UNIVERSITY OF LIVERPOOL

Structured Matrix Problems and Algorithms from Variational Image Deblurring

by

Ke Chen

Joint work with R H Chan (CUHK), Bryan Williams (UoL), Yalin Zheng and Simon Harding (UoL Eye and Vision Science)

Centre *for* Mathematical Imaging Techniques

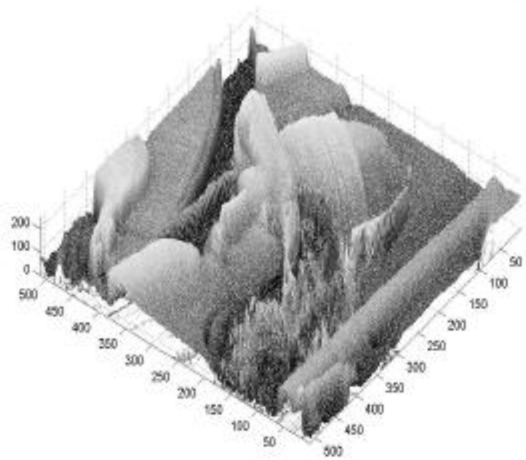
www.liv.ac.uk/cmit

Department *of* Mathematical Sciences

Imaging Introduction 1/2

Introduction – Discrete Image vs Surface function

$$\begin{pmatrix} 88 & 84 & 92 & 95 & 95 & 94 \\ 90 & 88 & 94 & 88 & 99 & 90 \\ 77 & 96 & 90 & 89 & 94 & 100 \\ 84 & 86 & 82 & 92 & 92 & 92 \\ 86 & 92 & 84 & 94 & 91 & 93 \\ 82 & 87 & 84 & 83 & 96 & 101 \end{pmatrix}$$



A Discrete Image vs Surface function → Variational

Introduction 2/2

Different Frameworks for Image Processing

Statistical/Stochastic Models:

Maximum Likelihood Estimation with uncertain data

Transform-Based Models:

Fourier/Wavelets --- Process features of images (noise/patch)

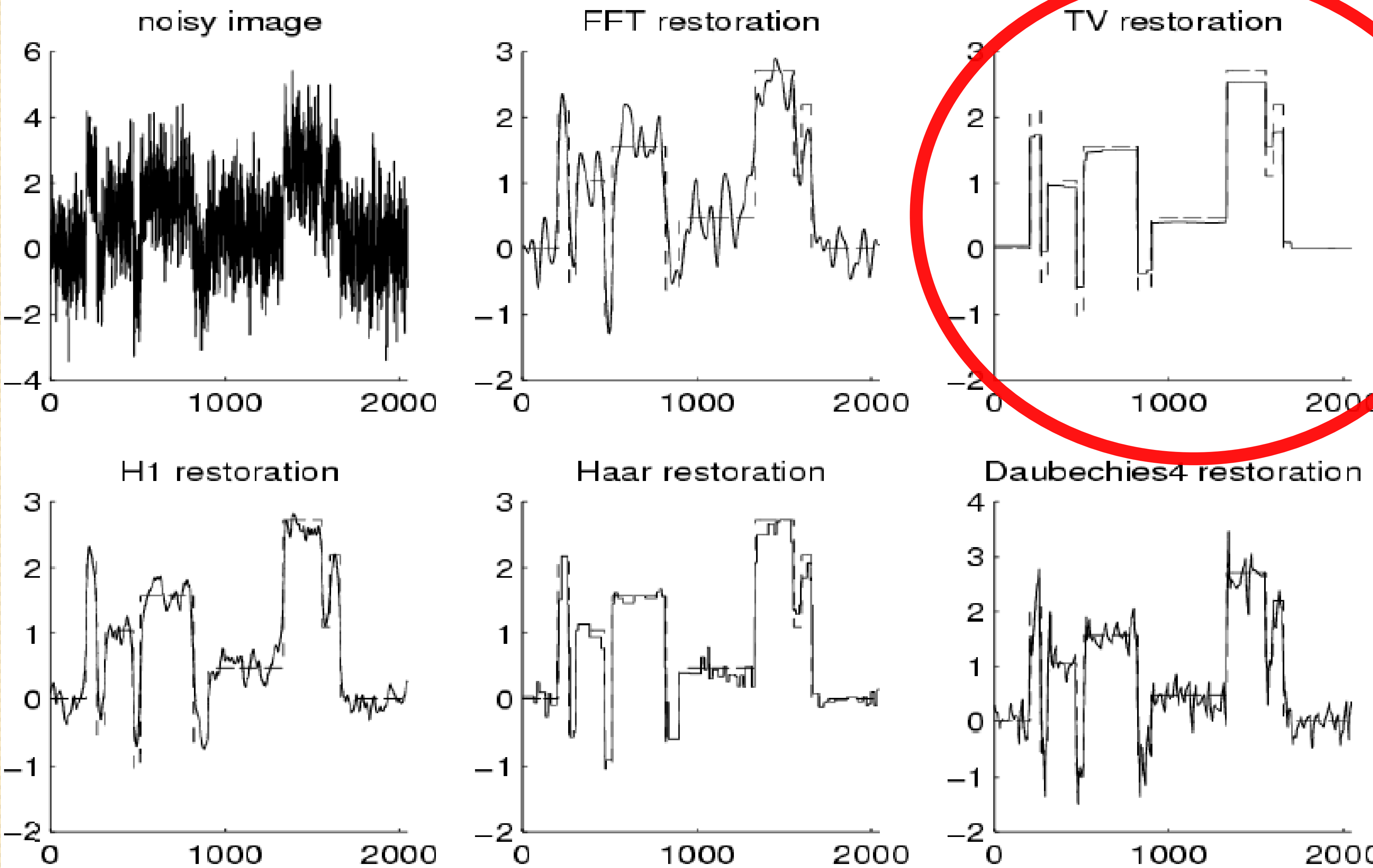
in transform space (e.g. thresholding)

Nonlinear PDE Models:

Evolve image according to local derivative/geometric info,

e.g. denoising → diffusion

Not all restoration techniques work well Chan-Strong (CAM96-07)



Liverpool publications: Google → Ke Chen

A. **Image Denoising:**
   Savage-C.(05 IJCM); TChan-C(06 SIAM MMS; 06 NumerAlg) ;
   TChan-C.-Carter(07 ETNA); RChan-C.(07 IJCM; 08 SISC);
   C.-Tai (2007 JSC); C.-Savage(07 BIT);
   Brito-C. (10 SIAM IS; 10 IEEE TIP);Yang-C.-Yu (12 JCM, 12 IJNAM);
   Zhang-C.-Yu(20120 IJCM; 12 SIAM NA)

A. **Image Deblurrring**:
   RChan-C.(2010 SISC)

B. **Image Inpainting:**
   Brito-C.(2008 JCM); Brito-C.(10 IJMM)

C. **Image Segmentation**:
   Badshah-C.(2008 CiCP; 09 IEEE TIP; 10 CiCP);
   Rada-C. (CiCP 2012)

D. **Image Registration**:
   Chumchob-C.(2009 IJNAM, 2011 JCAM and 2012 NMPDE);
   Chumchob-C.-Brito (2011 SIAM MMS)

OUTLINE of this TALK

I. **Image denoising**
II. **Image denoising and deblurring**
III. **Image blind deconvolution**

I. Image denoising

Observed image $\mathbf{z} = \mathbf{u} + \mathbf{n}$ where $\mathbf{u}$ = true image and $\mathbf{n}$ = Gaussian white noise.

Inverse Problem

Rudin-Osher-Fatemi(1992) denoising model is:

$$\min_{\mathbf{u}} J(\mathbf{u}) = \min_{\mathbf{u}} [\underbrace{\alpha \int_\Omega |\nabla \mathbf{u}|}_{\text{regularization term}} + \underbrace{\frac{1}{2}\|\mathbf{u} - \mathbf{z}\|_2^2}_{\text{data fitting term}}].$$

Euler-Lagrange Equation:

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = \mathbf{0} \implies \alpha \nabla \cdot \left(\frac{\nabla \mathbf{u}}{|\nabla \mathbf{u}|} \right) + \mathbf{u} = \mathbf{z}.$$

To solve Euler-Lagrange Equation

$$\mathcal{N}(\mathbf{u}) := \alpha\nabla\cdot\left(\frac{\nabla\mathbf{u}}{|\nabla\mathbf{u}|}\right) + \mathbf{u} = \mathbf{z}.$$

- nonlinear PDE—difficult to solve
- singular at $\nabla\mathbf{u} = \mathbf{0}$—smooth TV:

$$\mathcal{N}_\beta(\mathbf{u}) := \alpha\nabla\cdot\left(\frac{\nabla\mathbf{u}}{\sqrt{\mathbf{u}_x^2 + \mathbf{u}_y^2 + \beta}}\right) + \mathbf{u} = \mathbf{z}.$$

Competing Uni-grid Methods

- Time-marching (ROF, 92): $\dfrac{\partial \mathbf{u}}{\partial t} = \mathcal{N}_\beta(\mathbf{u}) - \mathbf{z}$
- Fixed-point method (Vogel/Oman, 96):

$$\mathcal{N}_\beta(u) = [\mathcal{L}_\beta(\mathbf{u})]\mathbf{u}$$

- Primal-dual method (Chan/Golub/Mulet, 96):

$$\mathbf{w} = \frac{\nabla \mathbf{u}}{\sqrt{\mathbf{u}_x^2 + \mathbf{u}_y^2 + \beta}}$$

- Dual method (Carter 02, Chambolle 04): $(\beta = 0)$

$$\|\mathbf{u}\|_{TV} = \max_{|\mathbf{p}_{i,j}| \le 1} \left\{ \mathbf{u}^T \mathrm{div}\mathbf{p} \right\}$$

$$u \rightarrow z - \alpha \mathrm{div} p$$

Multi-grid Methods for the E-L Eqn

I) Linearized PDE: $\mathcal{N}_\beta(\mathbf{u}) \rightarrow [\mathcal{L}_\beta(\bar{\mathbf{u}})]\mathbf{u}$ where

$$[\mathcal{L}_\beta(\bar{\mathbf{u}})]\mathbf{u} := \alpha \nabla \cdot \left(\frac{\nabla \mathbf{u}}{\sqrt{\bar{\mathbf{u}}_x^2 + \bar{\mathbf{u}}_y^2 + \beta}} \right) + \mathbf{u}$$

- Chan/Chan/Wan (95)
- Vogel (1995)
- Chang/Chern (1996, AMG 2003)

Inner-outer iterations

Multi-grid Methods for the E-L Eqn

II) Nonlinear PDE: solve $\mathcal{N}_{\beta}(\mathbf{u}) = \mathbf{z}$

- ☐ Weickert (2003)
- ☐ Savage/Chen (2004)
- ☐ Henn (2004)

No inner iterations

Nonlinear Full Approximation Scheme

On level k, we solve $\mathcal{N}_k(\mathbf{w}_k) = \mathbf{f}_k$. At current $\bar{\mathbf{w}}_k$, the residual equation is:

$$\mathcal{N}_k(\bar{\mathbf{w}}_k + \mathbf{v}_k) = \mathbf{f}_k.$$

So the coarse grid equation to solve is:

$$\mathcal{N}_{k-1}(\mathbf{w}_{k-1}) = \mathcal{N}_{k-1}(\bar{\mathbf{w}}_{k-1}) + \mathcal{R}_k^{k-1}[\mathbf{f}_k - \mathcal{N}_k(\bar{\mathbf{w}}_k)],$$

where $\mathbf{v}_{k-1} = \mathbf{w}_{k-1} - \mathcal{R}_k^{k-1}\bar{\mathbf{w}}_k$. Then

$$\mathbf{w}_{\text{new}} = \bar{\mathbf{w}}_k + \mathcal{P}_{k-1}^k \mathbf{v}_{k-1} = \mathcal{P}_{k-1}^k \mathbf{w}_{k-1}.$$

To solve the ROF TV Optimization by Multilevel Methods

- Ta'ssan (1987), Nash (1997, 2000) Brandt (2000)
- Solve $\min_{\mathbf{u}} J(\mathbf{u})$ directly
- use $\nabla J(\mathbf{u})$ to measure residuals but not solving $\nabla J(\mathbf{u}) = 0$
- $\nabla J(\mathbf{u}) = 0$ is used to design coarse level functionals.
- minimization problems to solve ("*smooth*") in each grid

Nash's Approach

Minimize

$$J(\mathbf{u}) = \left[\alpha \int_\Omega \sqrt{\mathbf{u}_x^2 + \mathbf{u}_y^2 + \beta} + \frac{1}{2}\int_\Omega (\mathbf{u} - \mathbf{z})^2\right].$$

The *"closeness"* of the current $\mathbf{u}_k$ to solving

$$\min_{\mathbf{u}} J(\mathbf{u})$$

is measured by its E-L equation: $\mathbf{r}_k = \nabla J(\mathbf{u}_k)$.

Target is:

$$\nabla J(\mathbf{u}_k) = 0.$$

Require $J(\mathbf{u})$ to be differentible.

Nash's Approach

Coarse grid problem is to minimize:

$$F(\mathbf{u}_{k-1}) := J(\mathbf{u}_{k-1}) - \mathbf{g}_{k-1}^T \mathbf{u}_{k-1},$$

where

$$\mathbf{g}_{k-1} = \nabla J(\mathcal{R}_k^{k-1}\mathbf{u}_k) - \mathcal{R}_k^{k-1}\nabla J(\mathbf{u}_k).$$

Note that $\nabla F(\mathbf{u}_{k-1}) = \mathbf{0}$ implies

$$\nabla J(\mathbf{u}_{k-1}) = \mathbf{g}_{k-1}.$$

This is the FAS (NMG)

An Optimisation Smoother

$$J(u_1, u_2, \ldots, u_n) = \alpha \sum_{j=1}^{n} |u_j - u_{j+1}| + \frac{1}{2} \sum_{j=1}^{n} (u_j - z_j)^2$$

is minimized by coordinate descent:

$$\begin{aligned} & \min_{u_1} J(u_1, u_2, \ldots, u_n) \to \min_{u_2} J(u_1, u_2, \ldots, u_n) \\ \to \quad & \cdots \\ \to \quad & \min_{u_n} J(u_1, u_2, \ldots, u_n) \to \min_{u_1} J(u_1, u_2, \ldots, u_n) \\ \to \quad & \cdots \end{aligned}$$

Note that

$$\arg\min_{u_i} J(u_1, \ldots, u_i, \ldots, u_n) \in \{z_j \pm 2\alpha, u_{j\pm 1}, z_j\}.$$

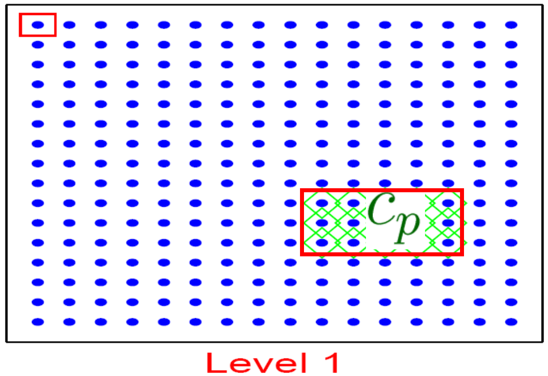
Drawbacks of Previous Approach

- require differentiability of $J(\mathbf{u})$
- can get stuck for constant patches

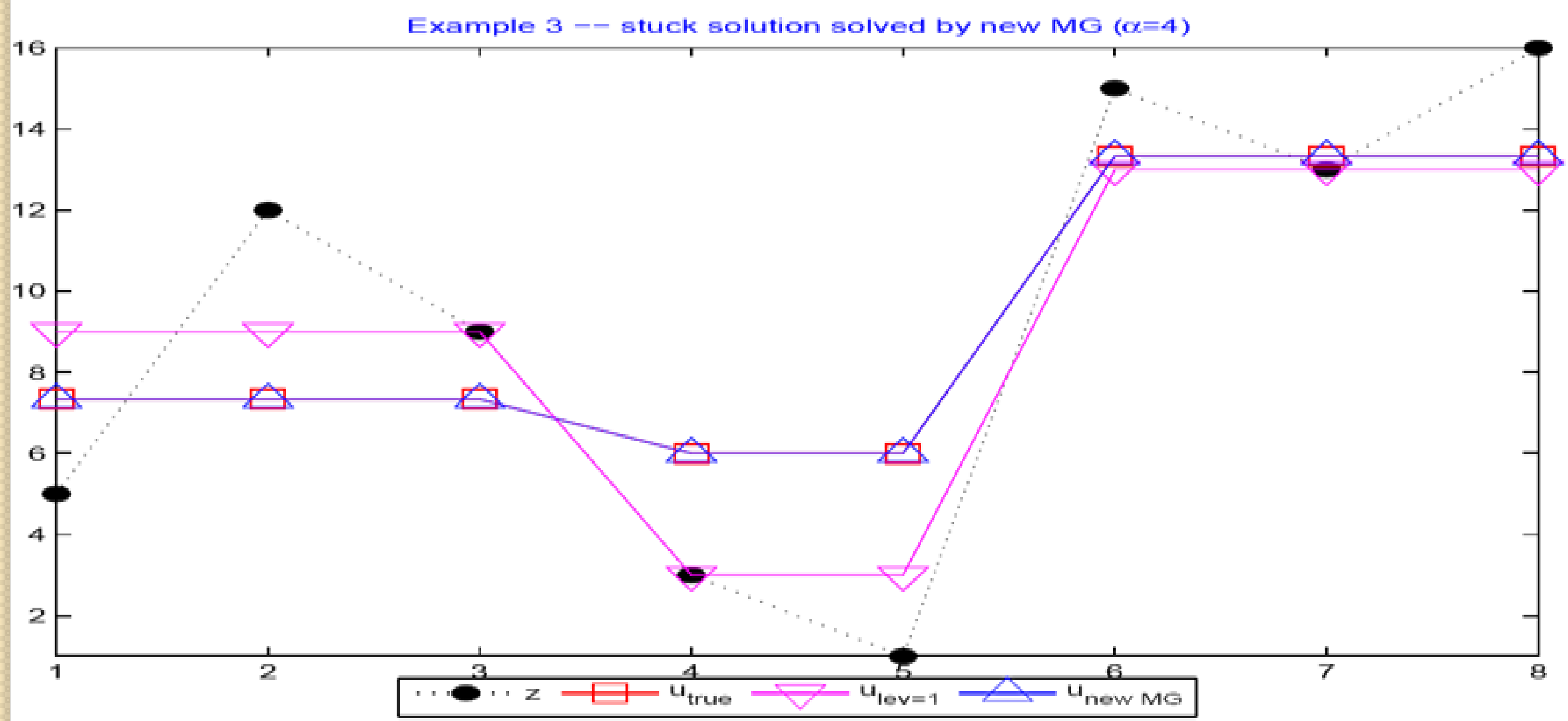
Both Uni-level and

Multilevel gets stuck

T. Chan and Chen's Approach



$\min_{c_p} J(\bar{\mathbf{u}} + c_p)$
over the patch



T. Chan/Chen (SIMMS, 2006)

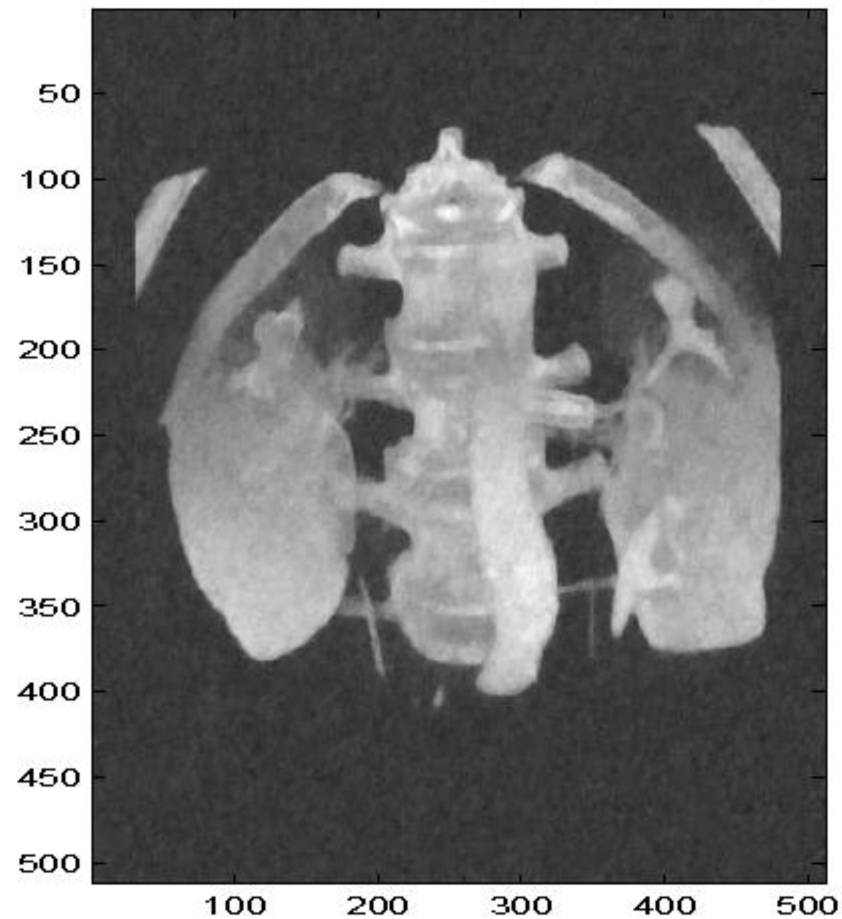
- Complexity per cycle:

  $O($ level number $\times$ pixel number $) = O(N \log N)$

- Convergence:
  - $J(\mathbf{u}_{\text{new}}) \leq J(\mathbf{u}_{\text{old}})$
  - converge globally if all constant patches can be identified

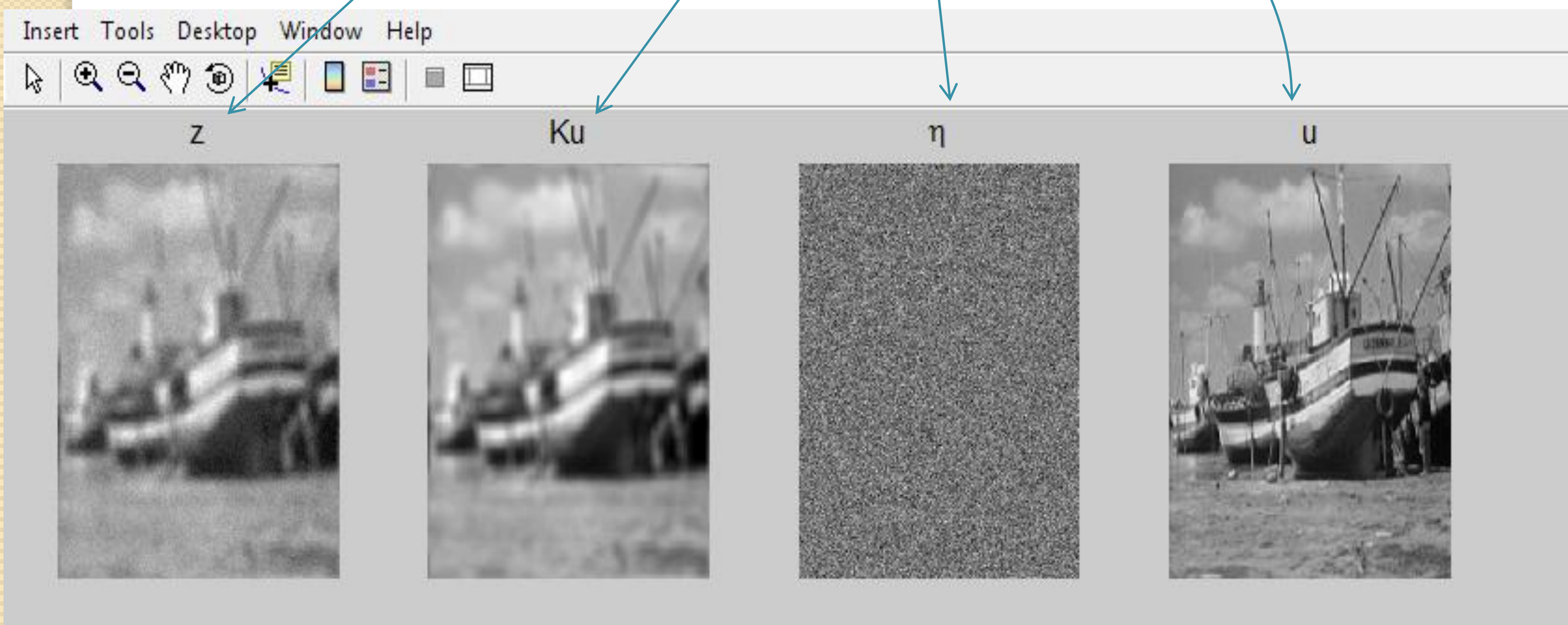
- Numerically:
  - convergence independent of N
  - total work proportional to $N \log N$

TV Results – Sharp Restoration



II. Image denoising and deblurring

$$z = Ku + \eta$$



Variational ROF model: Opt vs PDE

$$\min_u J(u) = \bar{\alpha} \underbrace{\int_\Omega |\nabla u| \, dxdy}_{\text{Total variation}} + \frac{1}{2} \| Ku - z \|_2^2$$

$$\Updownarrow$$

$$-\bar{\alpha} \nabla \left(\frac{\nabla u}{|\nabla u|} \right) + K^* K u = K^* z$$

- Differential (Sparse) + Integral (Dense) Operator: Hard
- Fixed Point Methods: Vogel, Chan et al x 2, Chang et al
- Feasible Method: M Ng et al / Yin et al – Approx/ 2-stages
- Non-feasible Method: MG / Multilevel

The Huang-Ng-Wen (2008, MMS) method

$$\min_{u} J(u) = \bar{\alpha} \underbrace{\int_{\Omega} |\nabla u| \, dxdy}_{\text{Total variation}} + \frac{1}{2} \| \mathbf{K} u - z \|_2^2$$

$\Downarrow$ *approximated by the following* alterating mins

$$\min_{u,v} J(u,v) = \bar{\alpha} \underbrace{\int_{\Omega} |\nabla u| \, dxdy + \frac{\bar{\beta}}{2} \| u - v \|_2^2}_{\text{Stage 1 for } u \text{ and Stage 2 for } v} + \frac{1}{2} \| \mathbf{K} v - z \|_2^2$$

20 alternating iterations ➔ 40 problems to be solved in all

II(a). Attempts to develop a multigrid / multilevel method

- **Multigrids for PDE:** **linear with FP**

  Vogel et al (95) – negative results shown

  Chan-Chan-Wan (97) – MG for L2-norm (not TV)

- **Mutilevel for Optimization:** **nonlinear / dim reduction**

  Chan-Chen (06) method for denoising ?→ for denoising and deblurring?

II(b). A new direct optimization method

Joint work with **R H Chan** (2008)

$$\min_{u} J(u) = \bar{\alpha} \int_{\Omega} |\nabla u| \, dxdy + \frac{1}{2} \| Ku - z \|_2^2$$

$$\Downarrow \quad \min_{u_{11}, u_{12}, \ldots, u_{nn}} J(\mathrm{u}) = \alpha \sum_{i,j} |\nabla \mathrm{u}|_{ij} + \frac{1}{2} \sum_{j} [(\mathrm{Ku})_j - z_j]^2$$

$$\Updownarrow$$

$$\min_{u_{11}, u_{12}, \ldots, u_{nn}} J(u) = \alpha \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} \sqrt{(u_{i,j} - u_{i,j+1})^2 + (u_{i,j} - u_{i+1,j})^2}$$

$$+ \sum_{i=1}^{n} \sum_{j=1}^{n} \Big(\sum_{\ell=1}^{n} \sum_{m=1}^{n} K_{i,j;\ell,m} u_{\ell,m} - z_{i,j} \Big)^2$$

Level 1 minimization

$O(n^2)$ per pixel $\times$ n^2pixels $= O(n^4)$

Let $u = \tilde{u} + c_{i,j}\mathrm{e}_{i,j}, \quad c_{i,j} \in \mathrm{R}, \ \mathrm{e}_{i,j} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \quad \mathrm{K}u = \mathrm{K}\tilde{u} + c_{i,j}\mathrm{K}\mathrm{e}_{i,j} = \mathrm{K}\tilde{u} + c_{i,j}\mathrm{w}_t$

$$J^{\mathrm{loc}}(c_{i,j}) = \alpha T(c_{i,j}) + \frac{1}{2}\| c_{i,j} w_t + K\tilde{u} - z \|^2$$

$$= \alpha T(c_{i,j}) + \frac{1}{2}\| c_{i,j} w_t - \tilde{z} \|^2$$

Level 2 minimization

$$O(n^2) \text{ per pixel } \times \frac{n^2}{4} \text{ pixels } = O(n^4)$$

Let $u = \tilde{u} + c_{i,j}\mathrm{e}_{i,j}, \quad \mathrm{K}u = \mathrm{K}\tilde{u} + c_{i,j}\mathrm{K}\mathrm{e}_{i,j} = \mathrm{K}\tilde{u} + c_{i,j}\mathrm{w}_t, \quad c_{i,j} \in \mathrm{R}, \; \mathrm{e}_{i,j} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ \vdots \end{pmatrix}$

$$J^{\mathrm{loc}}(c_{i,j}) = \alpha T(c_{i,j}) + \frac{1}{2}\| c_{i,j} w_t + K\tilde{u} - z \|^2$$

$$= \alpha T(c_{i,j}) + \frac{1}{2}\| c_{i,j} w_t - \tilde{z} \|^2$$

Consider how to generate w_t

$$\alpha T'(c_{i,j}) + w_t^T w_t c_{i,j} = w_t^T \tilde{z}$$

$$\left[\begin{array}{cc|cc|cc|cc||cc|cc|cc|cc}
a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p \\
h & a & b & c & d & e & f & g & p & i & j & k & l & m & n & o \\
\hline
g & h & a & b & c & d & e & f & o & p & i & j & k & l & m & n \\
f & g & h & a & b & c & d & e & n & o & p & i & j & k & l & m \\
\hline
e & f & g & h & a & b & c & d & m & n & o & p & i & j & k & l \\
d & e & f & g & h & a & b & c & l & m & n & o & p & i & j & k \\
\hline
c & d & e & f & g & h & a & b & k & l & m & n & o & p & i & j \\
b & c & d & e & f & g & h & a & j & k & l & m & n & o & p & i \\
\hline\hline
i & j & k & l & m & n & o & p & a & b & c & d & e & f & g & h \\
p & i & j & k & l & m & n & o & h & a & b & c & d & e & f & g \\
\hline
o & p & i & j & k & l & m & n & g & h & a & b & c & d & e & f \\
n & o & p & i & j & k & l & m & f & g & h & a & b & c & d & e \\
\hline
m & n & o & p & i & j & k & l & e & f & g & h & a & b & c & d \\
l & m & n & o & p & i & j & k & d & e & f & g & h & a & b & c \\
\hline
k & l & m & n & o & p & i & j & c & d & e & f & g & h & a & b \\
j & k & l & m & n & o & p & i & b & c & d & e & f & g & h & a
\end{array}\right], \quad
\left[\begin{array}{c}
a+b+e+f \\
h+a+d+e \\
g+h+c+d \\
f+g+b+c \\
e+f+a+b \\
d+e+h+a \\
c+d+g+h \\
b+c+f+g \\
i+j+m+n \\
p+i+l+m \\
o+p+k+l \\
n+o+j+k \\
m+n+i+j \\
l+m+p+i \\
k+l+o+p \\
j+k+n+o
\end{array}\right]$$

Task 1 – $O(n^2 \log n)$ work to work out partial sums of columns

Algorithm 2 (Computation of partial sums of K) *Let $T_1 \in \mathbb{R}^{n\times n}$ be the of the BCCB matrix K (which is reshaped from column 1 of K). Below for $k \geq 2$, we shall form the root matrix $T_k \in \mathbb{R}^{n\times n}$ that contains partial sums of K's 4^{k-1} columns and $W_k = \|T_k\|_F^2$.*

for $k = 2, \ldots, L+1$, do:

– *Set $b = 2^{k-2}$ (the width of blocks on level $k-1$)*

– *Assign $Y = T_{k-1}$*

1. *Inter-block summation*
   – *Set the first block $X_1 = Y(:, 1:b) + Y(:, n-b+1:n)$*
   – *for $\ell = b+1, 2b+1, 3b+1, \ldots, (n-b+1)$, do*
   Set block ℓ: $X_\ell = Y(:, \ell:\ell+b-1) + Y(:, \ell-b:\ell-1)$
   end
   – *Set the matrix $X = [X_1,\ X_{b+1},\ X_{2b+1},\ X_{3b+1}, \ldots,\ X_{(n-b+1)}]$*

2. *Inner-block summation*
   – *Set the first block $Y_1 = X(1:b,:) + X(n-b+1:n,:)$*
   – *for $\ell = b+1, 2b+1, 3b+1, \ldots, (n-b+1)$, do*
   Set block ℓ: $Y_\ell = X(\ell:\ell+b-1,:) + X(\ell-b:\ell-1,:)$
   end
   – *Set the matrix $Y = [Y_1;\ Y_{b+1};\ Y_{2b+1};\ Y_{3b+1}; \ldots,\ Y_{(n-b+1)}]^T$*

3. *Compute $W_k = \|Y\|_F^2$ which is the same as $w_t^T w_t$ on level k.*

– *Store $T_k = Y$*

end

Task II – $O(n^2 \log n)$ work to work out all RHS of local problems

Algorithm 3 (Computation of products $w_t^T \tilde{z}$) *Let fft2(K, v) denote the process of computing the matrix-vector product $K^T v$ via the FFT. Suppose that $\tilde{z}$ has been updated before starting level k. Now on level k, we shall compute the vector $V = w_t^T \tilde{z}$ by the following steps*

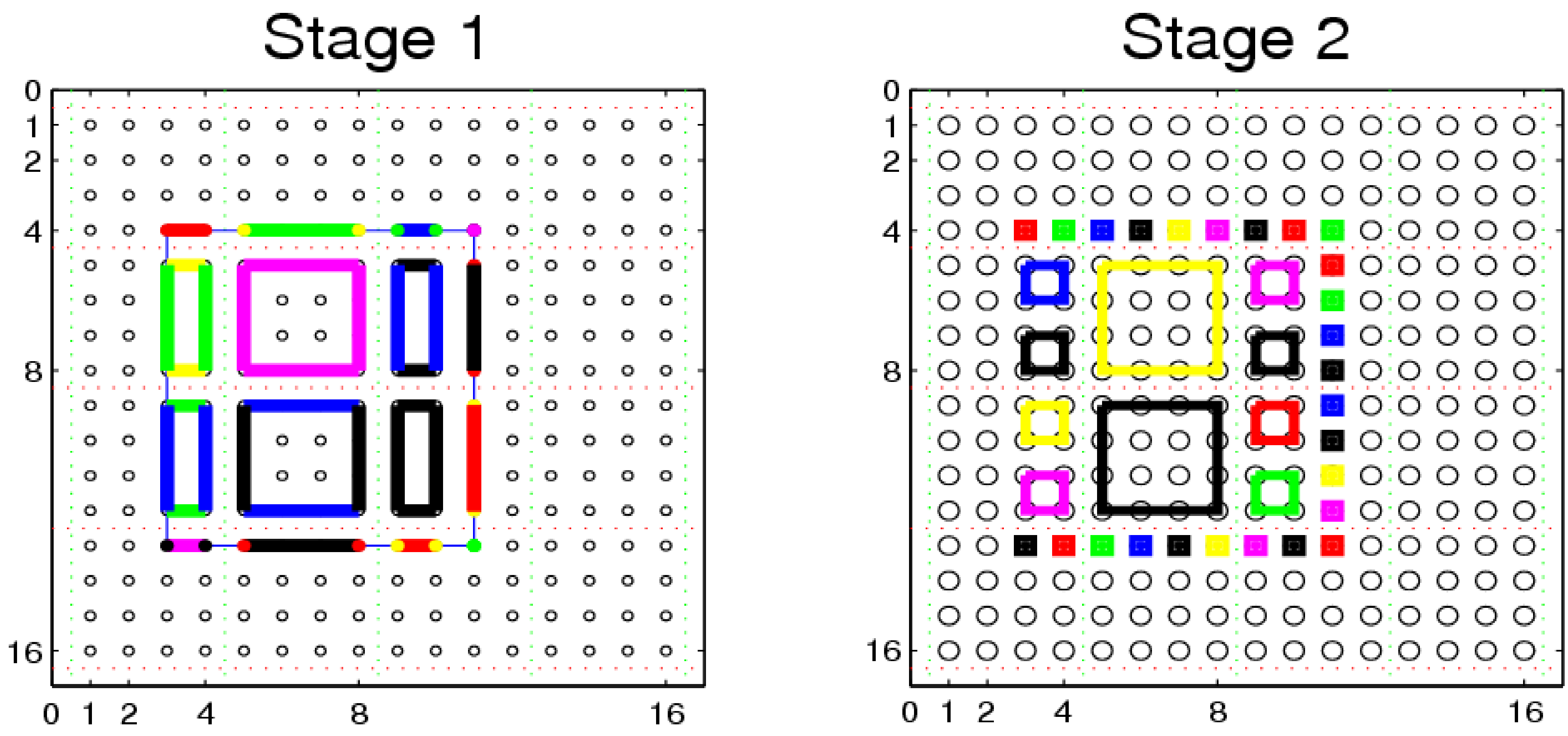
1. *Complete $W = \text{fft2}(K, \tilde{z})$; here $W \in \mathbb{R}^{n^2}$.*
   If $k = 1$, exit the algorithm with $V = W$ otherwise continue;

2. *for $\ell = 0, \ldots, k-2$ do*
   - *Set the counter $j = 0$, the block size $b = 2^\ell$ and the number of blocks $m = n/b$ along one way.*
   - *for column $c = 1, 3, \ldots, m-1$ do*
     set $s = (c-1)m$;
     for row $r = 1, 3, \ldots, m-1$ do
     j=j+1;
     compute the partial sum $p_j = \sum_{i=0}^{1} (W_{s+r+i} + W_{s+r+m+i})$.
     end for r
     end for c
   - *Set $W = [p_1, p_2, \ldots, p_j]^T$ and the new size $m = m/2$;*

   end for ℓ.

3. *End the algorithm with $V = W$.*

Task III – $O(n^2 \log n)$ work to work out irregular partial sums of columns



Overall Algorithm: Initial Stage (FMM like) + Iteration Stage (Multilevel like)

Algorithm 6 *Given z and an initial guess $\widetilde{u} = z$, with $L+1$ levels,*

Pre-calculation.

1. *Apply Algorithm 2 to compute all root matrices T_k and $w_t^T w_t = \|T_k\|_F^2$ for partial sum matrices on level $k = 1, 2, \ldots, L+1$.*

Multilevel Iterations.

2. *Iteration starts with $u_{old} = \widetilde{u}$ ($\widetilde{u}$ contains the initial guess before the first iteration and the updated solution at all later iterations).*

for *ν times on each level $k = 1, 2, 3, \ldots, L+1$:*

3. *Compute $\widetilde{z} = z - K\widetilde{u}$ and form $K^T\widetilde{z}$ via the FFT.*

for *each block on level k,*

4. *form each $w_t^T\widetilde{z}$ from $K^T\widetilde{z}$ and compute the minimizer c of (23).*

end *block.*

5. *add all the corrections (from all blocks on level k), $\widetilde{u} = \widetilde{u} + P_k c$, where P_k is the interpolation operator distributing $c_{i,j}$ to the corresponding $b \times b$ block on level k as illustrated in (16).*

end *level k.*

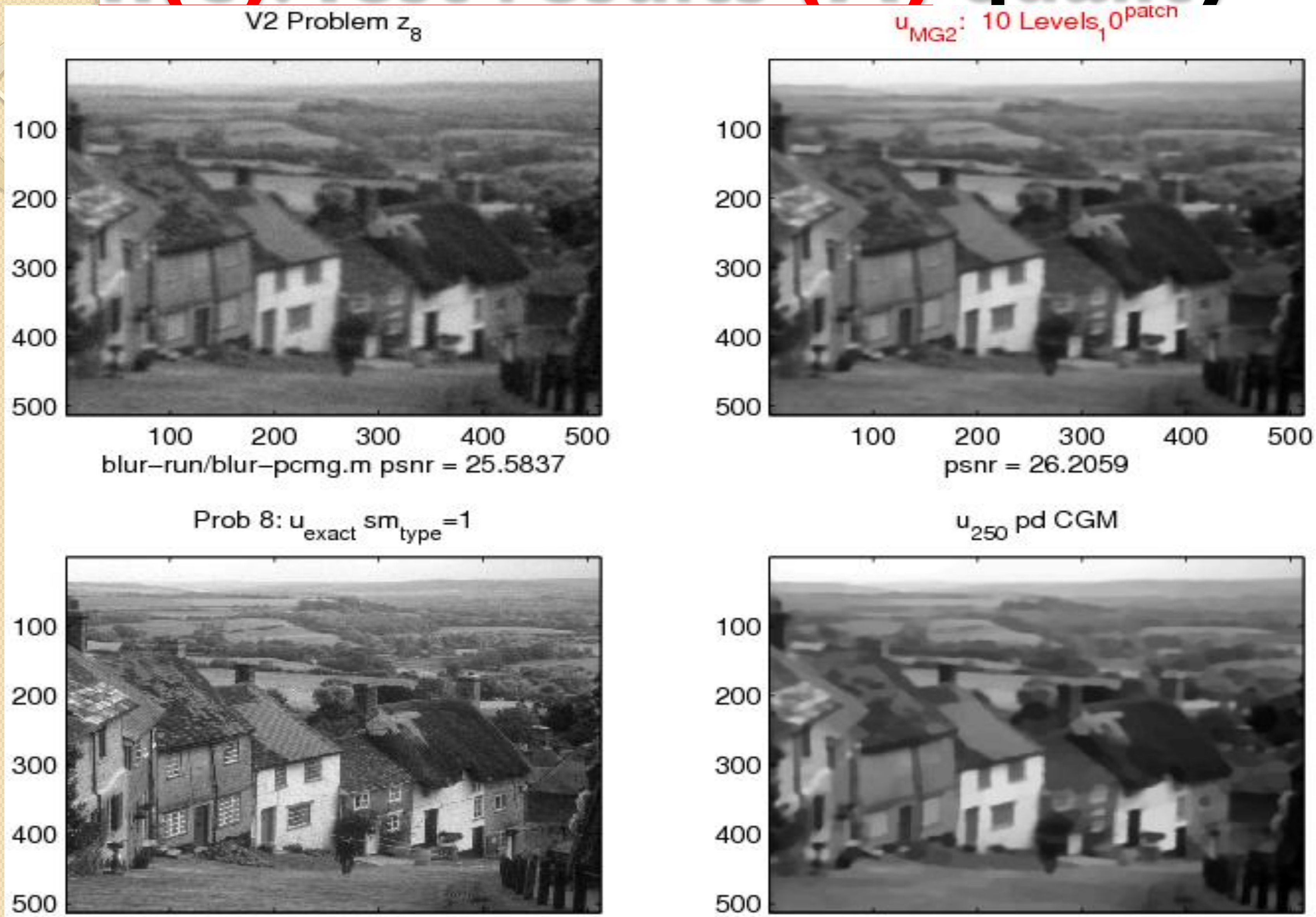
6. *On level $k = 1$, check the possible patch size for each position (i, j):*

$$patch = \{(i_\ell, j_\ell) \ : \ |u_{i_\ell, j_\ell} - u_{i,j}| < \varepsilon\}$$

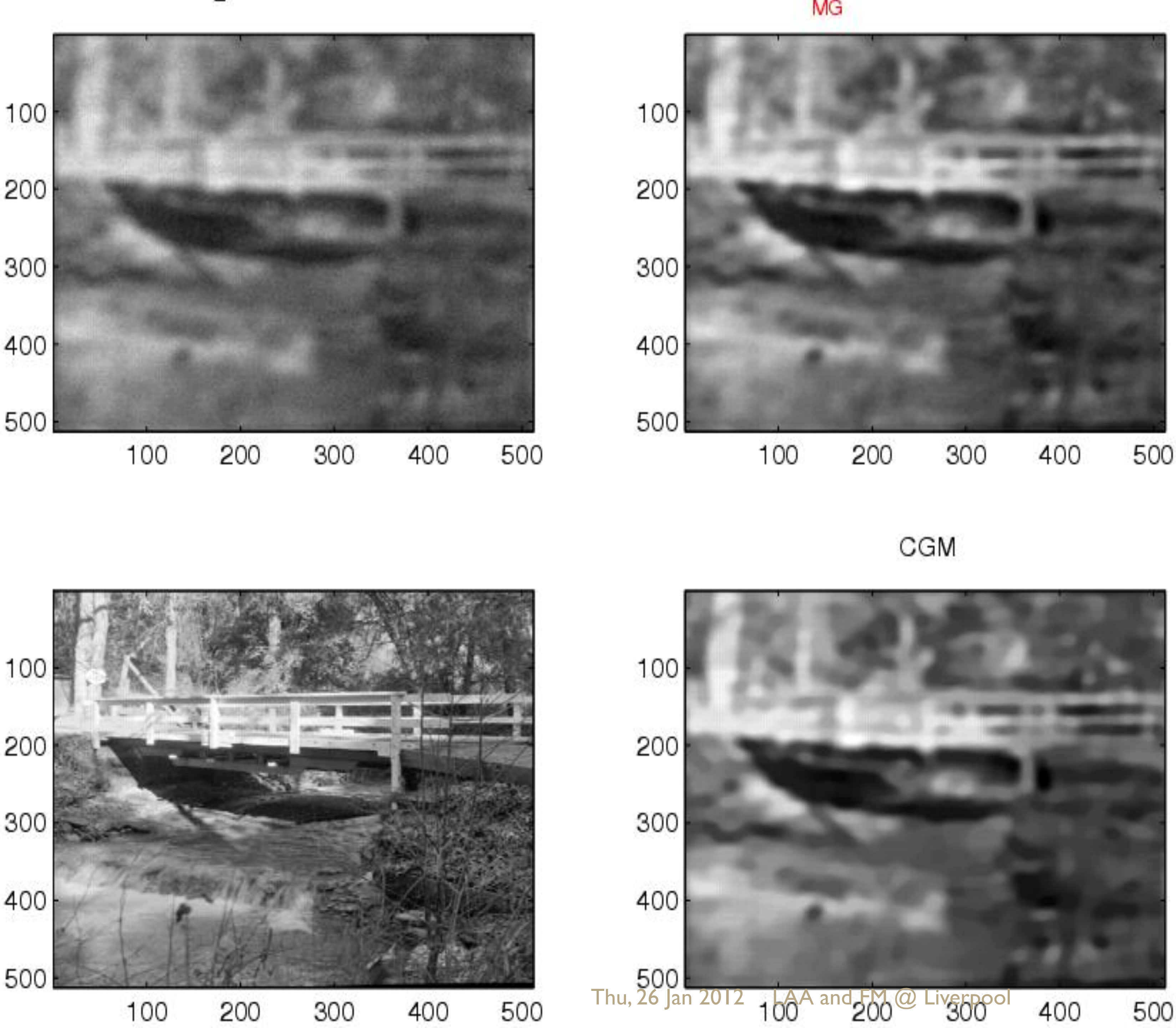
for some small ε.
First use Algorithms 5 to compute the partial sum vector w_t.
Then implement the piecewise constant update as with Steps $3-5$.

7. *If $\|\widetilde{u} - u_{old}\|_2$ is small enough, exit with $u = \widetilde{u}$ or return to Step 2 and continue with the next multilevel cycle.*

II(c). Test results (A) quality



psnr=25.80

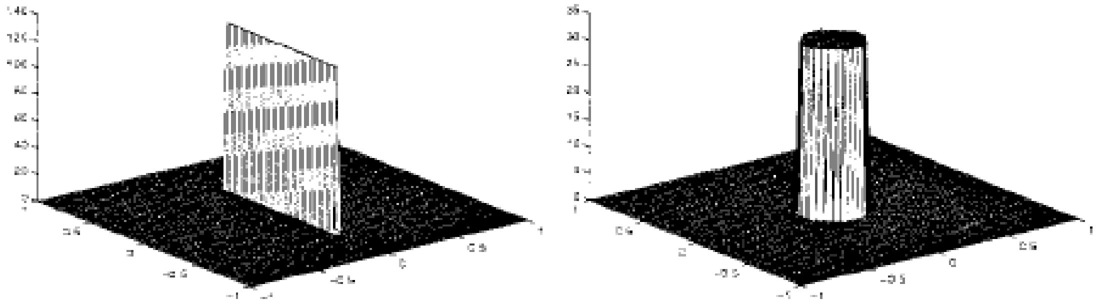

MG
CGM
100
200
300
400
500

II(d). Test results (B) Speed comparison

Problem number	Size n	CGM Method		New Algorithm 6	
		PSNR	CPU	PSNR	CPU
1	128	19.3	113.8	18.8	8.7
	256	20.9	646.7	21.0	29.0
	512	20.1	3350.0	19.9	193.5
	1024	*	*	22.2	8313.1
2	128	20.5	114.6	19.8	8.7
	256	23.0	648.5	22.8	38.9
	512	21.8	3356.1	19.9	192.1
	1024	*	*	24.8	8307.3
3	128	21.9	113.9	21.5	8.7
	256	24.1	649.2	23.8	36.6
	512	23.1	3359.2	23.3	228.3
	1024	*	*	25.7	8309.9
4	128	22.7	114.4	22.6	8.5
	256	24.9	648.1	24.9	28.9
	512	22.5	3362.4	22.8	229.1
	1024	*	*	25.2	9214.9

III. Image blind deconvolution

$$\min_{u,h}\left\{\alpha_1\int|\nabla u|\,dxdy+\alpha_2\int|\nabla h|\,dxdy+\frac{1}{2}\int(h*u-u_0)^2\,dxdy\right\}$$



T.F. Chan and W.C. Wong.
IEEE transactions on Image Processing,
Vol. 7, No. 3, March 1998.

PDEs to solve

$$h(-x,-y)*(h*u-u_0)-\alpha_1\nabla\cdot\left(\frac{\nabla u}{|\nabla u|}\right)=0,\quad x\in\Omega$$

$$u(-x,-y)*(u*h-u_0)-\alpha_2\nabla\cdot\left(\frac{\nabla h}{|\nabla h|}\right)=0,\quad x\in\Omega$$

Non-uniqueness of the solution

$$[-u(x,y), -h(x,y)],$$

$$[u(x \pm c, y \pm d), h(x \pm x, y \pm d)],$$

$$[\frac{\alpha_2}{\alpha_1} h(x,y), \frac{\alpha_1}{\alpha_2} u(x,y)]$$

$$\int h(x,y)\ dxdy = 1$$

$$u(x,y) \geq 0, h(x,y) \geq 0, h(x,y) = h(-x,-y)$$

Constraints to guarantee uniqueness of the solution

Alternating Minimization Algorithm

A. AM Algorithm

- Start with $u^0 = z$ and $h^0 = \delta(x, y)$, the delta function. Assume we have u^n and h^n.
- Solve for h^{n+1} by (iterating on i)

$$u^n(-x, -y)x \star (u^n \star h^{n+1}_{i+1} - z) - \alpha_2 \nabla \cdot \left(\frac{\nabla h^{n+1}_{i+1}}{|\nabla h^{n+1}_i|} \right) = 0.$$

Impose

$$h^{n+1}(x, y) = \begin{cases} h^{n+1}(x, y), & \text{if } h^{n+1}(x, y) > 0, \\ 0 & \text{otherwise} \end{cases}$$

$$h(x, y) = [h(x, y) + h(-x, -y)]/2$$

$$h^{n+1} = \frac{h^{n+1}}{\int_\Omega h^{n+1}(x, y)\, dx\, dy.}$$

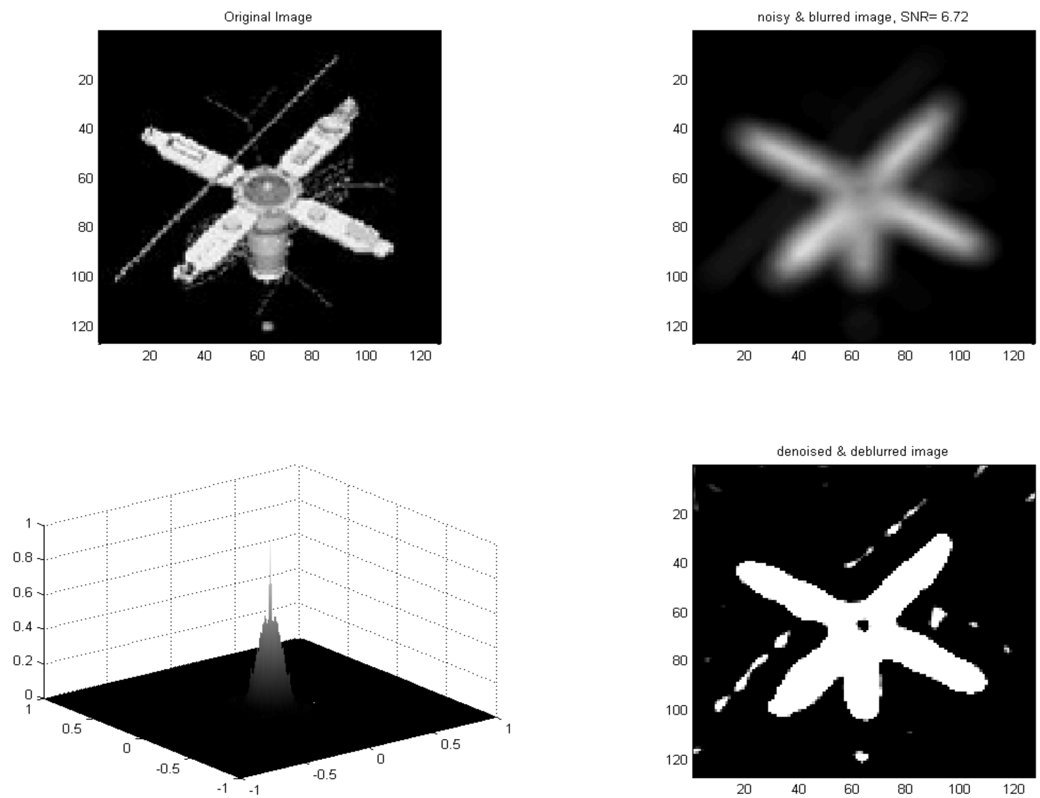
- Solve for u^{n+1} by (iterating on i)

$$h^{n+1}(-x, -y) \star (h^{n+1} \star u^{n+1}_{i+1} - z) - \alpha_1 \nabla \cdot \left(\frac{\nabla u^{n+1}_{i+1}}{|\nabla u^{n+1}_i|} \right) = 0.$$

Impose

$$u^{n+1}(x, y) = \begin{cases} u^{n+1}(x, y), & \text{if } u^{n+1}(x, y) > 0 \\ 0, & \text{otherwise.} \end{cases}$$

Fair Double TV solution



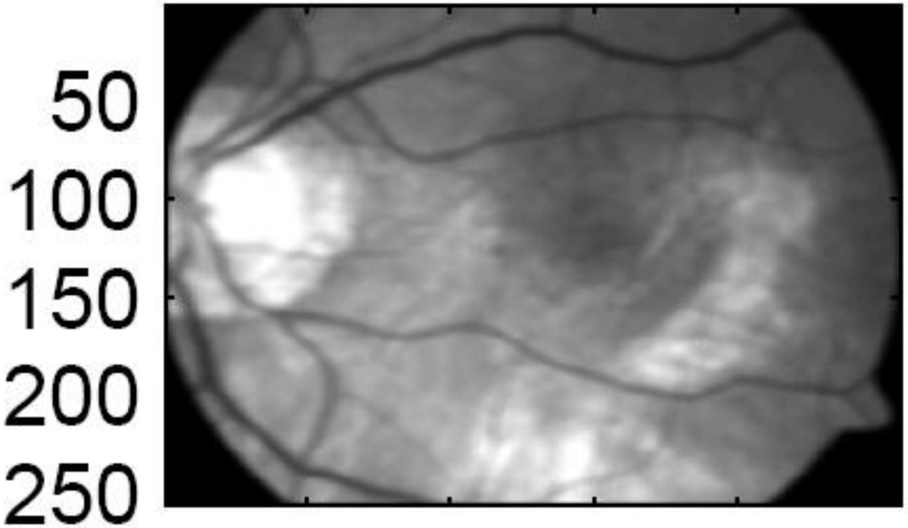
A more difficult problem

Medical images on the other hand are not so easily tractable.

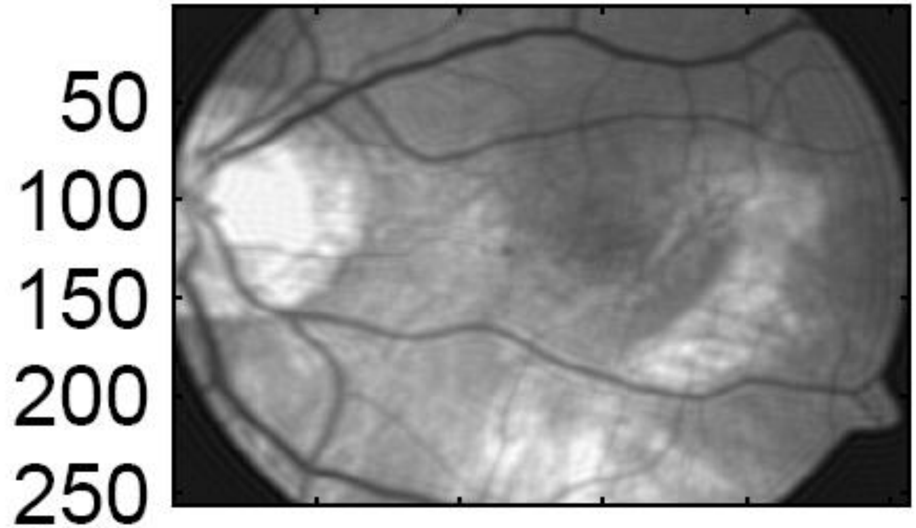
They are highly noisy and blurred

Preliminary Results Encouraging

Received Image



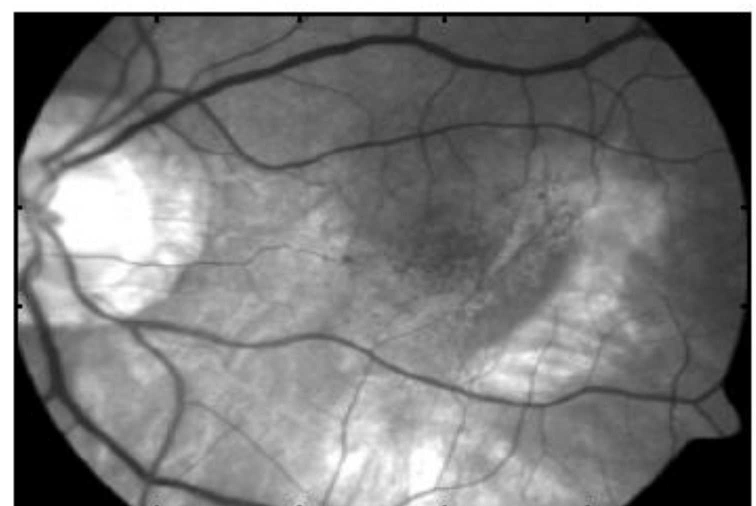
Restored Image



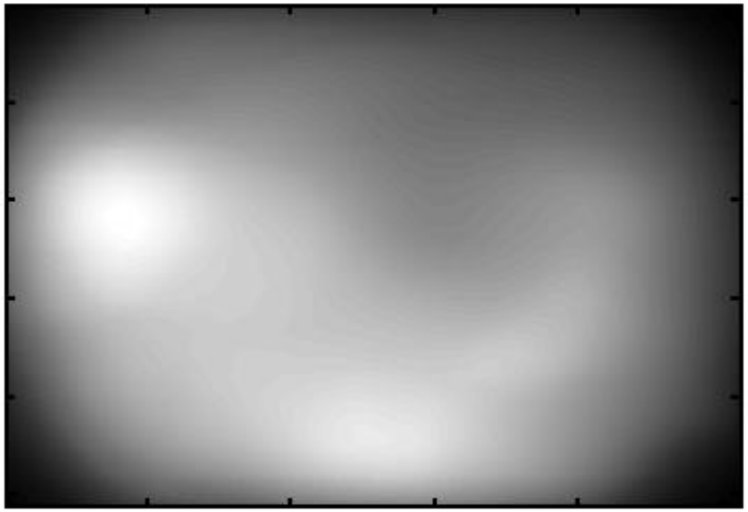
Motion Blur

$$k = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$ is approximated by iterations

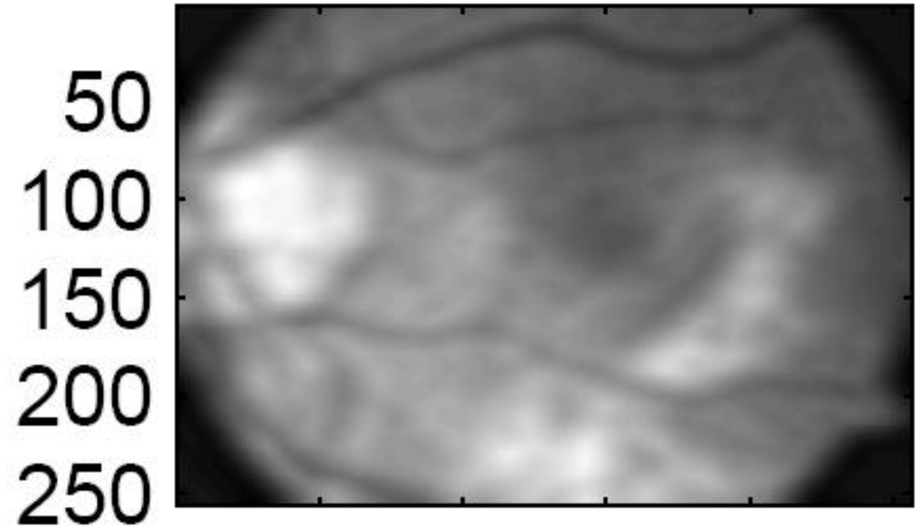
Original Image



Received Image

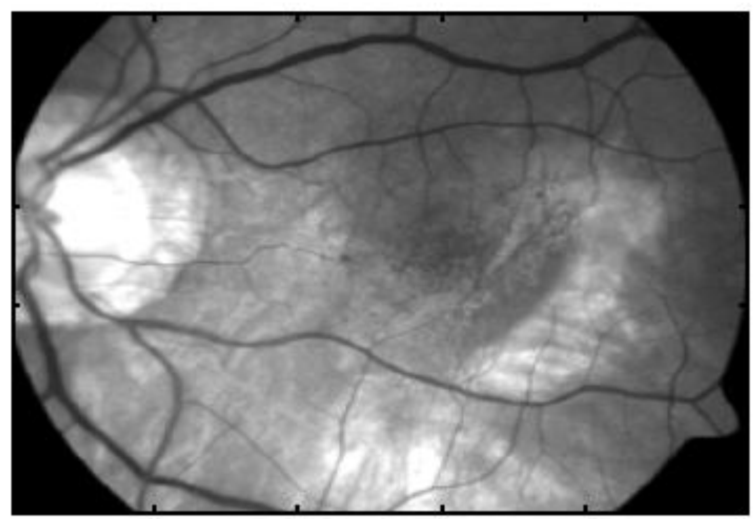


Restored Image



Gaussian Blur

Original Image



$$k_{i,j} = \exp\left(-\frac{(x_i^{\,2}+y_j^{\,2})}{\sigma}\right)$$ is approximated by iterations

Received Image

Restored Image

Received Image

Restored Image

Conclusions

- In Imaging Deblurring framework, (nonlinear) regularisers complicate integral operators. Optimisation multilevel methods effective.
- Generalization to blind deblurring is non-trivial and no-going work