Slit Scan Notes

Fred Mitchell

May 21, 2022

Contents

1 Slit Scan Notes

1.1 Disclaimer

These are my basic notes on this project, and are not meant for general consumption, and therefore are not gauranteed to be accurate or even useful for anyone other than myself. Please see the README.org or the README.pdf for that.

1.2 Loading Images

- We could resize one of the images to be the exact same size/dimensions of the other. Would simplify the math quite a bit. But might cause some wierdness

in some circumstances. Of course, the user can correct the issue and make sure both images are the same dimensions, etc.

- Trying to determine how to transverse the images slit-wise. In mapping the images, we are simply given the x,y coordinates. Of course, we can work backwards from there to determine the "slit", and should cover this in the math description in the README.

1.2.1 Pipelining

We can manipulate the source images to be the same size, and rotate them to do a horizontal scan. For 2 images, we create each canvas seperately, then combine them during the compositing phase.

Indeed, we shall take a pipelining approach, so we can run all of this in parallel to utilize all the available cores.

1.3 Scanning the images

It just occured to me that we should scan from the basis of the canvas, since that is the target, and simply do the math transforms to the source images. This way, we can handle blending / dithering / smoothing between pixels on the destination image (canvas) very naturally.

In fact, we can utilize the normal mapping function of the destination, and simply address the pixels via the index fuction from the sources.

1.3.1 Orign of Graphics.Image images

The origin is in the upper-left corner and descends down and right. This is typical of computer graphics going all the way back to the raster scan days which almost invariably stated from the upper left and scanned to the right and down.

From a mathematical perspective, I have always found this annoying, but for purposes of keeping things "simple", I will embrace that in my math here.

1.3.2 Scan Direction

On the source images, we will initally scan from right to left, meaning that conceptually the source image is moving from left to right. Time t is a Double parameter, even though a single tick of t represents one pixel.

Scan speed is based on scans per second, which will be somewhat related to frames per second, but not perfectly.

1.3.3 Scan Index (si)

The scan index roughly relates to the frame index (fi) but is based on the number of scans per second. On the first frame, si will be less than the actual scan width of both the source and canvas. This will eventually change, where the scan index will exceed the scan width of the source, in which case – for now, it will wrap around.

For the canvas, it will simply go to the end and stop.

In actuality, I don't think there's anything special we need to do for the canvas. It's all on the source, and handling the wrap-around correctly.

1.3.4 Logic change

I now understand that I must copy a single slit into an array and just keep advancing this array on each iteration, not the way I am doing it with the matrix!!! The logic will have to be shifted.

- We shall do this by having an intermediate buffer that will buffer up the slits. The image buffer will have to be shifted over by one row or column to accept the next slit.

  We then apply the matrix operators not to the source image, but to the intermediate buffer, to get the final effect we are looking for.

1.4 Outputting the video

We shall use "out" as the path-filename fragment, where a 4-digit (or 5-digit?) sequence number shall be appened, along with the .EXT (default is png).

Later on, we'll change this to do a video file directly.

```
ffmpeg -framerate 30 -pattern_type glob -i '*.png' -c:v libx264 -pix_fmt yuv420p out.mp4
```

1.5 Future planned enhancements