Jeet Vora
2019701006

Assignment 0

CSE578 - Computer Vision
January 24, 2020

1 Video ↔ Images

1.1 Write a program to convert a given video to its constituent images. Your output should be in a specified folder.

Description **:** Need to obtain frames from video and store it into specified folder.
Solution **:**
OpenCV library is used to extract frames from videos.
VideoCapture(Filepath) : Read the video(.mp4 format)
read() : Read data depending upon the type of object that calls
imwrite(filename, img[, params]) : Saves an image to a specified file.

Steps **:**

1. Open the Video file using cv2.VideoCapture()
2. Read frame by frame
3. Save each frame using cv2.imwrite()
4. Release the VideoCapture and destroy all windows

Experiments Performed **:** Extracting frames at different frame rates example, framerate = 0.5 gives frames for each half minute. Similarly, performed for framerate = 1, 0.1, 0.01

Input video(link) **:** Input Video
Outputs(link) **:** Output Frames

Figure 1: Frames captured from video

Assignment 0

***Code* :**

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import os

def Capture_Frames(path,output_dir,frameRate):

    # Path to video file caputers into object
    vidObj = cv.VideoCapture(path)

    # Used as counter variable
    count = 0

    # checks whether frames were extracted
    success = 1
    sec = 1
    while success:

        vidObj.set(cv.CAP_PROP_POS_MSEC,sec*1000)

        # vidObj object calls read function extract frames
        success,image = vidObj.read()
        if not success:
            print("Can't receive frame. Exiting ...")
            break

        # Saves the frames with frame-count
        # save frame as JPG file
        cv.imwrite(os.path.join(output_dir, "image_"+str(count)+".jpg"), image)

        count+=1
        sec = sec + frameRate
        sec = round(sec, 2)

if __name__ == '__main__':

    # Calling the function
    path = "input_data\\video1.mp4"
    output_dir = "output_data\\question1"
    Capture_Frames(path,output_dir,frameRate=1)
```

1.2 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.

Description **:** Need to obtain video from frames and store it into specified folder.
Solution **:**
OpenCV library is used to generate video from given images.
VideoWriter() : is used to create video files

Steps **:**

1. Form an array of frames in sequence
2. Creating VideoWriter object in which size of video and fps is specified
3. Write each image into VideoWriter object.
4. Release the VideoWriter.

Experiments Performed **:** Generating videos by specifying various fps(frames per second), example fps = 1, 10, 20, 30
Input(link) **:** Input Frames
Output video(link) **:** OutputVideo
Code **:**

```
def Frames_to_Video(input_dir,output_dir,fps):
    img_array = []

    # Taking each image and forming array of images
    for count in range(len(os.listdir(path))):
        filename = path+'/image_'+str(count)+'.jpg'
        img = cv.imread(filename)
        height, width, layers = img.shape
        size = (width,height)
        img_array.append(img)

    # Defining the path and filename to save with fps and size(w*h)
    vidoutObj = cv.VideoWriter(os.path.join(output_dir,'project.avi'),cv.VideoWriter_fou

    # each image in image_array is written to videowriter object
    for i in range(len(img_array)):
        vidoutObj.write(img_array[i])
    vidoutObj.release()
```

```
if __name__ == '__main__':

    # Calling the function
    path = "input_data\\question3"
    output_dir = "output_data"
    Frames_to_Video(path,output_dir,fps=10)
```

2 Capturing Frames from webcam

2.1 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.

Description **:** Need to obtain frames from webcam or any external connected camera.
Solution **:**

1. Open the Video file using cv2.VideoCapture() with parameter 0.
2. Read frame by frame
3. Save each frame using cv2.imwrite()
4. Release the VideoCapture and destroy all windows

Experiments Performed **:**Extracting frames at different frame rates example, framerate = 0.5 gives frames for each half minute. Similarly, performed for framerate = 1, 0.1

Output(link) **:** Output frames from webcam

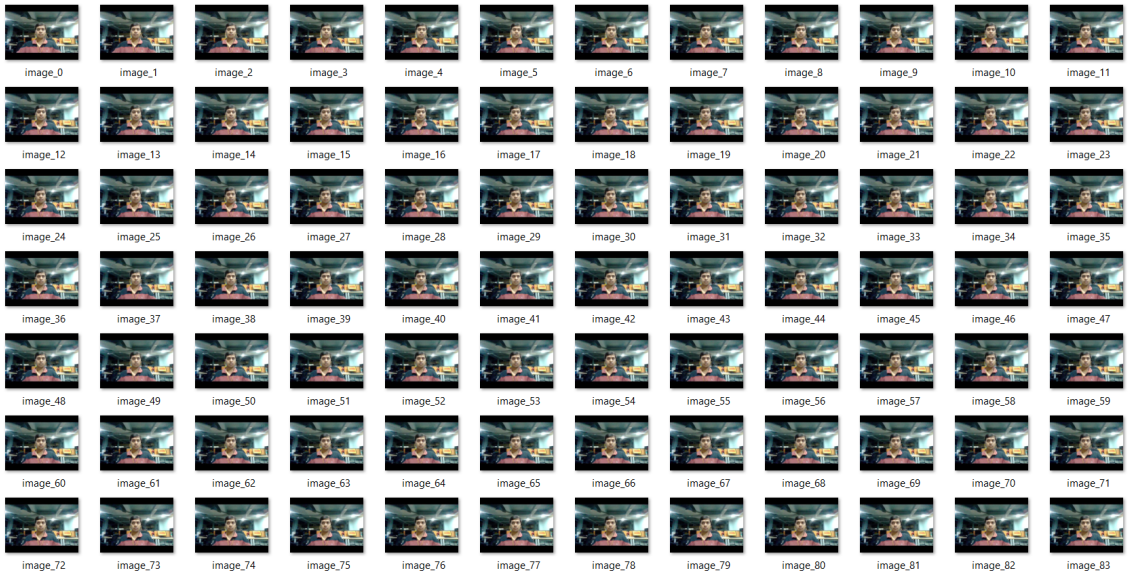

Figure 2: Frames captured from webcam

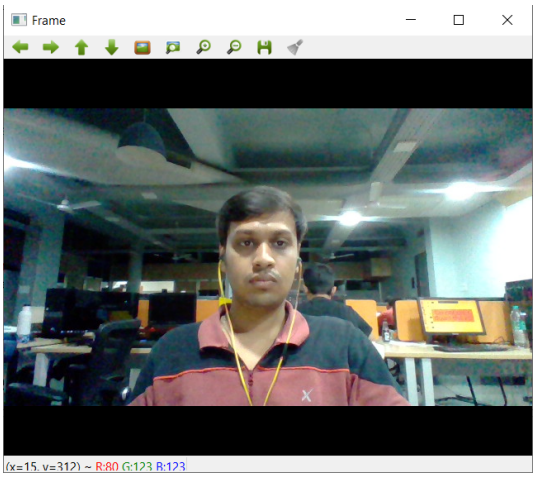

Figure 3: Video rendering captured from webcam

***Code* :**

```
def Cam_Capture_Frames(output_dir,frameRate):
    capture = cv.VideoCapture(0)

    if not capture.isOpened():
        print("Cannot open camera")
        exit()
    count = 0
    success = True
    print('Press q to exit')
    sec = 1
    while success:

        capture.set(cv.CAP_PROP_POS_MSEC,sec*1000)
        # Capture frame-by-frame
        success, frame = capture.read()
        # if frame is read correctly ret is True
        if not success:
            print("Can't receive frame. Exiting ...")
            break

        # Display the resulting frame
        cv.imwrite(os.path.join(output_dir, "image_"+str(count)+".jpg"), frame)
        cv.imshow('Frame', frame)
        if cv.waitKey(1) == ord('q'):
            break
        count+=1
        sec = sec + frameRate
        sec = round(sec, 2)
    # When everything done, release the capture
    capture.release()
    cv.destroyAllWindows()

 if __name__ == '__main__':

    # Calling the function
    output_dir = "output_data\\question2"
    Cam_Capture_Frames(output_dir,frameRate=0.5)
```

3 Chroma Keying

Description **:** Chroma keying is a technique used for combining two frames or images by replacing a color or a color range in one frame with that from the another frame. It is often used in film industry to replace a scene's background by using a blue or green screen as the initial background and placing the actor in the foreground. The principle behind chroma keying is that the color blue is the opposite color of skin tone, so a distinction between the two is very clear, making it easier to select the color without worrying about any part of the actor being included in the selection. The whole blue selection is then replaced with another frame as the background. Chroma key is also known as color keying and color separation overlay; it is also commonly called blue screen or green screen.

Solution **:**

1. Open both foreground(fg) and background(bg) video files using cv2.VideoCapture()
2. Read frame by frame from each fg and bg.
3. Convert bgframe to HSV and specify color range which is to be removed.
4. Create inverted mask with specified range of color to be removed.
5. Perform bitwise AND of inverted mask with bgframe to remove color from background.
6. Perform bitwise AND of mask with fgframe to remove the pixels from fgframe which will be replaced by bgframe.
7. Perform addition of both fgframe and bgframe.
8. Release the VideoCapture and destroy all windows

Input **:** Input videos ***Output*** **:** Output video Chroma Keying

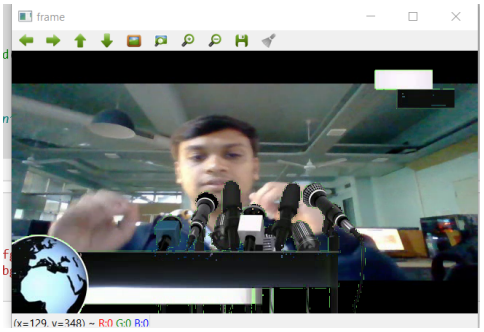

Figure 4: Chroma Keying

Code **:**

```
def chroma_keying(bgpath,fgpath,output_dir):
    img_array = []
    # Reading two videofile fg and bg
    bg = cv.VideoCapture(bgpath)
    fg = cv.VideoCapture(fgpath)

    success,bg_frame = bg.read()
    ret,fg_frame = fg.read()

    count = 0
    success = True

    print('Press q to exit')
    while success:
        success,bg_frame = bg.read()
        ret,fg_frame = fg.read()

        if not success or not ret:
            print("Can't receive frame. Exiting ...")
            break

        bg_frame = cv.resize(bg_frame,(640,360))
        fg_frame = cv.resize(fg_frame,(640,360))

        # converting bgr to hsv color space
        hsv = cv.cvtColor(bg_frame,cv.COLOR_BGR2HSV)

        # defining green color range and creating a mask
        lower_green = np.array([50,150,10])
        upper_green = np.array([70,255,255])
        mask1 = cv.inRange(hsv, lower_green, upper_green)

        # inverting the mask to remove detected green color
        mask2 = cv.bitwise_not(mask1)

        # bitwise anding bg_frame with inverted mask to remove green color from frame
        res1 = cv.bitwise_and(bg_frame,bg_frame,mask=mask2)

        # bitwise anding fg_frame with mask of green color detected to replace green c
```

```
        res2 = cv.bitwise_and(fg_frame, fg_frame, mask = mask1)

        # adding both bg and fg frames together
        output = cv.addWeighted(res1,1,res2,1,0)

        cv.imshow('frame',output)

        height, width, layers = output.shape
        size = (width,height)
        img_array.append(output)

        if cv.waitKey(1) & 0xFF == ord('q'):
            break

    vidoutObj = cv.VideoWriter(os.path.join(output_dir,'chroma_keying.avi'),cv.VideoWrit
    for i in range(len(img_array)):
        vidoutObj.write(img_array[i])
    vidoutObj.release()
    # Release everything if job is finished
    bg.release()
    fg.release()
    cv.destroyAllWindows()

if __name__ == '__main__':

    # Calling the function
    fgpath = "input_data\\question3\\fg.mp4"
    bgpath = "input_data\\question3\\bg.mp4"
    output_dir = "output_data\chroma_keying"
    chroma_keying(bgpath,fgpath,output_dir)
```

4 Face detection :

4.1 Read about the technique of face detection. Create a video of that integrates the face detected in frames using bounding box using this technique possibly with one video including yourselves.

Description : Detection of faces in frames of video using bounding box.

Solution **:** Face detection using Haar cascades is a machine learning based approach where a cascade function is trained with a set of input data. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc.

Input **:** Webcam captured video
Output **:**

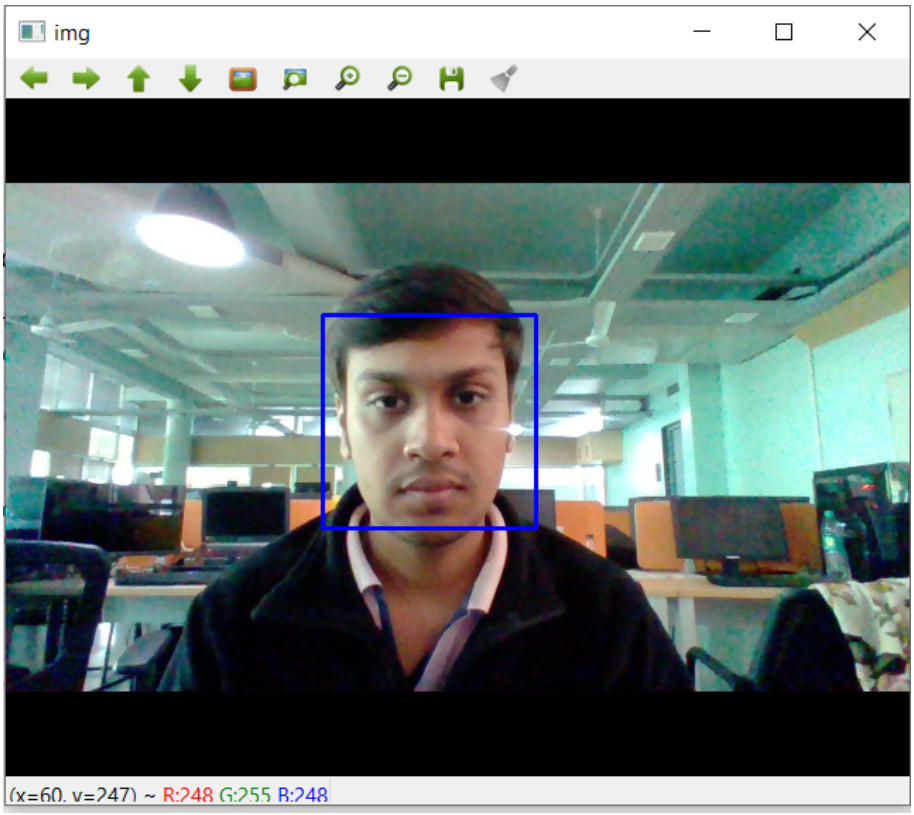

Figure 5: Face Detection

Code **:**

```
def Face_Detection():
    # Load the cascade
    face_cascade = cv.CascadeClassifier('haarcascade_frontalface_default.xml')

    # To capture video from webcam.
    capture = cv.VideoCapture(0)

    while True:
        # Read the frame
        _, img = capture.read()
```

```
        # Convert to grayscale
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        # Detect the faces
        faces = face_cascade.detectMultiScale(gray, 2, 4)
        # Draw the rectangle around each face
        for (x, y, w, h) in faces:
            cv.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
        # Display
        cv.imshow('img', img)
        # Stop if escape key is pressed
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    # Release the VideoCapture object
    capture.release()

if __name__ == '__main__':
    # Calling the function
    Face_Detection()
```

Learning **:**The OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30 to 50 of these stages or cascades, and it will only detect a face if all stages pass.

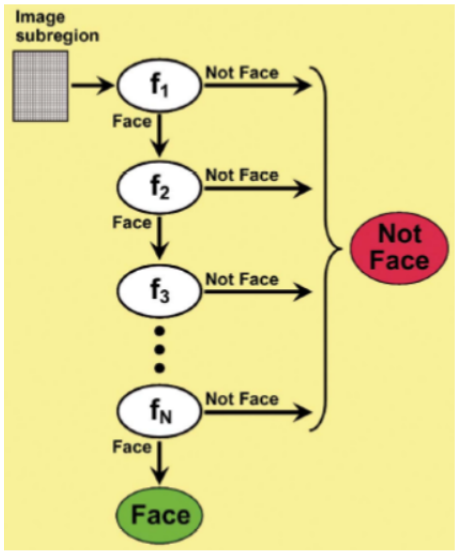

Figure 6: Face Detection using Haar Cascades