CV Assignment 0

Aryan Jain **2019101056**

***the complete drive link which contains jupyter notebook and all inputs and outputs is available at:**

https://drive.google.com/drive/u/0/folders/1X-PREcb9vKmqzgEXL9jFlJM30rCgyPZn

Problem 1

Description:

Given a video, it is asked to break the video into frames and save them in a specified folder. In the next part it is asked to combine the frames saved in the folder to generate a video whose frame rate can be varied based on user's choice.

Solution:

Used cv2.VideoCapture() to get an object of the input video and then used read() function to get the frames. In my case the video used to generate the frames is named 'input.mp4' and the frames are stored in 'images' folder in the same directory. The video formed by combining the saved frames is stored as 'output.mp4' . For the 2nd part I took the frames per second as 30, which can be changed based on user's preference by just changing the 'fps' variable in the code.

Experiments:

The 'fps' variable in the 2nd part (frames per second) is adjusted in order to get the continuous effect in the video, rather than a fast forward flow of frames.

Challenges:

As such no challenges were faced.

Learning:

The persistence of vision of human eye is 1/16 seconds, that means a human eye can distinguish 16 frames in a second. So to create an optical illusion effect or a continuous flow effect in videos, we need to keep frame rate to be more than 16 fps. So was did and I pick fps as 30 for my case.

Results:

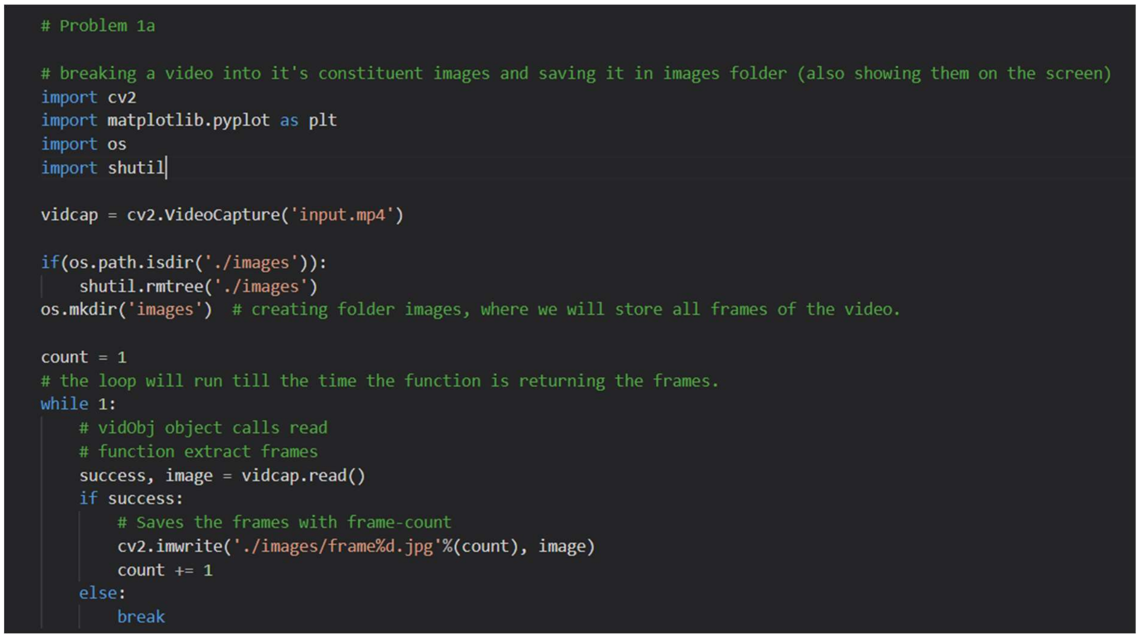
The frames formed using the input.mp4 video are available at this link (for 1a) :

https://drive.google.com/drive/u/0/folders/1thN-VW_31uWMAEm0cyhTRyVG1ig-P5fJ

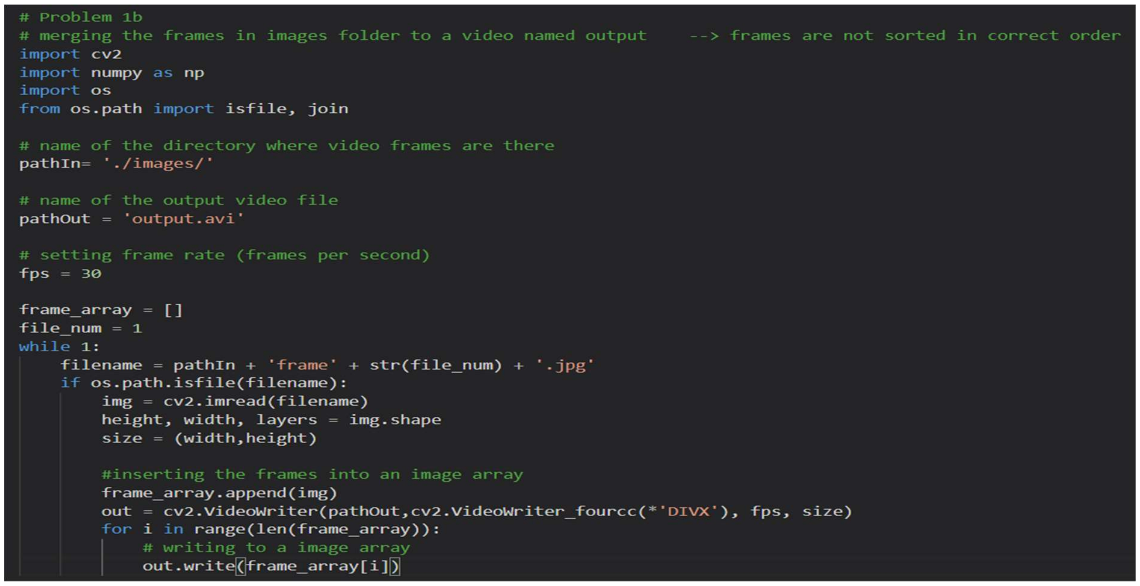
The video (output.avi) is available at this link (for 1b):

https://drive.google.com/drive/u/0/folders/1X-PREcb9vKmqzgEXL9jFlJM30rCgyPZn

Code :



```python
# Problem 1a

# breaking a video into it's constituent images and saving it in images folder (also showing them on the screen)
import cv2
import matplotlib.pyplot as plt
import os
import shutil

vidcap = cv2.VideoCapture('input.mp4')

if(os.path.isdir('./images')):
    shutil.rmtree('./images')
os.mkdir('images')  # creating folder images, where we will store all frames of the video.

count = 1
# the loop will run till the time the function is returning the frames.
while 1:
    # vidObj object calls read
    # function extract frames
    success, image = vidcap.read()
    if success:
        # Saves the frames with frame-count
        cv2.imwrite('./images/frame%d.jpg'%(count), image)
        count += 1
    else:
        break
```



```
# Problem 1b
# merging the frames in images folder to a video named output    --> frames are not sorted in correct order
import cv2
import numpy as np
import os
from os.path import isfile, join

# name of the directory where video frames are there
pathIn= './images/'

# name of the output video file
pathOut = 'output.avi'

# setting frame rate (frames per second)
fps = 30

frame_array = []
file_num = 1
while 1:
    filename = pathIn + 'frame' + str(file_num) + '.jpg'
    if os.path.isfile(filename):
        img = cv2.imread(filename)
        height, width, layers = img.shape
        size = (width,height)

        #inserting the frames into an image array
        frame_array.append(img)
        out = cv2.VideoWriter(pathOut,cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
        for i in range(len(frame_array)):
            # writing to a image array
            out.write(frame_array[i])
```

Problem 2

Description:

In this problem, we are asked to capture the frames from a webcam connected to the computer and save them in some folder. Also, we need to show the video on the screen while recording.

Solution:

Here we create an object (named 'cap') using cv2.VideoCapture(0) and run an infinite 'while loop' using cap.isOpened() function. The loop will run for infinite time and will break only when the 'q' (as exit) is pressed. Meanwhile the frame are saved in 'webcam_images' folder and video is shown on the screen using 'cv2.imshow()' function.

Experiments:

No experiment is involved.

Challenges:

No challenge is involved.

Learning:

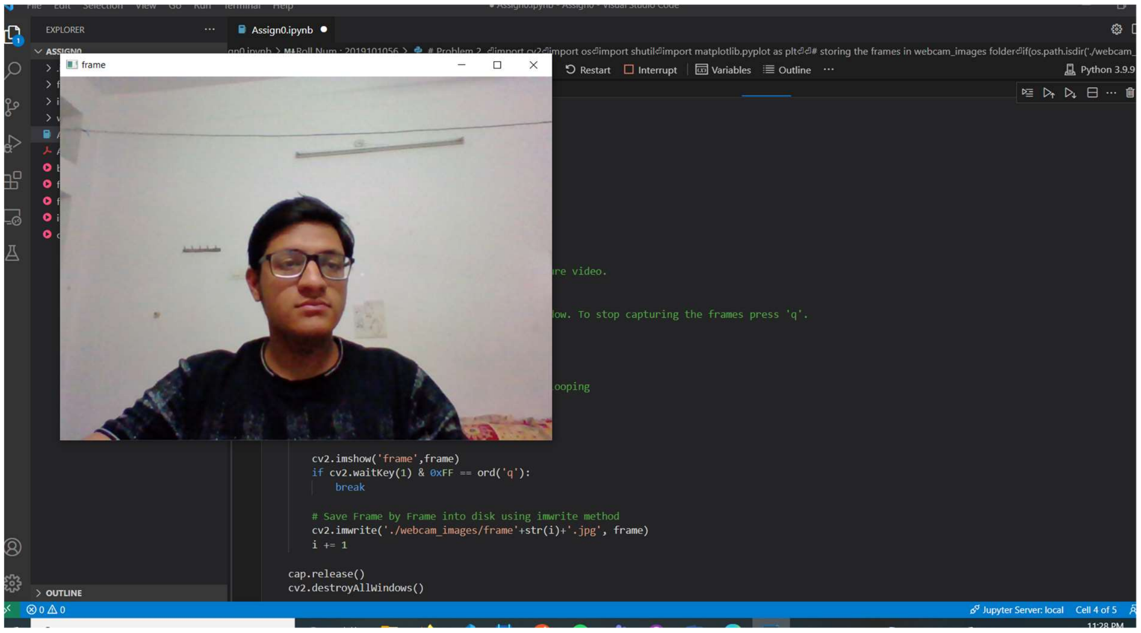
Learnt how to capture frames from a webcam connected to the computer and simultaneously display the video recorded on the screen.

Results:

The frames captured by webcam are saved in a local folder and is made available at :

https://drive.google.com/drive/u/0/folders/1TG7iQ8a0mUGDHOEKnctOWmYwq7SsYH0I

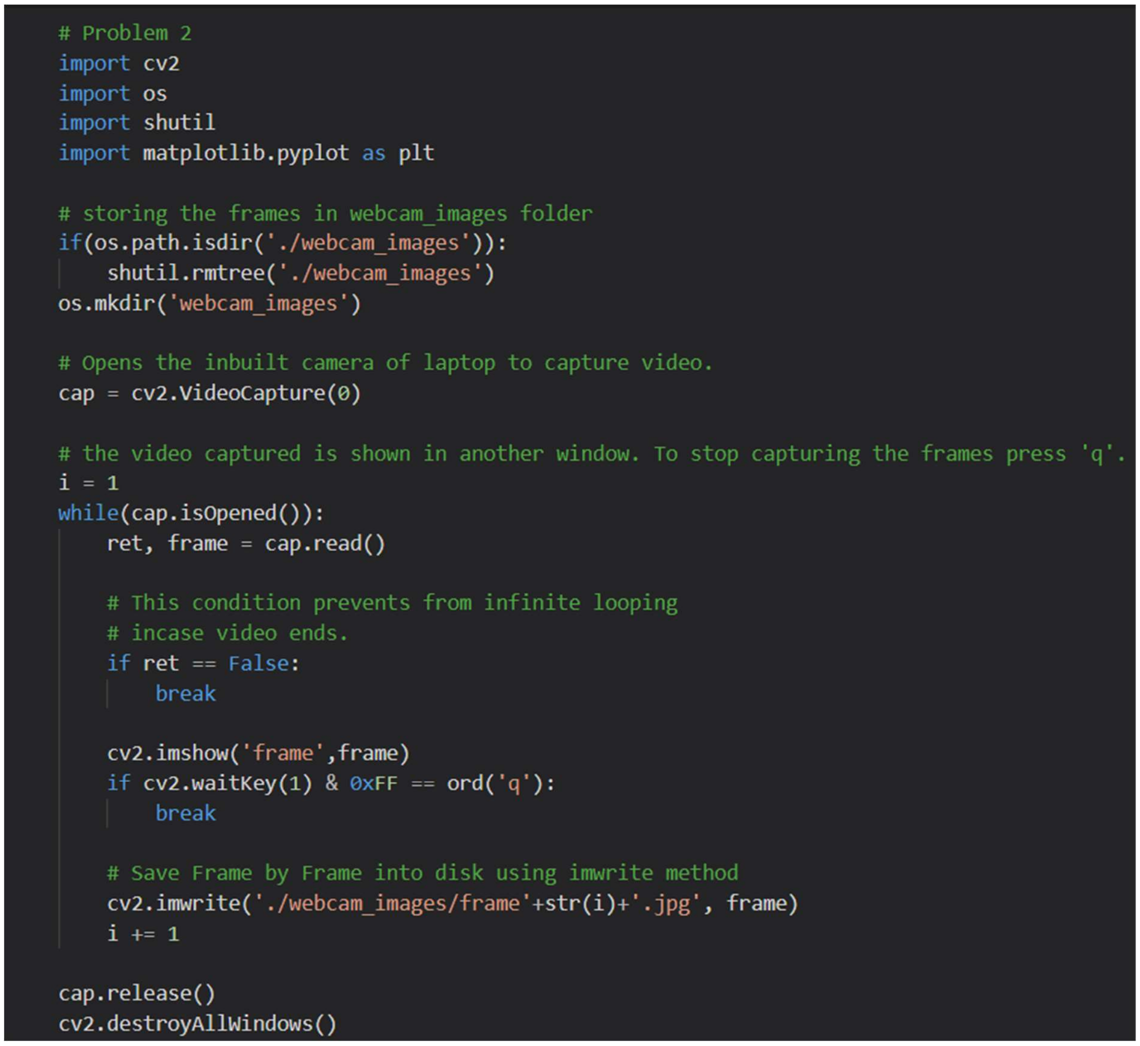
Below is the proof of displaying the video on the screen while capturing the frames through webcam.



```
gn0.ipynb > M↓Roll Num : 2019101056 > # Problem 2 ⏎import cv2⏎import os⏎import shutil⏎import matplotlib.pyplot as plt⏎⏎# storing the frames in webcam_images folder⏎if(os.path.isdir('./webcam_
```

```
re video.

low. To stop capturing the frames press 'q'.

ooping

    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    # Save Frame by Frame into disk using imwrite method
    cv2.imwrite('./webcam_images/frame'+str(i)+'.jpg', frame)
    i += 1

cap.release()
cv2.destroyAllWindows()
```

Code:



```python
# Problem 2
import cv2
import os
import shutil
import matplotlib.pyplot as plt

# storing the frames in webcam_images folder
if(os.path.isdir('./webcam_images')):
    shutil.rmtree('./webcam_images')
os.mkdir('webcam_images')

# Opens the inbuilt camera of laptop to capture video.
cap = cv2.VideoCapture(0)

# the video captured is shown in another window. To stop capturing the frames press 'q'.
i = 1
while(cap.isOpened()):
    ret, frame = cap.read()

    # This condition prevents from infinite looping
    # incase video ends.
    if ret == False:
        break

    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    # Save Frame by Frame into disk using imwrite method
    cv2.imwrite('./webcam_images/frame'+str(i)+'.jpg', frame)
    i += 1

cap.release()
cv2.destroyAllWindows()
```

Problem 3

Description:

Using the Chroma Keying technique, we are asked to create an interesting composition of 2 videos.

Solution:

In the first video (in my case I called it as ‘foreground.mp4’), we will take a green screen background video. And we can take nice background as the 2nd video (called ‘background.mp4’ in my case). We will divide the videos into frames, and for the first video we will create a mask using which we will be remove the green screen from the background and extract the require portion. The same mask can be used for the background video frames to mask the region where the characters from the foreground video are coming.

After this we will combine both the images and store these new frames in another folder named ‘fg-bg_images’. After this we will combine the frames into video just like we did in 2nd part of Problem 1.

Challenges:

The main challenge was in picking the values for ‘lower_green’ and ‘upper_green’ triplet, which is used for creating mask for the green screen video. (These triplets are defined in the code).

Experiment:

The values in these triplets(lower_green and upper_green) are needed to be chosen in such a way that it removes the green screen from the background of the main character in the foreground.mp4 video, which is a tricky task because all green screen background is not same in all videos. (Different green color cloth can be used to film the scenes at different places, different time). So we need to choose an appropriate value for it.

Learning:

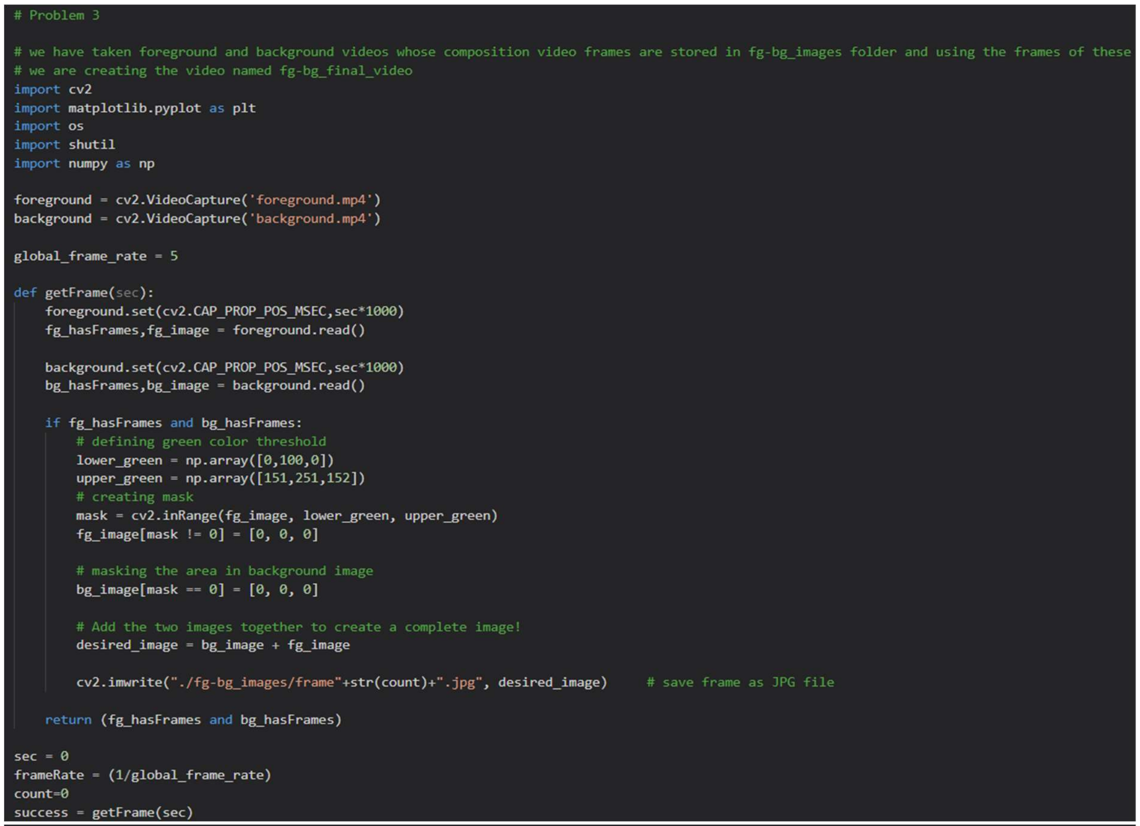
Learnt how to apply Chroma Keying and understood how it is used in general to create videos and movies with nice background.

Results:

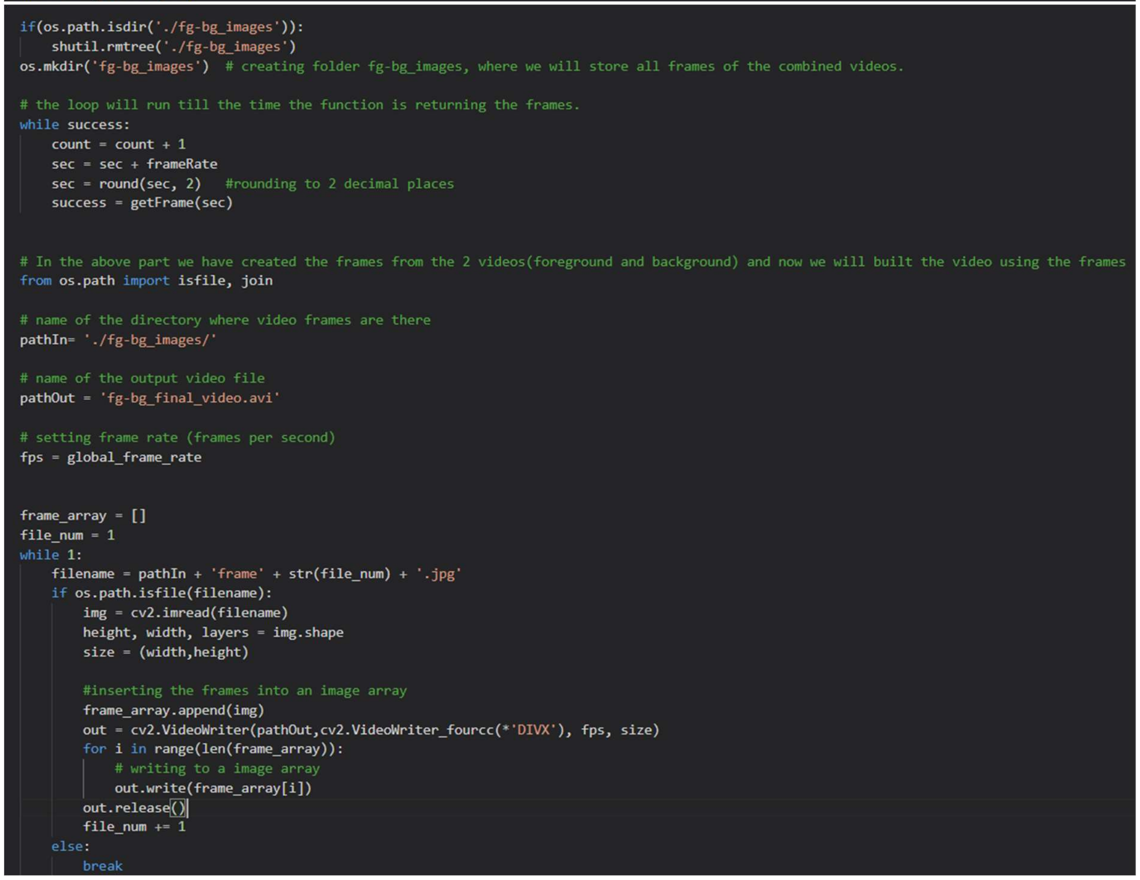
The final video (fg-bg_final_video.avi) formed by the composition of 2 videos using the Chroma Keying is available at :

https://drive.google.com/drive/u/0/folders/1X-PREcb9vKmqzgEXL9jFlJM30rCgyPZn

Code:



```python
# Problem 3

# we have taken foreground and background videos whose composition video frames are stored in fg-bg_images folder and using the frames of these
# we are creating the video named fg-bg_final_video
import cv2
import matplotlib.pyplot as plt
import os
import shutil
import numpy as np

foreground = cv2.VideoCapture('foreground.mp4')
background = cv2.VideoCapture('background.mp4')

global_frame_rate = 5

def getFrame(sec):
    foreground.set(cv2.CAP_PROP_POS_MSEC,sec*1000)
    fg_hasFrames,fg_image = foreground.read()

    background.set(cv2.CAP_PROP_POS_MSEC,sec*1000)
    bg_hasFrames,bg_image = background.read()

    if fg_hasFrames and bg_hasFrames:
        # defining green color threshold
        lower_green = np.array([0,100,0])
        upper_green = np.array([151,251,152])
        # creating mask
        mask = cv2.inRange(fg_image, lower_green, upper_green)
        fg_image[mask != 0] = [0, 0, 0]

        # masking the area in background image
        bg_image[mask == 0] = [0, 0, 0]

        # Add the two images together to create a complete image!
        desired_image = bg_image + fg_image

        cv2.imwrite("./fg-bg_images/frame"+str(count)+".jpg", desired_image)     # save frame as JPG file

    return (fg_hasFrames and bg_hasFrames)

sec = 0
frameRate = (1/global_frame_rate)
count=0
success = getFrame(sec)
```



```python
if(os.path.isdir('./fg-bg_images')):
    shutil.rmtree('./fg-bg_images')
os.mkdir('fg-bg_images')  # creating folder fg-bg_images, where we will store all frames of the combined videos.

# the loop will run till the time the function is returning the frames.
while success:
    count = count + 1
    sec = sec + frameRate
    sec = round(sec, 2)   #rounding to 2 decimal places
    success = getFrame(sec)


# In the above part we have created the frames from the 2 videos(foreground and background) and now we will built the video using the frames
from os.path import isfile, join

# name of the directory where video frames are there
pathIn= './fg-bg_images/'

# name of the output video file
pathOut = 'fg-bg_final_video.avi'

# setting frame rate (frames per second)
fps = global_frame_rate


frame_array = []
file_num = 1
while 1:
    filename = pathIn + 'frame' + str(file_num) + '.jpg'
    if os.path.isfile(filename):
        img = cv2.imread(filename)
        height, width, layers = img.shape
        size = (width,height)

        #inserting the frames into an image array
        frame_array.append(img)
        out = cv2.VideoWriter(pathOut,cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
        for i in range(len(frame_array)):
            # writing to a image array
            out.write(frame_array[i])
        out.release()
        file_num += 1
    else:
        break
```