BLG453E
Homework-1

09.11.2020

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

On the floors of Tokyo
A-down in London town's a go go
A-with the record selection,
And the mirror's reflection,
I'm a dancin' with myself

Billy Idol, Dancing With Myself

- You should write all your code in Python language.
- Cheating is highly discouraged. If you are planning to use different libraries or functions, please ask me about it.
- Ninova only stores files under 20 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.

1 - Part 1: Dancing alone (20 pts.)

In this homework, we will use an image sequence from the famous dancing cat video[1] to get familiar with OpenCV, NumPy and MoviePy libraries. Some frames which will be used are given in Figure 1.

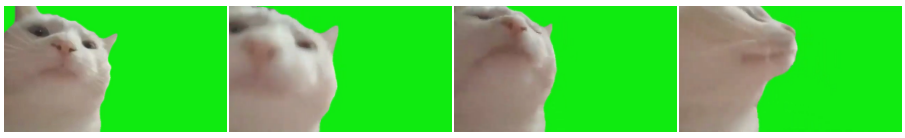

Table 1: Some example greenscreen cat frames

[1] https://www.youtube.com/watch?v=-_c1dLgTjoo

The first part of the homework focuses on an introductory problem especially for students who are not comfortable with numpy and OpenCV libraries.[2]

In this part, we will

- Read a background image and some cat images from the specified folders.
- Place the cat images onto the background image.
- 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, you can use $cv2.imshow$ to show the image inside a window and to give a close condition to the window, you can use $cv2.waitKey(0)$ which unpauses the program by a keyboard click.

```
import numpy as np
import os
import cv2
import moviepy.editor as mpy


background = cv2.imread('Malibu.jpg')
cv2.imshow('Background Image Window', background)
cv2.waitKey(0)
```

First thing we should do here is to resize the background image according to our cat frames. As our cat frames have a shape of $(360, 640, 3)$, we should resize height of the background image according to this.

```
background_height = background.shape[0]
background_width = background.shape[1]
ratio = 360/background_height

background = cv2.resize(background, (int(background_width*ratio), 360))

print(background.shape)
```

Then, we should read every image inside the cat folder and place it onto the background image. An example placement is given below.

```
image = cv2.imread(main_dir + '/cat_5.png')

image_g_channel = image[:,:,1]#Green channel of the image
image_r_channel = image[:,:,0]#Red channel of the image

foreground = np.logical_or(image_g_channel < 180, image_r_channel > 150)#
    The pixels having cat image has green values lesser than 180 and red
    values greater than 150.
```

[2] 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)

```
7
nonzero_x, nonzero_y = np.nonzero(foreground) #The 'foreground' variable
    here is only a True-False map with the same size. Using np.nonzero
    function we can find the locations of True values.
9
nonzero_cat_values = image[nonzero_x, nonzero_y,:] # A matrix of shape
    (..., 3) containing the pixel values belonging to the cat part.
11
new_frame = background.copy()
13
new_frame[nonzero_x, nonzero_y,:] = nonzero_cat_values #To the previously
    obtained indices, the cat part is placed.
15
new_frame = new_frame[:,:,[2,1,0]] #The frame here is currently in RGB
    order. However, the moviepy library defaultly uses BGR order. Thus, it
    may be good to reverse the channels.
```

After obtaining the frames like that, to use **moviepy.editor.ImageSequenceClip**, we should append the frames to a list. Then, using *set_audio* function, we can place a background music and using *write_videofile* we can save this composition as a video.

```
images_list = []
2 #
# Some code here
4 #
    images_list.append(new_frame)
6
clip = mpy.ImageSequenceClip(images_list, fps = 25)
8 audio = mpy.AudioFileClip('selfcontrol_part.wav').set_duration(clip.
    duration)
clip = clip.set_audio(audioclip=audio)
10 clip.write_videofile( 'part1_video.mp4' , codec='libx264')
```

If everything goes well, you will obtain a similar video to mine[3]. A frame from the video is given in Figure 1.

Figure 1: Placing the dancing cat on the left

[3] https://web.itu.edu.tr/sahinyu/part1_video.mp4

2 - Part 2: Dancing with myself (10 pts.)

As mentioned in the song, make the cat dance with his reflection on the mirror. An example frame is given below.

Figure 2: Task 2: Two cats on the dancefloor

3 - Part 3: Dancing with my shadow (10 pts.)

Using a previously defined transform mapping (e.g. window/level transform), make the cat on the right darker. Try different parameters and observe the differences.

4 - Part 3: Dancing with my friend (30 pts.)

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

- Obtain a histogram for **cat parts** of each frame and use the average of them. This will give us **average cat histogram**. Obtain histogram for each channel.
- Make histogram matching for the new cat using a target image.

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 3.

Figure 3: Task 3: Histogram matching result

5 - Part 4: Disco dancing (30 pts.)

In this part of the homework, we will obtain more flashing images for both cats to have a disco effect. To do this, for every frame,

- For the cat on the left,
  - calculate the cat's histogram
  - randomly perturb the cat's own histogram (*Hint:* You can use random noise.)
  - Make histogram matching to this perturbed histogram
- For the cat on the right,
  - Calculate the cat's own histogram
  - Randomly perturb the target histogram
  - Make histogram matching to this perturbed histogram