Matlab Code for Poisson Image Reconstruction from Image Gradients

```
% Read Input Gray Image
imgstr = 'test.png'; disp(sprintf('Reading Image %s',imgstr));
img = imread(imgstr);      [H,W,C] = size(img);       img = double(img);
% Find gradinets
gx = zeros(H,W); gy = zeros(H,W);       j = 1:H-1; k = 1:W-1;
gx(j,k) = (img(j,k+1) - img(j,k));      gy(j,k) = (img(j+1,k) - img(j,k));

% Reconstruct image from gradients for verification
img_rec = poisson_solver_function(gx,gy,img);

figure;imagesc(img);colormap gray;colorbar;title('Image')
figure;imagesc(img_rec);colormap gray;colorbar;title('Reconstructed');
figure;imagesc(abs(img_rec-img));colormap gray;colorbar;title('Abs error');
```

```
function [img_direct] = poisson_solver_function(gx,gy,boundary_image);
% function [img_direct] = poisson_solver_function(gx,gy,boundary_image)
% Inputs; Gx and Gy -> Gradients
% Boundary Image -> Boundary image intensities
% Gx Gy and boundary image should be of same size
[H,W] = size(boundary_image);
gxx = zeros(H,W);   gyy = zeros(H,W);
f = zeros(H,W);     j = 1:H-1;    k = 1:W-1;

% Laplacian
gyy(j+1,k) = gy(j+1,k) - gy(j,k);       gxx(j,k+1) = gx(j,k+1) - gx(j,k);
f = gxx + gyy;                          clear j k gxx gyy gyyd gxxd

% boundary image contains image intensities at boundaries
boundary_image(2:end-1,2:end-1) = 0;
disp('Solving Poisson Equation Using DST');    tic
j = 2:H-1;          k = 2:W-1;    f_bp = zeros(H,W);
f_bp(j,k) = -4*boundary_image(j,k) + boundary_image(j,k+1) +
      boundary_image(j,k-1) + boundary_image(j-1,k) + boundary_image(j+1,k);
clear j k

f1 = f - reshape(f_bp,H,W);% subtract boundary points contribution
clear f_bp f

% DST Sine Transform algo starts here
f2 = f1(2:end-1,2:end-1);                    clear f1
%compute sine transform
tt = dst(f2);       f2sin = dst(tt')';       clear f2

%compute Eigen Values
[x,y] = meshgrid(1:W-2,1:H-2);    denom = (2*cos(pi*x/(W-1))-2) + (2*cos(pi*y/(H-1)) - 2) ;

%divide
f3 = f2sin./denom;                           clear f2sin x y

%compute Inverse Sine Transform
tt = idst(f3);      clear f3;     img_tt = idst(tt')';       clear tt

time_used = toc;    disp(sprintf('Time for Poisson Reconstruction = %f secs',time_used));

% put solution in inner points; outer points obtained from boundary image
img_direct = boundary_image;
img_direct(2:end-1,2:end-1) = 0;
img_direct(2:end-1,2:end-1) = img_tt;
```

```
function b=dst(a,n)
%DST    Discrete sine transform            (Used in Poisson reconstruction)
%      Y = DST(X) returns the discrete sine transform of X.
%      The vector Y is the same size as X and contains the
%      discrete sine transform coefficients.
%      Y = DST(X,N) pads or truncates the vector X to length N
%      before transforming.
%      If X is a matrix, the DST operation is applied to each
%      column. This transform can be inverted using IDST.

error(nargchk(1,2,nargin));

if min(size(a))==1
    if size(a,2)>1
        do_trans = 1;
    else
        do_trans = 0;
    end
    a = a(:);
else
    do_trans = 0;
end
if nargin==1,       n = size(a,1);      end
m = size(a,2);

% Pad or truncate a if necessary
if size(a,1)<n,
  aa = zeros(n,m);         aa(1:size(a,1),:) = a;
else
  aa = a(1:n,:);
end

y=zeros(2*(n+1),m); y(2:n+1,:)=aa;             y(n+3:2*(n+1),:)=-flipud(aa);
yy=fft(y);                 b=yy(2:n+1,:)/(-2*sqrt(-1));

if isreal(a), b = real(b); end
if do_trans, b = b.'; end
```

```
function b=idst(a,n)
%IDST   Inverse discrete sine transform    (Used in Poisson reconstruction)
%
%      X = IDST(Y) inverts the DST transform, returning the
%      original vector if Y was obtained using Y = DST(X).
%      X = IDST(Y,N) pads or truncates the vector Y to length N
%      before transforming.
%      If Y is a matrix, the IDST operation is applied to
%      each column.

if nargin==1
  if min(size(a))==1
    n=length(a);
  else
    n=size(a,1);
  end
end

nn=n+1;             b=2/nn*dst(a,n);
```