Multi-stage image restoration in high noise and blur settings.

Sergey Voronin
Intelligent Automation, Inc., Rockville, MD, USA.

Abstract

We describe a simple approach useful for improving noisy, blurred images. Our approach is based on the use of a parallel block-based low rank factorization technique for projection based reduction of matrix dimensions and on a customized iteratively reweighted CG approach followed by the use of a Fourier Wiener filter. The regularization scheme with a transform basis offers variable residual penalty and increased per-iteration performance. The outlined approach is particularly aimed at high blur and noise settings.

Keywords: image restoration, regularization, Fourier inversion, high noise and blur.

1. Introduction

It is often necessary to enhance blurry images corrupted with noise. While many approaches have been developed for both denoising and deblurring [9, 10], most of the methods are not designed for high noise and blur settings. In this article, we outline a modular approach for the latter case. We first describe the problem setup. Mathematically, blur is typically accomplished using a convolution operation with a particular function (such as a 2D Gaussian) or multiplication with a Toeplitz matrix [3], while the noise is an additive term. We can model the process in Figure 1 and the corresponding

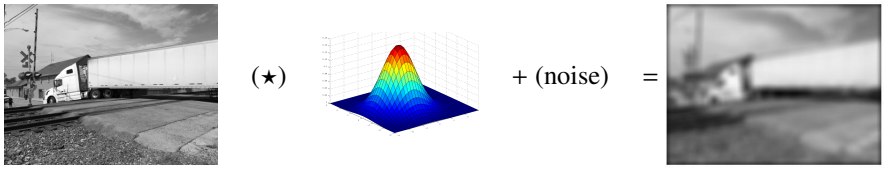

Figure 1: Image degradation process.

reconstruction as:

$$b = \bar{x} * g + n \implies F[b] = F[\bar{x}]F[g] + F[n] \implies \bar{x} = F^{-1}\left[\frac{F[b] - F[n]}{F[g]}\right], \tag{1.1}$$

with b the blurred image, $\bar{x}$ the original image, g the blur source, F the Fourier transform, and n the noise component (typically not explicitly known). The Fourier inversion procedure works well when the noise level is low. When the noise level is high, this simple approach fails as the $1/F[g]$ term blows up the noise where $F[g]$ (the optical transfer function (OSF) [7]) is small. A slight remedy is to use $1/(F[g] + \alpha^2)$ with small $\alpha \in \mathbb{R}$ (which can vary with the value of $F[g]$). A generalization of this is the Wiener filter. When the noise is more substantial, a standard procedure is to use a regularized formulation for the inverse problem:

$$\bar{w} = \arg\min_w \|RW^{-1}w - b\|_l + \lambda\|w\|_p \quad ; \quad x = W^{-1}\bar{w} \tag{1.2}$$

where W is some optional transform basis (such as a Wavelet transform, or $W = I$), R is the blurring convolution matrix, b is the blurred and noisy input and x is the resulting output image (to be compared against $\bar{x}$), obtained after the application of the inverse transform (W^{-1}). The parameters l and p control the type of penalty function applied to the residual and transformed data terms. For example, $l = 2$ and $p = 1$ is one plausible choice, if the image can be sparsely represented in the transformed domain. The blur matrix R is typically constructed in a column by column fashion from the estimated blurring source, if it is known (or estimated to be of a certain type). For example, it can be constructed by taking convolution with standard unit vectors e_j and the 2D Gaussian function (if Gaussian type blur is assumed).

```
npr = mn^2;
R = zeros(npr,npr);
for j=1:npr
    ej = zeros(npr,1);
    ej(j) = 1;
    Rj = conv2(reshape(ej,m,n),gaussian2d(5,2.0),'same');
    R(:,j) = Rj(:);
end
img_blur = R*orig_img;
```

where the width of the Gaussian controls the amount of blur. With a Toeplitz matrix construction, one may use one or two sided multiplication to obtain the blurred image with control parameter $L > 0$:

```
BR = toeplitz([ones(L,1); zeros(n-L,1)],[1;zeros(n-1,1)])/L;
BL = toeplitz([ones(L,1); zeros(m-L,1)],[1;zeros(m-1,1)])/L;
img_blur = BL*orig_img*BR;
```

An issue with (1.2) is that the dimensions of R and that of $M = RW^{-1}$ (which does not need to be explicitly formed) are $MN \times MN$ for an image of size $M \times N$. In this article we outline a combined tractable approach of (1.1) and (1.2) which is useful for the general case. The approach works by means of a projector which reduces the size of the involved matrices and couples the iteration together with Fourier based filtering.

2. Conjugate-gradient based IRLS scheme

The solution of (1.2) can be obtained via iterative thresholding techniques [1]. However, as the blur matrix R is large, such an approach is time consuming. Iterative deconvolution can be performed on blocks of an image in parallel. However, to avoid edge artifacts, the use of larger blocks is desirable. Moreover (particularly for the high noise / blur setting we consider), it is desirable to be able to impose custom penalties on the residuals and for the regularization term, which may be difficult to do with thresholding techniques. We propose the use of randomized projections with the projection matrix constructed in parallel and the use of a conjugate gradient based algorithm to speed up the iterative solve. Such a scheme can be applied to general inverse problems and not just to image enhancement applications. The iteratively reweighed least squares based method we outline is able to handle various norms on both portions of the equation (1.2), which is useful in case the noise introduces outlier pixel regions in the image.

We make use of the work from [11], where a generalized iteratively reweighted least squares (IRLS) method is developed and discuss here how to apply the scheme to large problem sizes. Consider, for a general $\bar{m} \times \bar{n}$ system $Mw \approx b$, the regularized functional defined by:

$$F_{l,p}(w) = ||Mw - b||_l^l + \lambda ||w||_p^p = \sum_{i=1}^{\bar{m}} \left| \sum_{j=1}^{\bar{n}} M_{ij} w_j - b_i \right|^l + \lambda \sum_{i=1}^{\bar{n}} |w_i|^p, \tag{2.1}$$

where l and p are assumed in the range $[1, 2]$ and control the type of penalty on the residual term and the coefficients. The IRLS approach is based on the approximation:

$$|y_k| = \frac{y_k^2}{|y_k|} = \frac{y_k^2}{\sqrt{y_k^2}} \approx \frac{y_k^2}{\sqrt{y_k^2 + \epsilon^2}},$$

where in the rightmost term, a small $\epsilon \neq 0$ is used, to insure the denominator is finite, regardless of the value of y_k. The resulting algorithm [11] for the minimization of (2.1) can be written as:

$$\left(M^T S^n M + (D^n)^T (D^n) \right) w^{n+1} = M^T S^n b \tag{2.2}$$

with two diagonal, iteration dependent matrices D^n and S^n. Optionally, a smoothing term $\lambda_2 L^T L$ can be included in the left hand side of (2.2), with L a tridiagonal $(-1, 2, -1)$ matrix. The diagonal matrix D^n has elements $\sqrt{\frac{1}{2}\lambda p y_k^n}$ and S^n has diagonal elements $l|r_i^n|^{l-2}$ (with $r_i^n = (Mw^n - b)_i$; for i where $|r_i^n| < \epsilon$, we can set the entry to $l\epsilon^{l-2}$ with the choice of ϵ user controllable, tuned for a given application). Here, the iteration dependent weights are given by:

$$y_k^n = \frac{1}{\left[(w_k^n)^2 + \epsilon_n^2 \right]^{\frac{2-p}{2}}}. \tag{2.3}$$

The diagonal matrices (or simply vectors holding the diagonal elements) are updated at each iteration and the system in (2.2) can be solved approximately via a few iterations of CG or LSQR based algorithms at each iteration $n = 0, \ldots, N$. Another advantage of the IRLS approach is that the powers p and l in (2.1) can be made component dependent. This then allows for better inversion of partially sparse signals (if of course, the location of the sparse part can be efficiently estimated). Alternatively, p, l can be adjusted as iteration progresses, particularly to incorporate the possibility of non-convex penalties.

In order to implement the scheme in (2.2), we adapt the classic CG scheme, as described in [6]. The iteration dependent factors S and D do not need to be formed explicitly and can be applied via element-wise multiplication, as shown in

Figure 3. Thresholding to remove small coefficients left over after CG iterations should be used to prevent build up errors from small values. For this task, the function below can be used, which varies the threshold based on the parameter p, with each component of w replaced with $\text{sgn}(w_i)\max(0, |w_i| - \lambda|w_i|^{p-1})$. This formulation varies the shape of the threshold with p:

```
function w2=pThreshold(w, lambda, p)
    w2 = w;
    for i=1:length(w)
        w2(i) = sign(w(i))*max(0,abs(w(i)) - lambda*abs(w(i))^(p-1));
    end
end
```

2.1 Continuation scheme

The system (2.2) is typically solved along the L-curve [4], starting at a large λ (generally a value close to $\|M^T b\|_\infty$ - the nonzero cutoff for the ℓ_1 functional) and decreasing down in logarithmic fashion using the logarithmically spaced sequence:

$$S = \frac{\log(\lambda_{\max}) - \log(\lambda_{\min})}{N-1}; \quad \lambda_i = \exp(\log(\lambda_{\max}) - S(i-1)), \quad i = 1, \ldots, N.$$

At each λ, the IRLS scheme is performed for a few iterations. Typically N is in the range of $[10, 20]$, with 5 CG-based iterations at each λ being sufficient. The use of the randomized projections and of the CG approach can significantly reduce computation demands over the whole L-curve. Typically, when a sparse promoting penalty is imposed in (2.2), we hard threshold the smallest coefficients at the end of each L-curve iteration i. One possibility is to compute some percentile (e.g. 15%) of the absolute values of w obtained from the optimization problem at each λ value and set to zero all coefficients (hard threshold) whose magnitude falls below this value.

2.2 Use of Fourier filtering

Fourier filtering is effective at mitigating blur, if the source of the blur can be accurately modeled and the noise is contained. After the end of iteration along the L-curve, or in between successive values of λ along the L-curve, we can perform a Fourier filtering approach based on the Wiener filter. Consider again the problem: $b(x, y) = g(x, y) * \bar{x}(x, y) + n(x, y)$, with b the blurred/noisy image, g the blurring source, n the noise, and $\bar{x}$ the original, all in two dimensions. Upon Fourier transforming, we obtain: $B(u, v) = G(u, v)F(u, v) + N(u, v)$. A filter can be used to estimate $\hat{F}(u, v) = W(u, v)B(u, v)$ with:

$$W(u, v) = \frac{1}{G(u, v)} \frac{|G(u, v)|^q}{|G(u, v)|^q + \alpha^q}$$

with B the Fourier transform of the corrupted image, G the optical transfer function, and $\hat{F}$ the estimate of the Fourier transform of the recovered image for some $q > 0$ and $\alpha > 0$, which can be tied to the inverse signal to noise ratio (1/SNR). We can take for this, the inverse from the expression $\max\left[\frac{1}{MN} abs(\hat{F}\hat{F}^T)\right]$ involving the Fourier transform of the recovered image [3]. Notice that by varying the eps1 value, we can control the inclusion of small values of G in the filter, even with regularization. For noisy inputs, a relatively high eps1 value can prevent the blowup of the noise components.

```
eps1 = 1e-6;
eps2 = 1e-4;
skip = 100;
G = psf2otf(g,size(g)); G = fftshift(G);
indices = find(abs(G)>eps1);
alpha = eps2;
for i=1:length(indices)
    ind = indices(i);
    if mod(i,skip) == 0
        Syy = abs(1/(M*N)*F*F');
        alpha = max(max(max(Syy)),eps2);
    end
    W(ind) = (1./abs(G(ind)))*(abs(G(ind))^q/(abs(G(ind))^q + alpha^q));
    F(ind) = W(ind)*B(ind);
end
f=abs(ifft2(F));
```

Figure 2: Fourier filtering approach with varying α value.

```
n = min(length(rhsb),max_iter);
r = rhsb - fcg_eval(params); s = r;
for numIter = 1:n
    u = fcg_eval(params);
    alpha = dot(s,r)/dot(s,u);
    w = w + alpha*s;
    r = rhsb - fcg_eval(params);
    if sqrt(dot(r,r)) < epsilon
        return
    else
        beta = -dot(r,u)/dot(s,u);
        s = r + beta*s;
    end
end

function y = fcg_eval(params)
    Ap = A*p;
    m = length(Ap);
    RAp = apply_R(Ap,res,m,lval,eps);
    AtRAp = At*RAp;

    Dp = apply_D(p, wns, lambda, pval);
    DDp = apply_D(Dp, wns, lambda, pval);
    y = AtRAp + DDp;
end
```

```
function r = apply_D(x, wns, n, lambda, p)
    r = x;
    for k=1:n
        r(k) = x(k)*0.5*lambda*p*wns(k);
    end
end

function r = apply_S(x,res,m,l,eps)
    r = x;
    for j=1:m
        rej = abs(res(j));
        if rej < eps
            rej = eps;
        end
        r(j) = x(j)*l*rej^(l-2);
    end
end

function w=pThreshold(v, tau, p)
    n = length(v);
    w = v;
    for i=1:n
        w(i) = sign(v(i))*max(0,abs(v(i))
               - tau*abs(v(i))^(p-1));
    end
end
```

Figure 3: Main steps in the iteration of the generalized CG based scheme from [11].

3. Randomized low rank projection

We now go on to discuss a fast, scalable approach to obtain a projection of the approximate $Mw \approx y$ system with $M = RW^{-1}$ and its application to the system in (2.2). In many cases, the singular values of the R matrix exhibit fast decay, with the same then holding true for M. As a result, a simple projection of the form $Q^T(Mw \approx y) \Rightarrow Q^T Mw \approx Q^T y$ reduces the problem dimension and speeds up iteration as long as Q has less columns than M. In the same way, in (2.2), M is replaced with the smaller $Q^T M$ and b with $Q^T b$, significantly reducing the problem size, without appreciable accuracy loss as long as the rank of Q is not too small. This projection though, wouldn't work with arbitrary orthogonal matrices Q, as we seek to preserve the range of the original matrix. Low rank constructions based on the QB factorization are ideal for such case. To make this approach efficient we make use of randomization based techniques. The idea behind randomization is simple to understand. A survey is found in [8] and included references. Suppose we have a matrix $M \in R^{M\times N}$ whose range we wish to sample. We can take some random vectors $v_1, \ldots, v_l$ (whose pairwise inner products will be with high probability close to 0) and compute $Mv_1, \ldots, Mv_l$. If the numerical rank of A is substantially smaller than $\min(M, N)$, then we do not need to compute many such computations in order to accurately sample the range of M. More precisely, if we form $\Omega \in R^{N\times l}$ with Gaussian iid (independent, identically distributed) entries and compute $Y = M\Omega$, then perform Gram-Schmidt on the columns of Y yielding a matrix Q. Then with large enough l relative to the numerical rank of M, we will have that range$(Q) \approx$ range(M). In this situation, it can be shown that $QQ^T M \approx M$, with the result also holding for complex-valued matrices with the transpose replaced by the Hermitian.

The scheme in Figure 4 constructs Q given M and several parameters, most notably ϵ, such that $\|QQ^T M - M\| < \epsilon$, in the same norm as used in the algorithm (the operator norm as default may be assumed). The idea behind the scheme is to draw samples of the range of M, via matrix-vector multiplications with randomly drawn vectors and then orthogonalize the resulting set of outputs using a Gram-Schmidt procedure. The algorithm shown is simply a blocked implementation of such a routine (where instead of one vector, a set of b_p vectors at a time samples the range). In place of ϵ, the block size and iteration count can be explicitly specified, if desired. The main idea behind the steps needed to expand the matrix Q can be analyzed through the following sequence of steps: Input: $Q \in \mathbb{R}^{M\times r}, y \in \mathbb{R}^M$ such that $Q^T Q = I_r$.

function $[Q, B] = \texttt{randQB\_pb}(M, \varepsilon, q, b_p)$

(1) **for** $i = 1, 2, 3, \ldots$
(2) $\quad \Omega_i = \texttt{randn}(n, b_p)$.
(3) $\quad Q_i = \texttt{orth}(M\Omega_i)$.
(4) $\quad$ **for** $j = 1 : q$
(5) $\quad\quad Q_i = \texttt{orth}(M^T Q_i)$.
(6) $\quad\quad Q_i = \texttt{orth}(MQ_i)$.
(7) $\quad$ **end for**
(8) $\quad Q_i = \texttt{orth}(Q_i - \sum_{j=1}^{i-1} Q_j Q_j^T Q_i)$
(9) $\quad B_i = Q_i^T M$
(10) $\quad M = M - Q_i B_i$
(11) $\quad$ **if** $\|M\| < \varepsilon$ **then stop**
(12) **end while**
(13) Set $Q = [Q_1 \ \cdots \ Q_i]$ and $B = [B_1^T \ \cdots \ B_i^T]^T$.

```
q=4; updateM = 1;
[Q1,B1] = randpbQB2(Mp1,q,kstep,nstep*2,updateM);
[Q2,B2] = randpbQB2(Mp2,q,kstep,nstep*2,updateM);
updateM = 0;
Bm = [B1;B2];
[Q3,B3] = randpbQB2(Bm,q,kstep,nstep,updateM);
Qm = [Q1, zeros(size(Q1,1),size(Q2,2)) ; ...
       zeros(size(Q2,1),size(Q1,2)), Q2];
Q = Qm * Q3;
B = B3;
```

Figure 4: A blocked and adaptive QB algorithm proposed in [8] along with a sample parallel call sequence.

Iteration:

$$\begin{aligned} \bar{y} &= (I - QQ^T)y \\ q &= \frac{\bar{y}}{\|\bar{y}\|} \\ \bar{Q} &= [Q, q] \end{aligned}$$

Output: $\bar{Q} \in \mathbb{R}^{M\times(r+1)}$. The output of the above procedure [8] is a matrix $\bar{Q} \in \mathbb{R}^{M\times(r+1)}$ such that:

- (a) range$(\bar{Q})$ = span(range$(Q) \cup y$).
- (b) $\bar{Q}^T \bar{Q} = I_{r+1}$.

Note: we must have $y \notin$ range(Q) as otherwise $QQ^T y = y$ and $\bar{y} = (I - QQ^T)y = 0$. (For randomly drawn y from large dimension space, this is not likely). Conditions (a) and (b) indicate that the range of Q has been expanded and orthogonality has been preserved. The blocked scheme in Figure 4 is simply an extension of the above procedure.

The parameter q controls the power iteration: $(MM^T)^q M = U\Sigma^{2q+1}V^T$, compared to the original SVD of M. Due to the power scaling term for Σ, values of $q > 1$ typically improve performance for matrices M with modest singular value decay. Lines 4-7 implement the power iteration. Line 8 is a re-orthonormalization procedure that ensures accuracy in the presence of floating point operations. Lines 9 and 10 implement the recurrence relations in line with the above range expanding procedure and line 11 checks the exit condition, for when $\|(I - Q_j Q_j^T)A\| < \epsilon$, the scheme is exited.

A simple matrix splitting strategy allows this scheme to be applied in parallel over parts of the matrix, which is important for matrices which have large dimensions, such as the case we consider. We can proceed by subdividing M into blocks along the rows. We assume here that the number of blocks is a power of two. Without loss of generality, we assume the use of four blocks:

$$M = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} \approx \begin{bmatrix} Q_1 B_1 \\ Q_2 B_2 \\ Q_3 B_3 \\ Q_4 B_4 \end{bmatrix} = \begin{bmatrix} Q_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & Q_3 & 0 \\ 0 & 0 & 0 & Q_4 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix}$$

We then perform QB factorizations on the blocks of the B matrix:

$$M^{(1)} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \approx Q_{12} B_{12} \quad ; \quad M^{(2)} = \begin{bmatrix} B_3 \\ B_4 \end{bmatrix} \approx Q_{34} B_{34}$$

Finally, we perform a QB factorization on:

$$M^{(3)} = \begin{bmatrix} B_{12} \\ B_{34} \end{bmatrix} \approx Q_{1234} B_{1234}$$

It follows that:

$$M \approx \begin{bmatrix} Q_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & Q_3 & 0 \\ 0 & 0 & 0 & Q_4 \end{bmatrix} \begin{bmatrix} Q_{12} & 0 \\ 0 & Q_{34} \end{bmatrix} \begin{bmatrix} B_{12} \\ B_{34} \end{bmatrix} \approx \begin{bmatrix} Q_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & Q_3 & 0 \\ 0 & 0 & 0 & Q_4 \end{bmatrix} \begin{bmatrix} Q_{12} & 0 \\ 0 & Q_{34} \end{bmatrix} Q_{1234} B_{1234}$$
$$= Q^{(3)} Q^{(2)} Q^{(1)} B^{(1)} = QB$$

The benefit of this formulation is that the QB algorithm can be performed on smaller matrices in parallel. In particular, we handle the decompositions of blocks $M_1, \ldots, M_4$ in parallel, following which we can do in parallel the decompositions of matrices $M^{(1)}$ and $M^{(2)}$ and finally that of $M^{(3)}$. The reason for blocking M along the rows instead of columns is that we would like to keep the orthonormality of the resulting matrix Q which is done by working with block diagonal matrices. Notice that performing QB factorizations at later steps is not essential. Instead, we can replace QB with the full QR factorization computed on the smaller matrices (which would be efficient to do). The matrix update (line 10) can in practice be turned off for the later stage to reduce runtime (see Fig. 4, right). The ranks chosen at the initial (first order) steps should be made to insure that $Q^T M$ is of significantly smaller dimensions than M.

Let us consider the hierarchical factorization approach with explicit error accounting for two blocks:

$$M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} Q_1^{(1)} B_1^{(1)} + E_1^{(1)} \\ Q_2^{(1)} B_2^{(1)} + E_2^{(1)} \end{bmatrix} = \begin{bmatrix} Q_1^{(1)} & 0 \\ 0 & Q_2^{(1)} \end{bmatrix} \begin{bmatrix} B_1^{(1)} \\ B_2^{(1)} \end{bmatrix} + \begin{bmatrix} E_1^{(1)} \\ E_2^{(1)} \end{bmatrix} = \begin{bmatrix} Q_1^{(1)} & 0 \\ 0 & Q_2^{(1)} \end{bmatrix} \begin{bmatrix} Q_1^{(2)} B_1^{(2)} + E_1^{(2)} \\ Q_2^{(2)} B_2^{(2)} + E_2^{(2)} \end{bmatrix} + \begin{bmatrix} E_1^{(1)} \\ E_2^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} Q_1^{(1)} & 0 \\ 0 & Q_2^{(1)} \end{bmatrix} \left[\begin{bmatrix} Q_1^{(2)} & 0 \\ 0 & Q_2^{(2)} \end{bmatrix} \begin{bmatrix} B_1^{(2)} \\ B_2^{(2)} \end{bmatrix} + \begin{bmatrix} E_1^{(2)} \\ E_2^{(2)} \end{bmatrix} \right] + \begin{bmatrix} E_1^{(1)} \\ E_2^{(1)} \end{bmatrix}$$

Then in particular, collapsing the row matrices, it follows that:

$$M = Q^{(1)}(Q^{(2)} B^{(2)} + E^{(2)}) + E^{(1)} = Q^{(1,2)} B^{(2)} + Q^{(1)} E^{(2)} + E^{(1)}$$

Since the Q^i's are orthogonal, the overall error bound is additive, i.e. $\|M - QB\| \leq \|E^{(1)}\| + \|E^{(2)}\|$. This is useful to know in a distributive environment, as the errors at each iteration of different blocks can be communicated to e.g. the root processor, which can make a decision about the rank to use for each block so that a final output tolerance is met.

4. Combined approach

We now summarize the combined image restoration approach we propose for high noise and blur settings. We again note that this approach is especially formulated for the high noise and blur case and is not meant to be competitive to state of art schemes for low corruption regimes. The approach consists of a combination of the generalized IRLS approach performed with different basis projections. E.g. we can make use of sharp (e.g. Haar) and smoother (e.g. CDF-97) Wavelets or Fourier basis. We make use of the ability to apply high penalty on the residuals and follow though with adaptive interpolation (described below) and then overall enhancement with Fourier based filtering (Fig. 2). The interpolation step simply combines two images produced from regularization with the sharp and smooth basis into one image. For example, if im1.png corresponds to the image produced with the sharp basis and im2.png to that with the smooth basis, the linear interpolation produces results which are closer to the smooth version.

```
im1 = double(imread('im1.png'));
im2 = double(imread('im2.png'));
imc = cat(3,im1,im2);
imc = permute(imc,[3 1 2]);
im_res = squeeze(interp1([0,1],imc,.8));
```

The combination can be performed adaptively, by computing a gradient map of one of the images, subdividing in blocks, and combining smooth and sharp results with respect to the observed smoothness (weighing more heavier towards the smooth version for smooth regions). Of course, more than two bases can be used and merged in a similar way. Alternatively, more advanced merging techniques can be employed. One particular class of methods is based on the use of the 2-D wavelet transform to decompose the image into high frequency and low frequency portions. The high frequency and low frequency components are then merged according to different rules [5, 3]. In Matlab, the commands *wavedec2,detcoef2,wcodemat* can be utilized to extract the approximation (low frequency) and detail (higher frequency portions) of an image under a specified transform. Alternatively, superresolution based methods [2] can also be employed for the merge step.

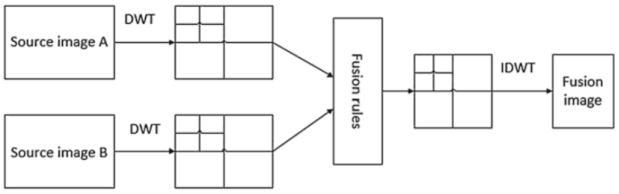

Figure 5: Fusion scheme diagram for fusing multiple images (e.g. obtained with iterative regularization using different projection bases) into one [3].

Algorithm 1: Inverse problem/ Fourier, high noise/blur image enhancement approach.

Data: Corrupted image, assumed blur matrix and transform basis routines (W_1, W_2), parameters for Q^T projection and for L-curve traversion.
Result: recovered image
Obtain projectors Q_1, Q_2 for the matrices $M_i = RW_i^{-1}$ via blocking by rows and perform block QB procedure to obtain projected Q's for M_1 and M_2.
while *not reached L-curve end* **do**
  Perform generalized IRLS inversion with smaller, projected $Q^T M_i$ matrices, with $p, q \approx 1$, to obtain regularized solutions w_1, w_2.
  if *tolerance on residuals not reached* **then**
    move to next step of L-curve
  else
    Threshold w_1, w_2 below set percentile and compute $x_i = W_i^{-1} w_i$ for $i = 1, 2$.
  end
end
Merge x_1, x_2 using the adaptive interpolation procedure described or 2-D Wavelet based fusion scheme with separate merge expressions for LF and HF portions.
Perform follow-on Wiener-based Fourier filter step.

5. Numerical results

When the blur source can be accurately modeled and no significant noise is present, the regularized Fourier inversion procedure (i.e. Wiener filter) is sufficient to be used on its own. An example is shown in Figure 6. When the noise level is high (or when special artifacts like scan lines may be present), optimization based inversion yields better results. In Figure 7, we illustrate results of the described generalized IRLS CG approach we have outlined which we use together with the randomized projection approach to decrease the matrix size and make the problem tractable on smaller machines. In particular, the input image is of size 125×125. The resulting blur matrix R which we construct from a convolution operation with a Gaussian is of dimensions 15625×15625. We take Q of rank 1200 such that the resulting projection $Q^T M$ (the projected blur matrix multiplied by the inverse of the CDF-9/7 Wavelet basis) is of reduced size 1200×15625, so that the inversion can be performed without issues on a laptop with 8 GB memory. The same machine is not able to iterate the big, un-projected system. The rank (or tolerance level) to use, depends on the singular value decay of the blur matrix, which is directly proportional to the amount of noise and blur in the image. We ran over 15 steps of the L-curve, for 10 iterations per step with 10 iterations of the CG algorithm for each IRLS iteration. Utilizing the generalized IRLS scheme with $l, p = 1.1$, we were able to partially mitigate the scan line artifacts.

Next, we illustrate the results of the combined Fourier filtering and regularization based approach, made possible via the cost reduction enabled via the randomized block based projection we have described. We consider the image of a horse pictured below (grayscale PNG, 150×129), which we corrupt with blur noise and scan line artifacts. As shown in Figure 9, we divide the image into two parts (sized 75×129) for parallel processing. We run iterative denoising / deblurring using the IRLS CG algorithm with a QB projection for speed up, using two transform bases: CDF 97 wavelets and the Hartley transform. The resulting blur matrices for each part are 2450×9675, reduced via the Q^T projection from size 9675×9675. The two step approach can noticeably improve results of strongly corrupted images. We can utilize the SSI measure [12] to quantify performance improvements over the unperturbed images in Figure 8. Figure 9 illustrates that the image quality for the parts improves upon inversion with the generalized IRLS scheme, the interpolated merge step, and the use of the Wiener filter on the merged images.

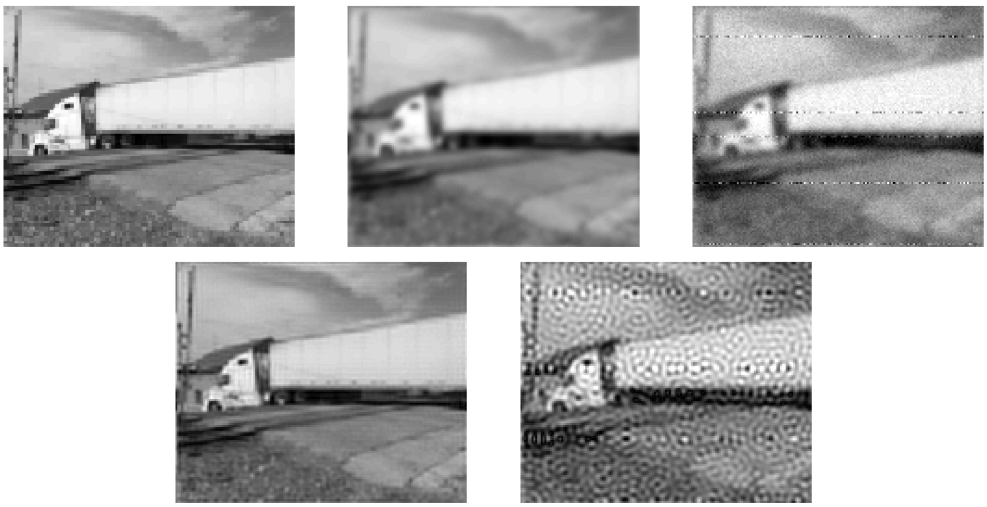

Figure 6: Row 1: Successive image degradation with blur (image 2) and noise addition (image 3). Row 2: Fourier Wiener filter deblur with OSF derived from a Gaussian function applied to images 2 and 3.

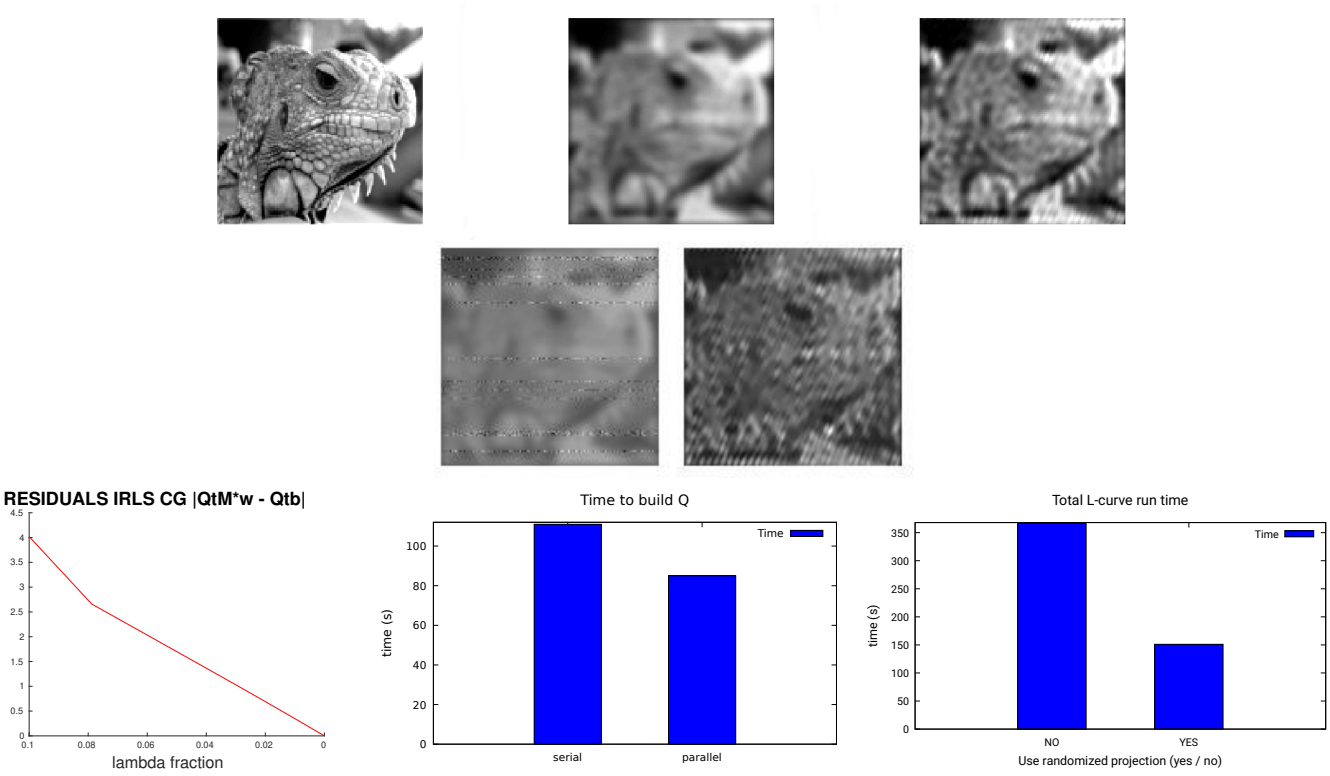

Figure 7: IRLS CG based inversion results. Row 1: original, blurred/noisy, recovered. Row 2: blurred/noisy with additional artifacts and recovered image. Row 3: Residual values versus regularization parameter and runtimes to build projector Q with serial and hierarchical schemes and times to complete L-curve with and without projection.

References

[1] Jalal M Fadili and Jean-Luc Starck. Sparse representation-based image deconvolution by iterative thresholding. In *Astronomical Data Analysis ADA'06*, 2006.

[2] Sina Farsiu. Mdsp resolution enhancement software user's manual. *University of California, Santa Cruz*, 2004.

[3] Shengrong Gong. *Advanced Image and Video Processing Using MATLAB*. Springer, 2018.

Figure 8: Original image and corrupted image (blur, noise, scan line artifacts), separated into two parts.

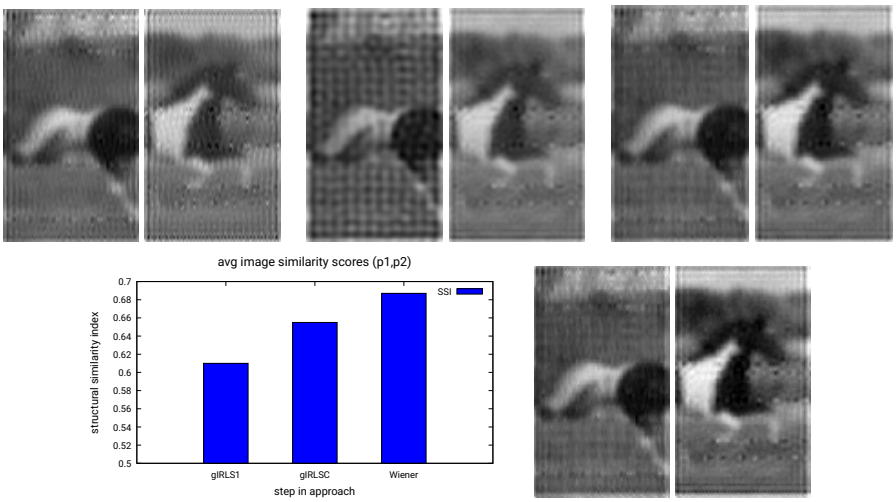

Figure 9: IRLS CG improved images (bases 1 and 2), merged images, and Fourier filtering results on the inverted images (bottom row). SSI image similarity metric values with respect to the unperturbed parts for different stages.

[4] Per Christian Hansen. *The L-curve and its use in the numerical treatment of inverse problems*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1999.

[5] Muwei Jian, Junyu Dong, and Yang Zhang. Image fusion based on wavelet transform. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 1, pages 713–718. IEEE, 2007.

[6] Carl T Kelley. *Iterative methods for optimization*, volume 18. Siam, 1999.

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

[8] Per-Gunnar Martinsson and Sergey Voronin. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. *SIAM Journal on Scientific Computing*, 38(5):S485–S507, 2016.

[9] Minu Poulose et al. Literature survey on image deblurring techniques. *International Journal of Computer Applications Technology and Research*, 2(3):286–288, 2013.

[10] RC Puetter, TR Gosnell, and Amos Yahil. Digital image reconstruction: Deblurring and denoising. *Annual review of Astronomy and Astrophysics*, 43, 2005.

[11] Sergey Voronin, Christophe Zaroli, and Naresh P Cuntoor. Conjugate gradient based acceleration for inverse problems. *GEM-International Journal on Geomathematics*, 8(2):219–239, 2017.

[12] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.