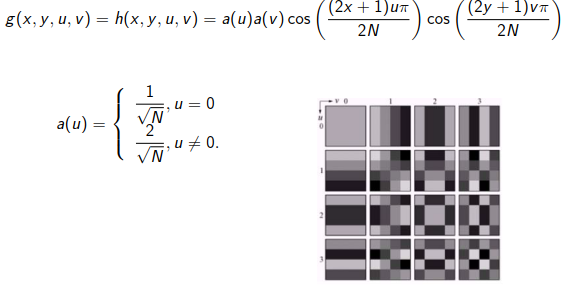
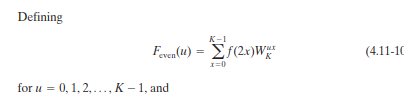
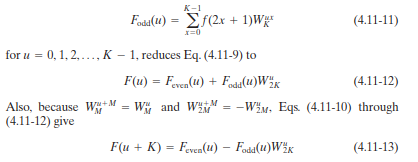
Prob1, DCT transform is of high additivity blocking artifacts in decompressed images. Fourier transform is of high quality decompressing image but is of poor additivity.

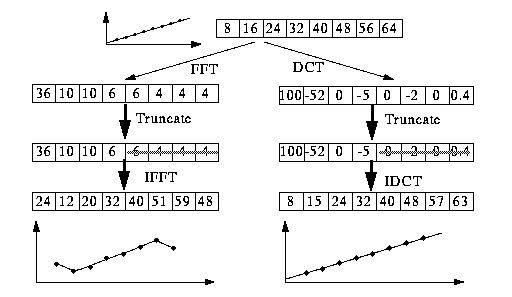
DCT:



FFT:



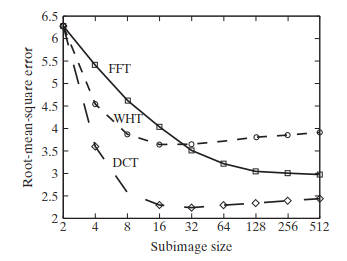


(\*\*)

Most transform coding systems are based on the DCT, which provides a good compromise between information packing ability and computational complexity. Compared to the other input independent transforms, it has the advantages of having been implemented in a single integrated circuit, packing the most information into the fewest coefficients‡ (for most images), and minimizing the block-like appearance, called blocking artifact, that results when the boundaries between subimages become visible. This last property is particularly important in comparisons with FFT .

FFT can approximate linear signals well few coefficients.

Another significant factor affecting transform coding error and computational complexity is subimage size.



One of the four coefficients (25%) of each transformed array was retained. The coefficient in all cases was the dc component, so the inverse transform simply replaced the four subimage pixels by their average value. Note that the blocking artifact that is prevalent in this result decreases as the subimage size increases to and in Figs. Figure below shows a zoomed portion of the original image for reference.

(b)

results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FFT | 298,00 | -39,41 + 80,67i | -28,00 + 30,0i | -36,59 + 12,67i |
|  | -26,00 | -36,59 - 12,67i | -28,00- 30,00i | -39,41 - 80,67i |
| DCT | 105,36 | -53,51 | -2,77 | -2,54 |
|  | 0,71 | -3,32 | -1,15 | -2,26 |



First raw is the original picture, the second is using the FFT transform while the third is using the DCT transform. According to the kernel on the picture(\*\*), we can see the greylevel after the FFT is decreasing while the one after the DCT jump up to zero again. This is also consistent with the experiment result.

(change the Matlab Code imshow into image can get the colored image.)

detail sees the matlab code

(c)

results:

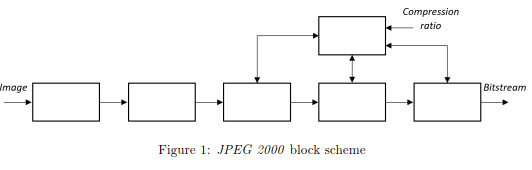
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IFFT | 8 | 16 | 24 | 32 | 46 | 50 | 58 | 64 |
| IDCT | 8 | 16 | 24 | 32 | 46 | 50 | 58 | 64 |



First raw is using the inverse FFT transform while the second is using the inverse DCT transform. According to the kernel on the picture (\*\*), we can see the greylevel after both the FFT and DCT are increasing.

detail sees the matlab code

Prob2



Rate Control

Tier II encoding(bitstream organization)

Tier I encoding(EBCOT)

Quantization

Discrete Wavelet

Transform(DWT)

Pre-processing

Prob3

(a)Usually, the JPEG use the 8\*8 quantization for both luminance and chrominance, Here shows the result of using different blocks 2\*2,4\*4,8\*8,16\*16, 32\*32



[Grab your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

[31.7197,30.8831,32.3807,34.0880,35.7927,37.4188]

[30.7197,29.8831,31.3807,32.0880,34.7927,36.4188]

[29.7197,28. 8831,30.3807,32.0880,33.7927,35.4188]

[30.4297,31.5831,33.7807,34.7205,34.7027,38.4188]

[31.7251,32.2222,33.4598,34.7205,35.4158,36.4287]

[27.7896,28.1589,29.5745,32.6971,34.2571,36.2457]

The larger the block sizes are the higher the peak of signal to noise ratio is.

(b) I only use the 8\*8 block and the zig-zag scan order because it’s coefficient used differently as 1,2,15,36,54,64 are unknown and I failed to know their quantization matrix.

result: PSNR=29.86.

*PSNR*=10log10( *peakval*2/*MSE)*

clear; close all; clc;

%%

x = [8, 16, 24, 32, 46, 50, 58, 64];

figure;

subplot(3,1,1)

imshow(x,[0,255])

%image(x)

subplot(3,1,2)

I1 = fft2(x);

imshow(I1)

subplot(3,1,3)

I2 = dct(x);

imshow(I2)

%%

figure

subplot(2,1,1)

I1\_1 = ifft(I1);

imshow(I1\_1,[0,255])

subplot(2,1,2)

I1\_2 = idct(I2);

imshow(I1\_2,[0,255])

%% Load YUV (raw) image

clear; close all; clc;

addpath('Images/');

imageName = 'lenna.yuv';

imageInfoName = 'lenna.inf';

[Y, U, V] = readImage(imageName, imageInfoName);

% Encoder

% Double precision plus center values around 0 (shifted block)

YCh = double(Y) - 128\*2;

UCh = double(U) - 128;

VCh = double(V) - 128;

%--------------------------------------------------------------------------

% Forward Discret Cosine Transform:

%

% - Discrete Cosine Transform (DCT) of NxN image blocks

%(T = H \* F \* H\_transposed), where H is the matrix containing the DCT

%coefficients (NxN matrix) and F is an NxN image block.

%--------------------------------------------------------------------------

for i = 1:6

H{i} = {zeros(2^i)};

end

%i=3;

% TASK: DCT coefficients (create NxN DCT matrix)

L1 = 0.35\*ones(1,8);

L3 = [0.5 0.42 0.27 0.1 -0.1 -0.28 -0.42 -0.5];

L2 = [0.46 0.2 -0.2 -0.46 -0.46 -0.2 0.2 0.46];

L4 = [0.42 0.1 -0.5 -0.28 0.28 0.5 0.10 -0.42];

L5 = [0.35 -0.35 -0.35 0.35 0.35 -0.35 -0.35 0.35];

L6 = [0.28 -0.5 0.1 0.42 -0.42 -0.1 0.5 -0.3];

L7 = [0.2 -0.46 0.46 -0.19 -0.19 0.46 -0.46 0.2];

L8 = [0.1 -0.3 0.42 -0.5 0.5 -0.42 0.28 -0.1];

H{3} = {[L1; L2; L3; L4; L5; L6; L7; L8]};

[Y U V] = ColorTransform(rgb);

for i=1:6

% TASK: Compute DCT for each block of the image

fun1 = @(block\_struct) imresize(block\_struct.data,8/length(YCh));

%fun2 = @(block\_struct) imresize(block\_struct.data,8/length(UCh));

YCh = blockproc(YCh,[2^i 2^i],fun1);

%UCh = blockproc(UCh,[8 8],fun2);

%VCh = blockproc(VCh,[8 8],fun2);

YCh = cell2mat(H{i}).\*YCh.\*cell2mat(H{i})';

%UCh = [cell2mat(H{i})].\*UCh.\*[cell2mat(H{i})]';

%VCh = [cell2mat(H{i})].\*VCh.\*[cell2mat(H{i})]';

%--------------------------------------------------------------------------

% Quantization of the DCT coefficients:

%

% - Standard JPEG quantization tables that represent a quality of 50%:

%

% For luminance (Y):

%

Q\_table\_Y = [16 11 10 16 24 40 51 61;

12 12 14 19 26 58 60 55;

14 13 16 24 40 57 69 56;

14 17 22 29 51 87 80 62;

18 22 37 56 68 109 103 77;

24 35 55 64 81 104 113 92;

49 64 78 87 103 121 120 101;

72 92 95 98 112 100 103 99];

%

%

% For chrominance (U and V):

%

Q\_table\_UV = [17 18 24 47 99 99 99 99;

18 21 26 66 99 99 99 99;

24 26 56 99 99 99 99 99;

47 66 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99];

%

%

% - Quantization:

% DCT element

% q = round( ----------------- )

% Q\_table element

%

%--------------------------------------------------------------------------

% TASK: Apply quantization

qY = round(cell2mat(H{i})/Q\_table\_Y);

%qU = round([cell2mat(H{i})]/Q\_table\_UV);

%qV = round([cell2mat(H{i})]/Q\_table\_UV);

% Decoder

%--------------------------------------------------------------------------

% Inverse quantization of DCT Coefficients

%IQDCT = [Q\_table\_Y.\*qY;Q\_table\_UV.\*qU;Q\_table\_UV.\*qV];

IQDCT = Q\_table\_Y.\*qY;

%--------------------------------------------------------------------------

% TASK: Apply inverse quantization

bY = IQDCT.\*qY;

%bU = IQDCT.\*qU;

%bV = IQDCT.\*qV;

%--------------------------------------------------------------------------

% Inverse Discrete Cosine Transform:

L1\_1 = 0.35\*ones(8,1);

L2\_1 = [0.5 0.42 0.27 0.1 -0.1 -0.28 -0.42 -0.5]';

L3\_1 = [0.46 0.2 -0.2 -0.46 -0.46 -0.2 0.2 0.46]';

L4\_1 = [0.42 -0.1 -0.5 -0.28 0.28 0.5 0.10 -0.42]';

L5\_1 = [0.35 -0.35 -0.35 0.35 0.35 -0.35 -0.35 0.35]';

L6\_1 = [0.28 -0.5 0.1 0.42 -0.42 -0.1 0.5 -0.3]';

L7\_1 = [0.2 -0.46 0.46 -0.19 -0.19 0.46 -0.46 0.2]';

L8\_1 = [0.1 -0.3 0.42 -0.5 0.5 -0.42 0.28 -0.1]';

G{i} = {[L1\_1 L2\_1 L3\_1 L4\_1 L5\_1 L6\_1 L7\_1 L8\_1]'};

%--------------------------------------------------------------------------

% TASK: Apply inverse DCT

QY=cell2mat(G{i})\*bY\*cell2mat(G{i})';

%QU=[celltomatG{i}]\*bU\*[celltomat(G{i})]';

%QV=[celltomatG{i}]\*bV\*[celltomat(G{i})]';

%--------------------------------------------------------------------------

% Inverse Shifted Block

%--------------------------------------------------------------------------

% TASK: Apply inverse Shifted Block operation

%fun1 = @(block\_struct) imresize(block\_struct.data,length(YCh)/8);

%fun2 = @(block\_struct) imresize(block\_struct.data,8/length(UCh));

YCh=imresize(YCh,512/2^i);

%YCh\_1 = blockproc(YCh,[512 512],fun1);

%UCh\_1 = blockproc(UCh,[8 8],fun2);

%VCh\_1 = blockproc(VCh,[8 8],fun2);

YCh\_1(i) = uint8(YCh) + 128\*2;

%UCh1 = int8(QU) + 128;

%VCh1 = int8(QV) + 128;

%imshow(Ych\_1,[0,255])

%I = hsv2rgb([Ych1,Uch1,Vch])

[rgb] = InvColor(Y,U,V)

%----------------------------------------------------------

% Display results

%----------------------------------------------------------

% TASK: Compute PSNR

[peaksnr21(i), snr21(i)] = psnr(YCh\_1(i),Y)

end

plot(snr21)

%%

i=3;

a=zigzag(8,1,1)

fun1 = @(block\_struct) imresize(block\_struct.data,8/length(YCh));

%fun2 = @(block\_struct) imresize(block\_struct.data,8/length(UCh));

YCh = blockproc(YCh,[8 8],fun1);

%UCh = blockproc(UCh,[8 8],fun2);

%VCh = blockproc(VCh,[8 8],fun2);

YCh = cell2mat(H{i}).\*YCh.\*cell2mat(H{i})';

%UCh = [cell2mat(H{i})].\*UCh.\*[cell2mat(H{i})]';

%VCh = [cell2mat(H{i})].\*VCh.\*[cell2mat(H{i})]';

%--------------------------------------------------------------------------

% Quantization of the DCT coefficients:

%

% - Standard JPEG quantization tables that represent a quality of 50%:

%

% For luminance (Y):

%

Q\_table\_Y = [16 11 10 16 24 40 51 61;

12 12 14 19 26 58 60 55;

14 13 16 24 40 57 69 56;

14 17 22 29 51 87 80 62;

18 22 37 56 68 109 103 77;

24 35 55 64 81 104 113 92;

49 64 78 87 103 121 120 101;

72 92 95 98 112 100 103 99];

%

%

% For chrominance (U and V):

%

Q\_table\_UV = [17 18 24 47 99 99 99 99;

18 21 26 66 99 99 99 99;

24 26 56 99 99 99 99 99;

47 66 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99];

%

%

% - Quantization:

% DCT element

% q = round( ----------------- )

% Q\_table element

%

%--------------------------------------------------------------------------

% TASK: Apply quantization

qY = round(cell2mat(H{i})/Q\_table\_Y);

%qU = round([cell2mat(H{i})]/Q\_table\_UV);

%qV = round([cell2mat(H{i})]/Q\_table\_UV);

% Decoder

%--------------------------------------------------------------------------

% Inverse quantization of DCT Coefficients

%IQDCT = [Q\_table\_Y.\*qY;Q\_table\_UV.\*qU;Q\_table\_UV.\*qV];

IQDCT = Q\_table\_Y.\*qY;

%--------------------------------------------------------------------------

% TASK: Apply inverse quantization

bY = IQDCT.\*qY;

%bU = IQDCT.\*qU;

%bV = IQDCT.\*qV;

%--------------------------------------------------------------------------

% Inverse Discrete Cosine Transform:

L1\_1 = 0.35\*ones(8,1);

L2\_1 = [0.5 0.42 0.27 0.1 -0.1 -0.28 -0.42 -0.5]';

L3\_1 = [0.46 0.2 -0.2 -0.46 -0.46 -0.2 0.2 0.46]';

L4\_1 = [0.42 -0.1 -0.5 -0.28 0.28 0.5 0.10 -0.42]';

L5\_1 = [0.35 -0.35 -0.35 0.35 0.35 -0.35 -0.35 0.35]';

L6\_1 = [0.28 -0.5 0.1 0.42 -0.42 -0.1 0.5 -0.3]';

L7\_1 = [0.2 -0.46 0.46 -0.19 -0.19 0.46 -0.46 0.2]';

L8\_1 = [0.1 -0.3 0.42 -0.5 0.5 -0.42 0.28 -0.1]';

G{i} = {[L1\_1 L2\_1 L3\_1 L4\_1 L5\_1 L6\_1 L7\_1 L8\_1]'};

%--------------------------------------------------------------------------

% TASK: Apply inverse DCT

QY=cell2mat(G{i})\*bY\*cell2mat(G{i})';

%QU=[celltomatG{i}]\*bU\*[celltomat(G{i})]';

%QV=[celltomatG{i}]\*bV\*[celltomat(G{i})]';

%--------------------------------------------------------------------------

% Inverse Shifted Block

%--------------------------------------------------------------------------

% TASK: Apply inverse Shifted Block operation

%fun1 = @(block\_struct) imresize(block\_struct.data,length(YCh)/8);

%fun2 = @(block\_struct) imresize(block\_struct.data,8/length(UCh));

YCh=imresize(YCh,512/2^i);

%YCh\_1 = blockproc(YCh,[512 512],fun1);

%UCh\_1 = blockproc(UCh,[8 8],fun2);

%VCh\_1 = blockproc(VCh,[8 8],fun2);

YCh\_1(i) = uint8(YCh + 128);

%UCh1 = int8(QU) + 128;

%VCh1 = int8(QV) + 128;

%imshow(Ych\_1,[0,255])

%I = hsv2rgb([Ych1,Uch1,Vch])

%----------------------------------------------------------

% Display results

%----------------------------------------------------------

% TASK: Compute PSNR

[peaksnr21(i), snr21(i)] = psnr(YCh\_1(i),Y)

end