Removing Motion Blur from Barcode Images

Saeed Yahyanejad[1]
Institute of Networked and Embedded Systems
Klagenfurt University, Austria
saeed.yahyanejad@uni-klu.ac.at

Jacob Ström
Ericsson Research
164 80 STOCKHOLM, Sweden
jacob.strom@ericsson.com

Abstract

Camera shake during photography is a common problem which causes images to get blurred. Here we choose a specific problem in which the image is a barcode and the motion can be modeled as a convolution. We design a blind deconvolution algorithm to remove the translatory motion from a blurred barcode image.

Based on the bimodal characteristics of barcode image histograms, we construct a simple target function that measures how similar a deconvoluted image is to a barcode. We minimize this target function over the set of possible convolution kernels to find the most likely blurring kernel. By restricting our search to dome-shaped kernels (first monotonously increasing and then monotonously decreasing) we decrease the number of false solutions. We have tried our system on a collection of a 138 barcode images with varying camera blur, and the recognition rate increases from 32% *to* 65%.

1. Introduction

Today, barcodes can be found on numerous items, such as packaged food, books, newspapers and more. There exist different ways of reading these machine readable codes. One way is to use dedicated barcode readers, but especially for general purpose devices such as mobile phones, including a dedicated barcode reader may be expensive and take up valuable space. An alternative is to acquire an image of the barcode using the camera that is anyway built into the device and process the image in order to decode the barcode. Reading barcodes by image processing may be slower and less reliable than using dedicated barcode scanners, but in some cases they are better: When reading barcodes on a monitor screen, for instance, dedicated systems based on reflected laser light do not work.

Sometimes however, barcode images acquired with cameras will not have the required quality to be recognized and decoded. This could happen for a large number of reasons, including difficult lighting, out-of-focus blur, etc, but in this paper we focus on motion blur, especially the type that can be modeled as a convolution. We will deconvolute the image using an existing, non-blind deconvolution algorithm and a starting kernel. We will then modify the kernel so that our target function (acting on the deconvoluted image) is minimized. After minimization, we will have both the kernel and the deblurred image. Hence our deconvolution method is blind. The question of how frequently people end up with such blurred images highly depends on the use-case, type and settings of the camera and other parameters. For instance, photography in poor lighting conditions results in a noisy image. In the case of barcode images it may be compensated by noise reduction, but as an alternative we may have the shutter open longer to get more light through the camera. In this case the risk of motion blur also increases.

The next section describes related work, and it is followed by a section on the blurring model we have used. Section 4 will describe the barcode deblurring in more detail. The last three sections will treat result, discussion and conclusion.

2. Related work

Lately, there have been breakthroughs in the area of deblurring of general images. The new approach is to use statistics about image gradients, and other characteristics that most natural images have, to guide the algorithm to a correct estimation of the blurring kernel [4, 7, 19, 20]. These algorithms produce stunning results but are quite slow and sometimes need good guesses of the kernel length to give good results. Furthermore, since they are using statistics from natural images, they do not necessarily work well on barcode images which are quite different from normal images. Thanks to the authors of [20], we got access to the executable of their algorithm and tried it on barcode images. Unfortunately the results were unsatisfactory, and this can be due to the difference in image statistics between

[1]This work was done while Saeed was conducting his master's thesis project at Ericsson Research, 164 80 Stockholm, Sweden.

natural images and barcodes.

Thus it makes sense to process barcode images in a different way from regular images. Generally we know much more about how they should look, especially if they are one-dimensional. We have not found so much previous work in the particular field of barcode deblurring. Esedoglu performs optimization of a gradient-based target function based on gradient descent [3]. His algorithm aims at processing barcode signals acquired from dedicated barcode readers, and therefore solves a slightly different problem from this paper. The approach by Kim and Lee [8] takes into account general signals including barcode images. They also consider nonuniform illumination in images as a part of the deblurring algorithm, which improves the practical performance. But both mentioned methods are mainly focused on out-of-focus blur or other types of gaussian blur. This paper will concentrate mainly on motion blur, and we will show that rather large blurring kernels can be handled.

3. Blurring model

In general, an image $o(x, y)$ of size $N \times M$, which is blurred within a maximum kernel size of $L \times T$, can be written as:

$$B(x,y) = \sum_{i=1}^{L} \sum_{j=1}^{T} k_{x,y}(i,j) o(x - i + l_0, y - j + t_0) \quad (1)$$

Hence, the blurring of an image is modeled as an integration (or, in the discrete domain, summation) of different pixels in the original image which is defined by a variable kernel $k_{x,y}$, i.e., different positions do not necessarily have the same kernel. Constants l_0 and t_0 show in which part in the original image the kernel is operating. By assuming that the kernel does not change over the image, we can simplify this to a convolution:

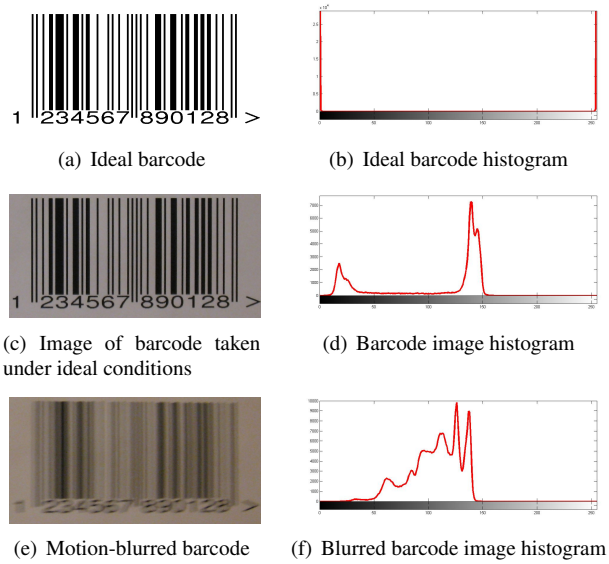
$$b(x,y) = k(x,y) * o(x,y) \equiv \sum_{i=1}^{L} \sum_{j=1}^{T} k(i,j) o(x-i+1, y-j+1), \quad (2)$$

where $B(x, y) = b(x + l_0 - 1, y + t_0 - 1)$. In real cases we have a true original image $\mathbf{o}$, which is convoluted by a blurring kernel $\mathbf{k}$. After that, noise $\mathbf{n}$ is added, and we get the blurred image $\mathbf{b}$:

$$b(x,y) = k(x,y) * o(x,y) + n(x,y) \quad (3)$$

4. Barcode deblurring

Blind deconvolution of an image to remove motion blur aims at recovering the original image as well as the kernel. As pointed out by Magain et al. [12] and Bishop et al. [1], this is essentially an under-constrained problem since many combinations of an image plus a blurring kernel can produce the same blurred image. To select a plausible solution among these, some extra constraints are needed, such as the smoothness constraint described by Magain et al. [12], or constraints on the distribution of image gradients as used by Fergus et al. [4].



(a) Ideal barcode (b) Ideal barcode histogram

(c) Image of barcode taken under ideal conditions (d) Barcode image histogram

(e) Motion-blurred barcode (f) Blurred barcode image histogram

Figure 1. The blurring effect is composed of two steps: The first step from the first row to the second row shows the conversion of an ideal barcode to an image which decreases the quantity of the exact black and white points. The second step from the second row to the last row indicates that when an image gets blurred the histogram gets more spread out. The right column shows the gray-scale distribution histogram of the left column.

Figure 1 shows how an ideal barcode gets blurred: The first image 1(a) is the ideal barcode, the second image 1(c) is an actual image of the barcode produced by the camera without motion blur, and the third image 1(e) shows the type of image we get during motion blur. As you can see in the corresponding histograms 1(b) and 1(d), as soon as we take a real image 1(c), the acquired image does not contain the exact black and white pixels any more because of noise, varying illumination, differences in light reflection, digitalization and other artifacts. We want to use deconvolution to go from images such as 1(e) to ones such as 1(c). The main reason not to go all the way to the ideal image is that reversing the first step (going from Figure 1(c) to Figure 1(a)) requires a nonlinear function, whereas reversing the second step (going from Figure 1(e) to Figure 1(c)) can be handled by a (linear) deconvolution. When the number of unknown parameters increases, which occurs in our case, the number of possible solutions grows exponentially. On the other hand we know that the original image is a barcode which is blurred as a result of a relative translational movement

between camera and barcode object which can be modeled as a convolution. The basic idea is to start by guessing a kernel, deconvolute the blurred image using the kernel and then evaluate how close to a barcode the deblurred image is. We will then change the kernel, evaluate again and use the new kernel if the new deblurred image is more similar to a barcode. The main contribution of this paper is the evaluation function with which we measure the similarity of an image to a barcode. We will now go into more detail.

We will use the Wiener-filter based version of the MATLAB function `deconv`. Assuming we have a starting guess for the kernel $\mathbf{k}$ and the noise-to-signal ratio n, we can get an estimation of the original signal by

$$\hat{\mathbf{o}} = Deconv(\mathbf{b}, \mathbf{k}, n), \tag{4}$$

where $\mathbf{b}$ is the blurred image. We then construct a target function $\lambda(\cdot)$ that models the deviation from a real barcode image by checking the characteristics which real barcodes should possess. This means that a low value of the target function $\lambda(\hat{\mathbf{o}})$ indicates that $\hat{\mathbf{o}}$ resembles a barcode well — a high value analogously indicates that $\hat{\mathbf{o}}$ is far from resembling a barcode. Since the other variables except the kernel are known, the target function can be stated as a function of the kernel:

$$\lambda(\hat{\mathbf{o}}) = \lambda(Deconv(\mathbf{b}, \mathbf{k}, n)) = \lambda'(\mathbf{k}) \tag{5}$$

Finally we need to use an optimization algorithm to find the kernel that minimizes our target function. We treat the MATLAB function `deconv` as a black box and hence we cannot compute analytical differentials with respect to the elements of the kernel $\mathbf{k}$. Therefore we cannot use gradient based optimization methods such as gradient-descent or Gauss-Newton. Instead we use a variant of simulated annealing to do the optimization. Next, we will describe the algorithm in more detail.

4.1. Barcode deblurring algorithm

We will assume that the bars in the bar code are roughly aligned with the y-axis of the image. We will also assume that a pre-processing step has extracted the (blurred) barcode image from the rest of the image. In our test images this was done by manual cropping. The rest of our algorithm is fully automatic. As is shown in Figure 2, we start by converting the cropped-out image 2(a) to grayscale 2(b).

Since we are dealing with one-dimensional barcodes and all bars have equal information in the vertical direction, the motion vector component in the vertical direction will not affect our result (considering the bar height infinite). We can therefore convert the two-dimensional image to a one-dimensional image by averaging over the columns:

$$b(y) = \frac{1}{N} \sum_{x=1}^{N} b(x, y) \tag{6}$$

This procedure will reduce the noise since the mean value converges to the noise-free value of that column. It will also reduce the complexity. Consequently the kernel will also become one-dimensional:

$$k(y) = \sum_{x=1}^{L} k(x, y) \tag{7}$$

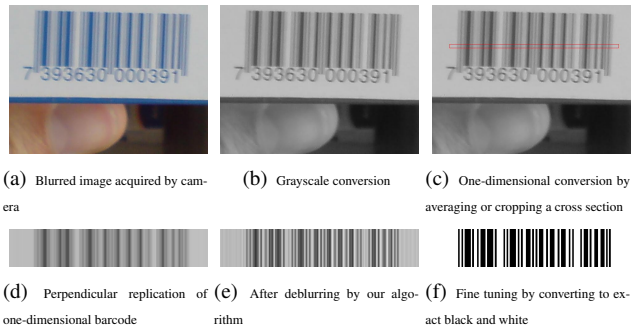
Averaging over the entire height of the barcode should remove most of the noise, but this only works well if the bar code is truly axis-aligned with the image, or if some kind of rectification is performed. Therefore we have instead averaged only over a few rows (in our case 10 rows which based on image resolution is between 2% and 5% of the barcode height) in the middle of the barcode. This works if the barcode is roughly axis-aligned.

The next step is to construct the target function λ that will measure how close we are to a real barcode. As we mentioned already, barcodes have some specific characteristics that may help us to recognize them. As seen in the second step of Figure 1, the histogram of a barcode image without motion blur will have a clear bimodal shape. Therefore, we design our target function to measure this bimodality of the intensity distribution. An ideal barcode is comprised of pure black (intensity=0) and white (intensity=1) pixels, but blurring produces more gray colors in the range of $(0, 1)$ and pushes the intensity distribution toward the mean. Our target function therefore favors a histogram with two peaks where the variance of each peak is small while the distance between them is large. In more detail, if the vector $\mathbf{o} = [o(1) \quad o(2) \quad \dots \quad o(M)]$ is the one-dimensional barcode, our target function $Bivar$ is

$$Bivar(\mathbf{o}) = \begin{cases} \frac{Var(\mathbf{o}_1)+Var(\mathbf{o}_2)}{[mean(\mathbf{o}_1)-mean(\mathbf{o}_2)]^2}, & |mean(\mathbf{o}_1)-mean(\mathbf{o}_2)|\leq 1 \\ \infty, & |mean(\mathbf{o}_1)-mean(\mathbf{o}_2)|>1 \end{cases}$$

$$\text{where} \quad \begin{cases} \mathbf{o}_1 = \{\mathbf{o} < mean(\mathbf{o})\} \\ \mathbf{o}_2 = \{\mathbf{o} > mean(\mathbf{o})\} \end{cases}, \tag{8}$$

and where $mean(\mathbf{o})$ is $\frac{1}{M}\sum_{i=1}^{M} o(i)$. The numerator of the Bivar target function is hence defined as the variance of data-points above average plus the variance of the data-points below the average. This sum is then divided by the square difference between averages of each part. By minimizing the Bivar function we minimize the variances over each peak in the intensity distribution histogram shown in Figure 1(d). This means that our target function tries to accumulate the intensity distribution in two different points (by decreasing the variance at each point) instead of pushing the distribution away from the mean (which had been done by simply maximizing $Var(\mathbf{o})$). Division by the squared distance between the two peaks in the Bivar function will help us to keep the distance between the two peaks as large as



(a) Blurred image acquired by camera (b) Grayscale conversion (c) One-dimensional conversion by averaging or cropping a cross section

(d) Perpendicular replication of one-dimensional barcode (e) After deblurring by our algorithm (f) Fine tuning by converting to exact black and white

Figure 2. Processing performed by our algorithm to acquire the original barcode

possible to avoid merging the peaks. Note that the minimum intensity value is zero, and the maximum is one. Therefore, we should never have a difference of the averages larger than 1. If we still get a larger difference, we know that the kernel is wrong so we set the Bivar value to infinity. This means, that particular kernel will never be selected. The original image in Figure 2(e) was recovered using this target function.

To start the search we need a starting kernel. Assuming we know the length L of the kernel, we can start with a uniform kernel

$$\mathbf{k}_0 = [\underbrace{\frac{1}{L} \quad \frac{1}{L} \quad \dots \quad \frac{1}{L}}_{\text{L times}}]$$

and then we calculate $\lambda'(\mathbf{k}_0)$ and save it as the initial target value. To find the best kernel we will use a variant of *Simulated Annealing* [9], to search for the minimum of the function $\lambda'(\mathbf{k})$: We add a random vector τ of length L with components in the interval $[0, \frac{1}{2^q}]$: $\mathbf{k}_c = \mathbf{k}_0 + \tau$ to create the new candidate vector $\mathbf{k}_c$. Here the variable q will be analogous to (the inverted) temperature of the simulated annealing system; as q increases, smaller and smaller deviations from the current position will be taken. Then $\mathbf{k}_c$ is normalized so that it sums to one: $|\mathbf{k}_c| = 1$. After this, we calculate the $\lambda'(\mathbf{k}_c)$ again, and if the new value of $\lambda'(\mathbf{k}_c)$ is smaller than $\lambda'(\mathbf{k}_0)$, we use this value $\mathbf{k}_1 = \mathbf{k}_c$. Otherwise the old value is kept $\mathbf{k}_1 = \mathbf{k}_0$. We repeat this step a constant number of times. In our system we used 1000 repeats. (For most of the barcodes in our test set, using 100 or even only 10 iterations was enough, but for a few barcodes 1000 iterations were needed. A practical method may therefore be to try decoding after 10, 100 and 1000 iterations.) Next we make our neighborhood interval narrower around the newly found kernel (by increasing q) and reapply the same procedure again. The kernel which minimizes our target function the most among all random kernels that we have tested, is now our estimation of the kernel for that kernel length.

To find the best kernel length, we simply try all different lengths between 1% of the barcode length up to 10% of the length. Blurring kernels smaller than 1% are to small to cause any problems for the subsequent barcode decoding software (at least for the resolutions we tried), so it is not necessary to try smaller sizes. For blurring kernels larger than 10% it is usually not possible to recover the original image. The reason for this is that the number of possible solutions increases quickly with increased kernel size, and hence it is harder to find the correct one.

Some false solutions can be eliminated by considering only the type of motions that are possible in the real world: We assume that the majority of blurring effects will occur as a cause of camera shake. It seems likely that natural shakes made by hand cannot change the velocity of the camera too abruptly. It turned out that most of the motion blurred barcode images that were examined could be deblurred well by a "dome shaped" kernel such as the samples shown in Figure 3. By "dome shaped" we mean that the kernel is first monotonously increasing, up to a maximum point, after which it is monotonously decreasing. By forcing the kernels to be dome shaped, we can improve the speed of the convergence. We enforce this by adding a sorting step right after the vector τ has been added: The maximum value of the kernel is found, all values preceding this value are sorted in increasing order, and all values after the maximum value are sorted in decreasing order. The new, dome shaped kernel is then passed to the normalization step, and the process continues.

4.1.1 Restoring the barcode

In the final step we simply use the estimated kernel in our Wiener deconvolution algorithm along with the blurred image and the appropriate noise-to-signal ratio:

$$\hat{\mathbf{o}} = Deconv(\mathbf{b}, \mathbf{k}, n)$$

The image shown in Figure 2(e) is an example of a deblurred barcode image from the blurred barcode shown in Figure 2(d). Increasing the noise-to-signal parameter in the Wiener deconvolution algorithm will make the resulting deconvoluted images smoother. Since the set of all possible smooth images is smaller than the set of all possible images, this will reduce the search space and make it easier to find an acceptable solution. On the other hand, setting the noise-to-signal ratio too high will prevent our algorithm from finding sharp barcode images that are blurred with large kernels. For our test set a noise-to-signal ratio of either 0.001 or 0.01 was suitable in all cases. Therefore we simply run our algorithm twice, once with $n = 0.01$ and once with $n = 0.001$ and then we see which one is best (i.e., which one can be decoded). In the end we also perform some post processing to reach a more desirable output: We simply set the pixels with lower-than-average intensity to zero (as black), and

the pixels with intensities higher than the average to one (as white). A sample result is shown in Figure 2(f).

5. Result

We tested our system on 138 images of barcodes mostly taken by a 3.2 mega-pixel mobile phone camera (*Sony Ericsson K810i*) under various degrees of motion blur. We fed all the images into a barcode reader software written for mobile devices and 45 images among all samples were decoded successfully without any deblurring. We then fed these rather sharp 45 images through our deblurring system, and they were hardly changed by the system — indicating that the method does not destroy already good data. The remaining 93 images that could not be decoded by the existing barcode reader were then fed into our deblurring algorithm. After deblurring, 44 of the 93 images could now be correctly decoded. For the 45+44=89 images that were correctly decoded, we could assume that the convolution kernel must have been (at least roughly) correctly estimated. For these cases where the kernel was recovered correctly, it was interesting to study the kernel length. Typically, if the kernel length was smaller than half of the width of the narrowest bar, the barcode could sometimes be decoded even without deblurring. However, with deblurring, it was possible to decode images with blurring kernels of up to four times the width of the narrowest bar. In Figure 3 you can see some of the sample results.

The computation time on a standard PC running at 2.66 GHz, for a barcode image width of 1000 pixels, is approximately 8 seconds with 1000 iterations and 2 seconds with 10 iterations. No attempts to optimize the code for speed were made.

6. Further discussion

In the process of constructing our target function we tried some approaches based on image gradients and also some others in the frequency domain. These approaches were based on assumptions that bar code images after deblurring would contain very sharp edges, i.e., gradients of large magnitude or high frequency magnitudes in the Fourier domain. None of these alternative approaches turned out to work as well as our Bivar criterion.

The target function used basically measures the bimodality of the intensity distribution. If there is an intensity gradient going from the left in the image to the right, this will smear out bimodality, even for a non-blurred image. To mitigate this situation, it might be possible to compensate for the global lighting before using the proposed method or performing partial deblurring, but we have not tried this. In general, our deblurring may fail in cases such as having heterogenous blur as a result of non-uniform kernel, varying illumination, high noise level and when other blurring

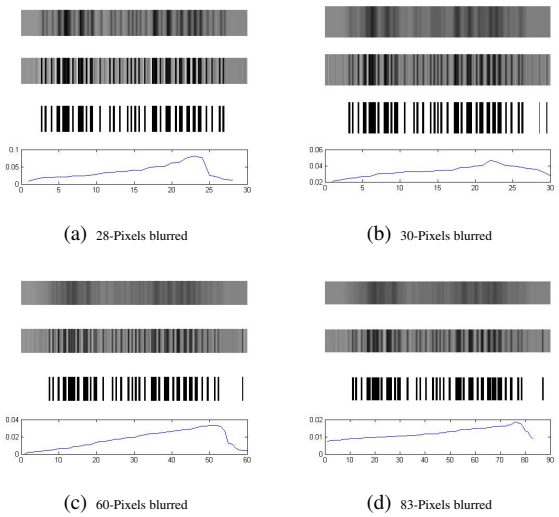


Figure 3. Deblurring samples made by our algorithm with different motion kernels. In each figure first row is the (blurred) input, the second row shows the deblurred image by our deblurring algorithm, the third row shows the fine-tuned barcode image and the last row depicts the approximated kernel found by our algorithm which is used for deblurring. Note that all kernels in this example are "dome shaped", i.e., they are first monotonously increasing up to their maximum point, after which they are monotonously decreasing. (Sample images have different length in the range of 500 to 1200 pixels)

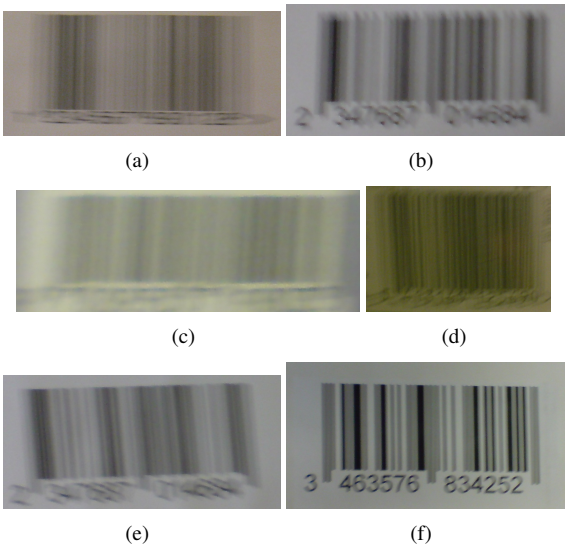
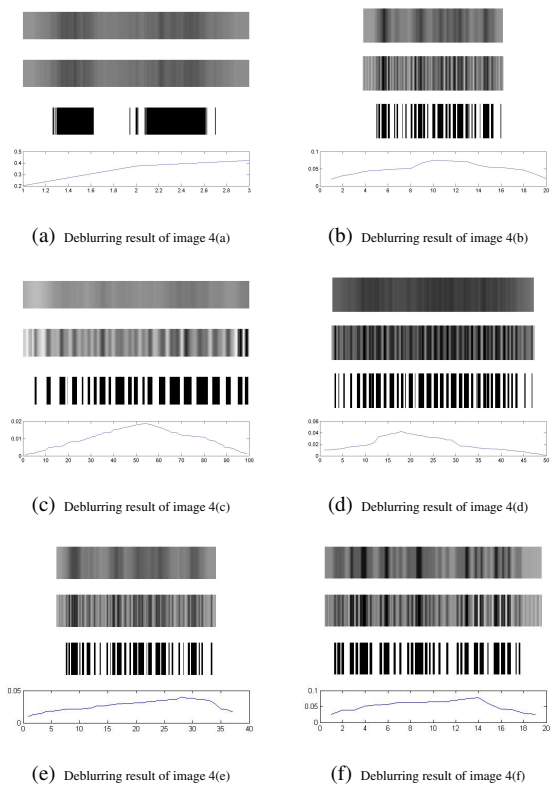


Figure 4. Some blurred barcode images for which our deblurring did not work well.

artifacts such as out-of-focus are dominant. Figure 4 shows such blurred barcode images when our deblurring algorithm did not work so well. Figure 5 shows the result of our algorithm over the images in Figure 4. As you can see, some-



(a) Deblurring result of image 4(a) (b) Deblurring result of image 4(b)

(c) Deblurring result of image 4(c) (d) Deblurring result of image 4(d)

(e) Deblurring result of image 4(e) (f) Deblurring result of image 4(f)

Figure 5. Deblurring results over the corresponding images in Figure 4. In these cases the decoding was unsuccessful.

times barcodes are not decodable, although the output result from our algorithm shows a significant improvement in deblurring. Very small changes in barcodes may completely ruin their uniqueness and consequently they can not be decoded.

7. Conclusion

We have presented a heuristic algorithm for deblurring barcode images captured by conventional digital cameras. We use the bimodal characteristics of barcode intensity histograms to construct a target function which can quantify the similarity of an image to a real barcode. By exploiting an existing non-blind deconvolution algorithm we have constructed a blind deconvolution algorithm by guessing the kernel which makes our deconvoluted image most barcode-like. Our algorithm will reach its best performance when the translatory motion blurring which can be modeled as a convolution is dominant over other image degradation artifacts.

References

[1] T. Bishop, D. Barbacan, B. Amizic, T. Chan, R. Molina, and A. K. Katsaggelos. Blind image deconvolution: problem formulation and existing approaches. In P. Campisi and K. Egiazarian, editors, *Blind image deconvolution: Theory and Applications*, chapter 1, page 2. CRC press, 2007.

[2] S. Cho and S. Lee. Fast motion deblurring. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM.

[3] S. Esedoglu. Blind deconvolution of bar code signals. *Inverse Problems*, (20):121–135, November 2003.

[4] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 787–794, New York, NY, USA, 2006. ACM.

[5] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Pub (Sd), 1992.

[6] J. Jia. Single image motion deblurring using transparency. In *CVPR*, 2007.

[7] N. Joshi, C. L. Zitnick, R. Szeliski, and D. J. Kriegman. Image deblurring and denoising using color priors. In *CVPR*, Miami, Florida, 2009.

[8] J. Kim and H. Lee. Joint nonuniform illumination estimation and deblurring for bar code signals. *Optics Express*, 15:14817–+, 2007.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[10] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *CVPR*, pages 1964–1971. IEEE, 2009.

[11] L. Lucy. An iterative technique for the rectification of observed distributions. *Astronomy*, 79:745–754, 1974.

[12] P. Magain, F. Courbin, and S. Sohy. Deconvolution with correct sampling, 1997.

[13] J. G. Proakis and D. G. Manolakis. *Digital signal processing : principles, algorithms, and applications*. Macmillan Coll Div, 2 edition, 1992.

[14] S. Riad. The deconvolution problem: An overview. *Proceedings of the IEEE*, 74:82–85, 1986.

[15] H. W. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, January 1972.

[16] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.

[17] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.

[18] T. Wittman. Deblurring and restoration in barcode signal processing. *SIAM Conference on Imaging Science*, 2004.

[19] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum. Blurred/non-blurred image alignment using sparseness prior. *Computer Vision, IEEE International Conference on*, pages 1–8, 2007.

[20] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum. Progressive inter-scale and intra-scale non-blind image deconvolution. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.