In terms of external libraries, we are using Java IO to grab the tif image, and read that image into a byte array. We are also using java swing to create the GUI and display the images after all conversions have been done on them. Specifically we use imageIO and BufferedImage to load in the image files to the GUI.

The method we used is reading in the tif image as a byte array and that is the representation we have of the coloured image which we then use to generate the other 3 images.

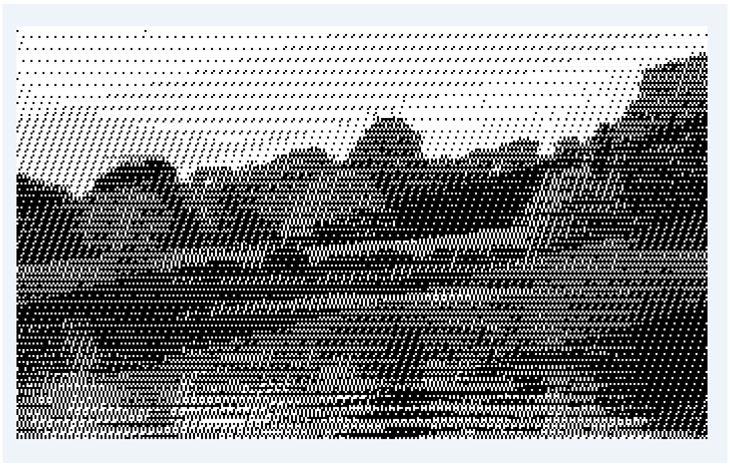
To generate the grayscale image, we take the coloured image byte array and actually create a new byte array which will be the grayscale image. We first generate the header using big endian and placing the IFD entries right after the 8 byte header. Next we generate the grayscale image. This is done by grabbing the RGB values of the coloured image by looking at the byte offset, and counting by the bytes in each strip, converting them into YUV, and taking the Y value as our gray level intensity value. With this value we place each pixel value on 1 strip, and then set the IFD entries for width, height, bitsperSample, compression, PhotometricInterpretation, StripOffsets, and stripbytecounts.

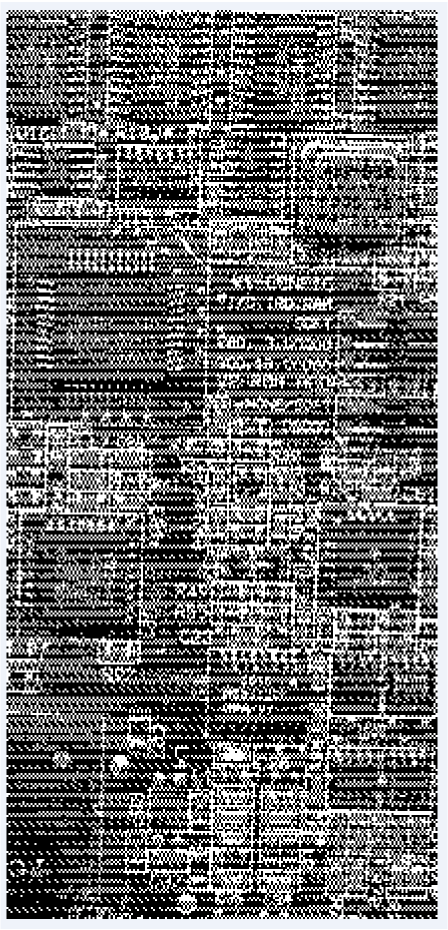
To generate the order dithered image, we follow a similar process for the grayscale image. The headers are generated first in big Endian, and the grabbing of values is the same here, and we still convert the RGB values to YUV, but instead we take the Y as an intensity and check it against our dither matrix. For this dithering, we pretty much used a hardcoded 4x4 bayer matrix. I originally was using my own generated matrix, but doing a further search, a bayer matrix is one that looks a lot cleaner and is a bit more evenly distributed than my own, so I switched later to this bayer matrix. Depending on the pixel number in the row, we do modulo 4 to find where on the matrix we need to check against for the intensity.

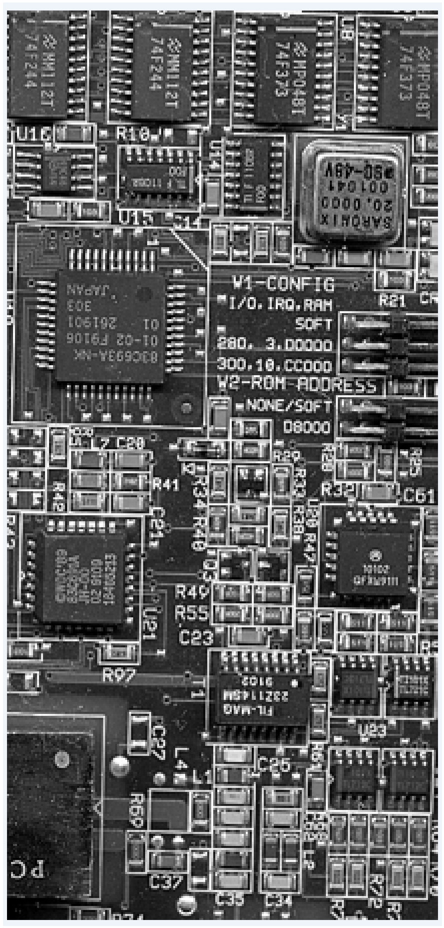
To generate the range adjusted image, again we are doing the same intake and reading of bytes, and generating the header. The difference is that after converting the RGB values to YUV, we are checking to see if the Y value (the luminance) is at or below a given threshold of 255. If the Y value is below this threshold, we simply add that value onto the Y. This effectively increases the darker spots (and can be done in reverse to darken the lighter spots) of the picture by the threshold. I tried doing this a few different ways, one being if the luminance is below a certain threshold I changed that value to be at least the threshold, and this caused the image to become very distorted, with the ranges changing much more drastically, whereas adding those values seemed to offer a much cleaner final image as we can see for some of the images in the demo video. The range can get pretty distorted at times, but as was discussed in a QA session is natural to happen.

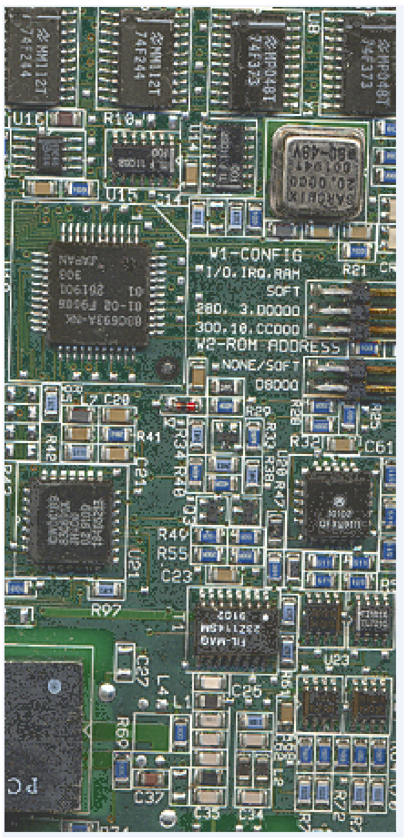
Our program also creates image files for all 3 of the images (as shown in the demo video) and can be opened to see that processed image. We do this simply by opening a new file named dithered,grayscale, or ranged respectively and then write, byte by byte to that file.

Below one each page, we have screenshots of the generated images in the order dithered, grayscale and then range adjusted.









Swiss Quality

Swiss Quality

Swiss Quality

ippon

ippon

ippon