Schuyler Martin
CSCI-731-01
April 11th, 2017

ASCII Art, Part II

Premise

After my ASCII image art project from last semester, I have really wanted to write something that would convert a whole video into ASCII art, in a reasonable amount of time. So I have accomplished that for the “free elective” project for this course. The main goal was to write something that could be fast enough to handle a full-blown movie in a reasonable amount of time. The MATLAB project I wrote last semester could render a frame in about 10 seconds. With this simplified C variant, I am able to render a 20 minute, 30fps video in about 10 minutes.

I also wanted this project to use a different method of ASCII rendering than I had done previously. Instead of using luminance values, I look only at the Canny edges in the scene. Instead of using multiple kinds of characters, I stick with one character. Instead of rendering to text files, I use OpenCV’s font renderer to make images. Instead of rendering in just black and white, I render the font based off of local color values in the original frame. All of these design decisions were made to decrease the rendering time while simultaneously creating a different art style.

Major Issues

- I had significant issues at the beginning with the scale of this project. I do not think I have ever processed so much data before. The main issue was with the initial design. I normally read in the frame data all at once and place it in a vector for faster processing. However doing this with even small video files proved to be too difficult. Within a few seconds I would run out of RAM and my OS would start consuming large amounts of space on my SWAP partition. So I eventually removed this first pass and just worked with the video stream writer directly. This significantly reduced my memory overhead and saved a lot of time.
- OpenCV’s provided font is not fixed-width. That means it is significantly harder to make meaningful drawings out of the ASCII art. To get around this, I limited myself to rendering only one type of character for the whole video and scale that video’s size relative to the bounding box size of that character. The character is specified as a macro and can be fun to play with. I personally think centered, blocky characters like x,*,+, etc work the best.
- I also had a lot of issues determining if I was making progress in processing the video or not. So I implemented a text-based loading bar to show progress as the frames are being rendered.

Future Work

- I would really like to parallelize this project. Unfortunately there seems to be no standard way to use threads with OpenCV. That is to say that I would need to add additional dependencies to my project in order to pull this off, which I am not a big fan of. That being said, if I could find a solution, I am also not sure if the video streams are thread safe. If they are, I could easily speed up the rendering process by placing the processing code in a parallel for loop.

Output Images

All of the output images come from Pink Floyd's short film, *The Final Cut*