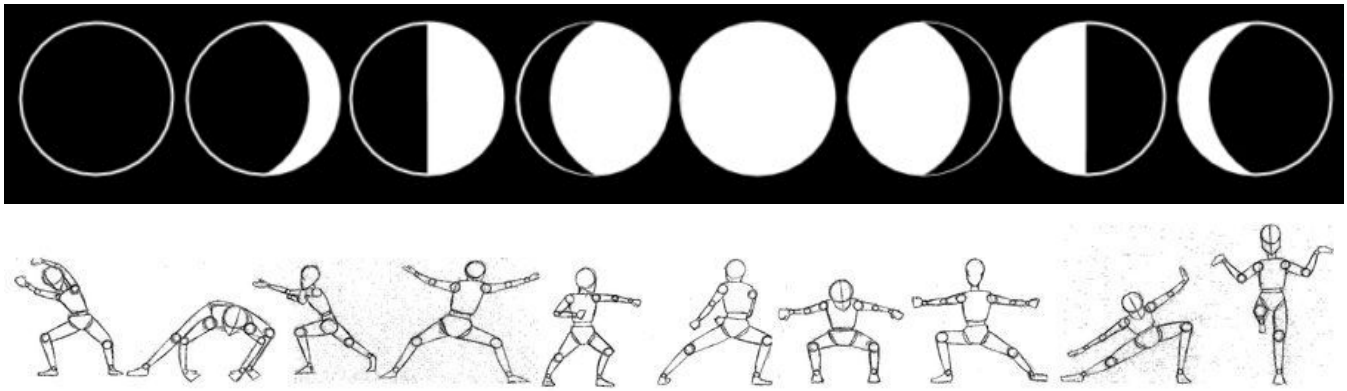
Machine Learning for the Arts
UCSD SPRING 2019
FINAL PROJECT

Phases



Roy Jara
Elliott Lao

DESCRIPTION

Concept:

The most fundamental starting idea for the development of this project was the desire to have an interaction between the viewer and the artwork. We chose video generation because we are aware that it is more attractive to the eye compared to still images or audio. We considered the possibility of a real-time interaction between audience and artwork but decided to avoid this due to the computational constraints such systems present. And so, we discarded the idea of processing the input video frames into an output. Instead we decided to use the input from the user as a controller by capturing the different parts of the body and tracking their position within the frame. This is like a visual version of a theremin (which alters pitch and volume depending on hand placement in air on top of it. We also wanted to revisit one of the visual generation tools explored in class and realized that if we generated a two dimensional latent space, we could use the body movement to navigate this space.

Finally, the conceptualization of the idea led us to want to make a project in which the output is a system that uses body movement as the input and generates a visual output that varies accordingly with that input. We didn't want to bound it to any specific visual content and leave it open as a tool for other people to creatively generate content this way. In real time installations, the system could be used for a combination between dance, projection mapping, audio generation systems for a combined analog and digital performance.

Technique:

Face Generation:

For presentation purposes we selected StyleGAN to generate a latent space of synthesized faces. The StyleGAN model works by first mapping the input image into an intermediate latent space. This space is then used to control the generator during the convolution layers of the image generator part of the system. These controls adjust how much of the style of the learned image the generator receives. For instance, if we generate faces, we can control how much the image synthesized from a given input has features of the style image. In faces, the "style" affects visible features such as skin tone, face shape, face position, facial features and geometry, hair and even the background of the image. Therefore our process was to have four actual images of real people, and then use the StyleGAN to synthesize images that very gradually include features of the other faces, as we move around the 2D grid.

Specifically for our project, we used a pretrained model of StyleGAN on the Flickr-Faces-HQ dataset(FFHQ).[1] This provided an easy way for us to generate high quality, realistic portrait images without having to train a model by ourselves. Our 2D grid was generated through modifying and combining two functions/feature highlights (*draw_truncation_trick_figure* and *draw_style_mixing_figure*) provided by the StyleGAN authors and referenced in the StyleGAN paper.[2] *draw_truncation_trick_figure* was used to create an interpolation between two pictures based upon a certain seed and generate a set of interpolated images. We then fed these two sets of interpolated images into *draw_style_mixing_figure*, this allowed us to create a grid of images that gradually incorporated features from the different seeds along the 2D grid in a row-column fashion, with the four seed images being on the corners.

Figure 1. - Latent Space of Synthesized Faces

Tf-openpose[3]:

To obtain user input, we used an implementation of Carnegie Mellon University's Open Pose research called tf-openpose. The implementation's main goal was to provide a faster and more efficient way to generate very similar results to the original code from CMU's research, allowing the algorithm to perform moderately fine even in less powerful systems, such as a Macbook pro (with no graphics card). The algorithm itself works by using a convolutional neural network to detect the number of people in the image as well as their respective body parts, locating them within the frame and plotting them in real-time using python plotting packages such as matplotlib.

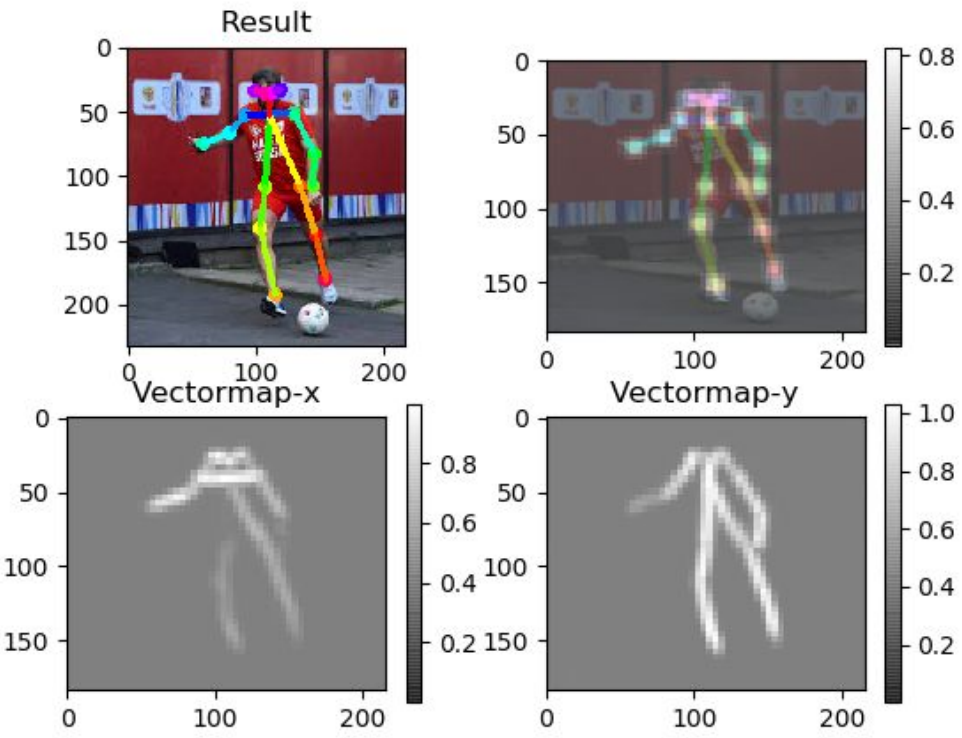


Figure 2. - Example plot from tf-openpose body pose estimation

Due to the ease of reusability of provided example code and documentation we were able to make alterations to these scripts in order to get them to process all of the frames of a given directory and instead of plotting them using matplotlib, we just saved the (x,y) coordinate pair in to a text file with the name of the body part. Therefore, we are able to retrieve the movements for every body part recognized by the model.

Mapping:

Lastly, we generated an output video by parsing the text file generated after processing all the frames of the video using tf-open pose. We equalized the values to range of values of the limb being tracked, mapped them to the pregenerated latent space and compiled both the input and output frames into one video using openCV.

Process:

Originally we had wanted to use the tf-openpose model to control a range of human expressions, where by moving an arm would correspond to changes in facial expression, such as from happy to sad. To accomplish this we originally wanted to create a dataset of one of our faces and feed it through to a DCGAN to generate the different potential frames of our output. However, looking at the implementation of DCGAN that we could find, we thought that the low output resolutions (max. 256x256) of the models that we could train would make for a less visually appealing output. So instead we opted for using StyleGAN and its feature mixing abilities to generate a simulated 2-D interpolation between four base faces. We thought that the

high resolution and realism of StyleGAN's outputs would make for a more visually appealing final product.

Figure 3. - Sample images representative of a DCGAN output from DCGAN-tensorflow[4]

Furthermore, we had originally wanted to use Open Pose in real time but we had found that the technical and hardware requirements were too demanding for both our computers and too challenging to implement on Datahub in the given timeframe. Thus we pivoted to a non real time usage of Open Pose that would analyse a pre recorded video input and provide the locations of body parts as a control scheme.

After having generated the video of morphing faces from the original input video, we thought it would be for meaningful to be able to see the input video alongside the output. Such that viewers could see input movements invoking a corresponding change in the face shifts. Thus we made a script that combined each corresponding frame of the input video with its output portrait image frame and rendered the combined frames as the final video. Also, repeated viewings along the latent space seemed boring to us, so we decided to generate a few extra sets of latent spaces to phase between.

Result:

Out final results consists of two side by side videos, on the left is the generated morphing face output and on the right is the corresponding video used as an input. Three of these videos can be seen on the vimeo link provided.

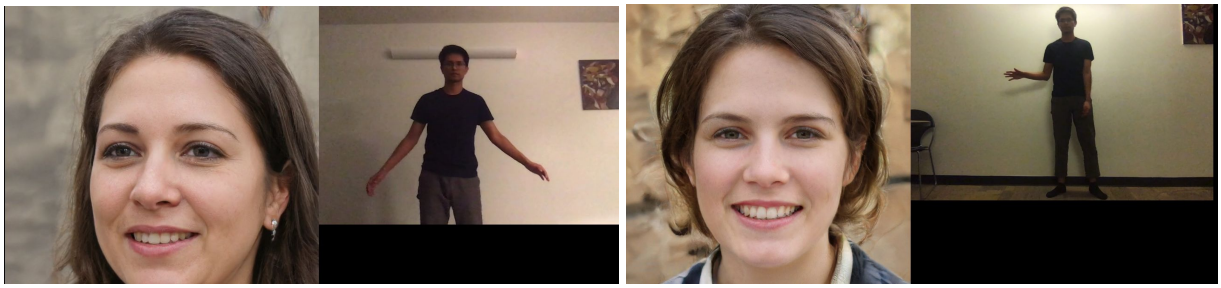


Figure 4. - Thumbnails of sample final output

Reflection: .

Our final results were simply chosen based upon their visual appeal and intrigue to us. For some inputs where our tracked body part did not move much, the final output video could be seen as boring and static. Thus we tried to present outputs that showed a clear visual change and movement through the 2D grid. We are fairly satisfied with the ideas and techniques that we have used for the project. The higher resolution images seemed to have much more visual

impact compared to the images from our brief foray with DCGAN and once setup, tf-openpose worked well. Future ideas that we would still like to have explored would be to have some sort of real time interaction between the viewer and the artwork.

REFERENCE:

[1] Flickr-Faces-HQ Dataset (FFHQ) https://github.com/NVlabs/ffhq-dataset

[2] Tero Karras, Samuli Laine: “A Style-Based Generator Architecture for Generative Adversarial Networks”, 2018; [http://arxiv.org/abs/1812.04948 arXiv:1812.04948].

[3] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei: “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”, 2018; [http://arxiv.org/abs/1812.08008 arXiv:1812.08008]

[4] DCGAN-tensorflow https://github.com/carpedm20/DCGAN-tensorflow

CODE:

https://github.com/ucsd-ml-arts/ml-art-final2-phases

RESULT:

Three video inputs and subsequently generated videos can be found here:
https://vimeo.com/user89820281