Image Processing – A1, part 2

# Improving Image Contrast

The topic I chose for part two of this assignment, was contrast enhancement. The three algorithms I chose to use where: morph\_CE, morph\_toggleCE, and histhyper. I also created a simple algorithm through some experimentation which I called drew\_CE.

I performed a little bit of a twist on this part of the assignment, as instead of performing the contrast enhancements on grey scale images, I decided to apply these algorithms to colour images. To apply these algorithms to colour images required converting colour images from RGB to HSV, and then applying greyscale algorithm to the H component of the colour image. Next the enhanced image are converted back to RGB so it can be saved normally.

# Discussing the Algorithms

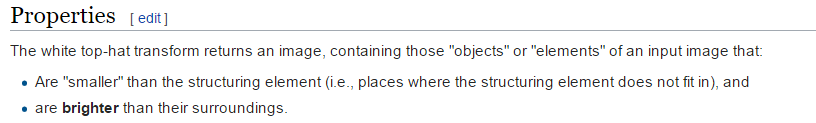
Each of the algorithms I used will now be gone through, starting with morph\_CE.

## morph\_CE

The morph algorithm creates two additional versions of the original image, and then merges these new images directly with the original image using simple addition and subtraction.

The first of the two images is created by using an opening top hat algorithm applied to the original image, and the second of the two new images is created by applying a closing top hap function to the image.

What do the top hat functions do? A white top hat transformation takes the surrounding pixels, and makes the current pixel brighter than those surrounding pixels.



Closing Top hat does the opposite by creating an image with “elements” that are darker than their surroundings. So the openth and closeth functions are creating a darker and lighter version of the image, using a structuring element to determine pixels taken into consideration to determine what happens to the current pixel being modified.

|  |  |  |
| --- | --- | --- |
| Opening Top Hat White top-hat has the effect of making light colours surrounded by light colours even lighter without big impact on darks | [Openth Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUxpPrxN-oJvq-YRQ) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) |
| Closing Top Hat black top-hat has the effect of making dark colours surrounded by dark colours even darker without big impact on lights. | [Closeth Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUyVSR92kLl43VKdg) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) |
| Morph\_CE By combing the original with the darkened, and lightened images, the darks become darker, and the lights become lighter improving the images contrast. | [Morph\_CE](https://1drv.ms/i/s!As-TTArLWDqlhoUzgeg-D9Fgko1R_g) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) |

Darkening the dark parts of the image, lightening the light parts of the image, and then combining the results seems like a good idea, however the result on my demo makes the eyes look eerie. In fact for the above demo image, I think that the closing top hat alone applied looks better than the full morph\_CE variant.

## morph\_toggleCE

There morph toggle mixes two parts: an eroded version of the image, and a dilated version of the image.

The gist of the morph toggle algorithm is to create a disk structuring element that determines what pixels are considered when determining the effect on the current pixel. Using the disk structuring element, two versions of the original image are created. First: an eroded image, and second a dilated image. Then for each pixel, the difference between the original pixel, and the eroded/dilated pixel is calculated. The pixel with the smaller difference from the original pixel is chosen.

|  |  |  |
| --- | --- | --- |
| Erode Erode, as its name intuitively suggests, erodes the image. This darkens the dark areas, even the dark areas that border on white areas interestingly. It also makes the image appear “fuzzy”. | [Eroded Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUWMhVH3R9pL6pBZg) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) |
| Dilate Dilate does opposite of erode, meaning dilate lightens the image. Its seems to fairly aggressively eat away at dark areas surrounded by very light areas. | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\contrast1 - dilate.jpg  [Dilated Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUVKvjCIsWf-innEQ) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\contrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) |

From the above examples, it is easy to see that neither erode nor dilate gets us closer to improving the original images contrast in a good-looking manner. So somehow, morph toggle blends these two images to get much better results. The way morph toggle blenders the two images is by comparing the pixels from the two images against the original, and choosing the pixels that are closest or “most like” the original image pixel.

By taking the pixel closest to the original images pixel, some of the darkening benefits of erode can be gained, while having that darkening rereigned in by the dilated version in the lighter areas of the image. A battle and balance between dark and light if you will.

|  |  |  |
| --- | --- | --- |
| C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1 - less like original.jpg  [Pixel most different from original](https://1drv.ms/i/s!As-TTArLWDqlhoUYLXnmDuSz-2QDUw) | [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1 - morph_toggleCE.JPG  [Pixel least different from original](https://1drv.ms/i/s!As-TTArLWDqlhoUX4vtiRKNh-GEZBg)  (AKA. toggle\_morphCE) |

## Histhyper

To be honest, I get the general idea of this algorithm, but I do not understand the math that goes into calculating the transformation factor.

|  |  |
| --- | --- |
| Image result for i'm afraid we'll have to use math | *“I’m afraid we’ll have to use…*  *MATH”* |

|  |  |
| --- | --- |
| 1. The image is split into “bins” by the histogram function. Each bin condenses the y range of the image to a single histogram value, for each given subsection of the image. 2. Each histogram value / bin value is gone through and used to calculate a transformation value for all 256 shades of grey. 3. Each pixel is gone through, and has the adjustment / transformation value applied to it. |  |

|  |  |  |
| --- | --- | --- |
| HistHyper When testing HistHyper, it did have very good contrast enhancement results, but it seemed to cause a decrease in quality. In some images, this trade off is worth it.  This image is an extreme example, being mostly white. In this extreme example, the contrast has been massively improved, but the quality of the image is much worse. | [HistHyper Link](http://i.imgur.com/rrs7vhT.jpg) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\contrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) |

## drew\_CE

This algorithm is very simple. Go through each pixel, and if the value is greater than 255/2 or 127.5, then increase it, and if the pixel value is less than 127.5 decrease it. The idea being to make darks darker, and lights lighter.

How much does this algorithm adjust the pixel value by? Half of the difference between the current pixel value, and 127.5.

This algorithm is a result of me spending several hours trying to come up with a contrast improvement algorithm without having background knowledge, or any idea what I was doing. This was one of the first methods I tried, and it was pretty much the only method that worked. I thought that if I spent several hours on this and was able to create my own algorithm, it would help me get a more intuitive sense of how the more complicated algorithms work. *Nope, its magic, the people who came up with the other algorithms are wizards.*

The advantages of this algorithm are that: it is very simple, and it does slightly improve contrast of grey images, while basically not affecting quality of the image at all.

The disadvantages of this algorithm is that it is only applicable to images that are mostly grey. The results of applying this image algorithm to the lady’s face was like histhyper however, which makes me feel a little better about my algorithm being terrible.

|  |  |  |
| --- | --- | --- |
| Drew\_CE Does have useful results on images that are very grey. For images that are almost all darker than 127.5, or lighter than 127.5, they won’t have their contrast enhanced, in fact, these images will just get worse.    I tried variations where it would take the mean into account, but I could not get it to work better than the simple adjust away from the center method. | [Drew\_CE link](http://i.imgur.com/pWpTYqG.jpg) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\contrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) |

# Experiments Performed

The experiments I performed were very straight forward. I simply ran each algorithm and compared the resulting image to each of the other resulting images. This is about the extent of the experiments I carried out. I will now go through each image to show off the results side by side.

*All images were taken without permission, please don’t sue me, the university already took my money.*

## Low Contrast 1

Source: <https://www.umaryland.edu/onecard/how-to-get-your-one-card/student-id-online-photo-submission/>

|  |  |  |
| --- | --- | --- |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1 - drew_CE.JPG  [Drew\_CE link](http://i.imgur.com/pWpTYqG.jpg) | C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1 - morph_toggleCE.JPG  [Morph\_toggleCE](https://onedrive.live.com/?authkey=%21AOL7YkSjYfhhGQY&cid=A53A58CB0A4C93CF&id=A53A58CB0A4C93CF%2198967&parId=A53A58CB0A4C93CF%2198956&o=OneUp) | For this image HistHyper did awful, Morph\_CE gave her creepy eyes, and Drew\_CE made the contrast worse. So of all the contrast enhancement algorithms, Morph\_ToggleCE wins, yet the best image is the orignal in my opinion, as Morph\_ToggleCE makes the image look like it was colored by pencil crayons.  This image was artifically generated to show students an example of an id photo with poor contrast. |
| C:\Users\andre\AppData\Local\Microsoft\Windows\INetCacheContent.Word\contrast1.jpg  [Original Image Link](https://1drv.ms/i/s!As-TTArLWDqlhoUNuid1jzYaSWC2bg) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast1 - histhyper.jpg  [HistHyper Link](http://i.imgur.com/rrs7vhT.jpg) |
| lowcontrast1 - morph_CE  [Morph\_CE](https://1drv.ms/i/s!As-TTArLWDqlhoUzgeg-D9Fgko1R_g) |  |

*Kind of looks like Furiosa from the new Mad Max.*

## Low Contrast 2

|  |  |
| --- | --- |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast2 - drew_CE.JPG  [Drew\_CE](https://1drv.ms/i/s!As-TTArLWDqlhoUCCj_fZHLSTTyaBw) | [Morph\_toggleCE](https://1drv.ms/i/s!As-TTArLWDqlhp8JXbDeNoJLQUrUFQ) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast2.jpg  [Original Image](https://1drv.ms/i/s!As-TTArLWDqlhoBu-Ft06hqwjCcZ8w) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast2 - histhyper.jpg  [HistHyper](https://1drv.ms/i/s!As-TTArLWDqlhp59EW1In-iijkveqw) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCacheContent.Word\lowcontrast2 - morph_CE.JPG  [Morph\_CE](https://1drv.ms/i/s!As-TTArLWDqlhp8Kz6lLVk-734wV9Q) |  |
| For this image, Morph\_ToggleCE seems to have done the least, making the image appear blurry, without noticeable improvement in the contrast. HistHyper improved the contrast by far the most, but also reduced the images quality by introducing a lot of harsh edges. Morph\_CE appears to have done very little other than change the shade of the image from a blusih grey, to a greenish grey along with making the image appear less smooth.  In my opinion Drew\_CE is the winner here, improving the contrast by a little without reducing image quality. However it is easy to see the weakness of Drew\_CE along the bottom of the image, the simple move away from center 127.5 strategy of Drew\_CE does not handle edges very well, since it only focuses on the shade of a single pixel, the fog along the bottom of the image remains. | |

## Low Contrast 3

|  |  |
| --- | --- |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast3 - drew_CE.JPG  [Drew\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtZLbEU6nT6ilt_IGg) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast3 - morph_toggleCE.JPG  [Morph\_toggleCE](https://1drv.ms/i/s!As-TTArLWDqlhtZJJb6r98uw2nJB6w) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\contrast3.jpg  [Original Image](http://3.bp.blogspot.com/-SC-w7eTgpM0/URE9NsI_nuI/AAAAAAAAAGE/YmlWnimNuPM/s320/7957178556_001939ffc5_z.jpg) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast3 - histhyper.jpg  [HistHyper](https://1drv.ms/i/s!As-TTArLWDqlhtZKXdwQo_mbGdDbtg) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast3 - morph_CE.JPG  [Morph\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtQsty73yPUjv8TUHQ) |  |
| For this image both Morph\_CE and Morph\_ToggleCE buchered this image.  HistHyper almost gave an amazing resulting image, but struggled and had issues with the right along the right side of the image. Drew\_CE gave a modest improvement in contrast, while reducing image quality by quite a bit.  Because of HistHypers failure to deal the white properly, I don’t think any of the contrast improvement algorithms can be said to be a winner. Otherwise HistHyper seems like it did a really good job. | |

## Low Contrast 4

|  |  |
| --- | --- |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast4 - drew_CE.JPG  [Drew\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtZPO8lApZXRkzcrqA) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast4 - morph_toggleCE.JPG  [Morph\_ToggleCE](https://1drv.ms/i/s!As-TTArLWDqlhtZMwtg8f0vLgYGDQQ) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast4.jpg  [Original Image](https://s-media-cache-ak0.pinimg.com/564x/1c/76/c5/1c76c5df7979be39e25abf1ab1f97a8f.jpg) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast4 - drew_CE.JPG  [HistHyper](https://1drv.ms/i/s!As-TTArLWDqlhtZRbomjqt7w7DyfoQ) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast4 - morph_CE.JPG  [Morph\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtZQ1E_kBPJ4FbQLPQ) |  |
| Once again Morph\_ToggleCE and Morph\_CE appear to make a mess of this image. HistHyper and Drew\_CE gave a very similar result. HistHyper and Drew\_CE are so close ranking one over the other is probably entirely subjective.  That said, persoanlly I like Drew\_CE better for this image, as Drew\_CE results in an image with very dark blacks, and very white whites, and kind of leaves everything in between mostly in place. Also the grass looks more dead in HistHyper, where as in Drew\_CE the grass appears to be lush. | |

## Low Contrast 5

|  |  |
| --- | --- |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast5 - drew_CE.JPG  [Drew\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtZX9Z2LnCnCcLZiSA) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast5 - morph_toggleCE.JPG  [Morph\_ToggleCE](https://1drv.ms/i/s!As-TTArLWDqlhtZNPRoIcYMCePUZ3w) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast5.jpg  [Original Image](http://presetpond.com/uploads/2962/IMG_6683%20edit.jpg) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast5 - histhyper.jpg  [HistHyper](https://1drv.ms/i/s!As-TTArLWDqlhtZOkDNGHZI3tB3VZg) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast5 - morph_CE.JPG  [Morph\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtZQ1E_kBPJ4FbQLPQ) |  |
| YET AGAIN Morph\_CE and Morph\_ToggleCE seem to have made a mess of this image. HistHyper seems to have done the best job for this image, with Drew\_CE making the contrast worse, resulting in a moodier image, and some bricks being made way too white and looking strange. | |

## Low Contrast 6

|  |  |
| --- | --- |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast6 - drew_CE.JPG  [Drew\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtZWiWMXmQJFUBwLfA) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast6 - morph_toggleCE.JPG  [Morph\_ToggleCE](https://1drv.ms/i/s!As-TTArLWDqlhtZT21O_GxYJUzzm-w) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast6.jpg  [Original Image](https://static1.squarespace.com/static/54ca877ce4b014ea90e14bda/t/54f99a94e4b05e6b5c3d3050/1425644181215/) | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast6 - histhyper.jpg  [HistHyper](https://1drv.ms/i/s!As-TTArLWDqlhtZUKoHu3X82nR64ZQ) |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lowcontrast6 - morph_CE.JPG  [Morph\_CE](https://1drv.ms/i/s!As-TTArLWDqlhtZVSt4fPfub5qKTdA) |  |
| All four algorithms seem to have made the image worse in my opinion. The most “okay” being the Morph\_CE algorithm, with all the other algorithms making a mess. HistHyper seems to be the only algorithm that improved contrast, unfortunately it didn’t improve contrast in a way that improved the image quality. | |

# Quantitative Analysis of Algorithms

To start, a ground image was taken, and purposely has its contrast decreased. Each algorithm was run 3 times in a row on the reduced contrast version, as a single run didn’t produce very noticeable results.

|  |  |
| --- | --- |
| *C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\compare - morph_CE.JPG*  *Morph CE,* SSIM: 0.2582 | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\compare - morph_toggleCE.JPG  *Morph Toggle CE,* SSIM: 0.05766 |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\compare.jpg  *The original image,* SSIM: 1 | C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\compare_grey.jpg  *Contrast reduced,* SSIM: 0.8176 |
| C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\compare - drew_CE.JPG  *Drew\_CE,* SSIM: 0.7893 | *C:\Users\drew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\compare - histhyper.jpg*  *HistHyper,* SSIM: 0.4747 |

All the algorithms made the reduced contrast image further from the original image in terms of quality, according SSIM, which I find surprising.

Interestingly Drew\_CE is the only algorithm that didn’t butcher the image, while the amount of colour the image Drew\_CE produced is muted, when compared to both the original and the contrast reduced versions, the contrast is improved from the contrast reduced version.

I guess I hadn’t used images with high amounts of colour in my previous testing. It appears HistHyper is experiencing conversion errors from HSV back to RGB, I’m not sure why this is, as the previous 7 images I’ve run the code on did not experience this issue, and neither do the other algorithms for this same image here, which use the same RGB to HSV and back conversion code. I would have to say HistHyper seems to increased contrast, the purple grass and trees stand out from the sky, in a terrible looking way.

The results Morph and Morph toggle doesn’t look bad, but they don’t look like the original image. They look more like pencil crayon and crayon filters applied to the image, with only slightly improved contrast. Without running these algorithms multiple times, they don’t produce a noticeable difference in contrast, but running these algorithms multiple times, they become cartoony.

|  |  |
| --- | --- |
|  | Of all the algorithms, drew CE seems to have done the least amount of damage to the image… I’m not sure if that’s a good thing or not.  According to the SSIM algorithm, morph toggle CE is nothing like the original image, and all the algorithms are futher away form the original than the purposely ruined. |

In conclusion, contrast was improved with the use of these algorithms, however none of the images became closer to the original image, meaning the SSIM actually got further away from the original image than the contrast reduced was. This result seems reasonable given the fact covering an image with a uniform transparent grey filter, is much simpler than going backwards by manipulating a very non-uniform image. Also, while shoe horning colour images into grey scales image algorithms by converting to HSV does have working examples, there is clearly an issue with HistHyper when dealing with colour images with… a lot of colour.

# Notable Computational Differences Between Algorithms

When running the algorithms with the small images I used in the examples above, my program would run very fast, within a few seconds, even on my low power laptop processor.

However, when running the algorithms on big images, it took much longer. Instead of sitting around waiting for the programs to finish, I decided to queue up each algorithm to run 3 times using a simple bash script. Each run was on the same image. Then when the program finished running python would throw the time taken into a text file label: <algorithmName> - <timeComplete>.txt. This way I could leave the algorithms running over night, with the result of each run safely saved to a separate text file.

|  |
| --- |
|  |
| *A graph showing the average of five runs for each of the algorithms. Each run was on the same included image called: bigImage.png.* |
|  |
| *The raw data.* |

|  |  |
| --- | --- |
|  | To gather the raw data, I simply ran each of the 4 algorithms five times in a simple bash script. Each run used the same image: bigImage.jpg  The python program saves how long its execution time in seconds to a text file labeled the time it finished, and the algorithm name that was run.  When the 20 bash scripts finished, I had 20 text files, each showing how long an algorithm took to run, these values were copied directly into the spreadsheet. |

# Difficulties Processing Images within the Context of the Topic Chosen

* I don’t understand the math of histhyper
* The RGB to HUV conversion code I ported to python is in the file: rgb2yuv.js, which I took from [here](https://www.mikekohn.net/file_formats/yuv_rgb_converter.php). When first used, all the images would come out with a lot of artifacts, or be entirely coloured a purple / pink hue as seen in the Quantitative Analysis of Algorithms section. Small changes to the code had to be made to get HUV working to use colour images in place of grey scale. The main change made was simple: cast the resulting pixel calculations to a data type big enough to hold the larger possible values. Even though the resulting pixel values is still between 0 and 255, using calculations on the H component would often compute to values higher than a byte could hold, resulting in run time errors.

|  |  |
| --- | --- |
|  | I added the 3: “.astype(numpy.int64)”, and bingo, I could pass the H component of HUV instead of a grey scale value.  The pixel value is still locked between 0 and 255, but for whatever reason, passing an H component value could overflow a byte during these pixel calculations. |

* I had no idea what I was doing when trying to create my own algorithm. Trying to create my own algorithm for something I knew little about, gave a mix of mystery and adventure.
  + Unfortunately, I don’t think my algorithm is innovative, or original, I’m sure there’s a better algorithm in existence that is very similar. It does work well in a narrow range of applicable cases. Currently my algorithm has a “pivot” point of 127.5, causing it to only work with grey images. Though I tried to make the “pivot” dynamic based on the image, (by calculating the mean color intensity of each pixel) this caused problems that using flat 127.5 didn’t have, without any significant benefit over the much simpler 127.5 method.
  + If I had more time, I would try to solve the following problems with my Drew\_CE
    - Find a better way to calculate the “pivot” point, so that this algorithm works with a much wider band of images than the hard coded 127.5 method.
    - Figure out how to split the image into subsections, and calculate a “pivot” point independently for each image subsection. Then smooth the subsection pivot points together in a manner that would probably be like antialiasing. This smoothing would be in attempt to downplay the effects of contrast enhancement algorithms seem to have when they use image subsections (like histhyper, with its visible blocky artifacts. I would try to make the blocks aware of the neighbouring blocks average contrast to smooth from one block to the next more smoothly)
* Given the nature of marking for this course, I knew I could either half ass it, and hand it in on time. Or I could most-of-my-ass it, and hand it in a few weeks late for a better overall result. Unfortunately, things got out of hand, weeks became months, and it’s difficult to grow any additional ass.

# Running my program

## Running directly

Program parameters:

python a1p1.py <relative path to input image from a1p1.py script> <relative path to output image from a1p1.py script> <algorithm to run>

Example:

python a1p1.py images/lowcontrast1.jpg images/lowcontrast1\_drewCE.jpg drew\_CE

The available arguments for the third argument are: morph\_toggleCE, histhyper, morph\_CE, drew\_CE.

## Running using run.bash

To make managing entering paths, image names, and algorithm names simpler, I made a bash file that can be run in place of typing out everything by hand: ./run.bash

|  |  |
| --- | --- |
|  | To change what image is being operated on, change line 5.  If you want to change what folder to open from, relative to the a1p1.py script, change line 6.  To change what algorithms are being run, look at like 33, 34, 35, and 36. Currently all 4 algorithms will be run on the current image. |