Evaluation of Deconvolution Methods

R. Brandt. Supervised by Dr. M.H.F. Wilkinson

Abstract—Where blind deconvolution is the recovery of a true image from a (noisy) convolved image, non-blind deconvolution is the recovery of a true image from the point spread function it is convolved with and a (noisy) convolved image. Four blind and four non-blind deconvolution methods were assessed and compared. Experimental evaluation was performed with different noise types (i.e. shotnoise, Gaussian noise and impulse noise), noise intensities, point spread function types (i.e. out-of-focus blur, Gaussian blur, linear motion blur and non-linear motion blur), point spread function sizes, and true images. The results of the said empirical evaluation were presented in this document. Correlations were found between variables (such as method, noise type, noise intensity, etc.) and reconstruction quality as well as runtime and memory usage. The in this document presented results indicated that the use case dictates which deconvolution method is appropriate.

1 Introduction

The main sources of image degradation in many imaging systems are convolution (resulting in a blurred image) and contamination with a combination of different types of noise (resulting in undesired specks) [21, 2]. Where convolution is caused by the band-limited nature of imaging systems, bad focus or motion, noise is caused by the electronics of the recording and transmission processes [21, 2] or photons.

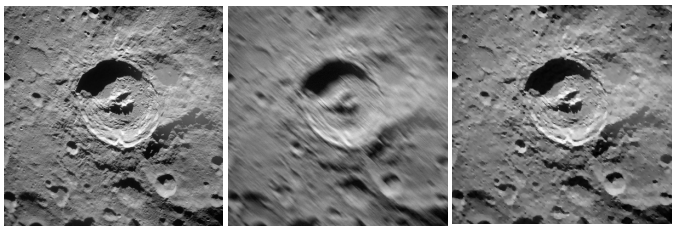
Reducing the said degradation without image processing may be impossible in practice. For example, in x-ray imaging improved image quality occurs with increased x-ray beam intensity, which is bad for the x-ray subject's health [15]. In areas ranging from microscopy to astronomy, deconvolution of images is required to (partially) remove undesired convolution from noisy images [9, 21].

The blurring and noise adding process may be modeled as the linear shift-invariant system

$$g = f \otimes h + \varepsilon, \tag{1}$$

where g denotes the blurred image with noise, f is the true image, $\otimes$ is the convolution operator, h is the point spread function (PSF), and ε is the additive noise [6, 15]. h is nonnegative and has a small support relative to the size of f [16]. N.B.: The system assumes uniform blurring [20] and additive noise.

In non-blind deconvolution, f is estimated from a given g and h [6]. It may be hard in practice to obtain the PSF, giving rise to the blind deconvolution problem. For example, in live video streaming the PSF may not be pre-determinable to allow non-blind deconvolution [15]. In blind deconvolution, f and h are estimated from a given g. An estimate of f is denoted as $\widehat{f}$ and an estimate of h is denoted as $\widehat{h}$ in the remainder of this document.



(a) True image f (b) Degraded image g (c) Deconvolved image $\widehat{f}$

Fig. 1: A true image (a) was convolved and applied noise to, which resulted in (b). Thereafter, (b) was deconvolved by a deconvolution method, which resulted in (c). Fig. (c) is an estimate of (a).

• *R. Brandt is an MSc Computing Science student at the University of Groningen, E-mail: r.brandt@student.rug.nl, S-number: 2509644*

Both blind and non-blind deconvolution are ill-posed problems. In blind deconvolution, multiple combinations of f, h and ε may result in one g.

In non-blind deconvolution with the absence of noise, deconvolution is equal to division in the frequency domain. Perfect deconvolution is hence performed (if $\mathfrak{F}(h) \neq 0$) by computing

$$f = \mathfrak{F}^{-1}\left(\frac{\mathfrak{F}(g)}{\mathfrak{F}(h)}\right),$$

where $\mathfrak{F}$ is the Fourier transform operator and $\mathfrak{F}^{-1}$ is the inverse Fourier transform operator. However, perfect deconvolution may be impossible in the presence of noise. ε may be unknown or only partially known and can hence not e.g. be subtracted when there is contamination with additive noise. Due to the existence of noise ε, inversion of the PSF is often ill-conditioned: The direct inverse of the function usually has a large magnitude at high frequencies, resulting in much more noise in the reconstructed image [15]. Furthermore, multiple combinations of f and ε can generate one g, making the problem ill-posed.

To overcome the ill-posed nature of the problem, deconvolution methods embed hard constraints or regularization: prior knowledge of f, h and ε [2, 1].

Four blind [1, 14, 17, 12] and four non-blind [9, 13, 7, 8] deconvolution methods were assessed and compared. Experimental evaluation was performed with different noise types (i.e. shotnoise, Gaussian noise and impulse noise), noise intensities, PSF types (i.e. out-of-focus blur, Gaussian blur, linear motion blur and non-linear motion blur), PSF sizes, and true images. The results of the said empirical evaluation are presented in this document.

The compared methods are detailed in Section 2. The experimental setup is described in Section 3. Experimental results are presented in Section 4. Results are discussed in Section 5. Conclusions are drawn in Section 6. Future work is outlined in Section 7.

2 Description of Methods

The evaluated deconvolution methods are detailed in this section. Information provided in any subsection of this section is (obviously) derived from the in that subsection referenced deconvolution method proposing paper.

2.1 Blind Image Deblurring With Unknown Boundaries Using Alternating Direction Method Of Multipliers (*mB1*)

The blind deconvolution method proposed in [1] uses the optimization tool "Alternating direction method of multipliers" (defined in [5]) and initially focuses on main edges of an image and later takes details into account. Unknown boundary conditions are assumed by the method as well as a positive, uniformly applied PSF with limited support.

Let g, f, and h be lexicographically ordered vectors containing pixels during the remainder of the description of this method. $g \in \mathbb{R}^n$ is

Algorithm 1 Blind Image Deblurring With Unknown Boundaries Using Aternating Direction Method Of Multipliers.

Require: Blurred image g
Require: Parameters q, λ, $\alpha < 1$, $\mu > 0$, ρ
Require: Stopping criteria S_1 and S_2.
Set $\widehat{h}$ to the identity filter, $\widehat{f} = g$.
repeat
  $\mathbf{G}^j = \mathbf{F}_j \quad$ for $j = 1,...,m$
  $\mathbf{G}^{m+1} = \mathbf{H}$
  $\mu^{(1)} = ... = \mu^{(m)} = \mu$
  $\mu^{m+1} = \rho$
  $d_0^j = 0 \quad$ for $j = 1,...,m+1$
  $u_0^j = \mathbf{G}^j \widehat{f} \quad$ for $j = 1,...,m+1$
  $\widehat{f} =$ ADMMF$(u_0,\, d_0)$ *i.e.* $\min_f C_\lambda(\widehat{f},\, \widehat{h})$
  $u_0^j = \mathbf{G}^j \widehat{h} \quad$ for $j = 1,...,m+1$
  $\widehat{h} =$ ADMMH$(u_0,\, d_0)$ *i.e.* $\min_h C_\lambda(\widehat{f},\, \widehat{h})$
  $\lambda = \alpha\lambda$
until *Stopping criterion S_1 is satisfied*
return Deconvolved image $\widehat{f}$.
return Kernel estimate $\widehat{h}$.

procedure ADMMF(u_0, d_0)
  $k = 0$
  repeat
    $r_{k+1} = \sum_{j=1}^{m+1} \mu^j (\mathbf{G}^j)^T (u_k^j + d_k^j)$
    $\mathbf{K} = (\rho \mathbf{H}^T \mathbf{H} + \mu \sum_{j=1}^{m} \mathbf{F}_i^T \mathbf{F}_i)^{-1}$
    $z_{k+1} = \mathbf{K}(\rho \mathbf{H}^T (u_k^{m+1} + d_k^{m+1}) + \mu \sum_{j=1}^{m} \mathbf{F}_j^T (u_k^j + d_k^j))$
    for $j = 1$ to $m+1$ **do**
      if $j \leq m$ **then**
        $u_{k+1}^j =$v-shrink$((\mathbf{G}^j z_{k+1} - d_k^j), \lambda/\mu, q)$
      else
        $u_{k+1}^j = (\rho \mathbf{I} + \mathbf{M}^T \mathbf{M})^{-1} (\mathbf{M}^T g + \rho(\mathbf{G}^j z_{k+1} - d_k^j))$
      end if
      $d_{k+1}^j = d_k^j - (\mathbf{G}^j z_{k+1} - u_{k+1}^j)$
    end for
    $k = k+1$
  until *Stopping criterion S_2 is satisfied*
  return u
end procedure

procedure ADMMH(u_0, d_0)
  $k = 0$
  repeat
    $r_{k+1} = \sum_{j=1}^{2} \mu^j (\mathbf{G}^j)^T (u_k^j + d_k^j)$
    $z_{k+1} = (\mu^1 \mathbf{X}^T \mathbf{X} + \mu^2 \mathbf{I})^{-1}$
    for $j = 1$ to 2 **do**
      if $j == 1$ **then**
        $u_{k+1}^j = (\mu \mathbf{I} + \mathbf{M}^T \mathbf{M})^{-1} (\mathbf{M}^T g + \mu(\mathbf{G}^j z_{k+1} - d_k^j))$
      else
        $u_{k+1}^j = \text{prox}_{l_{S+}} (\mathbf{G}^j z_{k+1} - d_k^j)$
      end if
      $d_{k+1}^j = d_k^j - (\mathbf{G}^j z_{k+1} - u_{k+1}^j)$
    end for
    $k = k+1$
  until *Stopping criterion S_2 is satisfied*
  return u
end procedure

the degraded image, $f \in \mathbb{R}^m$ is the true image, and $h \in \mathbb{R}^m$ is the PSF. $\mathbf{H} \in \mathbb{R}^{n \times m}$ is a matrix representing convolution with h. $\mathbf{X}$ is a matrix representing convolution with f. $\mathbf{M} \in \{0,1\}^{n \times m}$ is a masking matrix.

The cost function minimized by the method is

$$C_\lambda(f,h) = \frac{1}{2}||g - \mathbf{M}\mathbf{H}f||_2^2 + \lambda \sum_{i=1}^{m} (||\mathbf{F}_i f||_2)^q + l_{S+}(h),$$

where

$$l_{S+}(h) = \begin{cases} 0 & h \in S+ \\ \infty & h \notin S+ \end{cases},$$

$S+$ is the set of filters with positive entries in a given support to enforce a kernel with positive entries, $\mathbf{F}_i \in \mathbb{R}^{4 \times m}$ is a matrix corresponding to four directional Sobel edge filters at pixel i, and $q \in [0,1]$.

In a loop, until stopping criterion S_1 is satisfied, the cost function is first minimized keeping h constant, then it is minimized keeping f constant. Both of the previous minimizations are performed with the aforementioned alternating direction method of multipliers. Lastly, regularization parameter $\lambda > 0$ is decreased.

The complete algorithm is summarized in Algorithm (1). N.B.: $\mathbf{I}$ denotes an identity matrix. The used vectorial shrinkage function v-shrink(g, ψ, q) is defined as

$$\text{v-shrink}(g, \psi, q) = \begin{cases} g \text{ shrink}(1, \psi ||g||_2^{q-2}, q) & \textbf{if } ||g||_2 \neq 0 \\ 0 & \textbf{else} \end{cases},$$

where $\text{shrink}(g, \psi, q) = \min_f \frac{1}{2}||g - f||_2^2 + \psi |f|^q$.

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[1]. Free permission is given to use the implementation for nonprofit research purposes.

2.2 Blind Deconvolution Using a Normalized Sparsity Measure (*mB2*)

The blind deconvolution method proposed in [14] uses the ratio of the ℓ_1 norm to the ℓ_2 norm (ℓ_1 / ℓ_2) on the high frequencies of an image as regularization.

The ℓ_1 norm can penalize the high-frequency bands. As image noise is present in high-frequency bands, minimizing the norm is a way of denoising an image. Penalizing the high-frequency bands will (obviously) also favour blurry images.

The ℓ_1 / ℓ_2 function is a normalized version of ℓ_1, making it scale invariant. Blur decreases the ℓ_2 norm more than the ℓ_1 norm, making the presence of blur derivable from the ratio. Both blur and noise increase the ratio.

Gaussian independent identically distributed noise is assumed to have been added. Uniform blur is assumed, but the algorithm can be extended to allow 3-D motion blur deconvolution.

The cost function which is minimized embeds the ℓ_1 / ℓ_2 norm as second term:

$$\min_{f,h} \lambda ||f \otimes h - g||_2^2 + \frac{||f||_1}{||f||_2} + \psi ||h||_1,$$

where λ and ψ control the relative strength of the kernel and image regularization terms. ℓ_1 regulularization is applied to h to reduce noise in the kernel.

It should be noted that ℓ_1 / ℓ_2 and hence the cost function is non-convex. To combat this, the image estimate and PSF estimate are interchangeably minimized.

The image estimate is updated as

$$\min_f \lambda ||f \otimes h - g||_2^2 + \frac{||f||_1}{||f||_2},$$

where the denominator is kept constant during a minimization iteration because the minimized function is otherwise non-convex. Iterative shrinkage-thresholding algorithm (ISTA) is a method to solve general

[1] http://www.lx.it.pt/~mscla/BID_ADMM_UBC.htm

linear inverse problems. The forenamed algorithm is used to update the image estimate.

The PSF estimate is updated as (constrained by $h \geq 0, \sum_i h_i = 1$)

$$\min_h \lambda ||f \otimes h - g||_2^2 + \psi ||h||_1.$$

The complete algorithm is summarized in Algorithm (2). Please note the following: **H** is the matrix which corresponds to convolution with h; Derivative filters Δ_x and Δ_y are equal to $[1,-1]$ and $[1,-1]^T$ respectively; Multiscale estimation of the kernel using a coarse-to-fine pyramid of image resolutions is looped over on line 2 to avoid some local minima of the cost function; S is the soft shrinkage operation on a vector:

$$S_\alpha(x)_i \;=\; \max(|x_i| - \alpha, 0)\ \mathrm{sign}(x_i).$$

Algorithm 2 Blind Deconvolution Using a Normalized Sparsity Measure.

Require: Blurred image g
Require: Parameters $\alpha,\ \lambda,\ \psi,\ M,\ N,\ t$
1: Apply Δ_x and Δ_y to g, creating y
2: **for** coarse-to-fine levels **do**
3: x = XUPDATE($\widehat{h}, x$)
4: $\widehat{h}$ = HUPDATE($\widehat{h}, x$)
5: Interpolate solution to finer level as initialization.
6: **end for**
7: Deconvolve g using $\widehat{h}$ to give sharp image $\widehat{f}$ by minimizing

$$\min_{\widehat{f}} \lambda ||\widehat{f} \otimes \widehat{h} - g||_2^2 + ||\Delta_x g||_\alpha + ||\Delta_y g||_\alpha$$

return Deconvolved image $\widehat{f}$.
return Kernel estimate $\widehat{h}$.

8: **procedure** XUPDATE(h, x^0)
9: **for** $j = 0$ to $M-1$ **do**
10: $\lambda' \;=\; \lambda ||x^j||_2$
11: $x^{j+1} = \mathrm{ISTA}(h, \lambda', x^j)$
12: **end for**
13: **return** Updated image x^M
14: **end procedure**

15: **procedure** HUPDATE(h,x)
16: Update h using unconstrained iterative re-weighted least squares:

$$\min_h \lambda ||x \otimes h - y||_2^2 + \psi ||h||_1.$$

17: Set negative elements to 0, and normalize.
18: **return** Updated kernel h
19: **end procedure**

20: **procedure** ISTA(h, λ, x^0)
21: **for** $j = 0$ to $N-1$ **do**
22: $v = y - t\mathbf{H}^T(\mathbf{H}x^j - y)$
23: $x^{j+1} \;=\; S_{t\lambda}(v)$
24: **end for**
25: **return** Updated image x^N
26: **end procedure**

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[2]. Free permission is given to use the implementation for research purposes.

[2] https://dilipkay.wordpress.com/blind-deconvolution/

2.3 Efficient marginal likelihood optimization in blind deconvolution (*mB3*)

The maximum a posteriori ($MAP_{f,h}$) approach looks for a pair $(\widehat{f}, \widehat{h}) \;=\; \max \log p(f,h|g)$, which has sparse derivatives and minimizes the convolution error. However, since the total contrast of all derivatives in a blurred image is usually lower than in a sharp one, $MAP_{f,h}$ tends to favor estimated pairs where h is a delta kernel and f is the input blurred image g. To solve this problem, the blind deconvolution method proposed in [17] optimizes the MAP_h score instead. Estimating the kernel alone is better conditioned because the number of parameters to estimate is lower. The method estimates $\widehat{h}$ by computing

$$\widehat{h} \;=\; \max p(h|g) \;=\; \max \int p(f,g|h)df.$$

Taking the integral over all possible f is challenging. To combat this problem, multiple strategies are proposed in the paper. The strategy used to generate results was called "Free-energy algorithm with diagonal co-variance approximation" by the authors.

The method assumes Gaussian noise and a sparse kernel. The function minimized is

$$-\log p(g,f|h) = \frac{||h \otimes f - g||^2}{2\eta^2} + \sum_{i,\gamma} \frac{||\Delta_{i,\gamma}(f)||^2}{2\sigma^2} + c,$$

where c denotes a constant, $\Delta_{i,\gamma}(f)$ denotes the output of $\Delta_\gamma \otimes f$ at the ith pixel, η^2 is the variance of Gaussian noise, and Δ is a set of derivative filters.

Minimization is performed by alternating between solving for the kernel and solving for the image. Given $\widehat{h}$, a mean latent image estimate $\widehat{f}$ is computed using iterative reweighted least squares. A weighted regularizer on the derivatives is added to the cost function of convolution error minimized in this step. The covariance around the mean image $\widehat{f}$ is approximated with a diagonal matrix. Then, the kernel is computed accounting for both the covariance and mean image.

The complete algorithm is summarized in [17], Algorithm 1.

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[3]. Free permission is given to use the implementation for research purposes.

2.4 Blind deconvolution using alternating maximum a posteriori estimation with heavy-tailed priors (*mB4*)

The blind deconvolution method proposed in [12] uses the $MAP_{f,h}$ approach (see Section 2.3). The method assumes Gaussian noise.

Let during the remainder of the description of this method, g, f, and h be vectors containing pixels (lexicographically ordered). As before, $g \in \mathbb{R}^m$ is the degraded image, $f \in \mathbb{R}^m$ is the true image, and $h \in \mathbb{R}^m$ is the PSF.

The function

$$L \;=\; -\log(P(f,h|g)) + const \;=\; \frac{\psi}{2}|f \otimes h - g|_2^2 + Q(f) + R(h) + const$$

is minimized by chaining either f or h while keeping the other constant. In each of the previous cases, the augmented Lagrangian method is used. $Q(f)$ and $R(f)$ are regularizers.

$$Q(f) \;=\; \sum_i ([\Delta_x f]_i^2 + [\Delta_y f]_i^2)^{\frac{p}{2}}, 0 \leq p \leq 1,$$

where Δ_x and Δ_y are partial derivative operators. This term represents the distribution of gradients of natural images.

Laplace distribution is enforced on the positive kernel values to force sparsity and zero on the negative values through

$$R(h) \;=\; \sum_i \Psi(h_i),\ \Psi(h) \;=\; \begin{cases} h_i & \textbf{if } h_i \geq 0 \\ +\infty & \textbf{else} \end{cases}$$

[3] http://webee.technion.ac.il/people/anat.levin/

Algorithm 3 Blind deconvolution using alternating maximum a posteriori estimation with heavy-tailed priors

Require: Blurred image g
Require: Parameters ψ, α, β.
Require: Stopping criteria S_1 and S_2.
Require: Maximum number of iterations $iter_{max}$
for $iter = 1$ to $iter_{max}$ **do**
  $\widehat{f}$ = FUPDATE($\widehat{h}, \widehat{f}$)
  $\widehat{h}$ = HUPDATE($\widehat{h}, \widehat{f}$)
end for
return Deconvolved image $\widehat{f}$.
return Kernel estimate $\widehat{h}$.

procedure FUPDATE(h, f)
  $v_x^0 = 0,\ v_y^0 = 0,\ a_x^0 = 0,\ a_y^0 = 0,\ j = 0$
  repeat
    Solve $(\mathbf{H}^T\mathbf{H} + \frac{\alpha}{\psi}(\Delta_x^T\Delta_x + \Delta_y^T\Delta_y))f^{j+1} =$

$$\mathbf{H}^T g + \frac{\alpha}{\psi}(\Delta_x^T(v_x^j + \alpha_x^j) + \Delta_y^T(v_y^j + a_y^j)) \text{ for } f^{j+1}$$

    $\{[v^{j+1}]_i], [v_i^{j+1}]\} =$

$$LUT_p([\Delta_x f^{j+1} - a_x^j]_i, [\Delta_y f^{j+1} - a_y^j]_i), \forall i$$

    $a_x^{j+1} = a_x^j - \Delta_x f^{j+1} + v^{j+1}$
    $a_y^{j+1} = a_y^j - \Delta_y f^{j+1} + v_y^{j+1}$
    $j = j + 1$
  until Stopping criterion S_1 is satisfied.
return Deconvolved image f.
end procedure

procedure HUPDATE(h, f)
  $v_h^0 = 0,\ a_h^0 = 0,\ j = 0$
  repeat
    Solve $(\mathbf{U}^T\mathbf{U} + \frac{\beta}{\psi}\mathbf{I})h^{j+1} = \mathbf{U}^T g + \frac{\beta}{\psi}(v_h^j + a_h^j)$ for h^{j+1}
    $[v_h^{j+1}]_i = \max([h^{j+1} - a^j]_i - \frac{1}{\beta}, 0), \forall i$
    $a_h^{j+1} = a_h^j - h^{j+1} + v_h^{j+1}$
    $j = j + 1$
  until Stopping criterion S_2 is satisfied.
return Kernel estimate h.
end procedure

The complete algorithm is summarized in Algorithm (3). Please note that LUT refers to a lookup table. $\mathbf{U}$ is convolutional operator constructed from f.

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[4]. Free permission is given to use the implementation for research purposes.

2.5 Fast High-Quality non-Blind Deconvolution Using Sparse Adaptive Priors (*mNB1*)

The non-blind deconvolution method proposed in [9] uses sparse adaptive priors which preserves strong edges, while penalizing ones below a given threshold (noise level). The method assumes a linear blurring model with Gaussian white noise. The reconstructed image is

$$\widehat{f} = \min_f ||hf - g||_2^2 + \sum_{s=1}^{5} \lambda_s ||d_s f - w_s||_2^2,$$

where the matrices $d_s, s \in 1,..,5$, represent the first and second-order-derivative filter operators: Δ_x, Δ_y, Δ_{xx}, Δ_{yy} and Δ_{xy}. $\lambda_s > 0$ are regularization weights. w_s allows to specify a set of priors on the derivatives of f, they are the expected or specified responses of these filters for the true image f: $w_s = d_s f$. h is a matrix representing convolution with the blurring kernel. f and g are vectorized images as in $mB1$.

The complete algorithm is summarized in Algorithm (4). $.^*$ represents complex conjugate. $./$ represents element-wise matrix division. $*$ is the element-wise matrix-product operator. $\mathbf{H} = \mathfrak{F}(h)$, $\mathbf{G} = \mathfrak{F}(g)$, $\mathbf{D}_s = \mathfrak{F}(d_s)$, and $\mathbf{W}_s = \mathfrak{F}(w_s)$. EPS() is an edge-preserving filter. A threshold representing some noise level is represented by ψ.

Algorithm 4 Fast High-Quality non-Blind Deconvolution Using Sparse Adaptive Priors.

Require: Blurred image g
Require: Blurring kernel h
Require: Parameters λ_s, ψ
1: $w_s = 0$
2: $A = \mathbf{H}^* * \mathbf{H} + \sum_{s=1}^{5} \lambda_s \mathbf{D}_s^* * \mathbf{D}_s$
3: $B = \mathbf{H}^* * \mathbf{G} + \sum_{s=1}^{5} \lambda_s \mathbf{D}_s^* * \mathbf{W}_s$
4: $\widehat{f} = \mathfrak{F}^{-1}(A./B)$
5: $\widehat{f} =$ EPS($\widehat{f}$).
6: $w_s = \frac{d_s \widehat{f}}{(\frac{\psi}{d_s f})^4 + 1}$.
7: $A = \mathbf{H}^* * \mathbf{H} + \sum_{s=1}^{5} \lambda_s \mathbf{D}_s^* * \mathbf{D}_s$
8: $B = \mathbf{H}^* * \mathbf{G} + \sum_{s=1}^{5} \lambda_s \mathbf{D}_s^* * \mathbf{W}_s$
9: $\widehat{f} = \mathfrak{F}^{-1}(A./B)$
return Deconvolved image $\widehat{f}$.

An initial approximation is obtained in line 1-4 using Tikhonov regularization (i.e. with $w_s = 0$). A new estimate is obtained by applying an edge-preserving smoothing filter to the initial estimate to reduce noise while preserving important edges in line 5. The actual regularization priors as a set of sparse first and second-order derivatives of $\widehat{f}$ are computed in line 6. The final version of the deconvolved image is obtained in line 7-9.

It should be noted that the algorithm limits border ringing artifacts caused by the fact that real word convolution isn't circular. To that end, the input image is padded before performing deconvolution and the result is cropped to remove the extra pixels. Padding is done by replicating the images first and last columns and rows a number of times depending on the kernel size.

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[5]. Permission to use the said implementation appears to be given for research purposes or is otherwise granted through fair use.

2.6 Fast Image Deconvolution using Hyper-Laplacian Priors (*mNB2*)

The non-blind deconvolution method proposed in [13] uses the heavy-tailed (hyper-Laplacian) distribution of gradients in natural scenes as priors. The method assumes the presence of zero mean Gaussian noise. Frequency domain operations are used which assume circular boundary conditions.

The cost function which is minimized is

$$\min_{f,w} \sum_{i=1} \left(\frac{\lambda}{2}(f \otimes h - g)_i^2 + \frac{\beta}{2}(||\mathbf{F}_i^1 f - w_i^1||_2^2 + ||\mathbf{F}_i^2 f - w_i^2||_2^2) + |w_i^1|^\alpha + |w_i^2|^\alpha \right),$$

where f is the true image of N pixels. λ is a regularization weight. $\Delta_1 = [1, -1]$ and $\Delta_2 = [1, -1]^T$. $\mathbf{F}_i^j x = (x \otimes \Delta_j)_i$ for $j = 1,...,J$. β is an optimization weight. $|.|^\alpha$ is a penalty function.

The cost function is minimized by separately minimizing for w and f while keeping the other fixed.

[4] http://zoi.utia.cas.cz/deconv_sparsegrad

[5] http://www.inf.ufrgs.br/~oliveira/pubs_files/FD/FD_page.html

The complete algorithm is summarized in Algorithm (5). **H** is the matrix corresponding to convolution with h.

Algorithm 5 Fast image deconvolution using hyper-Laplacian priors.

Require: Blurred image g
Require: Blurring kernel h
Require: Parameters β_0, β_{inc}, β_{max}, α, λ
Require: Maximum number of iterations $iter_{max}$
1: $\beta = \beta_0, \widehat{f} = g$
2: *Precompute the constant terms used in line 7.*
3: **while** $\beta < \beta_{max}$ **do**
4: **for** $i = 0$ to $iter_{max}$ **do**
5: $v = \mathbf{F}_i^j \widehat{f}$
6: $w = \min_w |w|^\alpha + \frac{\beta}{2}(w-v)^2$
7: $\widehat{f} = \mathfrak{F}^{-1}\left(\frac{\mathfrak{F}(\mathbf{F}^1)^* * \mathfrak{F}(w^1) + \mathfrak{F}(\mathbf{F}^2)^* * \mathfrak{F}(w^2) + (\lambda/\beta)\mathfrak{F}(\mathbf{H})^* * \mathfrak{F}(g)}{\mathfrak{F}(\mathbf{F}^1)^* * \mathfrak{F}(\mathbf{F}^1) + \mathfrak{F}(\mathbf{F}^2)^* * \mathfrak{F}(\mathbf{F}^2) + (\lambda/\beta)\mathfrak{F}(\mathbf{H})^* * \mathfrak{F}(\mathbf{H})}\right)$
8: **end for**
9: $\beta = \beta_{inc} * \beta$
10: **end while**
return Deconvolved image $\widehat{f}$.

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[6]. Permission to use the said implementation appears to be given for research purposes or is otherwise granted through fair use.

2.7 An augmented Lagrangian method for total variation video restoration (*mNB3*)

The non-blind deconvolution method proposed in [7] uses the augmented Lagrangian method for total variation for restoration. The method can be used for deconvolution of videos as well as images.

The method stacks the frames of a video to form a 3-D data structure. By imposing regularization functions along the spatial and temporal direction, both spatial and temporal smoothness is enforced.

Two minimization problems can be solved by the method.

The TV/L2 minimization problem

$$\min_{\widehat{f}} \frac{\mu}{2}||\mathbf{H}\widehat{f} - g||_2 + ||\widehat{f}||_{TV},$$

as well as the TV/L1 minimization problem

$$\min_{\widehat{f}} \mu||\mathbf{H}\widehat{f} - g||_1 + ||\widehat{f}||_{TV}.$$

The TV norm $||f||_{TV}$ mentioned in the previous is defined as

$$||f||_{TV} = \sum_i (\beta_x|[\Delta_x f]_i| + \beta_y|[\Delta_y f]_i| + \beta_t|[\Delta_t f]_i|),$$

where Δ_x, Δ_y and Δ_t are the forward finite-difference operators along respectively the horizontal, vertical, and temporal directions. β_x, β_y and β_t are constants, and f_i denotes the ith component of the vector representation of f.

The TV/L2 minimization problem was solved during the performed experiments. The complete algorithm is summarized in [7], Algorithm 1.

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[7]. Free permission is given to use the implementation for research purposes.

[6] https://dilipkay.wordpress.com/fast-deconvolution/

[7] http://zoi.utia.cas.cz/deconv_sparsegrad

2.8 Handling Outliers in Non-blind Image Deconvolution (*mNB4*)

The non-blind deconvolution method proposed in [8] was designed to be robust against saturated/clipped pixels (due to overexposure), non-Gaussian noise, and nonlinear camera response curves which violate the linear blur model presented in Equation (1). It is assumed only for inliers that additive noise is spatially independent, and follows a Gaussian distribution. A shift-invariant PSF is assumed.

The complete algorithm is summarized in Algorithm (6).

Algorithm 6 Handling Outliers in Non-blind Image Deconvolution

Require: Blurred image g
Require: Blurring kernel h
Require: Probability that g_x is an inlier P_{in}.
Require: Probability that g_x is an outlier P_{out}.
Require: Noise intensity parameter λ.
Require: σ is the standard deviation of the Gaussian distribution.
Require: Maximum number of iterations $iter_{max}$
1: $w_x^m = 1, w_x^h = 1, w_x^v = 1 \; \forall x$
2: $\widehat{f} = \min_{\widehat{f}} \sum_x w_x^m||g_x - (h \otimes \widehat{f})_x||_2 + \lambda\psi(\widehat{f})$
3: **for** $iter = 1$ to $iter_{max}$ **do**
4: $E[m_x] = \begin{cases} \frac{\mathcal{N}(g_x|\widehat{f}_x^o,\sigma)P_{in}}{\mathcal{N}(g_x|\widehat{f}_x^o,\sigma)P_{in}+CP_{out}} & \textbf{if } \widehat{f}_x^o \in DR \\ 0 & \textbf{else} \end{cases}$
5: $\widehat{f} = \min_{\widehat{f}} \sum_x w_x^m||g_x - (h \otimes \widehat{f})_x||_2 + \lambda\psi(\widehat{f})$
6: **end for**
return Deconvolved image $\widehat{f}$.

Please note that DR is the dynamic range in the input image.

$$\psi(\widehat{f}) = \sum_x \{w_x^h||\Delta_h(\widehat{f})_x||_2 + w_x^v||\Delta_v(\widehat{f})_x||_2\}.$$

$\mathcal{N}$ is a Gaussian distribution. C is a constant defined as the inverse of the width of the dynamic range in the input image. Δ_h and Δ_v are differential operators along the x and y directions, respectively.

Observed pixel intensities are classified as inliers if their formation satisfies Equation (1) or as outliers otherwise. The outliers are excluded from the deconvolution process. Since the said classification is unknown, an expectation-maximization method which alternatingly computes the expectation of the classification mask m and deconvolution using the expectation.

The *MATLAB* implementation provided by the algorithm's authors was used to generate results[8]. Free permission is given to use the implementation for research purposes.

3 Experimental Setup

A synthetic dataset was created consisting out of triples of a true image f, a point spread function h, and a convolved image with noise added to it g. The eight deconvolution methods were each given exactly the same third (and second in case of a non-blind deconvolution method) elements of the 576 triples. Each of the said triples was created using a unique combination of a kernel type (one of 4), kernel size (one of 3), noise type (one of 3), noise intensity (one of 4), and true image (one of 4). Reconstruction quality and efficiency were determined for 4.608 combinations of an algorithm (one of 8) applied to one of the 576 data set entries.

Furthermore, a synthetic dataset was created consisting out of triples of a constant true image f at one of eight scales, a constant point spread function h scaled with the same factor as the image, and a convolved image with a constant amount and type of noise added to it g. Reconstruction efficiency was determined for 64 combinations of an algorithm (one of 8) applied to one of the 8 data set entries.

The experimental setup is described in detail in the remainder of this section.

[8] https://github.com/CoupeLibrary/handleoutlier

3.1 Noise

Image degradation due to noise may be caused by noise types such as shot, impulse, and Gaussian noise [4].

Shot noise is caused by the fact that electromagnetic waves consist out of photos which are units which cannot be subdivided (i.e. no fraction of a photon is possible) and are emitted with random variation [4, 22]. When an imaging system records an image consisting out of two pixels by counting photons for a number of milliseconds where the true image would be uniform in gray level, shot noise may cause the pixels to differ in gray level because a different number of photons for each pixel is recorded. Shot noise has a probability density function equal to that of the Poisson distribution [4]. Note that Poisson noise is neither additive nor multiplicative [22].

Impulse (salt and pepper) noise may be caused by bit errors in data transmission and is either equal to the maximum or minimum value a pixel can have (e.g. 0 or 255 in an 8-bit image) [4]. Impulse noise may also be caused by dead pixels [8].

Gaussian noise arises in amplifiers and detectors in imaging systems and has a probability density function equal to that of the Gaussian distribution [4].

The robustness to all of the forenamed noise types was evaluated. Noise intensity of different types of noise was made comparable by expressing noise intensity in PSNR dB, i.e. the difference between $f \otimes h$ and g in terms of PSNR. The commonly used metric Blurred Signal-to-Noise Ratio (BSNR) was not used because the said metric assumes additive noise [11]. To apply the noise types, the *MATLAB* function imnoise was modified. The function does not take a value in dB as input by default. To convert the input of the function to noise intensity in dB, a binary search algorithm was implemented. Furthermore, all existing random number generators were assigned constant seed values which make sure the same noise is applied when the function is called with the same parameters.

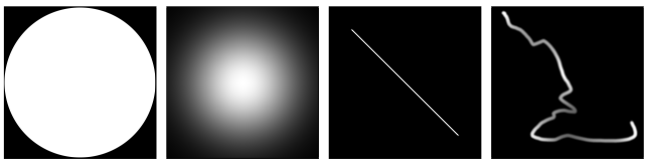
3.2 Point spread functions

Multiple types of point spread functions were used to create blurred images, each is illustrated in Fig. (2). The choice of point spread functions was inspired by those used in [1].

An image containing bokeh which is visible in out-of-focus blur is produced by PSF (a). Gaussian blur which corresponds to a low pass filter is produced by PSF (b). Linear motion blur of which the reconstruction may be easier than that of PSF (d) is produced by PSF (c). Non-linear motion blur is produced by PSF (d).

N.B.: Natural images may contain non-uniform blurring. For example, the size of bokeh in images may be bigger for objects which are further away from the focus plane of a camera. Furthermore, 3D rotation of the camera may have caused motion blur. The point spread functions considered in the experiments are all spatially invariant because only very few of the considered methods are able to deal with non-uniform blur.

The point spread functions were re-sized into multiple sizes and convolved with the in Section 3.3 presented true images.



(a) (b) (c) (d)

Fig. 2: Point spread functions producing (a) Out-of-focus, (b) Gaussian, (c) Linear motion, and (d) Non-linear motion blur.

3.2.1 Boundary conditions

Convolution between true image f with size $M \times N$ and PSF h with kernel size $s \times s$ (before padding) may be performed assuming for example that pixels outside the boundary are equal to a constant (e.g. 0), are equal to the value of the nearest border pixel or that the whole image is cyclically repeated.

All of the forenamed assumptions may be false. The boundary condition most true to nature was used to apply convolution to the true images: f and h are multiplied in the frequency domain. Thereafter, g is cropped such that only the pixels

$$\left\{(x,y) \in \mathbb{W}^2 \mid x > \frac{s}{2} \wedge x < M - \frac{s}{2} \wedge y > \frac{s}{2} \wedge y < N - \frac{s}{2}\right\}$$

are kept which are unaffected by the ambiguous boundary.

3.3 True images

Textures may be characterized as being either grainy, rough/bumpy, smooth or uniform. Images were collected which contain one or more of these characteristics. In order to allow for the results presented in this document to be compared with those presented in [19], the same true images were used. The images were used to create synthetically blurred and noisy images.

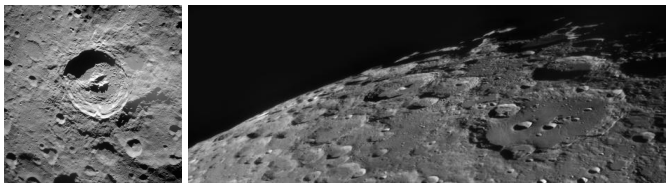
The sun images contain smooth streams. Due to the smoothness, it may be more difficult for blind methods to recover the PSF.

The moon images contain a very bumpy and rough surface. The sharpness of the relatively sharp edge between the dark background and moon in deconvolved images is interesting as well as the sharpness of other bumpy parts of the image.

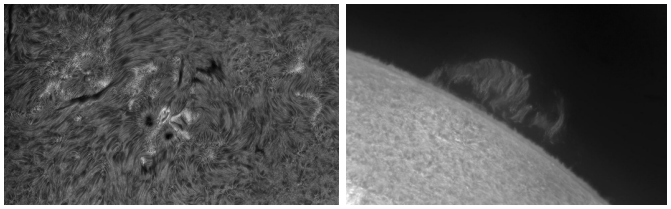
Image m2 and su2 contain a uniform black background. It is interesting to observe to what extent noise is removed from these regions. Furthermore, the strong edge between the background and foreground will show to what extent the deconvolution methods introduce ringing artifacts which are commonly introduced by deconvolution methods near strong edges [18].

The images were re-sized such that they each consist of approximately an equal number of pixels. A better possibility to find a correlation between image type and time and memory usage is in this way aimed for. Results should still be well comparable with those presented in [19] since the used metrics are invariant to image size.

Details about each of the true images are provided in Table (1). The true images are illustrated in Fig. (3).



(m1) The King crater (m2) The south pole



(su1) Sun surface (su2) Solar prominence

Fig. 3: True images as detailed in Table (1).

Id	Name	Size	Source
m1	The King crater	416×416	NASA[9]
m2	The south pole	682×254	Dr. M.H.F. Wilkinson[10]
su1	Sun surface	538×322	Dr. M.H.F. Wilkinson
su2	Solar prominence	526×329	Dr. M.H.F. Wilkinson

Table 1: True images.

3.4 Metrics

Comparing restoration results requires a measure of image quality. Because the dataset is synthetic, quality measures which use both the true and reconstructed as input were used. In order to assess the alikeness of deconvolved images with respect to their true image, the Peak signal-to-noise ratio *PSNR*) and Structural similarity index (*SSIM*) were used. These measures were chosen because they are well-known metrics [10] which allow linking the results detailed in this document with those presented in related papers.

In order to determine whether results need to be expressed in both the *SSIM* and *PSNR* (i.e. using one of them doesn't suffice), both were computed in all performed experiments. Subsequently, the Spearman's rho correlation (defined in [3]) between the two metrics was computed. The said value which is equal to -0.75 indicates that the two metrics are not derivable from each other. A correlation of -1/+1 of the said type indicates perfect derivability where a value of 0 indicates there is no correlation.

In each experiment, both metrics were computed between $\widehat{f}$ and f, $\widehat{h}$ and h, and g and f.

3.4.1 Peak signal-to-noise ratio *PSNR*

A mean-squared error is strongly influenced by the range of gray levels in images [9, 10]. Peak Signal-to-Noise Ratio *PSNR* avoids this problem by scaling the *MSE* according to the image range [9, 10]. In the presence of images with equal gray level range, the Spearman's rho correlation between the two metrics is (as expected) -1, making it unnecessary to include results expressed in *MSE*.

The peak signal-to-noise ratio *PSNR* is defined as [9, 10]:

$$PSNR = 10\, log_{10}\, \frac{I_{max}^2}{MSE},$$

where I_{max} is the maximum signal extent (e.g. $2^8 = 255$ in eight 8-images), and *MSE* is the mean square error defined in equation (2).

$$MSE = \frac{1}{n} \sum_x \sum_y [\widehat{f} - f]^2, \tag{2}$$

where n is number of pixels making up f, f is the true image, and $\widehat{f}$ is the deconvolved image.

A higher *PSNR* value indicates a higher image quality since it approaches ∞ as the *MSE* approaches zero.

3.4.2 Structural similarity index *SSIM*

Structural similarity index [23] differs from *PSNR* in that it is considered to be correlated with the quality perception of the human visual system [10]. The structural similarity index is defined as [10]:

$$SSIM = l(f,\widehat{f})c(f,\widehat{f})s(f,\widehat{f}), \tag{3}$$

where

$$l(f,\widehat{f}) = \frac{2\mu_f\mu_{\widehat{f}} + C_1}{\mu_f^2 + \mu_{\widehat{f}}^2 + C_1},$$

$$c(f,\widehat{f}) = \frac{2\sigma_f\sigma_{\widehat{f}} + C_2}{\sigma_f^2 + \sigma_{\widehat{f}}^2 + C_2},$$

$$s(f,\widehat{f}) = \frac{\sigma_{f\widehat{f}} + C_3}{\sigma_f\sigma_{\widehat{f}} + C_3},$$

μ_f is the average of f, $\mu_{\widehat{f}}$ is the average of $\widehat{f}$, σ_f is the variance of f, $\sigma_{\widehat{f}}$ is the variance of $\widehat{f}$, and $\sigma_{f\widehat{f}}$ is the covariance of f and $\widehat{f}$.

In Equation (3), the closeness of the mean luminance of the two images is evaluated by the first term. The closeness of the contrast of the two images is evaluated by the second term. The correlation coefficient between the two images is evaluated by the third term. The positive constants C_1, C_2 and C_3 are added to avoid a null denominator.

Where a *SSIM* value of 0 indicated that there is no correlation between the images, a value of 1 means that they are equal.

3.4.3 Computational complexity and memory usage

The computational efficiency of the deconvolution methods was compared by considering their runtime in seconds after each method was given the same input. The memory efficiency of the deconvolution methods was compared by considering the peak memory allocation of the function call initiating the deconvolution method after each method was given the same input. The *MATLAB* profile function was used to determine both runtime and peak memory allocation.

The memory and time complexity of the algorithms were estimated as a function of image size in kB and pixels respectively. The m1 image and Gaussian kernel were resized with the same eight factors, as detailed in Table (2).

Image Id	Size in px	Size in kB	PSF	Size in px
m1	500×500	739 KB	b	7×7
m1	1000×1000	2.946 KB	b	15×15
m1	1500×1500	6.636 KB	b	21×21
m1	2000×2000	11.789 KB	b	29×29
m1	2500×2500	18.409 KB	b	35×35
m1	3000×3000	26.490 KB	b	43×43
m1	3500×3500	36.025 KB	b	49×49
m1	4000×4000	47.018 KB	b	57×57

Table 2: Test cases for runtime and memory usage evaluation.

3.5 Parameters and Implementation settings

Method parameters were manually tweaked to give the best results in the presence of mediocre (35 dB) noise while balancing between speed and quality.

The existing implementations which were used to generate results were stripped to only include their core functionality, i.e. plotting and printing of results as well as needless storing intermediate results were disabled. Furthermore, methods which use lookup tables were configured such that the tables need to be computed each time an image is deconvolved. The previous was done to allow fair time/memory comparison.

MATLAB version 2018a was used to generate results.

4 Experimental Results

Correlations were found between variables (such as method, noise type, noise intensity, etc.) and reconstruction quality as well as runtime and memory usage. The correlations which were discovered are presented in this section. Note that when no correlation between variables is mentioned, none was observed.

[9] https://spaceflight.nasa.gov/gallery/images/apollo/apollo16/html/as16-122-19580.html

[10] http://rug.nl/staff/m.h.f.wilkinson

4.1 Quantitative comparison

4.1.1 Influence of noise on reconstruction quality

The influence of the noise type and intensity on reconstruction quality is presented expressed in SSIM in Table (4) and expressed in PSNR in Table (5).

Both of the forenamed tables consist out of two horizontal blocks. The top one indicates results of the four blind and the bottom one indicates the results of the four non-blind deconvolution methods. The column named "N" indicates the noise type which was applied. "G" is Gaussian noise, "S" is salt-and-pepper noise and "P" is Poisson noise. The column named "I" indicates the intensity of the applied noise in dB. Said values were calculated as explained in Section 3.1. Obviously, a noise intensity of ∞ indicates the absence of noise. Each pair of columns to the right of column "I" indicates the performance of one of the eight compared methods. For each of the methods, the SSIM/PSNR value between reconstructed image $\widehat{f}$ and true image f is indicated in the SSIM/PSNR columns. The "gain" columns indicate how much the SSIM/PSNR value of the reconstructed image has increased or decreased relative to the SSIM/PSNR value of convolved and noised image g and true image f. A large value indicates a great improvement, where a negative value indicates a reduction in quality.

Convolved and noised images g can differ in noise type and amount, but also in true image, PSF type, and size. All non-bold values are therefore medians of all results obtained for the indicated method, noise type and amount.

The bold values underneath each of the two horizontal blocks indicate overall median values. The bold values at the bottom of a noise type block indicate the median of all results where the specified noise type was applied.

4.1.2 Influence of PSF on reconstruction quality

The influence of the point spread function type and size on reconstruction quality is presented expressed in SSIM in Table (6) and expressed in PSNR in Table (7).

Both of the forenamed tables consist out of two horizontal blocks. The top one indicates results of the four blind and the bottom one indicates the results of the four non-blind deconvolution methods. The column named "P" indicates the PSF type which was applied. Kernel "a" creates out-of-focus blur, kernel "b" creates Gaussian blur, kernel "c" creates linear motion blur and kernel "d" creates non-linear motion blur. The column named "S" indicates the width/height of the applied blurring kernels in pixels. Each pair of columns to the right of column "I" indicates the performance of one of the eight compared methods. For each of the methods, the SSIM/PSNR value between reconstructed image $\widehat{f}$ and true image f is indicated in the SSIM/PSNR columns. The "gain" columns indicate how much the SSIM/PSNR value of the reconstructed image has increased or decreased relative to the SSIM/PSNR value of convolved and noised image g and true image f. A large value indicates a great improvement, where a negative value indicates a reduction in quality.

Convolved and noised images g can differ in PSF type and size, but also in true image, noise type, and size. All non-bold values are therefore medians of all results obtained for the indicated method, PSF type and amount.

The bold values underneath each of the two horizontal blocks indicate overall median values. The bold values at the bottom of a PSF type block indicate the median of all results where the specified PSF type was applied.

4.1.3 Influence of true image on reconstruction quality

The influence of true image on reconstruction quality is presented in Table (8). The "Img" column indicates which true image was deconvolved. The image identifiers are linked to image name, size and source in Table (1). The true images are illustrated in Fig. (3).

Each pair of columns to the right of column "Img" indicates the performance of one of the eight compared methods. For each of the methods, the SSIM and PSNR "gain" value is indicated. The values indicate either in terms of SSIM or PSNR how the reconstructed image has increased or decreased relative to the SSIM/PSNR value of convolved and noised image g and true image f. A large value indicates a great improvement, where a negative value indicates a reduction in quality.

Convolved and noised images g can differ in true image, but also in PSF type and size, and noise type and size. All non-bold values are therefore medians of all results obtained for the indicated true image. The bold values indicate overall median values.

4.1.4 Influence of PSF type on its reconstruction quality

The influence of which point spread function type was used for convolution on the reconstruction quality of the kernel by the blind deconvolution methods is presented in Table (3). The column named "PSF" indicates the PSF type which was applied. Kernel "a" creates out-of-focus blur, kernel "b" creates Gaussian blur, kernel "c" creates linear motion blur, and kernel "d" creates non-linear motion blur.

Each pair of columns to the right of column "PSF" indicates the performance of one of the four blind methods. For each of the methods, the SSIM and PSNR value is indicated between kernel approximation $\widehat{h}$ and true kernel h. All non-bold values are medians of all results obtained for the indicated PSF type. The bold values indicate overall median values.

	Method B1		**Method B2**	
PSF	SSIM	PSNR	SSIM	PSNR
a	0.88727	39.01393	0.92167	43.09829
b	0.91164	38.1657	0.91537	42.33528
c	0.84416	35.49457	0.90296	38.77692
d	0.76249	35.64483	0.86715	39.27326
	0.85044	**36.07067**	**0.90299**	**40.52819**

	Method B3		**Method B4**	
PSF	SSIM	PSNR	SSIM	PSNR
a	0.89334	39.74011	0.95936	47.28691
b	0.79003	33.47619	0.94175	41.45865
c	0.84218	36.77014	0.94792	45.03301
d	0.81497	36.85534	0.94694	44.15327
	0.82048	**36.55461**	**0.95029**	**44.66997**

Table 3: Influence of PSF type on its reconstruction quality. Table is explained in Section 4.1.4.

4.1.5 Computational efficiency

The total runtime divided by image size of g in pixels was constant for all algorithms for images with a size greater than 500×500. The previous suggests that the algorithms have a linear time complexity as a function of the number of pixels in the input image g. The runtime of *mB1*, *mB2*, *mB3*, *mB4*, *mNB1*, *mNB2*, *mNB3* and *mNB4* is equal to index 16.25, 5.1, 752.3, 13.5, 1, 1, 3.6, and 64.1 respectively. Given the same inputs, *mB1* will hence take about 16.25 times longer to terminate than *mNB2*.

No other significant correlations were found between variables (e.g. noise intensity) and the total runtime of methods.

4.1.6 Memory usage

The peak memory usage divided by image size of g in kB was constant for all algorithms. The previous suggests that the algorithms have a linear memory complexity as a function of image size in kB. For *mB1*, *mB2*, *mB3*, *mB4*, *mNB1*, *mNB2*, *mNB3* and *mNB4* said value is equal to index 1.55, 1.45, 1.5, 1.5, 1.10, 1, 2, and 1 respectively. Given the same inputs, *mB1* will hence consume about 1.55 times more memory than *mNB2*.

No other significant correlations were found between variables (e.g. noise intensity) and peak memory usage of methods.

Table 4: Influence of noise on reconstruction quality in SSIM. Table is explained in Section 4.1.1.

		Method B1		**Method B2**		**Method B3**		**Method B4**	
N	I	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain
G	30	0.21023	-0.16721	0.14408	-0.24692	0.3387	-0.091393	0.096589	-0.30035
	35	0.32665	-0.1693	0.29038	-0.23646	0.55905	0.022595	0.19855	-0.31868
	40	0.56317	-0.0079896	0.49065	-0.114	0.68433	0.096826	0.59107	-0.01136
	∞	0.67203	0.043548	0.64286	0.024796	0.68568	0.076189	0.66672	0.076465
		0.42956	**-0.1069**	**0.34264**	**-0.16822**	**0.56**	**0.05294**	**0.25729**	**-0.20628**
S	30	0.53373	-0.010884	0.52147	-0.043569	0.54716	-0.0006794	0.41742	-0.16675
	35	0.58164	-0.0042171	0.55735	-0.020431	0.58677	-0.0037617	0.49464	-0.14882
	40	0.61248	0.0092338	0.60148	0.0087899	0.62041	0.042721	0.44459	-0.28546
	∞	0.67203	0.043548	0.64286	0.024796	0.68568	0.076189	0.66672	0.076465
		0.60765	**0.0039694**	**0.575**	**-0.012893**	**0.62432**	**0.022612**	**0.48894**	**-0.10529**
P	30	0.25742	-0.16221	0.21809	-0.2439	0.44081	-0.056291	0.13946	-0.33041
	35	0.42628	-0.14159	0.3574	-0.19493	0.58734	0.0013371	0.27324	-0.27997
	40	0.60305	-0.032569	0.51136	-0.091589	0.6865	0.095757	0.56381	-0.060057
	∞	0.67203	0.043548	0.64286	0.024796	0.68568	0.076189	0.66672	0.076465
		0.46273	**-0.10264**	**0.40719**	**-0.16941**	**0.59692**	**0.045071**	**0.36963**	**-0.19712**
		0.50062	**-0.045811**	**0.4578**	**-0.11107**	**0.59406**	**0.036618**	**0.39576**	**-0.17571**

		Method NB1		**Method NB2**		**Method NB3**		**Method NB4**	
N	I	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain
G	30	0.17323	-0.22855	0.2922	-0.1066	0.045079	-0.36938	0.41703	0.040552
	35	0.36555	-0.13468	0.60589	0.096831	0.18892	-0.30049	0.67702	0.15867
	40	0.58478	0.061934	0.67097	0.079174	0.32244	-0.13261	0.75825	0.15107
	∞	0.90001	0.22909	0.6862	0.068875	0.58231	0.0658	0.76889	0.137
		0.45913	**-0.061149**	**0.60017**	**0.070649**	**0.1978**	**-0.28213**	**0.67474**	**0.13923**
S	30	0.40562	-0.13864	0.47095	-0.067707	0.14623	-0.31235	0.76827	0.17908
	35	0.66952	0.054373	0.61402	0.019025	0.44092	-0.062027	0.82947	0.14856
	40	0.81888	0.15902	0.66727	0.05902	0.5055	0.026421	0.76881	0.14071
	∞	0.90001	0.22909	0.6862	0.068875	0.58231	0.0658	0.76889	0.137
		0.70121	**0.089328**	**0.59197**	**-0.0020654**	**0.43208**	**-0.16101**	**0.77892**	**0.15319**
P	30	0.1946	-0.25636	0.38354	-0.10745	0.051723	-0.38366	0.47759	0.036005
	35	0.38943	-0.14247	0.60768	0.090866	0.1957	-0.30379	0.69699	0.12715
	40	0.63742	0.089227	0.67285	0.078423	0.33178	-0.13211	0.76191	0.14107
	∞	0.90001	0.22909	0.6862	0.068875	0.58231	0.0658	0.76889	0.137
		0.52133	**-0.03527**	**0.59979**	**0.049188**	**0.23183**	**-0.27497**	**0.68227**	**0.12688**
		0.56815	**0.019186**	**0.59614**	**0.039919**	**0.29928**	**-0.25077**	**0.7116**	**0.13847**

Table 5: Influence of noise on reconstruction quality in PSNR. Table is explained in Section 4.1.1.

		Method B1		**Method B2**		**Method B3**		**Method B4**	
N	I	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain
G	30	19.668	-3.352	13.948	-10.558	22.103	-2.573	14.085	-10.295
	35	22.442	-3.1036	16.64	-9.0273	24.62	-0.65083	15.848	-9.8154
	40	24.864	-0.93075	19.896	-5.5362	26.506	1.4302	23.513	-2.733
	∞	25.846	0.77792	24.006	-1.1153	26.794	1.6718	25.903	0.36761
		23.174	**-1.8524**	**17.991**	**-7.7081**	**25.034**	**0.6808**	**18.497**	**-7.6446**
S	30	22.967	-0.13603	20.447	-2.7989	23.364	-0.98153	17.846	-6.4775
	35	24.006	-0.23487	21.546	-2.9517	23.86	-0.90797	17.625	-7.1423
	40	24.82	-0.36948	22.655	-1.4471	25.829	0.13651	16.813	-10.885
	∞	25.846	0.77792	24.006	-1.1153	26.794	1.6718	25.903	0.36761
		24.469	**-0.11835**	**21.847**	**-2.1891**	**25.002**	**-0.1078**	**19.803**	**-6.132**
P	30	19.035	-3.8367	13.979	-10.762	21.202	-3.1954	13.725	-10.271
	35	22.025	-3.2126	16.379	-9.3131	23.86	-1.2647	15.955	-9.6191
	40	24.82	-1.3666	19.706	-5.9325	26.509	1.1721	22.119	-4.5056
	∞	25.846	0.77792	24.006	-1.1153	26.794	1.6718	25.903	0.36761
		22.983	**-2.1933**	**17.899**	**-7.8726**	**24.602**	**0.44992**	**17.906**	**-7.8994**
		23.592	**-1.3918**	**19.894**	**-5.9799**	**24.852**	**0.21349**	**18.742**	**-7.3545**

		Method NB1		**Method NB2**		**Method NB3**		**Method NB4**	
N	I	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain
G	30	16.575	-7.6149	17.748	-6.1816	9.1099	-14.438	21.608	-1.763
	35	21.321	-3.631	21.946	-0.93767	15.209	-9.8949	26.69	2.0256
	40	25.806	0.57472	22.336	-0.64258	16.672	-7.7756	27.579	2.6915
	∞	31.807	4.2376	22.42	-0.62958	18.722	-5.759	27.841	2.7809
		23.279	**-1.8504**	**21.251**	**-3.5766**	**14.405**	**-10.464**	**26.726**	**2.0831**
S	30	17.635	-6.3787	17.171	-6.2965	11.184	-12.442	27.816	4.9671
	35	22.083	-2.8758	20.081	-3.9084	16.2	-8.6465	29.77	4.9832
	40	25.835	0.7641	21.763	-1.8796	17.206	-7.0053	27.839	3.4337
	∞	31.807	4.2376	22.42	-0.62958	18.722	-5.759	27.841	2.7809
		23.77	**-1.2426**	**20.326**	**-4.2642**	**15.301**	**-9.6661**	**28.367**	**3.6893**
P	30	16.023	-8.0302	16.666	-6.4335	9.4325	-14.169	20.63	-2.55
	35	20.651	-4.3652	21.78	-1.3048	14.631	-10.484	25.608	1.6022
	40	25.489	0.41502	22.267	-0.62779	16.596	-7.9004	27.557	2.4682
	∞	31.807	4.2376	22.42	-0.62958	18.722	-5.759	27.841	2.7809
		22.841	**-2.4215**	**21.012**	**-3.8685**	**14.31**	**-10.875**	**26.159**	**1.8442**
		23.467	**-1.8504**	**20.899**	**-3.9048**	**14.852**	**-10.03**	**27.103**	**2.5501**

Table 6: Influence of point spread function on reconstruction quality in SSIM. Table is explained in Section 4.1.2.

		Method B1		**Method B2**		**Method B3**		**Method B4**	
P	S	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain
a	7	0.60492	0.023662	0.51709	-0.13066	0.63425	0.033176	0.46315	-0.23929
	11	0.46069	0.0096906	0.43652	-0.079783	0.52318	0.063262	0.31778	-0.15433
	15	0.32307	-0.026333	0.36181	-0.051538	0.40543	0.05873	0.31865	-0.1146
		0.48887	**-0.010884**	**0.42578**	**-0.085282**	**0.5451**	**0.047244**	**0.34829**	**-0.16773**
b	7	0.71686	0.017839	0.59558	-0.22189	0.72488	0.034461	0.58606	-0.23933
	11	0.5787	0.068201	0.51358	-0.13976	0.6248	0.027141	0.47001	-0.16332
	15	0.47704	0.040418	0.44663	-0.087804	0.5752	0.016528	0.46185	-0.16655
		0.59791	**0.036573**	**0.47808**	**-0.10768**	**0.64521**	**0.026554**	**0.49173**	**-0.17705**
c	7	0.65824	-0.036716	0.60081	-0.17385	0.72696	0.010707	0.50912	-0.27779
	11	0.59247	-0.04612	0.57285	-0.0858	0.55856	0.034954	0.48433	-0.20054
	15	0.51414	-0.039142	0.51865	-0.076467	0.47566	0.013611	0.44889	-0.12618
		0.58725	**-0.042681**	**0.57117**	**-0.082204**	**0.57992**	**0.014129**	**0.46991**	**-0.18949**
d	7	0.46911	-0.086399	0.43369	-0.19529	0.60997	0.017893	0.3888	-0.25641
	11	0.24051	-0.21234	0.3476	-0.1312	0.53181	0.057561	0.39646	-0.075761
	15	0.23274	-0.19193	0.32071	-0.12873	0.4552	0.063322	0.28979	-0.072169
		0.32863	**-0.16935**	**0.36888**	**-0.1356**	**0.55895**	**0.053749**	**0.34805**	**-0.14545**
		0.50062	**-0.045811**	**0.4578**	**-0.11107**	**0.59406**	**0.036618**	**0.39576**	**-0.17571**

		Method NB1		**Method NB2**		**Method NB3**		**Method NB4**	
P	S	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain	SSIM	SSIM gain
a	7	0.53694	-0.072714	0.62516	-0.0094179	0.28721	-0.36662	0.73119	0.13021
	11	0.47075	-0.0094272	0.49508	-0.027223	0.17425	-0.28258	0.60856	0.15615
	15	0.43526	0.0010673	0.4225	-0.045828	0.085778	-0.28818	0.55792	0.16429
		0.47944	**-0.034396**	**0.47654**	**-0.026837**	**0.1391**	**-0.31692**	**0.67556**	**0.15313**
b	7	0.69981	0.0040137	0.74748	0.045649	0.46959	-0.28722	0.86167	0.12685
	11	0.59428	0.024728	0.59099	0.067477	0.34403	-0.29679	0.77845	0.15166
	15	0.5251	0.026635	0.50653	0.062984	0.24622	-0.30261	0.71509	0.14266
		0.58025	**0.020222**	**0.66192**	**0.060657**	**0.34331**	**-0.29679**	**0.7923**	**0.13544**
c	7	0.68705	-0.052316	0.72349	-0.0078685	0.58752	-0.13082	0.79458	0.082892
	11	0.57492	-0.038751	0.61217	-0.020441	0.49785	-0.16074	0.71589	0.13402
	15	0.53318	-0.016473	0.54332	-0.053999	0.42036	-0.18936	0.67235	0.123
		0.59104	**-0.03158**	**0.58571**	**-0.038877**	**0.45969**	**-0.15744**	**0.7116**	**0.10236**
d	7	0.7199	0.066431	0.7524	0.12315	0.42204	-0.17044	0.76726	0.14716
	11	0.68447	0.093751	0.70616	0.18099	0.23506	-0.25789	0.63173	0.17904
	15	0.61963	0.103	0.6553	0.18537	0.18892	-0.20464	0.61858	0.19806
		0.6522	**0.09385**	**0.70593**	**0.15673**	**0.26375**	**-0.22057**	**0.67695**	**0.16644**
		0.56815	**0.019186**	**0.59614**	**0.039919**	**0.29928**	**-0.25077**	**0.7116**	**0.13847**

Table 7: Influence of point spread function on reconstruction quality in PSNR. Table is explained in Section 4.1.2.

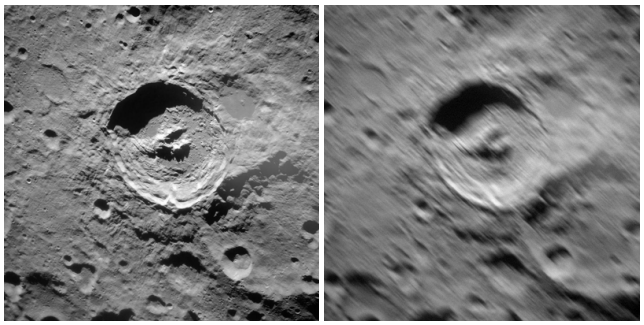
		Method B1		**Method B2**		**Method B3**		**Method B4**	
P	S	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain
a	7	25.55	0.010308	19.741	-8.1435	25.725	0.27163	18.194	-9.5283
	11	23.87	-0.018478	19.593	-5.6983	24.451	0.61446	18.206	-7.691
	15	22.823	-0.22195	20.565	-3.0732	23.928	0.4634	19.7	-5.1841
		24.278	**-0.1178**	**19.894**	**-5.1191**	**25.018**	**0.37445**	**18.791**	**-7.3008**
b	7	26.667	-0.34048	20.364	-10.249	27.295	0.14608	19.208	-10.871
	11	26.114	0.75099	19.999	-7.1808	23.846	0.0059521	18.997	-7.3724
	15	24.468	0.69874	20.839	-5.0981	23.695	-0.50425	18.884	-7.1211
		25.779	**0.5538**	**20.531**	**-6.4612**	**25.307**	**-0.1556**	**18.966**	**-7.4491**
c	7	24.025	-3.2653	19.457	-9.0991	26.509	-0.6538	17.815	-10.268
	11	23.057	-2.3294	20.922	-6.6516	24.176	-0.026613	19.446	-8.724
	15	22.606	-0.99553	21.175	-4.0477	23.146	-0.34069	20.919	-5.2753
		23.233	**-2.2114**	**20.13**	**-6.2588**	**25.266**	**-0.37919**	**19.641**	**-8.2839**
d	7	22.895	-2.7229	18.795	-7.5474	25.475	-0.28644	17.157	-9.7937
	11	20.031	-3.7069	18.201	-5.2544	24.678	1.0384	19.078	-5.5459
	15	19.442	-3.0362	18.378	-4.4545	23.75	0.69906	18.409	-4.2695
		20.711	**-3.146**	**18.466**	**-6.033**	**24.57**	**0.67174**	**18.346**	**-6.4452**
		23.592	**-1.3918**	**19.894**	**-5.9799**	**24.852**	**0.21349**	**18.742**	**-7.3545**

		Method NB1		**Method NB2**		**Method NB3**		**Method NB4**	
P	S	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain	PSNR	PSNR gain
a	7	22.814	-3.8045	19.854	-5.4359	14.463	-11.966	27.823	2.3107
	11	21.47	-2.8943	18.199	-5.6754	12.569	-11.492	26.719	2.7645
	15	20.671	-2.3237	16.841	-6.0743	10.398	-11.827	26.06	2.8076
		21.514	**-3.2449**	**18.335**	**-5.6754**	**12.34**	**-11.827**	**26.346**	**2.6273**
b	7	26.151	-2.023	22.39	-4.5238	17.842	-8.9569	30.228	3.0652
	11	24.118	-1.1348	20.084	-2.8462	15.426	-9.3902	27.903	2.728
	15	23.068	-0.69677	19.16	-2.7366	13.452	-10.03	27.069	2.9281
		23.834	**-1.3026**	**21.575**	**-4.5107**	**16.033**	**-9.7651**	**28.384**	**2.9592**
c	7	25.882	-2.3646	23.008	-3.4222	19.323	-9.4288	28.647	2.0337
	11	23.894	-2.397	20.341	-4.2771	16.156	-9.2194	27.103	2.1032
	15	22.714	-2.5133	19.365	-3.9021	13.117	-9.6861	25.876	1.8802
		23.894	**-2.485**	**20.969**	**-3.9081**	**15.654**	**-9.5718**	**27.142**	**1.9376**
d	7	25.239	-0.71112	24.257	-1.8906	18.008	-7.6727	28.225	2.7719
	11	24.125	0.27619	21.932	-1.8796	14.903	-9.8416	26.789	2.9846
	15	22.989	0.25305	20.757	-1.8474	14.178	-9.1843	26.213	3.2262
		24.005	**-0.33895**	**21.973**	**-1.8796**	**15.678**	**-9.1167**	**26.806**	**3.1719**
		23.467	**-1.8504**	**20.899**	**-3.9048**	**14.852**	**-10.03**	**27.103**	**2.5501**

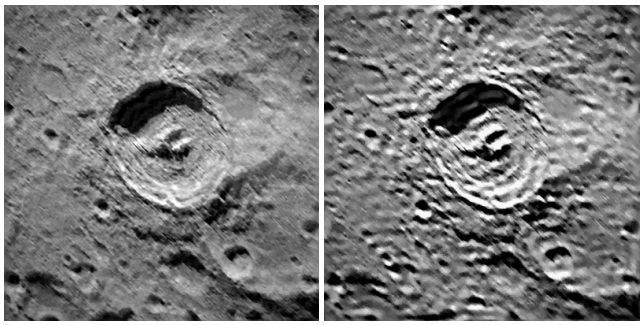
Table 8: Influence of true image on reconstruction quality. Table is explained in Section 4.1.3.

	Method B1		**Method B2**		**Method B3**		**Method B4**	
Img	PSNR gain	SSIMR gain	PSNRR gain	SSIMR gain	PSNR gain	SSIM gain	PSNR gain	SSIM gain
m1	-0.45176	-0.021864	-3.6415	-0.06247	0.67467	0.077227	-3.6163	-0.016016
m2	-1.4533	-0.075467	-2.6125	-0.06348	0.75681	0.05361	-4.7316	-0.12991
su1	-1.1805	-0.026495	-5.3089	-0.042745	-0.0094282	0.045466	-8.9364	-0.22622
su2	-3.278	-0.12072	-12.442	-0.26744	-0.90141	-0.0012658	-11.79	-0.26216
	-1.3918	**-0.045811**	**-5.9799**	**-0.11107**	**0.21349**	**0.036618**	**-7.3545**	**-0.17571**

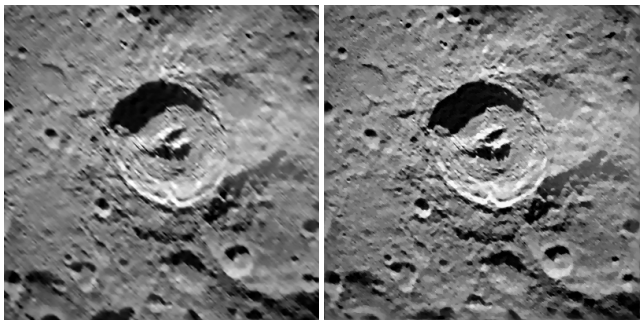
	Method NB1		**Method NB2**		**Method NB3**		**Method NB4**	
Img	PSNR gain	SSIM gain	PSNR gain	SSIM gain	PSNR gain	SSIM gain	PSNR gain	SSIM gain
m1	1.1084	0.14663	0.15536	0.1301	-4.5443	-0.00977	3.334	0.22456
m2	-0.36697	-0.013749	-5.4192	-0.010045	-10.103	-0.31903	3.3397	0.15622
su1	-1.8504	0.082856	-0.23788	0.085353	-6.9796	-0.10181	1.5289	0.14165
su2	-8.2236	-0.3031	-14.726	-0.14391	-20.283	-0.59035	2.6607	0.052952
	-1.8504	**0.019186**	**-3.9048**	**0.039919**	**-10.03**	**-0.25077**	**2.5501**	**0.13847**



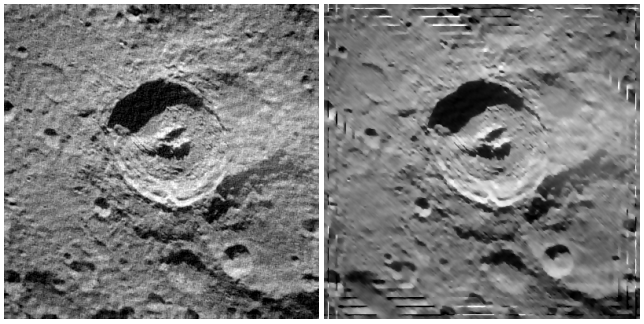
(a) True image f (b) Degraded image g_{40}



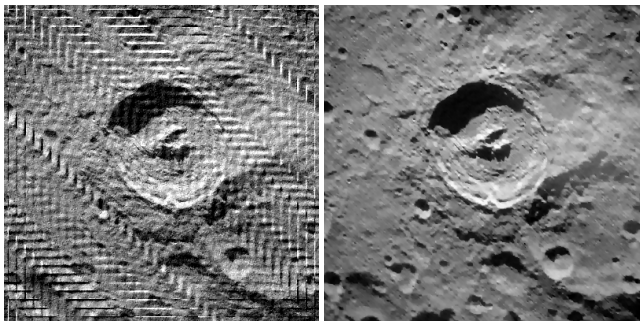
(c) Reconstruction $\widehat{f}$ by *mB1* (d) Reconstruction $\widehat{f}$ by *mB2*



(e) Reconstruction $\widehat{f}$ by *mB3* (f) Reconstruction $\widehat{f}$ by *mB4*

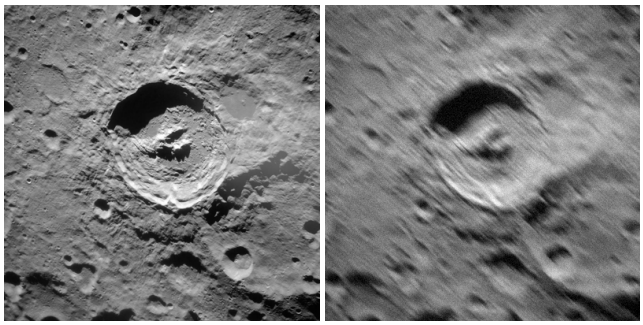


(g) Reconstruction $\widehat{f}$ by *mNB1* (h) Reconstruction $\widehat{f}$ by *mNB2*

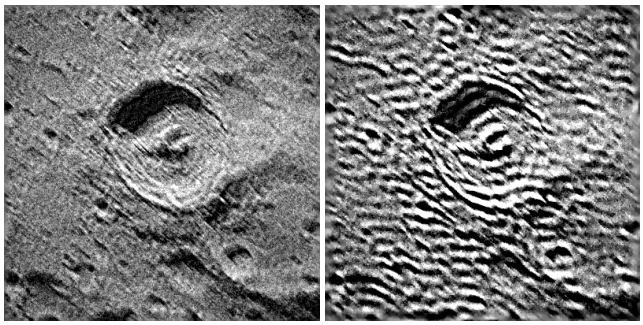


(i) Reconstruction $\widehat{f}$ by *mNB3* (j) Reconstruction $\widehat{f}$ by *mNB4*

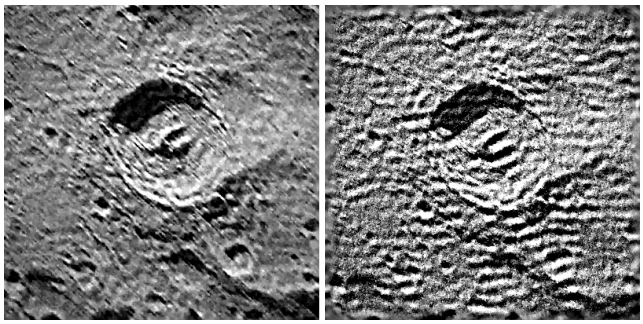
Fig. 4: Reconstruction of f in the presence of 40 dB noise.



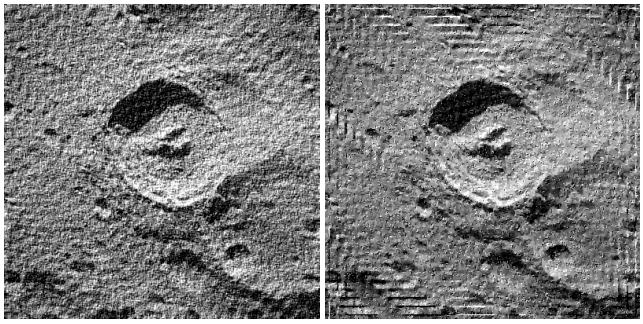
(a) True image f (b) Degraded image g_{30}



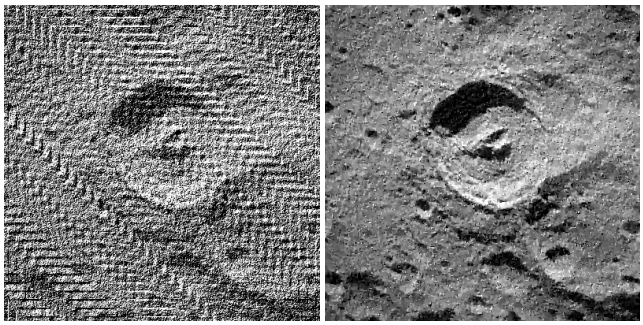
(c) Reconstruction $\widehat{f}$ by *mB1* (d) Reconstruction $\widehat{f}$ by *mB2*



(e) Reconstruction $\widehat{f}$ by *mB3* (f) Reconstruction $\widehat{f}$ by *mB4*

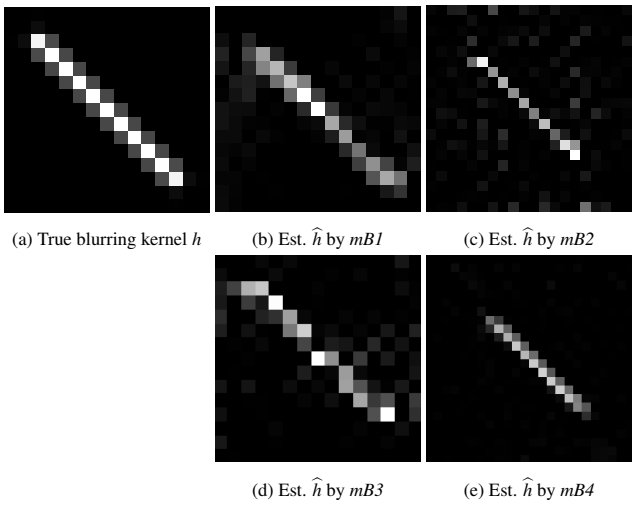


(g) Reconstruction $\widehat{f}$ by *mNB1* (h) Reconstruction $\widehat{f}$ by *mNB2*



(i) Reconstruction $\widehat{f}$ by *mNB3* (j) Reconstruction $\widehat{f}$ by *mNB4*

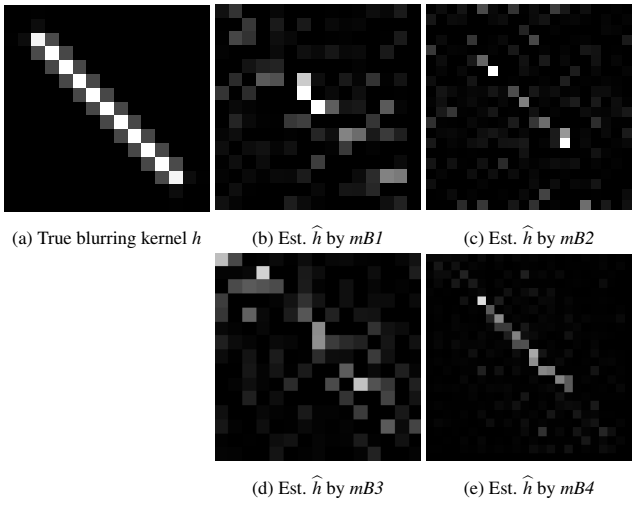
Fig. 5: Reconstruction of f in the presence of 30 dB noise.



(a) True blurring kernel h (b) Est. $\widehat{h}$ by *mB1* (c) Est. $\widehat{h}$ by *mB2*

(d) Est. $\widehat{h}$ by *mB3* (e) Est. $\widehat{h}$ by *mB4*

Fig. 6: Reconstruction of h in the presence of 40 dB noise.



(a) True blurring kernel h (b) Est. $\widehat{h}$ by *mB1* (c) Est. $\widehat{h}$ by *mB2*

(d) Est. $\widehat{h}$ by *mB3* (e) Est. $\widehat{h}$ by *mB4*

Fig. 7: Reconstruction of h in the presence of 30 dB noise.

4.2 Qualitative comparison

Noise has a great influence on reconstruction quality (as was explained in Section 1 and observable in Table (4) and Table (5)). Said influence is illustrated in Fig. (4-7). True image m1 (f) was convolved with a linear motion blur kernel of size 15×15 and 40 dB of Gaussian noise was added to it, resulting in degraded image g_{40}. The methods were used to deconvolude g_{40}. The estimates of f are illustrated in Fig. (4). The by the blind methods reconstructed kernels are illustrated in Fig. (6).

Thereafter, true image m1 (f) was convolved with a linear motion blur kernel of size 15×15 and 30 dB of Gaussian noise was added to it, resulting in degraded image g_{30}. The methods were used to deconvolved g_{30}. The estimates of f are illustrated in Fig. (5). The by the blind methods reconstructed kernels are illustrated in Fig. (7).

5 Discussion

5.1 Influence of noise on reconstruction quality

The influence of noise type and intensity on reconstruction quality is presented in Table (4) and Table (5), and illustrated in Fig. (4) and Fig. (5).

In the absence of noise, method *mB2*, *mNB2* and *mNB3* did not have positive PSNR gain values. The previous suggests that the methods are unable to improve degraded images (all methods generally decreased reconstruction quality when more noise was added). The methods do have positive (but relatively small) SSIM gain values in the same situation. Perfect deconvolution in the non-blind setting is easy to obtain in the absence of noise and if $\mathfrak{F}(h) \neq 0$ through division of $\mathfrak{F}(g)$ by $\mathfrak{F}(h)$. The best performing non-blind deconvolution method *mNB1* obtained a non-optimal SSIM of 0.9 (not 1) and an PSNR of 31.807 (not ∞). The best reconstructing blind deconvolution method in the absence of noise was *mB3*.

Where *mNB1* was the best performing method in the absence of noise, *mNB4* was the most noise resistant. Method *mNB4* was the only method which has positive SSIM gain values for any noise type and intensity. Methods *mB3*, *mNB1* and *mNB4* stand out as being relatively noise resistant.

In terms of both *SSIM* and *PSNR* gain, all noise types are best dealt with by method *mNB4*. Among the blind methods, *mB3* was best able to cope with all noise types in terms of both *SSIM* and *PSNR* gain. N.B.: *mB3* was unable to improve in terms of median PSNR gain when there was contamination with salt and pepper noise.

None of the methods are better able to cope with Poisson noise than any other noise type. Method *mB3* and *mNB2* are better able to cope with Gaussian noise than any other noise type. Method *mB1* and *mNB1* are better able to cope with salt-and-pepper noise than any other noise type.

In deconvolved noisy images, (ringing) artifacts can be observed primarily in the result of *mB2*, *mB4*, *mNB2* and *mNB3* as is illustrated in Fig. (4).

In the result of *mNB2*, the center of the image is relatively sharp, yet there are significant artifacts visible at the border. The result of *mNB4* was arguably the best for both noise intensities. The sensitivity to noise of *mB4* was highlighted by the significant difference in the quality of the result between 40 dB and 30 dB noise. Conversely, the robustness to noise of *mNB4* was highlighted by the small difference in quality of the result between 40 dB and 30 dB noise.

5.2 Influence of PSF on reconstruction quality

The influence of point spread function type and size on reconstruction quality is presented in Table (6) and Table (7).

In terms of both *SSIM* and *PSNR* gain, the best performing method in the presence of any PSF type is *mNB4*. Please note the good performance of method *mNB2* on by non-linear motion blurred images. Among the blind methods, *mB3* was best able to cope with out-of-focus, linear and non-linear motion blur. Method *mB1* was best able to cope with Gaussian blur.

Method *mB3*, *mNB1*, *mNB2* and *mNB4* are best able to cope with non-linear motion blur relative to other PSF types. Method *mB1* is best able to cope with Gaussian bur relative to other PSF types.

5.3 Influence of true image on reconstruction quality

The influence of true image on reconstruction quality is presented in Table (8).

All images were best deconvolved by method *mNB4*. Among the blind methods, all images were best deconvolved by *mB3*. N.B: no positive median SSIM or PSNR gain value was obtained by *mB3* when given a convolved version of image su2.

The su2 image appears to be particularly difficult to deconvolute for all methods. An explanation for this could be that the image contains few sharp edges and a lot of uniform blackness. m1 was, in general, the easiest to deconvolute, and does not contain any of said properties. A presence of uniform blackness may also explain the relatively bad performance of the algorithms when given a convolved version of image m2.

5.4 Influence of PSF type and noise on its reconstruction quality

The influence of which point spread function type was used for convolution on the reconstruction quality of the kernel by the blind deconvolution methods is presented in Table (3).

All PSF types are best reconstructed by method *mB4* except the Gaussian one which is when measured in PSNR best reconstructed by *mB2*. The previous is interesting because the reconstructed images of method *mB3* were generally better than those of the other blind methods. Where *mB3* is the worst at reconstructing the true kernel, it is best at reconstructing the true image.

The good reconstruction performance of *mB4* is also visible in Fig. (6) and Fig. (7). A relatively great robustness to noise of *mB4* can be observed in the said images.

6 Conclusion

Four blind and four non-blind deconvolution methods were assessed and compared. Experimental evaluation was performed with different noise types (i.e. shotnoise, Gaussian noise and impulse noise), noise intensities, PSF types (i.e. out-of-focus blur, Gaussian blur, linear motion blur and non-linear motion blur), PSF sizes, and true images. The results of the said empirical evaluation were presented in this document.

Correlations were found between variables (such as method, noise type, noise intensity, etc.) and reconstruction quality as well as runtime and memory usage.

The results show that the use case dictates which deconvolution method is appropriate.

When a low amount of noise (up to about 40 dB) is present in a convolved image, the use of method *mB3* and *mNB1* are advised if reconstruction quality is significantly more important than computational efficiency and memory usage. The use of *mB1* and *mNB1* are advised otherwise.

In case there is a high amount of noise in the convolved image, the best method also depends on time and memory constraints:

Methods *mB1* or *mB3* and *mNB2* or *mNB4* are advised for respectively blind and non-blind deconvolution when reconstruction quality is significantly more important than computational efficiency and memory usage.

Method *mB3* performed relatively badly when there was salt-and-pepper noise and is much less computationally efficient than *mB1*. The difference between the two in terms of reconstruction quality may be considered relatively small in this case. The usage of *mB1* is therefore advised when there is contamination with salt and pepper noise and the use of *mB3* is advised otherwise.

Method *mNB2* reconstructed particularly well when given a by non-linear motion blurred image and is much more computationally efficient than *mNB4*. The difference between the two in terms of reconstruction quality is relatively small in this case. The usage of *mNB2* is therefore advised when there is contamination with non-linear motion blur and the use of *mNB4* is advised otherwise.

Method *mB1* and *mNB1* or *mNB2* are advised for respectively blind and non-blind deconvolution if reconstruction quality is less important than computational efficiency and memory usage. The use of method *mNB1* is advised in the presence of linear motion blur, and the use of *mNB2* is advised otherwise.

7 Future work

The evaluated deconvolution methods can be improved and assessed more extensively:

The quality of blurring kernels estimated by *mB2* was better than those estimated by *mB1* and *mB3*. Still, the reconstruction quality of true images estimated by *mB2* was worse than those estimated by *mB1* and *mB3*. The previous suggests that line 7 in Algorithm 2 is a cause of *mB2*'s worse performance. Method *mB2* uses *mNB2* to perform what's to be done in said line. The reconstruction quality of method *mB2* may be increased when method *mNB4* replaces line 7 in Algorithm 2.

Likewise, even though the reconstruction quality of blurring kernels estimated by *mB4* was the best among all methods, the reconstruction quality of true images estimated by *mB4* was worse than those estimated by *mB3*. The reconstruction quality of method *mB4* may be increased when method *mNB1*, *mNB2* or *mNB4* are ran after termination of *mB4* and non-blind deconvolute g using the by *mB4* estimated kernel.

Artifacts at the borders of deconvolved images created by method *mNB2* may be reduced by padding convolved images before deconvolution and removing the introduced padding after deconvolution.

Reconstruction quality may be increased (especially in the presence of impulse noise) by applying a thresholded median filter as a preprocessing step to the methods. Likewise, it may be beneficial to reduce Poisson noise using e.g. a non-local mean or bilateral filter.

Other deconvolution methods could be investigated using the same experimental setup which was used to generate the results presented in this document. Furthermore, the performance of the considered methods when there is contamination with other noise types could be investigated as well as other uniform PSF types such as one creating box blur.

References

[1] M. S. Almeida and M. A. Figueiredo. Blind image deblurring with unknown boundaries using the alternating direction method of multipliers. In *20th IEEE International Conference on Image Processing (ICIP)*, pages 586–590. IEEE, 2013.

[2] M. Bertero and P. Boccacci. Image deconvolution. In *From Cells to Proteins: Imaging Nature across Dimensions*, pages 349–370. Springer, 2005.

[3] D. Best and D. Roberts. Algorithm as 89: the upper tail probabilities of spearman's rho. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(3):377–379, 1975.

[4] A. K. Boyat and B. K. Joshi. A review paper: noise models in digital image processing. *An International journal*, 6(2):63–75, 2015.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[6] M. Cannon. Blind deconvolution of spatially invariant image blurs with phase. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(1):58–63, 1976.

[7] S. H. Chan, R. Khoshabeh, K. B. Gibson, P. E. Gill, and T. Q. Nguyen. An augmented lagrangian method for total variation video restoration. *IEEE Transactions on Image Processing*, 20(11):3097–3111, 2011.

[8] S. Cho, J. Wang, and S. Lee. Handling outliers in non-blind image deconvolution. In *IEEE International Conference on Computer Vision (ICCV 2011)*, pages 1–8, 2011.

[9] H. E. Fortunato and M. M. Oliveira. Fast high-quality non-blind deconvolution using sparse adaptive priors. *The Visual Computer*, 30(6-8):661–671, 2014.

[10] A. Hore and D. Ziou. Image quality metrics: Psnr vs. ssim. In *20th international conference on Pattern recognition (ICPR)*, pages 2366–2369. IEEE, 2010.

[11] S. Jayaraman and S. Esakkirajan. *Digital image processing*. McGraw Hill, 2009.

[12] J. Kotera, F. Šroubek, and P. Milanfar. Blind deconvolution using alternating maximum a posteriori estimation with heavy-tailed priors. In *International Conference on Computer Analysis of Images and Patterns*, pages 59–66. Springer, 2013.

[13] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *Advances in Neural Information Processing Systems 22*, pages 1033–1041. Curran Associates, Inc., 2009.

[14] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 233–240. IEEE, 2011.

[15] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE signal processing magazine*, 13(3):43–64, 1996.

[16] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 1964–1971. IEEE, 2009.

[17] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2657–2664. IEEE, 2011.

[18] A. Mosleh, J. P. Langlois, and P. Green. Image deconvolution ringing artifact detection and removal via psf frequency analysis. In *European Conference on Computer Vision*, pages 247–262. Springer, 2014.

[19] J. Oosterhof. *Maximum Entropy and Regularised Filter deconvolution: A comparison*. University of Groningen, July 2017.

[20] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1628–1636, 2016.

[21] J.-C. Pesquet, A. Benazza-Benyahia, and C. Chaux. A sure approach for digital signal/image deconvolution problems. *IEEE Transactions on Signal Processing*, 57(12):4616–4632, 2009.

[22] H. Talbot, H. Phelippeau, M. Akil, and S. Bara. Efficient poisson denoising for photography. In *16th IEEE International Conference on Image Processing (ICIP)*, pages 3881–3884. IEEE, 2009.

[23] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.