**Report**

Author: Zhengqi Yang

Date: 11/12/2019

**Part 1: RMSE vs Number of Coefficients kept Graphs**

**NOTE: The Coefficients chosen were 10,000, 7,000, 4,000, and 1,000 for these two methods**

**KLT: See Figure 1**

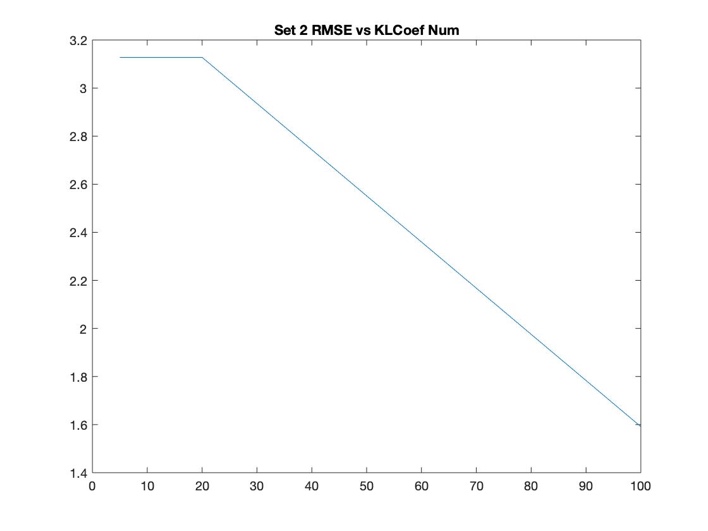
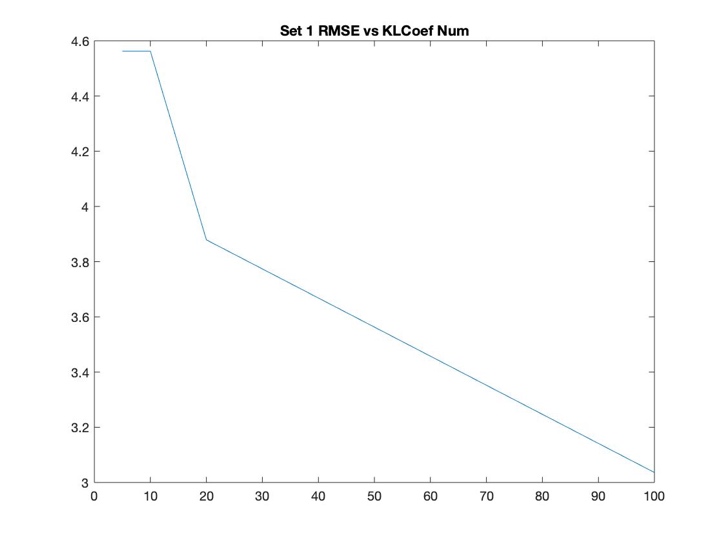


Figure 1: The left plot is for set 1 and the right one is for set 2.

**2D-DFT: See Figure 2**

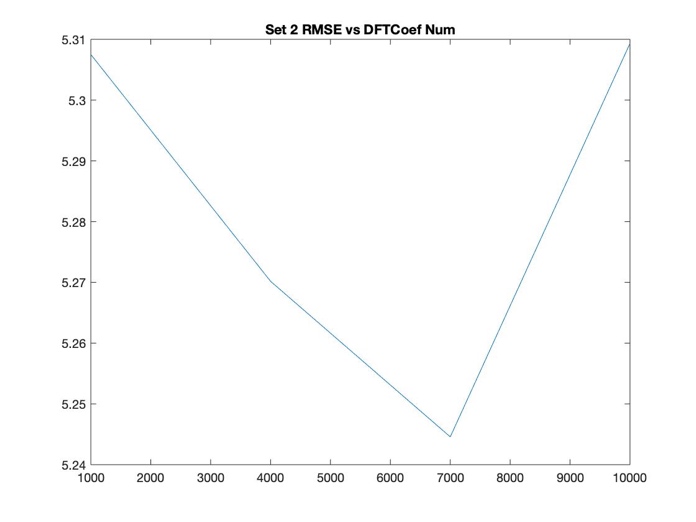
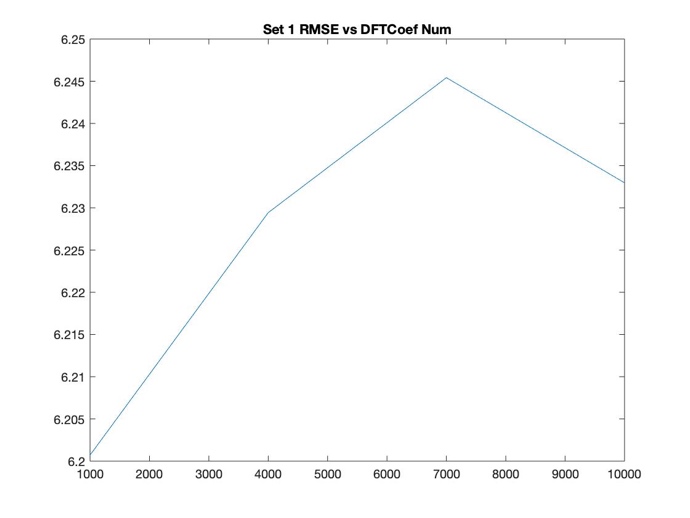


Figure 2: The left plot is for set 1 and the right one is for set 2.

**2D-DCT: See Figure 3**

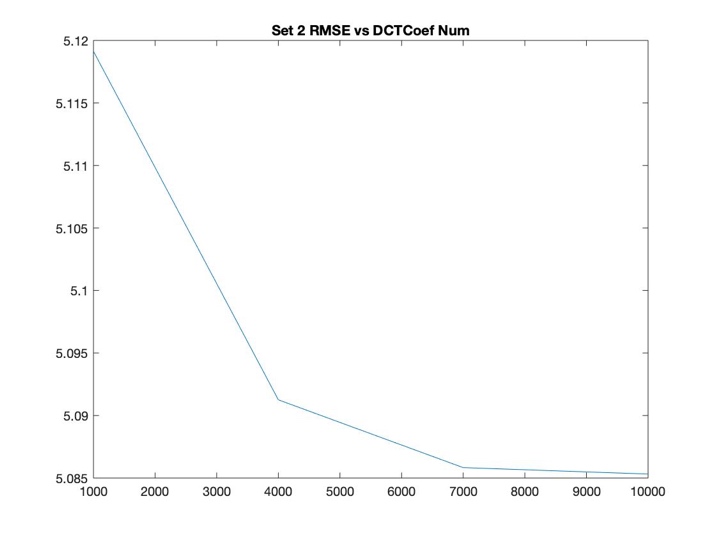
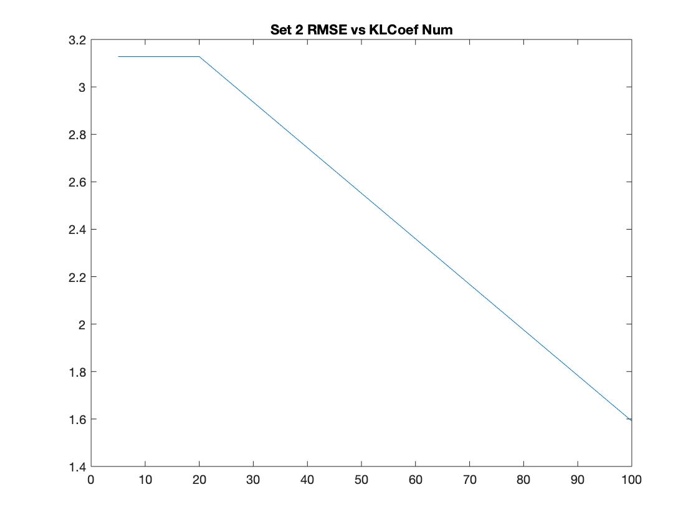
****

Figure 3: The left plot is for set 1 and the right one is for set 2.

|  |  |  |  |
| --- | --- | --- | --- |
| SET 1 | KLT | DFT | DCT |
| 10,000/ 100(KLT) | 3.0362 | 6.2330 | 6.1770 |
| 7,000 / 20(KLT) | 3.8789 | 6.2454 | 6.1773 |
| 4,000/ 10(KLT) | 4.5622 | 6.2294 | 6.1807 |
| 1,000/ 5(KLT) | 4.5622 | 6.2007 | 6.2040 |

|  |  |  |  |
| --- | --- | --- | --- |
| SET 2 | KLT | DFT | DCT |
| 10,000/ 100(KLT) | 1.5915 | 5.3093 | 5.0853 |
| 7,000 / 20(KLT) | 3.1277 | 5.2446 | 5.0858 |
| 4,000/ 10(KLT) | 3.1277 | 5.2702 | 5.0913 |
| 1,000/ 5(KLT) | 3.1277 | 5.3075 | 5.1191 |

**Part 2: Implementing KLT**

**See Appendix A**

**Part 3: Reconstruct with M top Coefficient – KLT**

In this part, the last image from each set (‘'Nikon\_D70s\_1\_23105.png' – set 1’ and ‘'Nikon\_D70s\_1\_23093.png' – set 2’) was chosen. The reconstructed images with full coefficients were shown in Figure 4. The Karhunen-Loeve Decomposition for Statistical Recognition and Detection algorithm was applied in this part. [1] Codes see Appendix A.

Figure 4: The left one is 'Nikon\_D70s\_1\_23105.png' from set 1 and the right one is 'Nikon\_D70s\_1\_23093.png' from set 2.

The reconstructed images with difference coefficients for both example images were shown in Figure 5. The RMSE vs Number of Coefficients graphs were shown in part 1.

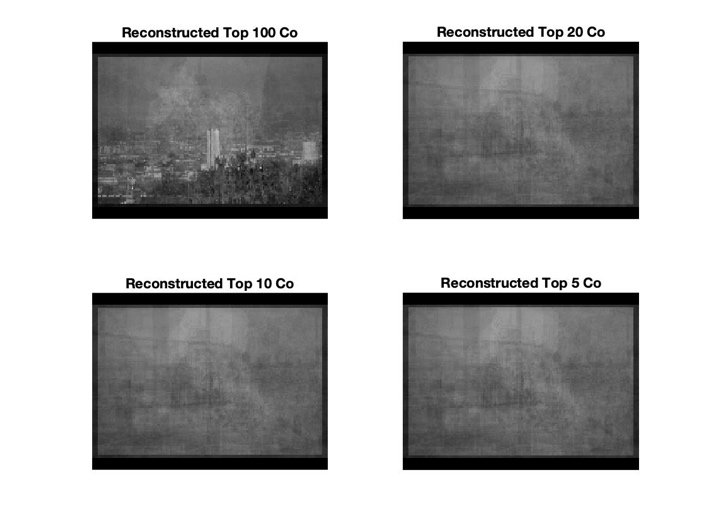
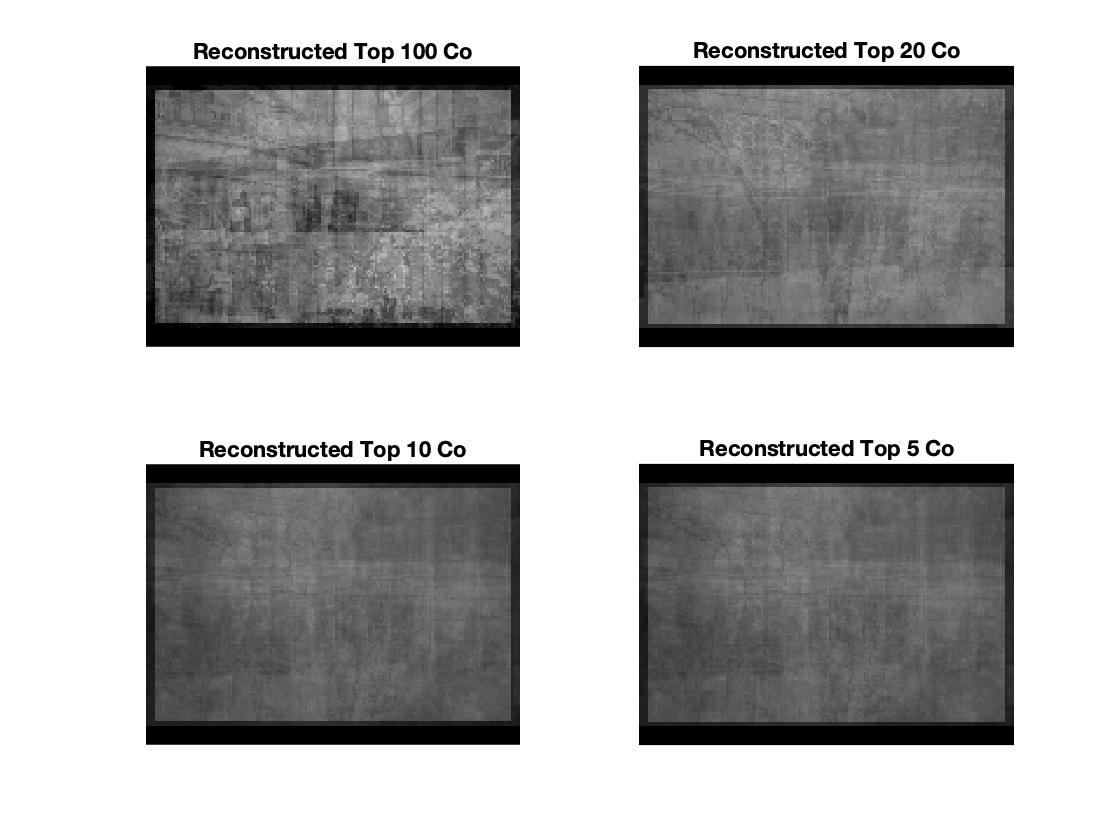


Figure 5: The left one is from set 1 and the right one is from set 2.

**Part 4: Reconstruct with M top Coefficient – DFT and DCT**

**NOTE: The Coefficients chosen were 10,000, 7,000, 4,000, and 1,000 for these two methods**

* **DFT**

In this part, fft2, ifft2 were the main functions used. The same example image from both sets was used in order to have a better comparison. The reconstructed images with full coefficients were shown in Figure 6. Codes see Appendix B[2]

Figure 6: The left one is from set 1 via DFT and the right one is from set 2 via DFT.

The reconstructed images with different coefficients were shown in Figure 7. The RMSE vs Number of Coefficients graphs were shown in part 1.

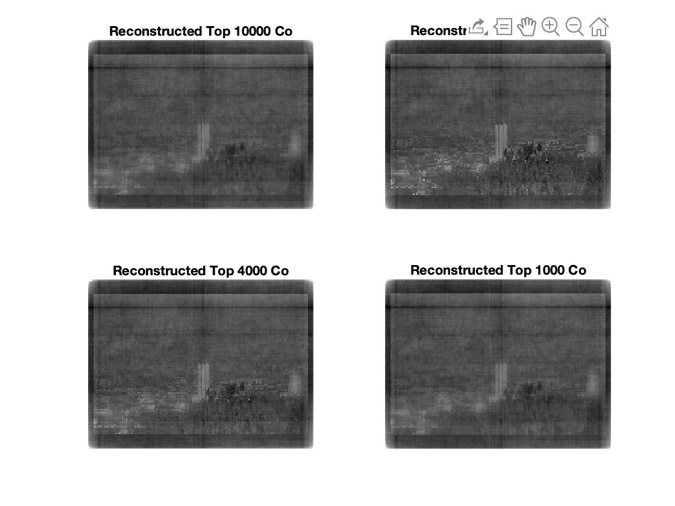
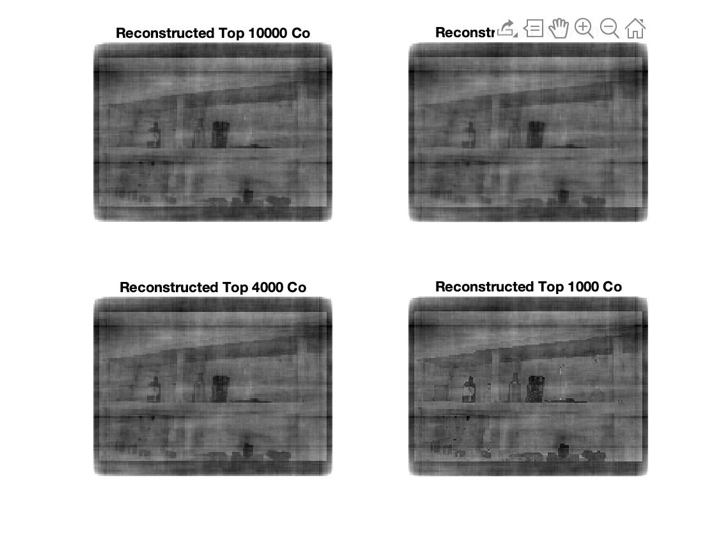


Figure 7: The left one is from set 1 and the right one is from set 2.

* **DCT**

Similarly, dct2, and idct2 were the main functions used. The same example image from both sets was used in order to have a better comparison. The reconstructed images with full coefficients were shown in Figure 8. Codes see Appendix C[2] 

Figure 8: The left one is from set 1 via DCT and the right one is from set 2 via DCT.

The reconstructed images with different coefficients were shown in Figure 9. The RMSE vs Number of Coefficients graphs were shown in part 1.

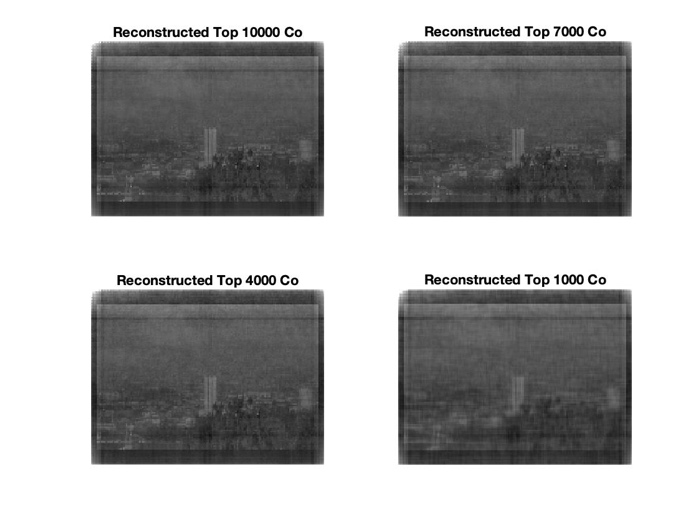
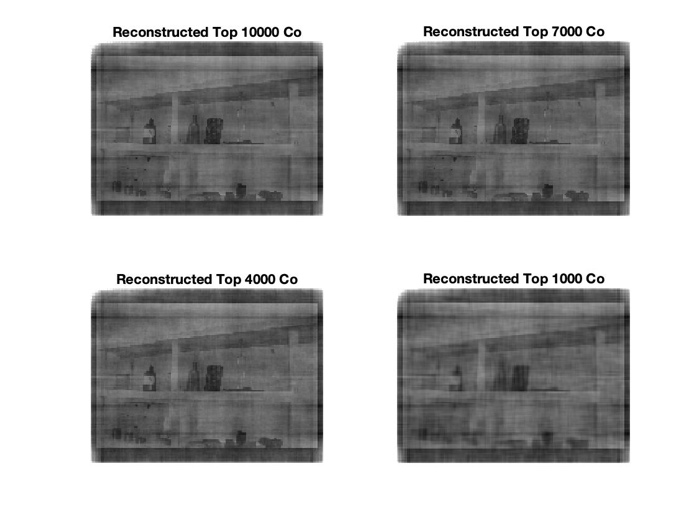


Figure 9: The left one is from set 1 and the right one is from set 2.

**Reference**

[1] A. Chirokov, *Karhunen-Loeve Decomposition for Statistical Recognition and Detection*, <https://www.mathworks.com/matlabcentral/fileexchange/6995-karhunen-loeve-decomposition-for-statistical-recognition-and-detection>

[2] <https://blog.csdn.net/IMWTJ123/article/details/79831242>

**NOTE: Since it worked same for both sets, I only attached code for set 1.**

**Appendix A**

clear

clc

close all

% Load the image set\_1

set\_1 = dir('set\_1');

len = length(set\_1);

k = 0;

for i = 3:len

filename = set\_1(i).name;

img = imread(filename);

k = k + 1;

x(:,k) = img(:);

end

nImages = k; %total number of images

imsize = size(img); %size of image (they all should have the same size)

nPixels = imsize(1)\*imsize(2); %number of pixels in image

x = double(x)/255; %convert to double and normalize

%Calculate the average

avrgx = mean(x')';

for i=1:nImages

x(:,i) = x(:,i) - avrgx; % substruct the average

end

%compute covariance matrix

cov\_mat = x'\*x;

[V,D] = eig(cov\_mat); %eigen values of cov matrix

V = x\*V\*(abs(D))^-0.5;

%image decomposition coefficients

KLCoef = x'\*V;

%top 100 coefficients

list100 = KLCoef(:);

list100 = list100';

sortl = sort(list100);

gate = sortl(301);

KLCoef100 = KLCoef;

for i = 1:20

for j = 1:20

if KLCoef100(i,j) < gate

KLCoef100(i,j) = 0;

end

end

end

%top 20 coefficients

list20 = KLCoef(:);

list20 = list20';

sortl = sort(list20);

gate = sortl(381);

KLCoef20 = KLCoef;

for i = 1:20

for j = 1:20

if KLCoef20(i,j) < gate

KLCoef20(i,j) = 0;

end

end

end

%top 10 coefficients

list10 = KLCoef(:);

list10 = list10';

sortl = sort(list10);

gate = sortl(391);

KLCoef10 = KLCoef;

for i = 1:20

for j = 1:20

if KLCoef10(i,j) < gate

KLCoef10(i,j) = 0;

end

end

end

%top 5 coefficients

list5 = KLCoef(:);

list5 = list5';

sortl = sort(list5);

gate = sortl(396);

KLCoef5 = KLCoef;

for i = 1:20

for j = 1:20

if KLCoef5(i,j) < gate

KLCoef5(i,j) = 0;

end

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% reconstruction of Image

i = 1:20; %index of image to be recontructed

reconst = V\*KLCoef';

% NOTE: The following for loop is generating 20 reconstructed images and saving

% these images so that I commentted this part in order to avoid repeating

% doing the same task each time.

% % % for image\_index = i

% % %

% % % diff = abs(reconst(:,image\_index) - x(:,image\_index));

% % % figure;

% % % subplot(1,2,1); imshow((reshape(avrgx+reconst(:,image\_index), imsize))); title('Reconstructed');

% % % subplot(1,2,2); imshow((reshape(avrgx+x(:,image\_index), imsize)));title('original');

% % % saveas(gcf, sprintf('reconst set 1 %d.jpg', image\_index));

% % % close all

% % % end

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% RMSE vs Top 100 KLCoef, Img 20

reconst100 = V\*KLCoef100';

image\_index = 20;

diff = abs(reconst100(:,image\_index) - x(:,image\_index));

rmse100 = sqrt(sum(diff.^2)/k);

% RMSE vs Top 20 KLCoef, Img 20

reconst20 = V\*KLCoef20';

image\_index = 20;

diff = abs(reconst20(:,image\_index) - x(:,image\_index));

rmse20 = sqrt(sum(diff.^2)/k);

% RMSE vs Top 10 KLCoef, Img 20

reconst10 = V\*KLCoef10';

image\_index = 20;

diff = abs(reconst10(:,image\_index) - x(:,image\_index));

rmse10 = sqrt(sum(diff.^2)/k);

% RMSE vs Top 5 KLCoef, Img 20

reconst5 = V\*KLCoef5';

image\_index = 20;

diff = abs(reconst5(:,image\_index) - x(:,image\_index));

rmse5 = sqrt(sum(diff.^2)/k);

figure

subplot(2,2,1); imshow((reshape(avrgx+reconst100(:,image\_index), imsize))); title('Reconstructed Top 100 Co');

subplot(2,2,2); imshow((reshape(avrgx+reconst20(:,image\_index), imsize))); title('Reconstructed Top 20 Co');

subplot(2,2,3); imshow((reshape(avrgx+reconst10(:,image\_index), imsize))); title('Reconstructed Top 10 Co');

subplot(2,2,4); imshow((reshape(avrgx+reconst5(:,image\_index), imsize))); title('Reconstructed Top 5 Co');

%Plot RMSE vs KLCoef Num

rmse\_set\_1 = [rmse100, rmse20, rmse10, rmse5];

KLCoefnum = [100, 20, 10, 5];

figure

plot(KLCoefnum, rmse\_set\_1);

title('Set 1 RMSE vs KLCoef Num');

**Appendix B**

clear

clc

close all

% Load the image set\_1

set\_1 = dir('set\_1');

len = length(set\_1);

k = 0;

for i = 3:len

filename = set\_1(i).name;

img = imread(filename);

k = k + 1;

x(:,k) = img(:);

end

nImages = k; %total number of images

imsize = size(img); %size of image (they all should have the same size)

nPixels = imsize(1)\*imsize(2); %number of pixels in image

x = double(x)/255; %convert to double and normalize

%Calculate the average

avrgx = mean(x')';

for i=1:nImages

x(:,i) = x(:,i) - avrgx; % substruct the average

end

image\_index = 20; % Img

image\_2d = x(:,image\_index);

image\_2d = reshape(image\_2d+avrgx, imsize);

A=fft2(image\_2d);

figure(1);

imshow(abs(ifft2(A)));title('2D-DFT Reconst');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Calculate M-top coefficients

%top 10000 coefficients

list100 = A(:);

list100 = list100';

sortl = sort(list100);

gate = sortl(120\*160-9999);

A100 = A;

for i = 1:120

for j = 1:160

if A100(i,j) < gate

A100(i,j) = 0;

end

end

end

figure(2);

subplot(2,2,1); imshow(abs(ifft2(A100))); title('Reconstructed Top 10000 Co');

%top 7000 coefficients

list20 = A(:);

list20 = list20';

sortl = sort(list20);

gate = sortl(120\*160-6999);

A20 = A;

for i = 1:120

for j = 1:160

if A20(i,j) < gate

A20(i,j) = 0;

end

end

end

subplot(2,2,2); imshow(abs(ifft2(A20))); title('Reconstructed Top 7000 Co');

%top 4000 coefficients

list10 = A(:);

list10 = list10';

sortl = sort(list10);

gate = sortl(120\*160-3999);

A10 = A;

for i = 1:120

for j = 1:160

if A10(i,j) < gate

A10(i,j) = 0;

end

end

end

subplot(2,2,3); imshow(abs(ifft2(A10))); title('Reconstructed Top 4000 Co');

%top 1000 coefficients

list5 = A(:);

list5 = list5';

sortl = sort(list5);

gate = sortl(120\*160-999);

A5 = A;

for i = 1:120

for j = 1:160

if A5(i,j) < gate

A5(i,j) = 0;

end

end

end

subplot(2,2,4); imshow(abs(ifft2(A5))); title('Reconstructed Top 1000 Co');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Calculate RMSE

% RMSE vs Top 10000 DFTCoef, Img 20

diff = abs(double(abs(ifft2(A100)) - image\_2d));

rmse100 = sqrt(sum(diff(:).^2)/k);

% RMSE vs Top 7000 DFTCoef, Img 20

diff = abs(double(abs(ifft2(A20)) - image\_2d));

rmse20 = sqrt(sum(diff(:).^2)/k);

% RMSE vs Top 4000 DFTCoef, Img 20

diff = abs(double(abs(ifft2(A10)) - image\_2d));

rmse10 = sqrt(sum(diff(:).^2)/k);

% RMSE vs Top 1000 DFTCoef, Img 20

diff = abs(double(abs(ifft2(A5)) - image\_2d));

rmse5 = sqrt(sum(diff(:).^2)/k);

%Plot RMSE vs DFTCoef Num

rmse\_set\_1 = [rmse100, rmse20, rmse10, rmse5];

DFTCoefnum = [10000, 7000, 4000, 1000];

figure(3);

plot(DFTCoefnum, rmse\_set\_1);

title('Set 1 RMSE vs DFTCoef Num');

**Appendix C**

clear

clc

close all

% Load the image set\_1

set\_1 = dir('set\_1');

len = length(set\_1);

k = 0;

for i = 3:len

filename = set\_1(i).name;

img = imread(filename);

k = k + 1;

x(:,k) = img(:);

end

nImages = k; %total number of images

imsize = size(img); %size of image (they all should have the same size)

nPixels = imsize(1)\*imsize(2); %number of pixels in image

x = double(x)/255; %convert to double and normalize

%Calculate the average

avrgx = mean(x')';

for i=1:nImages

x(:,i) = x(:,i) - avrgx; % substruct the average

end

image\_index = 20; % Img

image\_2d = x(:,image\_index);

image\_2d = reshape(image\_2d+avrgx, imsize);

A=dct2(image\_2d);

figure(1);

imshow(abs(idct2(A)));title('2D-DCT Reconst');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Calculate M-top coefficients

%top 10000 coefficients

list100 = A(:);

list100 = list100';

sortl = sort(list100);

gate = sortl(120\*160-9999);

A100 = A;

for i = 1:120

for j = 1:160

if A100(i,j) < gate

A100(i,j) = 0;

end

end

end

figure(2);

subplot(2,2,1); imshow(abs(idct2(A100))); title('Reconstructed Top 10000 Co');

%top 7000 coefficients

list20 = A(:);

list20 = list20';

sortl = sort(list20);

gate = sortl(120\*160-6999);

A20 = A;

for i = 1:120

for j = 1:160

if A20(i,j) < gate

A20(i,j) = 0;

end

end

end

subplot(2,2,2); imshow(abs(idct2(A20))); title('Reconstructed Top 7000 Co');

%top 4000 coefficients

list10 = A(:);

list10 = list10';

sortl = sort(list10);

gate = sortl(120\*160-3999);

A10 = A;

for i = 1:120

for j = 1:160

if A10(i,j) < gate

A10(i,j) = 0;

end

end

end

subplot(2,2,3); imshow(abs(idct2(A10))); title('Reconstructed Top 4000 Co');

%top 1000 coefficients

list5 = A(:);

list5 = list5';

sortl = sort(list5);

gate = sortl(120\*160-999);

A5 = A;

for i = 1:120

for j = 1:160

if A5(i,j) < gate

A5(i,j) = 0;

end

end

end

subplot(2,2,4); imshow(abs(idct2(A5))); title('Reconstructed Top 1000 Co');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Calculate RMSE

% RMSE vs Top 10000 DCTCoef, Img 20

diff = abs(double(abs(idct2(A100)) - image\_2d));

rmse100 = sqrt(sum(diff(:).^2)/k);

% RMSE vs Top 7000 DCTCoef, Img 20

diff = abs(double(abs(idct2(A20)) - image\_2d));

rmse20 = sqrt(sum(diff(:).^2)/k);

% RMSE vs Top 4000 DCTCoef, Img 20

diff = abs(double(abs(idct2(A10)) - image\_2d));

rmse10 = sqrt(sum(diff(:).^2)/k);

% RMSE vs Top 1000 DCTCoef, Img 20

diff = abs(double(abs(idct2(A5)) - image\_2d));

rmse5 = sqrt(sum(diff(:).^2)/k);

%Plot RMSE vs DCTCoef Num

rmse\_set\_1 = [rmse100, rmse20, rmse10, rmse5];

DCTCoefnum = [10000, 7000, 4000, 1000];

figure(3);

plot(DCTCoefnum, rmse\_set\_1);

title('Set 1 RMSE vs DCTCoef Num');