Computer Vision: Assignment 0

Utkarsh Upadhyay
2019101010

Task 1

Problem Statement

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.

Solution

Video →Images:
This is done using the user-defined function video_to_image(). We use cv2.VideoCapture to capture the given video, and use the 'set' function to update the current position of the video file (stored in CAP_PROP_POS_MSEC) by 'sec' each time. The 'read' function gives us the frame which we save using cv2.imwrite (each image is given a unique 5-digit id sequentially). The 'sec' variable is incremented by 'fr_dur' each time, as 'fr_dur' gives the time duration b/w 2 frames.

Images →Video:
This is done using the user-defined function image_to_video(). We first read all valid files in the directory specified by 'pathIn', and then sort them according to their 5-digit id's. Then we use cv2.imread to read the images and append the images to an empty list. Then we initialize the output video 'out' by the function cv2.VideoWriter. Here the 2nd argument, cv2.VideoWriter_fourcc(*'DIVX'), gives us the 4-character code of the codec (MPEG-4) used to compress the frames. The frames stored in the list are then appended to 'out'.

To control the frame rate (in fps) of output video, we can use the 'fps' variable. However, since we want to compare the fps b/w original and final video, we can use the 'FPS' variable to control the frame rate of output video w.r.t. the original video.

Challenges

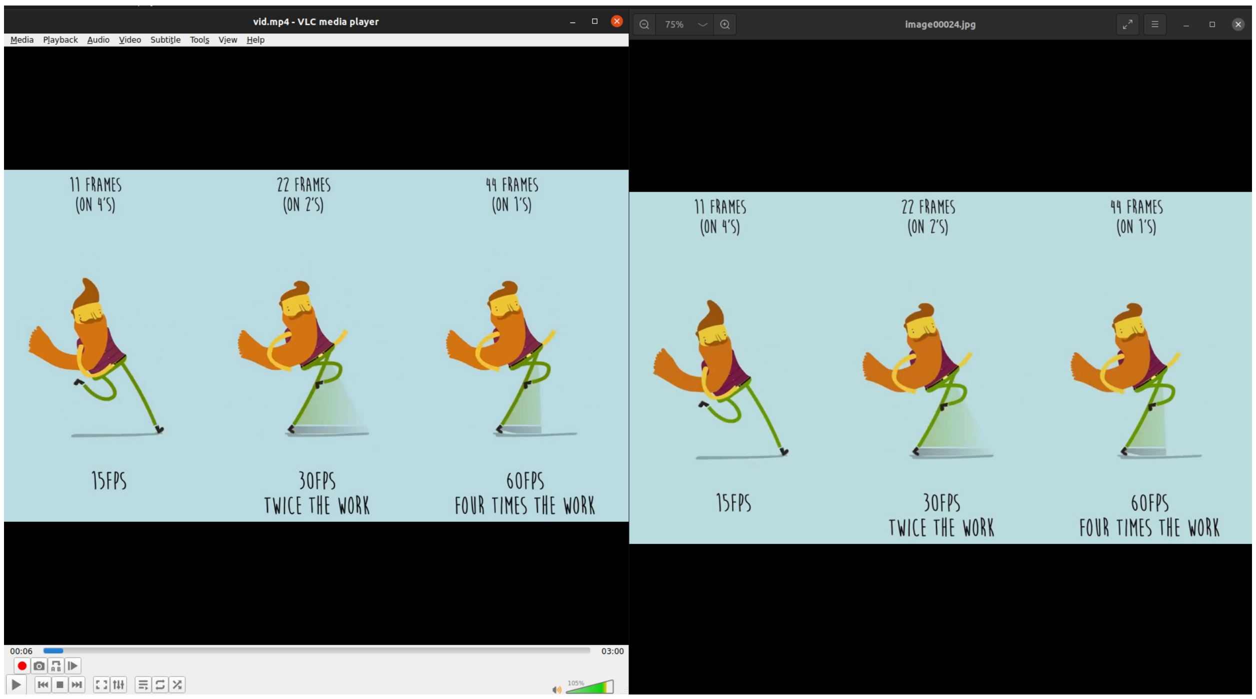
- Deciding the naming convention for the frames to make sorting easier.
- Selecting the extension of the output video file ('.avi') and matching it with the corresponding 'fourcc' (4-character code of codec used to compress the frames, 'DIVX').
- Hardware limitations.

Learnings

- How to control the frame rate during reading/writing a video.
- How to use in-built opencv functions like **VideoCapture**, **VideWriter**, **imread**, and **imwrite** to read videos, write videos, read images, and write images respectively.

Results

- Took a 'vid.mp4' video and converted it to its constituent frames at fps=4.



- Took the constituent frames to reconstruct the video (video.avi) at fps=40, or FPS=10 (w.r.t the original video).

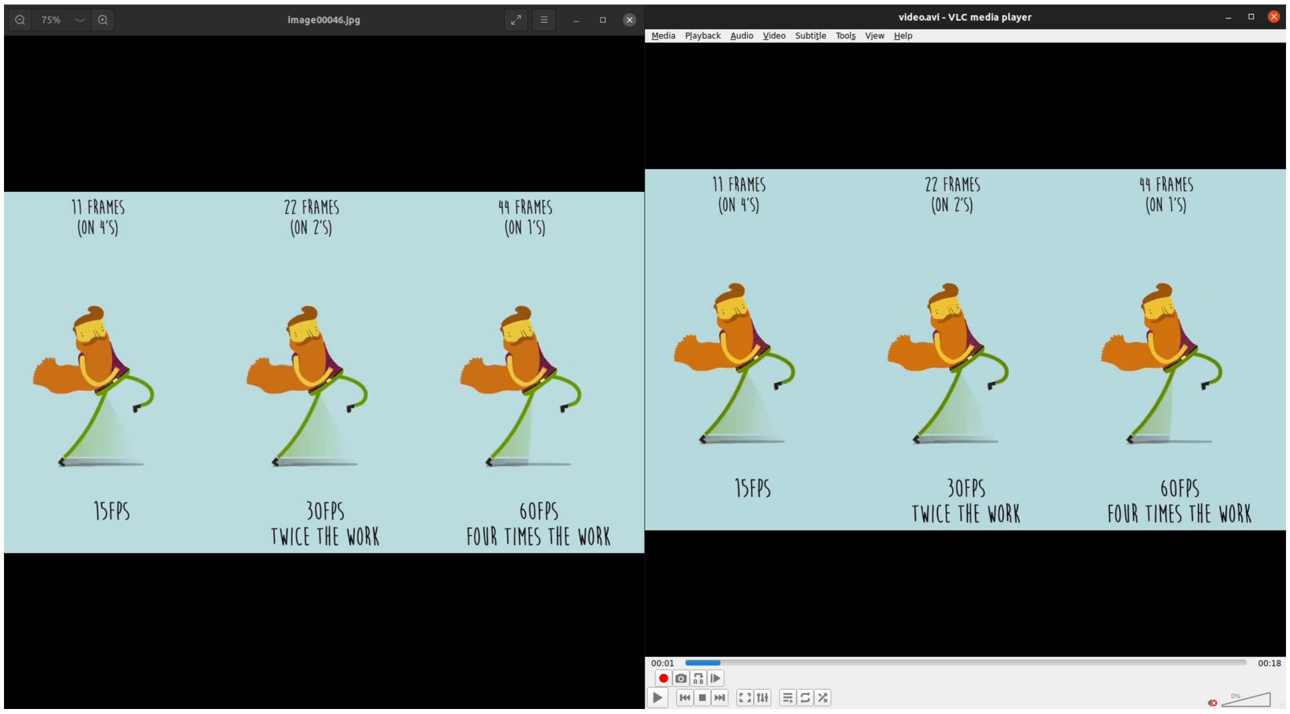


image00046.jpg
videoavi - VLC media player
Media Playback Audio Video Subtitle Tools View Help
11 FRAMES (ON 4'S)
22 FRAMES (ON 2'S)
44 FRAMES (ON 1'S)
15FPS
30FPS TWICE THE WORK
60FPS FOUR TIMES THE WORK
00:01
00:18

Code

```
import cv2
import os
from os.path import isfile, join, exists

FPS = 10
video = 'vid.mp4'

# Converting Video to Image

folderOut = './1/images/'
fps_im = 4
fr_dur = 1/fps_im

def video_to_image(video, folderOut, fr_dur):
```

```python
    if not exists(folderOut):
        os.makedirs(folderOut)

    vid = cv2.VideoCapture(video)
    sec = 0
    frame_ct = 0

    while(1):
        vid.set(cv2.CAP_PROP_POS_MSEC, sec*1000)
        ret, frame = vid.read()
        if ret == False:
            break
        sec += fr_dur
        sec = round(sec, 2)

        frame_ct += 1

        # saving the captured frames to disk
        img_name = 'image'+f'{frame_ct:05d}'+'.jpg'
        cv2.imwrite(folderOut+img_name, frame)


video_to_image(video, folderOut, fr_dur)

# Converting Image to Video

pathIn = './1/images/'
pathOut = './1/video.avi'
fps = FPS*fps_im


def image_to_video(pathIn, pathOut, fps):

    if not exists(pathIn):
        os.makedirs(pathIn)
    files = [f for f in os.listdir(pathIn) if isfile(join(pathIn, f))]
    # for sorting the file names properly
    files.sort(key=lambda x: x[5:10])

    frames = []
    for i in range(len(files)):
        filename = pathIn + files[i]
        frame = cv2.imread(filename)
        height, width, layers = frame.shape
        size = (width, height)
        frames.append(frame)

    # MPEG-4 codec (DIVX)
    out = cv2.VideoWriter(pathOut, cv2.VideoWriter_fourcc(*'DIVX'), fps,
size)
    for i in range(len(frames)):
        out.write(frames[i])
    out.release()


image_to_video(pathIn, pathOut, fps)
```

Task 2

Problem Statement

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.

Solution

We use the in-built opencv function cv2.VideoCapture(0) to capture the video from the built-in webcam. Then we continuously read frames from the webcam (while the webcam is opened), save the frames using cv2.imwrite, and display them in a separate window using cv2.imread and cv2.imshow. Additionally, if the user closes the frame window, we can detect the event using cv2.WIND_PROP_VISIBLE, and then close the webcam.

<u>**Challenges**</u>

- Deciding how to break from the continuous while loop which reads frames from the webcam.
- Checking how to detect if the user closes the frame window.

<u>**Learnings**</u>

- How to read frames from the webcam.
- How to detect a window closing event by user, in order to close the webcam.

<u>**Results**</u>

- Captured frames from the built-in webcam and displayed them in a separate window.

Code

```
import cv2
import numpy as np
import os
from os.path import exists

# Opens the inbuilt camera
cam = cv2.VideoCapture(0)
# Count number of frames captured
frame_ct = 0

folderOut = './2/'
if not exists(folderOut):
   os.makedirs(folderOut)

while(cam.isOpened()):
   ret, frame = cam.read()
   if ret == False:
       break
   frame_ct += 1

   # saving the captured frames to disk
   img_name = 'Frame'+str(frame_ct)+'.jpg'
   cv2.imwrite(folderOut+img_name, frame)

   # display the saved image in cv-window
   image = cv2.imread(folderOut+img_name)
   cv2.imshow('Frames', image)
   cv2.waitKey(1)

   # Break if user closes cv-window
   if cv2.getWindowProperty('Frames', cv2.WND_PROP_VISIBLE)<1:
       break

cam.release()
cv2.destroyAllWindows()
```

Task 3

Problem Statement

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.

Solution

Here, we first capture 2 videos, with one of them containing a green screen background, using cv2.VideoCapture(). Then we create an empty output video, with a fixed size and the same fps as the green screen video. Then we read a frame from the green screen video, resize it, and construct a mask based on the upper and lower limit of green pixels values that are to be considered as background. Then using bitwise and inverse, we extract the non-green pixels from the green frame, and append it on top of the normal video. This frame is then written in the output video file.

Experiments

- The upper and lower limit of [B,G,R] values for which we consider a pixel as green, was determined experimentally, based on the visual output.

Challenges

- Setting the output video’s frame rate equal to the fps of input green screen video.
- Constantly updating the upper and lower limit of [B,G,R] color values of pixels, based on different input videos, to correctly extract the foreground part of the image.
- Hardware limitations.

Learnings

- Understanding the concept of Chroma Keying with OpenCV using green screen.
- Understood the uses of matting (Chroma Keying is a special case for matting) in other areas like grabcut, etc.

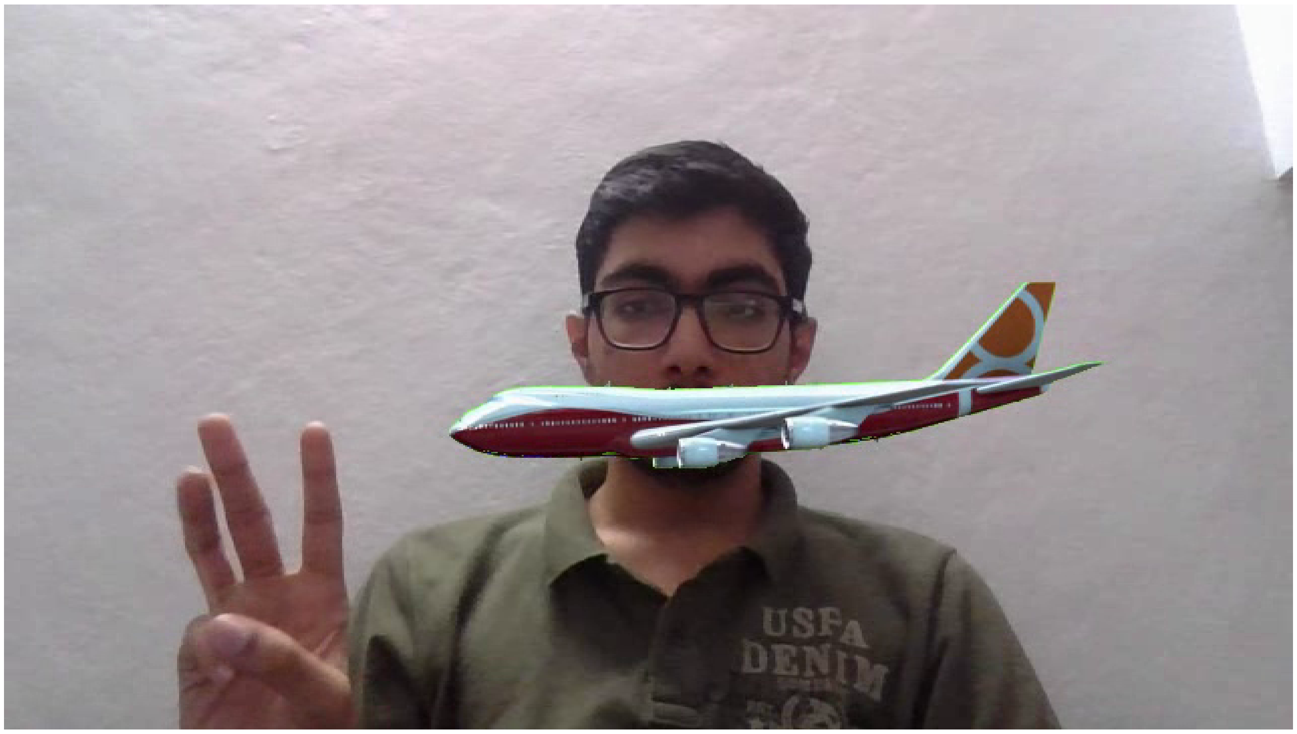
Results

- Successfully added a video with a green screen on top of a video of myself.

GREEN SCREEN VIDEO FRAME

NORMAL VIDEO FRAME

MERGED VIDEO FRAME



Code

```
import cv2
import numpy as np

# video_bg->with green screen
# video_fg->without green screen
video_gr = cv2.VideoCapture("green.mp4")
video_ngr = cv2.VideoCapture("/home/sthawk/Downloads/videoplayback.mp4")

while True:

   ret, frame = video_gr.read()
   if ret==False:
       break
   gr_video = cv2.resize(frame, (640, 480))
```

```
    ## [B value, G value, R value]
    lower_green = np.array([0, 150, 0])
    upper_green = np.array([100, 255, 100])
    # upper_green = np.array([104, 153, 70])

    # Creates a binary mask and filters the bg_frame
    # (white(255)->within range, black(0)->out of range)
    mask = cv2.inRange(gr_video, lower_green, upper_green)
    res = cv2.bitwise_and(gr_video, gr_video, mask=mask)

    f = gr_video - res

    ret, frame = video_ngr.read()
    if ret==False:
        break
    ngr_video = cv2.resize(frame, (640, 480))
    f = np.where(f == 0, ngr_video, f)

    cv2.imshow("mid1-res", res)
    cv2.imshow("video", gr_video)
    cv2.imshow("mask", f)
    if cv2.waitKey(25) == 27:
        break

video_gr.release()
video_ngr.release()
cv2.destroyAllWindows()
```