1. Objective

With at least three images with various directions of light-source, we can reconstruct the images by putting them certain coordinates of input light-source. Every algorithm is from the basis of the paper "What is the Set of Images of an Object under All Possible Illumination Conditions?" Belhumeur and Kriegman 1998.

2. Trained Image

I trained six images to making J vector (640*425*3 (N) by 6(M) matrix) by varying the direction of light-source.

2. Generating U matrix then get B matrix

Using the fact that the eigen-vectors of NxN matrix of J*Jt could be given from premultiplying the eigen vectors of the MxM matrix Jt*J by J, we can find the B matrix from multiplying J by u matrix from SVD of Jt*J. Then we choose the first 3 meaningful column of B then, B can be assigned by Nx3 matrix. So we can generate the new images which are given by certain light-source s (I = b(u,v)*s).

Before generating new images, we can also check(see) the basis images each along with the z, x, y axis by scaling or normalizing to 0 to 255 value of B vector. These are shown below.
### 3. Analysis

Generated images are shown at Fig 3. One thing we know that I notified the coordinate notification by z,x,y. As shown in above, when we choose the positive z axis (or point), the images became dark because of the light source being placed behind the object. So, every generated image which has positive z component looks dark. Further if we choose the larger value of the position of light-source, the image’s brightness will be bigger because value itself means the intensity of lightsource.

**Fig 3. Generated Image along the manipulated light sources**

Notation : (z,x,y)
TrainedImg5 = imread('5.jpg');
TrainedImg6 = imread('6.jpg');
NoTrainedImg = 6;
TrainedJv = [ TrainedImg1(:) TrainedImg2(:) TrainedImg3(:) TrainedImg4(:) TrainedImg5(:) TrainedImg6(:) ];
dTrainedJv = cast(TrainedJv, 'double');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note(2) First method using by svds
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [u,s,v] = svds(dTrainedJv); % It works M>=n --> svd == svds !!!
% Take only first 3 columns
% TempB = u(:, 1:3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note(3) Second method using what TA mentioned ( using eig_v of J'J )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:3
cNorm{i} = norm(dTrainedJv(:,i)); % just for testing
end
% let JtJ  J = dTrainedJv
JtJ = dTrainedJv' * dTrainedJv; % J'*J
[ U, V, S ] = svd(JtJ);
Ju = dTrainedJv * U; % J*U

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note(4) normalize by ratio  JUST ONLY FOR SHOW THE BASIS IMAGE!!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ju = Ju(:, 1:3);
for i = 1:3
uMax{i} = max(Ju(:,i));
singleMax{i} = max(uMax{i});
uMin{i} = min(Ju(:,i));
singleMin{i} = min(uMin{i});
Interval{i} = singleMax{i} - singleMin{i};
end
u(:, 1) = ((Ju(:, 1) - singleMin{1}) / Interval{1})*255;
u(:, 2) = ((Ju(:, 2) - singleMin{2}) / Interval{2})*255;
u(:, 3) = ((Ju(:, 3) - singleMin{3}) / Interval{3})*255;
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note(5) Choosing the first 3 columns of Ju ( forming B )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TempB = u(:, 1:3); % just for making basis images
TempB = Ju(:, 1:3); % just for generate images

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note(6) If we want to see basis image
% s1 = [ 1, 0, 0 ]; % basis image
s2 = [0, 1, 0]; % basis image
s3 = [0, 0, 1]; % basis image
s4 = [0, 0, 0];
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note(7) If we want to generate an image by specific light sources
% manual manipulation :
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s1 = [0.25, 0, 4.25];
s2 = [1.0, 0, 0];
s3 = [0, 0, -0.3];
s4 = [-1.5, 0, 3.0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the images !
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OutputImg{1} = TempB*s1';
OutputImg{2} = TempB*s2';
OutputImg{3} = TempB*s3';
OutputImg{4} = TempB*s4';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% normalize negative to 0 and exceed 255 to be 255
%m, n = size(TempB);
for j = 1:m
    if OutputImg{1}(j,1) < 0
        OutputImg{1}(j,1) = 0;
    end
    if OutputImg{1}(j,1) > 255
        OutputImg{1}(j,1) = 255;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Image show
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nImageRow = 425;
nImageCol = 640;
OutputImgUnit8re{1} = reshape(uint8(OutputImg{1}),[nImageRow, nImageCol, 3]);
OutputImgUnit8re{2} = reshape(uint8(OutputImg{2}),[nImageRow, nImageCol, 3]);
OutputImgUnit8re{3} = reshape(uint8(OutputImg{3}),[nImageRow, nImageCol, 3]);
OutputImgUnit8re{4} = reshape(uint8(OutputImg{4}),[nImageRow, nImageCol, 3]);
imshow(OutputImgUnit8re{1});
%imshow(OutputImgUnit8re{2});
%imshow(OutputImgUnit8re{3});
%imshow(OutputImgUnit8re{4});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Just for test ( multiple view )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j = 1:4
    subplot(2,2,j);
    imshow(OutputImgUnit8re{j});
end
end