close all

fprintf(1,'\nStarting EMTVdemo:\n\n');

% Set the parameters for the test problem.

theta = round(0:2:180); % No. of used angles.

eta = 0.05; % Relative noise level.

% No. of iterations.

k = 130; % for Main loop

l = 130; % for TV mehotde

t= clock;

% image load

Img = imread('lena.jpg');

[t1 t2 t3]=size(Img);

if t3==3

Img = rgb2gray(Img);

end

% Img = imresize(Img,[256 256]);

Img = im2double(Img);

[sz1 sz2] = size(Img);

N = sz1; % The discretization points

p=2\*N; % No. of parallel rays.

% Create the test problem.

[A,b\_ex,x\_ex] = EMparalleltomo(N,theta,p);

% b = b\_ex;

x\_ex = Img(:);

b\_ex = A\*x\_ex;

b = b\_ex;

fprintf(1,'Creating a parallel-bema tomography test problem\n');

fprintf(1,'with N = %2.0f, theta = %1.0f:%1.0f:%3.0f, and p = %2.0f.',...

[N,theta(1),theta(2)-theta(1),theta(end),p]);

% Noise level.

% e = poissrnd(eta,size(b\_ex));

% e = e/norm(e); % normalization

% b = b\_ex + e;

% Show the exact solution.

figure

imagesc(reshape(x\_ex,N,N)), colormap gray,

axis image off

c = caxis;

title('Exact phantom')

fprintf(1,'\n\n');

fprintf(1,'Perform k = %2.0f iterations with EMTV''s method.',k);

fprintf(1,'\nThis takes a moment ...');

% Perform the kaczmarz iterations.

Xkacz = EM\_TV(A,b,k,l);

% Show the kaczmarz solution.

figure

imagesc(reshape(Xkacz,N,N)), colormap gray,

axis image off

caxis(c);

title('EM\_TV reconstruction')

t4 = clock;

etime(t4,t);

function [X,info] = EM\_TV(A,b,K,l)

[m,n] = size(A);

% A = A'; % Faster to perform sparse column operations.

% Check the number of inputs.

if nargin < 3

error('Too few input arguments')

end

% Default value of starting vector x0.

x0 = ones(n,1);

% The sizes of A, b and x must match.

if size(b,1) ~= m || size(b,2) ~= 1

error('The sizes of A and b do not match')

elseif size(x0,1) ~= n || size(x0,2) ~= 1

error('The size of x0 does not match the problem')

end

% Initialization.

if isempty(K)

error('No stopping rule specified')

end

kmax=K;

% parameter setting

a = 0.8;

b = 0.25;

% Initialization before iterations.

xk = x0;

% normAi = full(abs(sum(A.\*A,1))); % Remember that A is transposed.

% I = find(normAi>0);

stop = 0;

k = 0;

xk1 =x0;

Lcon = ones(m,1);

while ~stop

disp(k);

k = k + 1;

% The EM algorithm

t1 = A'\*(b./(A\*xk));

t2 = (A'\*Lcon);

xk1 = xk.\*t1./t2;

xkTV = xk1;

if k<150

a=0.40;

else

a =0.45;

end

% TV method

for i=1:l

dxk = EMImgTV(xkTV);

vst = dxk/norm(dxk);

xkTV = xkTV+a\*vst.\*xkTV./(A'\*Lcon);

end

% Stopping rules.

if k >= kmax

stop = 1;

info = [0 k];

else

xk = xkTV;

end

end

X = xk;

function X = EMImgTV(img)

m = length(img);

N = sqrt(m);

e = 1e-8;

for col=2:N-1

for row=2:N-1

x1 = (img((row)\*N+col) - img((row-1)\*N+col))/ ...

sqrt(e+(img(row\*N+col) - img((row-1)\*N+col)).^2+(img((row-1)\*N+col+1) - img((row-1)\*N+col)).^2);

x2 = (img((row-1)\*N+col) - img((row-2)\*N+col))/ ...

sqrt(e+(img((row-1)\*N+col) - img((row-2)\*N+col)).^2+(img((row-2)\*N+col+1) - img((row-2)\*N+col)).^2);

x3 = (img((row-1)\*N+col+1) - img((row-1)\*N+col))/ ...

sqrt(e+(img((row)\*N+col) - img((row-1)\*N+col)).^2+(img((row-1)\*N+col+1) - img((row-1)\*N+col)).^2);

x4 = (img((row-1)\*N+col) - img((row-1)\*N+col-1))/ ...

sqrt(e+(img((row)\*N+col-1) - img((row-1)\*N+col-1)).^2+(img((row-1)\*N+col) - img((row-1)\*N+col-1)).^2);

X((row-1)\*N+col) = x1-x2+x3-x4;

end

end

% X=X(:);