St. John's University

CSCI 373

Senior Research Seminar

Motion De-Blurring: State of the Field

Author:
Anders Langley

Supervisor:
Dr. Mike Heroux

May 16, 2014

Contents

LIST OF FIGURES

Abstract—**Motion deblurring is one of the most difficult photograph restoration problems that has been approached. There are many obstacles associated with motion deblurring most of these revolving around estimation of the camera's motion. It is difficult to ascertain accurate information about a camera's movement by analyzing an image and most deblurring algorithms make unrealistic assumptions about the blur prior to the blur kernel estimation. Consequently, most current image deblurring algorithms produce poor and inconsistent results.**

I. Introduction

Motion blur is a common feature of modern photography. Pictures taken out the window of a moving car are likely to be blurry. If the exposure time is long enough, handheld cameras can experience significant motion blur as well. Any camera that is not placed on a tripod is susceptible to unexpected movement. Motion blur has become a more common problem as consumer cameras have gotten smaller. The closer the operator's hands get to the position of the sensor, the more each slight movement will be magnified in the resulting image.

The goal of motion deblurring is to remove the motion blur from a photograph that was taken while the camera moving. This is one of the most difficult photograph restoration problems that has been approached. There are many tools available, both free and commercial, that attempt to 'undo' motion blur but most are largely ineffective when dealing with real life motion blurs because there are a number of common blur types that standard deblurring algorithms are unprepared to deal with. Several of these obstacles, and methods that are being developed to work through them, will be described in Section III-C.

II. Survey

Since the invention of the camera, photographers have understood the problem of camera shake. In order to achieve clean, sharp photos, the camera must remain steady while the aperture is open. However, in certain situations, it can be almost impossible to keep the camera perfectly still. Over the years, many different methods have been introduced to address this issue, but each has had its flaws.

A. Definition

If a significant amount of movement occurs while the camera's aperture is open, the quality of the resulting image will be compromised. Some cameras are more susceptible to camera shake than others. Small, lightweight cameras are particularly sensitive to camera shake because the closer the operator's hand gets to the camera's sensor, the more each tiny hand movement will be magnified. [4] This is a pervasive problem because small, lightweight cameras are the preferred style of a significant portion of today's camera users. There are several factors that influence the likelihood of bad camera shake. Low light image capture is the cause of many blurry images. When photographing in low light, in order to avoid dark or grainy images, the exposure time must be lengthened. The longer the exposure time, however, the higher the chance of a twitch that could distort the final image. Optical zoom can also play a part in blurring an image as small movements will be magnified.

B. Background

When cameras were first put to use they were heavy and clumsy. Camera shake was not a pervasive problem at the time because they could not be operated in hand. Early cameras would always be placed on a table or tripod when fired and so any image blurring would necessarily come from the motion of the subject not motion of the camera. The tripod remains the most consistent way to ensure sharp images. In most situations, however, the photographer does not want to carry along a tripod, or does not have the time to set one up. For these reasons, most camera shots taken today are hand held and are, therefore, susceptible to camera shake.

"The most common commercial approach for reducing image blur is image stabilization" [8] which attempts to counteract camera shake "by offsetting lens elements [Lens Stabilization] or translating the sensor [Sensor Stabilization]". Fig 1, [8] For the last 30 years, Image Stabilization has been the primary method used to reduce image blurring as a result of excessive camera shake. Canon was one of the first companies to introduce the image stabilizer. In 1995 they developed a stabilized, EF75-300mm lens for SLR cameras.[1] The new technology was well received and Canon went on to release more lenses that utilized image stabilization. This technology proved effective for standard shooting, but these techniques were not powerful enough for macro photography. When the subject is very close, or highly magnified, the effects of camera shake are much more noticeable and the original image stabilizer lenses simply were not good enough to sufficiently counteract blurring in these situations. [1] In 2009 Canon released the EF100mm macro lens for Digital SLR cameras, which included what was called "Hybrid" Image Stabilization, an expanded image stabilization technology that stabilized not only the image being seen by the sensor, but also the image displayed on the camera's LCD screen. "This is especially relevant to handheld shooting at 1x, since the inability to properly compose and focus due to a shaky image in the viewfinder makes it extremely difficult to record sharp images." [1]

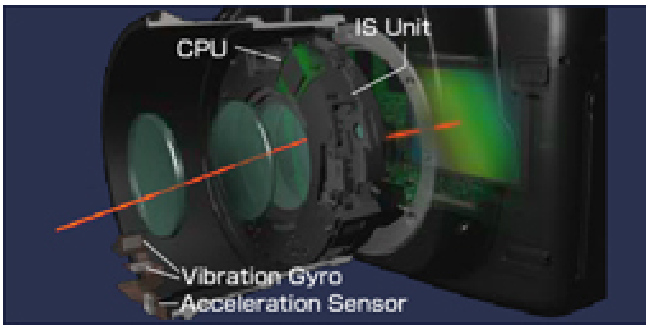


Fig. 1: Canon's Lens Stabilization technology.

Image Stabilization can be a very effective method for counteracting the effects of camera movement during exposure "however, it does not counteract the actual camera motion during an exposure nor does it actively remove blur—it only reduces blur." [8] For these reasons, it was necessary to introduce a new method for blur removal, this time via post-production.

For situations wherein Image Stabilization was ineffective, there are methods that attempt to 'undo' the blurring. There are many different methods each of which produces vastly differing results.[9] In specific situations, certain algorithms can be useful, however,there are problems associated with the theory behind this method.

III. Technical Analysis

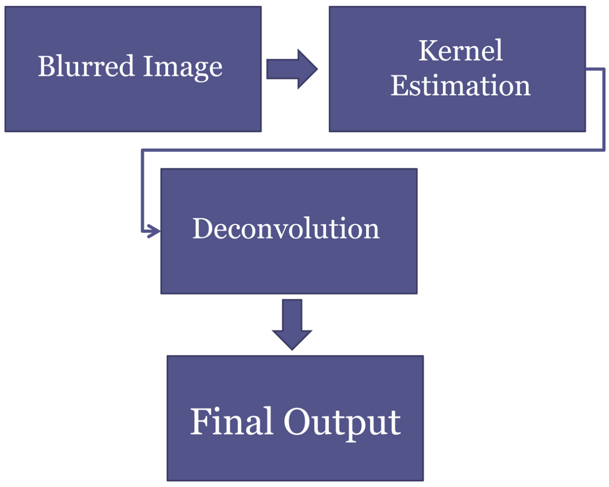


Fig. 2: The Motion deblurring pipeline.

A. *The Deblurring Pipeline*

The general theory behind motion deblurring is fairly simple and can be broken down into two steps. The process is displayed in Fig. 2. We start with a blurry image that we want to appear sharp and clear. The first thing we need to know in order to fix the image's blurriness is how the camera was moving at the time the picture was taken. Discovering this is called "blur kernel estimation". There are several approaches that can be used to accomplish this. These will be addressed further in

Section III-C. Once we have acquired the necessary information about the camera's movement, we move on to the second step in the deblurring process: "Deconvolution". In a blurry image, the individual pixels have been spread around in a "pattern". The blur kernel tells us what this pattern is, and using the kernel as a reference to determine the direction and intensity of the spread, the process of deconvolution can undo the blur by mixing color values taken from the affected areas and placing them back in their proper location. The deconvolution process relies heavily on an accurate kernel.

B. *The Blurred Image*

A blurry photograph can be defined by Equation 1 [3],[7],[4]. In this equation, 'B' denotes the blurred image which is the camera's direct output. 'I' is the latent image, or the intended result of the deblurring process. The symbol '$\otimes$' is used to denote pixel convolution. 'k' denotes the image's specific blur kernel, a piece of information defining the camera's motion during exposure. The blur kernel will be further defined in Section III-C. And 'n' represents image noise, which must be taken into account because noise is a common, and often unavoidable, problem in digital photography. Image noise affects the color values of individual pixels within the image and makes both kernel estimation, and deconvolution more difficult and less consistent [13],[14]. However, the issue of image noise is not a topic of this paper and will not be discussed at length.

$$B = I \otimes k + n \tag{1}$$

In theory, this Equation 1 can be solved for the latent image, but difficulties arise due to the presence of two unknowns: the latent image 'I', and the camera's movement during exposure 'k'. Solving for the 'I' is ultimately our goal, but this can only be done if we are able to insert an accurate 'k'. The first step therefore becomes determining the value of 'k'. This is done through a process called 'Blur Kernel Estimation'.

C. *Blur Kernel Estimation*

The blur kernel is a piece of information which represents the motion of the camera at the time of the exposure, and determines how each pixel is spread throughout the image. Each pixel of the latent image can be defined by a function of the pixels surrounding it. This is called the Point Spread Function (PSF). The blur kernel is the visual representation of the PSF. The kernel is generally represented by a small, square, black and white image wherein the relative brightnesses of the individual areas represent the extent to which the pixel in question was spread. Black means no spread, white means full spread.

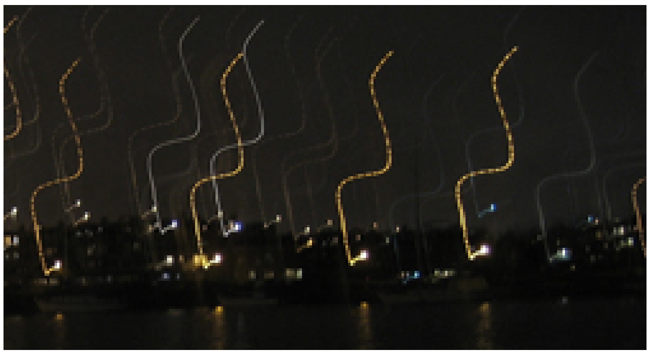


Fig. 3: An image with a discernible blur kernel. The camera's movement at the time of exposure can be easily determined due to the bright streaks from the street lights.

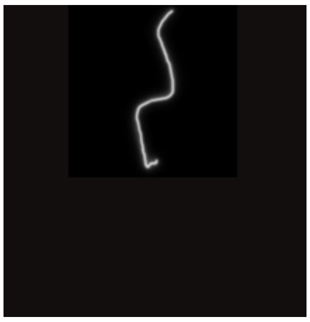


Fig. 4: The approximate kernel for the blurred image shown in Fig 3.

An example of a blur kernel is given in Fig. 4, the blurry image from which this kernel is derived is seen in Fig. 3. It is easy to understand how the kernel in Fig. 4 matches the image in Fig. 3. The light streaks leave a obvious motion path across the image and tell us exactly how the camera moved. However, it is not always this easy to determine camera motion. And, in many cases, it is nearly impossible to be certain that you have the right kernel because results can be acquired using any kernel, and there is no information with which the resulting image can be compared to ensure its accuracy.

Using Equation 1, we can apply any blur kernel and solve for ‘I’. Any kernel input that we use will result in an image, the trouble is, most outputs we acquire are nothing like the output we want [5].

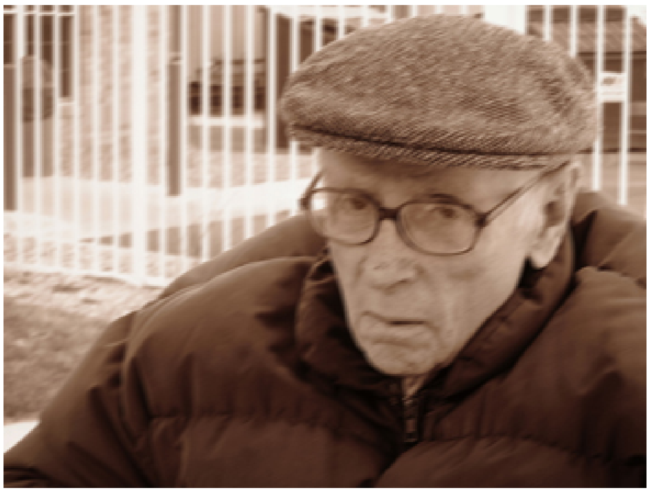


Fig. 5: An original, blurry image with an unknown blur kernel.

Fig. 6 demonstrates the problem of multiple solutions in kernel estimation. (a) is the image resulting from the deconvolution process using blur kernel (b). (c) is the image resulting from the deconvolution using kernel (d). And (e), the accurate output, is the result when kernel (f) is used. The original blurry image is shown in Fig. 5. Kernel (b) most accurately describes a gaussian blur which would, in most cases, be caused, not by a moving camera, but by a camera which was out of focus. (d) displays a blur kernel describing a vertical camera translation which was perfectly constant in speed and direction. Both of these kernels produce extremely inaccurate results and clearly do more harm than good. Kernel

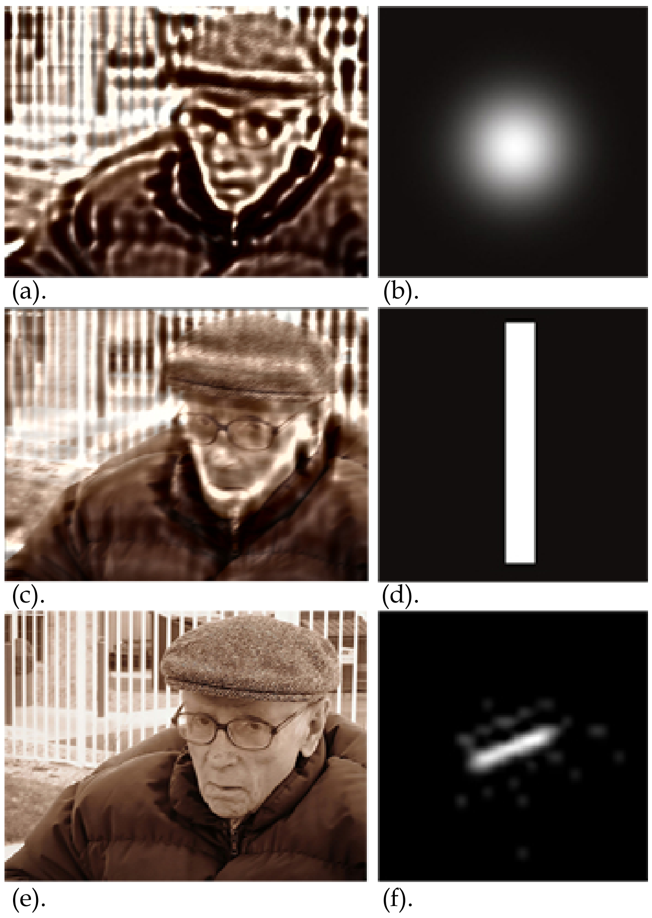


Fig. 6: A blurred image has many potential kernel solutions. All three of these images can be acquired from the image given in Fig. 5 through the same deblurring process if given a different blur kernel. Image (a) results from deblurring the original image using kernel(b). Image (c) results from deblurring the original image using kernel (d). Image (e), the desired solution, results from deblurring the original image using kernel (f).

(f) is the correct blur kernel (or close to it).

An issue with many current kernel estimation algorithms is that they make some unrealistic assumptions about the camera’s movement. First, it is likely that the algorithm will assume that the camera movement is entirely comprised of translation within the XY plane Fig. 7(a). In reality, however, the hand held camera is more likely to translate within XYZ space and rotate around all three axes as well Fig. 7(b) [9]. In general, kernel estimation algorithms are not equipped to deal with this 6-

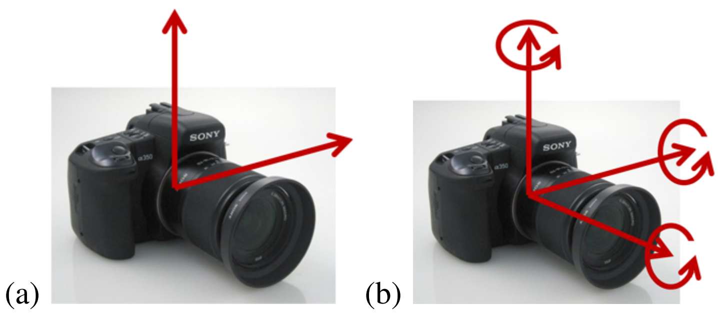
(a) (b)


Fig. 7: Most motion deblurring tools make unrealistic assumptions about how the camera was moving at the time of exposure. (a) Camera motion as anticipated by most kernel estimation algorithms [9]. (b) A much more accurate depiction of real camera motion. [9]

dimensional motion and so the kernel estimation is likely to be inaccurate. Second, kernel estimation algorithms often assume that the camera's movement was constant in speed and direction. Due to the erratic and unpredictable nature of human movement, it is extremely rare that this would occur outside of a controlled environment. These constraints drastically narrow the range of blurs that can be calculated and dealt with using current technology. One solution to these limitations has been proposed by Raskar et al. in their use of a coded, "fluttered" shutter when taking photos with long exposure times [11]. During the exposure time, the shutter will be "fluttered" open and closed at a coded rate creating a pattern within the blur which can be more easily calculated. "We compute a near-optimal binary coded sequence for modulating the exposure and analyze the invertibility of the process" [11].

Another, arguably more pressing, obstacle faced by deblurrers is the issue of Spatial Variance and/or Moving Subjects. In nearly all cases, a deblurring algorithm assumes that the PSF is equivalent throughout the image. In reality, this is very rarely true. The only situation under which this assumption would be completely accurate is if the photographer was shooting a photo directly perpendicular to a flat surface filling the camera's view. If the photograph contains any depth, or an object moving through the scene, there will be different blur patterns throughout the image. "The spatially variant blur kernel estimation is an even more difficult problem. The blur kernel, in this case, may vary in size, shape, and values among pixels. This generality makes it extremely difficult to estimate an appropriate PSF" [12]. "the convolution model [The blur equation seen in Equation 1] is often oversimplified for many practical motion-blurred images... Practical motion blurring tends to be a spatially varying blurring process" [6]. Fig. 8 demonstrates the problem of spatial variance in blurred images. We can see from the direction of the blur that the camera was being translated approximately horizontally through the scene, but the blur is not consistent throughout the image. The juice bottle in the foreground shows clear signs of horizontal blurring while the water bottle in the background simply shows signs of defocus blur. Clearly, the reason for this is that the camera was focused on the juice bottle and translated horizontally during exposure. The objects' relative distances from the camera caused parallax displaying an apparent faster movement of the juice bottle in the foreground, and relative stillness of the water bottle in the background. A similar effect can be observed when an image contains a moving object whose motion does not match the motion of the camera. The moving object will appear blurrier than the scene around it and the blur kernel of the image will not be consistent. As noted, the traditional blur model, Equation 1, is not capable of supporting inconsistent blur kernels and fails when confronted by a spatially variant image or an image containing moving objects [10],[12]. Many attempts have been made to solve this problem [10],[2],[12],[6],[8]. Anat Levin and his team at MIT have developed a camera rig which creates specifically designed motion blur which is able to produce identical blur throughout an image containing moving objects [10]. This photograph, then, can be accurately deblurred using a single blur kernel. Qi Shan and his team at the Chinese University of Hong Kong have suggested that the theory behind motion deblurring should be completely rewritten. Their model involves creating a transparency map of the blurred image and determining a specific blur value at each location [12],[7]. The advantage of this method is that it can, in theory, deal with very complex blur patterns including rotational blurs, which are difficult to define.

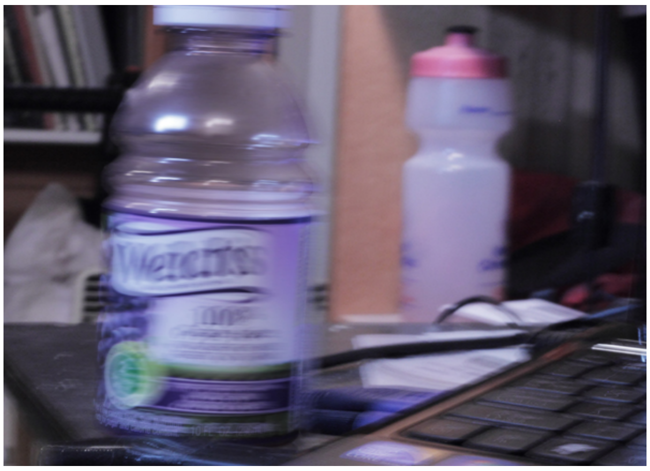


Fig. 8: A spatially variant scene with motion blur.

Another approach for blur kernel calculation is to avoid the estimation process altogether by measuring it at the time of exposure. Neel Joshi with his team at Microsoft Research, and Adams et al. through the "Frankencamera" project have developed external camera attachments built from inexpensive gyroscopes and accelerometers which are capable of tracking and recording the camera's acceleration through space and rotation around the sensor [8],[2]. The motion tracking rig designed by Joshi et. al. can be seen in Fig. 10. "The gyroscope object... tags frames with the IMU [Inertial Measurement Units] measurements recorded during the image exposure." [2]. This IMU information can be used to determine whether or not the camera was sufficiently steady to produce a sharp image and pass the information to the photographer [2], or it can be used to calculate an accurate blur kernel for the purpose of deblurring [8]."We derive a model that handles spatially-varying blur due to full 6-DOF camera motion and spatially-varying scene depth..." [8]. Utilizing either of these methods, the complex task of blur kernel estimation is not necessary. Though both teams hope this idea will eventually be adopted by camera manufacturers, the techniques are still under development and, currently, have not been applied to consumer cameras [2].

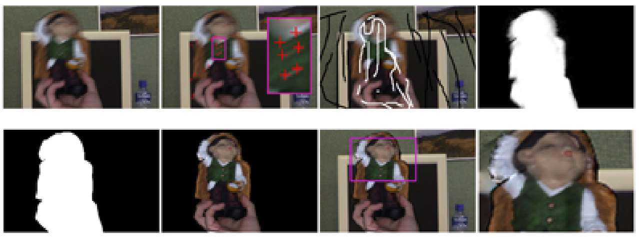


Fig. 9: Image deblurring using Transparency Mapping. [12]

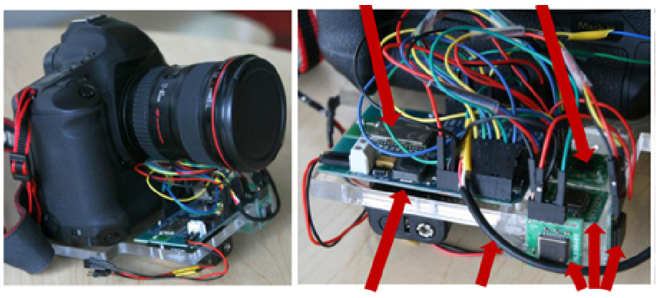


Fig. 10: 6D motion tracking rig developed by Joshi et. al. [8].

D. Deconvolution

Deconvolution is the process of recalculating color values within the blurry image based on the estimated blur kernel. The deconvolution process uses the blurred image equation solved for the Latent image 'I' (Equation 2) and inserts the estimated, or recorded, blur kernel as the value of 'k'. The deconvolution algorithm, then, recalculates the color value of each pixel as a function of the pixels surrounding it. The results of deconvolving the blurred image from Fig. III-B are shown in

Fig. 11. This deconvolution was performed using the blur kernel shown in Fig. 6(f). In contrast to the difficult problem of kernel estimation, the science behind deconvolution is well founded and relatively straightforward. In terms of research and development, the area of image deconvolution is not specifically lacking. So, if you have managed to calculate an accurate blur kernel, you can achieve an accurate deblurring of the image. The comparison of different deconvolution algorithms is less about determining which algorithm produces the best results, and more about the differences in speed [4].

$$I = \frac{B+n}{\otimes k} \tag{2}$$

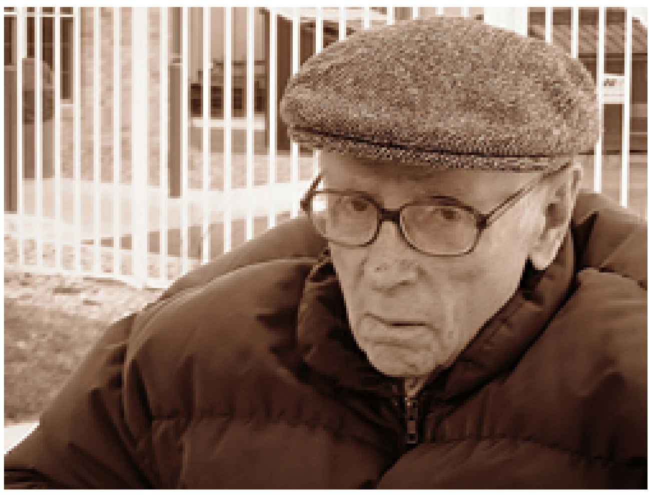


Fig. 11: The results of the deblurring process using the image displayed in Fig. 5, and the blur kernel displayed in Fig. 6(f). The kernel calculation and deconvolution was performed by Robust Motion Deblur.

IV. Future Trends

As has been noted, the deblurring process is by no means perfect. There are several problems with the majority of current deblurrers for which, so far, reliable solutions have not been located. Most of these problems appear in the area of blur kernel estimation. There are few real motion blurs whose kernels can be accurately calculated with the algorithms that are currently available to us (as will be demonstrated in Section V). Some say that, with more work, we can create improved algorithms for better kernel estimation [4], others say that kernel estimation is not reliable even in theory and that we need better ways to measure the camera's motion [8], still others would say that the entire theory behind kernel deconvolution is flawed and that the rules should be completely rewritten [12]. In any case, progress is being made. Major improvements have been made both in hardware and software development. The continued research and varied approaches to this problem make it conceivable that a solution will eventually be reached. Currently, however, it is unclear which approach will prove to be the best.

V. Motion Deblurring Project

Fig. 12: The three deblurring tools used in the deblurring tests. (a) SmartDeblur. (b) Deblur My Image. (c) Robust Motion Deblur.

A. Background

In order to better understand the deblurring process and the specifics of its uses and limitations, I downloaded and tested several pieces of motion deblurring software. I located and tried at least 5 different options, but narrowed my final selection to three which, together, I believe, offer a good variety of kernel estimation approaches. The first is called "SmartDeblur" Fig. 12(a). SmartDeblur is a very automatic deblurring process with very few manual adjustments available. Smart Deblur is a commercial product for which I have not yet paid the activation fee so, currently, every output that I receive has the words "SmartDeblur Unregistered Version" overlaid on top of the image. The second deblurring tool I chose is called "Deblur My

Image" Fig. 12(b). Deblur My Image, in contrast to SmartDeblur, relies entirely on manual kernel estimation. Once the blurry image is loaded, the blur kernel can be input either by tracing the path of the blur using a Bezier curve, or by inputting a blur kernel as an image file. After the kernel is traced or input, the program inserts the information into the deconvolution equation (Equation. 2) and calculates the result. The third and final tool that I chose is called "Robust Motion Deblur" Fig. 12(c). Kernel estimation with "Robust Motion Deblur" is neither entirely automatic, nor completely manual, so this program fits nicely between SmartDeblur and Deblur My Image. Robust Motion Deblur was created by Li Xu and Jiaya Jia, two members of the Computer Science and Engineering department at The Chinese University of Hong Kong [13]. This tool was created specifically for the purpose of expanding the field of motion deblurring so, not only is it free, it is also very well documented. The algorithms and methods behind this program are described in detail in a technical paper written by its creators. Using these three deblurring tools and a Canon T3i, I began testing the general capabilities of easily accessible motion deblurring products. Some of my results can be seen below.

B. Deblurring Tests

Fig. 13 show an example of a spatially variant image. (a) is the original blurry image. (b) is the output of Robust Motion Deblur. This is certainly an improvement over the original image, but it is not perfect. The jar at the center of the image is much clearer than it was before the deblurring, but the deconvolution process left some unsavory ringing around the edges of the objects in the scene. The output of Deblur My Image is displayed in (c). It is debatable as to whether or not this is actually an improvement over the original image. These results are similar to the output of a simple sharpening filter. Some of the details of the image have been sharpened slightly, but the ringing that has been

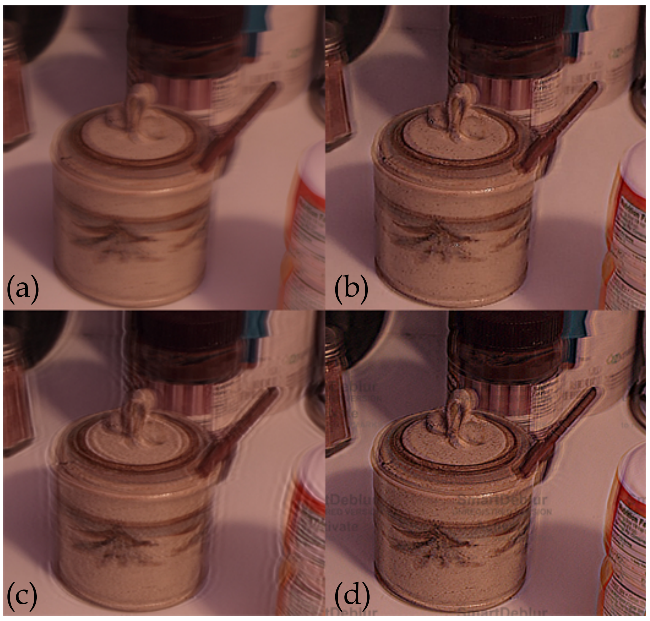


Fig. 13: Outputs of three different motion deblurring tools. (a), The Original image. (b), Robust Motion Deblur. (c), Deblur My Image. (d), Smart Deblur.

left by the deconvolution process is much more noticeable than the blur in the original image. (d) is the output of SmartDeblur. This deconvolution output, apart from the watermark and slightly more powerful edge sharpening, is quite similar to the output in (b). The blur spread has been compressed, but is still perceivable. Some artifacts have been left, and/or created, by the deblurring tool. Overall, however, the result is a definite improvement over the original with enhanced detail and a somewhat less noticeable pixel spread. The main obstacle this image poses is spatial variance. The angle at which the photo was taken, the shape and position of the objects within the scene, and the camera's translation produce a highly inconsistent motion blur which cannot be deblurred with a single blur kernel. Since I was not able to locate a deblurring tool capable of estimating, or deconvolving several kernels simultaneously, I was not able to achieve satisfactory results when dealing with spatially variant images.

Fig. 14 displays the results of motion deblurring

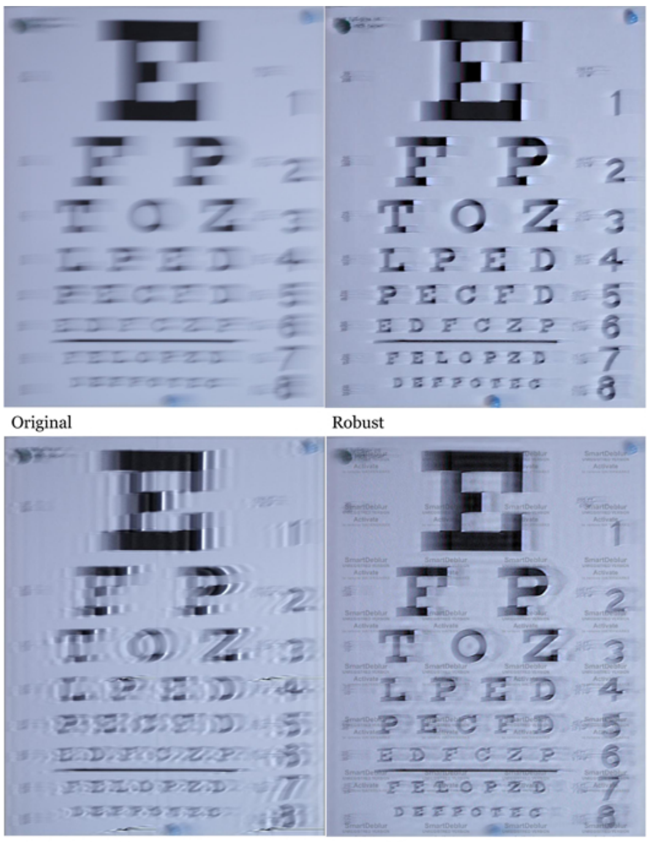


Fig. 14: A spatially invariant image with a medium sized horizontal motion blur, and three deblurring attempts. (a) is the original image. (b) is the output of Robust Motion Deblur. (c) is the output of Deblur My Image. (d) is the output of Smart Deblur.

on a spatially invariant image with a larger amount of pixel spreading. As in the previous example, Robust Motion Deblur made significant improvements to the original image while retaining some ringing and discoloring from the deblurring process. It is a definite improvement in that every letter on the eye chart is now legible, but there is still an undesirable 'fuzziness' to the text. Deblur My Image, again, seemed to achieve nothing more than some sharper lines in the image, and it is, again, debatable as to whether the final output is any improvement over the original. After repeatedly getting poor results from this tool, I came to the conclusion that manually estimated blur kernels are rarely the best option. The kernels created by both SmartDeblur and Robust Motion Deblur have consistently achieved better results than my best efforts when it came to kernel estimation. Contrary to the previous test, SmartDeblur produced an image less sharp than that of Robust Motion Deblur. SmartDeblur did produce improved image clarity but it was not able to reduce the blur as much as Robust Motion Deblur. This test did not suffer from spatial variance but it seems to be pressing the limits of the blur size that is able to be processed by the tools I was using.

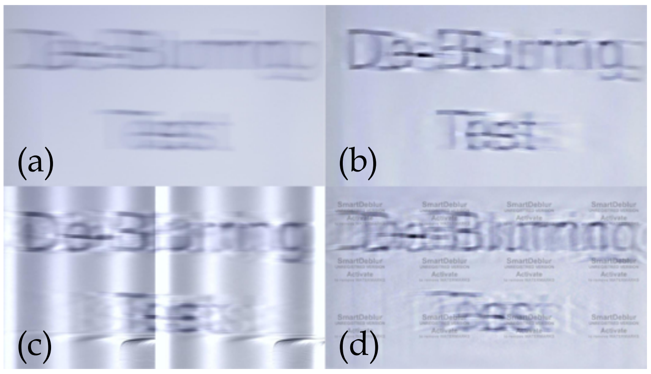


Fig. 15: A spatially invariant image with significant horizontal motion blur, and three deblurring attempts. This diagram displays the original blurry image alongside the outputs of Robust Motion Deblur, Deblur My Image, and SmartDeblur respectively.

The test documented in Fig. 15 takes the blur level even higher. This test was done to determine if any of the blur tools was able to improve upon a blur so intense that it rendered the entire image undecipherable. (a) is the original image, (b) is the output of Robust Motion Deblur, (c) is the output of Deblur My Image, and (d) is the output of SmartDeblur. The only result that appears to be even slightly more readable than the original image is (b). This was not surprising. The results that I had acquired in previous tests led me to anticipate better results from Robust Motion Deblur than from the other two. It is also noted in its documentation that Robust Motion Deblur is designed to accurately calculate large blur kernels which is exactly what is necessary to counteract significant pixel spreading. However, even Robust Motion Deblur was not able to satisfactorily deblur the image, and none of the "deblurred" images are improved to the point of making the words legible.

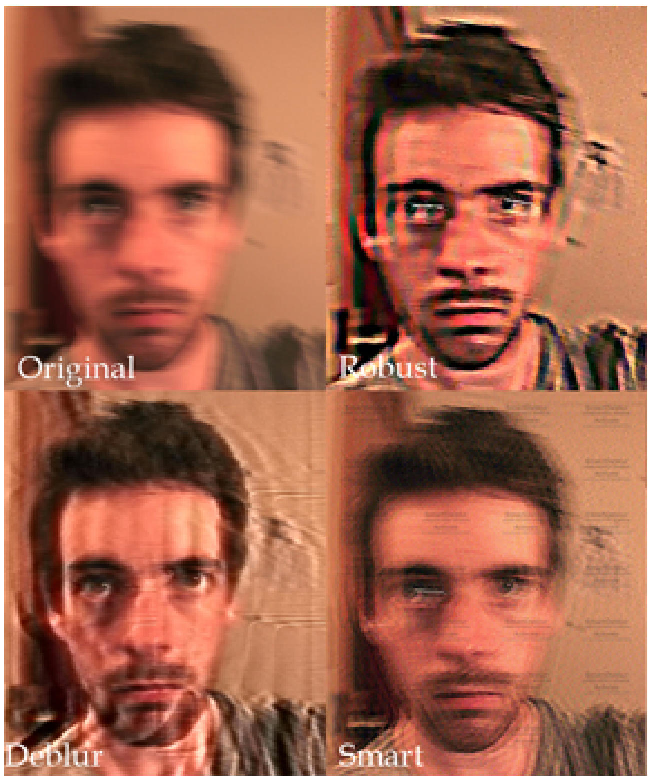


Fig. 16: A blurry image wherein the manual kernel estimation seemed to have a slight edge on the automatic deblurrers. The original image is on the top left. The result of Robust deblur is on the top right. The output of Deblur My Image is on the bottom left. The output of SmartDeblur is on the bottom right.

The image in Fig. 16 displays the results of the only test I ran wherein "Deblur My Image" (the software implementing manual kernel estimation) seemed to produce the best results. None of the programs gave specifically good outputs, but the output of "Deblur My Image" seemed to be the sharpest. The blur was relatively extreme, more so than the other two programs could determine. The blur was also very consistent throughout the image, and the blur pattern was easily discernible to a human. These seem to be the circumstances under which manual kernel estimation is superior to automatic kernel estimation.

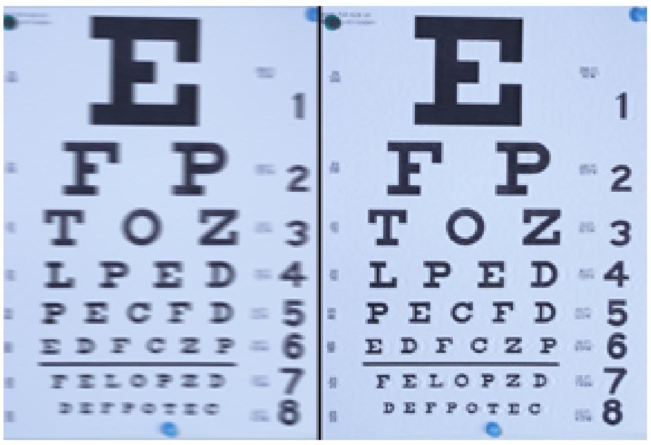


Fig. 17: A synthetically blurred image and the results of deblurring. The deblurring was done with Robust Motion Deblur

Fig. 17 displays a test done with a synthetically blurred image. The original photograph was not motion blurred. The image on the left was blurred synthetically with Photoshop's motion blur filter. The image on the right is the result of deblurring the synthetically blurred image. The deblurring process was able to restore the image's detail much more effectively than in any test I ran with real motion blur. As was noted in Section III-C, deblurring tools tend to assume that blur kernels are constant in regard to speed and direction, and are consistent throughout the image. The blur filter that was applied to the image perfectly met both of these criteria and, consequently, the deblurring process worked very well. Unfortunately, it is extremely rare to encounter a motion blur this consistent in real life.

C. Results

I was somewhat disappointed by the deblurring quality I was able to achieve with these tests. Of the three tools I was using, Robust Motion Deblur seemed to perform the best overall, but none of the three tools have, yet, been able to deblur any but the simplest of motion blurs. However, I plan to continue running tests and hope to determine more accurately what features make a motion blur "deblurrable".

VI. Conclusion

Motion deblurring is a difficult problem. The range of complex techniques that must be taken into account has, so far, rendered the problem unsolvable for most blurs. Real motion blur is far too complex to be consistently and accurately deblurred using tools that are readily available. There are several projects under development formulated to address these problems. Some follow the accepted kernel deconvolution style whereas others implement new blur calculation strategies. [2], [8], [12] Hopefully, future combinations of these ideas will create a more robust and reliable deblurring package. But there is much more work to be done before that point can be reached.

REFERENCES

[1] Canon's image stabilizer technology, 2013. http://web.canon.jp/imaging/lens/technology/ index.html.

[2] Andrew Adams, Eino-Ville Talvala, Sung Hee Park, David E. Jacobs, Boris Ajdin, Natasha Gelfand, Jennifer Dolson, Daniel Vaquero, Jongmin Baek, Marius Tico, Hendrik P. A. Lensch, Wojciech Matusik, Kari Pulli, Mark Horowitz, and Marc Levoy. The frankencamera: an experimental platform for computational photography. *ACM Trans. Graph.*, 29(4):1–12, 2010. 1778766.

[3] T. S. Cho, S. Paris, B. K. P. Horn, W. T. Freeman, and Ieee. Blur kernel estimation using the radon transform. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Conference on Computer Vision and Pattern Recognition, pages 241–248, NEW YORK, 2011. Ieee. ISI Document Delivery No.: BXB85 Times Cited: 7 Cited Reference Count: 16 Cho, Taeg Sang Paris, Sylvain Horn, Berthold K. P. Freeman, William T.

[4] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman. Removing camera shake from a single photograph. *ACM TRANSACTIONS ON GRAPHICS*, 25(3):787–794, JUL 2006.

[5] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman. Removing camera shake from a single photograph, 2006. 1141956 787-794.

[6] H. Ji and K. Wang. Robust image deblurring with an inaccurate blur kernel. *Ieee Transactions on Image Processing*, 21(4):1624–1634, 2012.

[7] Jiaya Jia. Single image motion deblurring using transparency. In *2007 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, VOLS 1-8*, IEEE Conference on Computer Vision and Pattern Recognition, pages 453–460, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2007. IEEE; hp invent; INI-GraphicsNet; VIOSO, IEEE. IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, JUN 17-22, 2007.

[8] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. Image deblurring using inertial measurement sensors. *Acm Transactions on Graphics*, 29(4):9, 2010. ISI Document Delivery No.: 624GZ Times Cited: 16 Cited Reference Count: 16 Joshi, Neel Kang, Sing Bing Zitnick, C. Lawrence Szeliski, Richard 17 Assoc computing machinery New york.

[9] Seungyong Lee and Sunghyun Cho. Recent advances in image deblurring. In *SIGGRAPH Asia 2013 Courses*, SA '13, pages 6:1–6:108, New York, NY, USA, 2013. ACM.

[10] Anat Levin, Peter Sand, Taeg Sang Cho, Fr, do Durand, and William T. Freeman. Motion-invariant photography, 2008. 1360670 1-9.

[11] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. *ACM Trans. Graph.*, 25(3):795–804, 2006. 1141957.

[12] Qi Shan, Wei Xiong, and Jiaya Jia. Rotational motion deblurring of a rigid object from a single image. In *2007 IEEE 11TH INTERNATIONAL CONFERENCE ON COMPUTER VISION, VOLS 1-6*, IEEE International Conference on Computer Vision, pages 738–745. IEEE, 2007. 11th IEEE International Conference on Computer Vision, Rio de Janeiro, BRAZIL, OCT 14-21, 2007.

[13] Li Xu and Jiaya Jia. Two-Phase Kernel Estimation for Robust Motion Deblurring. In Daniilidis, K and Maragos, P and Paragios, N, editor, *COMPUTER VISION-ECCV 2010, PT I*, volume 6311 of *Lecture Notes in Computer Science*, pages 157–170, HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY, 2010. Inst Natl Rech Informat & Automat; Google; Microsoft Res; Technicolor; Adobe; DynaVox Mayer-Johnson; Eur Res Consortium Informat Math; Gen Elect; IBM; Johnson Controls; Point Grey; Univ Houston; Siemens, SPRINGER-VERLAG BERLIN. 11th European Conference on Computer Vision, Heraklion, GREECE, SEP 05-11, 2010.

[14] Lu Yuan, Jian Sun, Long Quan, and Heung-Yeung Shum. Image deblurring with blurred/noisy image pairs, 2007. 1276379 1.

Appendix

A. CSBSJU Experiences as preparation for CSCI373 and beyond

The course work I have done for Computer Science classes at St. John's/St. Ben's has proved invaluable for this current project and in other areas as well. One of the most prominent skills that I have developed throughout my time at CSBSJU is programming which, though it has been useful in a number of situations, was not necessary for this particular project. Despite the fact that this project did not involve what is, seemingly, the most important CSCI discipline, there were many lessons, techniques, and thought processes that came in handy regarding this semester's project.

1) Technical Writing: One of the major things that came in handy this semester was my experience with technical writing. All through high school, and the first few semesters of college, my writing experience had been, mostly, limited to book reports, reflection papers, essays, etc. Consequently, I had become adept at making my train of thought sound good on paper. This, however, was entirely unlike what was required for this course. Technical writing, rather than relying on flowery language, sentence flow, and rhythm, requires a much more methodical approach, implementing nothing more than blatantly clear statements one after another, each making its point as simply and briefly as possible. Initially this was a bit difficult as it required a very different writing approach. Classes that are more based around writing, oddly, seemed to have less of a focus on the content of the writing than on how it was written and how much of it there was. I have, several times, been lauded by English professors and writing instructors, but always, specifically, for writing style. I found that, if I wrote down enough interesting sentences, I could attain very good grades on writing projects without really having much of a point to make. The difference with technical writing is that content is the main goal. In contrast to English professors, several Computer Science professors have promoted accuracy, brevity, and clarity above all else. My experience with writing in CSCI classes has been: if you don't have a good point to make, you can't fake it. CSCI profs will not hand out good grades to long meandering musings simply because it was a pleasure to read, they would always prefer to read a well thought-out piece which is as complete and condensed as possible. This was, essentially, a complete turnaround for me so it took a bit of practice, but, I believe, I have now learned to apply either style fairly well.

2) Algorithm Analysis/Theoretical Computing: While I didn't do any actual code writing or executing for this particular project, it was helpful to have a little bit of experience with algorithm analysis. As was mentioned in this SOTF paper, one of the main things that can differentiate deblurring options from on another is the execution speed. Though I could certainly have simply read about execution times, or tested specific deblurrers for relative speed, it was nice to have some understanding of why one deblurrer would be sluggish while another is blazingly fast. Though this was only possible for deblurrers whose source code or algorithm was available to the public.

It was also nice to have some foundation with theoretical computing. Without learning the basics of how a computer actually functions, the reasons why a deblurrer was designed in a specific way would have seemed mysterious. The foundations I received in CSCI310 and CSCI339 were extremely beneficial in understanding the basic nature of computers and why we interact with them the way we do.

3) Data Representation: Most of the data representation skills that proved helpful for this project revolved around interpretation of data more than creation of data, however the experience I had in courses like CSCI230 and CSCI239 creating data structures, flow charts and graphs was useful in acquiring information for this course. Somehow, I never had any experience reading or writing a flowchart before I took CSCI230, so that was one specific basic skill that I appreciate being given as far as this project was concerned.

4) Project outlining: One of the most notable features of CSCI230 (Software Development) was the idea of project preparation and outlining. Prior to that course, I had never attempted any project that required such strenuous documentation and planning. I have always been a proponent of the ”cross that bridge when you come to it” approach when it comes to planning so it was quite a new experience to literally figure out everything about a project before I even begin creating it. It was, however, a valuable experience. After that project was completed, I was able to grasp just how much time I actually saved by doing the planning ahead of time instead of trying to make everything work together during the coding stage. Since the CSCI230 course project, and other more recent projects, I have begun to understand the benefits of stepping through a project in its entirety before the real work is begun. This approach has certainly been useful regarding the course project for CSCI373. While much of the semester had already been broken down into stages for me (an extremely helpful aspect of the course), there was a high percentage of the planning and preparation that I had to do on my own (which, in the long run, was also helpful). I believe that my previous experience in project planning, derived from CSCI230 and other project-heavy courses, were beneficial in completing everything in an accurate and timely manner.

5) Problem Solving: This is one of the most basic skills that I learned throughout my time at CSBSJU, but it is also one of the most universally applicable and permeating. Since day one of my time in the St. John’s CSCI program (CSCI161 Intro to Problem Solving), problem solving has been one of most important points, if not the most important, that has been discussed. Everything Computer Science project I have completed during the past four years has related, in some way, to problem solving and the techniques associated with the particular problem at hand. In addition to aiding the speed and accuracy with which I was able to complete this project, I’ve found that the logical, methodical approach favored by Computer Scientists (and by computers) has become more prevalent in everyday life. Ever since I became comfortable with this approach, I naturally resort to very precise, methodical trains of thought and speech. I have found this to be both good and bad. On one hand, I have become very good at asking qualifying questions when something is not perfectly clear. And, due to this fact, I very rarely misunderstand instructions or statements. I also have a tendency to make absolutely certain that my statements are completely clear. Consequently, my overall effectiveness in personal communication has been greatly improved and I believe the ”think like a computer” strategy has played an important role in this. On the other hand, I sometimes find myself over analyzing things that people say to make sure that the syntax is absolutely correct. While this is definitely important when communicating with a computer, there should certainly be some leeway when it comes to human interaction because, I’m sure, such over analysis can become somewhat obnoxious and I will have to make a stronger effort to differentiate my communication with computers from my communication with other humans. On the whole, however, I believe this core discipline of Computer Science has proved to be quite a valuable resource in nearly every aspect of life. And I appreciate all the time and effort each member of the CSBSJU Computer Science faculty has put into helping me learn this.