Exploring Blurry License Plate Image Restoration

Connor Barker

Abstract:

In this report, I detail the various steps that I took when analyzing the different methods of enhancing blurry images with the goal of recovering legible text. It covers morphological operations, histogram equalization, 2D Image filtering and all of their roles in restoring a blurry picture. The technical section is broken into the 3 main topics addressed above and shows figures that were produced from my code. To accomplish the coding part of my project I used a computer vision package in Python called OpenCV.

My work is available on my GitHub repository at this link:

https://github.com/cbarker16/ece418finalproject

Introduction:

The problem that I chose to look into for my project had to do with image restoration and specific ways of dealing with noise in images. Specifically, I decided to look into the different factors of noise and blur that can result in image quality being degraded due to unforeseen circumstances when taking a picture. Since it was a great example of this, I focused on utilizing different methods to try and restore blurry/noisy images of license plates with the goal of improving the quality of the image. With my current experience, I would not be able to perfectly restore the images, however, I utilized many different tools used in image processing to compare them with each other as well as how they work in different situations. The three main points I researched had to do with image morphology, histogram equalization, and image filtering. To begin, I analyzed the morphological operations of dilation and erosion to see which situations they

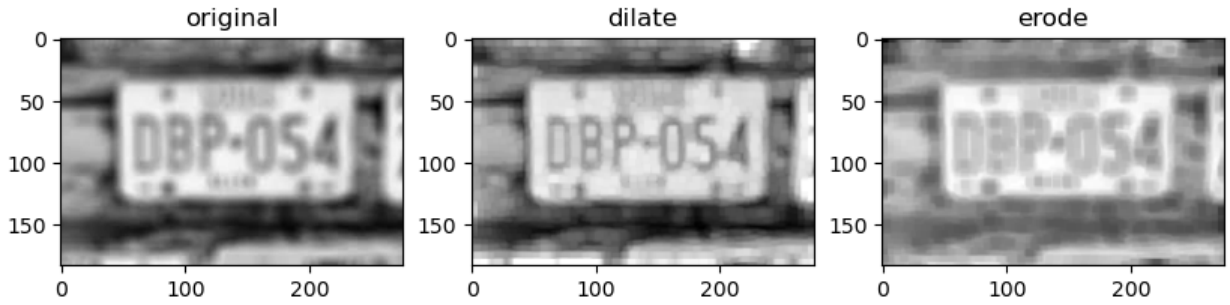
could be used in to eliminate certain types of noise inside of the letters on the license plate. Additionally, I looked into how histogram equalization could improve image quality as some of the pictures I worked with had very poor contrast between the background color of the plate and the letters. Consequently, histogram equalization will be helpful to improve the readability of the characters on the plate. Finally, I looked into 2D image filtering to help eliminate types of blur that may be present. The two filtering methods I used were applying a sharpening filter and using a Gaussian kernel to attempt to remove blurring artifacts.

Technical solution/approach

The 3 main points of my project had to do with understanding image morphology, histogram equalization, and 2D image filtering. By performing different tests, my goal was to understand the different situations in which these methods can be used, when they increase image quality, when they hurt image quality, as well as finding situations that utilize a combination of methods to obtain the best-restored image.

To begin, morphological operations ended up being very interesting when looking into their role in image restoration. In my testing, certain images contained a lot of noise inside of the characters on the plate which led to the readability of the plate being decreased. Additionally, I found that images that had blur often lead to letters being hard to make out due to their abnormal shape in comparison to what we are used to. To address this, I focused on using the morphological operations “dilate” and “erode”. In image processing, dilation looks at every pixel in the image and compares it with neighboring pixels according to a defined structuring element. In my project, I set any pixel that was originally touching a white/high-intensity pixel to also be white. Visually, this made the letters thin out and produce sharper edges around the letters on the plate as a lot of the blurry area, where there is a larger mix of low and high-intensity pixels, got smoothed out. On the other hand, I found that eroding the image leads to a decrease in visual quality. This is because when eroding an image, it sets any pixel that is neighboring a

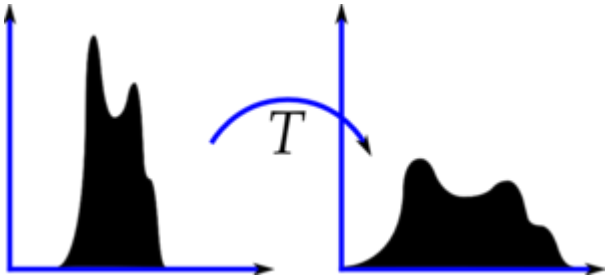
black pixel to also be black. Due to the license plate backgrounds often being white, I found that the letters ended up growing in size, often looking like blobs of low intensity that became increasingly harder to distinguish what character it was. Though there were situations where neither operation had a positive effect, I found that there were more situations where dilating increased visual quality and found that eroding almost always decreased visual quality.



Next, I analyzed how histogram equalization could improve image quality. The main goal of equalizing a histogram is to improve contrast. When looking at the histogram of an image, the distribution of intensity values often relates to the contrast. When having a larger difference in intensity, the edges in an image might appear to be sharper. Therefore, the goal of histogram equalization is to try and spread out the intensity values of the image's histogram while still maintaining the relative peaks and minimums, resulting in an image with greatly increased contrast in many cases.

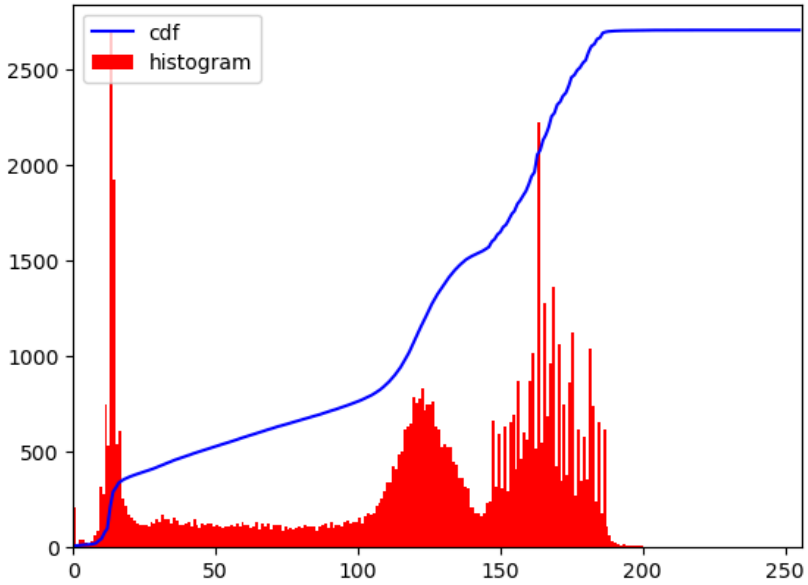
Example of histogram equalization

Image from OpenCV histogram equalization documentation: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html



The way that histogram equalization was useful in my project was that noise and blurring artifacts often decreased the contrast between the characters of the plate and the background. After applying histogram equalization to an image I found that the readability of the plate increased almost every time. Oftentimes the peaks of a histogram were not utilizing the full range of intensity values, leading to a suboptimal contrast in the image.

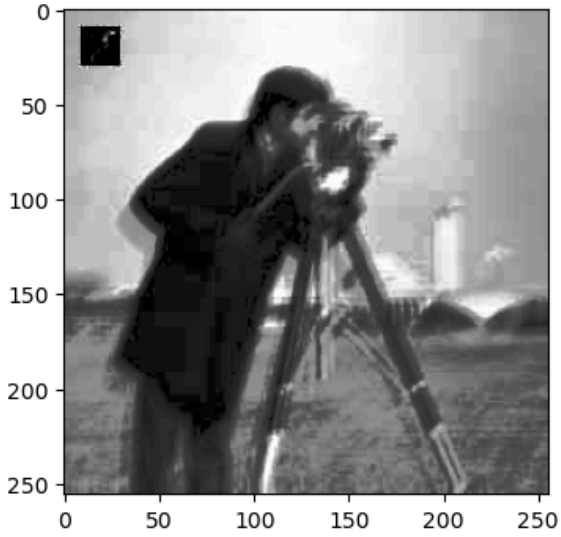
Example histogram of an image before equalization



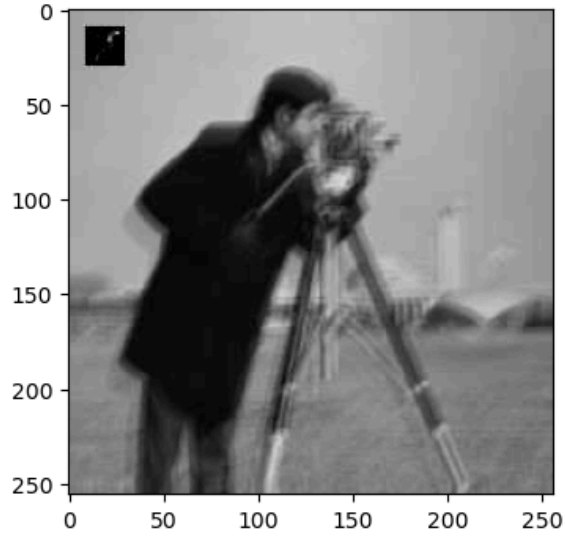
In my project, after equalizing this histogram, the CDF of the histogram was more or less linear and the intensity values were much more optimally spread out throughout the histogram. This caused a great increase in contrast which is very useful for the later parts of my project, as before trying to remove noise or deblur, it is important to prepare the image in order to yield the best results.

Example of images before and after histogram equalization:

After histogram equalization



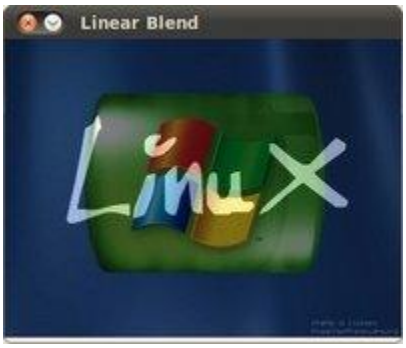
Before histogram equalization



Finally, a large part of this project was working with different methods of deblurring. The two methods I decided to compare were 2D convolution with a popular sharpening filter and using the “addWeighted()” function from the OpenCV image processing package. This function combines two images given two parameters corresponding to the desired weights of the addition. Below is an example of how this function works.

Example of the addWeighted() function blending the Linux and Windows logos

Image from OpenCV image blending documentation:: https://docs.opencv.org/3.4/d5/dc4/tutorial_adding_images.html



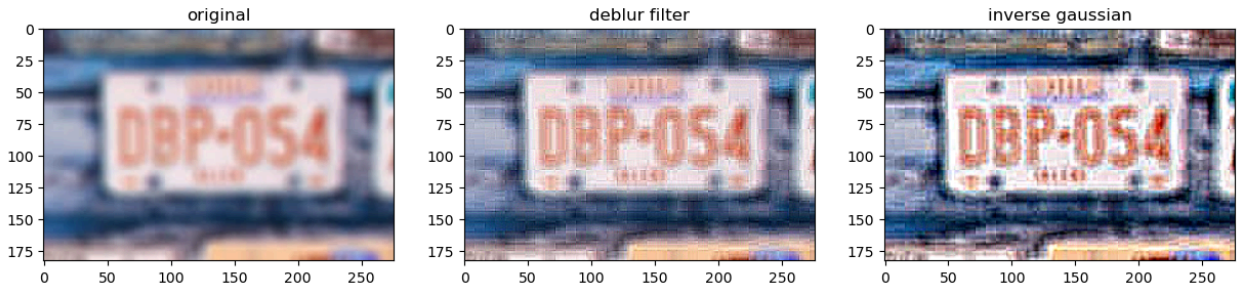
Using this function I was able to decrease noise in an image by approximating blur as gaussian noise. The way that I was able to accomplish this was constructing a gaussian kernel representing the type of noise that may be present in the image and using the addWeighted() function with a negative parameter for the noise. It is important to note that when using this function with negative values, the two parameters must

add to 1. There were many different combinations I tried as when increasing the difference between these parameters, I saw that the image would become clearer, but after a certain point it starts losing a significant amount of quality. Therefore, I felt it was important to find a balance between the amount of deblurring applied to the image while also maintaining acceptable quality. An interesting result I found was that some images do not benefit by being convolved with the sharpening filter while quality increases when using the inverse Gaussian method.

Example of the sharpening filter having a minimal effect on quality

This result is very surprising to me because both filtering methods did not do an amazing job at deblurring this image, though the inverse Gaussian seems to have improved the quality slightly. However, this result is more of an outlier as I often found that these different deblurring methods were very helpful when restoring image quality. Below is an example where both filters do an acceptable job of removing blur from the original image.

Example of both filters increasing image quality

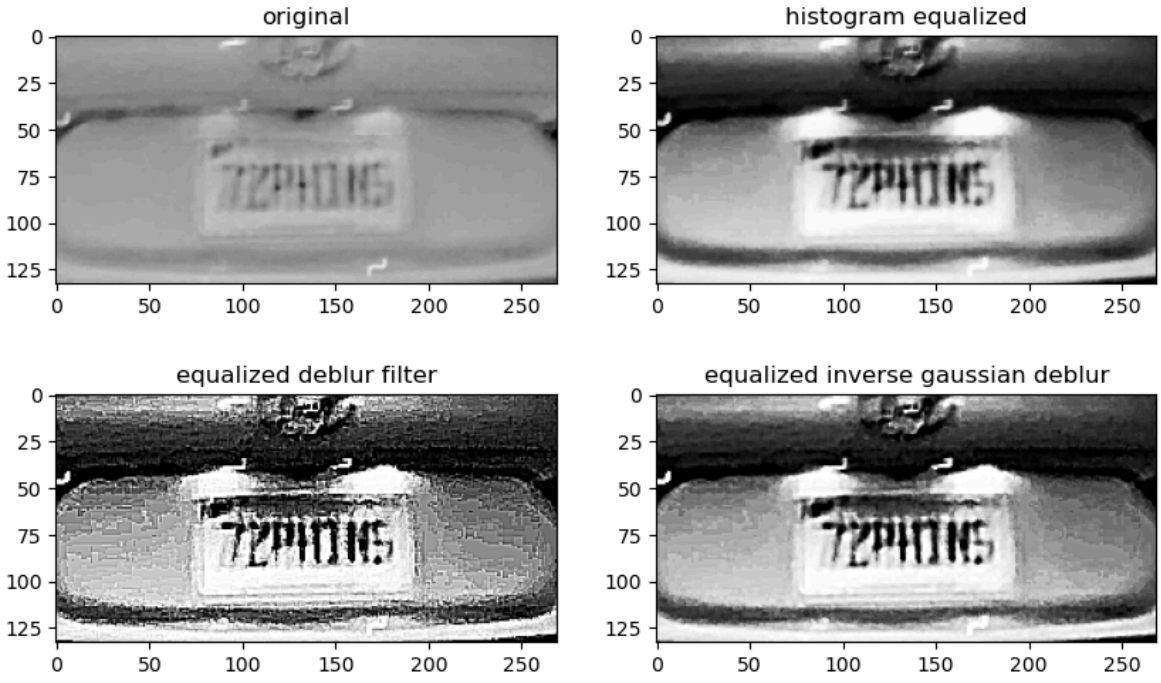


In this example, it is visible that the sharpening filter has a significantly increased effect on improving image quality compared to the previous example. One thing that is worth noting is that in both examples

so far, the inverse Gaussian deblurring method seemed to be the best option as, especially in the second example, the plate is significantly deblurred and legible.

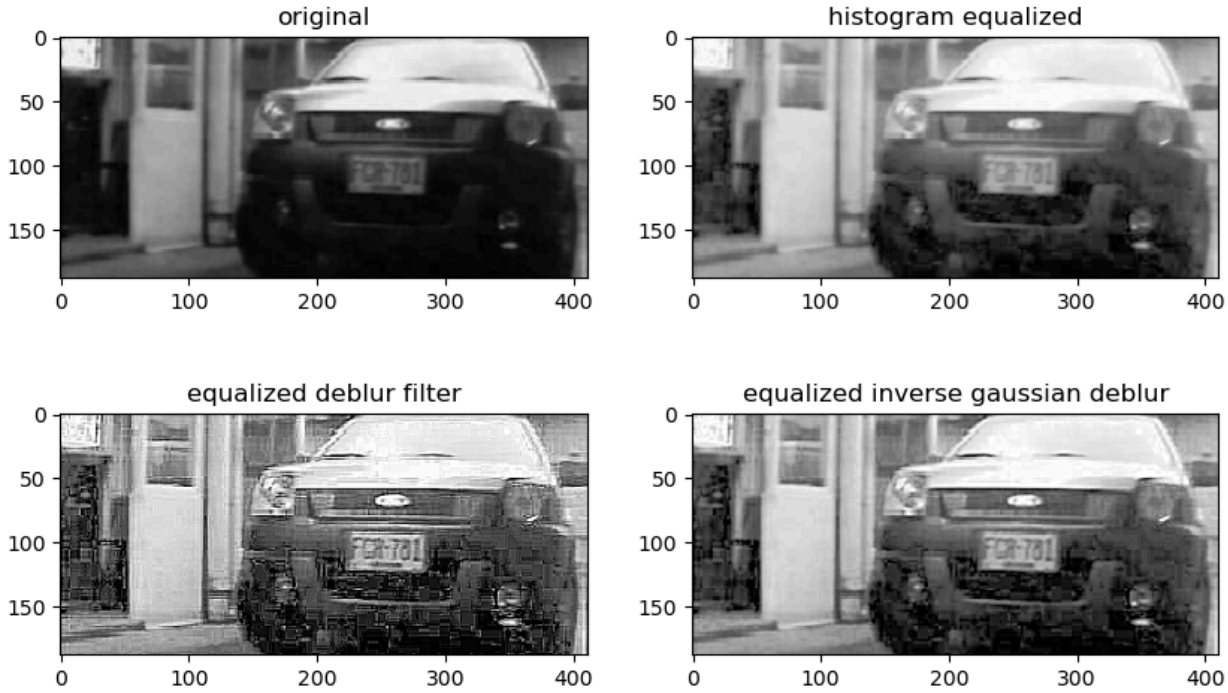
Now after analyzing all of these methods on their own, I looked into how these methods could work together. The first thing to mention is that morphological operations ended up having significantly worse results in my testing after combining with deblurring or histogram equalization. This is a weird result as it is hard to understand what causes this. On one hand, the decreased effect after histogram equalization could be caused by the change in intensity values of the new pixels leading to a different interaction of the dilation and erosion functions. However, I am not quite sure why these operations lose their usefulness after deblurring an image, though my first guess would be to deblurring artifacts left in the image and their interactions with the respective functions. Moving on to the positive results, I found that equalizing a blurry image's histogram before deblurring it results in a great increase in image quality. Below is an example of how equalizing the histogram affects image deblurring.

Effects of histogram equalization on image deblurring



In this example, the readability of the plate after equalizing the histogram and deblurring it is greatly increased. Both methods of deblurring were great in deblurring the image and making edges sharp. I think that both methods have positive aspects as the inverse Gaussian method produces a smooth image while maintaining edges while the sharpening filter leaves lots of noise in the background of the image, however, leaves the letters on the plate very sharp. In my opinion, I believe that the Gaussian method is better for deblurring as the absence of noise and deblurring artifacts is a very positive aspect of this reconstruction. Below is another example of how histogram equalization and deblurring work together so well.

Example 2 of histogram equalization before image deblurring



This example is a personal favorite part of my project because it does a great job at showing the pros and cons of choosing a deblurring method. The sharpening filter did an amazing job on this image and the license plate is very easy to read, however, it leaves a lot of deblurring artifacts in the final image. On the other hand, the Gaussian method does a great job at limiting the amount of deblurring artifacts that get left

behind, however, the edges of the text are slightly less sharp. To me, the image produced by the sharpening filter is better because the main focus of this process was increasing the legibility of the license plate so leaving artifacts isn't as important in my eyes. However, I completely understand why someone would prefer the Gaussian deblurred image as it maintains acceptable quality while limiting a lot of the extra noise that is present in the other image.

Conclusions

Overall, I found that there are a lot of useful tools in image processing and it is extremely important to understand the situations where they work best when trying to increase image quality. To recap, morphological operations work best with low amounts of blur but with high amounts of noise. These operations aren't the best at deblurring, however, they are great for removing noise inside of text and improving readability. To continue, I found that histogram equalization is a very important part of image processing, and in my findings, it was an important part of increasing quality almost every time. Finally, I found that the most powerful tools when pursuing this task were the different methods of deblurring images. For most cases, the Gaussian method of deblurring is acceptable, but it is important to understand the functions of other filters as they might be better for specific cases.

Bibliography

"LPR Cameras Setup Guide and Tools - Digital Watchdog." *Digital Watchdog®*, digitalwatchdog.happyfox.com/kb/article/39-megapix-lpr-best-practice-and-setup-recommendations/. Accessed 10 May 2024.

I used this article for some of their images as well as to learn about the importance of increasing contrast/exposure when enhancing an image

Verma, Aryan. "#17 OPENCV-Python | Image Sharpening, Noise Reduction, Blur | Gaussian, Median, Bilateral Filtering." *YouTube*, YouTube, 1 Aug. 2021, www.youtube.com/watch?v=SEj1imXHjqM.

This video taught me how to remove Gaussian blur with OpenCV. I used some of his example code on how to use the addWeighted() function and the code he used to set up the Gaussian kernel.

"Histogram Equalization." *OpenCV*, docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html. Accessed 1 May 2024.

I used code from this page of documentation on how to equalize the histogram of an image. Additionally, it was really helpful in teaching how to apply histogram equalization to different scenarios.

"Image Processing in Opencv." *OpenCV*, docs.opencv.org/4.x/d2/d96/tutorial_py_table_of_contents_imgproc.html. Accessed 29 Apr. 2024.

OpenCV is the main package that I used for my project so this documentation was the main way of learning how to use all of the functions that were available in it. All of the functions that I was using were inside of the image processing section of the package.

"Scikit-Image's Documentation." *Scikit-Image*, 20 Apr. 2024, scikit-image.org/docs/stable/.

This package was used in my project for opening images and converting them to grayscale to be more easily manipulated with OpenCV image processing functions. Additionally, I worked with the wiener filter that is inside this package.