Program Descriptions: Media Transform Reconstructors & Others

Maxwell Bakalos

12/31/2022

(Personal Project)

GitHub Link: https://github.com/maxbak753/Media-Effects-Programs

(These programs were written in MATLAB but may be converted to another language such as Python later)

MEDIA TRANSFORM RECONSTRUCTORS

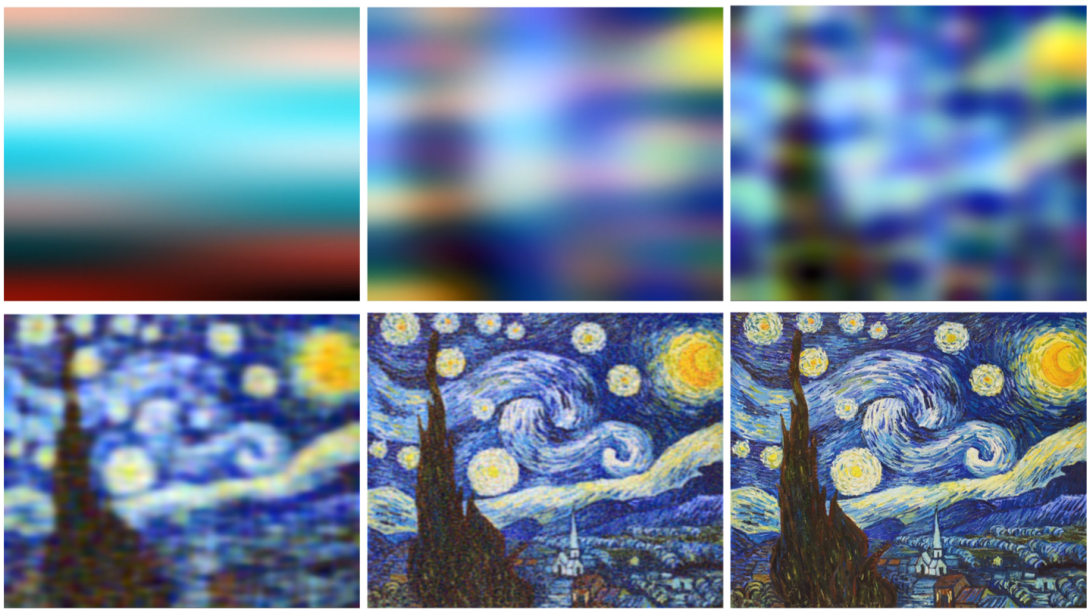

Figure 1 - Image Transform Reconstructor process using the Discrete Cosine Transform

Video Transform Reconstructor

Video_Transform_VideoMaker.m

This program takes an existing video as an input and outputs a new video that shows the gradual rebuilding of the original by increasing the number of transform basis signals in each frame. The highest energy transform basis signals are used first and eventually, all (or a certain percent) of the basis signals are added. They can either be added linearly or exponentially (exponential usually looks better). This creates a unique video effect (decompression). Either the Discrete Cosine Transform (DCT) or the Discrete Wavelet Transform (DWT) can be used. Multiple different wavelet shapes (db1 - db45) are available for the DWT.

Image Transform Reconstructor

Image_Transform_VideoMaker1.m

Similarly the video transform reconstructor, basis signals are slowly added in order from greatest to least energy, however, a single image is used instead of a series of image frames. DCT and DWT are available in the same way.

Audio Transform Reconstructor

audio_reconstructor_increment.m

Instead of doing a 2-dimensional DCT, this program does a 1-dimensional DCT on an audio file. The basis signals are slowly added in order from greatest to least energy. The number of increments can be specified (similar to the number of frames in the video/image reconstructors). The result is an audio file that starts with the most fundamental sounds and gradually progresses to full-quality audio (decompression).

OTHERS

Noise Creators (Audio, Image, Video)

Audio_Noise_Creator.m Image_Noise_Creator.m Image_Noise_Video_Creator.m

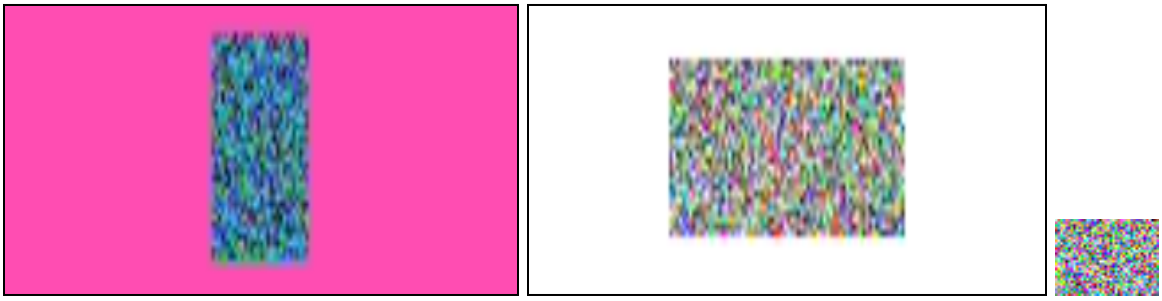

Figure 2 - (Left) noise video frame, (Middle) noise video frame, & (Right) noise image

These three programs can either create an audio file, image, or video of random noise. The noise can be gaussian random noise with a specified mean and variance, and a color balance (image/video), or it can be saturated to be only 0 or 1. I have also added the feature that this noise can be bounded either by a certain sine wave frequency (audio) or color (image/video).

Re-Scalers (Image & Video)

Image_Rescaler.m Video_ReScaler.m

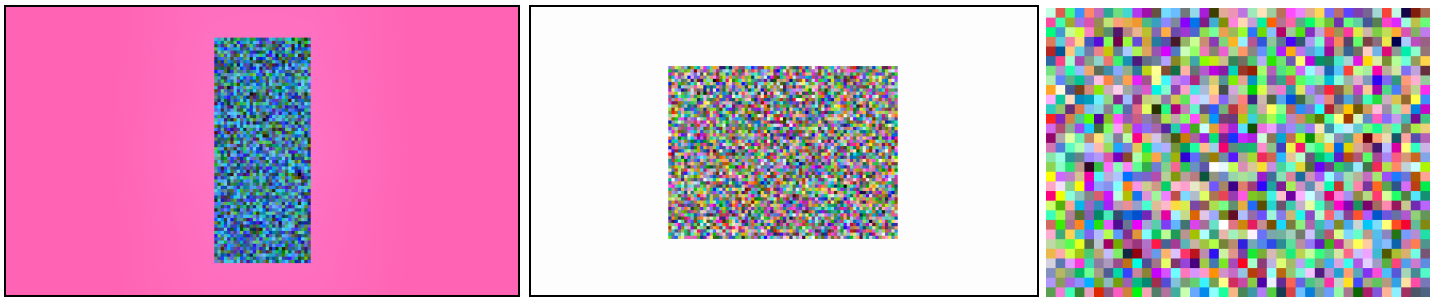

Figure 3 - Upscaled media from the noise creators

These programs simply upscale either a single image or each frame of a video by a specified amount. It does not interpolate between the known and unknown pixels in the upsampled image, it simply expands each pixel value into more pixels. This is different from the upsampling a computer usually does to pixelated images which makes them look blurry. It can be used to make pixel art images larger and still retain their pixelated aesthetic. I have created down-scalers (downsamplers) before but I have not yet added that functionality to these programs.

Image Clusterer

Image_Clusterer.m

Figure 4 - Clustered Images: (Left) original, (Middle) 2 clusters, & (Right) 3 clusters

This program takes an image as an input and outputs a “clustered” version of the image. It uses the k-Means clustering algorithm on the pixel values of the image to cluster the pixels into groups of similar colors, then maps each group to a specific output color in order to get an effect similar to the “posterize” effect in many image/video editors. It essentially re-quantizes the pixel values into groups based on the clustering algorithm.

Edge Detector (Video)

edge_detect_videoMaker.m

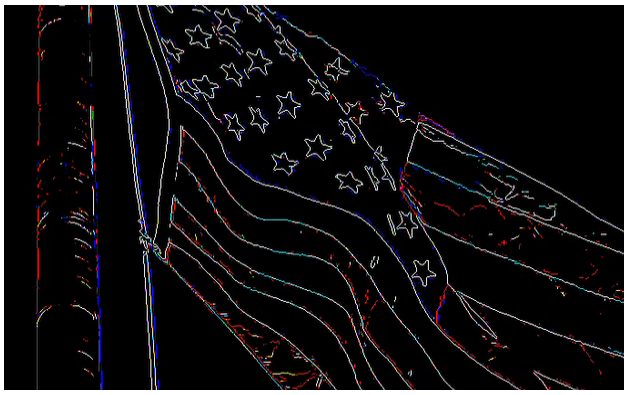

Figure 5 - Edge detected video frame in each color

This program takes a video as an input and outputs a new video where each frame is an edge-detected version of the previous frame. It can either detect edges in each color or use all colors to detect edges.