

Blind Text Deblurring with Flutter Shutter

Kathleen Feng, Daniel Stanley
Stanford University
450 Jane Stanford Way
{kzf, dstanley}@stanford.edu



Figure 1: Using our method with images that are flutter shuttered results in clearer and more readable text when compared to images that are captured using a single exposure.

Abstract

Photos taken from handheld devices often have some sort of motion blur; natural handshake and taking photos from a moving vehicle make it hard to take clear photos. In addition, being able to extract text from blurry photos is very beneficial to those who are visually impaired. In this paper, we propose a method to perform blind text deblurring: deblurring without knowledge of the exact blur kernel, or point spread function (PSF), on both motion blurred images and images that are captured using the flutter shutter technique. The blur kernel is learned from the given image using a least squares minimization approach, and the kernel is then processed to make it cleaner. We use the estimated kernel for deconvolution, using ADMM with a total variation prior. We show that images that are captured using flutter shutter and deblurred using our approach have higher quality than regular motion blurred images.

1. Introduction

Many photos today are captured with handheld devices, such as smartphones and tablets. The increased availability and usability of mobile devices have made it easier to capture photos anywhere, but the quality of the images varies widely with the environment in which the image is taken. Unsteady hands may introduce unwanted blur to the photos, and taking images from a moving vehicle, such as a

car or a train, also makes it harder to achieve a crystal clear image. Deblurring motion blurred images has been studied in the past, and many image processing pipelines on smartphones have an image stabilization stage or deblurring stage [5, 10]. Single image deblurring has also been widely studied before [7, 4, 13, 9, 8, 16], often using priors for natural images.

Text deblurring has a number of important applications, and there has also been a lot of work developing text-specific priors to estimate the blur kernel for a text image. Being able to extract text from photos is especially useful for individuals who are visually-impaired and need a way to understand signs and labels around them. Text priors for document images, or binary text images [3], as well as two-toned images [14] have been proposed. More recently, [11] proposes using an L_0 -regularized prior for text deblurring.

Typically, photographs are taken with the shutter open for the entire exposure time. However, the images do need not be taken with a single exposure. An alternate method, flutter shutter [12], instead proposes opening and closing the camera’s shutter according to a given pattern during the original exposure time. Flutter shutter preserves high-frequency details and turns deconvolution into a well-posed problem. Flutter shutter has been combined with other imaging techniques, such as superresolution [1].

In this paper, we propose combining establishing techniques of text deblurring and flutter shutter in order to achieve higher quality blind text deblurring. We estimate and learn the blur kernel from a single image, process the

kernel to clean any artifacts and noise, and use a standard deconvolution technique to recover the unblurred image. The approach can be used without knowing the flutter shutter pattern and using the flutter shuttered output image as the sole input. However, we also introduce a way to incorporate the flutter shutter pattern and improve the quality of deblurring. The paper makes the following contributions:

1. Establish an image processing pipeline to deblur a single image without knowing the blur kernel
2. Propose two methods to improve the estimated blur kernel, with and without using the flutter shutter pattern
3. Demonstrate the advantage of using flutter shutter to capture motion blurred images instead of using a single exposure

2. Related Work

Previous work regarding flutter shutter, text deblurring, and traditional deblurring using deconvolution are described in this section.

2.1. Flutter Shutter

Flutter shutter [12] is a technique for capturing photos without using a single exposure. Instead, the shutter is opened and closed rapidly according to a specified pattern during the original shutter time. Flutter shutter has proven to be a useful technique for deblurring blurry images due to motion because it preserves higher spatial frequency details that allow the deconvolution process to become a well-posed problem. Fluttering the shutter results in a coded blur and PSF that improves invertibility of the imaging process. Instead of using deconvolution algorithms, the original authors use a linear algebra approach, which allows them to incorporate different constraints, such as motion of partially-occluded background objects. However, the techniques proposed in [12] rely on the knowledge of the PSF and does not estimate the PSF.

The flutter shutter technique can also enhance the resolution of an object in motion [1]. Similar to above, coding the exposure improves the invertibility of the resolution enhancement problem. Due to hard edges and sharp changes in color and pattern in text images, flutter shutter seems like a good candidate for clearing up blurry text.

2.2. Text Deblurring

There has been a lot of effort already completed on achieving high-resolution text images from low-resolution or blurry inputs. Deblurring algorithms, such as maximum likelihood, IBP, POCS, MAP, and priors, such as Tikhonov and TV, have all been used for text deblurring [15]. More advanced priors exist as well, such as a Markov Random Field informed by example images [6]. [11] proposes using

an L_0 -regularized prior to perform text deblur. Text characters and their backgrounds tend to each have uniform intensity values, and pixels intensities tend to be sparse and peak near the minimum and maximum values. Blurry images tend to have denser intensity distributions. Establishing a prior that focuses on pixel intensity, as well as their gradients, works well for text.

The authors of [11] also estimate the blur kernel using a least squares minimization problem that can be solved efficiently using FFTs. While this approach performs better than existing and state-of-the-art approaches, it uses single-exposure, motion blurred images. Using images that are captured using the flutter shutter technique may increase quality even more.

2.3. Deconvolution

Deconvolution refers to finding an estimate of the original image from a blurry and/or noisy measurement of the image. There are a number of approaches to deconvolution. Inverse filtering, while the most straightforward, is often ill-posed and impractical. Other options include Wiener filtering, which adds a damping factor to inverse filter. The total variation (TV) approach preserves edges by promoting sparse gradients. Deconvolution can be solved iteratively using the alternating direction method of multipliers (ADMM) [2]. ADMM using the TV prior splits the problem into three steps per iteration and works well with text because the TV prior preserves edges. However, deconvolution requires the blur kernel and does not perform estimation or learn the kernel from the blurry image.

3. Methodology

To perform blind text deblurring, we use a three-step process that estimates the blur kernel given one image, processes the kernel to hopefully better match the actual blur kernel, and uses the learned kernel in deconvolution to recover the unblurred, clear image.

3.1. Test Image Generation

In order to compare our reconstructed images with ground truth, it is helpful to artificially generate the motion blurred and flutter shuttered images; to do this we need to define the blur kernel. We make the assumption that the object is perfectly in focus and that the direction of motion is a straight line [12]. We can then generate a PSF for a given direction and speed of motion. We break up the long camera exposure into many discrete time instances and superimpose the contribution of one moving pixel at each of those times. For simple motion blur this results in a straight line with total weight one, while for flutter shutter this line is chopped depending on what times the shutter is open and

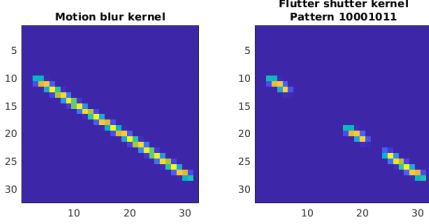


Figure 2: Comparison of PSFs for motion blur and flutter shutter approaches.

closed (Figure 2). To produce the final image we simply convolve an in-focus image with this PSF.

3.2. Pipeline

Our full reconstruction pipeline for images with unknown blur kernel consists of 3 steps, as shown in Figure 3. We will discuss each step separately. Steps 1 and 3 are based closely on prior work, while step 2 is specific to our approach.

3.2.1 Kernel Learning

We first approximate the blur kernel based only on the blurred image and a text prior. We guess a uniform blur and try to reconstruct the image according to the prior described in [11]. Treating this reconstruction as ground truth, we then try to determine the blur kernel. We find the kernel that minimizes the difference between the gradients of the blurred and reconstructed images. We alternate between these steps, and with each iteration the reconstruction improves because of the added information from the text prior.

If the actual kernel is smaller than the kernel matrix used in this algorithm, all translations of the actual kernel that fit within the matrix are equally valid answers. Unfortunately, if we run this algorithm with such a large matrix many superimposed copies appear at once with different translations. This issue is fixed using an image pyramid: learning begins with a lower-resolution image and smaller kernel matrix, and throughout learning both are upsized together. This encourages one particular translation of the actual kernel to dominate. There are still other faint copies at other translations, which is one reason we need step 2 (Section 3.2.2), and the translation of the dominant kernel is somewhat random leading to difficulty in comparison to the ground truth (Section 3.3).

As seen in Figure 4, the PSNR of the reconstructed image improves with each iteration of estimating the kernel, showing that the estimated kernel continues to more accurately represent the actual kernel.

3.2.2 Kernel Fitting

Although the kernel learning algorithm consistently finds the general shape of the kernel there is significant noise in the final output. We propose two strategies for reducing this noise based on the assumption that the shape of the kernel is due to motion blur.

The simpler strategy is to determine a threshold below which any value is considered noise. The threshold value is set as a percentage of the maximum value in the learned kernel. These low values are set to zero and the entire kernel is renormalized. This strategy has the advantage that it works even if the flutter shutter pattern is not known.

If the flutter shutter pattern is known, or the image has a regular motion blur kernel, we can use a more in-depth fitting method. The first step is to check a number of straight lines across the image at different angles and positions to find the one that overlaps the most bright pixels. The position of this line is refined until it lies exactly along the main kernel blur. Next, a variety of blur lengths and positions along the line are considered to see which best matches the learned kernel. Once the exact position, direction, and length of the blur are determined we discard the noisy learned kernel and generate a clean kernel based on those extracted parameters.

3.2.3 Reconstruction with Fitted Kernel

Once we have determined a kernel we can reconstruct the original image using a method that requires the kernel to be known. It is important to note that the kernels used at this stage are usually still significantly noisy, and this affects the choice of reconstruction algorithm. We use ADMM with a total variation prior because we found it to be the most successful under these conditions.

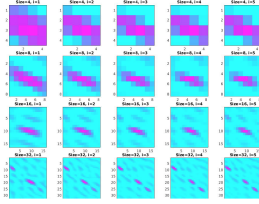
3.3. Comparison to Ground Truth

We use PSNR to quantify how well images are reconstructed. Unfortunately our learned kernel is sometimes translated spacially from the one used for blurring so our reconstructed image is translated compared to the ground truth. This is not an issue in a real-world context because we care only about the quality of the visible text, not its exact position within the frame. To solve this issue we simply calculate PSNR for many translations of the reconstructed image and take the highest result. We do not consider sub-pixel shifting because it smears the sharp edges enforced by our reconstruction prior.

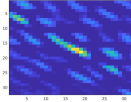
3.4. Choice of Parameters

Our complete reconstruction method requires parameters for ADMM, for text deblurring, and for kernel learning. Although kernel learning makes calls to the text deblurring al-

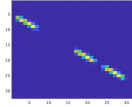
Step 1: Learn the blur kernel using text-specific estimation algorithm from [4]



Step 2: If have the flutter shutter pattern, use pattern to clean learned kernel



Flutter pattern:
1000101



Step 3: Use ADMM to perform deconvolution using learned kernel

$$\begin{aligned} &\text{minimize } f(x) + g(z) \\ &\text{subject to } Ax + Bz = c \end{aligned}$$

$$\begin{aligned} L_p(x, z, y) = &f(x) + g(z) \\ &+ y^T(Ax + Bz - c) \\ &+ \frac{\rho}{2} \|Ax + Bz - c\|_2^2 \end{aligned}$$

$$x^{k+1} = \arg \min_x L_p(x, z^k, y^k)$$

$$z^{k+1} = \arg \min_z L_p(x^{k+1}, z, y^k)$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)$$

Figure 3: Three-step approach for blind deblurring for text images.

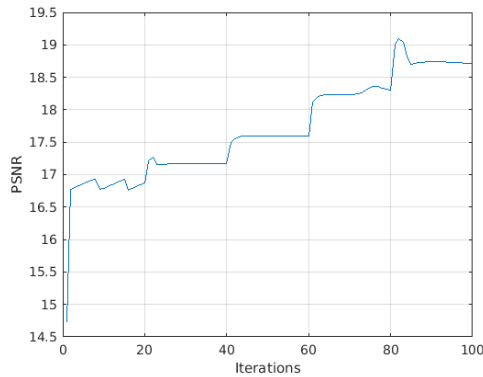


Figure 4: PSNR of reconstructed image as the kernel is learned. Resolution is normally increased every 5 iterations; in this visualization, it is 20 iterations to demonstrate the plateau at each resolution.

gorithm from [11], we allow a separate set of parameters for text deblurring depending on which context it is being used in.

Because some of the parameters represent a ratio between L2 norm of the image and L0 norm of the image gradient, they vary significantly based on the background color (L2 norm is proportional to contrast, L0 is constant) and text size (number of pixels proportional to square of image size, number of gradient edges linear). As a result we allow parameters to be chosen separately for each test image. In a realistic context this amount of fine-tuning is not possible, but we imagine parameters can be chosen for a known class of image (one page of a book, a license plate, etc.).

Finally, we tuned parameters separately for motion blur and flutter shutter images to be sure we were comparing the best version of each method.

Our final choices for parameters are shown in Ta-

ble 1. More detailed information can be found by looking at the code at <https://github.com/norabarlow/ee367>.

4. Results

Using both motion blurred images and flutter shuttered images, we evaluate our three-step approach on a document image. Table 2 compares reconstruction of flutter shutter images to regular motion blur on the image. Using the learned kernel without any cleanup in ADMM results in the worst image quality visually, and the PSNR is the same for both cases. The motion blurred image after using the learned kernel is not as blurry but the text is almost unreadable. The flutter shuttered image is readable, but there are a lot of shadows and artifacts; remnants of the text can be seen in light gray around the black text.

By simply eliminating values in the learned kernel that fall below a certain threshold, the image quality increases for both images. Here the threshold is 15% of the maximum value in the kernel. The flutter shuttered image is better than the motion blurred image. The PSNR of the motion blurred image increases from 11.35 dB to 13.45 dB, while the PSNR of the flutter shutter increases to 15.10 dB from the same 11.35 dB. Fitting the kernel to the pattern increases PSNR even more for both images. While the motion blurred image still has some background noise, the text is much more readable and complete. PSNR also increases. The flutter shuttered image looks even better, with an almost-white background with very little background noise and clear, readable black text. Its PSNR is the highest at 16.80 dB. Using flutter shutter and the coded aperture approach appears to preserve more edges in the text characters. The letters look less blurry because the edges are more well-defined.

	Motion blur				Flutter shutter			
	Soft Block	Plane	Text	Pippin	Soft Block	Plane	Text	Pippin
ADMM: ρ	1e1	1e1	1e1	1e1	1e1	1e1	1e1	1e1
ADMM: λ	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
Deblur: σ	1e-2	1e-3	1e-3	1e2	1e-4	1e-3	1e-4	1e-5
Deblur: λ	3e0	1e-5	1e-5	1e-2	1.5e-1	1e-5	1e-5	1.5e-1
Deblur: β_{max}	1e-1	6e0	6e0	1.5e1	5e-3	6e0	6e0	5e-3
Deblur: μ_{max}	3e0	1e3	1e3	1e3	3e-1	1e3	1e3	3e-1
Deblur, learning: σ	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
Deblur, learning: λ	1.5e-1	1.5e-1	1.5e-1	1.5e-1	1.5e-1	1.5e-1	1.5e-1	1.5e-1
Deblur, learning: β_{max}	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3
Deblur, learning: μ_{max}	3e-1	3e-1	3e-1	3e-1	3e-1	3e-1	3e-1	3e-1

Table 1: Table of parameters used in various image reconstruction steps.

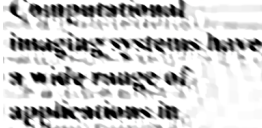
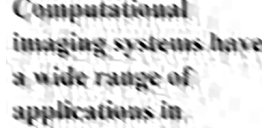
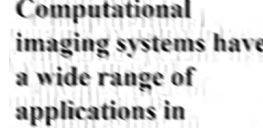
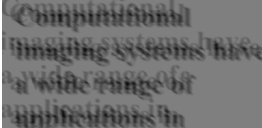
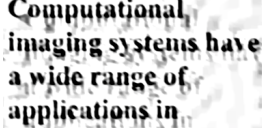
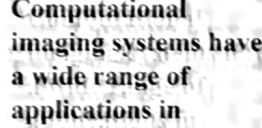
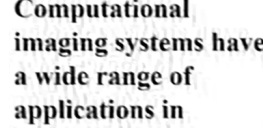
	Blurred Image	No Kernel Clean	Threshold Kernel	Fitted Kernel
Motion Blur		PSNR = 11.35 dB 	PSNR = 13.45 dB 	PSNR = 15.11 dB 
Flutter Shutter		PSNR = 11.35 dB 	PSNR = 15.10 dB 	PSNR = 16.80 dB 

Table 2: The proposed approach on both motion blurred and flutter shuttered images. “No kernel clean” uses the learned kernel without any cleanup. “Threshold kernel” cleans the learned kernel using a simple threshold to reduce noise. “Fitted kernel” uses the flutter shutter pattern to fit the learned kernel to the pattern.

5. Analysis and Evaluation

To evaluate the effectiveness of the proposed approach for blind deblurring of text images, we compare against existing deblurring and deconvolution algorithms as well as different variations of the proposed approach. Figures 5 and 6 compare different reconstruction approaches for motion blur and flutter shutter images, respectively. We also evaluate the choice of the flutter shutter pattern; Figure 7 compares images reconstructed with different flutter shutter patterns. The images that we choose include one document image with black text and white background, one text image with a little bit of color, and two non-document, full-color, text images.

5.1. Comparison to Previous Work

We compare our approach with previous approaches, with and without the learned kernel as well as with and without the various learned kernel fitting schemes. We also test and compare our proposed approach on images that are both motion blurred and flutter shuttered. The following schemes are compared:

1. ADMM using the known, exact blur kernel
2. Text deblurring algorithm from [11] using the known, exact blur kernel
3. Text deblurring algorithm using the learned kernel
4. ADMM using the learned kernel
5. ADMM using the learned kernel after applying a threshold
6. ADMM using the learned kernel after fitting it to the flutter shutter pattern

Figure 5 shows the comparison between motion blurred

images. Figure 6 shows the comparison between flutter shuttered images. The PSNR of each image, compared to the original image, is calculated.

Using the actual PSF with ADMM or with the text deblurring algorithm usually results in the best images, both in terms of PSNR and image quality. As expected, having the exact blur kernel helps a lot with the deblurring problem. However, the blur kernel may not usually be known in practice. Looking at schemes four through six, which is our blind three-step process with the learned kernel, in almost all cases, with the exception of the “Pippin” image between the thresholded and fitted learned kernel for flutter shutter, the PSNR increases with each scheme number and the image is visually less blurry. Any artifacts, such as ringing, are often resolved by using a simple threshold to clean up the learned kernel or using the flutter shutter pattern to clean the kernel. The thresholded learned kernel improves the image quality over just using the learned kernel, and using the kernel fitted to the flutter shutter pattern improves image quality even more.

In addition, the flutter shuttered images, with the exception of “Pippin”, all have higher PSNRs when using our approach with the fitted learned kernel. The images look clearer as well. The text in the plane image (second from the left column) is completely unreadable with the motion blurred image, while with flutter shutter the edges of the letters are visible, even if they are still a bit blurry. The edges of the letters of the “SOFT BLOCK” text (leftmost column) are much clearer and more sharp than the letters in the regular motion blurred images. In addition, fitting the learned kernel to the flutter shutter pattern allows our method to outperform the deblurring algorithm proposed in [11].

5.2. Choice of Flutter Shutter Pattern

Past work with flutter shutter has used long patterns of opening and closing the shutter over the course of one exposure, with Raskar et al. choosing a length of 52 bits for their near-optimal code [12]. We found that longer patterns require long motion blur so that each section of the pattern maps to at least one pixel on the PSF. Because of the image pyramid technique it is difficult for our kernel learning algorithm to learn the PSF of a long motion blur. For these reasons, we found it more effective to use a short flutter shutter pattern so that each bit of the pattern corresponds to a longer spacial section of the PSF. Figure 7 shows the effects of varying patterns. In most cases the short (length 8) pattern works best.

6. Discussion and Limitations

Our approach makes strict assumptions about the shape of the blur kernel that may not hold true in every real-life situation. Particularly, the kernel recognition step will not

Motion Blur				
Original			Computational imaging systems have a wide range of applications in	
Blurred				
	PSNR = 23.47	PSNR = 21.52	PSNR = 15.64	PSNR = 14.49
ADMM			Computational imaging systems have a wide range of applications in	
	PSNR = 18.77	PSNR = 27.54	PSNR = 35.63	PSNR = 7.42
Text Deblur(TD)			Computational imaging systems have a wide range of applications in	
	PSNR = 18.58	PSNR = 17.67	PSNR = 11.60	PSNR = 15.39
TD with Learned Kernel (LK)			Computational imaging systems have a wide range of applications in	
	PSNR = 18.18	PSNR = 16.94	PSNR = 11.35	PSNR = 13.90
ADMM with LK			Computational imaging systems have a wide range of applications in	
	PSNR = 20.10	PSNR = 20.64	PSNR = 13.45	PSNR = 13.97
ADMM with Threshold LK			Computational imaging systems have a wide range of applications in	
	PSNR = 22.67	PSNR = 21.22	PSNR = 15.11	PSNR = 14.41
ADMM with Fitted LK			Computational imaging systems have a wide range of applications in	

Figure 5: Comparison of different techniques with images that are motion blurred.

effectively match the real kernel if the direction of motion is curved or the camera is out of focus. These problems could be mitigated by a more sophisticated algorithm that considers more nonidealities, but performance would be lower as noise in the learned kernel is mistaken for these nonidealities.

Our approach also does not seem to work well with images with colored text and backgrounds. Both the plane images and the Pippin images were the least readable, compared to the other two images, which were more like document images rather than natural images. “Pippin” also is worse with flutter shutter, which we have yet to find an explanation for.

7. Conclusion

We find that in cases where the blur kernel is not known, our methods for cleaning the learned kernel improve upon the results of existing algorithms (Figure 5). Additionally, we agree with previous work that using flutter shutter improves the reconstruction of high-frequency images compared to a single long-exposure image (Table 2). We hope that in the future more handheld cameras will provide the option to flutter the shutter so that techniques like the ones

Flutter Shutter

Original			
Blurred			
ADMM	PSNR = 25.49 	PSNR = 22.59 	PSNR = 18.25
Text Deblur(TD)	PSNR = 21.60 	PSNR = 30.33 	PSNR = 14.52
TD with Learned Kernel (LK)	PSNR = 19.71 	PSNR = 17.32 	PSNR = 17.85
ADMM with LK	PSNR = 18.24 	PSNR = 11.76 	PSNR = 13.34
ADMM with Threshold LK	PSNR = 11.35 	PSNR = 10.91 	PSNR = 11.89
ADMM with Fitted LK	PSNR = 23.86 	PSNR = 13.05 	PSNR = 15.10
	PSNR = 24.43 	PSNR = 22.23 	PSNR = 13.04
	PSNR = 24.43 	PSNR = 22.23 	PSNR = 13.03

Figure 6: Comparison of different techniques with images that are flutter shuttered.

Blur kernel	PSNR = 22.67 	PSNR = 21.23 	PSNR = 15.01 	PSNR = 14.41
Flutter shutter 11010001	PSNR = 24.43 	PSNR = 22.23 	PSNR = 16.80 	PSNR = 13.03
Flutter shutter 101100101000111	PSNR = 24.35 	PSNR = 19.51 	PSNR = 15.22 	PSNR = 14.28
Flutter shutter 111100000100011001001001110101101	PSNR = 18.37 	PSNR = 21.56 	PSNR = 11.68 	PSNR = 13.13

Figure 7: Comparison of different lengths of flutter shutter pattern. Images shown are after fitting kernel and reconstructing with ADMM, equivalent to the bottom rows of Figures 5 and 6.

presented here can lead to clearer images and more readable text.

References

[1] A. Agrawal and R. Raskar. Resolving objects at higher resolution from a single motion-blurred image. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*,

pages 1–8, June 2007.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1122, Jan. 2011.

[3] X. Chen, X. He, J. Yang, and Q. Wu. An effective document image deblurring algorithm. In *CVPR 2011*, pages 369–376, June 2011.

[4] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH 06, page 787794, New York, NY, USA, 2006. Association for Computing Machinery.

[5] K.-F. Henning, A. Fritze, E. Gillich, U. Mnks, and V. Lohweg. Stable image acquisition for mobile image processing applications. volume 9404, 02 2015.

[6] Jangkyun Park, Younghee Kwon, and Jin Hyung Kim. An example-based prior model for text image super-resolution. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 374–378 Vol. 1, Aug 2005.

[7] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VII, ECCV12*, page 2740, Berlin, Heidelberg, 2012. Springer-Verlag.

[8] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR 2011*, pages 233–240, June 2011.

[9] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR 2011*, pages 2657–2664, June 2011.

[10] K. Liu, J. Qian, R. Yang, and X. Luo. Improved image stabilization algorithm based on region of interest. In *2010 3rd International Congress on Image and Signal Processing*, volume 2, pages 716–719, Oct 2010.

[11] J. Pan, Z. Hu, Z. Su, and M. Yang. Deblurring text images via l0-regularized intensity and gradient prior. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2908, June 2014.

[12] R. Raskar, A. Agrawal, and J. Tumblin. Coded exposure photography: Motion deblurring using fluttered shutter. *ACM Trans. Graph.*, 25:795–804, 07 2006.

[13] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH 08, New York, NY, USA, 2008. Association for Computing Machinery.

[14] Ta-Hsin Li and Keh-Shin Lii. A joint estimation approach for two-tone image deblurring by blind deconvolution. *IEEE Transactions on Image Processing*, 11(8):847–858, Aug 2002.

[15] C. Thillou and M. Mirmehdi. *An Introduction to Super-Resolution Text*, pages 305–327. 03 2007.

[16] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1107–1114, June 2013.