Image Filter Recreation: Nashville Filter

For my image processing research, we wanted to recreate the Nashville image filter. This filter is more subtle than most, with only a few tweaks to the colours of the final image. Ultimately though, it's subtlety made it more difficult to recreate, and while we achieved something similar, it's far from an exact recreation.

The Nashville filter is characterised by three main effects: warm highlights and cool shadows, a general increase in brightness (with more even lighting as well), and a softer image. The softness is interesting, because it is done without a blur. While it has a similar effect as a blur, the image still has some noise and fine details visible. Also, the image is not desaturated, because the Nashville filter actually tends to increase saturation.

The code for the MyNashville filter is shown below:

```
function MyNashvilleFilter(srcImgData, destImgData, width, height){
    var size = width * height * 4;

    var colTrans = [
                        [1.22,  -0.08,  -0.09],
                        [-0.06,  0.88,  0.21],
                        [-0.12,  0.25,  0.86]
                    ];

    for(var idx = 0; idx < size; idx += 4){

        for(var pix = 0; pix < 3; pix++){
            var total = 0;

            for(var offset = 0; offset < 3; offset++){
                total += colTrans[pix][offset] * srcImgData.data[idx+offset];
            }

            destImgData.data[idx+pix] = total;
        }

        var brightness = srcImgData.data[idx] + srcImgData.data[idx+1] + srcImgData.data[idx+2];
        brightness /= 3;

        var lumShiftEffect = 0.05;
        destImgData.data[idx] += (brightness-0.3)*lumShiftEffect;
        destImgData.data[idx+1] -= (brightness-0.3)*lumShiftEffect;
        destImgData.data[idx+2] -= (brightness-0.3)*lumShiftEffect;

        var pow = 3;

        var remap = function(x){
            var coefficient = 0.5/Math.pow(0.5, pow);
            return 255*(coefficient*Math.pow(x/255 - 0.5,pow)+0.5);
        };

        var weight = -0.3;

        var lightnessFX = 0.1;
```

```
            for(var pix = 0; pix < 3; pix++){
                destImgData.data[idx+pix] = destImgData.data[idx+pix] * (1-weight)
                                            + weight*remap(destImgData.data[idx+pix]);
                destImgData.data[idx+pix] = destImgData.data[idx+pix]*(1-lightnessFX)
                                                + 255*lightnessFX;
            }

            var avg = 0;
            for(var pix = 0; pix < 3; pix++){
                avg += destImgData.data[idx+pix];
            }

            avg /= 3;

            var desat = 0.2;

            for(var pix = 0; pix < 3; pix++){
                destImgData.data[idx+pix] = destImgData.data[idx+pix] * (1-desat) + avg * desat;
            }

            for(var pix = 0; pix < 3; pix++){
                if(destImgData.data[idx+pix] < 50){
                    destImgData.data[idx+pix] = (destImgData.data[idx+pix]/255)
                                                *(destImgData.data[idx+pix]/255)*255;
                }
            }

            destImgData.data[idx+3] = 255;

        }
    }
```

The first thing we do is remap the colours in each pixel. This is done to adjust tinting so that highlights are warmer, and shadows are cooler. This also effects midtones in a similar way, to try and make the effect a bit more subtle. The next step mostly affects the whites/blacks in the image, so we try to soften its impact with this step.

```
var colTrans = [
                    [1.22,  -0.08,  -0.09],
                    [-0.06,  0.88,  0.21],
                    [-0.12,  0.25,  0.86]
                ];
```

The next step is to exacerbate the effect we just achieved. Here, we calculate the brightness of each pixel, and use that to effect each channel. The brighter a pixel is, the more red it will get, and the less blue and green it will get. Since this effect can be a bit extreme, it's diluted by multiplying in the *lumShiftEffect* variable. This controls how much each pixel is tinted based on brightness in this second step.

```
var brightness = srcImgData.data[idx] + srcImgData.data[idx+1] + srcImgData.data[idx+2];
brightness /= 3;

var lumShiftEffect = 0.05;
destImgData.data[idx] += (brightness-0.3)*lumShiftEffect;
destImgData.data[idx+1] -= (brightness-0.3)*lumShiftEffect;
destImgData.data[idx+2] -= (brightness-0.3)*lumShiftEffect;
```

Now that we've dealt with tint adjustments, we move on to adjusting levels. In this case, we convert the 0-

255 value in the channel to linear space, cube it, and then map it back to 0-255. This has the effect of accentuating the shadows and highlights, while flattening the midtones. This partially helps achieve deeper shadows and bigger highlights, but also helps make the midtones look softer (as the Nashville filter does).

```
var pow = 3;

var remap = function(x){
    var coefficient = 0.5/Math.pow(0.5, pow);
    return 255*(coefficient*Math.pow(x/255 - 0.5,pow)+0.5);
};

var weight = -0.3;

var lightnessFX = 0.1;

for(var pix = 0; pix < 3; pix++){
    destImgData.data[idx+pix] = destImgData.data[idx+pix] * (1-weight)
                                + weight*remap(destImgData.data[idx+pix]);
    destImgData.data[idx+pix] = destImgData.data[idx+pix]*(1-lightnessFX)
                                + 255*lightnessFX;
}
```

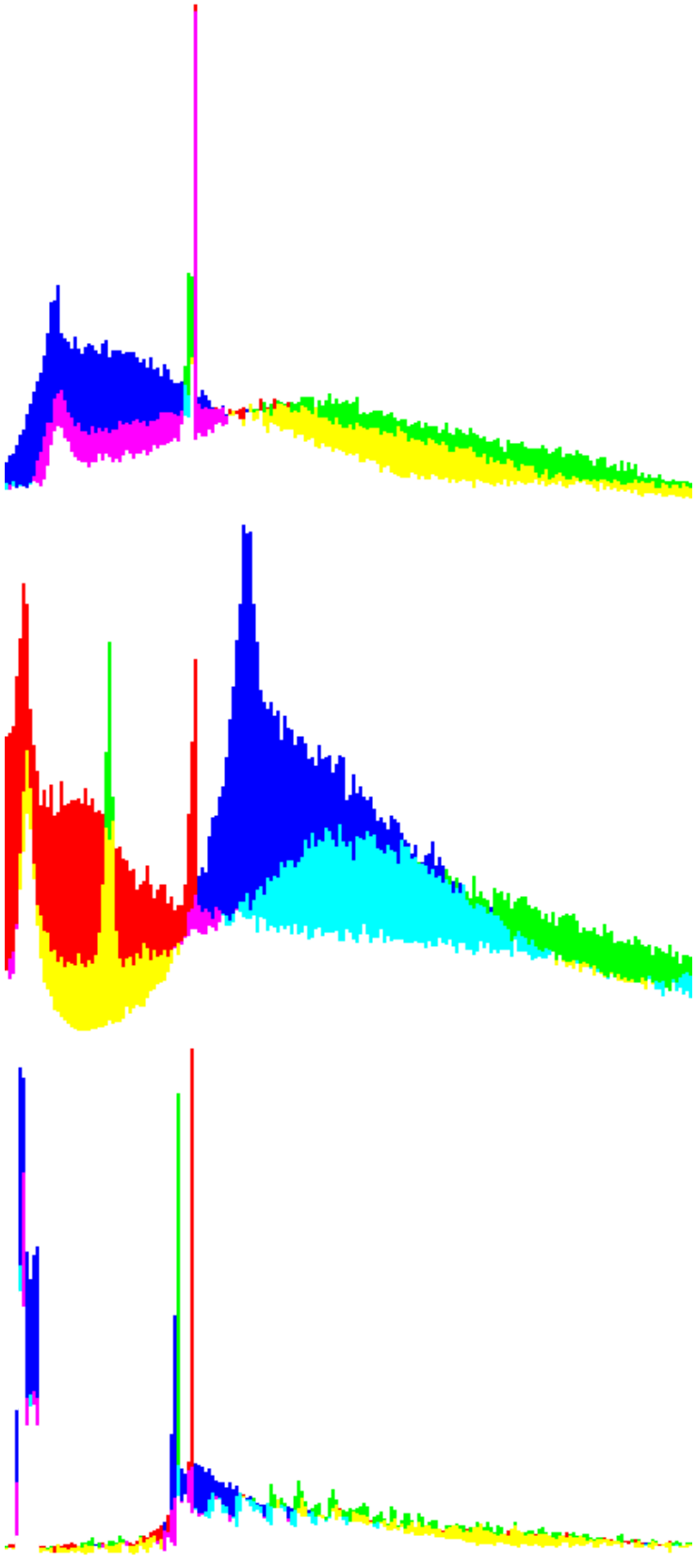
Although the filter is supposed to make the image more saturated, we're going to slightly desaturate it here. This is because the previous steps have boosted the saturation beyond what we want. So we bring it down slightly by just finding the average, and lerping each separate channel toward the average.

```
var avg = 0;
for(var pix = 0; pix < 3; pix++){
    avg += destImgData.data[idx+pix];
}

avg /= 3;

var desat = 0.2;

for(var pix = 0; pix < 3; pix++){
    destImgData.data[idx+pix] = destImgData.data[idx+pix] * (1-desat) + avg * desat;
}
```

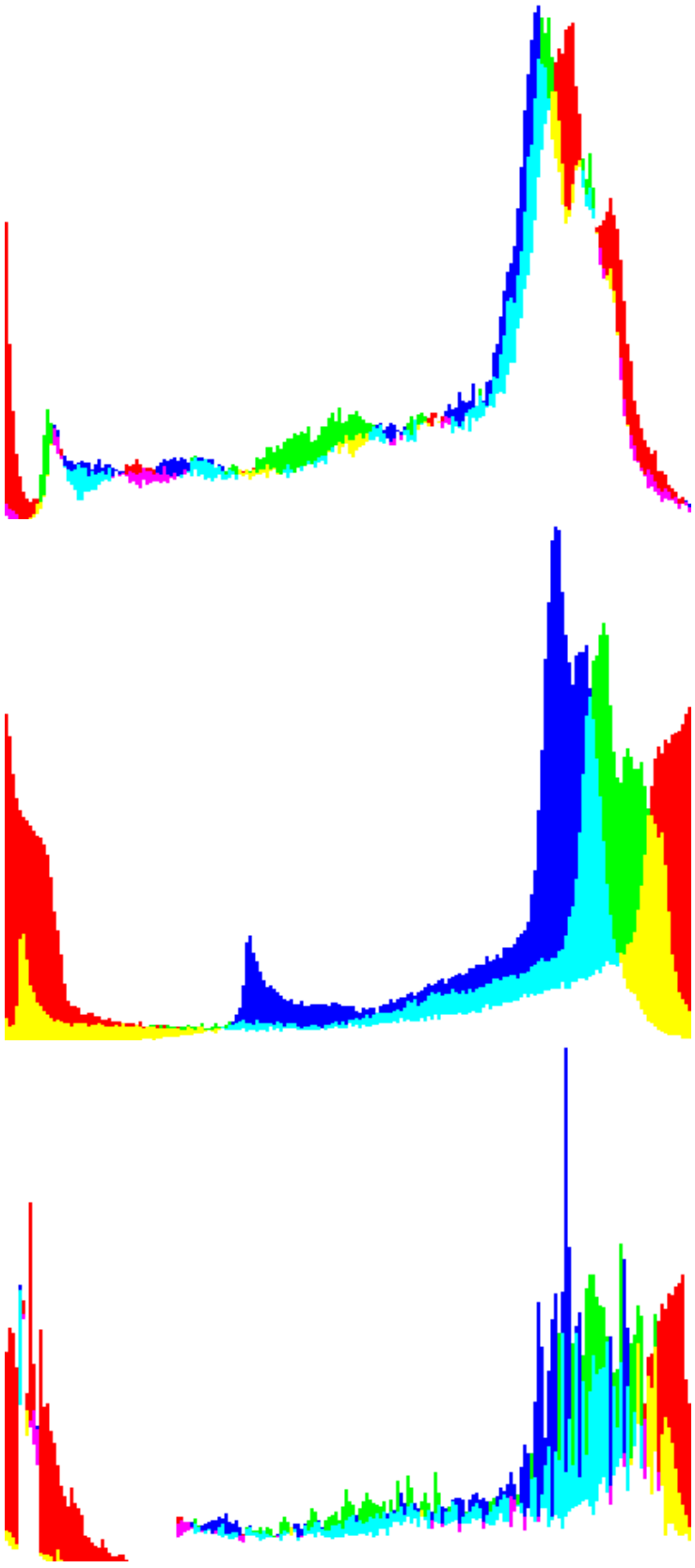
Finally, we deepen some of the shadows. The Nashville filter tends to lower the value of shadows in the image, and we've already done some of that. However it wasn't quite to the same extent. So, we simply normalize each channel, square it if it's below a threshold, and then remap it back to 0-255. This effect is harder to notice, since we've already manipulated brightness and re-mapped the channels' values, but it does deepen some of the darker areas in the image.

```
for(var pix = 0; pix < 3; pix++){
    if(destImgData.data[idx+pix] < 50){
        destImgData.data[idx+pix] = (destImgData.data[idx+pix]/255)
                                    *(destImgData.data[idx+pix]/255)*255;
    }
}
```

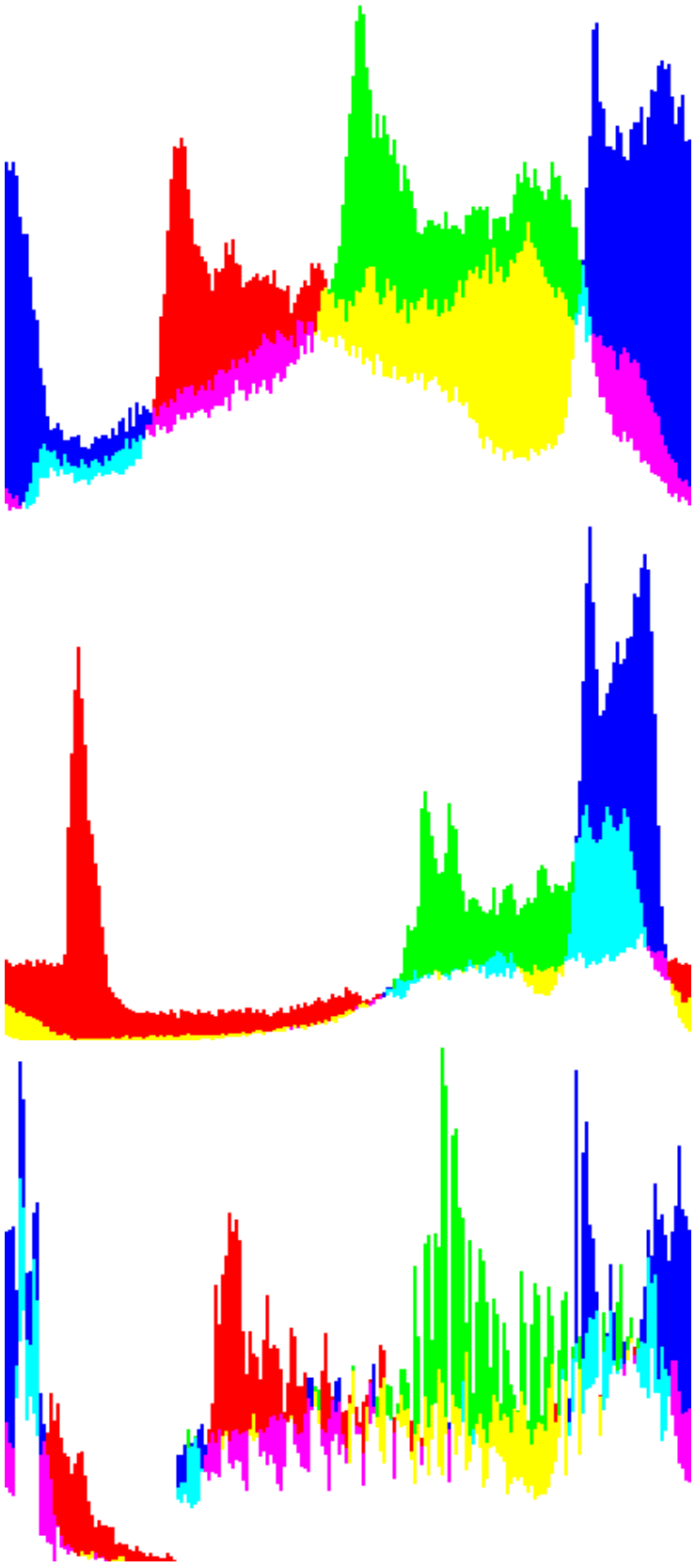
Original image	Instagram filer	Our recreation
Original image (histogram)	Instagram filer (histogram)	Our recreation (histogram)
Original image (saturation)	Instagram filer (saturation)	Our recreation (saturation)



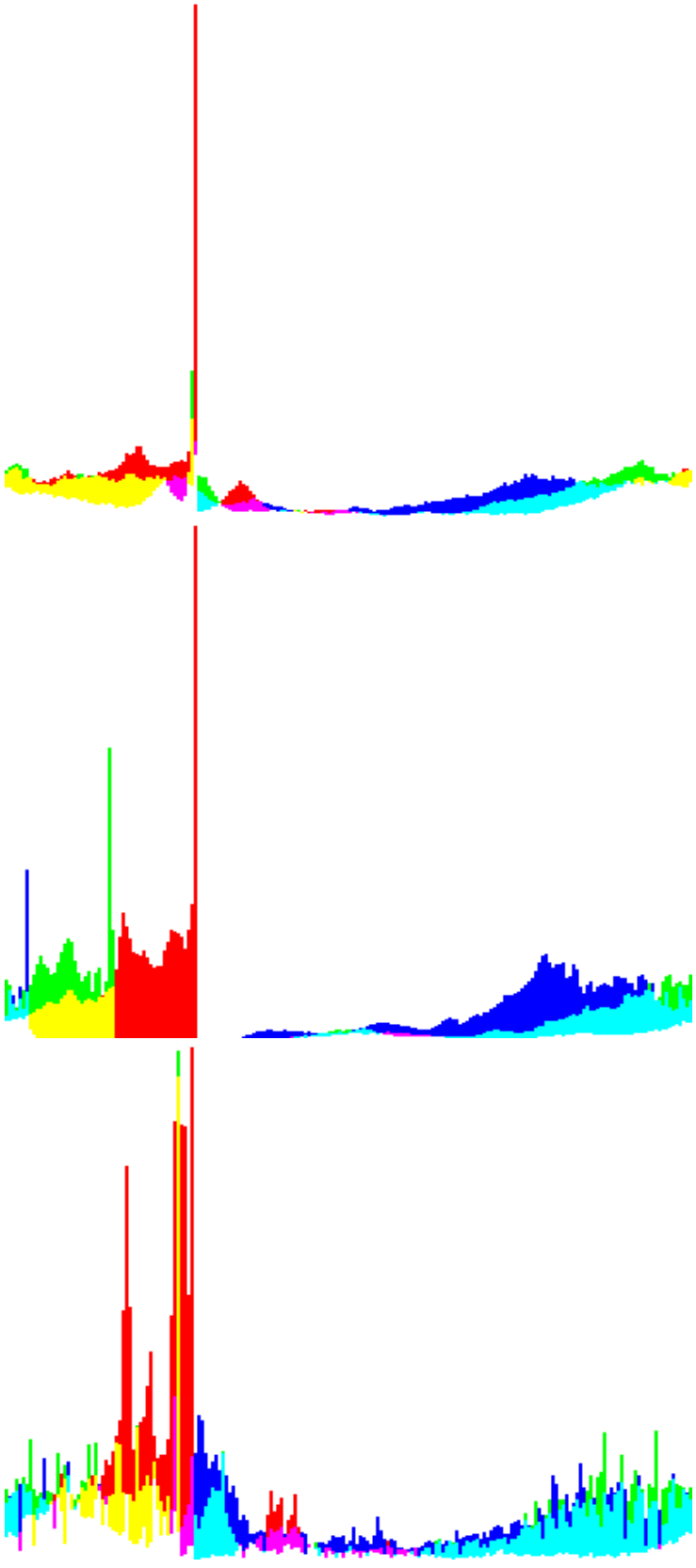
Original image	Instagram filer	Our recreation
Original image (histogram)	Instagram filer (histogram)	Our recreation (histogram)
Original image (saturation)	Instagram filer (saturation)	Our recreation (saturation)



Original image	Instagram filer	Our recreation
Original image (histogram)	Instagram filer (histogram)	Our recreation (histogram)
Original image (saturation)	Instagram filer (saturation)	Our recreation (saturation)



Original image	Instagram filer	Our recreation
Original image (histogram)	Instagram filer (histogram)	Our recreation (histogram)
Original image (saturation)	Instagram filer (saturation)	Our recreation (saturation)



Overall, the main issue is the softness mentioned above. The that effect of the filter proved extremely difficult to replicate. Some of it was achieved through shifting the lightness of the image, and some of it was achieved through the desaturation, but the difference was still there. The tinted highlights and shadows did work out, however, as can be seen from the histograms. The histograms for red are shifted higher, and the histograms for blue are shifted lower, for the Instagram filter and our recreated filter.

In the Instagram filter, compared to the original image, it can be seen that the saturation has increased overall. The same effect can be seen with our recreated filter compared to the original image's saturation.

One additional issue that can be seen in the histograms for the recreated filter. The re-mapping of the pixels' values causes the histogram to have some artefacts in it. Specifically, the threshold beneath which each value is squared forces the bottom level to go even further. This makes a sudden drop in some of the histograms around 50/255. This is less visible in the final image, and was useful in darkening shadows. So despite making the histogram further off accuracy, it was left in.