Rebecca Edson
CPSC 4040 Final Project
Mapping to a Limited Color Palette
Fall 2019

Overview
I chose to implement the second project suggestion, where an image is mapped to a limited color palette.

Input
- User runs the code, including a .png image filename in the command line
- User may optionally include a second .png image filename, where the image with each pixel individually mapped to a palette color is written to the file, or a third, where the image mapped with dithering is written to the file
- Standard output asks user if they would like to use an existing palette written in a .txt file
- User types 'Y' for yes or 'N' for no
- If yes, user enters palette filename (including .txt extension)
- If no, user enters the number of colors and each RGB pixel value separated by a space

Method
- "Normal" mapping – each pixel is matched to the nearest palette color by calculating the Euclidian distance between the original color and each palette color and setting the new pixmap value to the palette color with the shortest distance
- Mapping with dithering – quantization error distributed to neighboring pixels using the Floyd-Steinberg dithering algorithm

Output
- Original image displayed on screen
- Normally mapped image displayed by pressing 'n' or 'N' key
    - Image written to file if specified
- Dithered image displayed by pressing 'd' or 'D' key
    - Image written to file if specified
- Original image displayed by pressing 'r' or 'R' key
- Quit program by pressing ESC key

Included palettes

3-bit.txt
```
8
255 0 0
0 255 0
0 0 255
255 255 0
0 255 255
255 0 255
255 255 255
0 0 0
```

black-white.txt
```
2
0 0 0
255 255 255
```

dark.txt
```
13
102 0 0
102 51 0
102 102 0
51 102 0
0 102 0
0 102 51
0 102 102
0 51 102
0 0 102
51 0 102
102 0 102
102 0 51
32 32 32
```

neon.txt
```
12
255 0 0
255 128 0
255 255 0
128 255 0
0 255 0
0 255 128
0 255 255
0 128 255
0 0 255
127 0 255
255 0 255
255 0 127
```

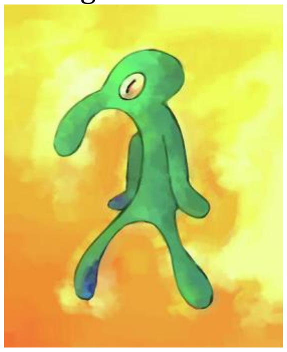
pastel.txt
```
13
255 204 204
255 229 204
255 255 204
229 255 204
204 255 204
204 255 229
204 255 255
229 204 255
204 204 255
229 204 255
255 204 255
255 204 229
255 255 255
```

grayscale.txt
```
9
0 0 0
32 32 32
64 64 64
96 96 96
128 128 128
160 160 160
192 192 192
224 224 224
255 255 255
```

pink.txt
```
9
51 0 25
102 0 51
153 0 76
204 0 102
255 0 127
255 51 153
255 102 178
255 153 204
255 204 229
```

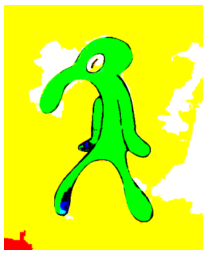
rgb.txt
```
3
255 0 0
0 255 0
0 0 255
```

Original Image: boldandbrash.png

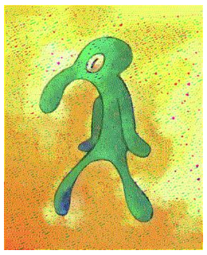


3-bit.txt

Normal

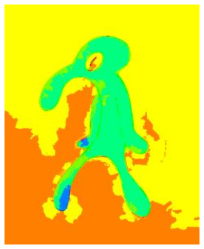


Dithered

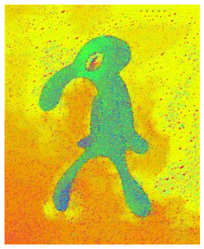


neon.txt

Normal

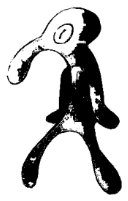


Dithered

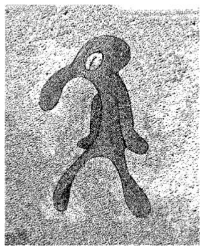


black-white.txt

Normal

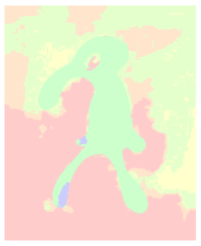


Dithered

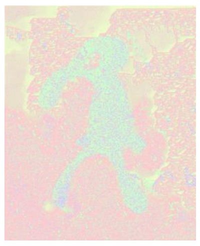


pastel.txt

Normal

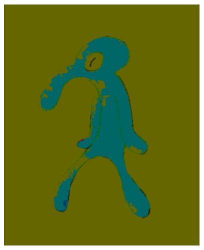


Dithered

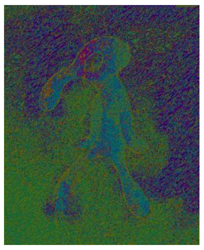


dark.txt

Normal

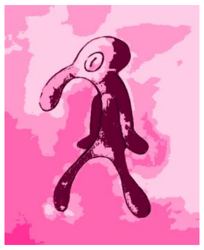


Dithered

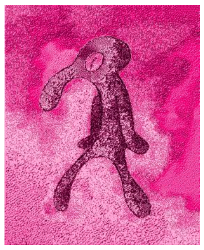


pink.txt

Normal

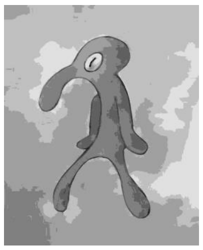


Dithered

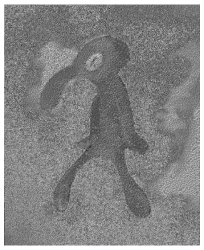


grayscale.txt

Normal

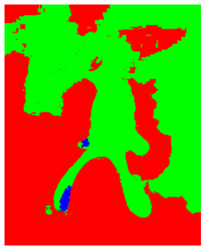


Dithered

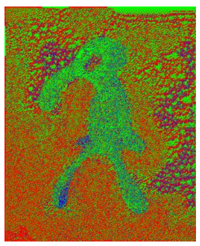


rgb.txt

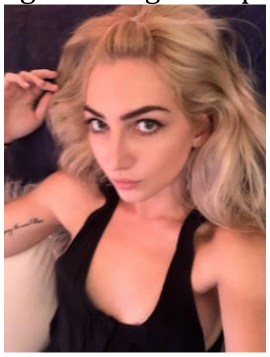
Normal



Dithered

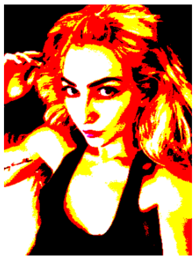


Original image: me.png

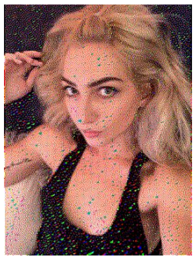


3-bit.txt

Normal

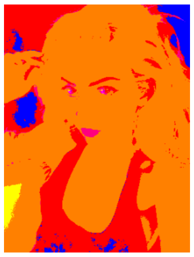


Dithered

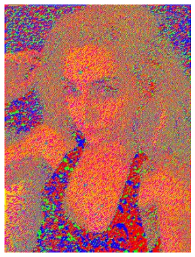


neon.txt

Normal

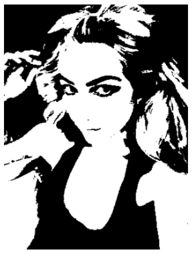


Dithered

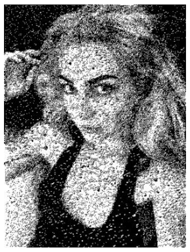


black-white.txt

Normal

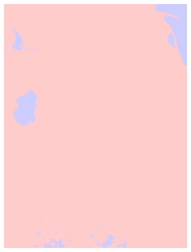


Dithered



pastel.txt

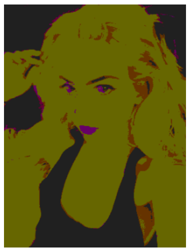
Normal



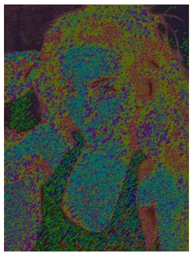
Dithered

dark.txt

Normal

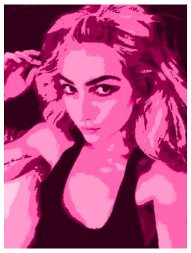


Dithered

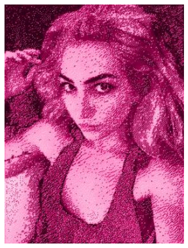


pink.txt

Normal

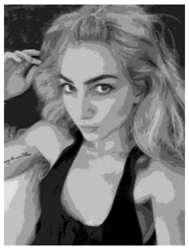


Dithered

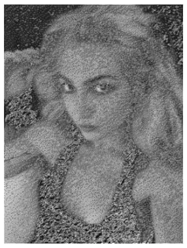


grayscale.txt

Normal

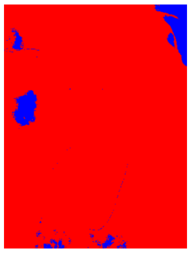


Dithered

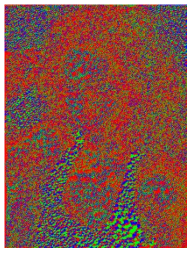


rgb.txt

Normal



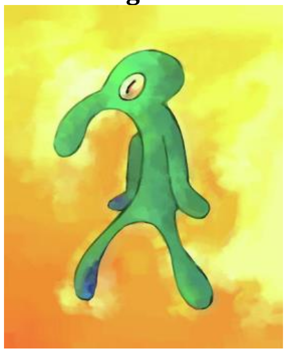
Dithered



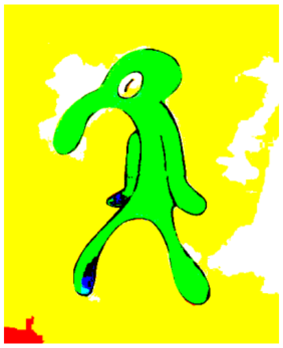
Observations

- While I have carefully reviewed and found no obvious issues with my implementation of the Floyd-Steinberg dithering algorithm, I obtained some interesting results that would require further study to fully explain. For instance,

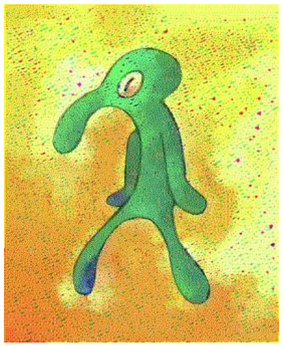
Original



Normal

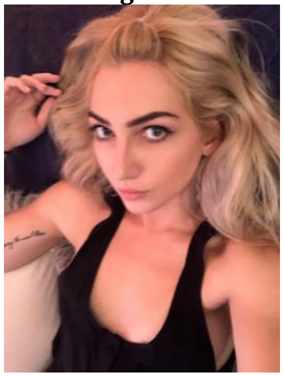


Dithered

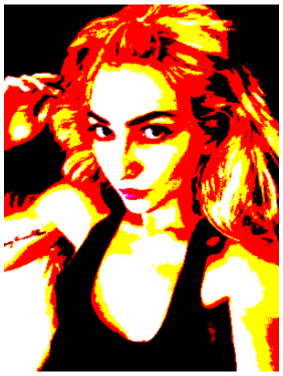


the dithered image of the palette, "3-bit.txt," most closely resembles the original, with more accurate representation of lighting and color distribution, but it also includes tiny splotches of random color. The dithered image of "3-bit.txt" is also most identical to the original "me.png," with features exactly matching those of the previous example:

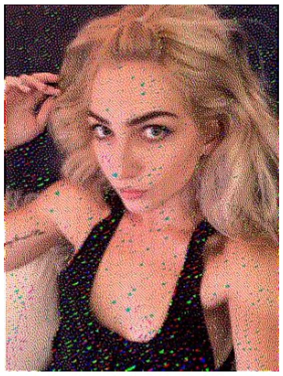
Original



Normal

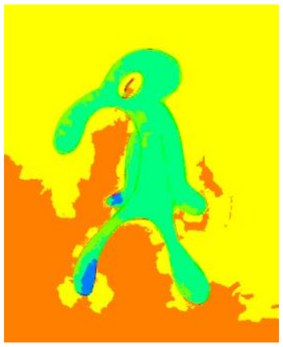


Dithered

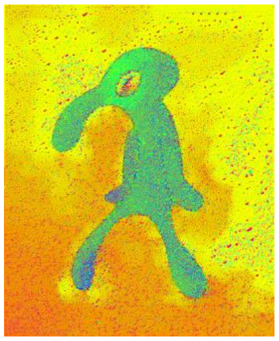


Similar results are shown in the dithered image mapped to the palette, "neon.txt,"

Normal



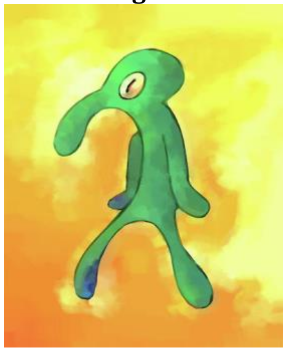
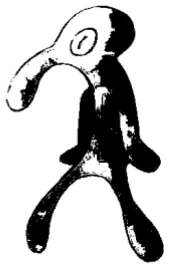
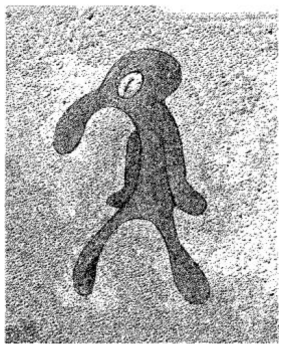
Dithered



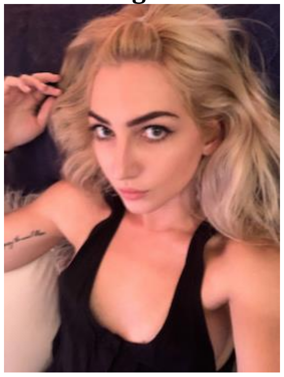
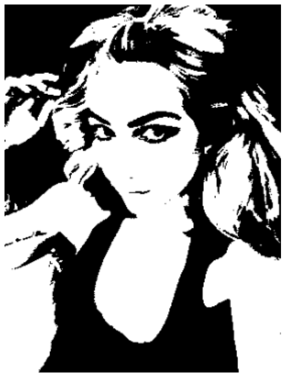
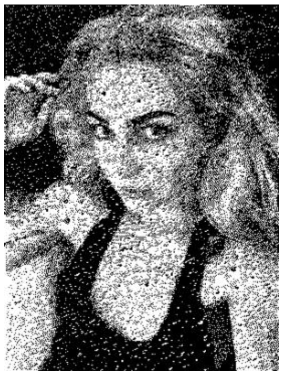
with a like pattern of strange-colored dots. The image is otherwise a close match to the original, the only difference from the previous example of "boldandbrash.png" being a greater contrast with the colors in the original palette.

- The “boldandbrash.png” images mapped to “black-white.txt” are some of the clearest and true to lighting for that image, whereas the “me.png” images mapped to that palette are some of the opposite comparison to the image:

Original **Normal** **Dithered**

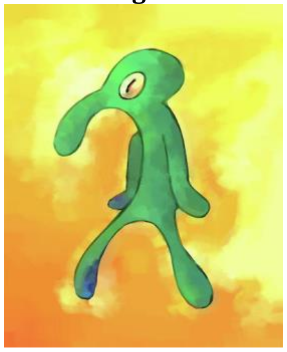
Original **Normal** **Dithered**

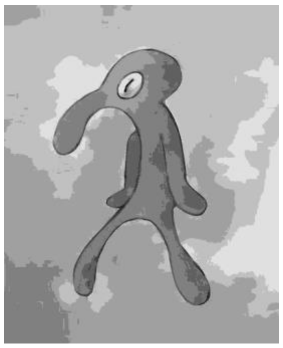
The normal mapped image for “boldandbrash.png” has accurate lighting detail for the figure, the only thing missing being the background, since the original background color is entirely closer to white than black. The dithered version is clear and includes exact details of the original. The normal mapped image for “me.png,” on the other hand, includes drastic lighting changes and little detail, and the dithered version is extremely grainy, masking details. This is probably due to darker background features in the original image as well as small, more detailed facial features with more intense color/lighting contrasts.

- The other more uniform palettes, namely “grayscale.txt” and “pink.txt,” are best with depicting the shading of the original images:

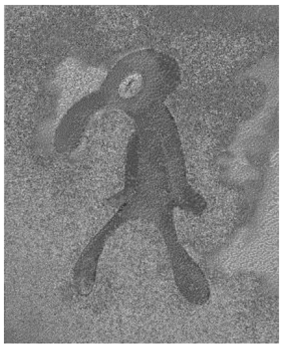
Original



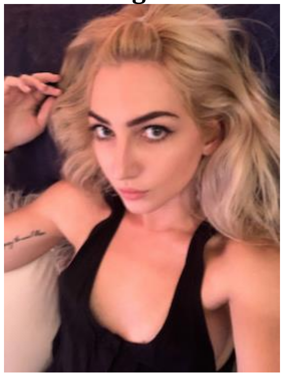
Normal



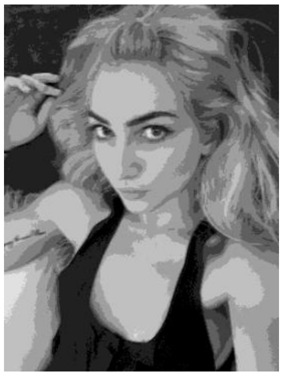
Dithered



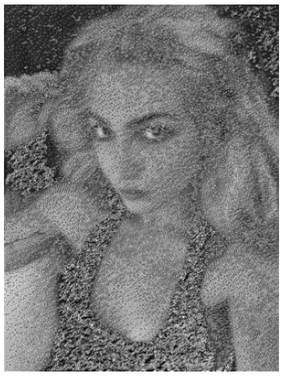
Original



Normal

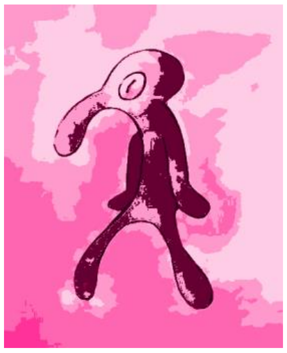


Dithered

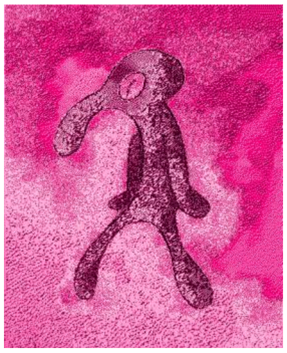


In both images mapped to “grayscale.txt,” the normal images are clearer and more accurately show the details of the original, while the dithered images are dark, grainy, and undetailed, especially at a distance. This is likely due to the similarity of colors within the palette, especially since the RGB values are the same for each color. Similar results are shown in the images mapped to “pink.txt,”

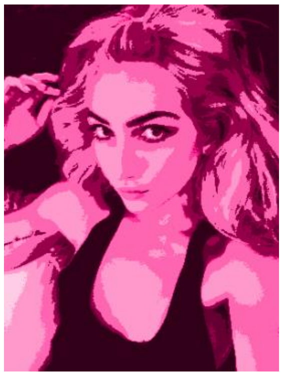
Normal



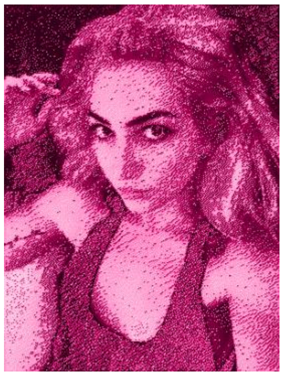
Dithered



Normal



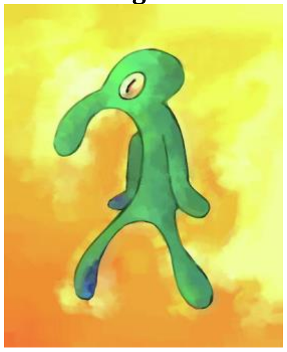
Dithered



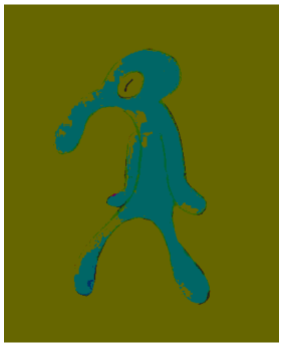
except the detail is undeniably clearer in the dithered image (perhaps since the RGB values are different across the colors). The shading in the dithered version of "boldandbrash.png," looks similar to that of the one mapped to "grayscale.txt," with greater clarity but less accurate lighting. The one mapped from "pink.txt" shows all of the details and contrasts of the original image, but the shading appears darker when it should be lighter and vice versa; this is evident in comparison to the normal version of this mapping. I noticed this phenomenon in both the dithered grayscale and pink versions of "me.png," where specifically the shirt is dithered in a way that makes it appear lighter than certain parts of the image that should be lighter. The shirt is the darkest part of the original image, and the normal image maps it to the darkest possible palette color. The dithered version mapped to "pink.txt" does show more detail in the shirt than the normal, but it does not have the stark contrast that the shirt renders in the original image.

- The remaining palettes did not translate as well with the images. The ones to which "boldandbrash.png" were mapped come by similar results to one another, respectively "dark.txt," "pastel.txt," and "rgb.txt:"

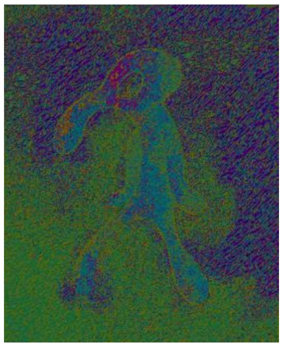
Original



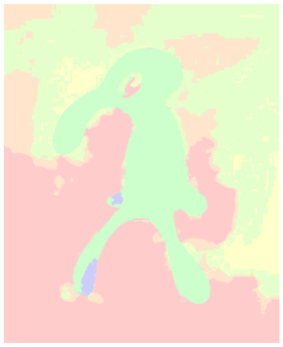
Normal



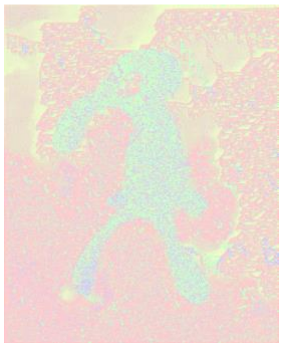
Dithered



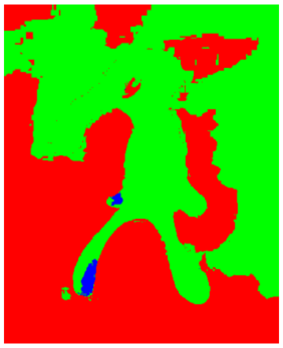
Normal



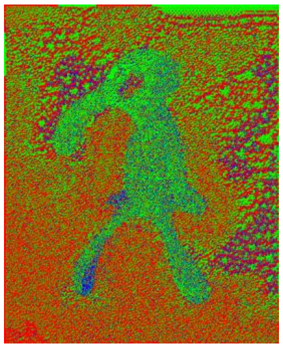
Dithered



Normal

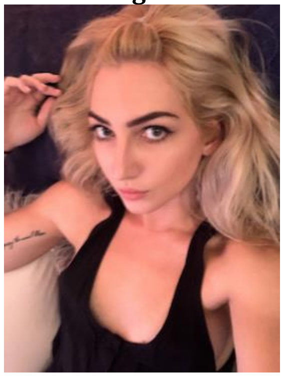


Dithered

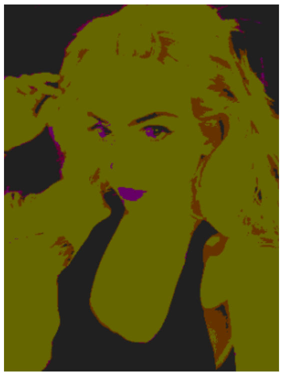


These palettes are the furthest from that of the original image, so they look the least like it; however, because of this the dithering is most effective at portraying the details of the original. The remaining palettes, “dark.txt,” “neon.txt,” “pastel.txt,” and “rgb.txt,” respectively, to which “me.png” is mapped produce images very dissimilar to the original,

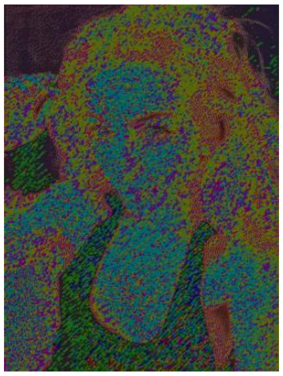
Original



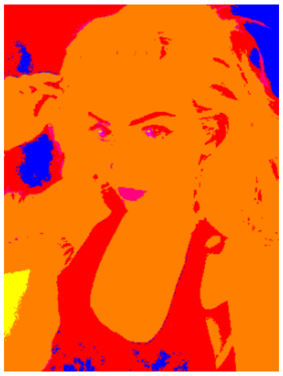
Normal



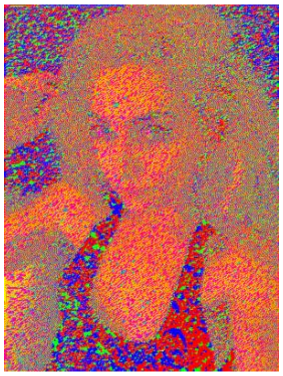
Dithered



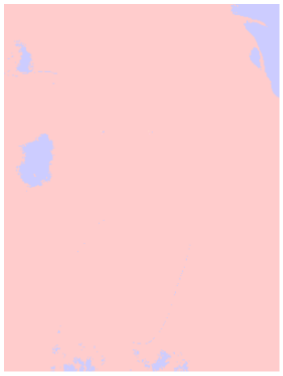
Normal



Dithered

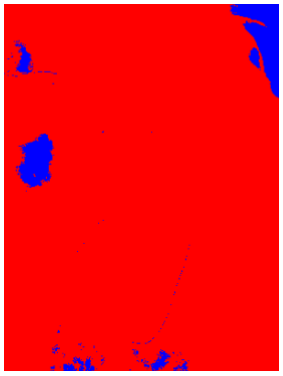


Normal

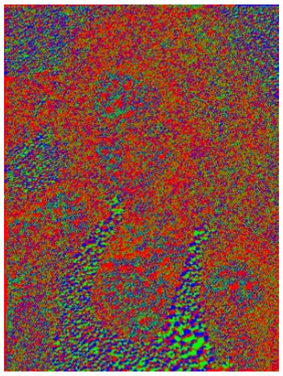


Dithered

Normal



Dithered

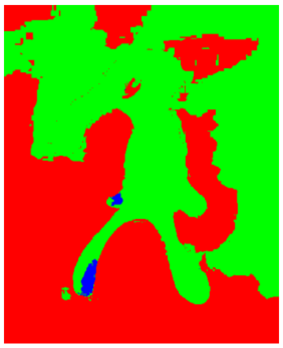


likely due to such limited colors and their contrast to the original palette. The images normally mapped to “dark.txt” and “neon.txt” are the clearest, only by displaying contrast for some of the important features. The ones normally mapped to “pastel.txt” and “rgb.txt” are nearly identical, the only difference being washed out colors in the pastel palette, and each pixel is only closest to one of two colors (a similar effect is also shown in “boldandbrash.png” normally mapped to “rgb.txt,” where it does not properly display the features of the original). All of these dithered images of “me.png” poorly distinguish any features of the original image, where some can be detected if looking closely. The skin tone is also completely mismatched except in the “neon.txt” image, where corresponding colors are at least discernable.

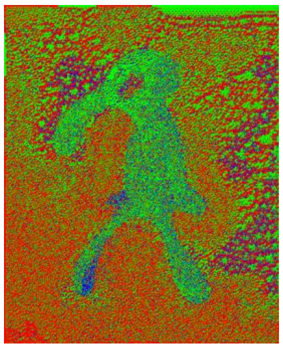
- Another interesting observation is how the ordering of colors in the palette file affect how the pixels are mapped to them. Occasionally there are colors in the original image that are equal distances between at least two colors in the palette. I wrote the code to loop through each palette color and save the one with the minimum distance from the original, but it will only save the current color if it is less than the previously saved color; therefore, if the colors are equal, the previously saved color will not change. I tested this by entering my own palette during execution that reversed the order of the colors in “rgb.txt:”

Original

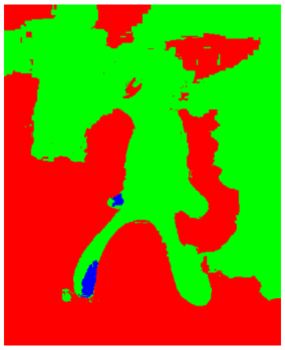
Normal



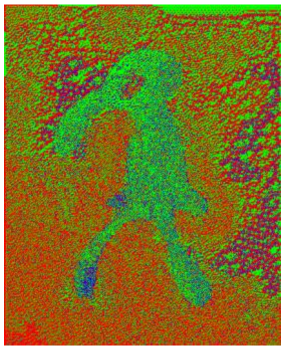
Dithered



Normal



Dithered



The above images are mapped to “rgb.txt,” and below are the same colors, ordered blue, green, and red. In the normal image mapped to the former palette, red takes priority over green, so there are a few more red pixels across the image. This effect is just reversed in the other normal image. Dithering eliminates this effect for the most part, as the two above images are nearly indistinguishable; there are just a couple of inconsistencies in the pixel pattern due to changes in error distribution.