BLG453E
Homework-1

09.10.2019

Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr

René Goscinny & Albert Uderzo, Asterix and The Big Fight

- You should write all your code in Python language.
- Cheating is highly discouraged.
- Ninova only stores files under 25 MB. If you could not upload your results, you can share them with me via Dropbox, or send me private YouTube video links for each part's results.
- In this homework, we will work on high resolution images. But you are free to resize them if your system is not capable to work on them.

1 - Part 1: "I am feeling blue" (20 pts.)

In this homework, we will use image sequences from DAVIS Challenge[1] which is a dataset originally created for semantic segmentation. You can download the **"TrainVal"** set

[1] https://davischallenge.org

Table 1: Some example images from DAVIS challenge and their segmentation maps

which contains image sets and their semantic annotations[2]. I also selected and uploaded some of them under my website[3]. Some of the image sequences and their segmentation maps are given in Table 1.**You must select and obtain video results for at least 3 image sequences for each part of the homework.**

This part of the homework focuses on an introductory problem especially for students who are not comfortable with numpy and OpenCV libraries.[4] In this part, we will

- Read images and image segmentation maps from the specified folders.
- Mask some color channels of an object using the segmentation map. You can select one or more color channels to mask.
- Write the obtained frames as a video.

To read an image in OpenCV, you can use $cv2.imread$ function. As shown in the code below, for segmentation maps, you should use $cv2.IMREAD\_GRAYSCALE$ flag. Since the segmentation maps are saved as indexed images (images that contain ids for pixels, not color values), reading without that flag makes it more difficult to use these maps. The following code shows how to read a sequence of images and do some operations on them. You can start from this skeleton code.

```
for i in range(len(all_images)):
    image = cv2.imread(main_img_dir +all_images[i])
    # Image is a numpy array with shape (800, 1920, 3)
    seg = cv2.imread(main_seg_dir +all_images[i].split('.')[0]+'.png', cv2.
    IMREAD_GRAYSCALE)
    #Seg is the segmentation map of the image with shape (800, 1920)
    seg = (seg == 38)
    #I know that seg has maps for objects with IDs 38 and 75. Thus I
    changed it into a binary map for 38.
```

[2] https://data.vision.ee.ethz.ch/csergi/share/davis/DAVIS-2017-Unsupervised-trainval-Full-Resolution.zip

[3] http://web.itu.edu.tr/sahinyu/DAVIS-JPEGImages.zip

[4] You can use the following sites to learn more about numpy (https://docs.scipy.org/doc/numpy/dev/) and OpenCV (https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html)

```
    image_without_the_guy = ##Show your work here
    image_of_the_guy = ##Show your work here

    image_of_the_guy[:,:,1:3] = ##Show your work here. I decreased the
    values of red and green channels by 75%.

    image = (image_without_the_guy + image_of_the_guy)


    #cv2.imwrite('x.png', image)
    #You can use imwrite function to check the results.
```

To save the images as a video you can use moviepy library. I personally prefer this simple but effective library rather than OpenCV's video writing operations (You are also free to use OpenCV's operations.). You can create an ImageSequenceClip object and write a video as given in the following lines. (**Note:** While OpenCV uses BGR channel order by default, moviepy uses RGB.)

```
images_list = []

##Add the processed images to the list.

clip = ImageSequenceClip(images_list, fps=25)
clip.write_videofile('part1_video.mp4',codec='mpeg4')
```

You can find an example output frame at Figure 1.

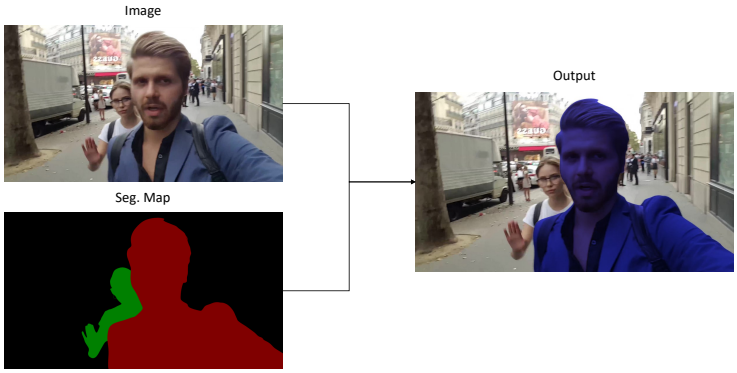

Figure 1: Task 1

2 - Part 2: Histogram matching (30 pts.)

In this part of the homework you will do histogram matching between video frames and a target image. To do this, you have two options:

- Obtain a histogram for each video frame and use the average of them.
- Obtain a huge image which is a concatenation of all images and use this image's histogram.

You can select the bin count according to the results. You are also free to select the target image. An example is shown in Figure 2.

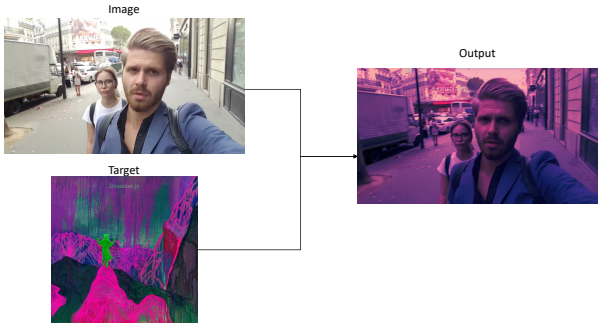

Figure 2: Task 2

3 - Part 3: Histogram matching from segmentation maps (50 pts.)

Finally, you will use multiple target images to assign different targets to different objects. An example result is shown in Figure 3. **For each object, use only the color values from the selected object. Think of each object separately.**

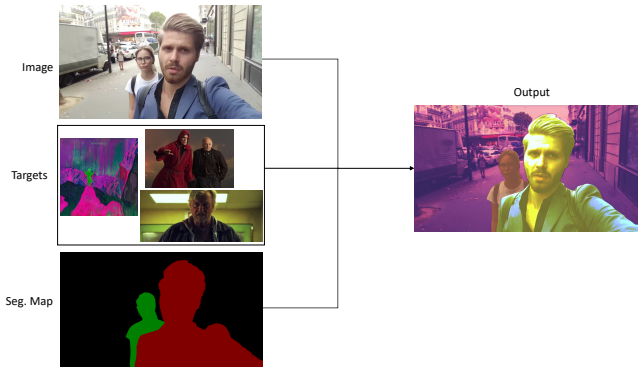

Figure 3: Task 3