Two-Blur Blind Valid Image Deconvolution by Linear Algebra

Andrew E. Yagle
Dept. of EECS, The University of Michigan, Ann Arbor, MI 48109-2122
aey@eecs.umich.edu

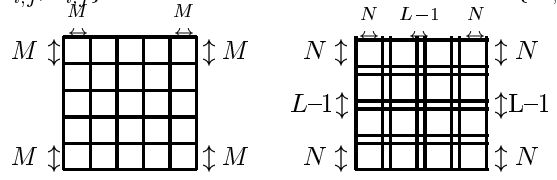
April 9, 2021

Abstract

The 2-D blind deconvolution problem is to reconstruct an unknown image from its known 2-D convolution with an unknown point-spread function (PSF) (2-D spatial impulse response). If both the unknown image and unknown PSF have known finite supports (finite nonzero regions), then the problem is overdetermined and has a unique solution to a scale factor. We address this problem in a separate paper. However, if only the PSF has known finite support, then the 2-D convolution is called a *valid* 2-D convolution, and the problem is underdetermined. We show *two* unknown PSFs can be uniquely determined from their *valid* 2-D convolutions with an unknown image, which cannot be determined without additional information (sparsity in a wavelet basis).

1 Problem Statement

The two unknown PSFs are each $(L \times L)$ functions $\{h^1_{i,j}, h^2_{i,j}, 0 \le i,j \le L\text{–}1\}$. The portion of unknown image is the $(M^2 \times M^2)$ region $\{x_{i,j}, 1 \le i.j \le M^2\}$, ($M^4$ pixels) which is a subset of a larger image. We partition the region into M^2 $(M \times M)$ subregions $\{x^k_{i,j},\ {}^{1\le i,j\le M}_{1\le k\le M^2}\}$ (Fig.). The goal is to compute $\{h^1_{i,j}, h^2_{i,j}\}$ from their valid convolutions with $\{x_{i,j}\}$.



1.1 2-D Convolution

The *linear* (usual) 2-D convolution of $h_{i,j}$ and $x_{i,j}$ is

$$y_{i,j} = \sum_{m=1}^{M}\sum_{n=1}^{M} h_{i-m,j-n}x_{m,n}, 1 \le i,j \le M+L-1 \tag{1}$$

which is an $(M+L-1)\times(M+L-1)$ image.

The *valid* 2-D convolution of $h_{i,j}$ and $x_{i,j}$ is

$$y_{i,j} = \sum_{m=1}^{M}\sum_{n=1}^{M} h_{i-m,j-n}x_{m,n}, L \le i,j \le M \tag{2}$$

which is an $(M-L+1)\times(M-L+1)$ image.

As a small but illustrative example, consider

$$x_{i,j} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}; \quad h_{i,j} = \begin{bmatrix} 11 & 12 \\ 13 & 14 \end{bmatrix}. \tag{3}$$

The *linear* convolution of $h_{i,j}$ and $x_{i,j}$ is

$$y_{i,j} = \begin{bmatrix} 11 & 34 & 57 & 36 \\ 57 & 143 & 193 & 114 \\ 129 & 293 & 343 & 192 \\ 91 & 202 & 229 & 126 \end{bmatrix}. \tag{4}$$

The *valid* convolution of $h_{i,j}$ and $x_{i,j}$ is

$$y_{i,j} = \begin{bmatrix} 143 & 193 \\ 293 & 343 \end{bmatrix}. \tag{5}$$

The valid convolution is the central part of the linear convolution, obtained by deleting the edge rows and columns from the linear convolution.

2 1-D Problem

To clarify the procedure, and since the problem is interesting in its own right, we first present the 1-D version of the procedure. The goal is to compute the unknown impulse responses $h^1[n]$ and $h^2[n]$, each having length L, and a length M^2 portion $x[n]$ of the unknown 1-D signal from the *valid* 1-D convolutions

$$y^m[n] = \sum_{n'=1}^{M^2} h^m[n-n']x[n'], L \le n \le M^2. \quad (6)$$

where m=1,2. Since only parts of the linear convolutions are known, this precludes use of z-transforms or the polynomial greatest common divisor.

The procedure for the 1-D problem is as follows.

First, divide up the length-M^2 signal $x[n]$ into M subsignals $\{x_i[n], i = 1 \dots M\}$ of length M, where

$$x_i[n] = x[n], (i-1)M+1 \le n \le iM, 1 \le i \le M. \quad (7)$$

Next. since all of the 1-D convolutions between $h^m[n]$ and $x_j[n]$ are valid, they can all be determined from the overall 1-D valid convolution between $h^m[n]$ and $x[n]$ (the given data) by discarding segments of $y^1[n]$ and $y^2[n]$ of lengths L–1. For example, let

$$x[n] = \{1,2,3,4,5,6,7,8,9,10\}; h[n] = \{1,2\}. \quad (8)$$

Their valid convolution has length 10–2+1=9 and is

$$y[n] = \{4,7,10,13,16,19,22.25,28\}. \quad (9)$$

Now partition $x[n]$ into two subsignals

$$x_1[n] = \{1,2,3,4,5\}; \quad x_2[n] = \{6,7,8,9,10\}. \quad (10)$$

The valid convolutions of $h[n]$ with $x_1[n], x_2[n]$ are

$$y_1[n] = \{4,7,10,13\}; y_2[n] = \{19,22,25,28\}. \quad (11)$$

Next, note that the valid 1-D convolution between $h[n]$ and $x[n]$ can be implemented as the Toeplitz matrix times column vector product

$$\begin{bmatrix} h_1[L-1] & h[L-2] & \cdots & h[1] & h[0] \\ 0 & h[L-1] & h[L-2] & \cdots & h[1] \\ 0 & 0 & h[L-1] & h[L-2] & \cdots \\ 0 & \cdots & 0 & h[L-1] & h[L-2] \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 \\ h[0] & \cdots & 0 \\ h[1] & h[0] & 0 \\ \cdots & h[1] & h[0] \end{bmatrix} \begin{bmatrix} x[1] \\ x[2] \\ \vdots \\ x[M] \end{bmatrix} = \begin{bmatrix} y[L] \\ y[L+1] \\ \vdots \\ y[M] \end{bmatrix}. \quad (12)$$

- The Toeplitz matrix is $N \times M$;
- the x-vector has length M;
- the y-vector has length N, where

$$N = M - L + 1 \quad (13)$$

is the length of the valid 1-D convolution.

Next, note that the valid convolutions of $h^m[n]$ and $x_j[n]$ can be combined into the matrix equation

$$\begin{bmatrix} \mathbf{Y_1} \\ \mathbf{Y_2} \end{bmatrix} = \begin{bmatrix} \mathbf{H_1} \\ \mathbf{H_2} \end{bmatrix} [\mathbf{X}]. \quad (14)$$

where the matrices $\mathbf{Y_1}$ and $\mathbf{Y_2}$ are defined as

$$\mathbf{y_j^1} = [y^1[1] \dots y^1[N]]^T; \mathbf{Y_1} = [\mathbf{y_1^1}| \dots |\mathbf{y_M^1}]. \quad (15)$$

$$\mathbf{y_j^2} = [y^2[1] \dots y^2[N]]^T; \mathbf{Y_2} = [\mathbf{y_1^2}| \dots |\mathbf{y_M^2}]. \quad (16)$$

$$\mathbf{x_j} = [x_j[1] \dots x_j[M]]^T; [\mathbf{X}] = [\mathbf{x_1}| \dots |\mathbf{x_M}]. \quad (17)$$

$\mathbf{H_1}$ and $\mathbf{H_2}$ are Toeplitz matrices like the one in (12).

Finally, note the matrices $\begin{bmatrix} \mathbf{Y_1} \\ \mathbf{Y_2} \end{bmatrix}$ and $\begin{bmatrix} \mathbf{H_1} \\ \mathbf{H_2} \end{bmatrix}$ are both $(2N \times M)$. They are 'tall' (#rows=#columns+1) if

$$2N = 2(M-L+1) = M+1 \rightarrow M = 2L-1 \quad (18)$$

in which case both matrices have a left null vector. Assuming the matrix $[\mathbf{X}]$ has full rank, the left null vector of $\begin{bmatrix} \mathbf{Y_1} \\ \mathbf{Y_2} \end{bmatrix}$ can be computed, and it will determine the elements of $\begin{bmatrix} \mathbf{H_1} \\ \mathbf{H_2} \end{bmatrix}$. The 1^{st} half of the left null vector will be $-h^2[L-1-n]$ and the 2^{nd} half of the null vector will be $h^1[L-1-n]$, to a scale factor, which is an unavoidable ambiguity in blind deconvolution. This requires the portion of the original signal have length $M^2 = (2L-1)^2$ according to (18) in order to form the matrix $[\mathbf{X}]$, which is $(M \times M)$.

It also means the original signal can be segmented into segments having lengths M^2 each, with each segment having a different impulse response. This amounts to a spatially-varying impulse response.

3 1-D Problem Example

An unknown signal of length 9 is validly convolved with two unknown impulse responses of lengths two each. The results of the two valid convolutions are

$$\begin{aligned} y_1[n] &= \{7,6,9,7,19,20,10,17\}; \\ y_2[n] &= \{15,16,19,19,47,42,26,39\}. \end{aligned} \tag{19}$$

Compute the two impulse responses and the signal.

We have L=2 and M=3 since the signal has length M^2=9.

Delete every 3^{rd} signal segment of length L–1=1:

$$\begin{aligned} y_1[n] &= \{7,6,7,19,10,17\}; \\ y_2[n] &= \{15,16,19,47,26,39\}, \end{aligned} \tag{20}$$

and rearranging the remaining elements gives

$$\mathbf{Y_1} = \begin{bmatrix} 7 & 7 & 10 \\ 6 & 19 & 17 \end{bmatrix}; \quad \mathbf{Y_2} = \begin{bmatrix} 15 & 19 & 26 \\ 16 & 47 & 39 \end{bmatrix}. \tag{21}$$

The left null vector of $\left[\begin{smallmatrix}\mathbf{Y_1}\\ \mathbf{Y_2}\end{smallmatrix}\right]$ is [–4,–3,2,1] after dividing by the last element of the row vector (this is the scale factor). The sign of the first impulse response is due to the necessity of the product of the left null vector and $\left[\begin{smallmatrix}\mathbf{H_1}\\ \mathbf{H_2}\end{smallmatrix}\right]$ to be zero. From the structure of the Toeplitz matrix in (12) the impulse responses are

$$h_1[n] = \{1,2\}; \quad h_2[n] = \{3,4\}. \tag{22}$$

Solving the two-Toeplitz system

$$\begin{bmatrix} \mathbf{Y_1} \\ \mathbf{Y_2} \end{bmatrix} = \begin{bmatrix} \mathbf{H_1} \\ \mathbf{H_2} \end{bmatrix} [\mathbf{X}] \tag{23}$$

and unwrapping [**X**] yields (see Appendix A)

$$x[n] = \{3,1,4,1,5,9,2,6,5\}. \tag{24}$$

These are the correct answers (after normalization).

4 2-D Problem

Now we consider the 2-D version of the problem.

The changes from the 1-D version are as follows.

- The unknown image $x_{i,j}$ is an $(M^2 \times M^2)$ subset of a larger image, which we term the superimage;
- The unknown PSFs are $(L \times L)$ $h^1_{i,j}$ and $h^2_{i,j}$;
- The known valid 2-D convolutions of $x_{i,j}$ with $h^1_{i,j}$ and $h^2_{i,j}$ are two $(M^2-L+1)\times(M^2-L+1)$ blurred images $y^1_{i,j}$ and $y^2_{i,j}$;
- The $(M^2 \times M^2)$ image is partitioned into M^2 $(M \times M)$ subimages $\{x^k_{i,j}\}$ k=1... M^2 (see Fig.);
- Before partitioning $y^1_{i,j}$ and $y^2_{i,j}$, bands of rows and columns of width L–1 are deleted (see Fig.);
- Each $(M^2-L+1)\times(M^2-L+1)$ blurred image $y^m_{i,j}, m = 1,2$ is partitioned into M^2 $(N \times N)$ subimages $\{y^{1,k}_{i,j}, y^{2,k}_{i,j}, 1 \le k \le M^2\}$ (see Fig.);
- Each subimage $y^{1,k}_{i,j}$ and $y^{2,k}_{i,j}$ is the valid 2-D convolution of subimage $x^k_{i,j}$ with $h^1_{i,j}$ and $h^2_{i,j}$;
- The matrix [**X**] is now $(M^2 \times M^2)$, and the k^{th} column consists of $x^k_{i,j}$ unwrapped by columns;
- Matrices $\mathbf{H_1}$ and $\mathbf{H_2}$ are now block Toeplitz with Toeplitz blocks (TBT); a matrix-vector product implements valid 2-D convolution;
- (18) becomes $2N^2$=$2(M$–L+1$)^2 > M^2$.

The following table lists the minimum value of M associated with each value of L. Recall N=M–L+1.

L	M	$2N^2$	M^2
2	4	18	16
3	7	50	49
4	11	128	121
5	14	200	196

If the #rows exceeds the #columns by more than one, there are multiple null vectors. In this case, the null vector times the TBT $\left[\begin{smallmatrix}\mathbf{H_1}\\ \mathbf{H_2}\end{smallmatrix}\right]$ must be rearranged into a TBT system, consisting of the elements of the null vectors, for the elements of $h^1_{i,j}$ and $h^2_{i,j}$.

5 2-D Problem Example

This small but illustrative example illustrates the following aspects of the procedure:

- The ability to handle a spatially-varying PSF in different parts of the superimage;
- The divide-and-conquer ability due to the use of valid 2-D convoluions throughout;
- The need to use sparsity of the image in a wavelet basis to compute it from $h^1_{i,j}$ and $h^2_{i,j}$.

The example is as follows:

1. A (48×48) superimage was created by downsampling a (200×200) clown image by (4×4), after clipping it to (192×192);
2. This (48×48) superimage was in turn partitioned into $(\frac{48}{16})^2$=9 (16×16) images. The two PSFs were spatially-invariant within each image, but varied over the superimage;
3. The 9 pairs of (2×2) PSFs, one pair for each image, were composed of the digits of π;
4. Each (16×16) image was in turn partitioned into $(\frac{16}{4})^2$=16 (4×4) subimages;
5. Each (4×4) subimage was validly convolved with a pair of (2×2) PSFs, resulting in a pair of (3×3) blurred subimages, where 4–2+1=3;
6. Unwrapping the 16 (4×4) subimages by columns resulted in a (16×16) matrix $[\mathbf{X}]$;
7. Unwrapping the 16 (3×3) blurred subimages by columns resulted in a (9×16) matrix. Stacking the two (9×16) (one for each of the two PSFs) matrices resulted in a (18×16) matrix $\left[\begin{smallmatrix}\mathbf{Y^1}\\ \mathbf{Y_2}\end{smallmatrix}\right]$;
8. The matrices $[\mathbf{X}]$ and $\left[\begin{smallmatrix}\mathbf{Y^1}\\ \mathbf{Y_2}\end{smallmatrix}\right]$ are related by

$$\begin{bmatrix}\mathbf{Y^1}\\ \mathbf{Y_2}\end{bmatrix} = \begin{bmatrix}\mathbf{H^1}\\ \mathbf{H_2}\end{bmatrix} [\mathbf{X}] \tag{25}$$

   where $[\mathbf{H^1}]$ and $[\mathbf{H^2}]$ are TBT matrices that implement valid 2-D convolutionw;
9. Despite $\left[\begin{smallmatrix}\mathbf{H^1}\\ \mathbf{H_2}\end{smallmatrix}\right]$ being a "tall" (18×16) matrix, it only has rank 14. So [25] is actually underdetermined, so even when the two PSFs are determined the image cannot be computed without additional information. This is discussed in detail in [1], where three PSFs are shown to be necessary to determine the image;
10. We use the sparsity of the image in the Haar wavelet transform as the additional information. We had planned to use iterative reweighted least squares (the FOCUSS algorithm) to minimize the ℓ_1 norm of the image, but simple least-squares gave visually good results (see below).

To compute the 2-D Haar wavelet transform, let

$$W = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{26}$$

Then the 2-D Haar wavelet transform of image X is

$$WXW^T = (W \otimes W)X(:) = (WW)X(:) \tag{27}$$

$$\text{and} \quad Y = HX \rightarrow Y = H(WW)^T(WW). \tag{28}$$

Finally, we note that if the image is known to have finite support then a single blurred image suffices to determine the image. A method is presented in [2].

6 References

1. G. Harikumar and Y. Bresler, "Exact Image Deconvolution from Multiple FIR Filters," *IEEE Trans. Image Processing* 8(6), 846-862, 1999.

2. A.E. Yagle, "2-D Blind Deconvolution for Compactly-Supported Images by Finding a Null vector of a Toeplitz-Block-Toeplitz Linear System of Equations," April 2021.

7 Appendix A

This Matlab program implements the 1-D procedure.

```
clear
X=[3,1,4,1,5,9,2,6,5];H1=[1 2];H2=[3 4];
L=length(H1);M=sqrt(length(X));
Y1=conv(X,H1);Y2=conv(X,H2);%length=M*M+L-1.
Y1V=Y1(L:M*M);Y2V=Y2(L:M*M);%length=M*M-L+1.
%GOAL:Compute H1 and H2 from Y1V and Y2V.
%Delete elements of Y1V and Y2V.
IY=[];for I=0:M-1;
IY=[IY M*I+[1:M-L+1]];end;
Y1V=Y1V(IY);Y2V=Y2V(IY);
%Form matrices Y21 and Y22.
Y21=reshape(Y1V,M-L+1,M);
Y22=reshape(Y2V,M-L+1,M);
%Find left null vector.
YY=[Y21;Y22];N=null(YY');N=N/N(2*L);
H1EST=N(2*L:-1:L+1);H2EST=-N(L:-1:1);
H1,H1EST,H2,H2EST
%Compute X2 from [Y21;Y22] and [H21;H22].
H21=toeplitz([H1(L);zeros(M-L,1)],...
[fliplr(H1) zeros(1,M-L)]);
H22=toeplitz([H2(L);zeros(M-L,1)],...
[fliplr(H2) zeros(1,M-L)]);
HH=[H21;H22];XEST=pinv(HH)*YY,
X2=reshape(X,M,M)
```

8 Appendix B

This Matlab program implements the 2-D procedure.

```
clear;M=11;L=4;
%Valid Blind Deconvolution From 2 Blurs
%Image is (M*M)X(M*M). PSFs are both LXL.
M=11;X=rand(M*M,M*M);
L=4;H1=rand(L,L)';H2=rand(L,L)';
Y1V=conv2(X,H1,'valid');N=M-L+1;
Y2V=conv2(X,H2,'valid');
%GOAL: Compute H1 & H2 from Y1V & Y2V.
%Delete rows and columnd of Y1V & Y2V.
II=[];for I=0:M-1;II=[II [1:N]+M*I];end;
Y1V=Y1V(II,II);Y2V=Y2V(II,II);
%Partition Y1V & Y2V and unwrap.
for I=1:M;for J=1:M;
Y21(1:N,1:N,I,J)=Y1V([1:N]+N*(I-1),[1:N]+N*(J-1));
Y22(1:N,1:N,I,J)=Y2V([1:N]+N*(I-1),[1:N]+N*(J-1));
X2(1:M,1:M,I,J)=X([1:M]+M*(I-1),[1:M]+M*(J-1));
end;end;
YY1=[];YY2=[];XX=[];
for I=1:M;for J=1:M;
YY1=[YY1 reshape(Y21(1:N,1:N,I,J),N*N,1)];
YY2=[YY2 reshape(Y22(1:N,1:N,I,J),N*N,1)];
XX=[XX reshape(X2(1:M,1:M,I,J),M*M,1)];
end;end;
%Find left null vector.
NN=null([YY2;YY1]');
N1=NN(1:N*N,1);N2=NN(N*N+1:2*N*N,1);
for I=1:N;
NN1(:,:,I)=toeplitz([N1(N*(I-1)+1);zeros(L-1,1)],...
[N1(N*(I-1)+1:N*(I-1)+N)' zeros(1,L-1)]);
NN2(:,:,I)=toeplitz([N2(N*(I-1)+1);zeros(L-1,1)],...
[N2(N*(I-1)+1:N*(I-1)+N)',zeros(1,L-1)]);end;
%Assemble TBT matrix from null vector.
A1=zeros(L,M);A2=zeros(L,M);
for I=0:L-1;for J=I+1:N;
A1(I*L+[1:L],J*M+[1:M])=NN1(:,:,J-I);
A2(I*L+[1:L],J*M+[1:M])=NN2(:,:,J-I);
end;end;
HEST=null([A1;A2]');
%Reverse and scale estimates of PSFs.
HEST=flipud(HEST)/HEST(L*L)*H2(1,1);
reshape(HEST(1:L*L),L,L),H1
reshape(HEST(L*L+1:2*L*L),L,L),H2
```

9 Appendix C

This Matlab program implements the 2-D rxample.

```
%Valid 2-D 2-blur blind deconvolution of clown.
%image with 2 spatially-varying PSFs.
clear;load clown.mat;%200X200.Downsample to 48X48.
FX=fft2(X);FX=FX;FX(25:202-25,25:202-25)=0;
X=real(ifft2(FX));X=X(4:4:193,4:4:193);
%Segment 48X48 superimage X into 9 16X16 images X2.
for K1=1:3;K11=[1:16]+(K1-1)*16;
for K2=1:3;K21=[1:16]+(K2-1)*16;
K3=(K1-1)+(K2-1)*3+1;
X2(1:16,1:16,K3)=X(K11,K21);end;end;
%Segment each 16X16 image X2 into 16 4X4 images X3.
for K=1:9;for K1=1:4;K11=[1:4]+(K1-1)*4;
for K2=1:4;K21=[1:4]+(K2-1)*4;K3=(K1-1)+(K2-1)*4+1;
X3(1:4,1:4,K3,K)=X2(K11,K21,K);end;end;end;
%Segment PSFs H,G into spatially-varying PSFs H2,G2.
H=[3 1 4 1 5 9;2 6 5 3 5 8;9 7 9 3 2 3;8 4 6 2 6 4];
H=[H;3 3 8 3 2 7;9 5 0 2 8 8];
G=[4 1 9 7 1 6;9 3 9 9 3 7;5 1 1 5 8 2;0 9 7 4 9 4];
G=[G;4 5 9 2 3 0;7 8 1 6 4 0];
for K1=1:3;K11=[1:2]+2*(K1-1);
for K2=1:3;K21=[1:2]+2*(K2-1);K3=(K1-1)+(K2-1)*3+1;
H2(1:2,1:2,K3)=H(K11,K21);
G2(1:2,1:2,K3)=G(K11,K21);end;end;
%Valid 2D convs of images w/ spatially-varying PSFs.
%Want to do this:  %for K=1:9;
%Y(1:15,1:15,K)=conv2(X2(:,:,K),H2(:,:,K),'valid');
%end;%but gives wrong answer!  But this does work:
%Y1=conv2(X2(:,:,7),H2(:,:,7),'valid');Y1(13:15,9:12)
%No idea why this is.  So implement using matrices.
HZ=zeros(3,4);
for K=1:9;H2(1:2,1:2,K)=H2(1:2,1:2,K)';
H11(1:3,1:4,K)=[H2(2,1,K) H2(1,1,K) 0 0;...
0 H2(2,1,K) H2(1,1,K) 0;0 0 H2(2,1,K) H2(1,1,K)];
H12(1:3,1:4,K)=[H2(2,2,K) H2(1,2,K) 0 0;0 ...
H2(2,2,K) H2(1,2,K) 0;0 0 H2(2,2,K) H2(1,2,K)];
HH1(1:9,1:16,K)=[H12(:,:,K) H11(:,:,K) HZ HZ;HZ ...
H12(:,:,K) H11(:,:,K) HZ;HZ HZ H12(:,:,K) H11(:,:,K)];
G2(1:2,1:2,K)=G2(1:2,1:2,K)';%2nd PSF matrices.
G11(1:3,1:4,K)=[G2(2,1,K) G2(1,1,K) 0 0;...
0 G2(2,1,K) G2(1,1,K) 0;0 0 G2(2,1,K) G2(1,1,K)];
G12(1:3,1:4,K)=[G2(2,2,K) G2(1,2,K) 0 0;0 ...
G2(2,2,K) G2(1,2,K) 0;0 0 G2(2,2,K) G2(1,2,K)];
GG1(1:9,1:16,K)=[G12(:,:,K) G11(:,:,K) HZ HZ;HZ ...
G12(:,:,K) G11(:,:,K) HZ;HZ HZ G12(:,:,K) G11(:,:,K)];
end;%Use matrices to compute valid 2-D convolutions.
%Unwrap subimages into columns.
for K=1:9;XX=[];for K3=1:16;
X4=X3(1:4,1:4,K3,K);X5=X4(:);XX=[XX X5];end;
X6(1:16,1:16,K)=XX;
end;%Assemble blurred images.
for K=1:9;YY(1:9,1:16,K)=HH1(:,:,K)*X6(:,:,K);
ZZ(1:9,1:16,K)=GG1(:,:,K)*X6(:,:,K);end;
for K=1:9;for K3=1:16;
Y3(1:3,1:3,K3,K)=reshape(YY(1:9,K3,K),3,3);
Z3(1:3,1:3,K3,K)=reshape(ZZ(1:9,K3,K),3,3);end;
for K1=1:3;K11=[1:3]+(K1-1)*3;
for K2=1:3;K21=[1:3]+(K2-1)*3;K3=(K1-1)+(K2-1)*4+1;
Y2(K11,K21,K)=Y3(1:3,1:3,K3,K);
Z2(K11,K21,K)=Z3(1:3,1:3,K3,K);end;end;end;
for K1=1:3;K11=[1:9]+(K1-1)*9;
for K2=1:3;K21=[1:9]+(K2-1)*9;
K3=(K1-1)+(K2-1)*3+1;
Y(K11,K21)=Y2(:,:,K3);
Z(K11,K21)=Z2(:,:,K3);end;end;
figure,imagesc(Y),colormap(gray),axis off
title('BLURRED IMAGE USING PSF #1')
figure,imagesc(Z),colormap(gray),axis off
title('BLURRED IMAGE USING PSF #2')
%Finished with problem setup (finally!)
%GOAL: Compute HEST and GEST and XEST from Y and Z,
for K=1:9;%Find left null vector of [ZZ;YY].
[U S V]=svd([ZZ(:,:,K);YY(:,:,K)]');N(:,1)=V(:,15);
N11=[N(1,1) N(2,1) N(3,1) 0;0 N(1,1) N(2,1) N(3,1)];
N21=[N(4,1) N(5,1) N(6,1) 0;0 N(4,1) N(5,1) N(6,1)];
N31=[N(7,1) N(8,1) N(9,1) 0;0 N(7,1) N(8,1) N(9,1)];
NZ=zeros(2,4);A1=[N11 N21 N31 NZ;NZ N11 N21 N31];
N12=[N(10,1) N(11,1) N(12,1) 0;0 N(10,1) N(11,1) N(12,1)];
N22=[N(13,1) N(14,1) N(15,1) 0;0 N(13,1) N(14,1) N(15,1)];
N32=[N(16,1) N(17,1) N(18,1) 0;0 N(16,1) N(17,1) N(18,1)];
NZ=zeros(2,4);A2=[N12 N22 N32 NZ;NZ N12 N22 N32];
%Compute estimates of 2 spatially-varying PSFs.
[U S V]=svd([A1;A2]');H1EST=V(:,8);
H1EST=flipud(H1EST)/H1EST(4)*G2(1,1,K);%Fix scale factor.
HEST(1:2,1:2,K)=reshape(H1EST(1:4),2,2);%Estimate 1st PSF.
GEST(1:2,1:2,K)=reshape(H1EST(5:8),2,2);%Estimate 2ns PSF.
%Form matrices HH1 and HH2 using HEST and GEST.
HEST(1:2,1:2,K)=HEST(1:2,1:2,K)';
H11EST(1:3,1:4,K)=[HEST(2,1,K) HEST(1,1,K) 0 0;...
0 HEST(2,1,K) HEST(1,1,K) 0;0 0 HEST(2,1,K) HEST(1,1,K)];
H12EST(1:3,1:4,K)=[HEST(2,2,K) HEST(1,2,K) 0 0;0 ...
HEST(2,2,K) HEST(1,2,K) 0;0 0 HEST(2,2,K) HEST(1,2,K)];
HH1EST(1:9,1:16,K)=[H12EST(:,:,K) H11EST(:,:,K) HZ HZ;...
HZ H12EST(:,:,K) H11EST(:,:,K) HZ;HZ HZ H12EST(:,:,K)...
H11EST(:,:,K)];
GEST(1:2,1:2,K)=GEST(1:2,1:2,K)';%2nd PSF matrices.
G11EST(1:3,1:4,K)=[GEST(2,1,K) GEST(1,1,K) 0 0;...
0 GEST(2,1,K) GEST(1,1,K) 0;0 0 GEST(2,1,K) GEST(1,1,K)];
G12EST(1:3,1:4,K)=[GEST(2,2,K) GEST(1,2,K) 0 0;0 ...
GEST(2,2,K) GEST(1,2,K) 0;0 0 GEST(2,2,K) GEST(1,2,K)];
GG1EST(1:9,1:16,K)=[G12EST(:,:,K) G11EST(:,:,K) HZ HZ;...
HZ G12EST(:,:,K) G11EST(:,:,K) HZ;HZ HZ G12EST(:,:,K)...
G11EST(:,:,K)];
%K&actual values of PSFs are displayed as integers.
%Computed values of PSFs are displayed to 4 digits.
K,H2(:,:,K)',HEST(:,:,K),G2(:,:,K)',GEST(:,:,K),end;
%Noe compute XEST from HEST & GESY & Y & Z.
%Now compute image using reconstructed PSFs and
%wavelet transform of image, which is sparse.
W=sqrt(2)/2*[1,-1,0,0;0,0,1,-1];%Haar wavelet
W=[W;.5,.5,-.5,-.5;.5,.5,.5,.5];%transform
WW=kron(W,W);
for K=1:9;%Y=H*X -> Y=(H*WW')*(WW*X).
YYY(:,:,K)=[YY(:,:,K);ZZ(:,:,K)];
HHH(:,:,K)=[HH1(:,:,K);GG1(:,:,K)]*WW';
X6HAT(1:16,1:16,K)=pinv(HHH(:,:,K))*YYY(:,:,K);
%Inverse wavelet transform.
X6HAT(:,:,K)=WW'*X6HAT(:,:,K);
%Assemble subimage estmates into superimage.
for K3=1:16;X3HAT(1:4,1:4,K3,K)=...
reshape(X6HAT(1:16,K3,K),4,4);end;
for K1=1:4;K11=[1:4]+(K1-1)*4;
for K2=1:4;K21=[1:4]+(K2-1)*4;
K3=(K1-1)+(K2-1)*4+1;
X2HAT(K11,K21,K)=X3HAT(1:4,1:4,K3,K);end;end;end;
for K1=1:3;K11=[1:16]+(K1-1)*16;
for K2=1:3;K21=[1:16]+(K2-1)*16;
K3=(K1-1)+(K2-1)*3+1;
XHAT(K11,K21)=X2HAT(:,:,K3);end;end;
figure,imagesc(XHAT),colormap(gray),axis off
title('RECONSTRUCTED IMAGE')
```