0516222 許芳瑀

Hw3

Introduction / Objectives:

透過實作 WHT、DFT、DCT 三種不同的方法，搭配不同的 quantization 以及 block size，觀察圖片的結果，最後用 erms、SNR 等方法計算誤差。

A review of the methods you have used (be concise)

- WHT

$$W(u,v)=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)Wal_H(u,x)Wsl_H(v,y) \qquad f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}W(u,v)Wal_H(u,x)Wsl_H(v,y)$$

x, u=0, 1, 2, …, $M-1$; y, v=0, 1, 2, …, $N-1$。

- DFT

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)e^{-j2\pi(ux/M+vy/N)} \quad u=0,1,2,\ldots M-1; v=0,\ldots,N-1$$

$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{j2\pi(ux/M+vy/N)} \quad x=0,1,2,\ldots M-1; y=0,\ldots,N-1$$

- DCT

$$B_{pq}=\alpha_p\alpha_q\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}A_{mn}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N}, \quad \begin{array}{l}0\le p\le M-1\\0\le q\le N-1\end{array}$$

$$\alpha_p=\begin{cases}1/\sqrt{M}, & p=0\\ \sqrt{2/M}, & 1\le p\le M-1\end{cases} \qquad \alpha_q=\begin{cases}1/\sqrt{N}, & q=0\\ \sqrt{2/N}, & 1\le q\le N-1\end{cases}$$

$$A_{mn}=\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\alpha_p\alpha_qB_{pq}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N}, \quad \begin{array}{l}0\le m\le M-1\\0\le n\le N-1\end{array}$$

$$\alpha_p=\begin{cases}1/\sqrt{M}, & p=0\\ \sqrt{2/M}, & 1\le p\le M-1\end{cases} \qquad \alpha_q=\begin{cases}1/\sqrt{N}, & q=0\\ \sqrt{2/N}, & 1\le q\le N-1\end{cases}$$

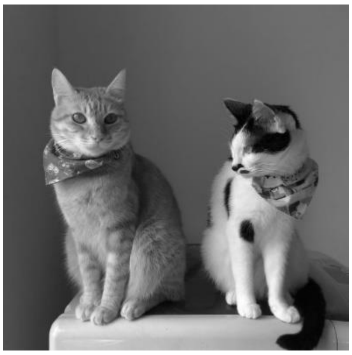
A explanation of the experiments you have done, and the results.

I use two picture to compare. They are 512*512 and 128*128.

- **WHT**

I. Different block size

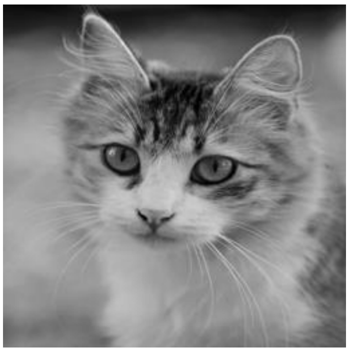
i. Block size = 8



Variance: 2.7248

eRMS: 2.3463e-14

SNR: 2.3991e+31

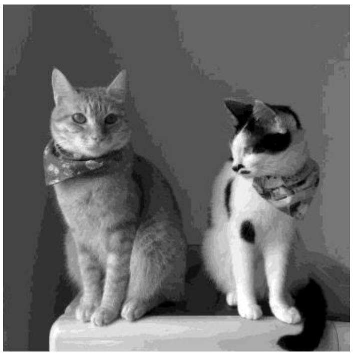


Variance: 4.2785

eRMS: 4.5743e-14

SNR: 8.6724e+30

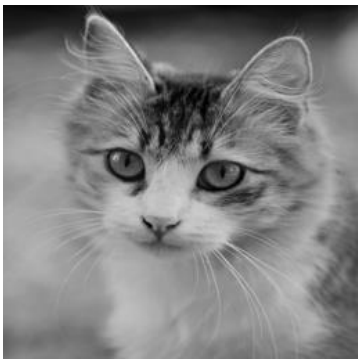
ii. Block size = 32



Variance: 3.3031

eRMS: 5.5639e-14

SNR: 4.2662e+30

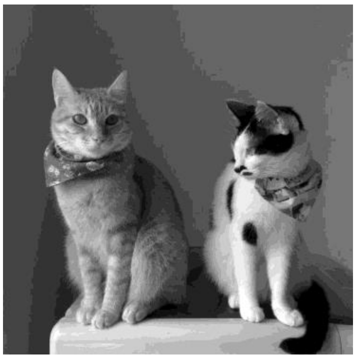


Variance: 4.1440

eRMS: 6.5500e-14

SNR: 4.2297e+30

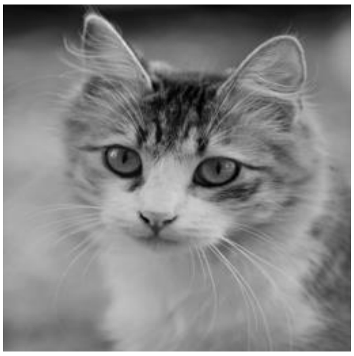
iii. Block size = 64



Variance: 3.1271

eRMS: 8.6990e-14

SNR: 1.7453e+30



Variance: 3.9172

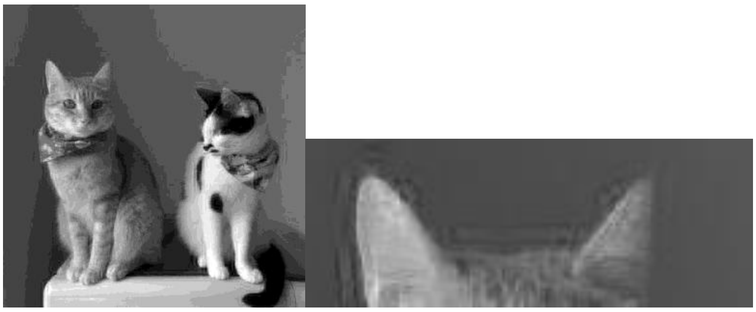
eRMS: 9.3346e-14

SNR: 2.0825e+30

II. Different quantization(block size = 16)

i. Keep only the first k coefficients.

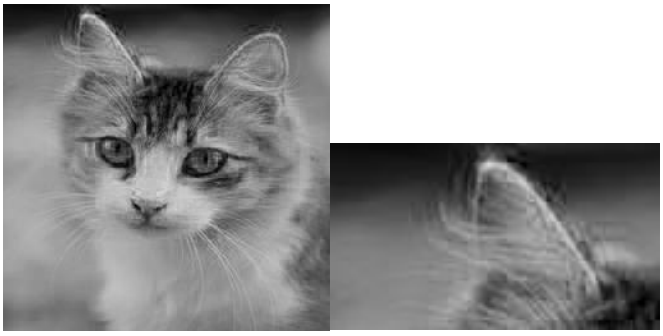
1. k = 169



Variance: 2.2716

eRMS: 4.4029

SNR: 680.2693

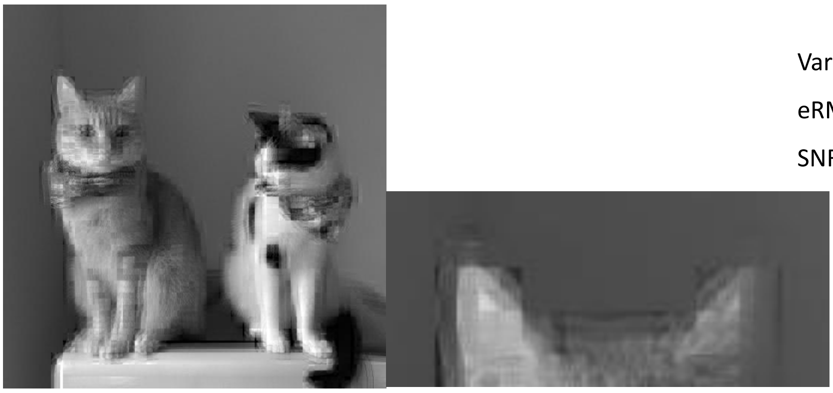


Variance: 3.0636

eRMS: 5.5559

SNR: 586.8644

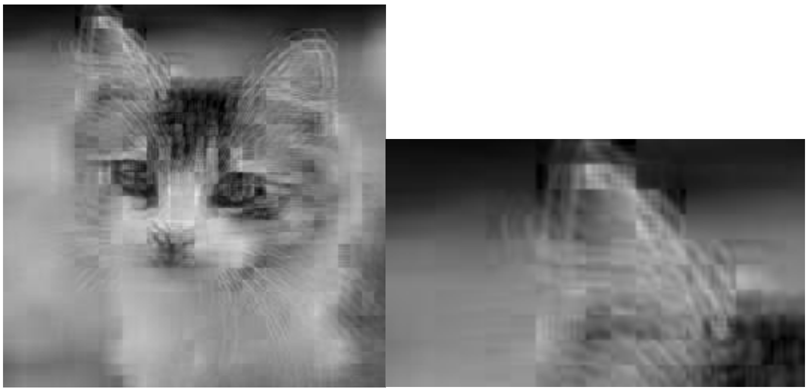
2. k = 100



Variance: 1.4794

eRMS: 9.5235

SNR: 144.6155



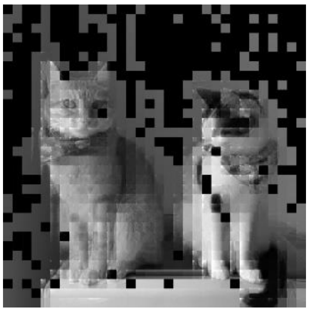
Variance: 1.9654

eRMS: 10.6141

SNR: 160.0714

ii. Keep only the coefficients with the k largest coefficients.

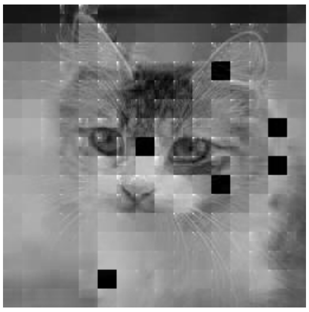
1. k = 100



Variance: 2.7631

eRMS: 54.8326

SNR: 3.3926

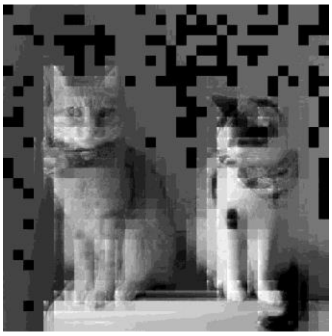


Variance: 3.6298

eRMS: 28.6029

SNR: 21.1803

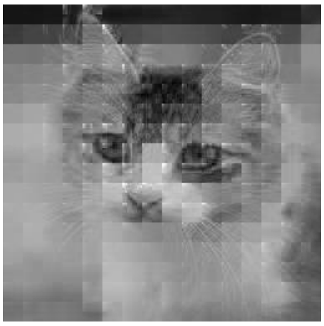
2. k = 25



Variance: 0.8982

eRMS: 35.6238

SNR: 9.4068



Variance: 0.9864

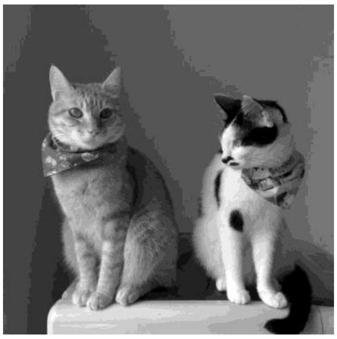
eRMS: 14.8240

SNR: 81.5771

- **DFT**

I. Different block size

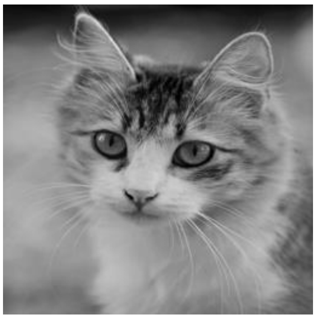
i. Block size = 8



Variance: 2.6449

eRMS: 2.0869e-13 + 1.3777e-14i

SNR: 2.9930e+29 – 3.9690e+28i

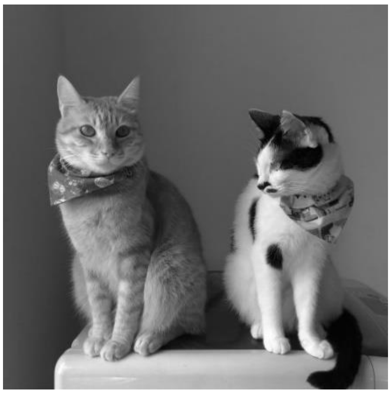


Variance: 3.8439

eRMS: 2.4127e-13 + 1.6307e-14i

SNR: 3.0749e+29 - 4.1756e+28i

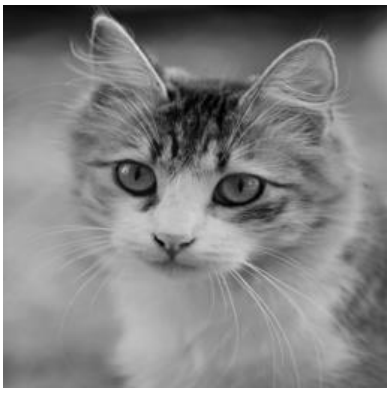
ii. Block size = 32



Variance: 4.0125

eRMS: 4.7541e-13 + 5.0969e-14i

SNR: 5.6457e+28 – 1.2246e+28i

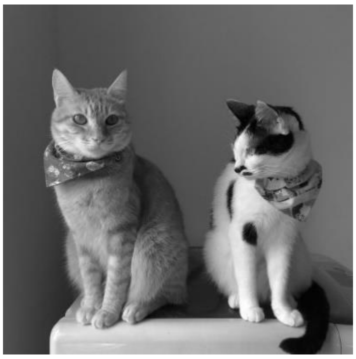


Variance: 4.3152

eRMS: 3.6519e-13 + 8.4655e-14i

SNR: 1.1596e+29 - 5.6813e+28i

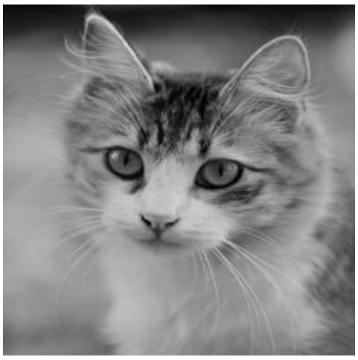
iii. Block size = 64



Variance: 4.1830

eRMS: 1.9382e-12 +
1.0264e-14i

SNR: 3.5153e+27 –
3.7233e+25i



Variance: 4.2418

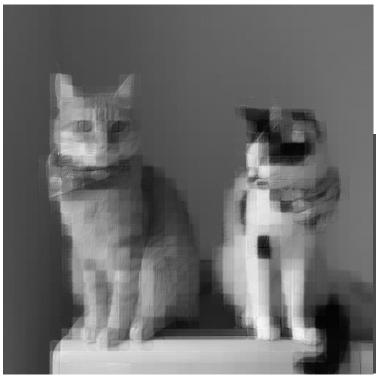
eRMS: 7.8735e-13 + 3.0501e-14i

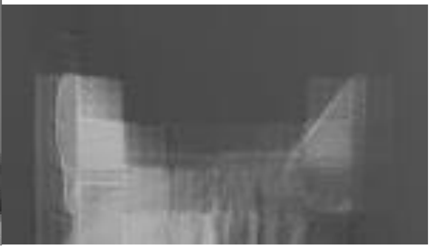
SNR: 2.9141e+28 - 2.2611e+27i

II. Different quantization(block size = 16)

i. Keep only the first k coefficients.

1. k = 169

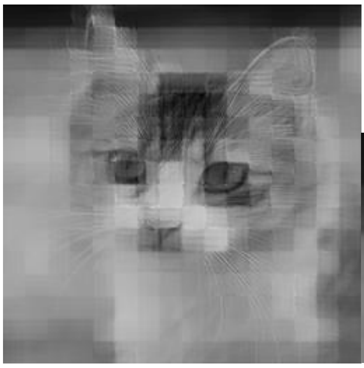


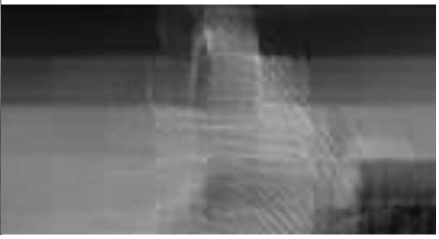


Variance: 2.4147

eRMS: 5.1481 - 0.0000i

SNR: 4.8496e+02 + 1.6940e-13i



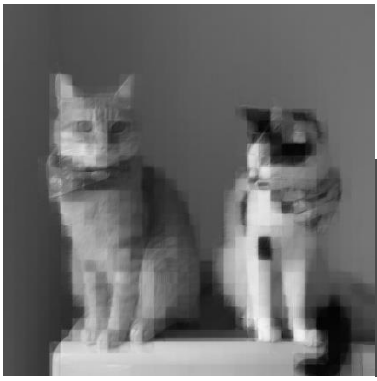


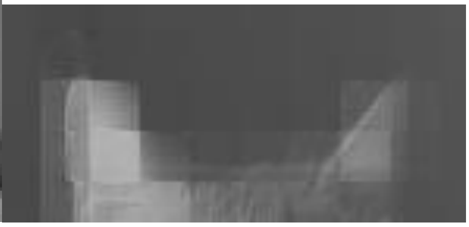
Variance: 2.9405

eRMS: 7.2623 - 0.0000i

SNR: 3.3728e+02 + 6.1669e-14i

2. k = 100

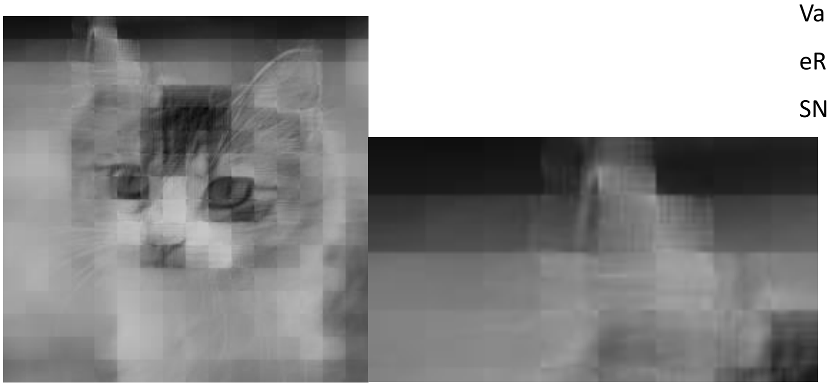




Variance: 1.6010

eRMS: 5.7467 + 0.0000i

SNR: 3.8840e+02 - 4.5393e-14i



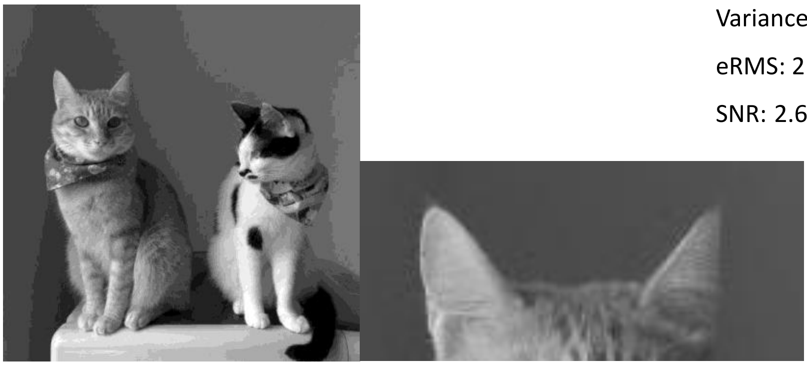
Variance: 1.9169

eRMS: 8.0680 + 0.0000i

SNR: 2.7292e+02 - 4.5734e-14i

ii. Keep only the coefficients with the k largest coefficients.

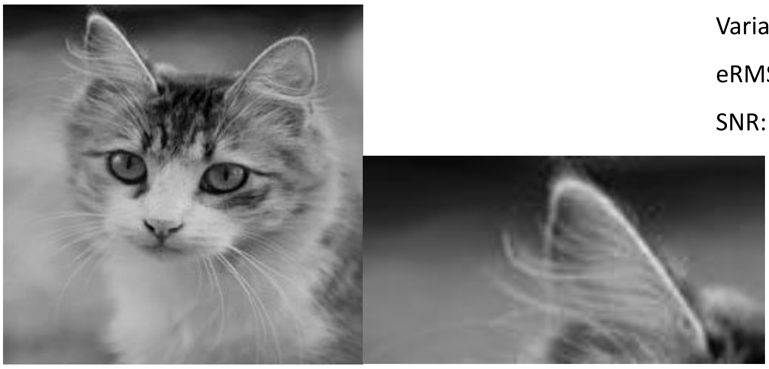
1. k = 100



Variance: 1.8922

eRMS: 2.2357 + 0.0000i

SNR: 2.6412e+03 - 6.7610e-13i

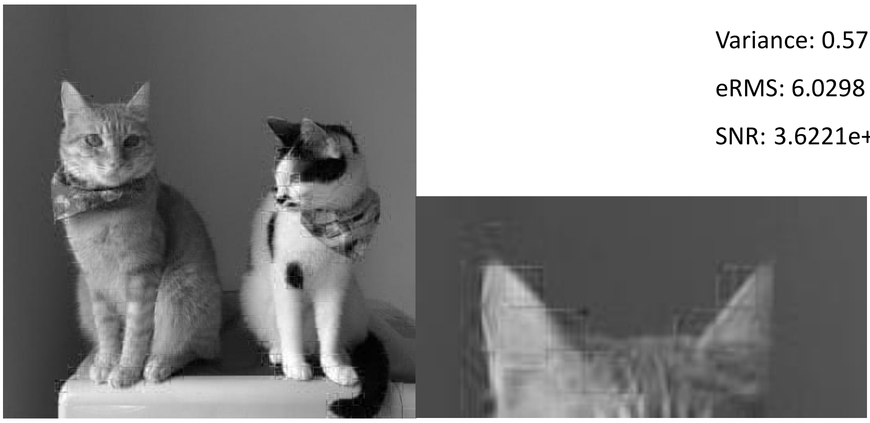


Variance: 2.0071

eRMS: 2.5308 + 0.0000i

SNR: 2.8321e+03 - 3.9082e-13i

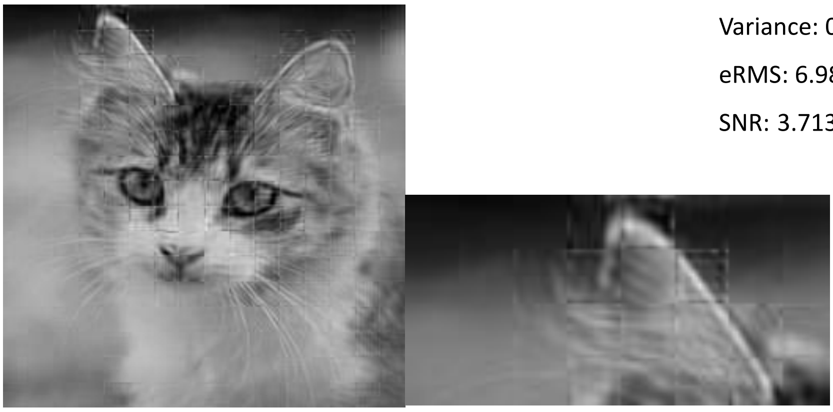
2. k = 25



Variance: 0.5711

eRMS: 6.0298 + 0.0000i

SNR: 3.6221e+02 - 7.7671e-14i



Variance: 0.5170

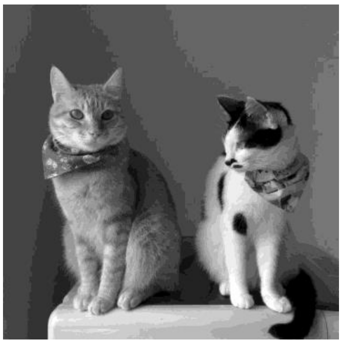
eRMS: 6.9810 + 0.0000i

SNR: 3.7133e+02 - 8.1685e-14i

- **DCT**

I. Different block size

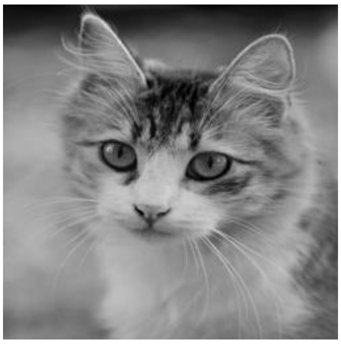
i. Block size = 8



Variance: 2.9441

eRMS: 2.4872e-13

SNR: 2.1350e+29

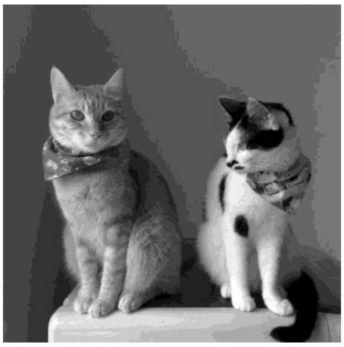


Variance: 3.3346

eRMS: 2.8721e-13

SNR: 2.1998e+29

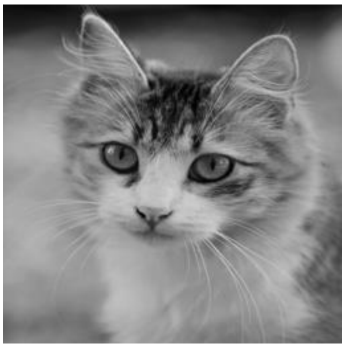
ii. Block size = 32



Variance: 3.7344

eRMS: 6.3638e-13

SNR: 3.2612e+28

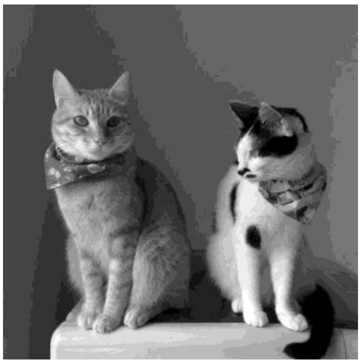


Variance: 3.2298

eRMS: 6.1525e-13

SNR: 4.7939e+28

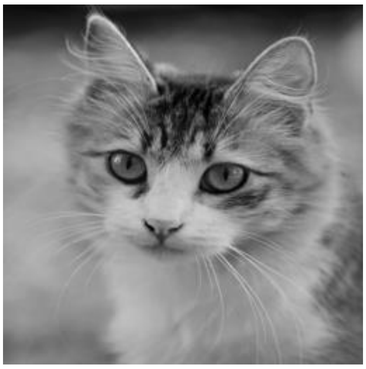
iii. Block size = 64



Variance: 3.4994

eRMS: 2.0509e-12

SNR: 3.1399e+27



Variance: 2.8351

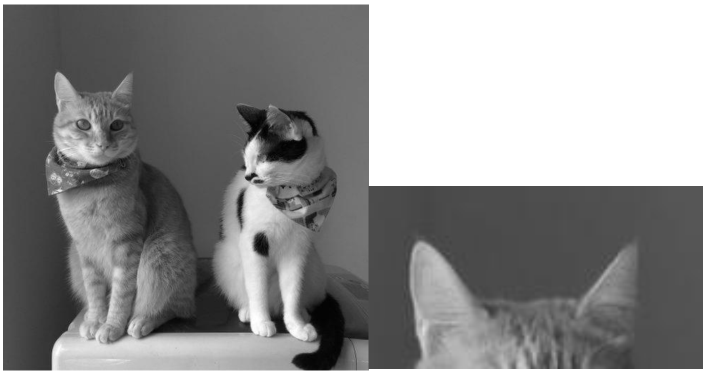
eRMS: 1.0900e-12

SNR: 1.5272e+28

II. Different quantization

i. Keep only the first k coefficients.

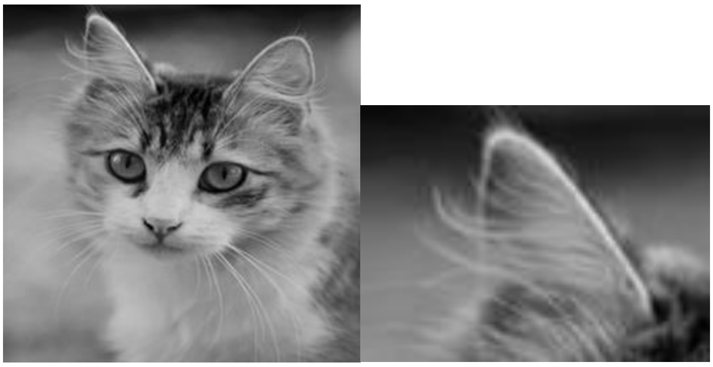
1. k = 169



Variance: 3.5672

eRMS: 1.8436

SNR: 3.8849e+03

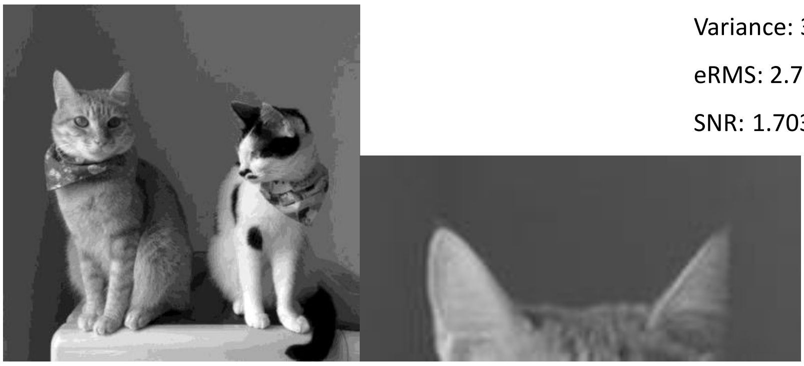


Variance: 3.3583

eRMS: 1.7266

SNR: 6.0859e+03

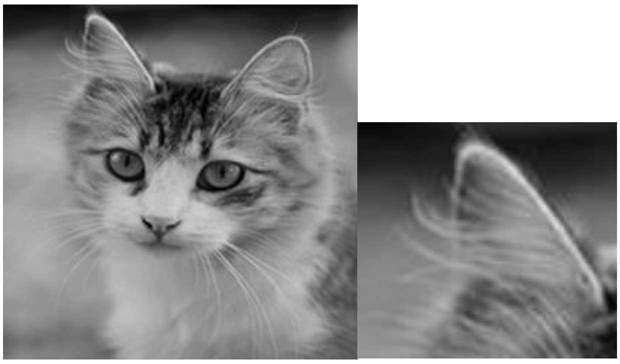
2. k = 100



Variance: 3.5672

eRMS: 2.7840

SNR: 1.7030e+03



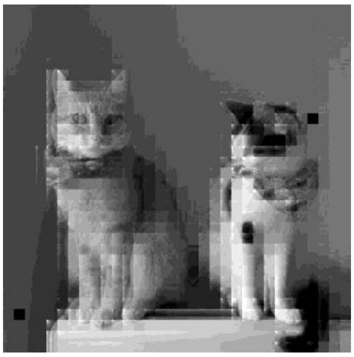
Variance: 3.3583

eRMS: 2.9856

SNR: 2.0348e+03

ii. Keep only the coefficients with the k largest coefficients.

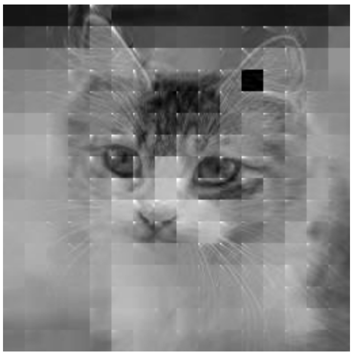
1. k = 100



Variance: 3.5672

eRMS: 15.3839

SNR: 54.8038

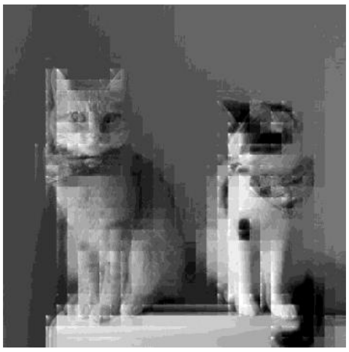


Variance: 3.3583

eRMS: 17.1702

SNR: 60.5511

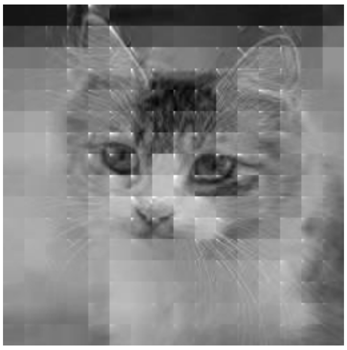
2. k = 25



Variance: 3.5672

eRMS: 14.9457

SNR: 58.1243



Variance: 3.3583

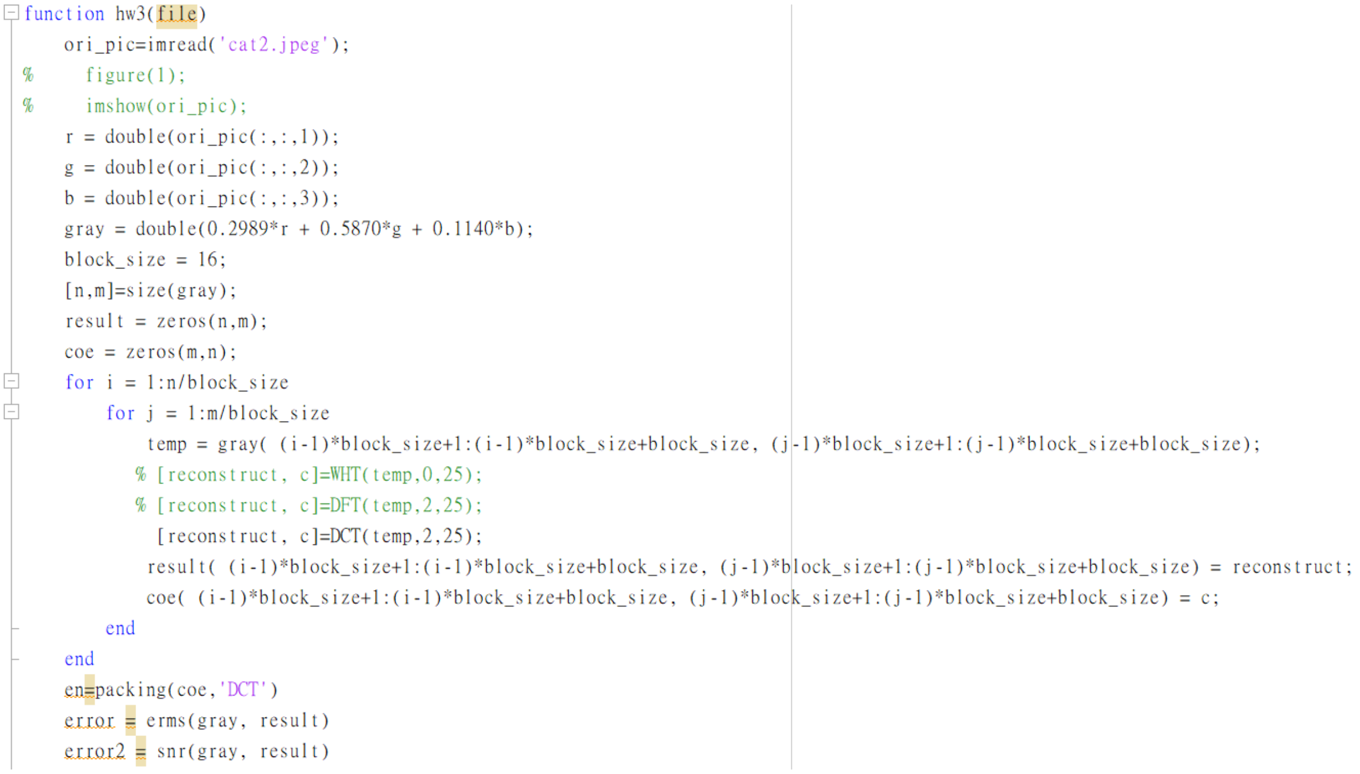
eRMS: 14.6007

SNR: 84.1221

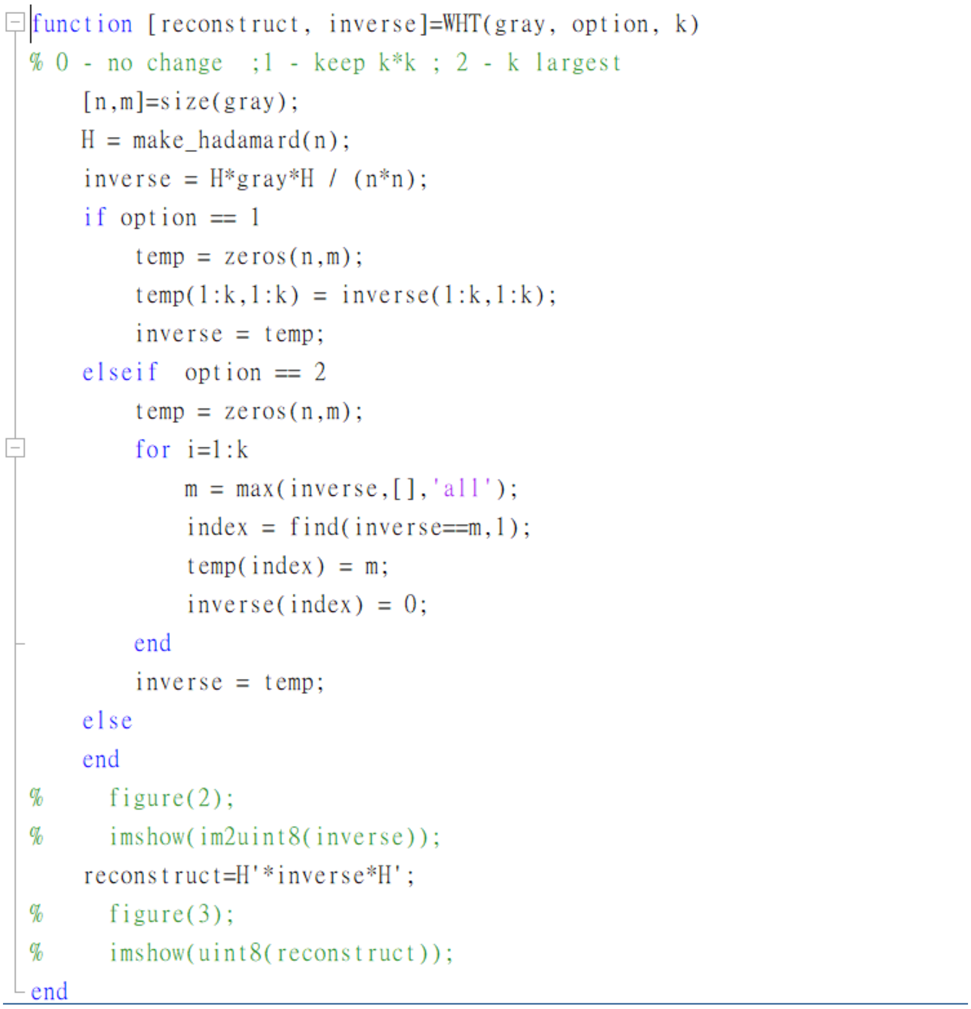
Discussions: Your observations, interpretations of results, and remaining questions.

大部分的圖片都跟預期的效果差不多，例如 quantize 更多的係數，圖片就越模糊、SNR 就越低，但像是 DCT 的 block size 改變的實驗中，反而是 size 8 的 eRMS 最小。而比較特別的是，如果用 WHT，再用 quantization 之後，圖片用人眼看會差很多。

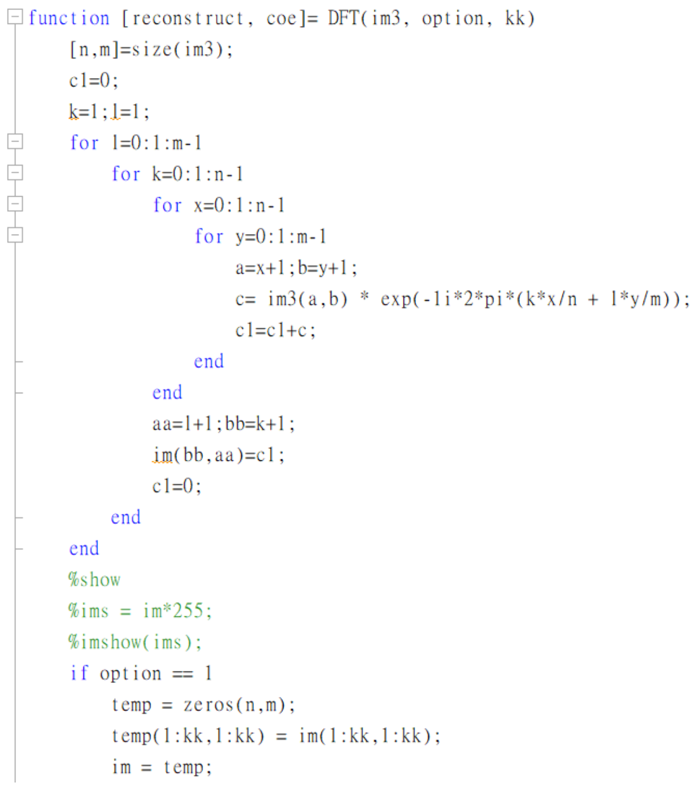
Code



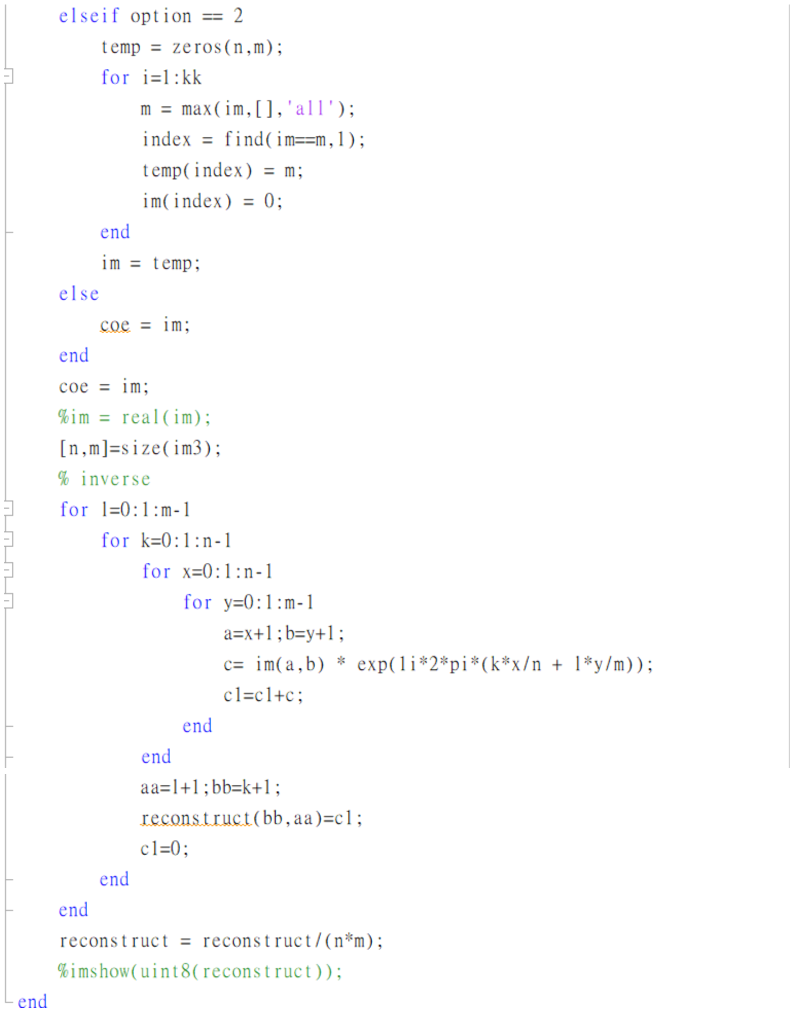
```
function hw3(file)
    ori_pic=imread('cat2.jpeg');
%     figure(1);
%     imshow(ori_pic);
    r = double(ori_pic(:,:,1));
    g = double(ori_pic(:,:,2));
    b = double(ori_pic(:,:,3));
    gray = double(0.2989*r + 0.5870*g + 0.1140*b);
    block_size = 16;
    [n,m]=size(gray);
    result = zeros(n,m);
    coe = zeros(m,n);
    for i = 1:n/block_size
        for j = 1:m/block_size
            temp = gray( (i-1)*block_size+1:(i-1)*block_size+block_size, (j-1)*block_size+1:(j-1)*block_size+block_size);
           % [reconstruct, c]=WHT(temp,0,25);
           % [reconstruct, c]=DFT(temp,2,25);
             [reconstruct, c]=DCT(temp,2,25);
            result( (i-1)*block_size+1:(i-1)*block_size+block_size, (j-1)*block_size+1:(j-1)*block_size+block_size) = reconstruct;
            coe( (i-1)*block_size+1:(i-1)*block_size+block_size, (j-1)*block_size+1:(j-1)*block_size+block_size) = c;
        end
    end
    en=packing(coe,'DCT')
    error = erms(gray, result)
    error2 = snr(gray, result)
```



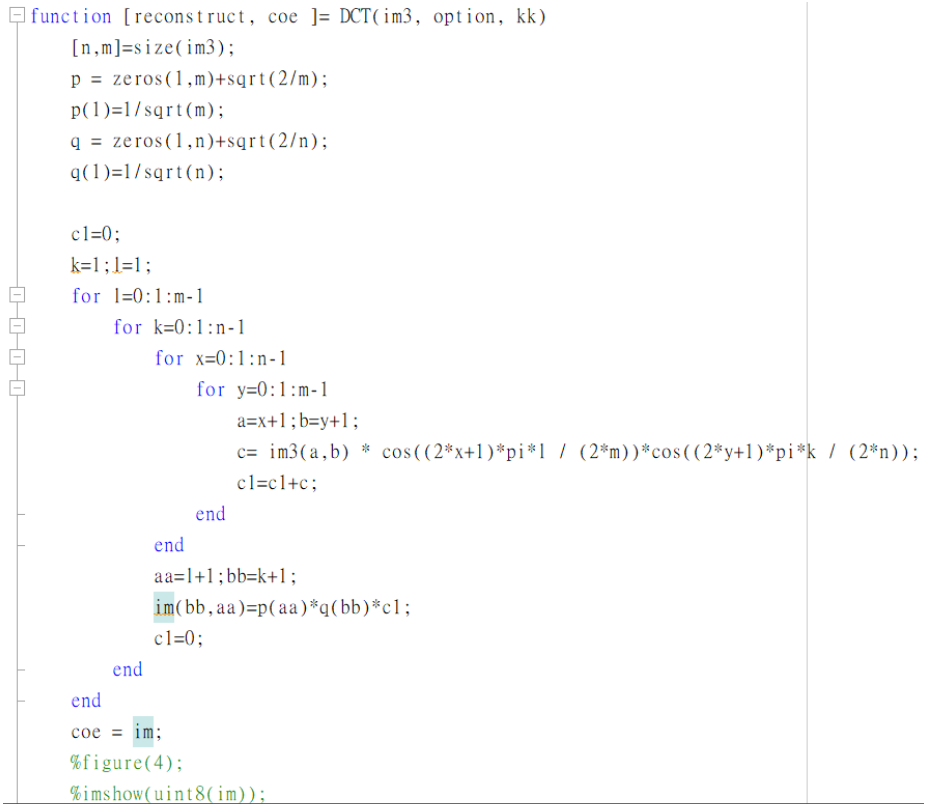
```
function [reconstruct, inverse]=WHT(gray, option, k)
% 0 - no change  ;1 - keep k*k ; 2 - k largest
    [n,m]=size(gray);
    H = make_hadamard(n);
    inverse = H*gray*H / (n*n);
    if option == 1
        temp = zeros(n,m);
        temp(1:k,1:k) = inverse(1:k,1:k);
        inverse = temp;
    elseif  option == 2
        temp = zeros(n,m);
        for i=1:k
            m = max(inverse,[],'all');
            index = find(inverse==m,1);
            temp(index) = m;
            inverse(index) = 0;
        end
        inverse = temp;
    else
    end
%     figure(2);
%     imshow(im2uint8(inverse));
    reconstruct=H'*inverse*H';
%     figure(3);
%     imshow(uint8(reconstruct));
end
```



```
function [reconstruct, coe]= DFT(im3, option, kk)
    [n,m]=size(im3);
    c1=0;
    k=1;l=1;
    for l=0:1:m-1
        for k=0:1:n-1
            for x=0:1:n-1
                for y=0:1:m-1
                    a=x+1;b=y+1;
                    c= im3(a,b) * exp(-1i*2*pi*(k*x/n + l*y/m));
                    c1=c1+c;
                end
            end
            aa=l+1;bb=k+1;
            im(bb,aa)=c1;
            c1=0;
        end
    end
    %show
    %ims = im*255;
    %imshow(ims);
    if option == 1
        temp = zeros(n,m);
        temp(1:kk,1:kk) = im(1:kk,1:kk);
        im = temp;
```

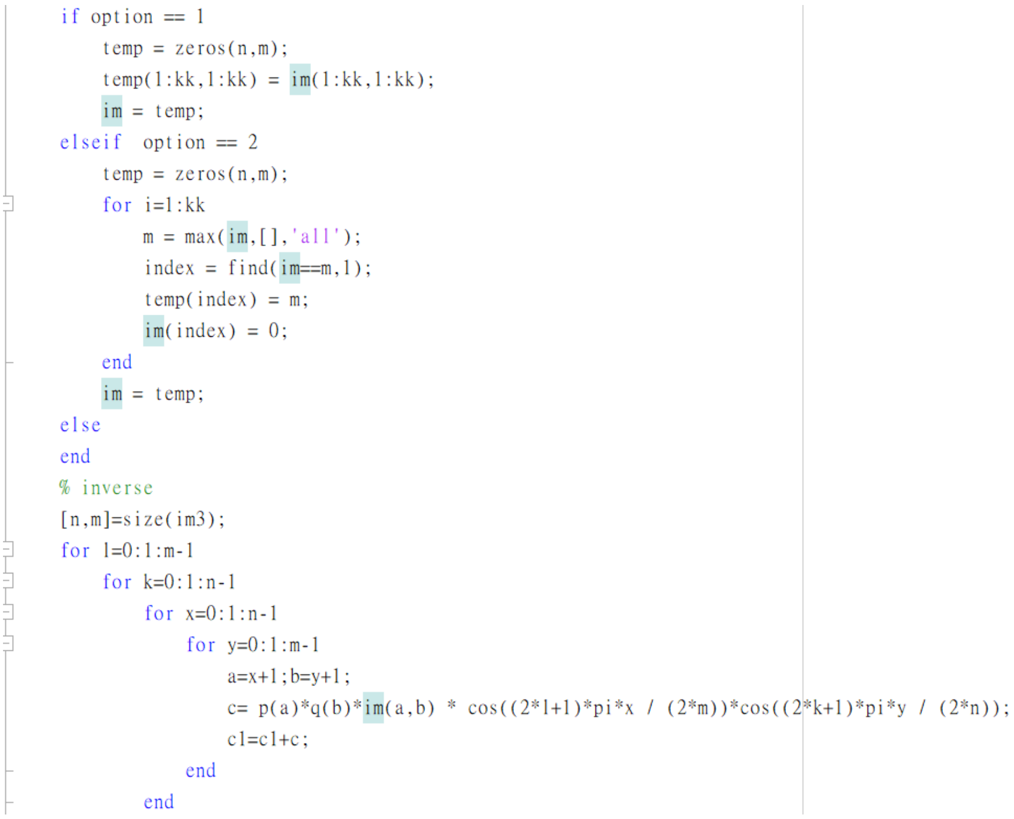


```
    elseif option == 2
        temp = zeros(n,m);
        for i=1:kk
            m = max(im,[],'all');
            index = find(im==m,1);
            temp(index) = m;
            im(index) = 0;
        end
        im = temp;
    else
        coe = im;
    end
    coe = im;
    %im = real(im);
    [n,m]=size(im3);
    % inverse
    for l=0:1:m-1
        for k=0:1:n-1
            for x=0:1:n-1
                for y=0:1:m-1
                    a=x+1;b=y+1;
                    c= im(a,b) * exp(1i*2*pi*(k*x/n + l*y/m));
                    c1=c1+c;
                end
            end
            aa=l+1;bb=k+1;
            reconstruct(bb,aa)=c1;
            c1=0;
        end
    end
    reconstruct = reconstruct/(n*m);
    %imshow(uint8(reconstruct));
end
```

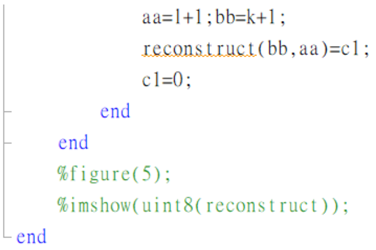


```
function [reconstruct, coe ]= DCT(im3, option, kk)
    [n,m]=size(im3);
    p = zeros(1,m)+sqrt(2/m);
    p(1)=1/sqrt(m);
    q = zeros(1,n)+sqrt(2/n);
    q(1)=1/sqrt(n);

    c1=0;
    k=1;l=1;
    for l=0:1:m-1
        for k=0:1:n-1
            for x=0:1:n-1
                for y=0:1:m-1
                    a=x+1;b=y+1;
                    c= im3(a,b) * cos((2*x+1)*pi*l / (2*m))*cos((2*y+1)*pi*k / (2*n));
                    c1=c1+c;
                end
            end
            aa=l+1;bb=k+1;
            im(bb,aa)=p(aa)*q(bb)*c1;
            c1=0;
        end
    end
    coe = im;
    %figure(4);
    %imshow(uint8(im));
```

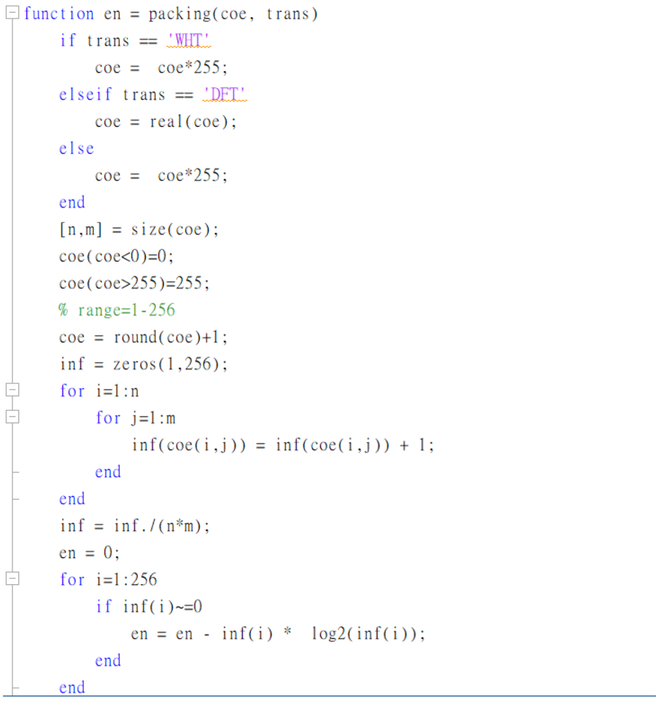


```
    if option == 1
        temp = zeros(n,m);
        temp(1:kk,1:kk) = im(1:kk,1:kk);
        im = temp;
    elseif  option == 2
        temp = zeros(n,m);
        for i=1:kk
            m = max(im,[],'all');
            index = find(im==m,1);
            temp(index) = m;
            im(index) = 0;
        end
        im = temp;
    else
    end
    % inverse
    [n,m]=size(im3);
    for l=0:1:m-1
        for k=0:1:n-1
            for x=0:1:n-1
                for y=0:1:m-1
                    a=x+1;b=y+1;
                    c= p(a)*q(b)*im(a,b) * cos((2*l+1)*pi*x / (2*m))*cos((2*k+1)*pi*y / (2*n));
                    c1=c1+c;
                end
            end
```

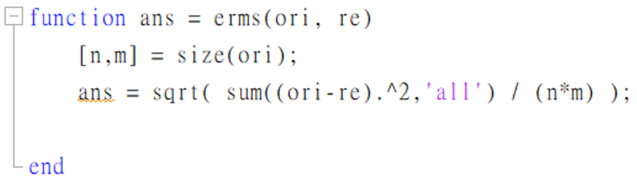


```
            aa=l+1;bb=k+1;
            reconstruct(bb,aa)=c1;
            c1=0;
        end
    end
    %figure(5);
    %imshow(uint8(reconstruct));
end
```

0516222 許芳瑀

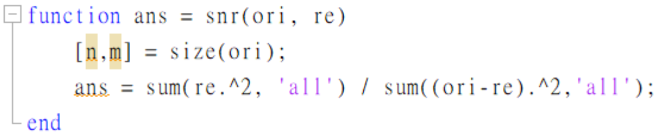


```
function en = packing(coe, trans)
    if trans == 'WHT'
        coe =  coe*255;
    elseif trans == 'DFT'
        coe = real(coe);
    else
        coe =  coe*255;
    end
    [n,m] = size(coe);
    coe(coe<0)=0;
    coe(coe>255)=255;
    % range=1-256
    coe = round(coe)+1;
    inf = zeros(1,256);
    for i=1:n
        for j=1:m
            inf(coe(i,j)) = inf(coe(i,j)) + 1;
        end
    end
    inf = inf./(n*m);
    en = 0;
    for i=1:256
        if inf(i)~=0
            en = en - inf(i) *  log2(inf(i));
        end
    end
```

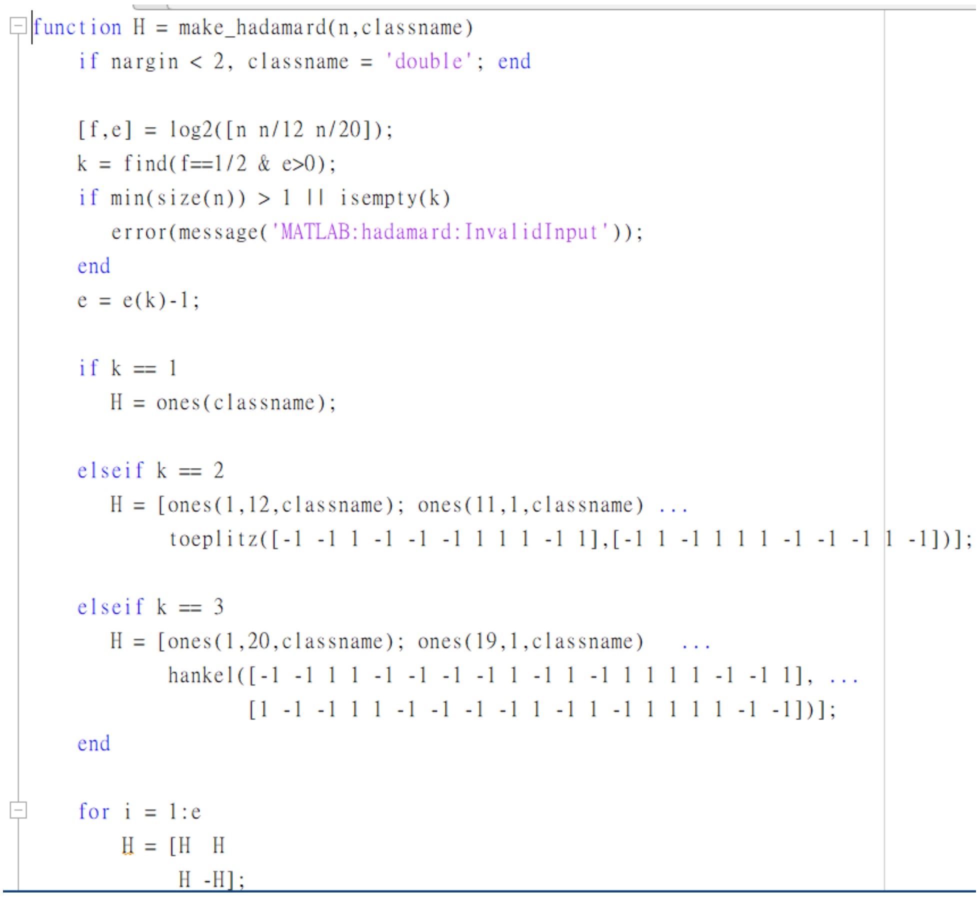


```
function ans = erms(ori, re)
    [n,m] = size(ori);
    ans = sqrt( sum((ori-re).^2,'all') / (n*m) );

end
```



```
function ans = snr(ori, re)
    [n,m] = size(ori);
    ans = sum(re.^2, 'all') / sum((ori-re).^2,'all');
end
```



```
function H = make_hadamard(n,classname)
    if nargin < 2, classname = 'double'; end

    [f,e] = log2([n n/12 n/20]);
    k = find(f==1/2 & e>0);
    if min(size(n)) > 1 || isempty(k)
       error(message('MATLAB:hadamard:InvalidInput'));
    end
    e = e(k)-1;

    if k == 1
       H = ones(classname);

    elseif k == 2
       H = [ones(1,12,classname); ones(11,1,classname) ...
            toeplitz([-1 -1 1 -1 -1 -1 1 1 1 -1 1],[-1 1 -1 1 1 1 -1 -1 -1 1 -1])];

    elseif k == 3
       H = [ones(1,20,classname); ones(19,1,classname)   ...
            hankel([-1 -1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1], ...
                   [1 -1 -1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1])];
    end

    for i = 1:e
        H = [H  H
             H -H];
```