Vlahos Method

2)No Blue

3)Gray or Flesh

4)Triangualtion

<u>My Implementation:</u> I obtained 4 solutions by experimenting 4 methods.

1) Thresholding the Image(Intuitive Way)

2) No Blue method but customized as per the image background requirements

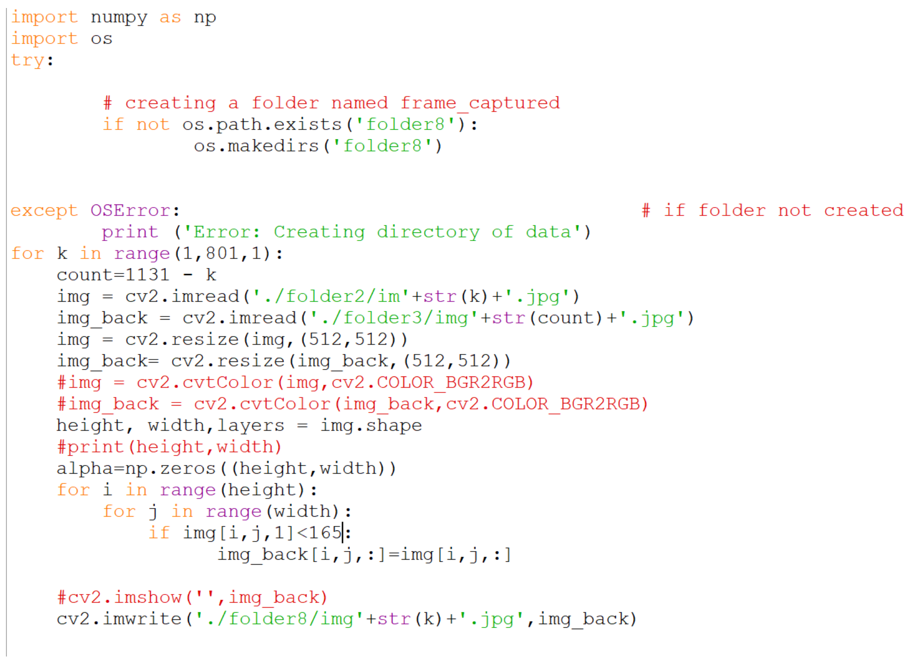
3) Hybrid of first 2 methods

4) Vlahos Formula

The best results are obtained by the Thresholding Method and the detailed results and problems faced with each technique is explained in the sections below:

<u>3.2) Thresholding the Image:</u> Here I selected the pixels of the foreground image which were having the green channel's values lesser than 165. Threshold was chosen on the basis of background colour. Then the values of such selected pixels were replaced at exact positions in the background picture for each frame and the video was created for each of the frame.

<u>Code for Thresholding the image:</u>



```
import numpy as np
import os
try:

        # creating a folder named frame_captured
        if not os.path.exists('folder8'):
                os.makedirs('folder8')


except OSError:                                        # if folder not created
        print ('Error: Creating directory of data')
for k in range(1,801,1):
    count=1131 - k
    img = cv2.imread('./folder2/im'+str(k)+'.jpg')
    img_back = cv2.imread('./folder3/img'+str(count)+'.jpg')
    img = cv2.resize(img,(512,512))
    img_back= cv2.resize(img_back,(512,512))
    #img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    #img_back = cv2.cvtColor(img_back,cv2.COLOR_BGR2RGB)
    height, width,layers = img.shape
    #print(height,width)
    alpha=np.zeros((height,width))
    for i in range(height):
        for j in range(width):
            if img[i,j,1]<165:
                  img_back[i,j,:]=img[i,j,:]

    #cv2.imshow('',img_back)
    cv2.imwrite('./folder8/img'+str(k)+'.jpg',img_back)
```

The only problem faced during this method was that single thresholding can't remove the negligible borderline green color of the foreground object, but fortunately that is hardly visible and can be considered negligible.

<u>3.3) No Blue method but customized as per the image background requirements:</u>

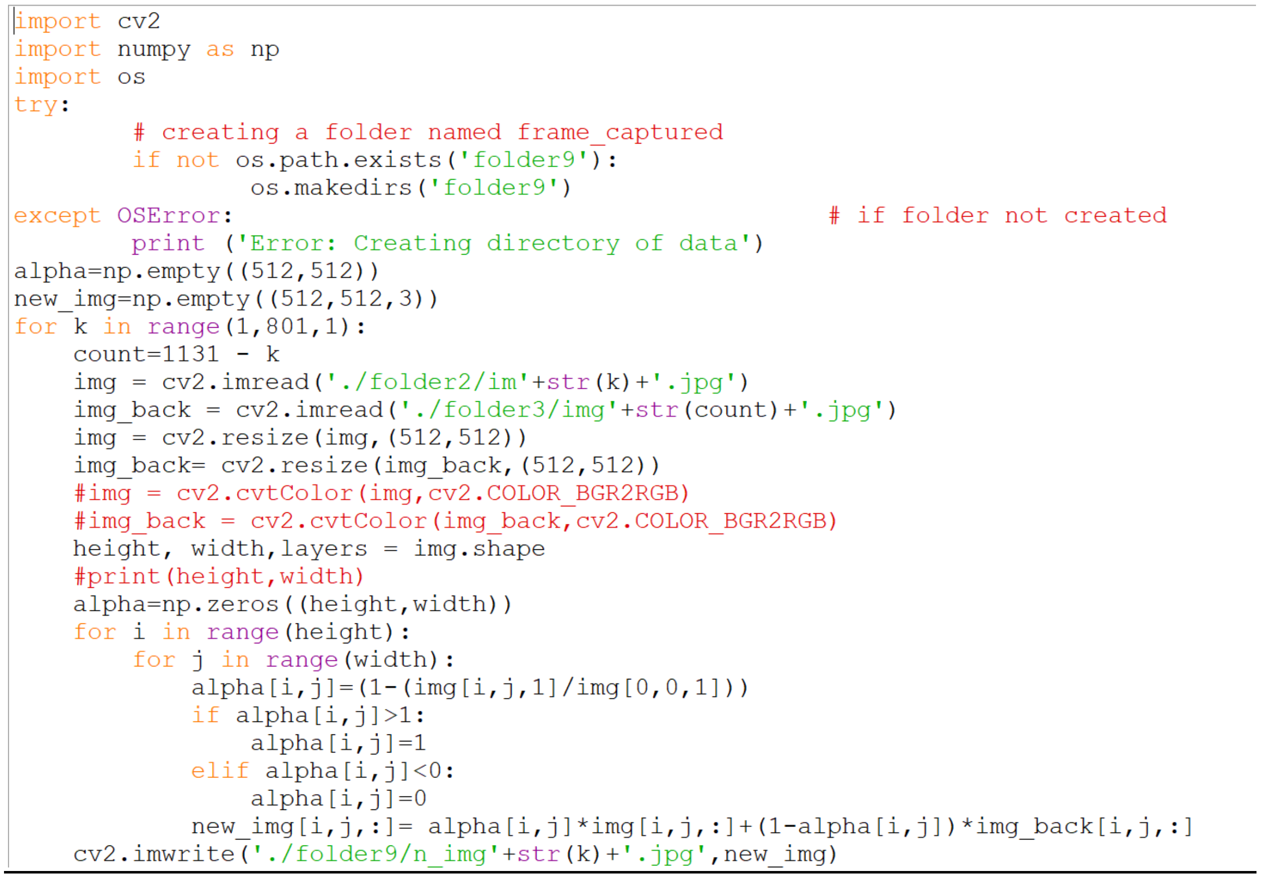
In paper, No blue method was formulated under following assumption:

There is no blue element in the foreground which is same as the background color hence foreground was considered as $[R_f,G_f,0]$ and hence $\alpha=1-(B_f/B_k)$ given $Bk!=0$

Here the background of the given foreground video is not [0,0,255] while it is [0,176,65]

So for applying the same method as above $\alpha=1-(G_f/G_k)$ was calculated and was put in the equation $f(C_f, C_k, \alpha)= \alpha * C_f + (1-\alpha)*C_b$ only for those pixels having value [0,176,65] others were put to be same as the foreground pixel values.

Code:



```
import cv2
import numpy as np
import os
try:
        # creating a folder named frame_captured
        if not os.path.exists('folder9'):
                os.makedirs('folder9')
except OSError:                                        # if folder not created
        print ('Error: Creating directory of data')
alpha=np.empty((512,512))
new_img=np.empty((512,512,3))
for k in range(1,801,1):
    count=1131 - k
    img = cv2.imread('./folder2/im'+str(k)+'.jpg')
    img_back = cv2.imread('./folder3/img'+str(count)+'.jpg')
    img = cv2.resize(img,(512,512))
    img_back= cv2.resize(img_back,(512,512))
    #img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    #img_back = cv2.cvtColor(img_back,cv2.COLOR_BGR2RGB)
    height, width,layers = img.shape
    #print(height,width)
    alpha=np.zeros((height,width))
    for i in range(height):
        for j in range(width):
            alpha[i,j]=(1-(img[i,j,1]/img[0,0,1]))
            if alpha[i,j]>1:
                alpha[i,j]=1
            elif alpha[i,j]<0:
                alpha[i,j]=0
            new_img[i,j,:]= alpha[i,j]*img[i,j,:]+(1-alpha[i,j])*img_back[i,j,:]
    cv2.imwrite('./folder9/n_img'+str(k)+'.jpg',new_img)
```

The method was not much satisfying as it lead to significant removal of green pixels within the object and also along with green channel, blue and red were also multiplied with same transparency as for the green region and hence a very dull image of the foreground was obtained.

3.4) Hybrid Of thresholding and No Blue Method:

In this method separate α was calculated for each of the channel with respect to the background pixel values[0,176,65], but as the R_f value is zero so alpha can't be calculated using this so for this channel threshold was used such that if R<10 then beackground pixels were used otherwise foreground pixels were considered.

<u>Code:</u>

```
import cv2
import numpy as np
import os
try:

        # creating a folder named frame_captured
        if not os.path.exists('folder6'):
                os.makedirs('folder6')


except OSError:                                        # if folder not created
        print ('Error: Creating directory of data')
alpha=np.empty((512,512))
new_img=np.empty((512,512,3))
for k in range(1,801,1):
    count=1131 - k
    img = cv2.imread('./folder2/im'+str(k)+'.jpg')
    img_back = cv2.imread('./folder3/img'+str(count)+'.jpg')
    img = cv2.resize(img,(512,512))
    img_back= cv2.resize(img_back,(512,512))
    #img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    #img_back = cv2.cvtColor(img_back,cv2.COLOR_BGR2RGB)
    height, width,layers = img.shape
    #print(height,width)
    alpha=np.zeros((height,width,layers))
    for i in range(height):
        for j in range(width):
                for l in range(layers):
                    if l!=2:
                        alpha[i,j,l]=(1-(img[i,j,l]/img[0,0,l]))
                        if alpha[i,j,l]>1:
                            alpha[i,j,l]=1
                        elif alpha[i,j,l]<0:
                            alpha[i,j,l]=0
                        new_img[i,j,l]= alpha[i,j,l]*img[i,j,l] +(1-alpha[i,j,l])*img_back[i,j,l]
                    elif l==2:
                        if img[i,j,l]<5:
                            new_img[i,j,l]=img_back[i,j,l]
                        else:
                            new_img[i,j,l]=img[i,j,l]


    cv2.imwrite('./folder6/n_img'+str(k)+'.jpg',new_img)
```

This gave much better results than the previous method of simply no green. But still this method had the shortcoming of slightly greater transparency of the foreground object than simple thresholding method.

<u>3.5) Vlahos Formula:</u>

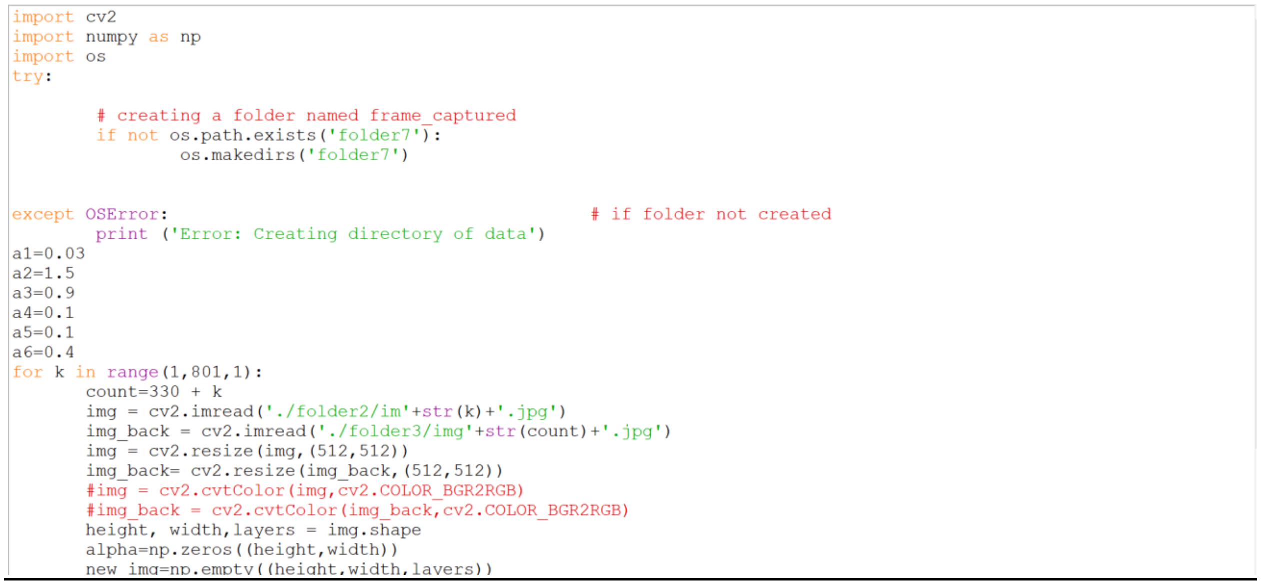
Vlahos arrived at certain formula about calculating correct α by many years of experience and experiment and not by an abstract mathematical approach.

$\alpha = 1 – ((R_f – a1) – a2 \max(a3{*}\, R_f, a4{*}B_f) – \max(a5(Rf – Bf), a6(Bf – Rf)))$

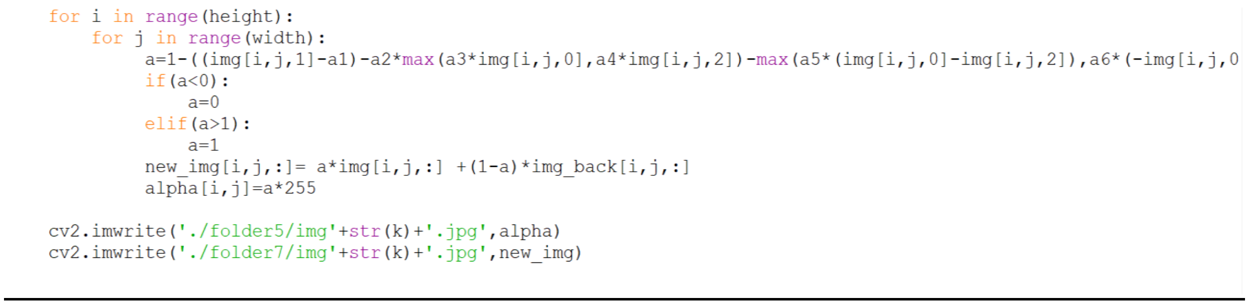
Here the result of the grayscale alpha channel depends on the parameters a1,a2,a3,a4,a5, so this can be solved as an optimization problem to give the optimal value for this to obtain chroma keying.

After hit and trial method: a1=0.03,a2=1.5,a3=0.9,a4=0.1,a5=0.1,a6=0.4 were chosen

<u>Code:</u>



```
import cv2
import numpy as np
import os
try:

        # creating a folder named frame_captured
        if not os.path.exists('folder7'):
                os.makedirs('folder7')


except OSError:                                        # if folder not created
        print ('Error: Creating directory of data')
a1=0.03
a2=1.5
a3=0.9
a4=0.1
a5=0.1
a6=0.4
for k in range(1,801,1):
       count=330 + k
       img = cv2.imread('./folder2/im'+str(k)+'.jpg')
       img_back = cv2.imread('./folder3/img'+str(count)+'.jpg')
       img = cv2.resize(img,(512,512))
       img_back= cv2.resize(img_back,(512,512))
       #img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
       #img_back = cv2.cvtColor(img_back,cv2.COLOR_BGR2RGB)
       height, width,layers = img.shape
       alpha=np.zeros((height,width))
       new_img=np.empty((height,width,layers))
```



```
    for i in range(height):
        for j in range(width):
            a=1-((img[i,j,1]-a1)-a2*max(a3*img[i,j,0],a4*img[i,j,2])-max(a5*(img[i,j,0]-img[i,j,2]),a6*(-img[i,j,0
            if(a<0):
                a=0
            elif(a>1):
                a=1
            new_img[i,j,:]= a*img[i,j,:] +(1-a)*img_back[i,j,:]
            alpha[i,j]=a*255

    cv2.imwrite('./folder5/img'+str(k)+'.jpg',alpha)
    cv2.imwrite('./folder7/img'+str(k)+'.jpg',new_img)
```

The method works well but due to lack of mathematical formulation it is difficult to reach the correct value of the tuning parameters and the values of parameters here is also sub-optimal.

<u>3.6) Results:</u>

Link for the Foreground Green screen Video: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/EUIWmExXLZRDjGx_lxonf-sByglbTgkOU14aYaHRXXJiIQ?e=haVnO6

Link for the Folder containing Foreground Green Screen Frames:

https://iiitaphyd-my.sharepoint.com/:f:/g/personal/apoorva_srivastava_research_iiit_ac_in/Ev6Uvn6W8SNEsRpx0x3f9CIBtddgHbhV9iUwlBFB9eyGQQ?e=BNyOtC

Link for the Background Video: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/ERSL5qDLyQdBmOpd_ltvUxMBIASVshQNFZuVmqSD_RGUvw?e=KPFdHj

Link for the Folder containing Background Video Frames:

https://iiitaphyd-my.sharepoint.com/:f:/g/personal/apoorva_srivastava_research_iiit_ac_in/EkAs-vCmPW1NofWL4J4b9-8BsGBydySe1mRlkhmQpAduuQ?e=NuoBJ2

Link for the Video obtained after doing the Chroma Keying Using Thresholding Technique: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/EZi2mxPDGeJIl6JZINbTo54ByfiEjSc825t1xpfA-vNSqg?e=3bLwXN

Link for the Video obtained after doing the Chroma Keying Using No Blue method but customized as per the image background requirements: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/EZuI-gXZ0RxIsHFq3GppnWABzm1zzWPlHcg1EaL2gSa4tw?e=hIHy9b

Link for the Video obtained after doing the Chroma Keying Using Hybrid of first 2 techniques: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/Ee5O1ohHQ3hEiy9UaTk7wXwB4fO3x2r6C9CZ9jbu7Dkkeg?e=v4WAf2

Link for the Video obtained after doing the Chroma Keying Using Vlahos Method: https://iiitaphyd-my.sharepoint.com/:v:/g/personal/apoorva_srivastava_research_iiit_ac_in/EatQKIB6S2tIsSZFNNl3cdsBupSK0wZ6AJ2j9km0013bvQ?e=IsUGml

Link for the folder containing alpha channel obtained for the Vlahos method: https://iiitaphyd-my.sharepoint.com/:f:/g/personal/apoorva_srivastava_research_iiit_ac_in/EsFYWcv_GslLiZvhMTneLUMBy9_SFpdtjQP9AiHuc0t9lQ?e=5U6feS