BLG453E
Homework-1

08.11.2021

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 with myself (20 pts.)

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

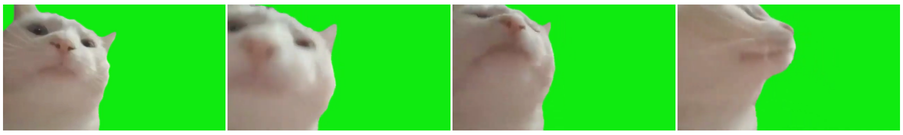

Table 1: Some example greenscreen cat frames

The first part of the homework focuses on an introductory problem especially for students who are not comfortable with numpy and OpenCV libraries.[1]

In this part, we will

- Read a background image and some cat images from the specified folders.
- Place the cat images onto the left of background image.
- Flip the cat images and place them onto right of 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.

```python
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.

```python
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.

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

foreground = np.logical_or(image[:,:,1] < 180, image[:,:,0] > 150)#The pixels having cat image.

```

[1] 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)

```
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.

nonzero_cat_values = image[nonzero_x, nonzero_y,:] # A matrix of shape (..., 3) containing the pixel values belonging to the cat part.

new_frame = background.copy()

new_frame[nonzero_x, nonzero_y,:] = nonzero_cat_values #To the previously obtained indices, the cat part is placed.

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 = []
#
# Some code here
#
    images_list.append(new_frame)

clip = mpy.ImageSequenceClip(images_list, fps = 25)
audio = mpy.AudioFileClip('selfcontrol_part.wav').set_duration(clip.duration)
clip = clip.set_audio(audioclip=audio)
clip.write_videofile( 'part1_video.mp4' , codec='libx264')
```

If everything goes well, you will obtain a similar video to mine[2]. A frame from the video is given in Figure 1.

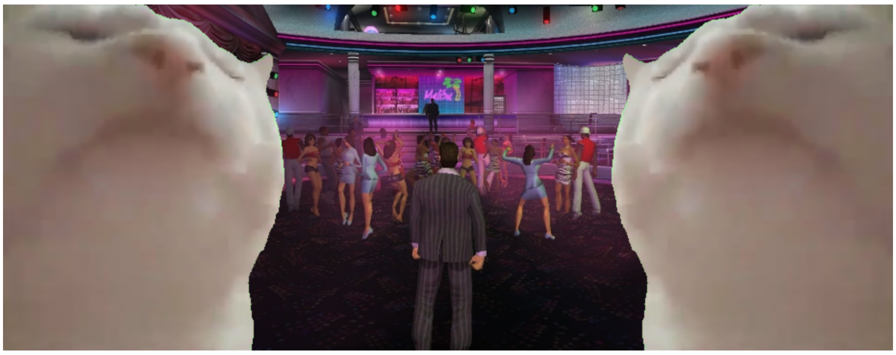

Figure 1: Two cats on the dancefloor

[2] https://web.itu.edu.tr/sahinyu/blg453e_hw1_part1.mp4

2 - Part 2: 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 2.

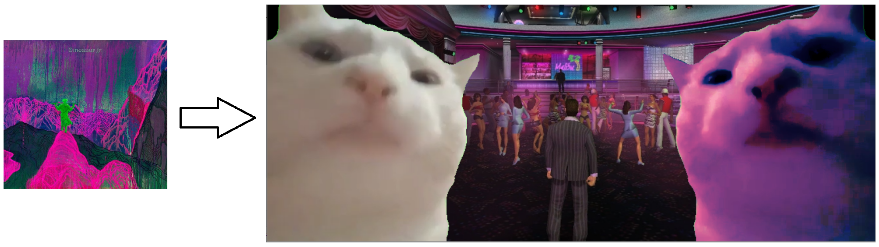

Figure 2: Histogram matching result

3 - Part 3: Another cat, from 90s (40 pts.)

The first time I've listened Myslovitz's self-titled album, every song in it stuck in my head for at least a week. I was just like the cat in this video[3]. In the first task of the homework, you will create a video for your favourite album. A frame of the target video is given in Figure 3.

- Create an 572x322 pixels sized empty image.
- For every frame of the video I extracted the corner points of each plane. Use your favourite album cover to change the video. For each plane's correspondences, create transform matrices (Do not use OpenCV's built-in functions. Find coefficients using $q = M^{-1}A$. To read the album corners, you can use the sample code with with the homework document.
- Warp the album covers using matrix multiplication and place onto empty image.
- Complete the video by adding the cat. Decide which planes are behind or front of the cat considering areas of the planes.

[3] https://web.itu.edu.tr/sahinyu/hw2_cat.mp4

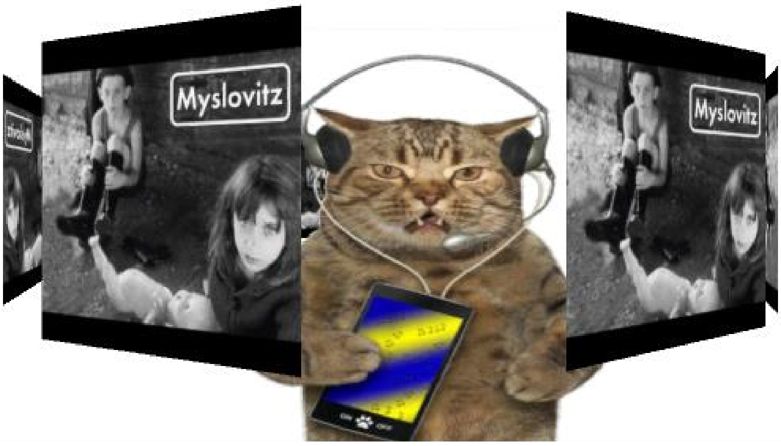

Figure 3: The cat is wishing to get rid of the album, but it is impossible.

4 - Part 4: Center of rotation (10 pts.)

For the album cover you used in the previous part, rotate it 60 degrees clockwise using the image center as the center of rotation. Then, use the top left corner of the image as the center and repeat the procedure. Compare the two outputs.

5 - Bonus: Data Augmentation (20 pts.)

For many machine learning tasks, using the train set as-it-is is not sufficient for a given method. Thus, the train data can be augmented for a better result. Both pointwise image processing and image transformations can be used as data augmentation for machine learning procedures.

In this part, the data you will use is from ICCV VIPriors Image Classification Challenge (2021)[4]. In this challenge, the main focus was learning from few data. Since you are not familiar with neural network training at this point, you will predict the classes of test images on a trained neural network structure as follows:

- For the train and test images, obtain latent features from the network.
- For each test image, use k-NN classification on the feature space.

You can find the feature extractor network and the dataset using this link[5]. Then, using different data augmentation methods[6], create new data. Save these data and then,

[4] https://competitions.codalab.org/competitions/33214

[5] https://www.dropbox.com/s/856mb0pr5f7e7vl/image-classification.zip?dl=0

[6] **Hint:** Search about imgaug and randaug

obtain latent features from the network again. Using these new features, try to increase the accuracy of your classification. Submit and evaluate your results on Kaggle. **Kaggle link will be announced soon.**

The awards for the competition winners are as follows:

- 1^{st} **place:** Digital Image Processing, Global Edition, by Rafael C. Gonzalez and Richard E. Woods
- 2^{nd} **place:** Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications, by Ian Pointer.
- $3^{rd} - 4^{th}$ **places:** A small souvenir

”

The students with top 10 results will make a small demo to me. Do not delete the data you created!

Good luck!