CV Assignment 0

M Krishna Praneet
2020113010

Link to Github Repo with all code :

> https://github.com/mkrishnapraneet/cv

Task 1 - Conversion between images and video

Problem :

Use opencv to convert a video to images (frames) and store them in a folder.

Also do the inverse, i.e. make a video by stitching together multiple images as frames.

Solution :

Video to images : Load the video using the VideoCapture function in opencv, and then obtain each frame using the read() function in a loop. Then use imwrite() method to save the current frame in the folder specified by the path passed as argument.

Images to Video : Sort the images in the correct order of appearance and then loop through each of them. Read each image using imread() and append the resulting object to the final video using .write() method.

Code

```
'''Video to Images'''

# capture video from path
cam = cv2.VideoCapture("./vid2img/game.mp4")

curFrame = 0
```

```
while(True):

    # read from frame
    retVal, frame = cam.read()
    if retVal:
        name = './vid2img/frame' + str(curFrame) + '.jpg'
        cv2.imwrite(name, frame)
        curFrame += 1
    else:
        # break if video is not left
        break

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

```
'''Images to Video'''
frameSize = (1920, 1080)

out = cv2.VideoWriter('img2vid/output_video.mp4',cv2.VideoWrite

image_files = sorted(glob.glob('./img2vid/img/frame*.jpg'), key=

for filename in image_files:
    img = cv2.imread(filename)
    out.write(img)

out.release()
```

Challenges and Experiments

One challenge I faced was to correctly exit the loop once all the frames in a video had been read. Initially, I got errors because of reading empty objects, which was resolved by checking if the object is valid, and exiting the loop if it is invalid.

Sample Outputs

Link to sample outputs : look at folders vid2img and img2vid

> https://iiitaphyd-my.sharepoint.com/:f:/g/personal/mulukutla_p_research_iiit_ac_in/Ep1zJKMhjl1Jkt89vU_z0i8ByuaeYXfPN0ungYkh983e5A?e=JIcYFW

Task 2 - Capture and Display from Webcam

Problem

Capture frames from a webcam connected to the computer and save them as images in a folder. Also display the video during capture.

Solution

Start a video capture using the VideoCapture() method in opencv. Pass it index ‘0’ to start recording from the webcam. Read each frame using a loop and show it using imshow() method. Also save a copy of the frame as a jpg file using the imwrite() method.

Code

```
# video capture object
vid = cv2.VideoCapture(0)
counter = 0
while(True):

    # Capture the video frame
    ret, frame = vid.read()

    # Display the resulting frame
    cv2.imshow('frame', frame)
```

```
    # save frame as jpg file
    cv2.imwrite('./webcam/frame' + str(counter) + '.jpg', frame)
    counter += 1

    # press 'q' to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

vid.release()
cv2.destroyAllWindows()
```

Challenges and Experiments

I faced the challenge of being unable to close the popup window in a clean fashion. The kernel would always crash after execution. After a little experimentation, I figured out that using waitkey() method in opencv, I can bind a key and exit when that key is pressed. Therefore I bound the Q key to quit and this solved my problem.

Sample Outputs

Link to sample outputs : look at folder webcam

> https://iiitaphyd-my.sharepoint.com/:f:/g/personal/mulukutla_p_research_iiit_ac_in/Ep1zJKMhjI1Jkt89vU_z0i8ByuaeYXfPN0ungYkh983e5A?e=JIcYFW

Task 3 - Chroma Keying

Problem

The task was to use chroma keying to make a composition of two different videos into one video by replacing something akin to a ‘green screen’ with another video.

Solution

First, I loaded both the videos and read each frame one by one using a while loop. Then I resized them to be of the same resolution and defined a range of green to look for in the first video (so as to replace it with another video). So in iteration, I have two images. I mask all the pixels of the first image that fall in the aforementioned range. Then I use bitwise and to identify and replace these pixels with the pixels of the second video. Then using imshow(), I display the merged images. I also save the resultant video in a folder.

Code

```
video1 = cv2.VideoCapture("./greenPC.mp4")
video2 = cv2.VideoCapture("./webcam/compiled_video.mp4")
out = cv2.VideoWriter('task3_output.mp4',cv2.VideoWriter_fourcc(

while True:

    ret1, frame1 = video1.read()
    ret2, frame2 = video2.read()

    if not (ret1 and ret2):
        break

    frame1 = cv2.resize(frame1, (640, 480))
    image = cv2.resize(frame2, (640, 480))

    u_green = np.array([150, 230, 150])
    l_green = np.array([0, 170, 0])

    mask = cv2.inRange(frame1, l_green, u_green)
    res = cv2.bitwise_and(frame1, frame1, mask = mask)

    f = frame1 - res
    f = np.where(f == 0, image, f)

    cv2.imshow("video1", frame1)
    cv2.imshow("mask", f)
```

```
    out.write(f)

    if cv2.waitKey(20) == 27:
        break

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

cv2.destroyAllWindows()
video1.release()
video2.release()
out.release()
```

Challenges and Experiments

I experimented a lot with the range of the pixels to mask, so as to get a clean merge between videos. I tried different techniques like wearing a green shirt etc, which worked, but were still imperfect. Thus, I used a video from online which had a green screen, and merged a video recorded on my webcam with it. I finally settled with the range of masked pixels to be from [0,170.0] to [150,230,150]. This range gave the best results.

Some of my learnings include the proper way to exit the loop when dealing with unknown sizes. I also learnt that the lighting in an image matters a lot, and can be a huge pain when trying to decide on the range of RGB values to mask.

Sample Outputs

Link to sample outputs : look at the file named task3_output.mp4 for the resulting merged video.

https://iiitaphyd-my.sharepoint.com/:f:/g/personal/mulukutla_p_research_iiit_ac_in/Ep1zJKMhjI1Jkt89vU_z0i8ByuaeYXfPN0ungYkh983e5A?e=JIcYFW