EE 610 Image Processing

Project 1: Restoring the brain image

Name: Swrangsar Basumatary   Roll: 09d07040

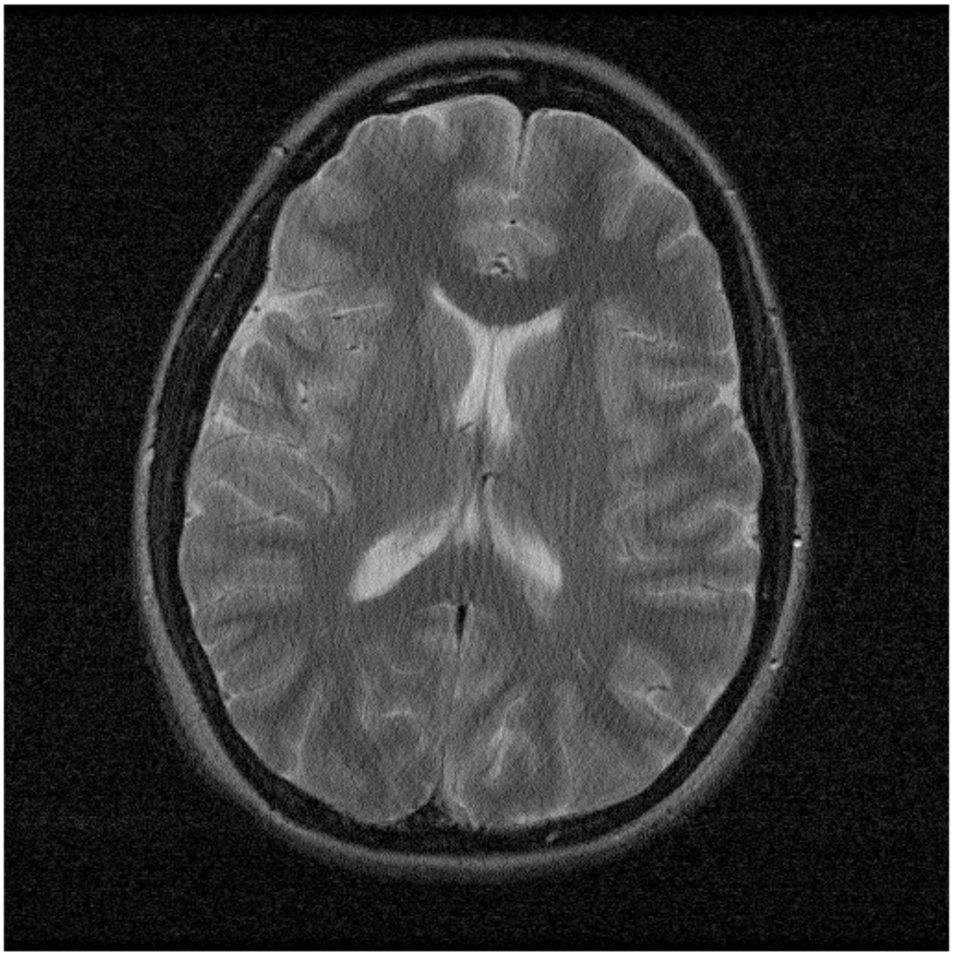
Convergence Values (brain image):

a) the objective function with all penalty weights assigned as zero:
obj: 11121152638.524910 goes to obj: 0.000000
b) only TV function with penalty = 0.77:
obj: 18503.321461 goes to obj: 355.482832
c) only FOV function with penalty = 1:
obj: 13.574067 goes to obj: 0.000000
d) all the functions together
obj: 4013397787.048397 goes to obj: 7349470.815444

Convergence Values (phantom image):

a) the objective function with all penalty weights assigned as zero:
obj: 11861156389.979265 goes to obj: 0.000000
b) only TV function with penalty = 5:
obj: 76851.425669 goes to obj: 2422.384663
c) only FOV function with penalty = 1:
obj: 12.563048 goes to obj: 0.000000
d) all the functions together
obj: 13250599996.589382 goes to obj: 75965666.847693

The corrupted brain image that is to be filtered is:



The fnlCg code:

```
function x = fnlCg(x0,sampler,data, param)
%----------------------------------------------------------------
---------
%----------------------------------------------------------------
-----------
x = x0;

% line search parameters - Dont touch..leave alone
maxlsiter = 150;
gradToll = 1.0000e-030;
alpha = 0.0100;
beta = 0.6000;
t0 = 1;
Itnlim = 16;
```

```
k = 0;

% copmute g0  = grad(Phi(x))
g0 = wGradient(x,sampler,data, param);

dx = -g0;

% iterations
while(1)

% backtracking line-search

    % pre-calculate values, such that it would be cheap to
compute the objective
    % many times for efficient line-search
    f0 = objective(x,dx, 0, sampler,data, param);
    t = t0;

        [f1]  =  objective(x,dx, t,sampler,data, param);

    lsiter = 0;

    while (f1 > f0 - alpha*t*abs(g0(:)'*dx(:)))^2 &
(lsiter<maxlsiter)
        lsiter = lsiter + 1;
        t = t * beta;
        [f1]  =  objective(x,dx, t,sampler,data, param);
    end

    if lsiter == maxlsiter
        disp('Reached max line search,.... not so good... might
have a bug in operators. exiting... ');
        return;
    end

    % control the number of line searches by adapting the
initial step search
    if lsiter > 2
        t0 = t0 * beta;
    end

    if lsiter<1
        t0 = t0 / beta;
    end
```

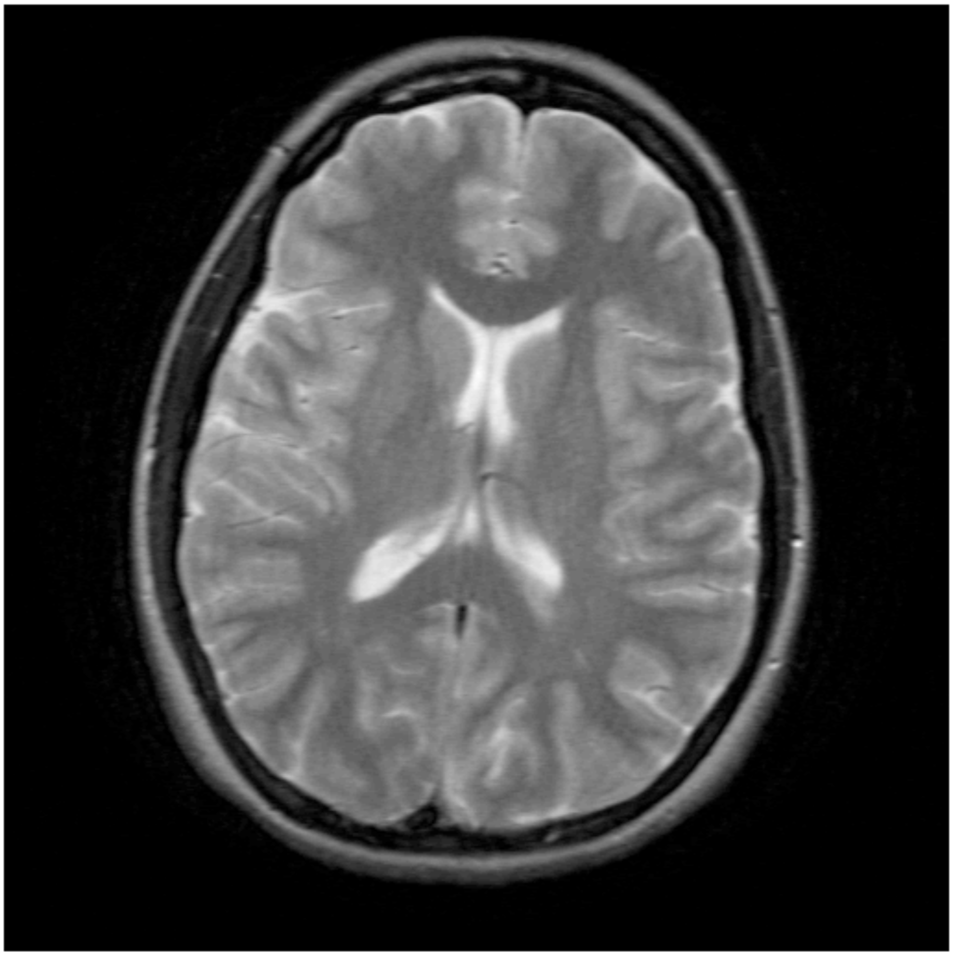
```
    x = (x + t*dx);

    %--------- uncomment for debug purposes
------------------------
    disp(sprintf('%d   , obj: %f ', k,f1));

%----------------------------------------------------------------

    %conjugate gradient calculation- Dont touch

    g1 = wGradient(x,sampler,data, param);
    bk = g1(:)'*g1(:)/(g0(:)'*g0(:)+eps);
    g0 = g1;
    dx =  - g1 + bk* dx;
    k = k + 1;

    %TODO: need to "think" of a "better" stopping criteria ;-)
    if (k > Itnlim) | (norm(dx(:)) < gradToll)
        break;
    end

end


return;


function [res] = objective(x,dx,t,sampler,data, param)
%DEFINE obj
x = x + (t * dx);
b = data;
Ax = sampler .* fftshift(fft2(fftshift(x)));
obj = (Ax - b);
res=(obj(:)'*obj(:)) + (param.TVWeight * TV(x)) +
(param.FOVWeight * fov(x));

function grad = wGradient(x,sampler,data, param)
%Define this function
gradObj=gOBJ(x,sampler,data);
grad = (gradObj) + (param.TVWeight * gTV(x)) + (param.FOVWeight
* gradFOV(x));

function gradObj = gOBJ(x,sampler,data)
% computes the gradient of the data consistency
%DEFINE gradObj
b = data;
```

```
Ax = sampler .* fftshift(fft2(fftshift(x)));
AhAx = ifftshift(ifft2(ifftshift(Ax)));
Ahb = ifftshift(ifft2(ifftshift(b)));
gradObj = 2 * (AhAx - Ahb);


function gradTV = gTV(x)
% compute gradient of TV operator
gradTV=filter2([0 -1 1],filter2([1 -1 0], x))
+filter2([0;-1;1],filter2([1;
-1; 0], x));

%YOU MAY WANT TO ADD MORE FUNCTIONS AND GRADIENTS
function totalvariation = TV(x)
ux = filter2([1 -1 0], x);
uy = filter2([1; -1; 0], x);
ux2 = ux .* (conj(ux));
uy2 = uy .* (conj(uy));
mag = sqrt(ux2 + uy2);
totalvariation = sum(mag(:)) ;


function fovMask = fovMask(data)
fieldRadius = 240/512;
[rows, cols] = size(data);
% X and Y matrices with ranges normalised to +/- 0.5
x =  (ones(rows,1) * [1:cols]  - (fix(cols/2)+1))/cols;
y =  ([1:rows]' * ones(1,cols) - (fix(rows/2)+1))/rows;
radius = sqrt(x.^2 + y.^2);        % A matrix with every pixel =
radius relative to centre.
i = radius < fieldRadius;
data(i) = 0;
fovMask = data;


function fov = fov(x)
x = fovMask(x);
mag = abs(x) .^ 2;
fov = sum(mag(:));

function gradFOV = gradFOV(x)
x = fovMask(x);
gradFOV = 2*x;
```

The brain demo code:

```
close all; clear all;

load brain512

sampler=mask./pdf;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reconstruction Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N = size(data);      % image Size
DN = size(data);     % Fourier data Size
param.TVWeight = 0.77;  % Weight for TV penalty
param.FOVWeight = 1;

% scale data
im_dc = ifftshift(ifft2(ifftshift(data.*sampler))); % matrix E
has been defined here
data = data/max(abs(im_dc(:)));

im_dc = im_dc/max(abs(im_dc(:)));

res = im_dc;  %Initial degraded image supplied to fnlcg function
figure(300), imshow(abs(res), []);

% do iterations
tic
for n=1:5
    res = fnlCg(res,sampler,data, param);  %initialize fnlcg
    im_res = res;
    figure(100), imshow(abs(im_res),[]), drawnow
end
toc
```

The output brain image is:



The Phantom image part

The corrupted phantom image that is to be filtered is:

The phantom demo code:

```
close all; clear all;

load brain512
originalImage = phantom('Modified Shepp-Logan', 512);

sampler = mask./pdf;
data = mask .* fftshift(fft2(fftshift(originalImage)));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reconstruction Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
N = size(data);      % image Size
DN = size(data);     % Fourier data Size
param.TVWeight = 5; % Weight for TV penalty
param.FOVWeight = 1;


% scale data
im_dc = ifftshift(ifft2(ifftshift(data.*sampler))); % matrix E
has been defined here
data = data/max(abs(im_dc(:)));

im_dc = im_dc/max(abs(im_dc(:)));

res = im_dc;  %Initial degraded image supplied to fnlcg function

figure(300), imshow(abs(res), []);
% do iterations
tic
for n=1:5
    res = fnlCg(res,sampler,data, param);  %initialize fnlcg
    im_res = res;
    figure(100), imshow(abs(im_res),[]), drawnow
end
toc
```

The output phantom image is: