EECS490_hw1_tkg11_Guen

1 -1. Convert to Grayscale

Motivation

It is common to want to convert to grayscale. Converting to gray scale can be used for printing color images in black and white or for preprocessing images before performing analysis on them. This can reduce overall computation time as well as simplify some processes that may work better with black and white. It is important, though, especially in the printing case, the the grayscale image sitll be an accurate representation of the original in shape, contrast, and saturation.

Approach

The algorithm iterates over all the pixels in the original image, extracts the rgb values, and converts them to grayscale using the formula $v = 0.299 * r + 0.587 * g + 0.114 * b$. These values are placed into a grayscale image of the corresponding size at the corresponding pixel.

Results

Discussion

The results are as expected, giving a clear and accurate representation of the original image in grayscale

1-2. Embedded Image

Motivation

Another common procedure, images are often watermarked for copyright. it is important that the algorithm work well so that the overlay is attractive and appears to include only the foreground of the watermark in the final image.

Approach

This algorithm works by first identifying a threshold value that considers a pixel a background or foreground pixel in the watermark. It is clear that the background of the watermark is white while most of the foreground is blue (or dark gray in grayscale), so the threshold value was picked to be 200, which is close to white, but encompasses values that are not 100% white (at 255). The algorithm then shifts to where the top left corner of the watermark should be in the final image, when the watermark is centered. It iterates over every pixel in the watermark, and if that pixel is background, leaves it as the pixel value from the original image, and if that pixel is foreground, replaces it in the final image with the pixel value of the watermark value.

Results

Discussion of Results

The results look good, although not great. The low resolution of the watermark and base image, as well as the thin edges on parts of the lettering make it so that the watermark blends in a little bit with the similar colors that also exist around the center of the base library image. Even though it appears that the background is correctly identified from the foreground in the watermark, these unavoidable circumstances make the watermark a little hard to distinguish in the final image. Overall, the procedure seems accurate though. Results for color and gray both have these qualities.

1-3. Negative of Image

Motivation

My friend wants a negative image because it looks cool, but also producing negative images can be useful when one wants bright parts of an image to appear dark and dark parts to appear light. This can be for a preprocessing step for an algorithm or simply to help with visual analysis.

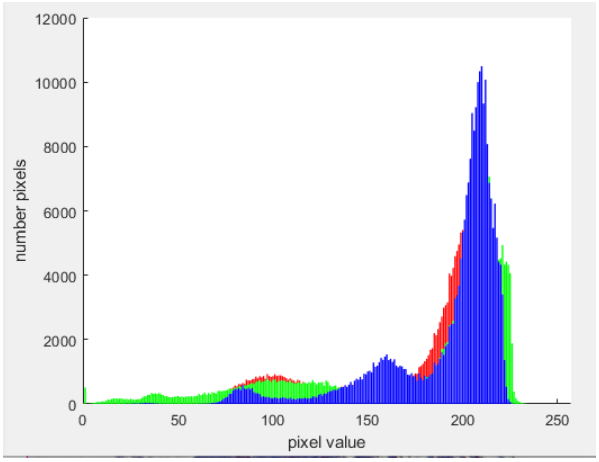
Approach

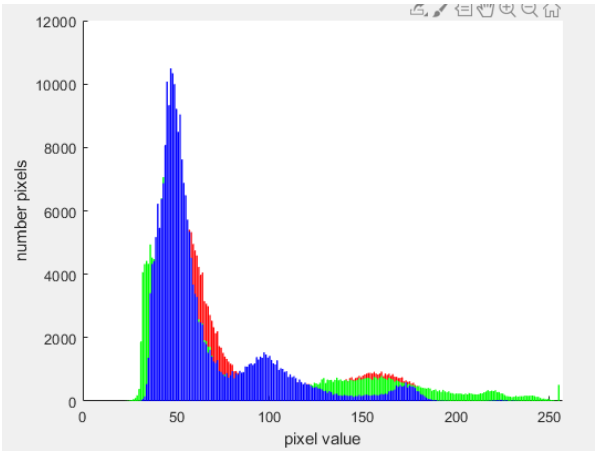
The algorithm iterates over each pixel in the original image and applies the formula: $v(f) = 255 - f$, where f is the current pixel value.

The histogram function works by separating taking a grayscale (1-channel) image and iterating over the entire image. It counts the number of pixels that appear at

each amplitude value (0-255). The color histogram is simply the histogram of each channel in the image plotted on top of one another.

Results





Discussion

Comparing the two histograms, we see the expected mirroring of the histogram, since the operation is 255-f. The majority of the pixel values in the original are up near 175-225, in line with the overall whiteness of the image. The negative histogram, with most values residing in the 25-75 range is indicative of the overall darkness of the negative image. You can clearly see why whites in the original look dark in the negative. However, because of the linear nature of the operation, the distribution of pixel values across the channels remains the same, so that the overall saturation and contrast remains the same between the two images, although the hues change.

2-1-1. Rose Dark Full Linear Scaling

Motivation

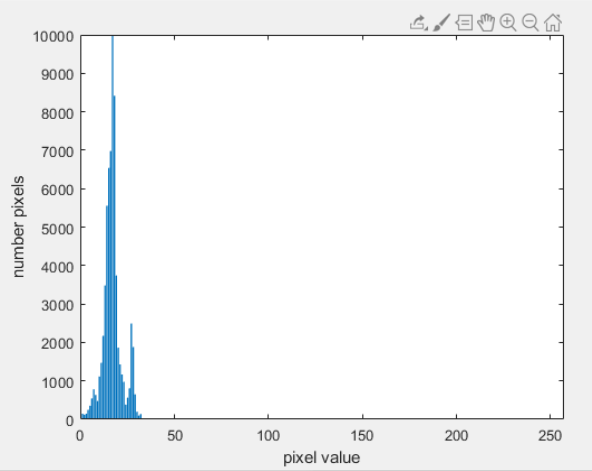
The given image is incredibly dark, and it is almost impossible to see the contours of the image. We want to increase the contrast of the image so that it can viewed or analyzed. We use full range linear scaling so that we get an accurate representation of the image while also making use of the full 0-255 dynamic range available to us.

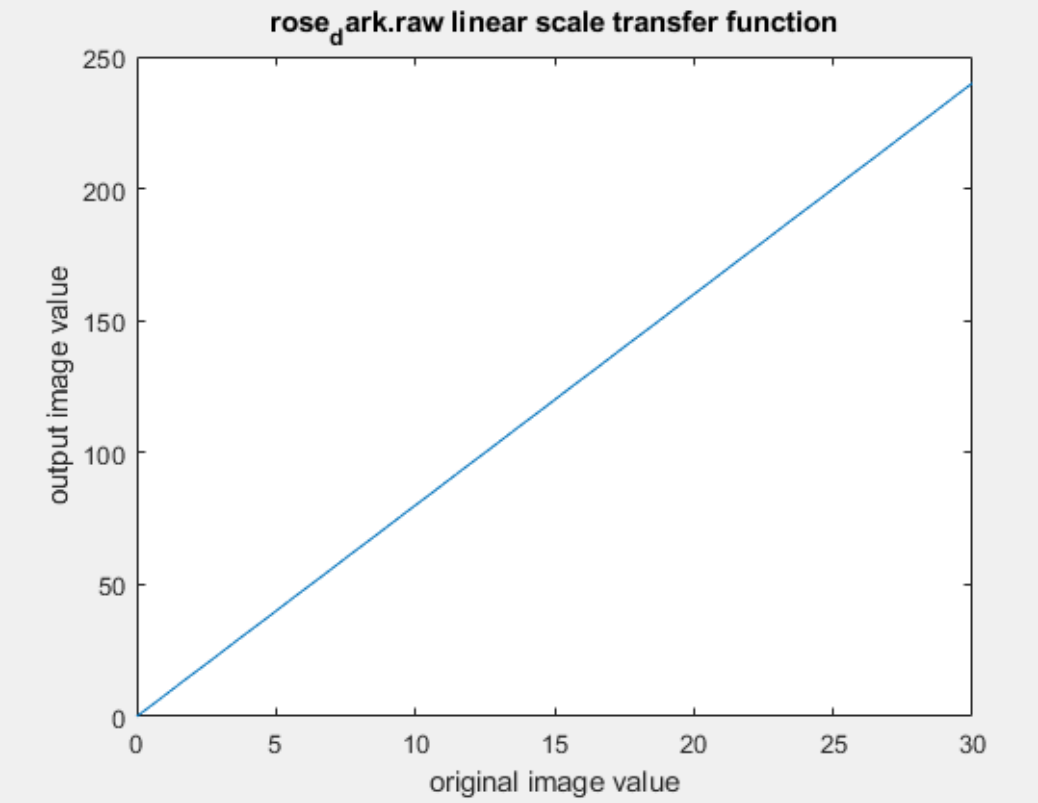
Approach

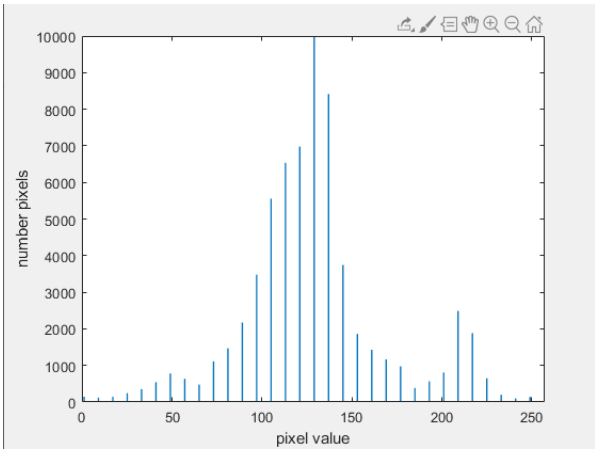
The approach here consists of finding $fmax, fmin$ being the maximum pixel value and the minimum pixel value in the original image respectively. Since this is a full linear scale, we want to scale the values across the entire dynamic range (0-255), so we also pick values $gmax = 255,\ gmin = 0$. Then we iterate over all of the pixels in the image and apply the linear scaling function $v(f) = gmin + (gmax - gmin)/(fmax - fmin) * (f - fmin)$.

The plot of the transfer function works just by creating a vector sample of pixel values from fmin to fmax and applying the transfer function to that sample, graphing the output on the Y-axis and the sample on the X-axis. *Note: All subsequent transfer function plots (including histogram equalization plots) work in the same way, only with different functions and constants)*

Results


number pixels
pixel value
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
50
100
150
200
250


rose_dark.raw linear scale transfer function
output image value
original image value
0
50
100
150
200
250
0
5
10
15
20
25
30



Discussion

The final image looks good, with certainly much higher contrast as compared to the original image. However, it does not show "enhanced" contrast across parts of the image that were originally close together in value. For example, the petals all appear to be a somewhat uniform color. This is to be expected, since, as shown in the histogram, the transfer function simply distributes the pixel values from the smaller range of the original image across the full range of the final image. It doesn't affect their distribution at all. Clipping could have been employed here to give an even higher contrast image, since it is clear that there are not many values at the high and low end of the final histogram. If we had removed these values, and distributed the remaining across the entire dynamic range, then the contrast would have looked even more apparent.

2-1-2. Rose Dark Histogram Equalization

Motivation

Again, the dark image is very dark and we want to increase contrast. However, the linear scaling method doesn't distribute the gray values evenly enough to give the higher contrast that may be necessary. The linear scale still suffers from the poor distribution of values shown in the original. We use the histogram equalization to distribute the gray values more evenly across the image.

Approach

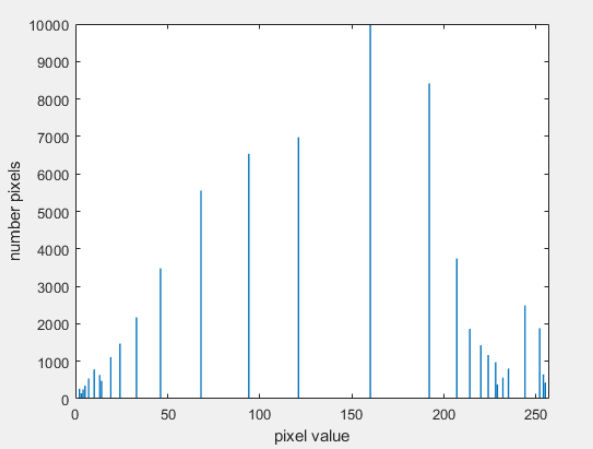
See histeq.m for code and more comments on procedure.

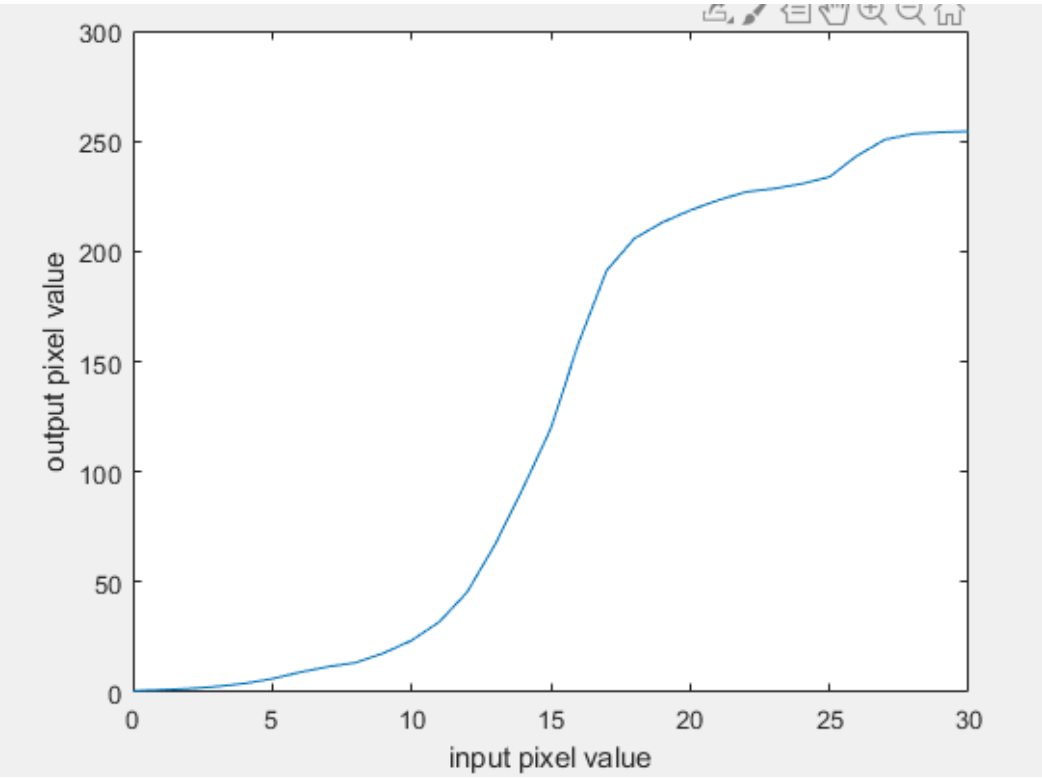
The algorithm for this calculates the histogram of the image and normalizes it by dividing each histogram value by the total number of pixels in the image. It then computes the cumulative distribution of this histogram, by applying (roughly) the formula $c(i) = c(i-1) + h(i)$ where c is the cumulative distribution, and $h(i)$ is the histogram value at pixel value i, over the entire range of the histogram. It then iterates over each pixel in the image and applies the formula $c(f) = (gmax - gmin) * c(f) + gmin$, being the transfer function to produce a uniform distribution of the final histogram.

For an explanation of the graphed transfer function see 2-1-1 Approach

Results

For original rose dark image and histogram see 2-1-1 Results





Discussion

The final image looks good, although perhaps unrealistically high contrast. It is hard to tell in grayscale though, since it is not often that I see a rose in grayscale. As compared to the linearly scaled image, you see much more contrast in the histogram equalized image for values that were originally close together. The petals in this case have very distinguishable gradients, and the depth of color change between bright and dark parts of the image looks much greater. This makes sense from the histogram since the histogram displays (as compared to the linearly scaled function) a much greater number of pixels distributed across to the high and low ends of the dynamic range, giving more low range darks and more high range whites as compared to the linearly scaled image. The transfer function also shows why there appears to be a more stark contrast for the large number of pixel values that were originally in the mid-range, since there is a clear inflection in the graph, showing about an 80% increase in value in the output dynamic range for only about a 15% increase in the input value.

2-2-1. Rose Mid Full Linear Scaling

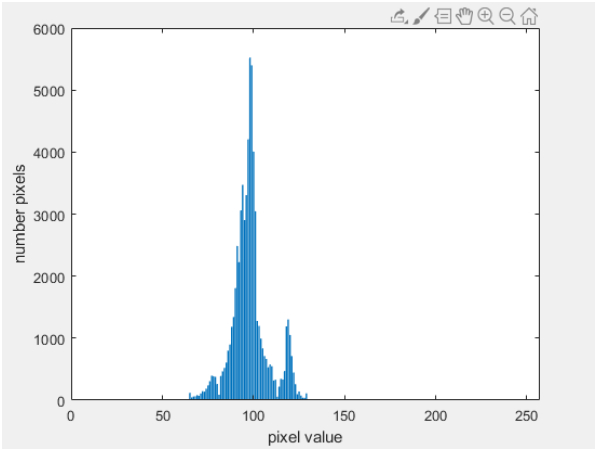
Motivation

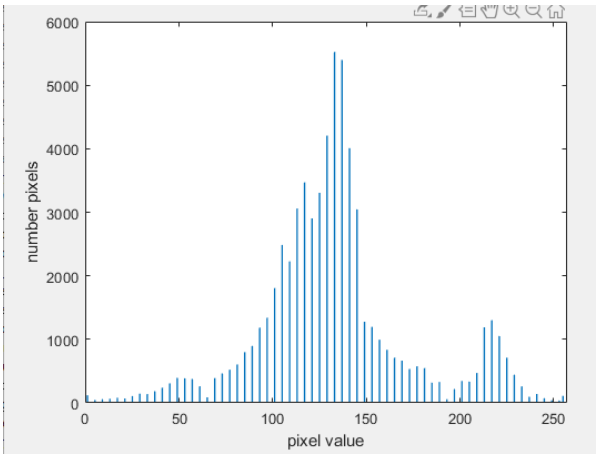
Very similar to 2-1-1, the image is of low contrast. Although more visible here, the contours are still poor.

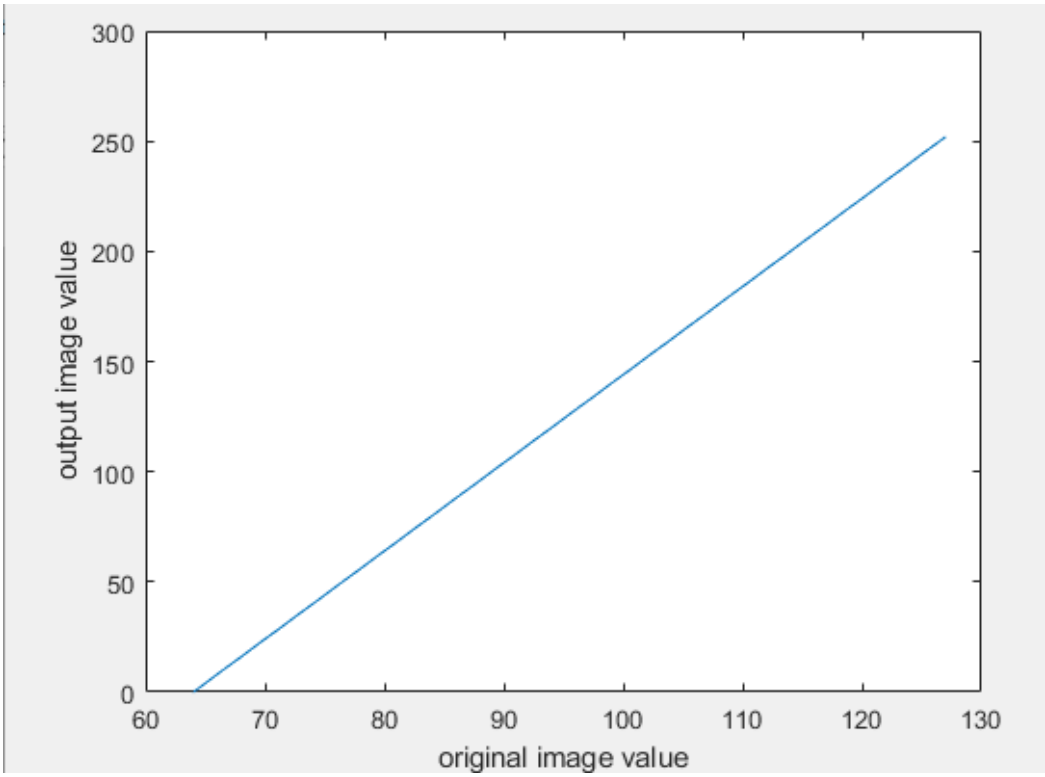
Approach

For full linear scaling approach see 2-1-1 Approach

Results







Discussion

For discussion of the general effect of linear scaling on an image see 2-1-1 Discussion.

Comparing this mid rose after full linear scaling to the dark rose after full linear scaling shows the a very similar image. This is to be expected, since the two images are very similar (as can be seen in the histogram), the only difference being that the values in this mid-range image span a slightly larger area of values as well as holding, on average, higher values. The values of the dark image were basically shifted up and spread out a bit. We see that the final histogram of this image seems to have "a higher resolution" since the original histogram showed a wider distribution of values. This gives a slightly higher contrast image, but in reality this difference appears somewhat negligible, since both images are similar and both have been mapped across the same range. The transfer function looks as expected, having a linear mapping from fmin-fmax to 0-255.

2-2-2. Rose Mid Histogram Equalization

Motivation

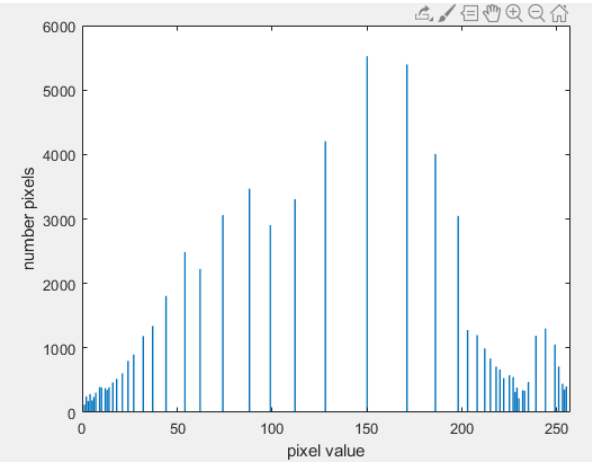
The same as for 2-1-2.

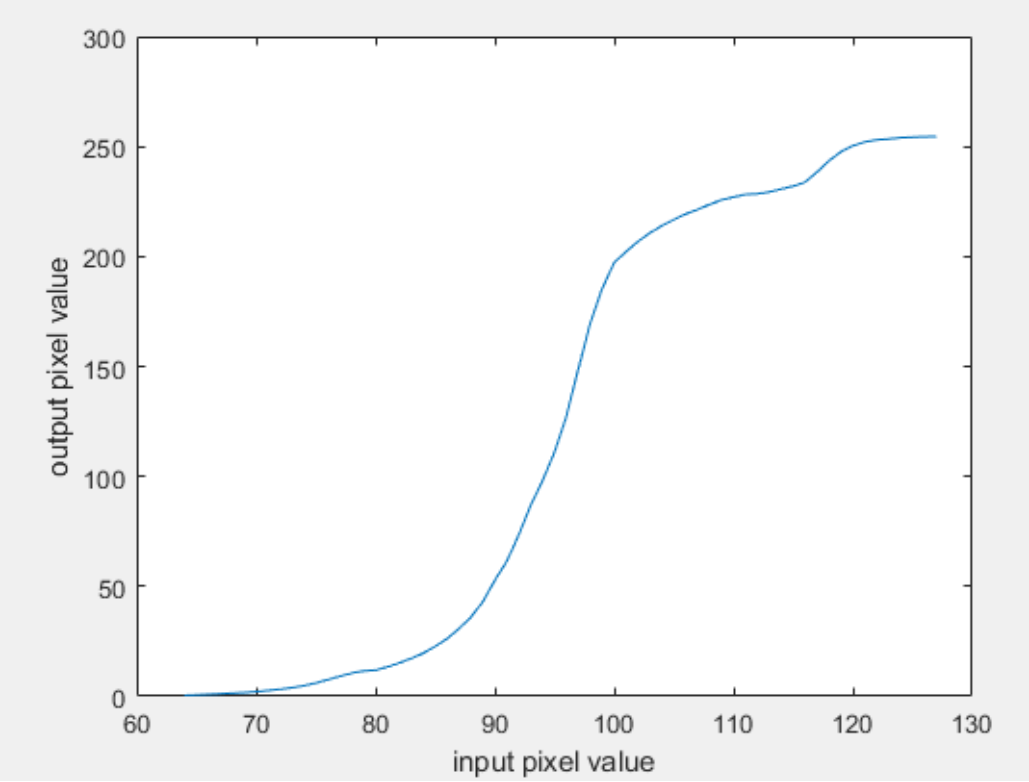
Approach

For histogram equalization approach see 2-1-2 Approach

Results

For original rose mid image and histogram see 2-2-1





Discussion

There is not much additional discussion to add here. Similar to the linear scaling method, we see a slightly higher "resolution" in the final image histogram as compared to the dark image due to the better distribution of values in the original image. The transfer function looks about the same as the dark image histogram, however its inflection is a little less steep due to the wider distribution of values of the original image. The photo looks good and high contrast, about the same as the dark rose.

2-3-1. Rose Bright Full Linear Scaling

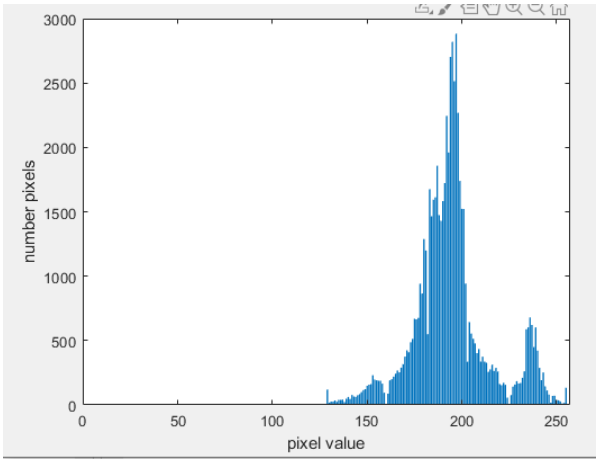
Motivation

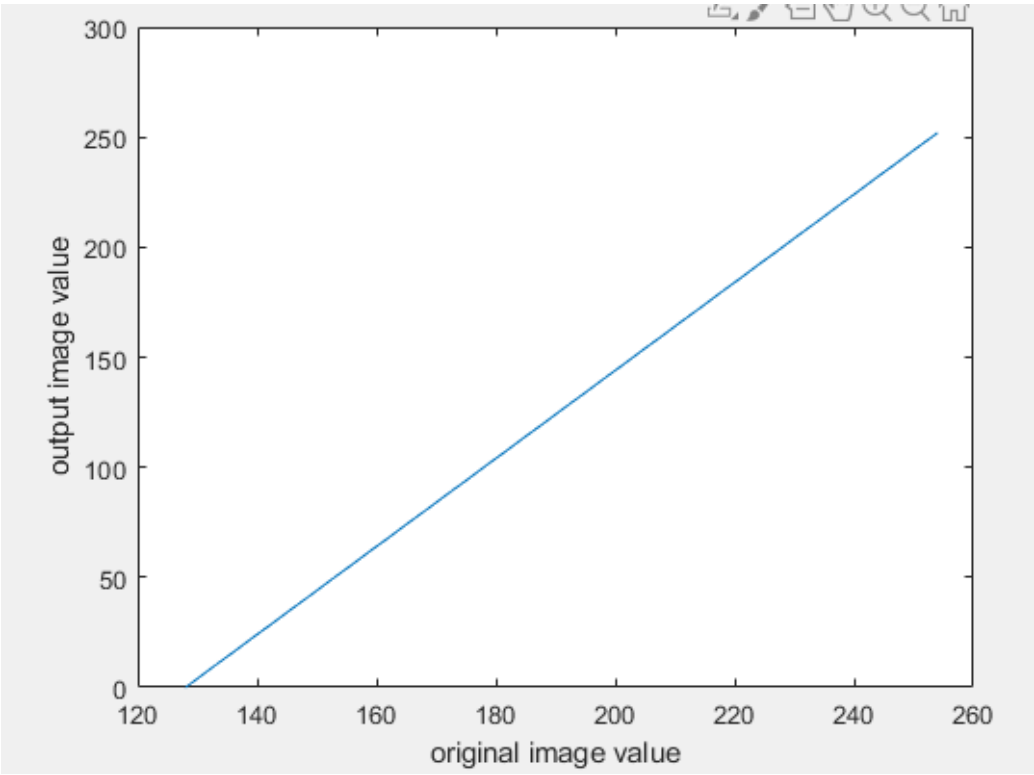
This image probably has the best base contrast, but it is very washed out and is just too bright. We want to increase that contrast and distribute the values more evenly across the dynamic range.

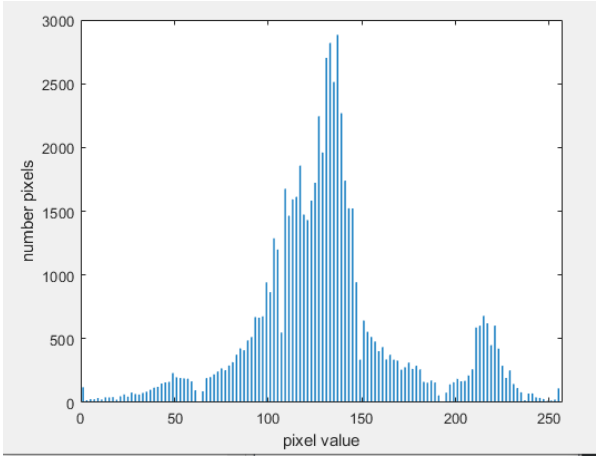
Approach

For full linear scaling approach see 2-1-1 Approach

Results


number pixels
pixel value
0
500
1000
1500
2000
2500
3000
0
50
100
150
200
250


output image value
original image value
0
50
100
150
200
250
300
120
140
160
180
200
220
240
260



Discussion

We see the same phenomena here that was observed between the rose mid and rose dark images. This image, rose bright, again shows a slightly wider distribution of pixel values, and as a result shows a higher final histogram "resolution", which shows slightly higher contrast. There appear to more values in the darker and lighter parts of the image as compared to the other two. The transfer function is, as expected, a linear correlation between input and output.

2-3-2 Rose Bright Histogram Equalization

Motivation

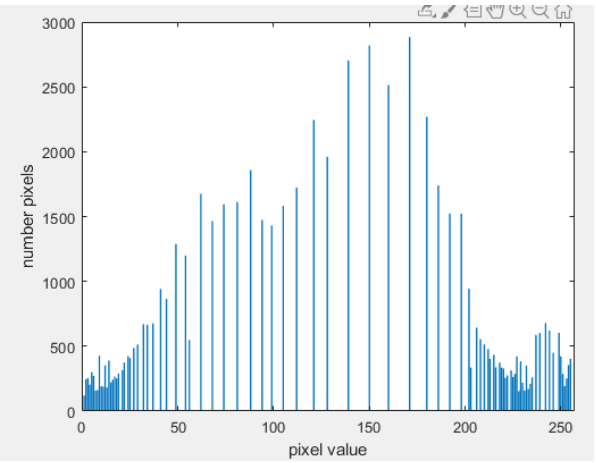
Same as 2-1-2 and 2-2-2

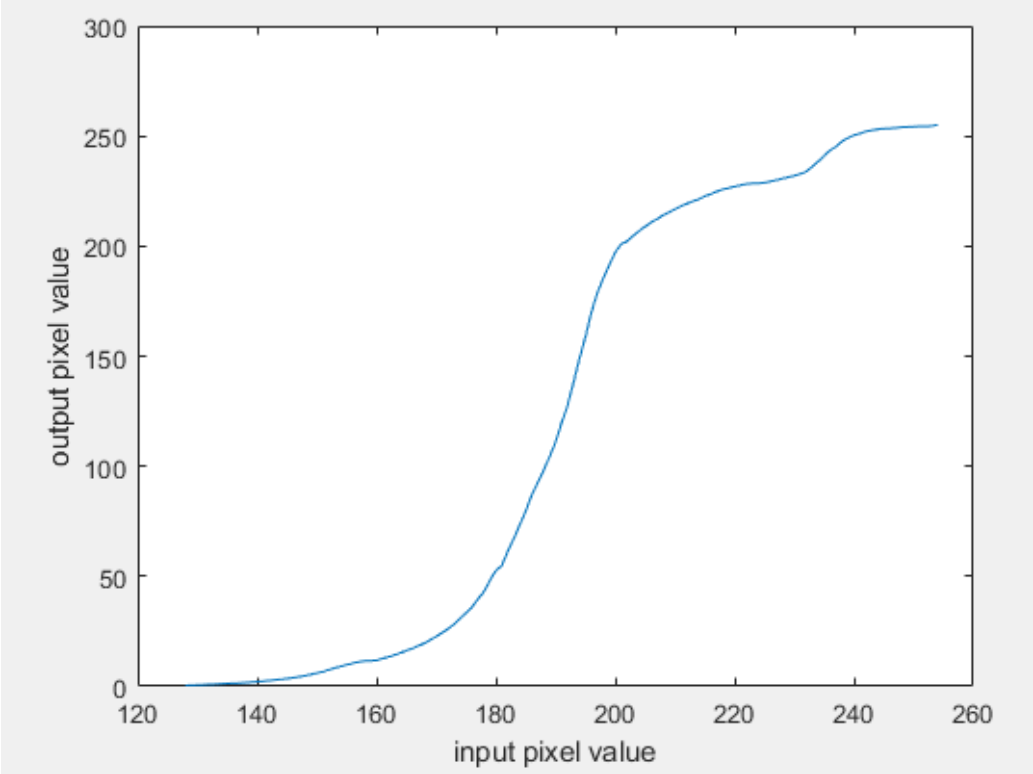
Approach

For histogram equalization approach see 2-1-2 Approach

Results

For original rose mid image and histogram see 2-3-1





Discussion

Once again, we see higher contrast as compared to the linear scaling method. We also see that rose bright has a slightly better distribution across the entire image, giving slightly better contrast. It also shows a slightly less steep inflection in the transfer function when transitioning from dark to light values.

3-1-1. Gray Uniform Noise Removal

Motivation

The image given is noisy and unappealing to look at. It is difficult to tell what the true grayscale value should be. We want to remove the noise in hopes of giving a more true-to-original image that is more visually appealing and may help with other image processing.

Approach

The approach here was to use a linear noise cleaning method with a mask because the uniform noise has a high spatial frequency, so applying a per-pixel mask across the entire image should work well. Even though it will smooth over the image, the consistency of the noise means that we won't be modifying parts of the image that don't have noise (since the whole thing has noise).

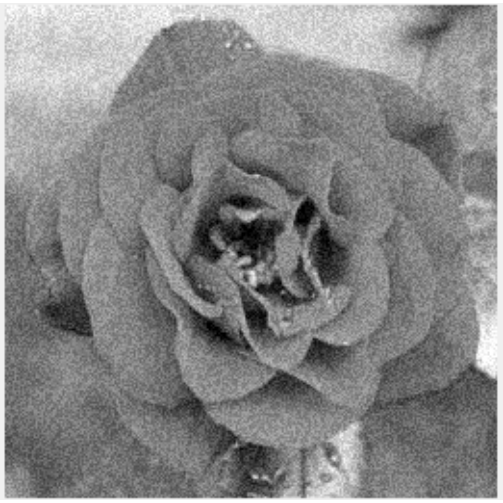
The algorithm works using an $N \times N$ mask and convolutions. First, the original image is expanded by $floor(N/2)$ pixels on the top, bottom, left, and right, with those new pixel values taking value 0. This allows the mask to have defined values when looking at pixels near the edges of the image. The mask is then moved along the image as a sliding window. For a pixel in the image, the mask overlaps its center on the pixel being evaluated. It then takes the subimage of the original image that is overlapped by the mask and calculates a new output value for that focus pixel of the original image. This new value follows the equation $\sum_{i=1}^{N}\sum_{j=1}^{N} M(i,j) * S(i,j)$ where $M(i,j)$ is the mask value at pixel (i,j) and $S(i,j)$ is the subimage value at pixel (i,j), relative to the top left corner of the mask. This operation is repeated across the entire image until completion.

For the results shown below, the mask $\frac{1}{16}[1,2,1;2,4,2;1,2,1]$ was used, as it subjectively appeared to reduce the noise the most as compared to other masks like $\frac{1}{9} * [1,1,1;1,1,1;1,1,1]$ from the textbook secion 10.3-2 to 10.3-3

Results

Original

Noisy Original



Denoised Original

Discussion

The results of this look pretty good. The uniform noise is sufficiently removed, and the final image appears to have much more uniform shading as compared to the noisy original. However, as is to be expected with linear noise filtering, some of the detail is lost in smoothing over the image to get rid of the noise. Even compared to the noisy original, the crispness of the edges is not very well maintained. However, the use of a 3×3 mask was an attempt to minimize this, and the balance here between bluriness and denoising seems good. Compared to the original image, the final is much worse, having lower contrast, clearly still having noise, and less crisp edges. This is to be expected though, since we are simply smoothing the image, not trying to reconstruct or predict what the original image values were pre-noise. A thing to note is that the use of the 0 value as padding for the convolutions means that the border of the image is darker than the rest. This could be mitigated by copying rows and columns or by using an average pixel value instead of 0.

3-1-2. Gray Gaussian Noise Removal

Motivation

Similar to the uniform noise removal, the image here is noisy and unappealing to look at. The noise may also make some other image processing tasks more difficult. We want to denoise it.

Approach

Again linear noise removal was chosen for similar reasons as in 3-1-1. Gaussian noise is similarly evenly distributed, and has a high spatial frequency that is conducive to convolutions with masks.

For linear noise removal approach see 3-1-1 Approach

The mask used was the same as in 3-1-1 and was chosen for its subjective quality of denoising without blurring too much.

Results

For the original, noise-less image see 3-1-1 Results

Discussion

Compared to both the noisy, the original, and the uniform denoised, this one looks okay. The lack of perfectly uniform noise (this being Gaussian noise) means that applying a static mask the way that we did does not provide quite as high quality noise removal. We see blurring of the image, and significant noise reduction, but not as much noise removal. Much of the edge crispness is lost. The same problem with dark edges from 0 padding occurs here as well.

3-2. Color Noise Removal

Motivation

Similar to the grayscale noise removal, this color image is actually very difficult to look at, and has large fluctuations in color from pixel to pixel. This would be a difficult image to use for most applications. We want to denoise it so that it looks better and is easier to use.

Approach

The approach here was to use non-linear noise removal techniques because, as can be seen in the Results section for the red, green, and blue channels, the noise was not well distributed. This is more akin to salt and pepper or impulse noise. There are clear shapes and contours, as well as areas of high and low noise density. We want to use non-linear noise cleaning here then, so that we can get good noise smoothing, but also retain some of that finer image detail and contour.

The algorithm works using a pseudomedian filter (although a median filter would have worked as well — or better — we chose the pseudomedian for its efficiency). First, the image is broken down into its red, green, and blue channels. The pseudomedian filter is then applied to each channel individually. Once filtered, the channels are combined again into the final rgb image.

The pseudomedian filter works by essentially using a sliding window of size $N \times N$, N is odd, that centers itself on each pixel in the image. For each pixel, it creates a "plus-sign" by creating a vector of pixels across the x and y axes, X, Y each of length N , and each having the middle value as the pixel being focused on. It then computes the pseudomedian for this plus shape following the equation:

$$\frac{1}{2} * max(maximin(X), maximin(Y)) + \frac{1}{2} * min(minimax(X), minimax(Y))$$

where maximin is the maximum of the minimums across $L - M$ iterations of a sliding window of size $M = (L + 1)/2$ over the input vector, where L is the length of the input vector. Minimax is the same except it takes the minimum of the maximums across the sliding window.

The value returned from this equation is then the pixel value of the output image.

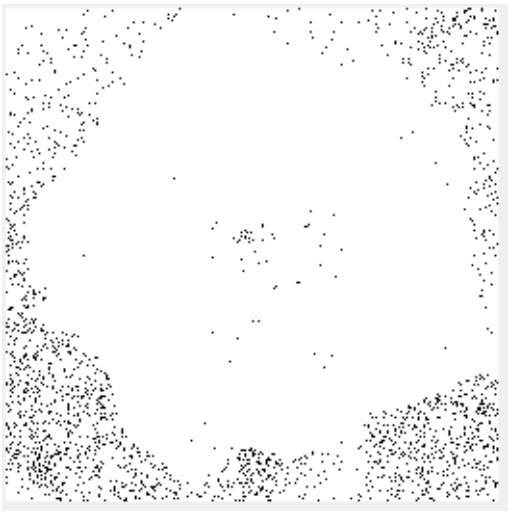
Results

The result below was computed with a window size of 5×5

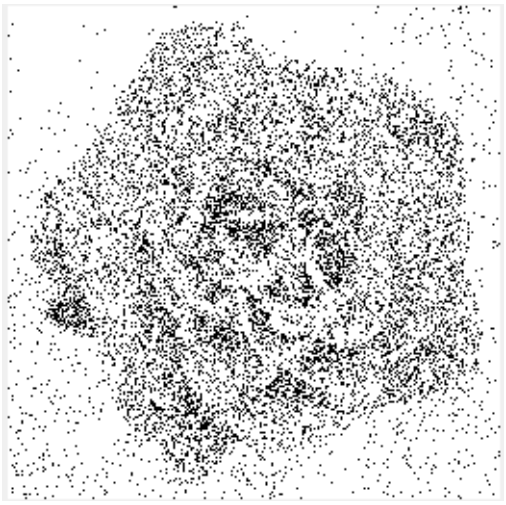
Original

Noisy Original

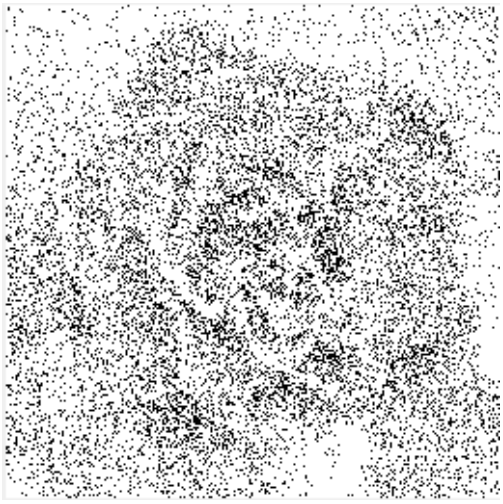
Red Channel:



Green Channel:



Blue Channel:



FInal Output:

Discussion

The results of this denoising procedure are ok but not great. What is good is that the edges, especially those of the outermost petals are still well defined. There is a clear dividing line that looks good visually. The denoising procedure, however, is not perfect. Although it certainly looks much better than the original noisy image, it appears that the noise reduction results in the noise taking on more generic shapes, as opposed to the single points of impulse noise from the original. This gives a better look more through blurriness than actual color accuracy. Comparing the result to the original image, the colors are still quite distorted, and the image is still rather noisy.

This is a difficult problem to solve though, and is to be expected since looking at the rgb channels of the noisy image makes it clear that certain areas, particularly the background, have low densities of each of the colors. This is what causes the very bright spots of color that are apparent in the background. It can be seen that the areas of the image with denser points across all three channels show better denoising and color accuracy in the final image, since they have more of a blend of the three colors. The denoising of the rose looks better than the denoising of the background for this reason.