Stylizing Audio Reactive Visuals

Han-Hung Lee, Da-Gin Wu, and Hwann-Tzong Chen
National Tsing Hua University, Taiwan
{rexleeppp, dottyjunkie870407}@gmail.com
htchen@cs.nthu.edu.tw

1 Introduction

VJ is an art-form of mixing videos so that the visuals match the mood or groove of the music being played. While the video clips may be sampled from movies or animations, the video frames alone are more static in a sense as the contents of the video frames are fixed. Audio reactive visuals enable more dynamic performances by mapping audio input of the music to some visual effect so that the visuals vibrate or *resonate* with the music. A common way to do this is to use FFT and filters to obtain a frequency band that might correspond to an instrument such as snare drums, and then map the magnitudes of the changes in this frequency band to the parameters of a visual effect such as blur or distortion. In this work, we explore the use of GANs [2] to produce audio reactive visuals by following these magnitude changes in the frequency band to traverse the latent space of GANs. Because the latent space is smooth and interpolatable, by concatenating the generated images we can form a smooth video clip that reacts to the audio clip. We choose to use StyleGan [4] in particular because it maps a latent vector to several styles that control coarse-to-fine structures of the image, which makes it intuitive to map to when we have multiple features. We also explore using *Nsynth* [5] to extract features from the audio clip and *GAN steerability* [3] to learn specific walks in latent space that correspond to effects such as zooming and rotation. We use different pretrained StyleGan models for our experiments. Video results can be found on the website listed in Supplementary Materials.

2 Method

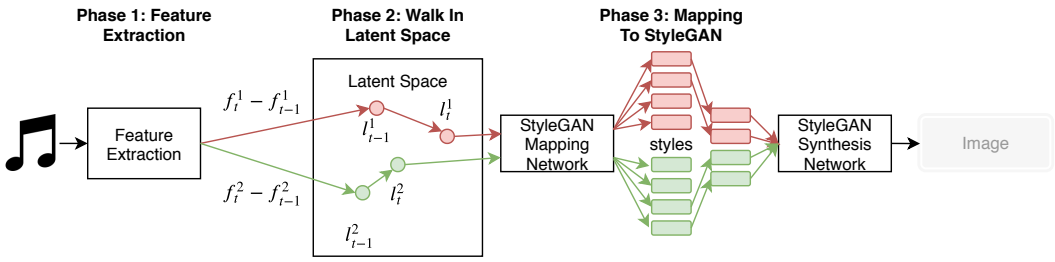

Figure 1: The workflow of our method.

Phase 1: Feature Extraction For the first phase, we extract features from the audio clip using zero-phase digital filtering for low-pass, band-pass, and high-pass filters. The root-mean-square energy is then calculated for each filtered frequency. We choose the frequency bands that are more noticeable such as bass or snare drums. We also encode the audio using Nsynth, a WaveNet-style [5] autoencoder to obtain features. In total 3 features and 16 features are obtained for each sample in time respectively for the above methods. (See Supplementary Materials for the details on frequency bands and Nsynth features.)

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Phase 2: Walking in the Latent Space Our baseline for navigating latent space is random walk. For each feature in every time-step/sample we calculate the changes in magnitude by subtracting the previous time-step and normalizing. Direction vectors are sampled from Gaussian and multiplied by the magnitudes. We walk in the latent space by adding the direction vectors back into the latent vectors from the previous step. Note that we normalize the direction vectors by some predefined value to limit the maximum walk length per time-step. Although random walk already works well, it may be more desirable to be able to control what kind of effect is applied when walking in latent space. We use the techniques of GAN steerability [3] to learn the walk vector for color and use the three features to control the RGB sliders of the image.

Phase 3: Mapping to StyleGan We feed each of the latent vectors that are obtained in the previous step into the mapping network of StyleGan and obtain multiple styles. Each latent vector contributes some styles to the image being synthesized. Styles that are fed into the beginning layers of the synthesis network control coarser structures of the image while styles that are fed towards the last layer control finer details in the image. (See Supplementary Materials for the descriptions on how the styles are mapped.)

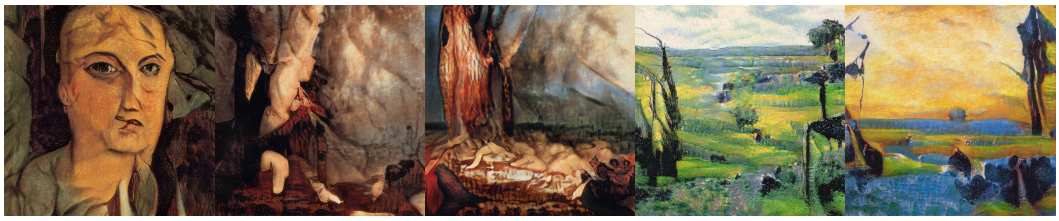

Figure 2: Screenshots of video frames produced using audio to walk in the latent space.

3 Discussion

Feature Extraction Although the encoding using Nsynth is straightforward and requires no human supervision or domain knowledge, it becomes non-trivial to map to StyleGan because the semantic meanings of the features are not known to us. On the other hand, when using hand crafted features, we lose the frequencies that are filtered out, causing the problem of mismatch between audio and video such as there is sound but the video is not moving. We think that it is worth exploring in the future how the features extracted by Nsynth correlate with the encoded audio for better mappings.

Mapping In our current implementation, we map more noticeable sounds in the audio clip to styles that control the coarser structures of the image such as the overall layout of the objects, and map less noticeable ones to styles that control finer detail such as the color of the objects. We hope to automate this process in the future by finding a perception metric for both sounds in the audio clip and structures of the image so that we can map more precisely and create more audio-visual coherent videos.

Learned Walks With random walk, when the max walk length per time-step is too high, the latent vector can move to areas that have low probabilities for the Gaussian distribution and cause glitches in the image. Moreover, we also have no control on the effects when moving in latent space with random walk. Using learned walks from Jahanian et al. [3] we can specify the preferred effects in latent space and use the features to control the sliders of the effect parameters. Right now we only learn the color walk vector and map the features to the RGB sliders because we are unsure if multiple learned walk vectors will interfere with one another or if their parameters are decorrelated. We will explore this in the future so that we are able to map each feature to a different effect.

Real Time Mixing With a GTX 1070 and output image resolution of 512^2 we obtain about 8–9 fps with StyleGan (excluding feature extraction). Because VJ is usually done in real time, we set our batch size to one during inference. Due to the relatively low resolution and frame rate, GANs are unsuitable at the moment for real-time mixing without more powerful GPUs. It might be worthwhile to look into methods such as model compression and NAS to make the models smaller and faster.

References

[1] Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., & Norouzi, M. (2017). Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. ICML.

[2] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., & Bengio, Y. (2014). Generative Adversarial Nets. NIPS.

[3] Jahanian, A., Chai, L., & Isola, P. (2019). On the "steerability" of generative adversarial networks. ArXiv, abs/1907.07171.

[4] Karras, T., Laine, S., & Aila, T. (2018). A Style-Based Generator Architecture for Generative Adversarial Networks. ArXiv, abs/1812.04948.

[5] Oord, A.V., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A.W., & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. SSW.

[6] Yu, F., Zhang, Y., Song, S., Seff, A., & Xiao, J. (2015). LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. ArXiv, abs/1506.03365.

Supplementary Materials

Experimental results can be found at

`https://hanhung.github.io/Creating-Audio-Reactive-Visuals-With-StyleGAN/`

Feature Extraction and Mapping Frequency settings and mapping for filters of hand crafted features are shown in Table 1 for images of resolution 512^2. We label the styles 0–15 by the order in which they are fed into the synthesis network of StyleGan. For the mapping of Nsynth features we first sort the features by the maximum magnitude change between time-steps from largest to smallest. We then map the features by this sorted order to the styles. If there are more features than styles we discard the ones that have the smallest magnitude changes.

Table 1: Filter settings and mapping to StyleGan styles for pretrained StyleGan on paintings[1].

Filter Type	Frequencies	Sounds	Styles (Layers)	Effect on Image
Low Pass	$\leq$ 150 Hz	Bass	0–2	Coarse Structure
High Pass	500–5000 Hz	Mid-Highs	3–5	Mid Level
Band Pass	200–350 Hz	Snare	6–15	Fine Detail

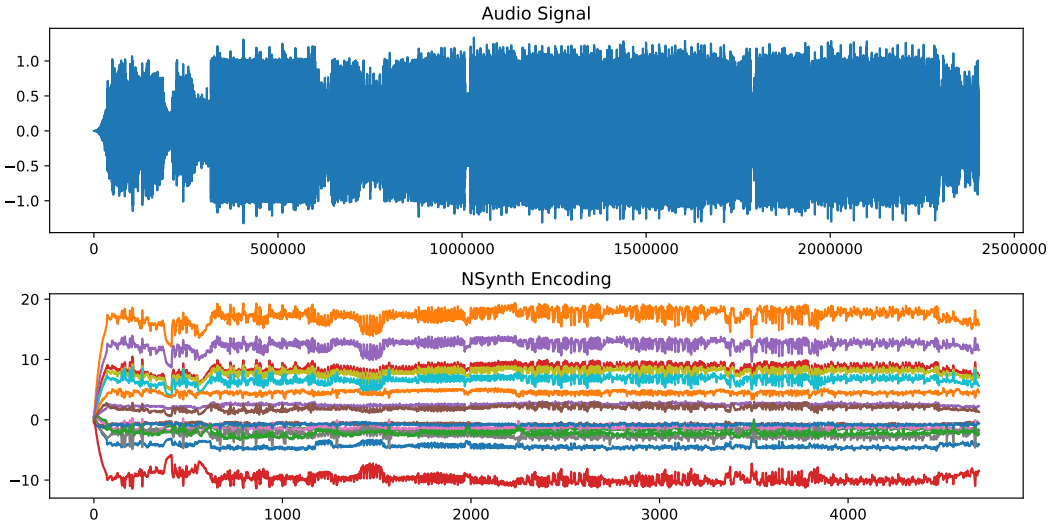

Figure 3: Visualization of encoding with Nsynth.

[1]Pretrained StyleGan on paintings can be found here: `https://www.reddit.com/r/MachineLearning/comments/bagnq6/p_stylegan_trained_on_paintings_512x512/`

Effects of Mapping Style Layers To demonstrate the effects of different style layers in StyleGan, we conduct some ablation studies by changing the latent vector of style layers mapped by one filter type while fixing the two others. A StyleGan pretrained on the LSUN Bedroom[2]dataset is used in this experiment. The filter settings and mapping to StyleGan styles is shown in Table 2. The final results are shown in Figure 4. We see that while changing the style layers mapped by the low pass filter (top row), the overall layout of the room changes such as the number of beds or how large the room is. When changing style layers mapped by the high pass filter (middle row), smaller changes are observed such as the ceiling fan light or the blanket on the bed while the overall layout of the room remains unchanged. Finally for the style layers mapped by the band pass filter (bottom row), only the textures of objects change.

Table 2: Filter settings and mapping to StyleGan styles for pretrained StyleGan on LSUN Bedroom.

Filter Type	**Frequencies**	**Sounds**	**Styles (Layers)**	**Effect on Image**
Low Pass	$\leq$ 150 Hz	Bass	0–2	Coarse Structure
High Pass	500–5000 Hz	Mid-Highs	3–5	Mid Level
Band Pass	200–350 Hz	Snare	6–13	Fine Detail

Figure 4: Results of the ablation experiments. The top to bottom rows are obtained by changing the style layers mapped by low, high and band pass filters respectively while fixing the two other filters.

[2]Pretrained StyleGan on LSUN Bedroom can be found here: `https://github.com/NVlabs/stylegan`