Xi'an Jiaotong-Liverpool University
西交利物浦大學

EEE330

IMAGE PROCESSING

Lab 3 Report

Author:
Ruihao Wang

Student Number:
1405884

May 3, 2018

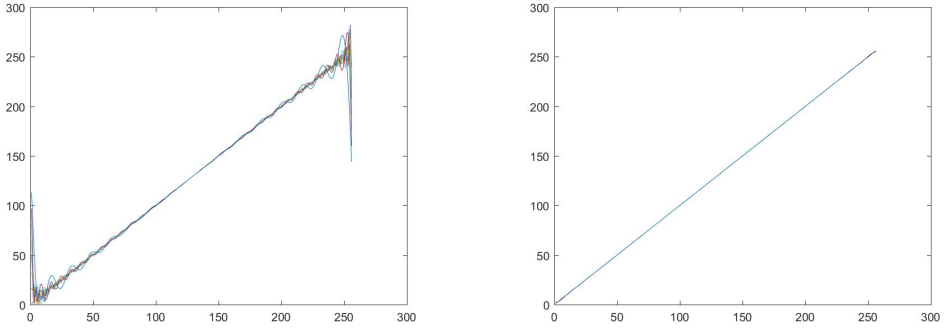
Contents

1 Introduction

This lab is a further study of image processing techniques. It is mainly about transformation of images which is transfer images from spatial domain to others for a easier processing implementation. Among schemes belong to this kind of technique, discrete Fourier transform (DFT) and discrete cosine transform (DCT) are two common ones in applications. The task of this lab, such as zero padding in frequency response center on them. By implement the processing as specification, the methodologies and meaning of DFT and DCT will be studied. In this report, the results of each task will be firstly reported and then several crucial points will be discussed as well. The full code of MATLAB implementation are collected in appendix.

2 Task 1: 1D DFT and DCT

This task is to apply both discrete Fourier transform (DFT) and discrete cosine transform (DCT) to a sequence ranging from 1 to 256. Then, inverse transforms are applied to recover original signal but various number of high frequency components will be removed to observe the impact. Such a removal is equivalent to applying a low pass filter to the transformed signal. Figure 1 shows the results of this task. Theoretically, both DFT and DCT converts original signal to the superposition of many cosine or sinusoid waves of various frequency but the results are quite different. It can be observed that DFT has more obvious distortion than DCT. As the ratio of removed components increases, the distortion tends to become even worse.



(a) DFT (b) DCT

Figure 1: Comparison of different transform schemes

Then, PSNR introduces a more intuitional evaluation of the reconstruction quality. The first row of Table 1 are the ratio of removed high frequency components while the following rows are calculated PSNR for both two reconstruction scheme. Generally, DCT has a better performance while the noise in reconstruction signal of DFT has even a higher level of original one. In

addition, PSNR decrease along with the increasing of removal ratio. Such removal of signal implies loss of information.

PSNR	32/256	64/256	96/256	128/256	160/256	192/256	224/256
DFT	-8.4948	-12.0352	-14.1475	-15.9394	-17.7054	-19.7659	-22.8945
DCT	57.8801	48.3201	42.0896	36.8363	31.5834	25.3545	15.8040

Table 1: PSNR of various number of components used for recovery

3 Task 2: Understanding 2D DFT

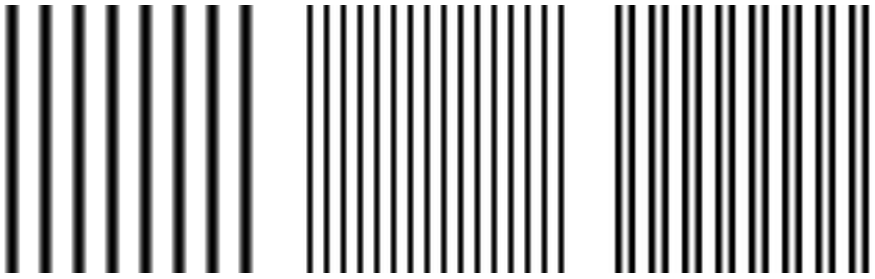
Different cosine patterns are generated on images as illustrated in Figure 2. Those patterns extend along horizontal axis with different period determined by N. This value specifies the number of pixels that construct an entire period of a cosine wave. In horizontal direction, the pixel values are assigned by equation below.

$$x[n] = 1 + cos(2\pi \frac{1}{N} n) \tag{1}$$

This implies N is entire 2π in angular while its reciprocal $\frac{1}{N}$ is the frequency of cosine pattern or the number of period within the width of image can be expressed as $\frac{256}{N}$. The expression of composed cosine signal is

$$x[n]_{comp} = 1 + cos(2\pi \frac{1}{N} n) + cos(4\pi \frac{1}{N} n) \tag{2}$$

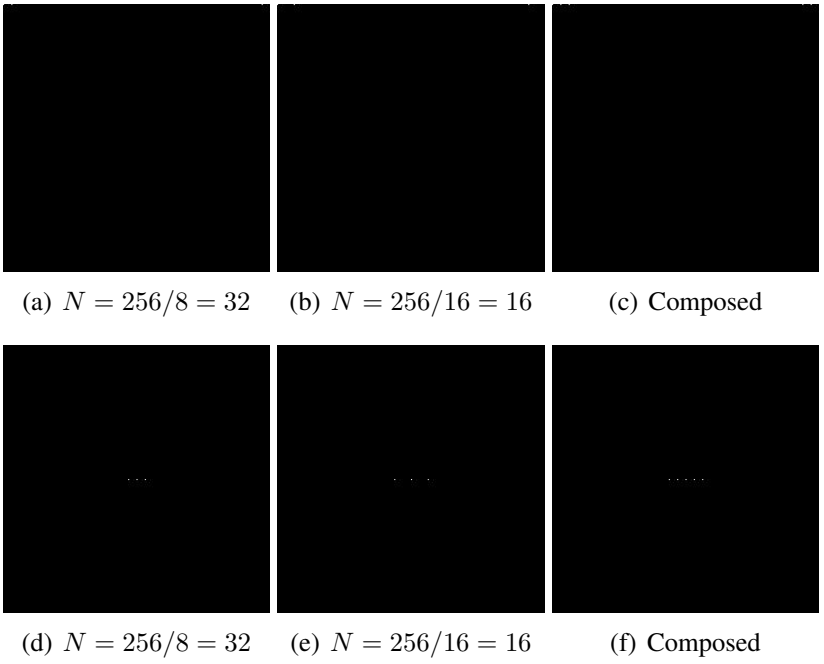
The first cosine term has frequency of $\frac{1}{N}$ while the second keeps $\frac{2}{N}$. In MATLAB code, the N is assigned value of 32 which means the composed pattern is the superposition of previous two single tone ones.



(a) $N = 256/8 = 32$ (b) $N = 256/16 = 16$ (c) Composed

Figure 2: Images of different cosine functions

After generation of cosine patterns, 2D-DFT is applied on images to transform them into frequency domain. To visualize the results, absolute values of transformed images are taken in log scale. The 'pixel values' in frequency domain are almost zero but, at two upper corner in figure 3(a) and 3(b), only three of them have a distinctive higher level. For the composed cosine pattern in figure 3(c), the number of significant pixels are 5 as it has two components with different period.



(a) $N = 256/8 = 32$ (b) $N = 256/16 = 16$ (c) Composed

(d) $N = 256/8 = 32$ (e) $N = 256/16 = 16$ (f) Composed

Figure 3: Apply `fft` and `fftshift`

Then, `fftshift` is applied to center the zero-frequency points as shown by the second row of figure 3. Such an operation makes the transformed images meaningful: the center of image is original points while vertical and horizontal dimension represents the frequencies of corresponding axises in original images. Using figure 3(a) and 3(d) as example, the original cosine pattern horizontally extends while it keeps constant in vertical direction. Thus, figure 3(d) has two components distribute in a row. Actually, the component at origin is meaningful on vertical axis rather than horizontal. It implies the original image is constant in vertical direction without any frequency of changing. The components indicates a DC value no matter for vertical or horizontal axis.

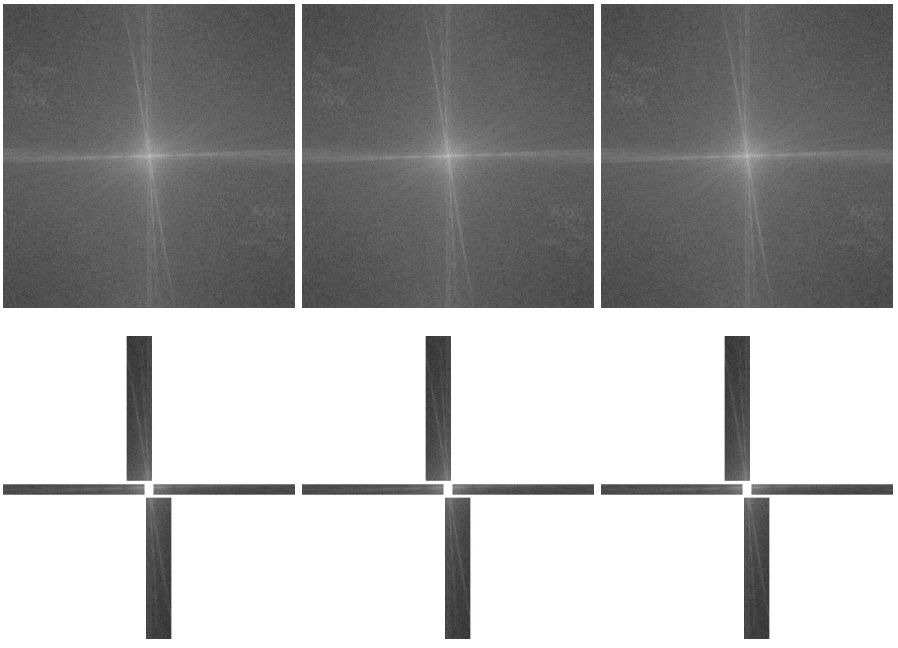
Comparing figure 3(d) and 3(e), the distance of two symmetric components are different. That exactly means they have difference frequency $\frac{1}{N}$: a larger N value leads a lower frequency and closer distance of symmetric components. As previously mentioned, the composed cosine pattern is the superposition of these two single tones. Thus, the frequency expression of image has 5 components.

4 Task 3: Elimination of fence

(a) Original image (b) Fence removed

Figure 4: Fence removal results

The aim of this task is to remove the fence in original image shown as figure 4(a). It is quite difficult to implement in spatial domain. Transformed into frequency domain, however, filtering out a portion of components can achieve this goal. As the original image has three channels (RGB) and their Fourier transforms are almost identical, the same mask is applied to each channel as shown in figure 5. Note that the dark 'cross' is the part of components that are filtered.



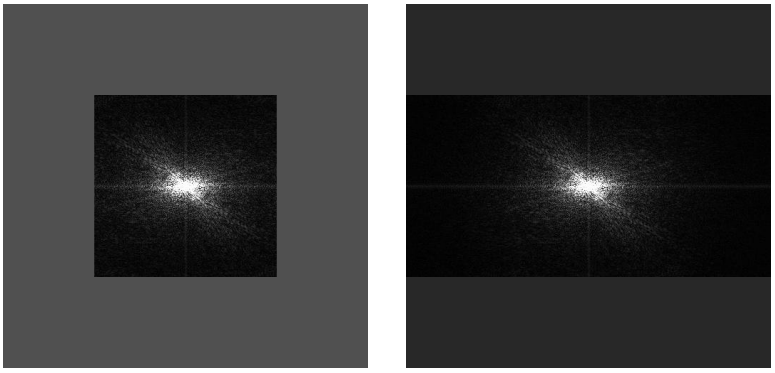
(a) R (b) G (c) B

Figure 5: FFT of RGB channels and mask for filtering

The fence extends along longitudinal direction are almost vertical while transverse ones are not horizontal obviously. For such a pattern, the frequency transform will holds a cross shape but the vertical band will slightly incline and approximately perpendicular to the transverse fence as shown in figure 5. Figure 4(b) is the result of filtering and reconstruction. Nevertheless, such a processing is quite coarse, it leads a remarkable loss of information and there are still some obvious vestige of fence.

5 Task 4: Zero-padding

This task introduce an image interpolation scheme by zero padding in images after transform them to frequency domain. There are two scales of images provided: (256, 256) and (512, 256). The objective of this task is to resize these two images into (512, 512) scale. According to the definition of DFT/IDFT, the padded zero values will not contribute the calculation of transform. Thus, more points for transformation results in a higher resolution to satisfy a larger scale. Figure 6 shows the results of zero-padding in frequency domain which represents by gray area surrounding the DFT transformation result.



(a) `im420` (b) `im440`

Figure 6: Zero-padding in frequency domain

	`im420`	`im440`
zero-padding	32.339	36.8108
`nearest`	28.2869	32.7515
`lanczos3`	29.9145	34.4147

Table 2: PSNR using different schemes of resize

Figure 12 indicates the interpolation results and compare them with ones from resize in spatial domain. In order to more precisely evaluate the performance, table 2 collects the PSNR of each

scheme. It can be concluded that frequency domain interpolation has a better performance as its PSNR is always higher than others for both images.

(a) Zero padding `im420` (b) `resize im420` (c) `lanczos3 im420`

(d) Zero padding `im440` (e) `resize im440` (f) `lanczos3 im440`

Figure 7: Results of different resize schemes

6 Task 5: 2D DCT

In this task, students are required to transfer 4×4 images into a domain that consists of basis derived according to 2-dimension discrete cosine transform (2D-DCT) whose expression is

$$F(k,l) = \alpha(k)\alpha(l) \sum_{n=0}^{N-1} \sum_{n=0}^{N-1} x(n,m) cos \left[\frac{(2n+1)l\pi}{2N} \right] cos \left[\frac{(2n+1)k\pi}{2N} \right] \tag{3}$$

where $k, l \in N$ and the coefficient $\alpha(k)$

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & , k = 0 \\ \sqrt{\frac{2}{N}} & , k \neq 0 \end{cases} \tag{4}$$

The 1D-DCT of a sequence with N elements has N cosine components with various frequency while equation (3) implies that 2D-DCT is equivalent to applying 1D-DCT respectively to

columns and rows of images. Therefore, generally for images with size of $N \times N$, their bases have the same size and the number of bases is N^2. In my own implementation, however, the direct use of definition function shown in equation (3) has a relatively lower speed. Thus, I finally use the method `dctmtx(N)` provided by toolbox. It returns a $N \times N$ transformation matrix $\mathbf{T}$ which can be used to implement DCT. The meaning of such a matrix will be discussed in later section and, here, the bases of 4×4 images are visualized in figure 8.

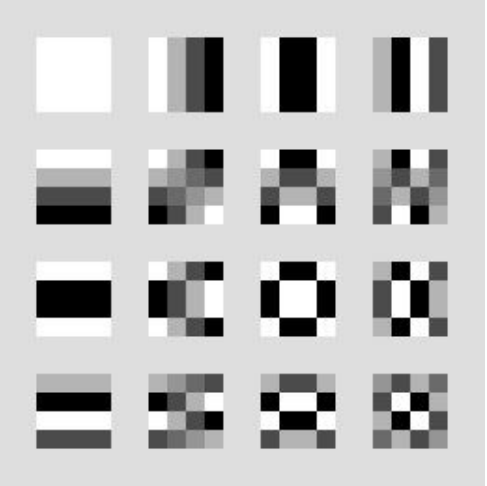


Figure 8: 2D-DCT bases generated

Instead of directly use `dct2` to apply 2D-DCT, it can be obtained with matrix $\mathbf{T}$ according to the matrix expression of 2D-DCT

$$\mathbf{F} = \mathbf{T}\mathbf{X}\mathbf{T}' \tag{5}$$

where $\mathbf{X}$ is original image and $\mathbf{F}$ is 2D-DCT representation of it. Combining such a representation with bases generated in preceding procedures, the image can be reconstruct in spatial domain. This reconstruction is a weighted summation of bases in DCT domain. The meaning of numbers inside 2D-DCT representation matrix is actually the coefficient of corresponding bases. The 2D-DCT transformation matrices for three given 4×4 images are shown in following equation (6).

$$dct_1 = \begin{bmatrix} 1 & 1.3065 & 1.00 & 0.5411 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} dct_2 = \begin{bmatrix} 1.00 & 0 & 0 & 0 \\ 0.5412 & 0 & 0 & 0 \\ -1.00 & 0 & 0 & 0 \\ -1.3066 & 0 & 0 & 0 \end{bmatrix} dct_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

After implementation weighted summation of bases according to these DCT representations,

images are reconstructed in spatial domain as illustrated in figure 9. The reconstructions are identical to their corresponding original images.

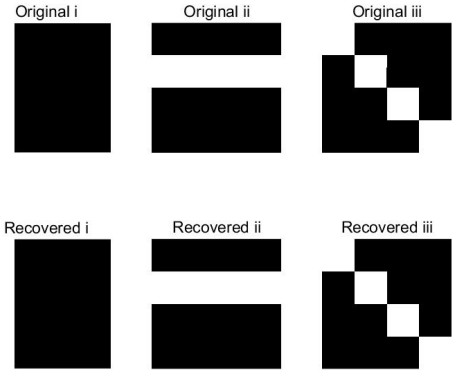


Figure 9: Comparison of original image and ones recovered from DCT

7 Discussion

7.1 Use of `fftshift`

In MATLAB toolbox, the result of `fft` is not ordered in frequency. The elements inside correspond to $[0, f_s/2]$ and $[-f_s/2, 0]$ [1]. `fftshift` can shift zero-frequency to the center of result sequence. In task one, students are requested to remove a portion of high frequency components. If such operations are applied directly to sequence from `fft`, the results are undesired. The proper procedures are applying `fftshift` and symmetrically remove the components at both ends of shifted sequence. `fftshift` is a quite useful function that can makes the sequences or image meaningful and understandable when visualize them.

7.2 Understanding DCT and bases

Recall the definition of DCT, it is necessary to obtain sampled cosine value for each components. Assume the length of input signal or width of square image is N. The length of each samples is N and there are N sets of such cosine samples.

$$F(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) cos[\frac{(2n+1)k\pi}{2N}] \tag{7}$$

As previously mentioned, those sets of sampled cosine values can be obtained directly by using formula above or, in an alternative way, calling `dctmxt(N)`. With a example where $N = 8$, following table is the contents of the matrix returned by this function

	1	2	3	4	5	6	7	8
1	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
2	0.4904	0.4157	0.2778	0.0975	-0.0975	-0.2778	-0.4157	-0.4904
3	0.4619	0.1913	-0.1913	-0.4619	-0.4619	-0.1913	0.1913	0.4619
4	0.4157	-0.0975	-0.4904	-0.2778	0.2778	0.4904	0.0975	-0.4157
5	0.3536	-0.3536	-0.3536	0.3536	0.3536	-0.3536	-0.3536	0.3536
6	0.2778	-0.4904	0.0975	0.4157	-0.4157	-0.0975	0.4904	-0.2778
7	0.1913	-0.4619	0.4619	-0.1913	-0.1913	0.4619	-0.4619	0.1913
8	0.0975	-0.2778	0.4157	-0.4904	0.4904	-0.4157	0.2778	-0.0975

Figure 10: Contents of matrix returned by `dctmtx`

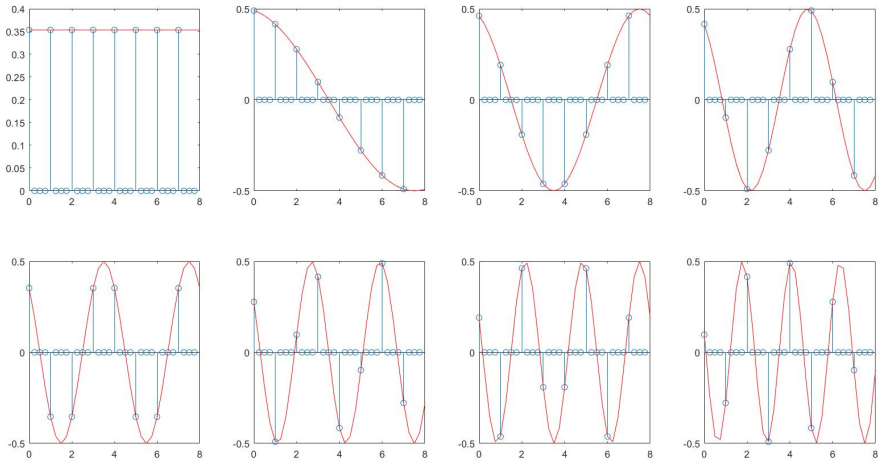


Figure 11: Each row of matrix represents a sampled cosine wave

Each column corresponds to a n value from 0 to N. Each row represents a sampled cosine wave with frequency of $\frac{k}{2N}$ and it is one of N bases for an 1D signal. Figure 11 visualizes such a relationship. As continuous cosine wave needs more subtle steps to generate, note that the zero points of discrete values are meaningless insertions for length matching in order to plot the figure and only valued sticks are from matrix in figure 10. Denote this matrix as `T`, we can obtain the DCT representation of an 1D signal $\mathbf{x}$ (assuming a row vector)

$$\mathbf{F} = \mathbf{T}\mathbf{x}' \tag{8}$$

The numbers in $\mathbf{F}$ are coefficient of each 8 bases (one row in matrix $\mathbf{T}$). Generalize DCT into 2D cases of images, we have N bases for both vertical and horizontal direction. Thus, there will be $N \times N$ bases for a $N \times N$ image.

7.3 Resize in frequency domain

This is just a trial driven by curiosity. Interpolation was applied in frequency domain which means the DFT representation of images were resized into (512, 512). Apply IDFT, the reconstruction of spatial images are shown in figure 12. The original portrait are duplicated. Compared with zero-padding which leads a higher resolution, interpolation of frequency representation results in duplications.

(a) `im420` (b) `im440`

Figure 12: Resize in frequency domain

8 Conclusion

This lab has enhance the knowledge of image transformation especially the properties and applications of 2D-DFT and 2D-DCT in image processing. Reconstruction of images from DCT representation and image filtering in frequency space has been implemented. During implementation, some details such as use of `fftshift` was investigated.

9 Appendix

Task 1: Generation of DCT

```
%%
% This script is used for EEE330 Lab_3 Task_4
% Author: Ruihao Wang
% ID: 1405884
% Function to generate 4*4 images of 2D-DCT
% Reference: Anonymous function
% https://ww2.mathworks.cn/help/matlab/matlab_prog/anonymous-
   functions.html

%%
function [bases, bases_norm] = generate_DCT(N)
bases = zeros(N, N, N, N);
bases_norm = zeros(N, N, N, N);
T = dctmtx(N);
T = T';
for i=1:N
    for j=1:N
        bases (:,:,i,j) = T(:,i)*T(:,j)';
    end
end

% Normalize the bases for visualization
for i=1:N
    for j=1:N
        tmp_bs = bases(:,:,i,j);
        min_b = min(tmp_bs(:));
        max_b = max(tmp_bs(:));
        range_b = max_b - min_b;
        if range_b == 0
            min_b = 0;
            range_b = max_b;
        end
        tmp_bs = (tmp_bs - min_b) / range_b;
        bases_norm(:,:,i,j) = tmp_bs ;
    end
end
```

Task 2: Zero-padding implementation

```
%%
```

```
% This script is used for EEE330 Lab_3 Task_4
% Author: Ruihao Wang
% ID: 1405884
% Implementation of 2D Zero-padding

%%
function padded = my_padding(img, w, h)
[m, n, c] = size(img);
% Get padding sizes
up_w = floor((w-m)/2);
up_h = floor((h-n)/2);
low_w = ceil((w-m)/2);
low_h = ceil((h-n)/2);
padded = zeros(w, h, c);
for i=1:c
    padded(up_w+1:w-low_w, up_h+1:h-low_h,c) = img(:,:,c);
end
```

Task 3: Project images to DCT bases

```
%%
% This script is used for EEE330 Lab_3 Task_4
% Author: Ruihao Wang
% ID: 1405884
% Function to do 2D-DCT and return the bases

%%
function [im_DCT, DCT_bases] = project2DCT(im)
T = dctmtx(size(im, 1));
im_DCT = T * im * T';
% ONLY return the linear index of bases (column first)
[rows, cols] = find(abs(im_DCT)>1e-15);
DCT_bases = [rows cols];
```

Task 4: Reconstruct images from DCT transform

```
%%
% This script is used for EEE330 Lab_3 Task_4
% Author: Ruihao Wang
% ID: 1405884
% Function to apply inverse DCT to recover images

%%
function rec_im = recover_DCT(dct_im, index, bases)
```

```
num_bases = size(index, 1);
N = size(dct_im, 1);
rec_im = zeros(N);
for i=1:num_bases
    coe = dct_im(index(i,1), index(i,2));
    base_mat = bases(:, :, index(i,1), index(i,2));
    rec_im = rec_im + coe * base_mat;
end
```

References

[1] MathWorks, "fft," [Online] https://www.mathworks.com/help/matlab/ref/fftshift.html.