Assignment-0 : OpenCV and Chroma Keying

Name - Sourabh Patidar

Roll No. - 2021201089

++

Requirements

- Language Used - Python 3.8.10
- Libraries required -
  - openCV
  - Numpy

Importing all the required libraries

In [1]:
```
import cv2
import os
import numpy as np
```

Task 1 - Video ↔ Images:

Problem Statement -

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.

Solution Approach -

The task involves two steps. First step is to extract all the frames from given video and after that using this frames create a video with specified frame rate.

1. **Extracting frames from a Video :** To read the video we can use VideoCapture() function of cv2 library in openCV. Then after store video in a variable we can read frames of video by read() function in cv2 library and store each frame in specified folder using imwrite function.
2. **Combining frames to form a Video :** Now to convert the frames into a video we will read each frame one by one and will store then in an array or list. Now we will create an VideoWriter object. VideoWriter object take outputfile name ,

cv2.VideoWriter_fourcc(*'DIVX'), frame rate and size of frame as arguments. As a last step we have to write all the frame in VideoWriter object.

Code -

In [2]:
```
def convert_video_to_frames(filename, outputPath):
    video_capture = cv2.VideoCapture(filename)
    count = 1
    while True:
        success,image = video_capture.read()
        if success == False:
            break
        cv2.imwrite(outputPath + "frame%d.jpg" % count, image)
        print("Frame {} saved!!!!".format(count))
        count += 1
    return
```

In [3]:
```
def convert_frames_to_video(inputPath,outputPath,frame_rate):
    frame_array = []
    no_of_frames = len(os.listdir(inputPath))
    for i in range(no_of_frames):
        frame = cv2.imread(inputPath+"frame%d.jpg" %(i+1))
        frame_array.append(frame)
    height, width, layers = frame_array[0].shape
    size = (width,height)
    out = cv2.VideoWriter(outputPath,cv2.VideoWriter_fourcc(*'DIVX'),
    for i in range(len(frame_array)):
        print("Frame {} added to video".format(i+1))
        out.write(frame_array[i])
    out.release()
```

In [4]:
```
filename = "Input/task1.mp4"
outputPath = "Output/Task1/"
convert_video_to_frames(filename, outputPath)
```

```
Frame 1 saved!!!!
Frame 2 saved!!!!
Frame 3 saved!!!!
Frame 4 saved!!!!
Frame 5 saved!!!!
Frame 6 saved!!!!
Frame 7 saved!!!!
Frame 8 saved!!!!
Frame 9 saved!!!!
Frame 10 saved!!!!
Frame 11 saved!!!!
Frame 12 saved!!!!
Frame 13 saved!!!!
Frame 14 saved!!!!
Frame 15 saved!!!!
Frame 16 saved!!!!
Frame 17 saved!!!!
Frame 18 saved!!!!
Frame 19 saved!!!!
```

In [5]:
```
inputPath = "Output/Task1/"
outputPath = "Output/task1.avi"
frame_rate = 25.0
convert_frames_to_video(inputPath, outputPath, frame_rate)
```

```
Frame 1 added to video
Frame 2 added to video
Frame 3 added to video
Frame 4 added to video
Frame 5 added to video
Frame 6 added to video
Frame 7 added to video
Frame 8 added to video
Frame 9 added to video
Frame 10 added to video
Frame 11 added to video
Frame 12 added to video
Frame 13 added to video
Frame 14 added to video
Frame 15 added to video
Frame 16 added to video
Frame 17 added to video
Frame 18 added to video
Frame 19 added to video
```

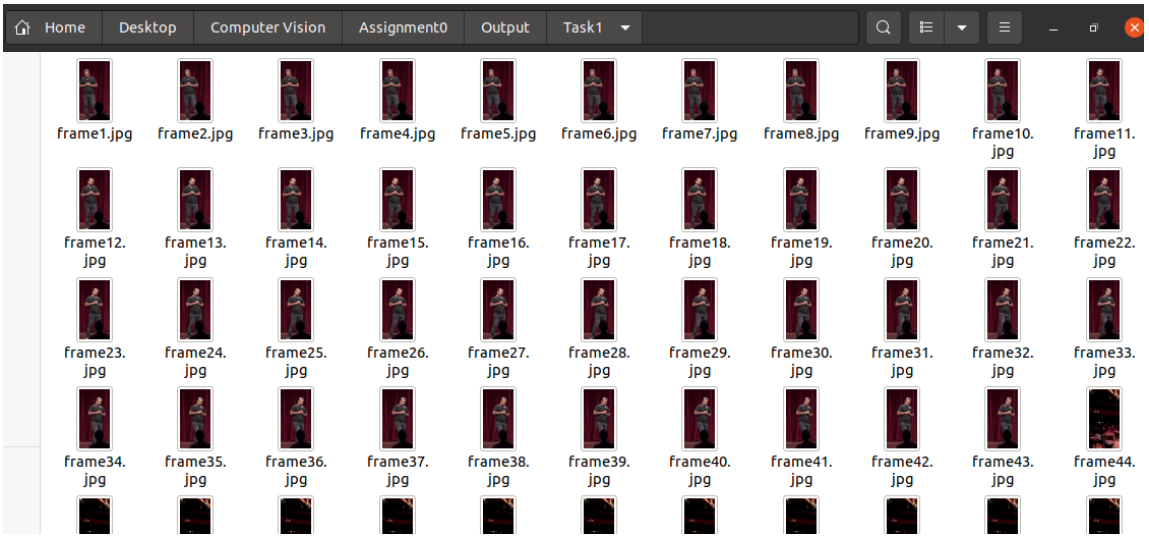
Challenges

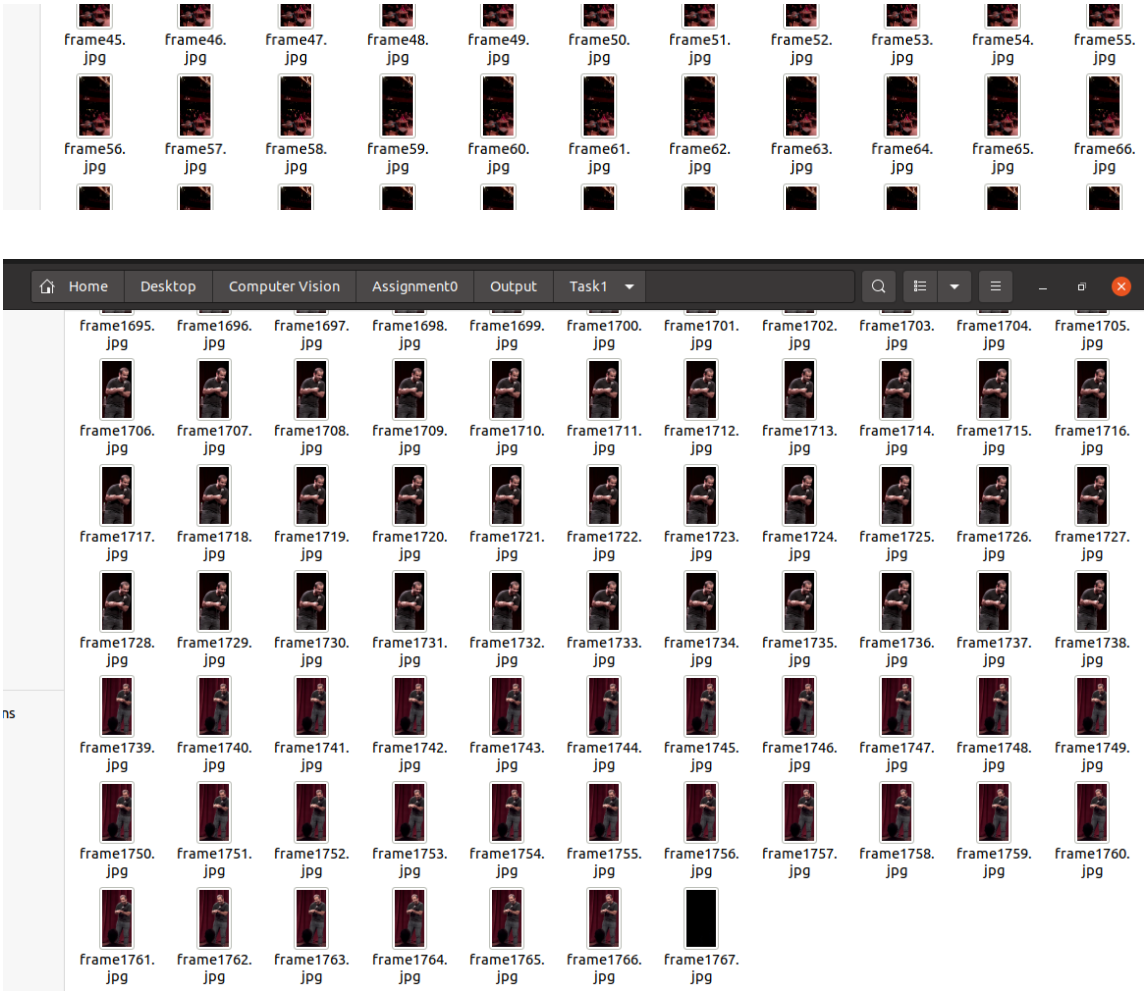
- Sorting the string format for name of the frames dynamically.
- Hardware Limitations.

Learning

- Reading video and extracting frames from a video using openCV.
- Combining frames to create a video using openCV.
- Managing frame rate of any video.

Sample Results





++

Task 2 - Capturing Images:

Problem Statement

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.

Solution Approach

Firstly to start webcam, we have to create a object of VideoCapture and then using read method of VideoCapture class we will capture the frames and then using imwrite function the frame can to saved to given folder location. To stop the webcam press 'esc' button.

Code

In [6]:
```
cap = cv2.VideoCapture(0)
i = 1
```

```
while (cap.isOpened()):
    ret, frame = cap.read()
    stop = cv2.waitKey(33)
    if stop == 27:
        break
    cv2.imshow('image', frame)
    cv2.imwrite('Output/Task2/frame' + str(i) + '.jpg', frame)
    cv2.waitKey(1)
    i += 1

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

```
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (935) ope
n OpenCV | GStreamer warning: Cannot query video position: status=
0, value=-1, duration=-1
```

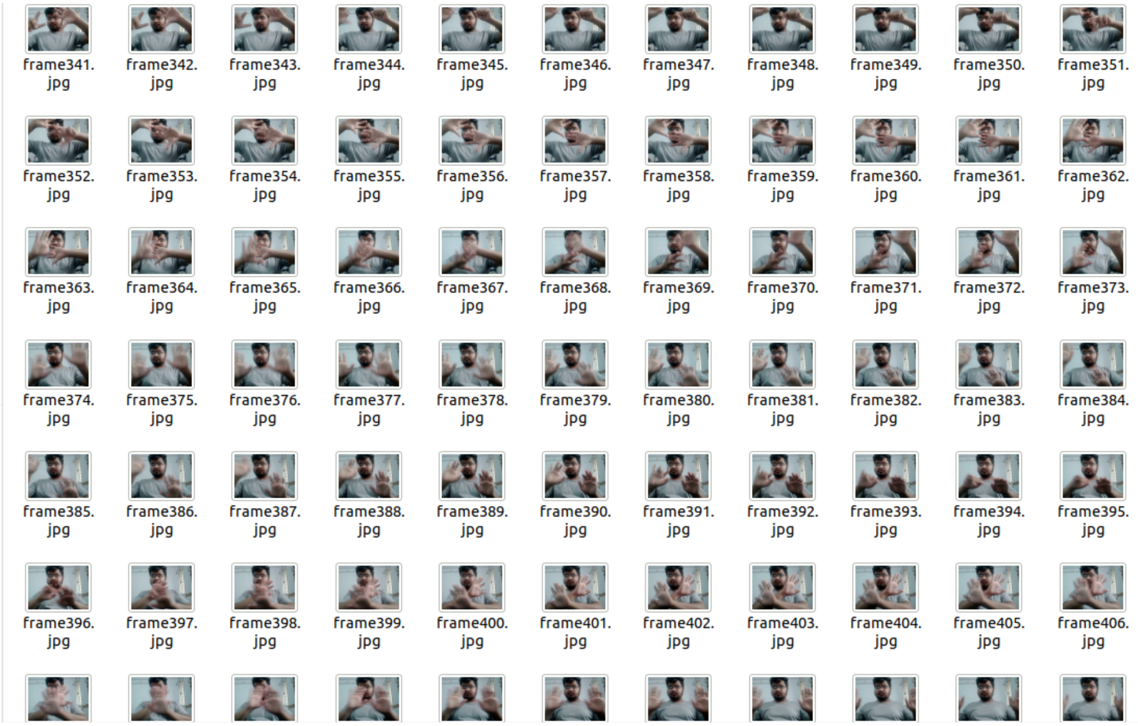
Challenges

- Showing the live frames.
- Maintaining the delay.
- Stoping the webcam from capturing the frames.

Learning

- Reading live frames from a webcam and storing it at appropriate folder .
- Displaying the live frames captured with some millisecond delay.

Sample Results



++

Task 3 - Chroma Keying:

Problem Statement

Read about the technique of chroma keying. Following are a few good starting points:

- Introduction: [http://en.wikipedia.org/wiki/Chroma (http://en.wikipedia.org/wiki/Chroma)](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.

Solution Approach

1. Extracting frames from video with green screen : First step for this experiment is to extract all the frames from the video with the green screen and store all the frames in some folder. To extract the frames we will use the convert_video_to_frames function from Task 1.

2. Extracting frames from background video : Now we will extract all the frames from the video we want to put in background and will save the frames in some other folder. We will follow same process as above to extract the frames.

3. Compositing Single Frame : Take frame1 from both the videos. Now create a copy of green screen frame and then find the mask of image using inRange function.The inRange function will convert the green part of screen to white and non-green part to black. Now we will take another copy of the original image and will convert the green screen to black using

the previous masked image. Now we got our masked green screen image. Now we will work with background image. Firstly we will crop the background to fit in the size of the green screen image. Now we will set the pixels equal to 0 in non-green part of green screen image. Finally we will get the composite frame by adding both the images. We will write this image in some folder using imwrite function.

4. Applying above step on all the frames : Now we can apply the above step to combine all the frames present in green screen video and background video. Note that no. of composite frames will be equal to minimum of no of frames in green_screen video and background video.

5. Converting new frames into a video : Now we have got all the frames for the composite video. We will now combine all the frames to form a final video. We can use convert_frames_to_video function from Task 1 to do the same.

Code

In [7]:
```
green_screen_video = "Input/green_screen.mp4"
background_video = "Input/back.mp4"
outputPath_gs = "Output/Task3/gs/"
outputPath_bg = "Output/Task3/bg/"
convert_video_to_frames(green_screen_video ,outputPath_gs)
convert_video_to_frames(background_video ,outputPath_bg)
```

```
Frame 1 saved!!!!
Frame 2 saved!!!!
Frame 3 saved!!!!
Frame 4 saved!!!!
Frame 5 saved!!!!
Frame 6 saved!!!!
Frame 7 saved!!!!
Frame 8 saved!!!!
Frame 9 saved!!!!
Frame 10 saved!!!!
Frame 11 saved!!!!
Frame 12 saved!!!!
Frame 13 saved!!!!
Frame 14 saved!!!!
Frame 15 saved!!!!
Frame 16 saved!!!!
Frame 17 saved!!!!
Frame 18 saved!!!!
Frame 19 saved!!!!
```

In [8]:
```
count = 1
lower_bound_of_green = np.array([0, 90, 0])
upper_bound_of_green = np.array([120, 255, 100])
while True:
    frame = cv2.imread('Output/Task3/gs/frame%d.jpg' % count)
    if frame is None:
        break
    frame_copy = np.copy(frame)
    mask = cv2.inRange(frame_copy, lower_bound_of_green, upper_bound_
    masked_image = np.copy(frame_copy)
    masked_image[mask != 0] = [0, 0, 0]
    background_frame = cv2.imread('Output/Task3/bg/frame%d.jpg' % cou
```

```
    if background_frame is None:
        break
    crop_background = background_frame[0:720, 0:1280]
    crop_background[mask == 0] = [0, 0, 0]
    final_image = crop_background + masked_image
    print("Frame Composited : ", count)
    cv2.imwrite("Output/Task3/final/frame%d.jpg" % count, final_image
    count = count+1
    cv2.destroyAllWindows()
```

```
Frame Composited :  1
Frame Composited :  2
Frame Composited :  3
Frame Composited :  4
Frame Composited :  5
Frame Composited :  6
Frame Composited :  7
Frame Composited :  8
Frame Composited :  9
Frame Composited :  10
Frame Composited :  11
Frame Composited :  12
Frame Composited :  13
Frame Composited :  14
Frame Composited :  15
Frame Composited :  16
Frame Composited :  17
Frame Composited :  18
Frame Composited :  19
```

In [9]:
```
inputPath = "Output/Task3/final/"
outputPath = "Output/task3.avi"
frame_rate = 25.0
convert_frames_to_video(inputPath, outputPath, frame_rate)
```

```
Frame 1 added to video
Frame 2 added to video
Frame 3 added to video
Frame 4 added to video
Frame 5 added to video
Frame 6 added to video
Frame 7 added to video
Frame 8 added to video
Frame 9 added to video
Frame 10 added to video
Frame 11 added to video
Frame 12 added to video
Frame 13 added to video
Frame 14 added to video
Frame 15 added to video
Frame 16 added to video
Frame 17 added to video
Frame 18 added to video
Frame 19 added to video
```

Challenges

- Finding most appropriate lower and upper bound of RGB values.

- Masking of both the frames.
- Croping background image to fit in the green screen frame.
- Hardware Limitations (not able to process longer video(>2 min) due to hardware limitation).

Learning

- Applying Chroma Keying to make composite of two videos/images.
- Re-sizing of an Image.
- Making composite of two images.
- Masking of an image.

Sample Results

Before :

Composite Frame :