IAC-06-B1.P.1.11

COMPARISON OF MOTION DEBLUR ALGORITHMS AND REAL WORLD DEPLOYMENT

Sebastian Schuon <schuon@mytum.de>, Klaus Diepold <kldi@tum.de>
Institute for Data Processing, Technische Universität München
Munich, Germany

Abstract

If a camera moves fast while taking a picture, motion blur is induced. There exist techniques to prevent this effect to occur, such as moving the lens system or the CCD chip electro- mechanically. Another approach is to remove the motion blur after the images have been taken, using signal processing algorithms as post-processing techniques. For more than 30 years, numerous researchers have developed theories and algorithms for this purpose, which work quite well when applied to artificially blurred images. If one attempts to use those techniques to real world scenarios, they mostly fail miserably. In order to study why the known algorithms have problems to de-blur naturally blurred images we have built an experimental setup, which produces real blurred images with defined parameters in a controlled environment. For this article we have studied the most important algorithms used for de-blurring, we have analyzed their properties when applied to artificially blurred images and to real images. We propose solutions to make the algorithms fit for purpose.

1 Purpose

Often security cameras are mounted on posts to overview a parking lot or similar venues. If the wind blows those posts may start to shake and the attached cameras often produce blurry images. Due to this motion blur details in the captured images, such as faces or license plates are unrecognizable or unintelligible. A similar problem is addressed in [22], where prerecorded images of a truck, moving at a speed of 60 mph, are de-blurred in order to recognize the truck's side printing.

In [16] another version of the same problem is described, but in a total different setting. The Hubble Space Telescope (HST) has two different capture modes: fine lock and gyro-hold. At times observations are done in gyro-hold mode, which does not offer any position control to keep the HST at a fixed location during capture. This mode is prone to motion during the capturing process, such that significant motion blur finds its way into the final image. To get images, which are as sharp as possible, advanced motion de-blur algorithms have been applied to the images, achieving good results.

Apart from these examples motion blur is a phenomena common to all photographers. There are either moving objects in the scene or the camera is moving during the capture progress, both situations leading to blurred images. In order to address this issue various techniques in hardware [4, 17] and software [1, 2, 6, 5, 18] have already been developed. All of them have technical constrains of some kind such as requiring additional hardware or producing sub-optimal results.

The corresponding image restoration process, referred to as motion de-blur, can be broken up in two parts: motion estimating and deconvolution. The first part deals with the challenge to identify the path the camera has followed during the image capture process. The second part uses this information to reverse the convolution during the image formation process in order to restore the original picture. Lately much effort has been put into the first part and some remarkable results have been published [1, 2, 3, 19, 18]. All the previously proposed de-blurring methods rely on a small number of algorithms to perform the second part, the deconvolution. Therefore we perform a quantitative comparison of the algorithms already in use and some which have been recently proposed in the literature. We point out promising candidates and give clues for further improvement. Notably deblur algorithms are demonstrated using synthetically blurred images, which provide quite different characteristics from motion blur encountered in the real world. We have build an experimental setup allowing us to generate real motion blur with predefined parameters allowing us to measure and compare the performance of algorithms under real world conditions.

2 Modelling Motion Blur

We use a linear, non-recursive (FIR) model to represent the degradation of digital (sampled) images caused by motion blur. We consider the original, blur-free $M \times N$-image $\boldsymbol{f}$ to be convolved with a convolution kernel $\boldsymbol{h}$, referred to as the Point Spread Function (PSF). Additionally, some noise is introduced during the capturing process, which is modeled with the additive noise term $\boldsymbol{n}$. Hence, the blurred $M \times N$ image $\boldsymbol{b}$, as it is captured by the moving camera, is modeled as

$$\boldsymbol{b} = \boldsymbol{h} \star \boldsymbol{f} + \boldsymbol{n}, \tag{1}$$

where the symbol $\star$ represents the convolution operator.

De-blurring images accounts to the application of the de-blurring operator D, which produces a deblurred image $\hat{\boldsymbol{f}}$ when applied to the blurred image $\boldsymbol{b}$, that is $D(\boldsymbol{b}) = \hat{\boldsymbol{f}}$.

2.1 Synthetic Motion Blur

For test purposes we create images which are synthetically blurred according to the blur model. This yields the advantage that we have access to the reference image $\boldsymbol{f}$, which is not known in real environments, for comparison against the restored image, which we will denote by $\hat{\boldsymbol{r}}$.

Motion blur is described by means of a Point Spread Function (PSF), which provides information of the underlying motion during the capture process. In the most simple case, that is, for a uniform linear motion along the x-axis with a speed of k pixels during the capturing period the PSF is given by a one-dimensional vector of the length $k+1$:

$$\boldsymbol{h_{lin}} = \frac{1}{k+1}\begin{bmatrix}1 & 1 & 1 & ... & 1\end{bmatrix}. \tag{2}$$

However, in a real environment with shaking cameras, neither the path of the moving camera nor the point spread function are known a priori and need to be estimated from the measured data.

In [2], Ben-Ezra and Nayar have proposed a method to determine the motion paths during the capturing process. Their analysis shows that the model for the PSF has to be extended to represent motion in a two-dimensional plane. The PSF is a matrix $\boldsymbol{h}$ of size $U \times V$, where each entry $h_{i,j}$, $i = 1, 2, \dots, U, j = 1, 2, \dots V$ represents the percentage the camera has been displaced by $i - \frac{U}{2}$, $j - \frac{V}{2}$ from the center during the capture.

$$\boldsymbol{h} = \frac{1}{K}\begin{bmatrix} h_{1,1} & h_{2,1} & \cdots & h_{1,V} \\ h_{2,1} & \ddots & & \\ \vdots & & \ddots & \\ h_{U,1} & & & h_{U,V} \end{bmatrix}, \tag{3}$$

where the parameter K is a normalizing constant to achieve that the sum over the entries of h equals 1.

$$K = \sum_{i=1}^{U}\sum_{j=1}^{V} h(i,j) \tag{4}$$

An example for such a PSF, which represents a triangular motion path is given in Eq. 5. Figure 1 shows the graphical representation of this PSF and its impact on a checkerboard structure test picture.

$$\boldsymbol{h_{comp}} = \frac{1}{20}\begin{bmatrix} 1 & & & & & & & 2 \\ & 1 & & & & & 2 & \\ & & 2 & & & 3 & & \\ & & & 4 & 5 & & & \end{bmatrix} \tag{5}$$

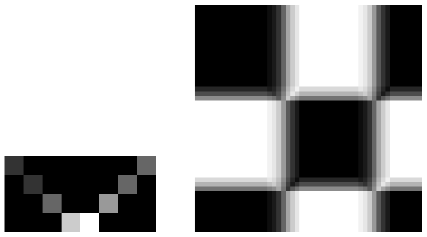

Figure 1:
Complex PSF (left) and resulting image (right)

2.2 Border Area

When generating synthetic blur as described above problems occur in the border region where no information of pixel values beyond the border of the image is available, which is necessary to compute the convolution properly. Several approaches exists to solve this issue:

- *circular*: the image is considered to be periodic, values are therefore taken from the opposite border
- *repetetive*: the very last pixel next to the boarder is repeated
- *mirroring*: the image is mirrored at the border, therefore providing values for the region beyond the boarder
- *constant*: the values beyond the boarder are considered to be constants (often black or white)

Another approach is to perform clipping around the boarder area, therefore reducing the size of the output image but computing more realistic results. The boarder issue has been addressed further in [23].

2.3 Real Motion Blur

In the literature a number of examples are shown where synthetically motion-blurred images are deblurred using various algorithms and where the results look quite promising. However, when applying deblurring algorithms to real-world pictures, which contain motion blur, it is revealed that those algorithms perform quite unsatisfactory. This may be due to poorly estimated PSFs or due to failures

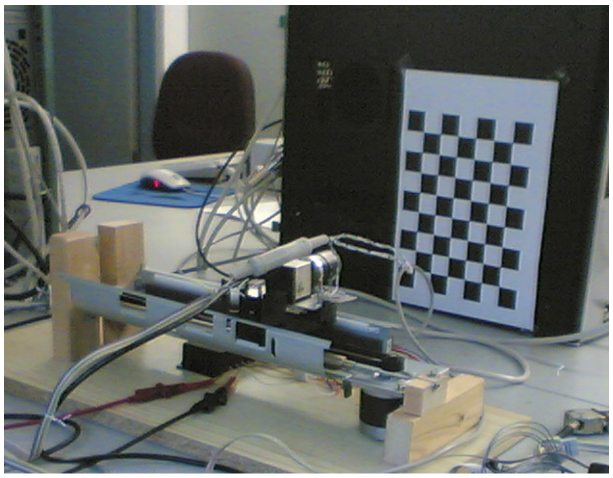

Figure 2: Experimental setup

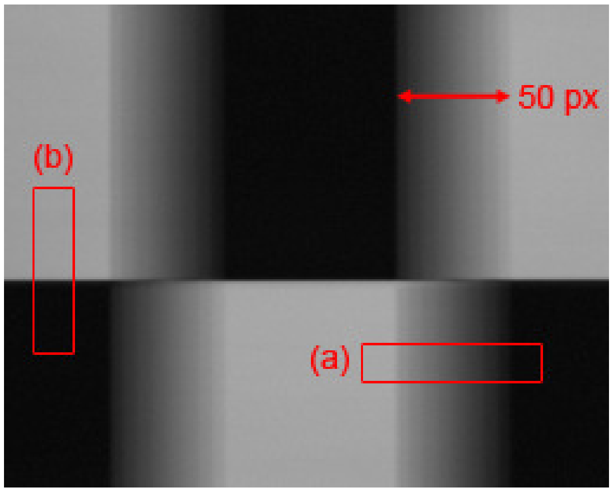

Figure 3: Estimating the PSF with real motion blur

in the underlying motion blur model itself. To investigate this issue in more detail and to locate the root of the problem, an experimental setup has been built, which allows us to capture blurred images in a controlled setting which facilitates to have access to well-defined and parameterized PSFs.

The unit shown in Figure 2 comprises a camera unit, a guiding rail and a stepper motor. The camera carriage is accelerated to a constant speed and takes a photo with a medium exposure time (around 100ms) to allow significant motion blur appear in the picture. As a motif a checkerboard structure has been chosen, since this allows an easy method for estimating the associated PSF.

An image captured with our experimental setup is shown in Figure 3. From the way we have set up the capturing process we assume the motion path

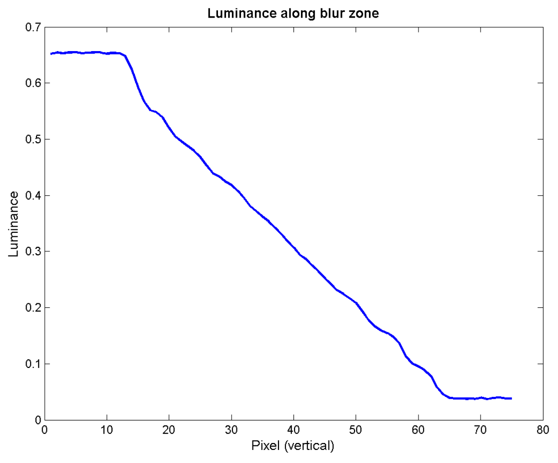

Figure 4: Verification of linearity

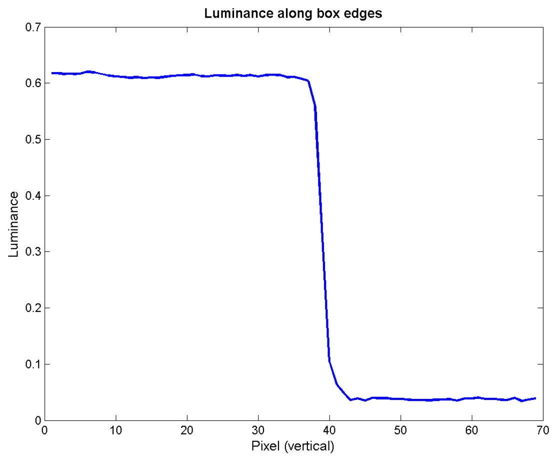

Figure 5: Verification of horizontal motion

of the camera to be linear and uniform in the horizontal direction. The first assumption is verified by looking at the plot shown in Figure 4. The plot depicts the luminance of the picture, which taken across the blurred zone (marked with (a) in Figure 3). The luminance curve is very close to linearly decreasing.

The second assumption that the motion is only in the horizontal direction is verified with a second luminance graph shown in region (b) of Figure 5. The sharp decay of luminance at the border between the two boxes proves that there is almost no motion in the vertical direction (otherwise the graph must look like shown in Figure 4, where the decay is linear and spread about a significant number of pixels).

Measuring the blur area in Figure 4 allows us to directly infere the length of the PSF (the length has been visualized in Figure 3). With the two assumptions verified above and the length given, the PSF can now be determined according to Eq. 2 as

$$\boldsymbol{h_{real}} = \frac{1}{50}\begin{bmatrix}1 & 1 & 1 & \dots & 1\end{bmatrix} \in \mathbb{R}^{1\times 50} \tag{6}$$

3 Description of Algorithms

3.1 Direct Approach (*lin*)

Starting from Eq. 1 we transform the equation into the frequency domain, which yields

$$\mathcal{F}(\boldsymbol{b}) = \mathcal{F}(\boldsymbol{h}) \cdot \mathcal{F}(\boldsymbol{f}) + \mathcal{F}(\boldsymbol{n}) \tag{7}$$

As the additive noise is unknown, we assume it to be zero ($\mathcal{F}(\boldsymbol{n}) = 0$). Rewriting Eq. 7 and performing the re-transformation into the spatial domain, we arrive at the restoration filter

$$\hat{\boldsymbol{f}} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\boldsymbol{b})}{\mathcal{F}(\boldsymbol{h})}\right) \tag{8}$$

This gives us a direct filter requiring only out-of-the-box mathematical methods. As zero noise has been assumed it is expected that problems will occur with noisy images.

3.2 Wiener-Filter (*wnr*)

The Wiener Filter seeks to minimize the following error function:

$$e^2 = E\left[\left(f - \hat{f}\right)^2\right] \tag{9}$$

where E denotes the expected value operator, f is the undegraded image and $\hat{r}$ its estimate. The solution to the thereof arising optimization task can be written as follows in the frequency domain (according to [8]):

$$\hat{F} = \left[\frac{1}{H(u,v)}\,\frac{|H(u,v)|^2}{|H(u,v,)|^2 + S_\eta(u,v)/S_\iota(u,v)}\right]\cdot B \tag{10}$$

with $H(u,v)$being the PSF in the frequency domain, $S_\eta(u,v)$ the power spectrum of the noise and

$S_\iota(u,v)$ the power spectrum of the undegraded image F. The ratio $NSR = S_\eta(u,v)/S_\iota(u,v)$ is normally referred to as the Noise to Signal Ratio. If no noise is present ($S_\eta(u,v) = 0$) Eq. (10) reduces to

$$\hat{F}(u,v) = \frac{B(u,v)}{H(u,v)} \tag{11}$$

Therefore we see the Wiener Filter is a generalization of the direct filter.

If the ratio NSR is unknown, it can be approximated with the ratio r of average noise power and average image power (parametric Wiener filter):

$$NSR \approx r = \frac{\eta_{average}}{\iota_{average}} \tag{12}$$

Even better results can be achieved using the autocorrelation function of the noise and the undegraded image [8]. A derivation of the formulas mentioned can be found in [7].

3.3 Regularized Filer (*reg*)

This algorithm [8] is based on finding a direct filter solution using a criterion C, which ensures optimal smoothness of the image restored. Therefore the filter construction task is to find the minimum of

$$C = \sum_{u=1}^{M}\sum_{v=1}^{N}[\nabla^2 f(u,v)]^2 \tag{13}$$

under the constraint of the rewritten Eq. 1

$$\left\|\boldsymbol{b} - \boldsymbol{h} \star \hat{\boldsymbol{f}}\right\|^2 = \|\boldsymbol{n}\|^2 \tag{14}$$

In the frequency domain the solution to this problem can be written as follows:

$$\hat{F}(u,v) = [\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma\,|P(u,v)|^2}] \cdot B(u,v) \tag{15}$$

where γ is the parameter which has to be adjusted to fulfill the constraint C and $P(u,v)$ is the Laplacian operator in the frequency domain.

3.4 Richardson-Lucy Deconvolution (*lucy*)

This algorithm was invented independently by Richardson [20] and Lucy [13]. Its usage (especially concerning MATLAB) is further outlined in [8]. The Richardson-Lucy (RL) algorithm is an iterative restoration algorithm that maximizes a Poisson statistics image model likelihood function. As summed up in [11] the RL algorithms consists out of one initial and three iterative steps:

1. A first approximation of the restored image $\hat{\boldsymbol{f}}_{\boldsymbol{0}}$ must be made, typically the constant average of all pixel values in the blurred image $\boldsymbol{b}$.
2. The current approximation is convolved with the PSF
$$\boldsymbol{\varphi_n} = \boldsymbol{h} \star \hat{\boldsymbol{f}}_{\boldsymbol{n}} \tag{16}$$
3. A correction factor is computed based on the ratio of the blurred image and the result of the last step
$$\boldsymbol{\phi_n} = \overleftarrow{\boldsymbol{h}} \star \frac{\boldsymbol{b}}{\boldsymbol{\varphi_n}} \tag{17}$$
where $\overleftarrow{\boldsymbol{h}}$ denotes the PSF in reverse order and $\frac{\boldsymbol{b}}{\boldsymbol{\varphi_n}}$ a "pixel-by-pixel" division.
4. A new approximate is composed out of the current one and the correction factor
$$\hat{\boldsymbol{f}}_{\boldsymbol{n+1}} = \hat{\boldsymbol{f}}_{\boldsymbol{n}} \cdot \boldsymbol{\phi_n} \tag{18}$$
where $\cdot$ denotes a "pixel-by-pixel" multiplication. The algorithm continues with step 2.

As with all iterative techniques the question arises when to stop the computation, but this will be addressed later on.

3.5 Maximum Likelihood Estimation (*mem*)

A complete description of this algorithm would beyond the scope of this paper, but good descriptions can be found in [10, 9].

In brief, the algorithm has the ability to alter the PSF used for deconvolution according to some constraints to an improved solution. The deconvolution itself is performed in a comparable fashion to the Richardson-Lucy algorithm.

3.6 TU Berlin (*tub*)

Mery and Filbert proposed an algorithm in [15] which seeks to minimize the equation

$$\left\|\tilde{f} - \tilde{b}\right\| \tag{19}$$

under the constraint of Eq. 1. $\tilde{f}$ is a vector of the first N pixels of a line of the restored image and $\tilde{b}$ respectively a vector of N pixels of a line of the blurred image. Using only N pixels instead of the whole vector allows the algorithm to be fast compared to other techniques.

The optimization problem is solved using Lagrange multipliers, resulting in a direct restoration algorithm. It should be noted that the algorithm in its current form can handle only uniform motion blur.

3.7 Sondhi (*sondhi*)

Sondhi [21, 7] addressed the problem of motion deblurring very early. He assumes the blur process to integrate over certain amount a of pixels during the capture process. A blurred image line (length L) can therefore be written as

$$b(x) = \int_{x=0}^{a} f(\tau) d\tau \tag{20}$$

Calculating the derivate on this equation, defining $\phi(x) = f(x-a)$, $K = \lfloor \frac{L}{a} \rfloor$ and rewriting it we arrive at

$$\hat{f}(x) = \sum_{k=0}^{\lfloor \frac{x}{a} \rfloor} b'(x-ka) + \phi(x - \left\lfloor \frac{x}{a} \right\rfloor a) \tag{21}$$

Taking some assumptions about ϕ into account and defining $\tilde{b}(x) = \sum_{0}^{\lfloor \frac{x}{a} \rfloor} b'(x-ka)$ the restored image line is given by

$$\hat{f}(x) = \tilde{b}(x) - \frac{1}{K} \sum_{0}^{K-1} \tilde{b}(x+ka) + \bar{b} \tag{22}$$

where $\bar{b}$ is the average value of a pixel line.

It should be mentioned that the version mentioned here does only apply to linear, uniform motion blur.

3.8 Advanced Landweber (*alm*)

This algorithm [12] is mentioned here for completeness, but it has been excluded from the comparison as our implementation was ten times slower than the rest of the field. Furthermore, it was originally proposed for removal of blur induced due to defocusing.

4 Comparison

4.1 Data Material

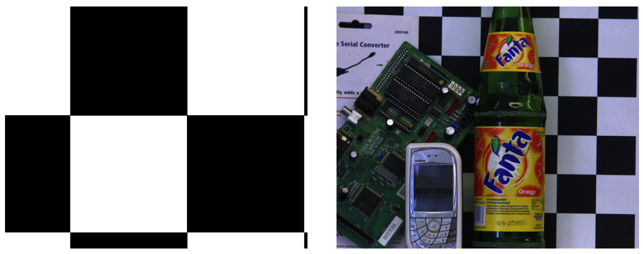

Figure 6: Test images reference
(left: checkerboard, right: natural)

Two different motifs have been selected for the algorithms to work on (Figure 6). The checkerboard structure (Figure 7) as the first test pattern represents a more synthetic motif, but the motion blur is clearly visible. Due to the regular and well known pattern the results of the algorithms can easily be analyzed. Furthermore, the sharp boarders of the squares allow estimation of the parameters of the PSF (see section 2.3).

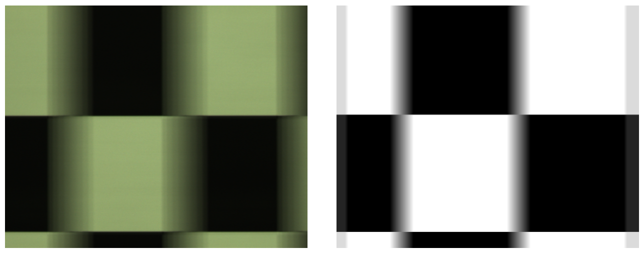

Figure 7: Test images Checkerboard
(left: real, right: synthetic)

The second motif (Figure 8), although still having the checkerboard structure as a background for reference and PSF estimation purposes, comprises items which are not only 2D (i.e. flat) but 3D (e.g. bottle) or have characters on them, which are normally hard to restore.

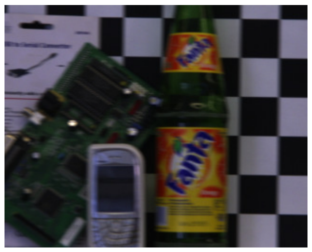

Figure 8: Test Image Natural Scene (real blur)

4.2 Computation Speed

One criteria for the evaluation of the de-blurring algorithms is the computation speed required to restore an image. Computation power may be limited in some environments, for example in mobile applications, or the sheer execution time is far beyond the expectation of users. For example the Richard-Lucy algorithm needs more than one hour for restoring a consumer class digital camera image on a modern computer hardware.

The time required to perform the deconvolution depends on a number of parameters. The most obvious one is the size of the input image. Figure 9 shows the computation time as a function of the image size. The time scale has been normalized to the result of the fastest algorithm *wnr* at the smallest pixel value computed (2500 pixel). It is clearly visible that the computation time of most algorithms rises linearly with increasing image size. A big difference shows up in the total computation time required. The time required by iterative algorithms is much greater, but can be influenced by the number of iterations (discussed further below), whereas the direct algorithms perform much faster. The difference among them is possible due to the implementation. As for the *wnr* and *reg* existing, optimized MATLAB implementation have been used whereas *tub*, *lin* and *sondhi* were implemented with out special optimization. Interestingly the *tub* algorithm does perform sligthly above linear.

The second factor determining the algorithms performance in terms of computation speed is the length respectively size of the PSF. This dependency is outlined in Figure 10. As before the computation is normalized to the calculation performed with *wnr*. It can be seen that most algorithms are invariant towards different sizes of the PSF with the exception of *sondhi* which seems to deliver low performance with small sizes of PSFs.

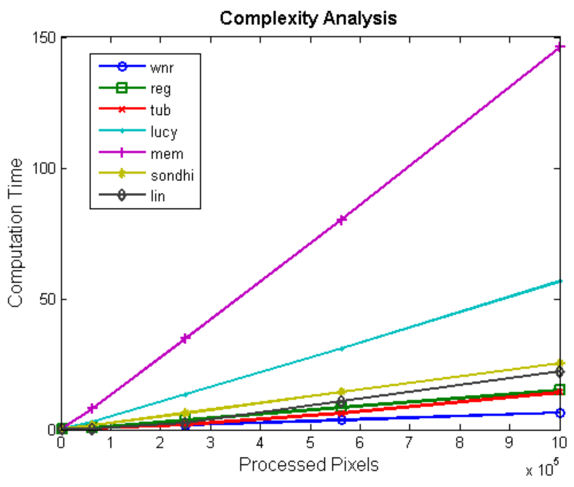

Figure 9: Computation Speed vs. Image Size

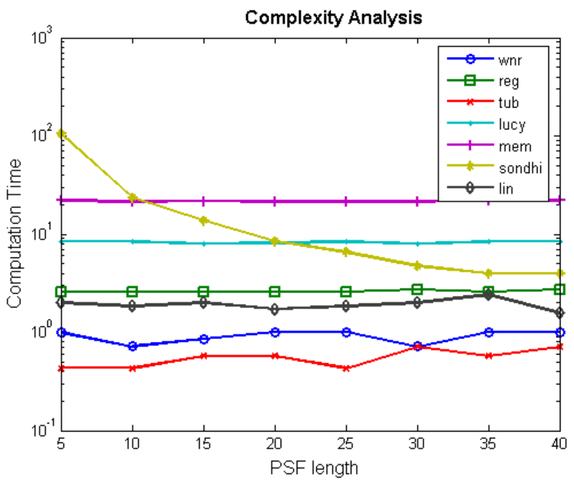

Figure 10: Computation Speed vs. PSF length

With iterative algorithms the number of iterations has a strong influence on the computation time and the restoration performance. It can safely be assumed that the computation time depends linearly on the number of iterations. Therefore the dependency of the PSNR is shown (Figure 11) at selected iteration numbers. It can clearly be seen that with the *lucy* algorithm an increase of iterations leads to an improved PSNR for the restoration result. But at a certain point the incremental improvement is negligible compared to the computation time required (in this case this is at about 20 to 30 iterations). For the *mem* algorithm, the PSNR does not improve any more beyond a certain number of iterations. This is possibly due to the algorithm reaching its optimum for the newly estimated PSF

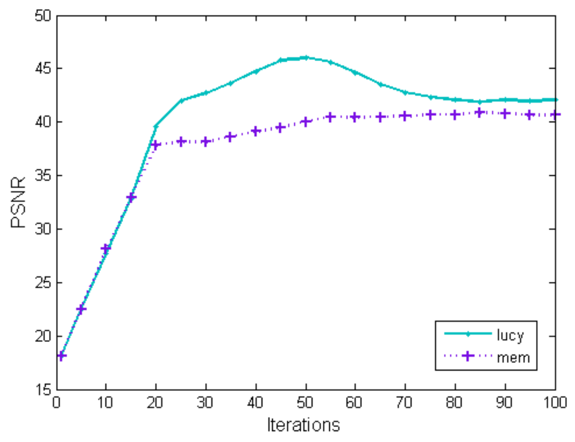

Figure 11: Number of iterations affecting PSNR

(the *mem* algorithm does try to estimate the PSF even more precisely). Computing more iterations than necessary can even have a negative effect (*lucy* beyond 50 pixels).

4.3 Restoration Quality

In order to evaluate the quality (similarity to the unavailable, perfectly captured image) of the restored image two different techniques have been applied. For pictures, where no reference image was available (real blurred images) a "no-reference perceptual blur metric" as proposed in [14] has been tested. Unfortunately it failed, as the lines introduced due to the ringing confused the algorithm. Therefore the results of this metric have been omitted. Furthermore the PSNR, defined as

$$20 \cdot log_{10} \left(\frac{1}{\sqrt{\frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} (\hat{f}(i,j) - f(i,j))^2}} \right) \tag{23}$$

between the synthetically restored image $\hat{f}$ and the reference image f is computed.

4.3.1 Synthetic Blur

In the first test case the algorithms were to restore synthetically blurred image. As the border areas are problematic for all algorithms (as discussed in section 2.2), the image was assumed to be periodic when adding the synthetic motion blur ("circular").

The restored images can be seen in Figure 13, where all algorithms achieve nearly perfect restoration results and the images contain only minor ringing in some cases.

Algorithm	PSNR [dB]
wnr	30.6
reg	30.6
tub	31.7
lucy	42.4
mem	39.0
sondhi	22.2
lin	35.4

Table 1:
Metric results for synthetic blur
(circular wrap around)

In the second run de-blurring of synthetically blurred images has been tested with pixel repetition at the border (Figure 12). It can be seen that obviously wrong pixels in the border region lead to massive ringing in the restored picture. Only the *tub* and the *sondhi* algorithm seem to be able to handle this issue adequately and present quite acceptable results.

Algorithm	PSNR [dB]
wnr	13.1
reg	13.1
tub	34.2
lucy	22.3
mem	13.5
sondhi	21.8
lin	8.8

Table 2:
Metric results for synthetic blur
(repetitive wrap around)

MATLAB offers a function *edgetaper* which is recommended to be applied to images which show a lot of ringing [8]. The *edgetaper* function blurs the ends of the image with the PSF later used for deconvolution. To evaluate this function, it has been applied to the images with repetitive pixel wrap around at the border region as these images proved to be challenging for the algorithms. Figure 14 presents the result of this case, where it can be seen that *edgetaper* does help to decrease ringing, but is unable to suppress it completely or assure equivalent results to the circular blur situation.

Algorithm	PSNR [dB]
wnr	24.6
reg	24.6
tub	34.9
lucy	33.6
mem	19.3
sondhi	22.1
lin	14.6

Table 3:
Metric results for synthetic blur with prior edgetaper (repetitive wrap around)

All the test cases presented before did not comprise noise. Therefore, they are just theoretical cases to study some effects of motion de-blurring. In order to get a better understanding of the schemes under real-world conditions, zero-mean Gaussian noise with variance 0.01 is added to the test image before restoration. The restored images for this case can be seen in Figure 15.

The *wnr* result still looks blurry, whereas the *reg* restoration was able to restore the squares, but still comprises noise and ringing. The *tub* algorithm delivers the sharpest image without any ringing, but some noise visible. Both, *lucy* and *mem* restore the image without any noise visible, but the edges are still a little bit blurred. The noise in the restoration results computed by *tub* and *sondhi* show a very similar behavior, but the later one induces some ringing and some blur at the edges of the boxes. The *lin* filter is not able to remove the noise in the image and introduces a lot of ringing.

Algorithm	PSNR [dB]
wnr	19.8
reg	21.0
tub	21.9
lucy	23.5
mem	23.3
sondhi	19.3
lin	13.4

Table 4:
Metric results for synthetic blur with Gaussian noise (repetitive wrap around, prior edgetaper)

For the *lucy* and *mem* algorithms a modified implementation has been used, which allows to specify a weight for certain pixels corresponding to the reliability of the associated values. On account of this, the border pixels have received a much lower weight, therefore suppressing the ringing induced by incorrect/missing pixel information in the border area. Pixels on all four borders have reduced weight, which explains the black borders.

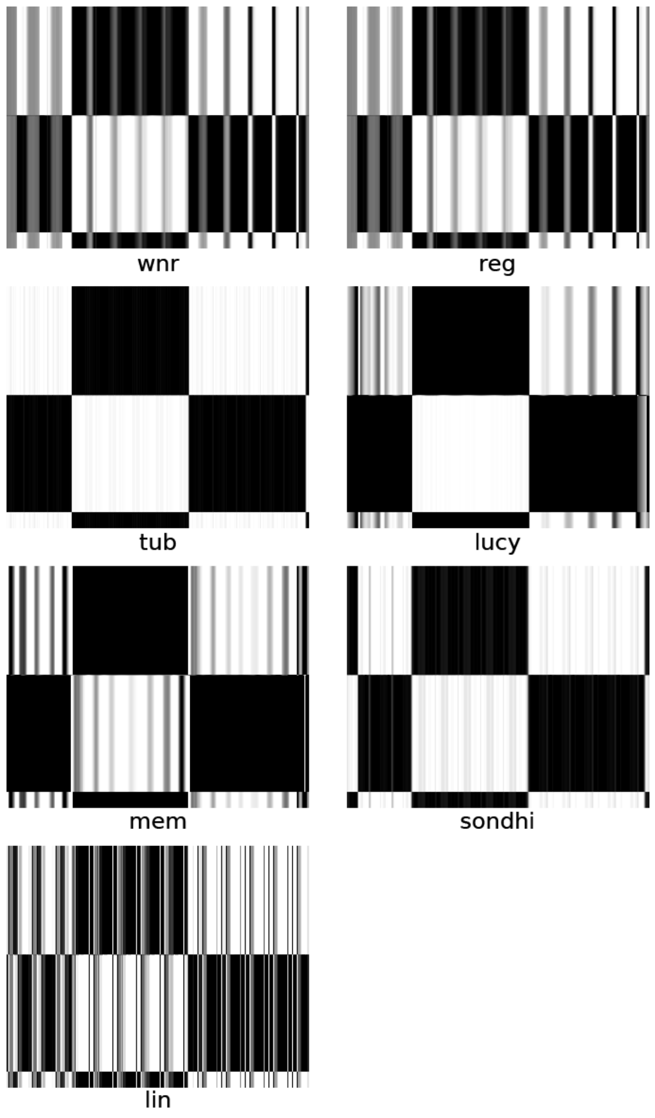

Figure 12:
Restored synthetic blur images
(repetitive wrap around)

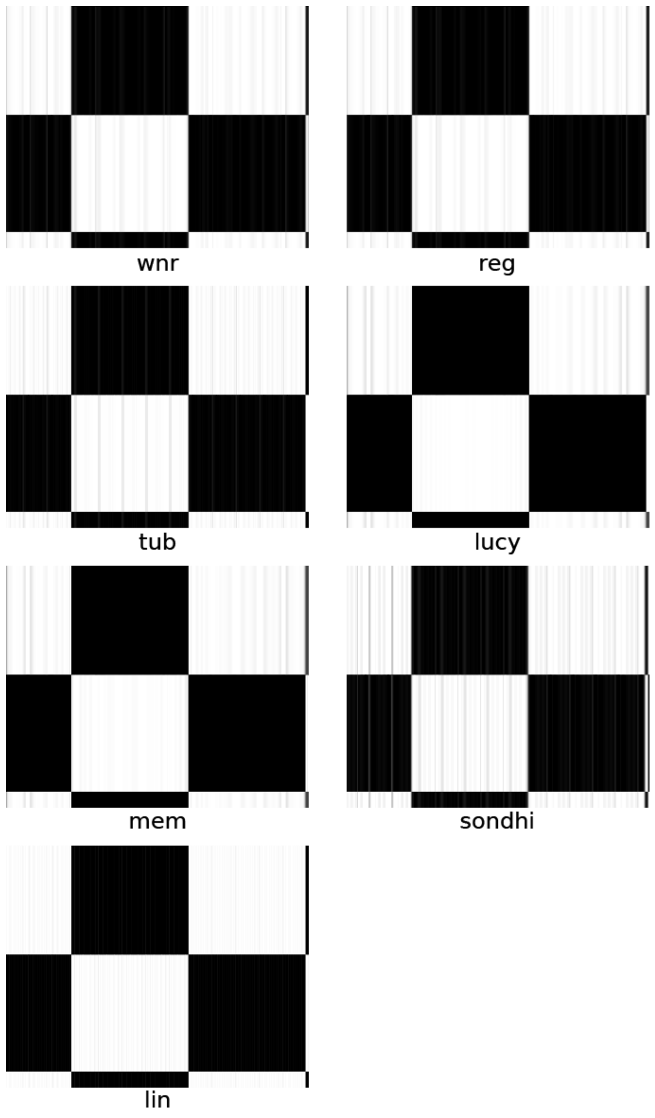

Figure 13:
Restored synthetic blur images
(circular wrap around)

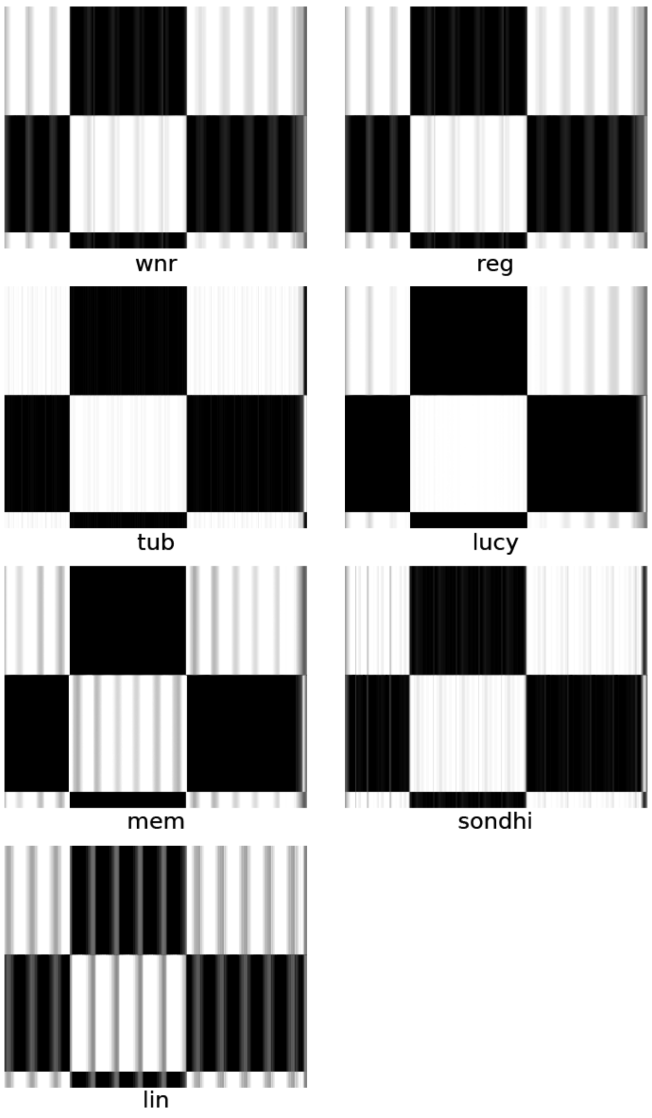

Figure 14:
Restored synthetic blur images with prior edgetaper (repetitive wrap around)

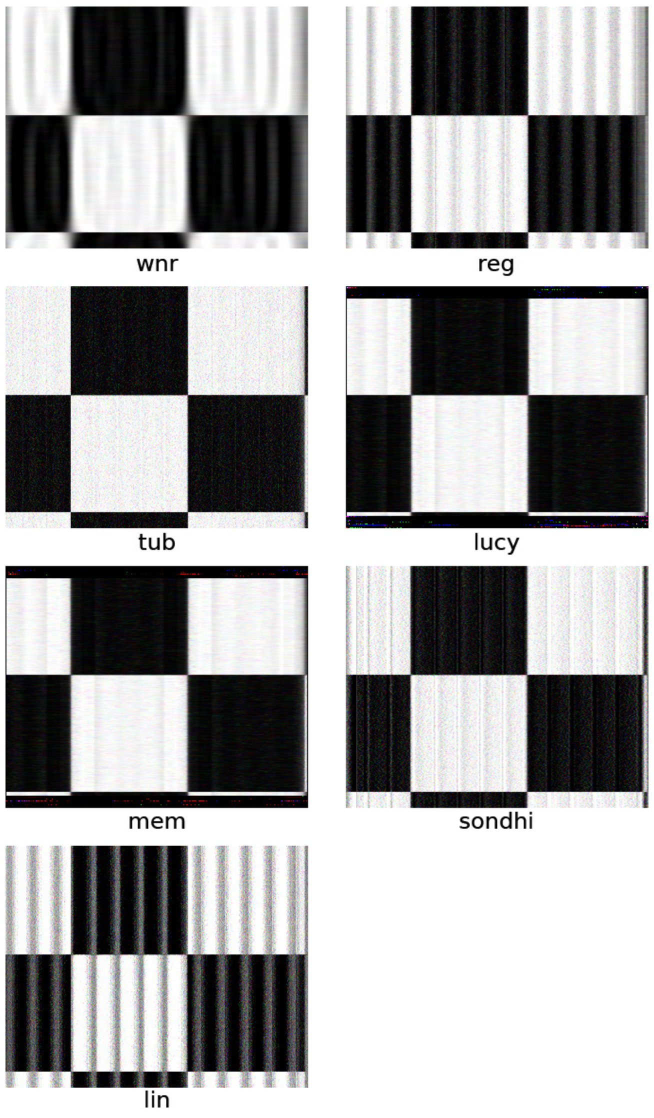

Figure 15:
Restored synthetic blur images with Gaussian noise (repetitive wrap around, prior edgetaper)

4.3.2 Real Blur

Figure 16 shows the restored images of the checkerboard motif, which has been captured by a real camera. The most simple algorithm *lin* fails completely, because the noise is amplified. The *wnr* and *reg* algorithm show both comparable results, which comprise a lot of low frequency ringing, but restore the contours more or less satisfactory. The problem with both algorithms is that the noise power can only be guessed (or determined by trial and error as it was here the case). The *tub* algorithm produces a lot of noise in the restored image, but the contours of the squares are the sharpest among the competitors. Furthermore, high frequency ringing is clearly visible in the restored image. A similar result, but with much less noise is produced by the *sondhi* algorithm. The *lucy* and *mem* algorithm produce more or less equivalent results, which are very close to the original checkerboard motif. Again, the pixels at the borders have received lower weight to inhibit ringing.

In a scene which comprises more than just a checkerboard structure the algorithms seem to have huge problems concerning ringing (Figure 17). The first challenge in restoration is noise, which is mastered properly only by *lucy* and *mem*. The algorithms *wnr* and *reg* still show acceptable results. The restored images of the *tub*, *sondhi* and *lin* algorithms are very noisy such that it is difficult to recognize anything. The remaining candidates have all difficulties with ringing, but only *lucy* and *mem* deliver acceptable results as their ringing artifacts are more smooth and therefore more pleasant for the human eye. The restored image of the *mem* algorithm is slightly better than the result produced by *lucy*, due to the ability of the *mem* algorithm to adapt itself to the estimated PSF, therefore correcting inaccuracy in the PSF estimation.

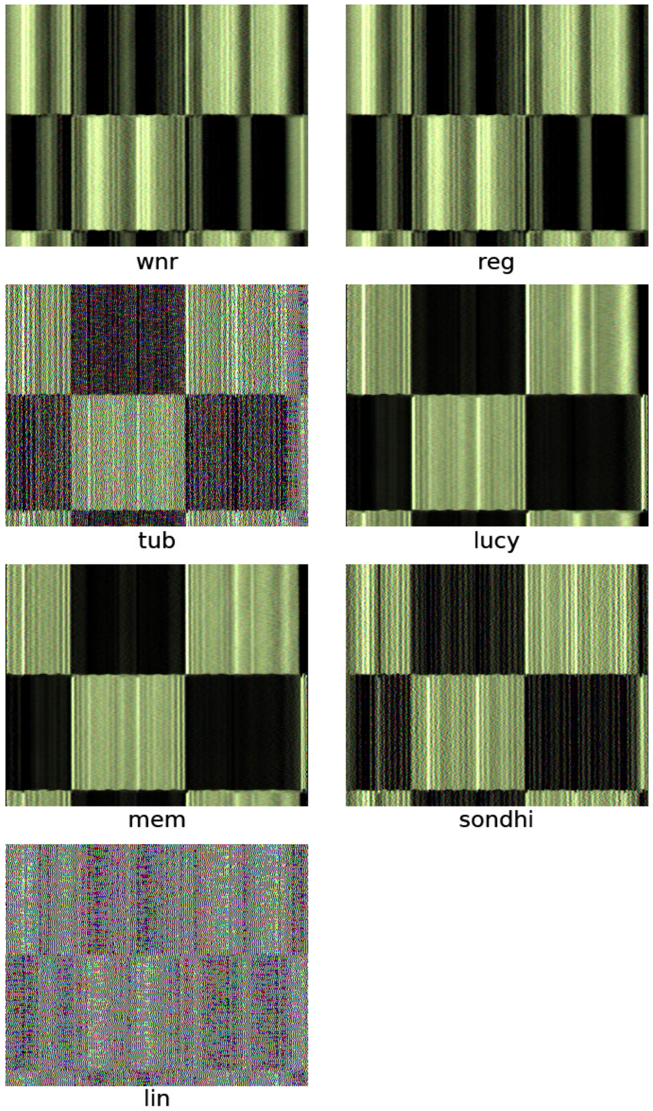

Figure 16:
Restored real blur (checkerboard structure)

5 Conclusion

We have seen that the de-blurring algorithms discussed perform different on synthetic and real motion blur. Two groups of algorithms performed best under both circumstances.

The first group, comprising the *lucy* and *mem* algorithm produce equivalent results for most cases.

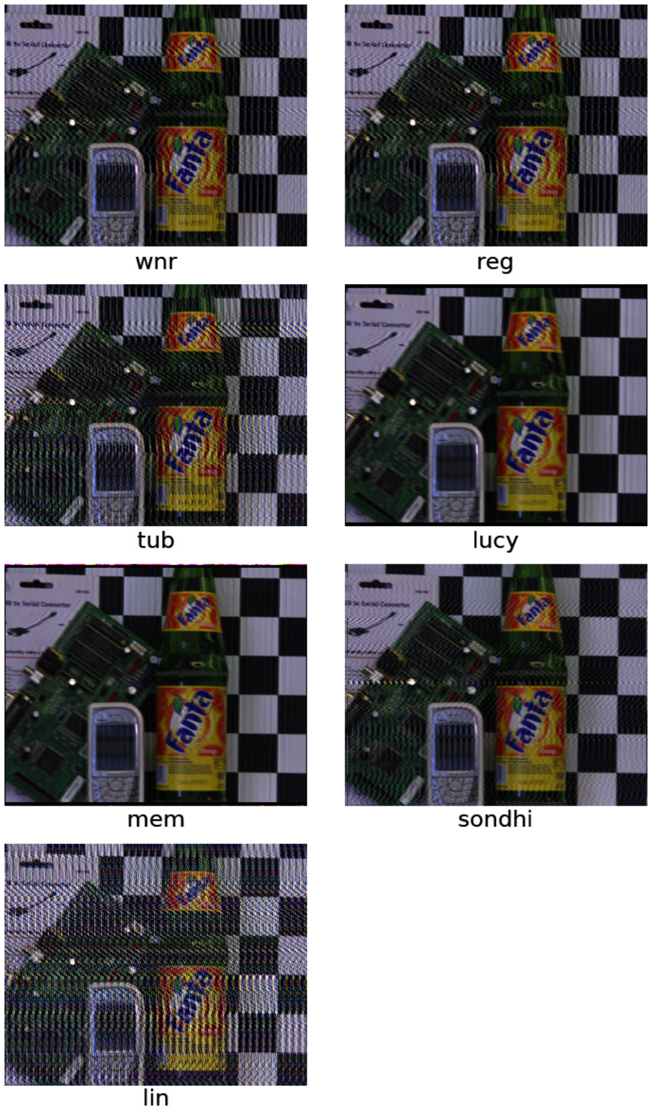

Figure 17:
Restored real blur images (natural scene)

The *lucy* algorithm is preferable as it requires less computation time. In cases where the PSF could only be estimated roughly, the *mem* algorithm is superior as it can adapt itself to the PSF and therefore correct inaccuracies in the estimated PSF. As both algorithms produce images quite pleasant to the human eye, they should be employed when restoring photographs (e.g. from a digital camera). The key for good results on real images with this two algorithms is the ability to weight the pixels and therefore mask pixels at the borders to suppress ringing. Still, their problems remain the tremendous need of computation time and the question of the optimum number of iterations.

The second group, comprising the *tub* and *sondhi* algorithms, deliver sharp restored images, but they also introduce a lot of noise. Therefore, they can be used in applications where sharpness is crucial, e.g. pattern or text recognition. An important advantage over the first group is, that they are remarkably faster and do not need an estimate for the number of iterations (as they are direct algorithms). Furthermore their implementation is quite straight forward.

One can consider to implement the possibility to use the weighting of pixels (as seen in the *lucy* and *mem* implementation) for the second group of algorithms, as this improved the results of the first group a lot.

The *wnr*, *reg* and *lin* algorithms do not produce acceptable results under real world conditions.

References

[1] M. Ben-Ezra and SK Nayar. Motion deblurring using hybrid imaging. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 1, 2003.

[2] M. Ben-Ezra and S.K. Nayar. Motion-Based Motion Deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):689–699, 2004.

[3] D. Capel and A. Zisserman. Super-resolution enhancement of text image sequences. *Proceedings International Conference Pattern Recognition*, pages 600–605, 2000.

[4] Nikon Corporation. Shake Reduction Technology.

[5] DynaPel. DYNAPEL STEADYHAND 2.2.

[6] R. Fergus, B. Singh, A. Hertzmann, S.T. Roweis, and W.T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics (TOG)*, 25(3):787–794, 2006.

[7] R.C. Gonzalez and R.E. Woods. *Digital image processing*. Addison-Wesley Reading, Mass, 1987.

[8] R.C. Gonzalez, R.E. Woods, and S.L. Eddins. *Digital image processing using MATLAB*. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[9] RJ Hanisch, RL White, and RL Gilliland. Deconvolution of Images and Spectra. *Academic Press, CA*, 1997.

[10] T.J. Holmes, S. Bhattacharyya, JA Cooper, D. Hanzel, V. Krishnamurthi, W. Lin, B. Roysam, DH Szarowski, and JN Turner. Light microscopic images reconstructed by maximum likelihood deconvolution. *Handbook of Biological Confocal Microscopy*, pages 389–402, 1995.

[11] X. Jiang, D.C. Cheng, S. Wachenfeld, and K. Rothaus. Motion Deblurring.

[12] L. Liang and Y. Xu. Adaptive Landweber Method to Deblur Images. *IEEE SIGNAL PROCESSING LETTERS*, 10(5):129, 2003.

[13] LB Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79(6):745–754, 1974.

[14] P. Marziliano, F. Dufaux, S. Winkler, T. Ebrahimi, SA Genimedia, and S. Lausanne. A no-reference perceptual blur metric. *Image Processing. 2002. Proceedings. 2002 International Conference on*, 3, 2002.

[15] D. Mery and D. Filbert. A Fast Non-iterative Algorithm for the Removal of Blur Caused by Uniform Linear Motion in X-ray Images. *15th World Conference on Non-Destructive Testing*, 8:15–21, 2000.

[16] J. Mo and RJ Hanisch. Restoration of HST WFPC2 Images in Gyro-Hold Mode. *Astronomical Data Analysis Software and Systems IV*, 77, 1995.

[17] Pentax. Shake Reduction Technology.

[18] R. Raskar, A. Agrawal, and J. Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. *ACM Transactions on Graphics (TOG)*, 25(3):795–804, 2006.

[19] A. Rav-Acha and S. Peleg. Restoration of multiple images with motion blur in differentdirections. *Applications of Computer Vision, 2000, Fifth IEEE Workshop on.*, pages 22–28, 2000.

[20] W.H. Richardson et al. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am*, 62(1):55–9, 1972.

[21] MM Sondhi. Image restoration: The removal of spatially invariant degradations. *Proceedings of the IEEE*, 60(7):842–853, 1972.

[22] C. Williams. It's Degrading; Its Not Delovely. *Security*, 42(5):50–52, 2005.

[23] J.V. Woods. BOUNDARY VALUE PROBLEM IN IMAGE RESTORATION.