BLG453E
Homework-2

30.11.2021

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

- 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: Salt & Pepper

Bugs: Hey Sam, pass the salt please.
Sam: Salt? GET IT YOURSELF!
Bugs Bunny: Uh oh, that'll cost you about...
Sam: Salt? Why didn't you say so. Here's your salt, Bunny, I hope you like it. Ooh that rackin' frackin'...
Bugs: The pepper please.
Sam: PEPPER! WE... Uh, yeah the pepper. Coming right up.

From Hare To Heir (1960)

1.1.(15 pts): Preprocessing

In this part, we will work on the video "*shapes_video.mp4*". For every frame of the video, a new card having a specific shape (star, square, pentagon) appears. However, each frame has a salt & pepper noise as given in Figure 1.

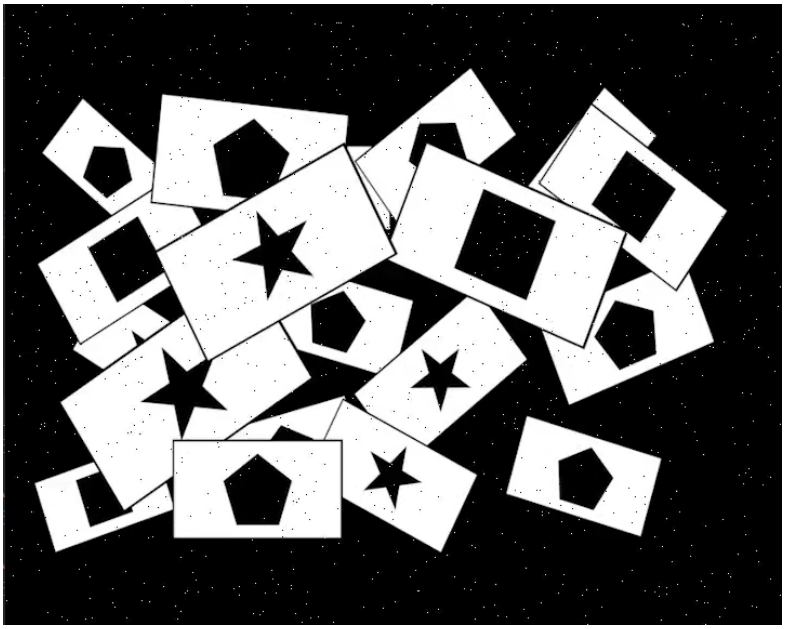


Figure 1: A frame from the video

Filter each frame to avoid noise and save the video as a new file. To read the video frame by frame, you can use the following lines.

```
import moviepy.video.io.VideoFileClip as mpy
import cv2

vid = mpy.VideoFileClip('shapes_video.avi')

frame_count = vid.reader.nframes
video_fps = vid.fps

for i in range(frame_count):
    frame = vid.get_frame(i*1.0/video_fps)
```

1.2.(15 pts): Counting the shapes

Implement Minimum Eigenvalue corner detector on the frames to obtain the corners of the shapes. Then, count the exact counts of triangle, star and pentagon cards. (**Hint:** You can do background subtraction between frames.)

2 - Part 2: I am nobody[1]

I'm Nobody! Who are you?
Are you – Nobody – too?
Then there's a pair of us!
Don't tell! they'd advertise – you know!

How dreary – to be – Somebody!
How public – like a Frog –
To tell one's name – the livelong June –
To an admiring Bog!

Emily Dickinson (1891)

2.1.(10 pts): StyleGAN

A Generative Adversarial Network[2] is a network which generates an output depending on an random or conditional input. The term "adversarial" comes from the training process, since two networks are used: Generator and Discriminator. While Generator network creates an output it also tries to fool the Discriminative network which has a task to differentiate real and fake images. A simple scheme for the training process is given in Figure 2.

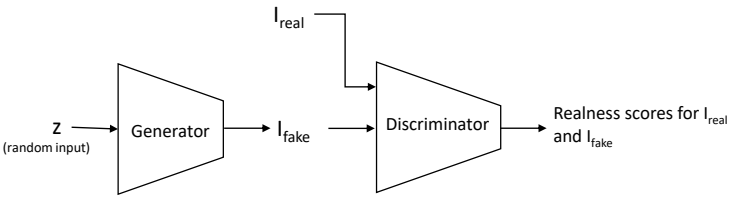


Figure 2: Generative and Discriminative networks in a GAN.

In this part of the homework, we will use StyleGANv3[3] to generate realistic face images. To do this, we will use the pretrained network with a simple script as given in the following code.

[1] I sincerely thank my friend V. Bugra Yesilkaynak for his help to prepare this part of the homework.
[2] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
[3] Karras, Tero, et al. "Alias-free generative adversarial networks." arXiv preprint arXiv:2106.12423 (2021)

```python
import pickle
import matplotlib.pyplot as plt
import torch
import cv2
import numpy as np
with open("stylegan3-t-ffhq-1024x1024.pkl", "rb") as f:
  a = pickle.load(f)
#Unpack the GAN network downloaded from the official NVIDIA website.

gan = a["G_ema"]
#StyleGAN contains both the Generator and the Discriminator. We only need the Generator to generate the images.
gan.eval()
#In PyTorch, every network has to be set as 'train' or 'eval'. To evaluate an input via the network, we should set it as eval.
for param in gan.parameters():
  param.requires_grad = False
#When training the network, the parameter change is done via the gradients. Setting that none of the parameters need gradients will speed up our process.

z = torch.randn(1, 512)
# The StyleGAN generator takes a vector of size (512) to generate an image.
img = gan(z, 0).numpy().squeeze()
#Obtain the image from the generator.

img = np.transpose(img, (1,2,0))
#Initially, the output of the network is like (RGB Channel, Width, Height). We should change the order of the channels to make it look like an OpenCV image.

img[img>1] = 1
img[img<-1] = -1
img = 255*(img + 1) / 2
#The network outputs images with float values, centered at zero. Thus, we should make the values between [0-255].

cv2.imwrite('test.png', img[:,:,[2,1,0]])
```

Create two random face images using StyleGAN generator. To do this, you should download *stylegan3-t-ffhq-1024x1024.pkl* from NVIDIA's official page[4]. The model here creates 1024x1024 sized images. If your system is not sufficient, you can also use other models from the same site for 256x256 outputs.

2.1.(10 pts): StyleGAN Animation

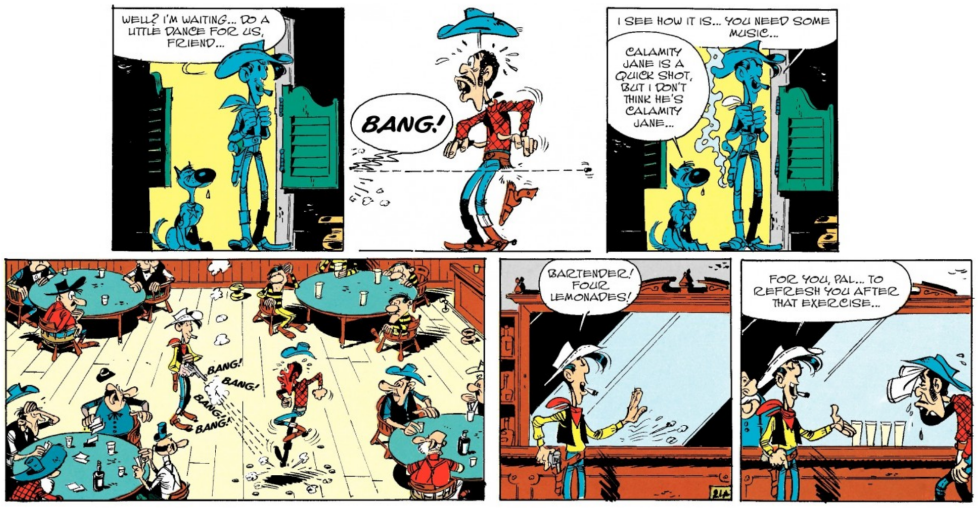
In the previous part, you have created two face images using two 512 sized vectors. Here we will create an animation of creating the second face by changing the first face. To do this, start by creating the first face and then slowly changing the face vector step-by-

[4]https://catalog.ngc.nvidia.com/orgs/nvidia/teams/research/models/stylegan3/files

Figure 3: Two face images created using StyleGAN.

step by a small amount, obtain the second face. Some examples can be found under my website[5].

3 - Part 3: Saturday Night Filter



Lucky Luke : The Daltons Redeem Themselves, Morris & Goscinny (1965)

Mutsuz insan yoktur, dans etmeyen insan vardır.

Tekin Abi, Tekin Abi ile Dans Saati (1980s)

For this part of the homework, I prepared a small dance game using Unity3D, in which we will make a 3D character dance according to the given shape inputs. You can download/reach the game via the following links:

[5] https://web.itu.edu.tr/sahinyu/face1_video.mp4, https://web.itu.edu.tr/sahinyu/face2_video.mp4, https://web.itu.edu.tr/sahinyu/face3_video.mp4

- **Windows Executable:** https://web.itu.edu.tr/sahinyu/saturday_night_filter_pc.zip

- **Online WebGL Game:** https://web.itu.edu.tr/sahinyu/saturday_night_filter_web/

The main page of the game is as given in Figure 4. We will be working on two songs: Vabank and Shame. Before selecting one of the songs, ensure that you entered your student ID in the textbox.

Figure 4: Main interface

For this homework, you will benefit from pyautogui library which is used to simulate mouse and keyboard interactions with Python. The example script given below first takes a screenshot of the game, then clicks random buttons.

```
import pyautogui
import time

time.sleep(5)
#In this 5 seconds you should switch to game screen to transfer the
    simulated keyboard inputs to the game.

myScreenshot = pyautogui.screenshot()
myScreenshot.save('test.png')
# An example screenshot is obtained. We will work on screenshots like this
    for this homework.

pyautogui.keyDown('shift')

pyautogui.keyDown('w')
time.sleep(1)
```

```
15 pyautogui.keyUp('w')

17 pyautogui.keyUp('shift')

19 #pyautogui.keyUp and pyautogui.keyDown functions are used to simulate
       holding a button. For simple presses, pyautogui.press can be used.
```

Clicking "Vabank" or "Shame" buttons will direct you to a game page as given in Figure 5. You can press ESC to go back to the main screen.

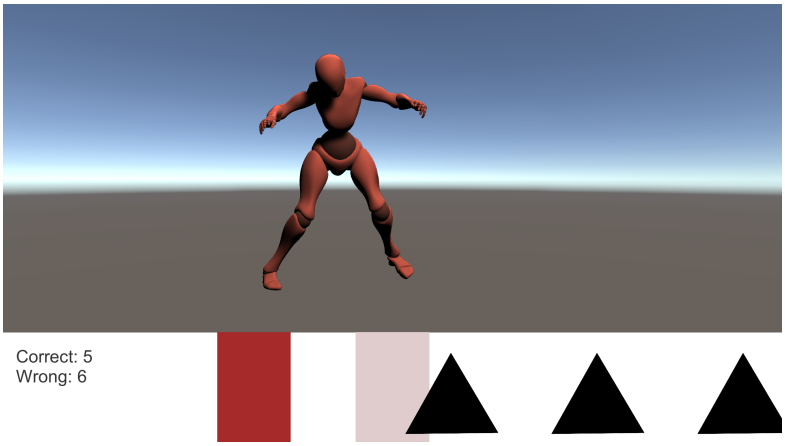


Figure 5: A game page

The game pages consist of our dancing 3D model and a 2D canvas at the bottom showing the shapes on a timeline. According to the dance, some shapes are flown from right to left. The player should press the corresponding button on time to obtain a point.

3.1.(50 pts): Better Than Jackson

Using your findings in the first part, write a script which beats the game. For this part, you are free to use every OpenCV function.

For each song, according to your scores and student ID, a small key will appear on the screen. **Do not forget to report these keys for each song. Also, report your screen resolution.**