Creating a Low-Light, Motion-Blurred Image Processing Pipeline

Alexander Evelson
Stanford University
aevelson@stanford.edu

Andrea Baldioceda
Stanford University
abaldio@stanford.edu

Abstract

When processing images, there is a consistent trade-off between blur and noise. Given that noise is high-frequency content, denoising by nature blurs an image. By contrast, deblurring introduces noise into the image. In low-light conditions, the situation is compounded by the need to brighten an image. The process of brightening the image itself introduces additional noise. Taking images while in motion, especially of wildlife, though not necessarily of course, introduces additional blur. Past projects have considered one issue or the other, but few have attempted to tackle the basket of issues in an end-to-end manner. We thus set out to create a pipeline that takes these dark, blurry images and processes them into sharp, bright images.

1. Introduction

Wildlife photography in low-light is often challenging. Photographers need to deal with moving objects, whether that is an animal moving, camera motion, or the photographer being in a moving vehicle. All these scenarios can introduce motion blur in the captured images. Besides motion blur, capturing images at low-light poses a difficulty on its own. In most cases, using a flash might not be an option as it might scare the animals away. But without using a flash, one might need to adjust the exposure time to have enough light, and maintain the camera steady. If using a mobile camera, adjusting the exposure time and other settings might not even be possible, and maintaining the device completely steady would be extremely difficult without using a tripod. Moreover, increasing the exposure time would mean having additional blur especially when capturing moving objects or not having the camera completely steady.

Our goal for this project is to create a post-processing solution for low-light blurry captured images to convert them into bright, sharp images.

2. Related Work

Although motion deblurring techniques and low-light photography have been studied at length, rarely do researchers consider them in tandem. As a result, the solutions to one of these issues may cause problems in scenarios that impose additional constraints on the other. For example, in 2011, Scott McCloskey proposed an algorithm that used temporally coded flash illumination for motion deblurring [4]. A flash-focused approach, however, violates the constraint of nocturnal wildlife photography that cautions against the use of bright lights, which might scare the subjects of the photographs - nocturnal creatures. Similarly, Zhe Hu et al. proposed a system for deblurring low-light images using light streaks, which although popular in urban environments are unlikely to be prevalent in the natural settings in which wildlife photography takes place [2].

Fortunately, there have been some breakthroughs in the space that allow photographers to capture images in low-light settings and deblur them without relying on flashes of visual light. Feng Li et al. proposed a hybrid camera for low-light imaging, a specialized solution that would address the needs of parties looking to capture photographs of moving objects in low-light conditions [3]. This, however, is still far from a generalized solution for casual tourists hoping to capture high-quality images during their nighttime excursions. The techniques that could apply in this use case were discussed in Bong-Seok Choi et al.'s paper on multispectral flash imaging in low light conditions [1]. Using this approach, we could brighten an image without introducing the great degree of motion blur that would come with long exposure and without frightening the subjects. The technique was further refined in 2019 by Jian Wang et al., who used stereoscopic dark flash to develop an image with the color and tone of a traditional RGB photograph with the low levels of noise that accompany traditional flash [5]. Applying this approach to our scenario, we hope to create an algorithm that allows users to brighten their low-light photographs and deblur them without requiring that they rely on flash in the process.

3. Theory, Method and Analysis

Instead of focusing on just one part of the image processing pipeline, we decided to take an end-to-end approach to see if we could come up with a process that properly addressed the needs of amateur photographers operating with low-light motion-blurred images. We thus broke the process up into three components and then looked at the way each part affected the others. At the end, we assessed the difference between the initial dark, blurred image and the final brightened, denoised and deblurred image.

3.1. Initial Image Selection

We obtained our images from the ExDARK dataset. We chose several images of animals taken in low-light conditions without flash. We then manually blurred them to simulate the scenario amateur photographers would find themselves in if they were moving their camera at the time of the image capture. Figure 1 shows the original images we used from the ExDark dataset.

(a) Image # 1 (b) Image # 2

(c) Image # 3 (d) Image # 4

(e) Image # 5

Figure 1: Original Low-Light Images from ExDark

To artificially blur the images we used MATLAB's motion blur function. We chose to use a motion blur because it simulates well the blur that will result from a camera shaking or moving while capturing an image. We kept the magnitude to within 30 pixels to allow for drastic but realistic blurring that was still recoverable. The angle of motion blur had a 360 degree range. Figure 2 shows the resulting images after applying the artificial motion blur.

(a) Image # 1 - Blurred (b) Image # 2 - Blurred

(c) Image # 3 - Blurred (d) Image # 4 - Blurred

(e) Image # 5 - Blurred

Figure 2: Artificially Motion Blurred Low-Light Images

3.1.1 Image Brightening

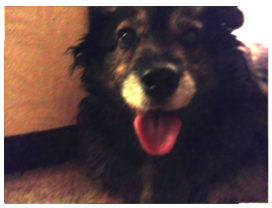
Because we wanted to focus on the deblurring and denoising components of the pipeline, where the trade off between noise and blur was most evident, we limited our approach here to a simple gain. We clipped our pixel outputs here to ensure that they were all within a reasonable color range. Because we stored the images as doubles in our implementation that range went from 0 to 1, but it could have just as easily gone from 0 to 255. Figure 3 shows the resulting images after applying brightening the artificially motioned blur low-light images.

3.1.2 Image Deblurring

For the image deblurring stage, we operated under two sets of conditions: assuming the blur kernel is known, and assuming the blur kernel is unknown.

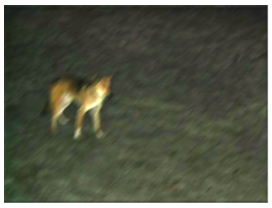
A. Known Blur Kernel

We first operated under the assumption that we knew the details of the blur kernel to test how efficiently the pipeline would work under simple conditions. We used two approaches to deblur the image in this case: Wiener deconvolution and the alternating direction method of multipliers



(a) Image # 1 - Blurred, Brightened

(b) Image # 2 - Blurred, Brightened



(c) Image # 3 - Blurred, Brightened

(d) Image # 4 - Blurred, Brightened

(e) Image # 5 - Blurred, Brightened

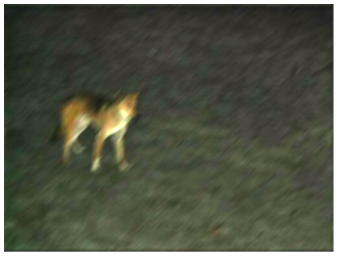
Figure 3: Artificially Motion Blurred Low-Light Images

(ADMM). Figure 4 shows the deblurred versions of image #3 (with artificial blur and brightened) after applying the Wiener deconvolution vs. applying ADMM.

The two approaches each have their own strengths and weaknesses. Images deblurred via Wiener deconvolution tend to be sharper, but sometimes overly so. As a result, artifacts form that add unwanted noise to the image. By contrast, ADMM tends to avoid the artifacts. However, it is a more time-consuming approach and does not quite sharpen the image as much as one would like. Depending on the context, a photographer might prefer one over the other, but for the sake of consistency, we proceeded with the Wiener deconvolution approach as our preferred baseline.

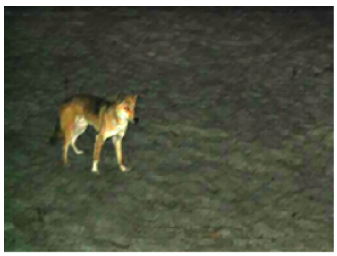
B. Unknown Blur Kernel

Using the Wiener deconvolution as a baseline, we decided to see if we could get comparable results in a situation more akin to that a photographer would face, one where the blur kernel was unknown. However, we constrained ourselves to cases where the blur kernel could be approximated to be a motion blur with a magnitude of no more than 30 pixels in each direction. Photographers taking images blurrier than that would be better suited recapturing their subjects.



(a) Blurred-Brightened Image #3

(b) Image deblurred using Wiener Deconvolution



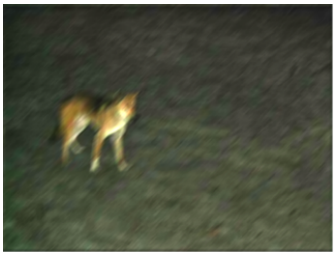
(c) Image deblurred using ADMM

Figure 4: Comparison of Deblurring algorithms on blurred image # 3

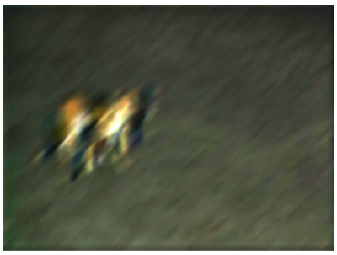
We first tested out the MATLAB deconvblind() function to see what the default approach is. We expected admirable results that would set a high standard. Instead, we witnessed results with a poor performance and set out to do better. Figure 5 shows the blurred and brightened image #3 along with the corresponding deblurred image using MATLAB's blind deconvolution function.

Given the poor results we achieved using MATLAB's blind deconvolution, we wanted to implement our own algorithm that would operate within our constraints. We needed a timely approach that could deblur a motion-blurred image with a magnitude/length of up to 30 pixels. To do so, we performed a series of Wiener deconvolutions for a fixed length (number of pixels) and varying thetas (angle) to deblur the motion blurred images. The theory was that even if the initial magnitude/length was wrong, the best theta would still have the best relative performance.

Our initial intuition was that poorly guessed thetas would



(a) Blurred-Brightened Image #3



(b) Deblurred Image using MATLAB's blind deconvolution

Figure 5: Blurred image # 3 and corresponding deblurred image using MATLAB's deconvblind()

yield images as blurry as or blurrier than the original. As a result, the image with the most high frequencies with a significant coefficient would be the best. To test that theory, we transformed our images into the frequency domain via a Fourier transform and counted the number of frequencies with coefficients that exceeded a given magnitude. For our threshold, we used 0.001 of the maximum coefficient. That, however, yielded poor results. The reason why can be seen in an example image with an improperly guessed theta and its frequency domain representation. Figure 6 shows the resulting image in the time and frequency domain when using a poor theta approximation.

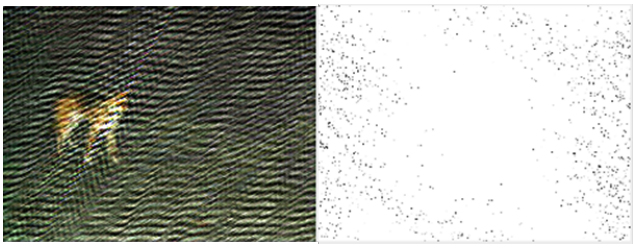


Figure 6: Image in spatial and frequency domains with poor theta approximation (estimate $\theta = 60$, len=15;
actual $\theta = 45$, len=15)

Getting the theta wrong was resulting in images with wavelike behavior, dramatically increasing the high frequency content. The right theta was actually the one that had the most low-frequency content, tested indirectly by looking at low coefficients. We implemented this approach and swept across possible thetas, first in large steps and then more finely.

Once we had the proper theta, we wanted to approximate the proper magnitude/length of the blur. We fixed the theta at the proper direction and then swept the possible range of length, first with large increments and then more finely. When we overshot the length, the situation was similar to an improper theta estimation; the resulting image had wavelike behavior that resulted from unwanted high-frequency content. Figure 7 shows the resulting image in the time and frequency domain when using the correct theta but an overestimate of the length.

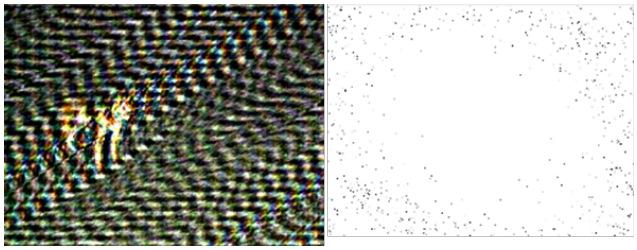


Figure 7: Image in spatial and frequency domains with proper theta but length overestimate (estimate $\theta = 45$, len=25;
actual $\theta = 45$, len=15)

However, when the length was an underestimate, the image had a more traditional blurry behavior. Figure 8 shows the resulting image in the time and frequency domain when using a the correct theta but an underestimate of the length.

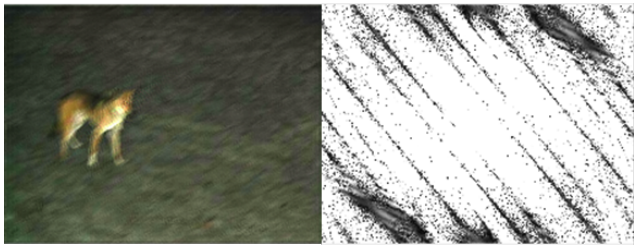


Figure 8: Image in spatial and frequency domains with proper theta but length underestimate (estimate $\theta = 45$, len=5;
actual $\theta = 45$, len=15)

We thus had to modify our algorithm so that it would not only filter out images that had too many high frequencies, but also prevent images with too many low frequencies from being considered optimal. To do so, we changed from using a low-pass filter for theta selection to a mid-pass filter for length selection. We now counted frequencies with a coefficient that was both below a certain threshold and above 0.005 of that threshold. Doing so consistently (across

lengths, thetas, and images) yielded outputs that were as appealing as or more appealing than traditional Wiener deconvolution. Figure 9 shows the resulting image in the time and frequency domain when using the correct theta and length.

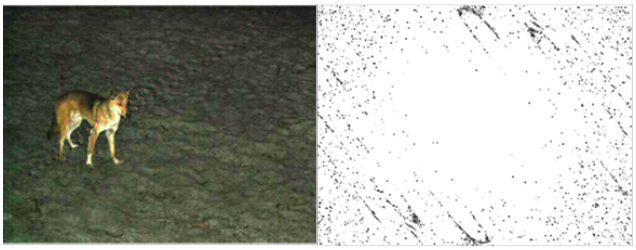


Figure 9: Image in spatial and frequency domains with proper theta and length (estimate $\theta = 45$, len=15; actual $\theta = 45$, len=15)

The approach still tended to leave some of the artifacts that we had previously witnessed, but that seems to be a shortcoming of Wiener deconvolution, which our approach uses for the actual deconvolution component. We postulate that using ADMM as the underlying method would change that. However, because we used Wiener deconvolution approach, our algorithm sometimes differed from the underlying length and chose to under-deblur to avoid creating the high-frequency artifacts.

3.1.3 Evaluation and Comparison to Other Methods

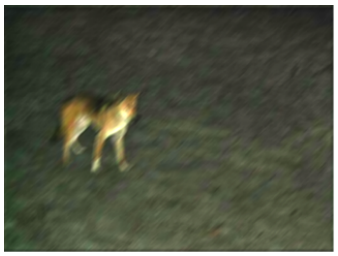
Compared to MATLAB's deconvblind() function, our approach was significantly more appealing, outperforming the former qualitatively. However, it also had a significant quantitative improvement over the MATLAB function. When Wiener deconvolution performed well on its own and the parameters were integers, our approach mirrored it exactly. When the underlying deconvolution was a poor fit (see Figure 10) or the parameters were not integers, our algorithm still generated a higher PSNR (see Table 1) than the MATLAB deconvblind() function. Figure 11 shows a direct comparison of our algorithm versus MATLAB's blind deconvolution algorithm. Looking at the results, it is clear that our algorithm outperformed MATLAB's blind deconvolution algorithm.

3.1.4 Image Denoising

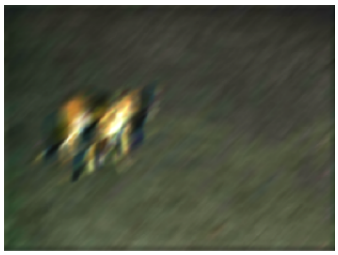
Once we knew that we could use our algorithm's output just as effectively as we could use the output of a method that required us to know the blur kernel, we fed it into the next step of the pipeline: image denoising. Here, we planned to deal with the noise that was introduced during both the image brightening and image deblurring phases. To do so, we evaluated three methods: a simple Gaussian blur kernel, a bilateral filter, and non-local means (NLM). Figure 12 shows the denoising results using the three different methods.

(a) Deblurred image via basic Wiener Deconvolution (b) Deblurred Image using our algorithm

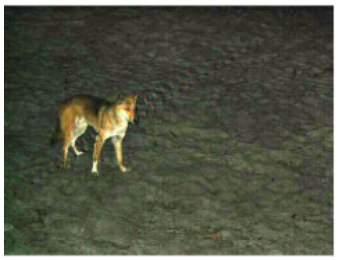
Figure 10: Wiener Deconvolution of Image (left) and Our Algorithm's Output (right) (estimate $\theta = 45$, len=5; actual $\theta = 45$, len=15)



(a) Brightened Motion Blurred Image



(b) Deblurred Image using MATLAB's deconvblind()



(c) Deblurred Image using our algorithm

Figure 11: Comparison of Our Algorithm witih MATLAB's deconvblind() function

Although the differences between the output of the Gaussian filter and the bilateral filter are subtle, the latter does a better job of edge preservation. While that might make it

PSNR Compared to Wiener Deconvolution	MATLAB deconvblind()	Our Algorithm
Integer Length (30) and Theta (45), Good Wiener Deconvolution performance	26.8795	Inf
Non-Integer Length (29.5) and Theta (87.2), Good Wiener Deconvolution Performance	23.7992	42.6849
Poor Wiener Deconvolution Performance	26.6886	27.7409

Table 1: PSNR Comparison between MATLAB's deconvblind() function and our algorithm

also preserve the artifacts, we do not think it does so to an unreasonable extent. For most simple uses, it should suffice as the method of choice. NLM, by contrast, does the best job of getting rid of the artifacts. It does, however, take significantly longer than the other methods and blurs the image immensely, making it lose much of the sharpness we hoped to gain during the deblurring process. For some, it may be an artistic choice because it does preserve the subject better than the background and thus brings it to the forefront of the viewer's attention, but we think that for most uses, a bilateral filter is the most reasonable and time-effective approach.

3.2. Results

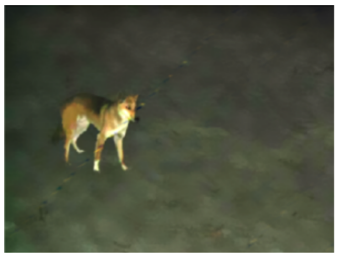
Our pipeline thus succeeds in taking an image from dark and blurry to bright, denoised, and with reasonable levels of noise. Although the artifacts persist, the final images are a significant improvement over the starting product. Figure 13 shows the complete pipeline of brightening, deblurring, and denoising for one image. Additionally, Figure 14 shows various low-light motion blurred images before and after going through our pipeline. It is clear that even though some artifacts persist, the overall results show great improvement in comparison to the initial image. Our pipeline effectively brightens, deblurres and denoises the low-light motion blurred images to achieve a bright sharp image.

3.3. Discussion and Conclusion

Our pipeline, complete with our novel approach to blind deconvolution, might not be the go-to method for generating the highest quality images, but it goes a long way towards showing the degree to which the trade-offs in an image processing pipeline can be mitigated. Within our constraints, we managed to generate outputs with minimal artifacts for several images, even across a wide range of angles and magnitudes of motion blur.

(a) Deblurred image Denoised via Gaussian Filter

(b) Deblurred image Denoised via Bilateral Filtering

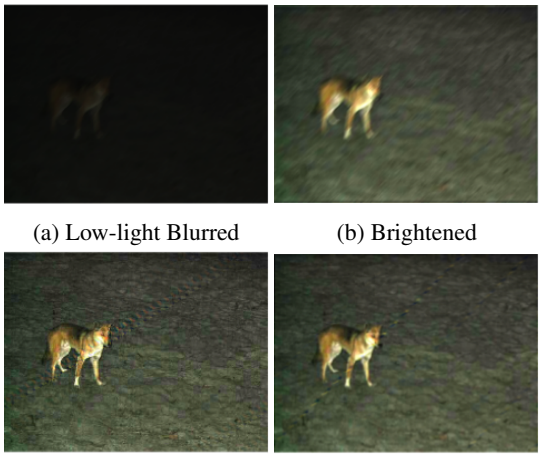


(c) Deblurred image Denoised via NLM

Figure 12: Comparison of denoising algorithms

Although it is nice to see that the trade-offs between noise and blur are not so drastic that they prevent a pipeline from taking into account how the qualities interact, the main benefit here lies in the deblurring component. Our algorithm consistently outperformed MATLAB's, qualitatively (see figure 11) and quantitatively (see Table 1), demonstrating that there is much room for growth in the area of blind deconvolution.

Perhaps most surprising was the fact that our algorithm at times created more appealing images than Wiener deconvolution where the blur kernel was known. While it is possible that the results would have been better had we just used ADMM as the underlying deconvolution method, we at least see from this that there are ways to improve on existing methods when dealing with motion blur.



(a) Low-light Blurred (b) Brightened

(c) Deblurred (d) Denoised

Figure 13: Full Pipeline for One Image – Dark and Blurry → Brightened → Deblurred → Denoised

3.4. Future Considerations

The success of our basic approach to motion deblurring, relative to the existing methods even where the blur kernel is known, demonstrates the opportunities for blind deconvolution. Many of the state-of-the-art approaches rely on deep learning, and that should allow for a relaxation of constraints (such as the one we used – motion blur in a single direction with a magnitude of less than 30 pixels). It should also create room for the generation of new image quality metrics that take into account how images change during attempts to motion blur and deblur them. Furthermore, our approach considered the scenario where the motion blur is uniform across the entire image. We would also be interested in a scenario where the subject of the image is in motion but the rest is still. This would be interesting to consider in the context of low-light wildlife photography, which would directly build on our initial motivation for this project.

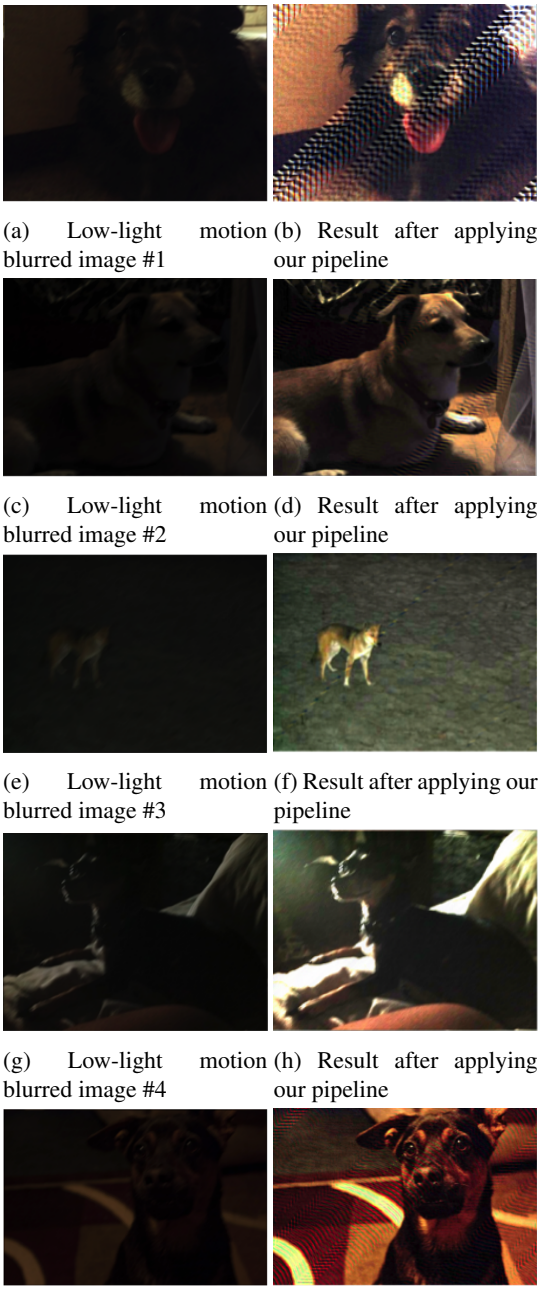
References

[1] B. Choi, D. Kim, W. Kyung, and Y. Ha. Multi-spectral flash imaging under low-light condition using optimization with weight map. *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 33–39, 2014.

[2] Z. Hu, S. Cho, J. Wang, and M. Yang. Deblurring low-light images with light streaks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2329–2341, Oct 2018.

[3] F. Li, Y. Ji, and J. Yu. A hybrid camera system for low-light imaging.

[4] S. McCloskey. Temporally coded flash illumination for motion deblurring. *2011 International Conference on Computer Vision*, pages 683–690, Oct 2011.



(a) Low-light motion blurred image #1 (b) Result after applying our pipeline

(c) Low-light motion blurred image #2 (d) Result after applying our pipeline

(e) Low-light motion blurred image #3 (f) Result after applying our pipeline

(g) Low-light motion blurred image #4 (h) Result after applying our pipeline

(i) Low-light motion blurred image #5 (j) Result after applying our pipeline

Figure 14: Comparison of low-light motion blurred images and the result after passing the images through our pipeline

[5] J. Wang, T. Xue, J. T. Barron, and J. Chen. Stereoscopic dark flash for low-light photography. *2019 IEEE International Conference on Computational Photography (ICCP)*, pages 1–10, 2019.