HOMEWORK 3

M10907324吳俊逸

1. Implementation of 2D-DCT and its inverse transform

Required Submission: Code, Image Output in Spatial and Frequency Domain

Original image：



Frequency Domain

Code：

clc;clear all;close all;

input=imread('lena.bmp');

J=dct2(input);

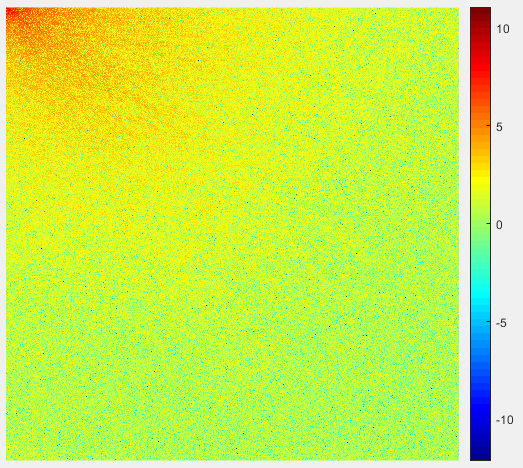
figure

imshow(log(abs(J)),[])

colormap(gca,jet(64))

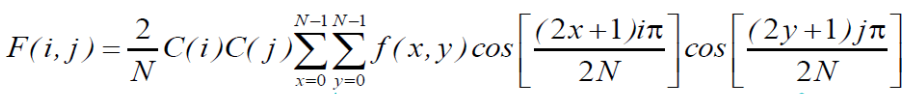
colorbar

Result：

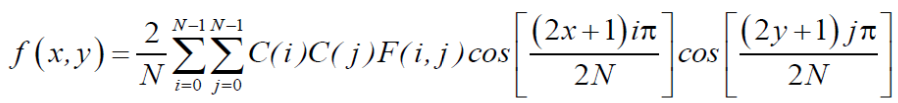


Spatial Domain

DCT Use formula：



IDCT Use formula：



Code：

clc;clear all;close all;

n=2;

% a=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16];

a=imread('lena.bmp');

[A,B] = size(a);

%DCT

for i=1:2:A

for j=1:2:B

%00

Ans\_DCT(i,j)=2/n\*(2^-0.5)\*(2^-0.5)\*[a(i,j)\*cos(0)\*cos(0)+a(i,j+1)\*cos(0)\*cos(0)+a(i+1,j)\*cos(0)\*cos(0)+a(i+1,j+1)\*cos(0)\*cos(0)];

%01

Ans\_DCT(i,j+1)=2/n\*(2^-0.5)\*1\*[a(i,j)\*cos(0)\*cos(0.25\*pi)+a(i,j+1)\*cos(0)\*cos(0.75\*pi)+a(i+1,j)\*cos(0)\*cos(0.25\*pi)+a(i+1,j+1)\*cos(0)\*cos(0.75\*pi)];

%10

Ans\_DCT(i+1,j)=2/n\*1\*(2^-0.5)\*[a(i,j)\*cos(0.25\*pi)\*cos(0)+a(i,j+1)\*cos(0.25\*pi)\*cos(0)+a(i+1,j)\*cos(0.75\*pi)\*cos(0)+a(i+1,j+1)\*cos(0.75\*pi)\*cos(0)];

%11

Ans\_DCT(i+1,j+1)=2/n\*1\*1\*[a(i,j)\*cos(0.25\*pi)\*cos(0.25\*pi)+a(i,j+1)\*cos(0.25\*pi)\*cos(0.75\*pi)+a(i+1,j)\*cos(0.75\*pi)\*cos(0.25\*pi)+a(i+1,j+1)\*cos(0.75\*pi)\*cos(0.75\*pi)];

end

end

%IDCT

for i=1:2:A

for j=1:2:B

%00

Ans\_IDCT(i,j)=2/n\*[(2^-0.5)\*(2^-0.5)\*Ans\_DCT(i,j)\*cos(0)\*cos(0)+(2^-0.5)\*1\*Ans\_DCT(i,j+1)\*cos(0)\*cos(0.25\*pi)+1\*(2^-0.5)\*Ans\_DCT(i+1,j)\*cos(0.25\*pi)\*cos(0)+1\*1\*Ans\_DCT(i+1,j+1)\*cos(0.25\*pi)\*cos(0.25\*pi)];

%01

Ans\_IDCT(i,j+1)=2/n\*[(2^-0.5)\*(2^-0.5)\*Ans\_DCT(i,j)\*cos(0)\*cos(0)+(2^-0.5)\*1\*Ans\_DCT(i,j+1)\*cos(0)\*cos(0.75\*pi)+1\*(2^-0.5)\*Ans\_DCT(i+1,j)\*cos(0.25\*pi)\*cos(0)+1\*1\*Ans\_DCT(i+1,j+1)\*cos(0.25\*pi)\*cos(0.75\*pi)];

%10

Ans\_IDCT(i+1,j)=2/n\*[(2^-0.5)\*(2^-0.5)\*Ans\_DCT(i,j)\*cos(0)\*cos(0)+(2^-0.5)\*1\*Ans\_DCT(i,j+1)\*cos(0)\*cos(0.25\*pi)+1\*(2^-0.5)\*Ans\_DCT(i+1,j)\*cos(0.75\*pi)\*cos(0)+1\*1\*Ans\_DCT(i+1,j+1)\*cos(0.75\*pi)\*cos(0.25\*pi)];

%11

Ans\_IDCT(i+1,j+1)=2/n\*[(2^-0.5)\*(2^-0.5)\*Ans\_DCT(i,j)\*cos(0)\*cos(0)+(2^-0.5)\*1\*Ans\_DCT(i,j+1)\*cos(0)\*cos(0.75\*pi)+1\*(2^-0.5)\*Ans\_DCT(i+1,j)\*cos(0.75\*pi)\*cos(0)+1\*1\*Ans\_DCT(i+1,j+1)\*cos(0.75\*pi)\*cos(0.75\*pi)];

end

end

% Ans\_DCT

imshow(Ans\_IDCT)

imwrite(Ans\_DCT,'lena\_DCT.png')

imwrite(Ans\_IDCT,'lena\_IDCT.png')

Result：

lena\_DCT.png



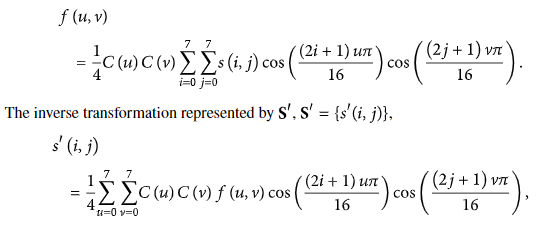
lena\_IDCT.png



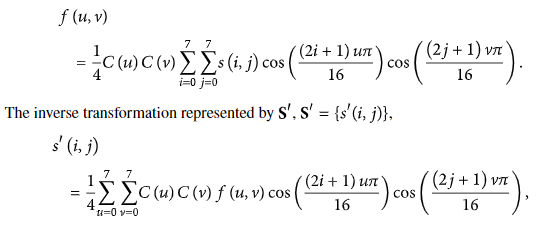
Additional Bonus: Fast DCT Algorithm

Spatial Domain

FDCT Use formula：



FIDCT Use formula：



Code：

clc;clear all;close all;

in\_image=imread('lena.bmp');

% The array of variables needed for "u", "v", "i", "j" in the FDCT formula

cosines\_DCT = [1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000

0.9808 0.8315 0.5556 0.1951 -0.1951 -0.5556 -0.8315 -0.9808

0.9239 0.3827 -0.3827 -0.9239 -0.9239 -0.3827 0.3827 0.9239

0.8315 -0.1951 -0.9808 -0.5556 0.5556 0.9808 0.1951 -0.8315

0.7071 -0.7071 -0.7071 0.7071 0.7071 -0.7071 -0.7071 0.7071

0.5556 -0.9808 0.1951 0.8315 -0.8315 -0.1951 0.9808 -0.5556

0.3827 -0.9239 0.9239 -0.3827 -0.3827 0.9239 -0.9239 0.3827

0.1951 -0.5556 0.8315 -0.9808 0.9808 -0.8315 0.5556 -0.1951];

alpha\_DCT = [0.1250 0.1768 0.1768 0.1768 0.1768 0.1768 0.1768 0.1768

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500];

% The array of variables needed for "u", "v", "i", "j" in the FIDCT formula

cosines\_IDCT = [1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000

0.9808 0.8315 0.5556 0.1951 -0.1951 -0.5556 -0.8315 -0.9808

0.9239 0.3827 -0.3827 -0.9239 -0.9239 -0.3827 0.3827 0.9239

0.8315 -0.1951 -0.9808 -0.5556 0.5556 0.9808 0.1951 -0.8315

0.7071 -0.7071 -0.7071 0.7071 0.7071 -0.7071 -0.7071 0.7071

0.5556 -0.9808 0.1951 0.8315 -0.8315 -0.1951 0.9808 -0.5556

0.3827 -0.9239 0.9239 -0.3827 -0.3827 0.9239 -0.9239 0.3827

0.1951 -0.5556 0.8315 -0.9808 0.9808 -0.8315 0.5556 -0.1951];

alpha\_IDCT = [0.1250 0.1768 0.1768 0.1768 0.1768 0.1768 0.1768 0.1768

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500

0.1768 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500 0.2500];

O\_DCT = double(zeros(8,8));

O\_IDCT = double(zeros(8,8));

[A,B] = size(in\_image);

for i=1:8:A

for j=1:8:B

% Create an 8\*8 array with the input image divided into 8\*8 values

I=[in\_image(i,j) in\_image(i,j+1) in\_image(i,j+2) in\_image(i,j+3) in\_image(i,j+4) in\_image(i,5) in\_image(i,6) in\_image(i,7);

in\_image(i+1,j) in\_image(i+1,j+1) in\_image(i+1,j+2) in\_image(i+1,j+3) in\_image(i+1,j+4) in\_image(i+1,5) in\_image(i+1,6) in\_image(i+1,7);

in\_image(i+2,j) in\_image(i+2,j+1) in\_image(i+2,j+2) in\_image(i+2,j+3) in\_image(i+2,j+4) in\_image(i+2,5) in\_image(i+2,6) in\_image(i+2,7);

in\_image(i+3,j) in\_image(i+3,j+1) in\_image(i+3,j+2) in\_image(i+3,j+3) in\_image(i+3,j+4) in\_image(i+3,5) in\_image(i+3,6) in\_image(i+3,7);

in\_image(i+4,j) in\_image(i+4,j+1) in\_image(i+4,j+2) in\_image(i+4,j+3) in\_image(i+4,j+4) in\_image(i+4,5) in\_image(i+4,6) in\_image(i+1,7);

in\_image(i+5,j) in\_image(i+5,j+1) in\_image(i+5,j+2) in\_image(i+5,j+3) in\_image(i+5,j+4) in\_image(i+5,5) in\_image(i+5,6) in\_image(i+1,7);

in\_image(i+6,j) in\_image(i+6,j+1) in\_image(i+6,j+2) in\_image(i+6,j+3) in\_image(i+6,j+4) in\_image(i+6,5) in\_image(i+6,6) in\_image(i+6,7);

in\_image(i+7,j) in\_image(i+7,j+1) in\_image(i+7,j+2) in\_image(i+7,j+3) in\_image(i+7,j+4) in\_image(i+7,5) in\_image(i+7,6) in\_image(i+7,7)];

% Start calculating FDCT, input is I array, output is O\_DCT array

for p = 1 : 8

for q = 1 : 8

s\_DCT = double(0);

for m = 1 : 8

for n = 1 : 8

% Make the sigma part behind the formula

s\_DCT = s\_DCT + (double(I(m,n)) \* cosines\_DCT(p,m) \* cosines\_DCT(q,n));

end

end

% Do the multiplication of C(u)C(v) in front of the formula

O\_DCT(p,q) = alpha\_DCT(p,q) \* s\_DCT;

end

end

% Start calculating FIDCT, input is O\_DCT array, output is O\_IDCT array

for m = 1 : 8

for n = 1 : 8

s\_IDCT = double(0);

for p = 1 : 8

for q = 1 : 8

% Do the sigma and multiplication part of the formula

s\_IDCT = s\_IDCT + (alpha\_IDCT(p,q) \* double(O\_DCT(p,q)) \* cosines\_IDCT(p,m) \* cosines\_IDCT(q,n));

end

end

% Store results

O\_IDCT(m,n) = s\_IDCT;

end

end

% Combine the results of FIDCT conversion from multiple 8\*8 arrays --> a whole image

for aa=1:1:8

for bb=1:1:8

Ans\_FIDCT(i+aa-1,j+bb-1)=uint8(O\_IDCT(aa,bb));

end

end

end

end

imshow(Ans\_FIDCT)

imwrite(Ans\_FIDCT,'lena\_FIDCT.png')

Result：

lena\_FDCT.png



lena\_FIDCT.png

