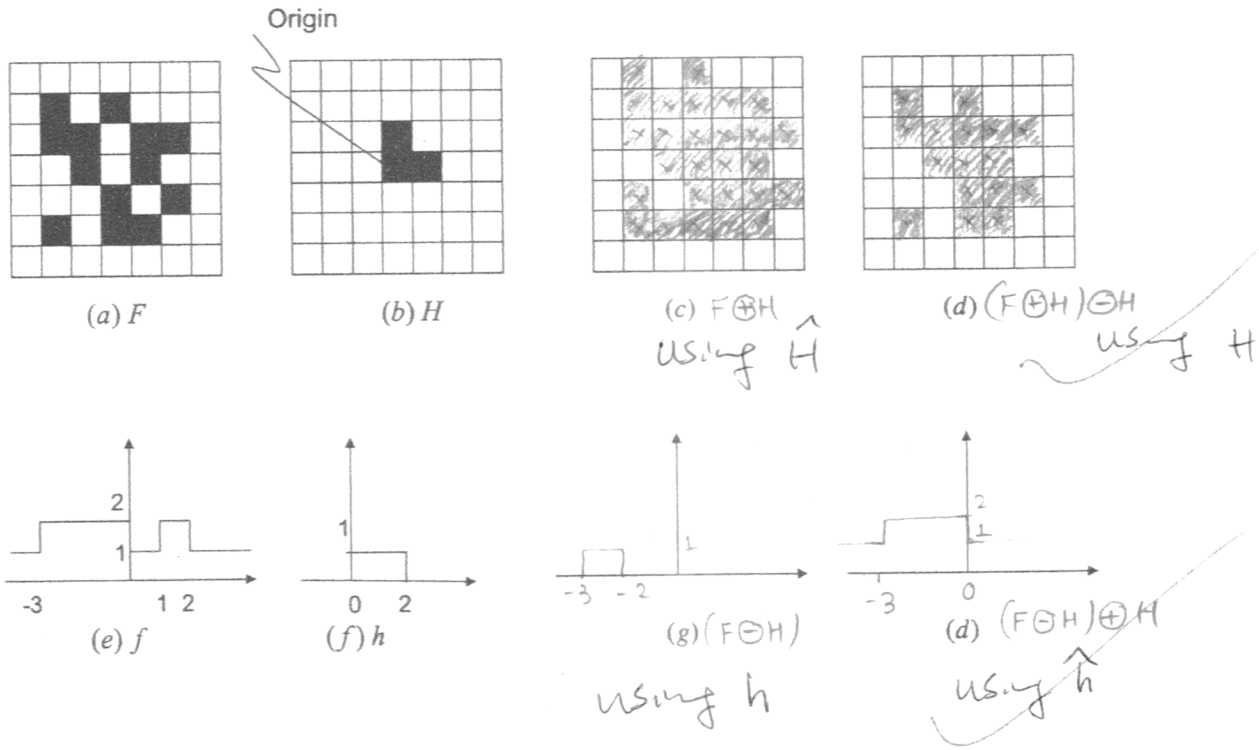
Final Exam (12/16/2013, 10:20AM-12:50PM)
Closed book, 1 sheet of notes (double sided) allowed.
No peeking into neighbors or unauthorized notes. No calculator or other electronics allowed.
Cheating will result in getting an F on the course.
Write your answers on this problem sheet for problems where space is provided. Otherwise write you answer in the blue book. Submit both the blue book and this problem set.

Your name: Solution _____

1. (20pt) (i) (10pt) Find the closing of the binary image, F, in Fig.(a) by the structuring element H in Fig.(b). You can use the grids in Fig.(c) and Fig.(d) to draw the intermediate and the final results. (ii) (10pt) Find the gray scale opening of the function sketched in Fig.(e) with the structural element in Fig.(f). Sketch the intermediate and final result in Fig.(g) and Fig.(h).



2. (15pt) Consider a 2D image f(x,y)=sin(6πx-4πy).

a) (3pt) Determine its Fourier transform F(u,v) and illustrate the spectrum (i.e., the impulses in the transform) in Fig. (a). Indicate the magnitude of each impulse. Please assume that x and u represent vertical position and frequency, respectively; and y and v represent horizontal position and frequency, respectively.

b) (3pt) Suppose this signal is sampled uniformly with sampling intervals Δx=Δy=Δ=1/5. Draw the spectrum of the sampled signal in Fig. (b). You only need to specify inside the frequency region defined by $|u| < f_s, |v| < f_s$, where $f_s=1/\Delta$.

c) (4 pt) Suppose the sampled signal is interpolated by an ideal low-pass filter $h_1(x,y)$ with frequency response

$$H_1(u,v) = \begin{cases} \Delta^2 & -f_s/2 < u,v < f_s/2, \\ 0 & otherwise \end{cases}$$

Draw the spectrum of the reconstructed signal in Fig. (c). Give the spatial representation of the reconstructed signal $f_{r1}(x,y)$.

d) (5pt) Suppose that a sample and hold filter is used instead. Its filter response can be written as

$$h_2(x,y) = h(x)h(y), \text{with } h(x) = \begin{cases} 1 & -\Delta/2 < x < \Delta/2, \\ 0 & otherwise \end{cases}$$

Determine the filter's frequency response, $H_2(u,v)$ (2pt). If the filter is further band-limited to the region described by $|u| < f_s, |v| < f_s$, give the spatial representation of the reconstructed signal $f_{r2}(x,y)$ (3pt). Note that you don't have to evaluate the value of $H_2(u,v)$ for any particular u or v. For example, for the value of $H(u,v)$ at u=1 and v=2, just write $H(1,2)$.

a) $f(x,y)=\sin(6\pi x-4\pi y)=\sin 2\pi(3x-2y)=\frac{1}{2j}\times[e^{j2\pi(3x-2y)}-e^{-j2\pi(3x-2y)}]$

$\therefore F(u,v)=\frac{1}{2j}[\delta(u-3,v+2)-\delta(u+3,v-2)]$

The magnitude of the two impulses are both $|\frac{1}{2j}|=\frac{1}{2}$

b) see the image. $f_s=\frac{1}{\Delta}=5$

c) There're only 2 points in the region. (2,2) and (-2,-2)

$F_r(u,v)=\frac{1}{2j}\times\frac{1}{\Delta^2}\times[-\delta(u-2,v-2)+\delta(u+2,v+2)]\times H_1(u,v)$

$\therefore f_r(x,y)=\sin 2\pi(-2x-2y)=-\sin(4\pi x+4\pi y)$

d) $h(x)=\begin{cases}1 & \frac{-\Delta}{2}<x<\frac{\Delta}{2}\\ 0 & otherwise\end{cases}\Rightarrow H(u)=\Delta\cdot sinc(\Delta\cdot u)=\frac{\sin(\Delta u\pi)}{\pi u}$

$\therefore H_2(u,v)=\frac{\sin(\Delta\pi u)}{\pi u}\times\frac{\sin(\Delta\pi v)}{\pi v}$

There're in total 4 pair points.

$F_{r1}(u,v)=\frac{1}{2j}\times\frac{1}{\Delta^2}\times[-\delta(u+3,v+3)+\delta(u-3,v-3)-\delta(u+3,v-2)+\delta(u-3,v+2)$
$+\delta(u+2,v+2)-\delta(u-2,v-2)+\delta(u+2,v-3)-\delta(u-2,v+3)]\,H(u,v)$

∵ $H_2(u,v)$ is symmetric, $H(3,3)=H(-3,-3)$ etc.

$\therefore f_r(x,y)=\frac{1}{\Delta^2}\times[H(3,3)\sin(6\pi x+6\pi y)_2+H(3,-2)\sin(6\pi x-4\pi y)+H(2,2)\sin(-4\pi x-4\pi y)$
$+H(2,-3)\sin(-4\pi x+6\pi y)]$

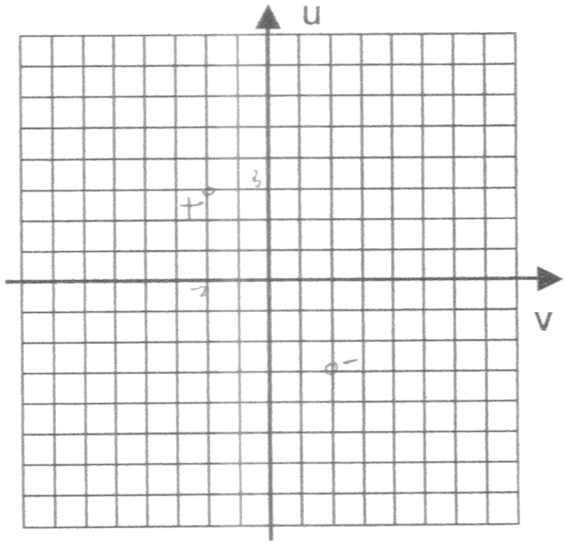


Fig. (a): Original Signal

(please see the dots; the lines are just for seeing pairs.)

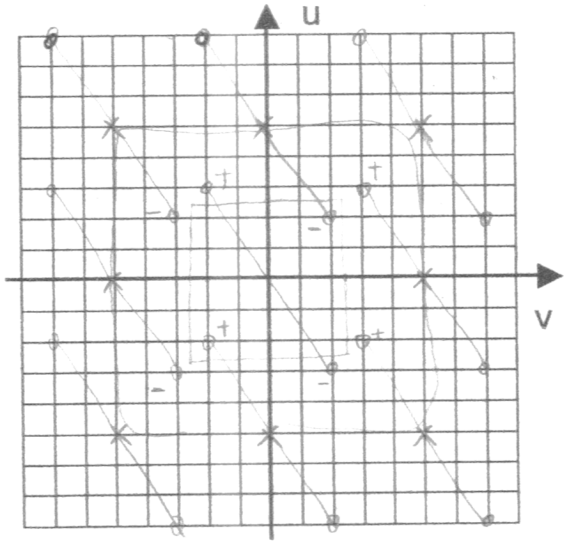


Fig.(b): Sampled Signal

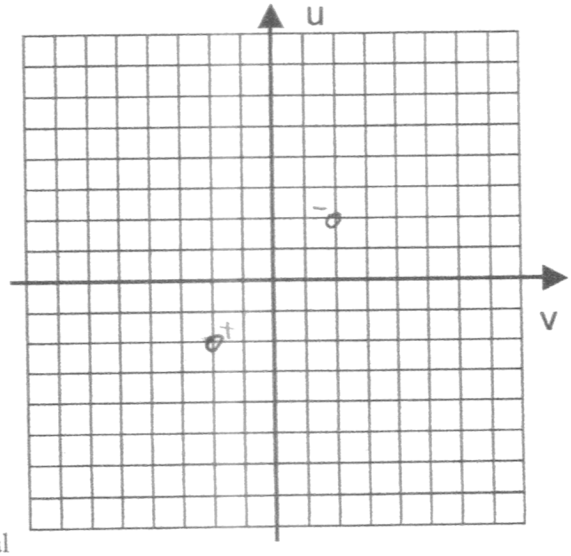


Fig. (c): Reconstructed Signal

Q.3

a) (0,10)
{ , (0,5), (0,3), (2,3), (1,3), (1,2), (0,1), (0,1), (2,1), (1,2), (3,1), (2,3), (0,5), EOB }
Here, 1st value is runlength & 2nd value is non-zero value.

b) runlength PDF

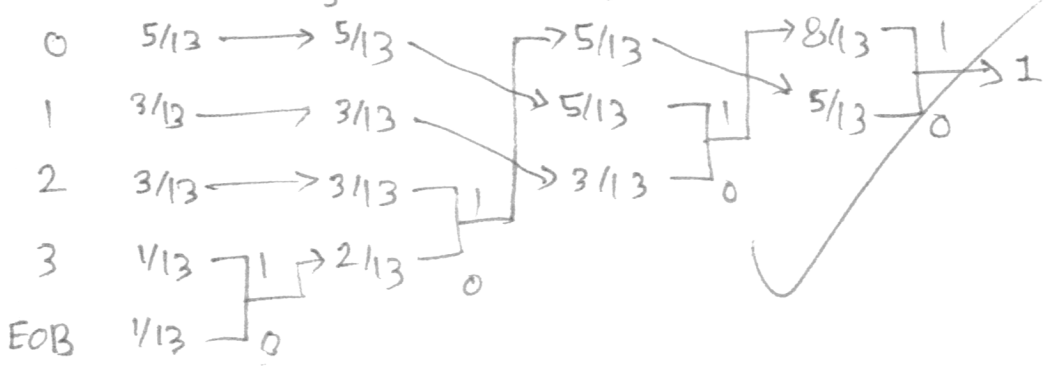
Q. 3

a) {10, 0, 5, 0, 3, 2, 3, 1, 3, 1, 2, 0, 1, 0, 1, 2, 1, 1, 2, 3, 1, 2, 3, 0, 5, EOB }
There is a bar under the runlength values.

b)

runlength	PDF	coeff.	PDF
0	5/13	10	1/13
1	3/13	5	2/13
2	3/13	3	4/13
3	1/13	2	2/13
EOB	1/13	1	4/13

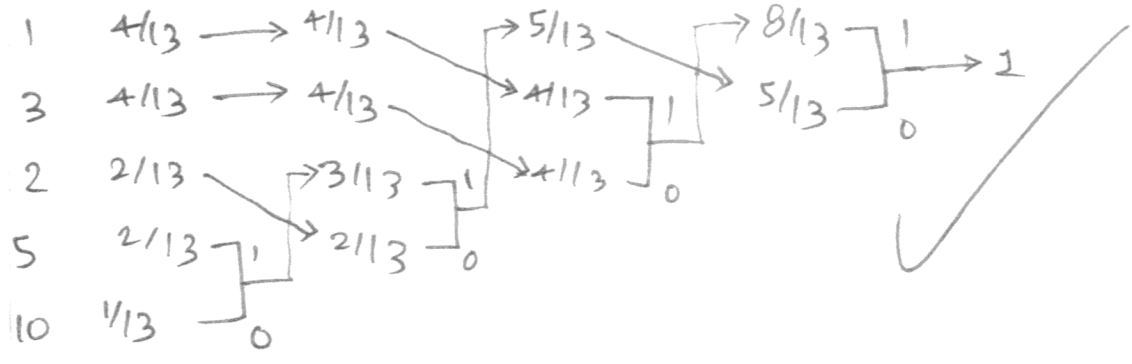
c) Huffman Coding for runlength:-



Runlength	Code
0	11
1	10
2	01
3	001
EOB	000

Huffman Coding for Coeff. :-

Coeff.



Coeff. ~~Runlength~~	Code
1	11
3	10
2	00
5	011
10	010

d) Binary representation :- 010 11 011 11 10 01 10 10 10 10
00 11 11 11 11 01 11 10 00 001 11 01 10 11 011
000

$\therefore$ Bit rate = $\frac{\text{Total no. of bits}}{\text{[illegible]}} = \frac{57}{64}$

4. The squared errors sum of the reconstructed samples are

$$\text{error} = \sum_k (f_k - \hat{f}_k)^2 = (f-\hat{f})^H (f-\hat{f})$$

The squared error sum of the transform coefficients are

$$\text{error}_1 = \sum_k (t_k - \hat{t}_k)^2 = (t-\hat{t})^H (t-\hat{t})$$

$\because$ The forward transform : $t = Af$. ie. $\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_k \end{bmatrix} = \begin{bmatrix} h_1^H \\ h_2^H \\ \vdots \\ h_k^H \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_k \end{bmatrix}$

A is unitary, i.e. $A^{-1} = A^H$, $AA^H = AA^{-1} = I$

$$\therefore \text{error}_1 = (t-\hat{t})^H (t-\hat{t})$$

$$= (Af - A\hat{f})^H (Af - A\hat{f}) = (f-\hat{f})^H A^H A (f-\hat{f})$$

$$= (f-\hat{f})^H (f-\hat{f}) = \text{error}$$

Alternatively, represent $\bar{f} = A^T \bar{t}$

$$A^T = [h_1, \cdots, h_k]$$

$$\text{Error in samples} = \|f - \hat{f}\|^2 = \|A^T \bar{t} - A^T \hat{t}\|^2$$

$$= \|A^T (t-\hat{t})\|^2 = (t-\hat{t})^T \underbrace{A \cdot A^T}_{I} (t-\hat{t})$$

$$= \|t-\hat{t}\|^2 = \text{Error in Coefficients}$$

5.

(a) ∵ The original values tend to distribute uniformly, while the prediction error has a very peaky distribution. And as the error values are usually very small, we can use less bits to encode them.

Symbols with a peaky distribution requires fewer bits to code than symbols w/ uniform distribution

In lossy coding, the quantization will make many small error values into zero without causing very much artifacts. We can use runlength + Huffman coding for example to decrease the bit rate.

The predictor can be designed linearly, i.e. $f_o = \sum_k a_k f_k$

f_o : current pixel, f_k : previous pixel.

It should be minimize the MSE by designing proper a_k.

Set $\frac{\partial \sigma_p^2}{\partial a_i} = 0$, $(\sigma_p^2 = (f_o - \sum_k a_k f_k)^2)$

we can get, if $\begin{bmatrix} R(1,1) & R(2,1) & \cdots & R(k,1) \\ R(1,2) & R(2,2) & \cdots & R(k,2) \\ \vdots & & & \vdots \\ R(1,k) & R(2,k) & \cdots & R(k,k) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} R(0,1) \\ R(0,2) \\ \vdots \\ R(0,k) \end{bmatrix}$ $\quad a = R^{-1}r$

Then the predictor is a linear minimum MSE (LMMSE predictor).

$R(m,n)$ is the correlation of f_m and f_n.

(b) The transform should generate many near-zero values, so that the signal energy in the transform domain is more concentrated or compact. After quantization many of the coefficients will become zero without causing much distortion. ② The tranform also needs to de-correlate the coefficients so that the coefficients can be coded independently. (decorrelation).

(c) ① Wavelet transform enables the user to have access to images at different resolution levels easily. By decoding more finer wavelet subbands, higher resolution can be obtained. (Spatial Scalability)

② Wavelet transform is more suitable for non-stationary signal, as its coefficients on the low frequency parts cover larger time span.

Disadvantage, it involves more complex caculation and longer time to compute.

Other advantages:

- No blocking artifacts associated with block transform coding at low bit rate
- Amplitude scalability

7 8. One solution (Direct computation of forward & inverse Haar transform)

```
[outfile, compression_ratio, PSNR] = Wavelet_Coding(infile, Q)
clear
clc
f = imread('barbara.png');        % infile
[H, W] = size(f);
LL = zeros(H/2, W/2);
LH = zeros(H/2, W/2);
HL = zeros(H/2, W/2);
HH = zeros(H/2, W/2);
fr = zeros(H, W);
for i = 1:2:H
    j = 1:2:W
    iblk = (i-1)/2+1 ; jblk = (j-1)/2 + 1
LL(iblk, jblk) = sum(sum(f(i:i+1, j:j+1)))/2;
LH(iblk, jblk) = (sum(f(i, j:j+1)) - sum(f(i+1, j:j+1)))/2;
HL(iblk, jblk) = (sum(f(i:i+1, j)) - sum(f(i:i+1, j+1)))/2;
HH(iblk, jblk) = ((f(i,j) + f(i+1, j+1)) - (f(i, j+1) + f(i+1, j)))/2;
end ; end;
LLg = floor(LL./Q).*Q + Q/2 ;
LHg = floor((LH + Q/2)./Q).*Q ;
HLg = floor((HL + Q/2)./Q).*Q ;
HHg = floor((HH + Q/2)./Q).*Q ;
```

```
count1 = 0;  count2 = 0;  count3 = 0;  count4 = 0;
for  m = 1 : H/2
     n = 1 : w/2
     if (LLg(m,n) ≠ 0)
          count1 = count1 + 1;
     else
          continue    end
     if (LHg(m,n) ≠ 0)
          count2 = count2 + 1;
     else
          continue    end
     if (HLg(m,n) ≠ 0)
          count3 = count3 + 1;
     else
          continue    end
     if (HHg(m,n) ≠ 0)
          count4 = count4 + 1;
     else
          continue
          end
count = count1 + count2 + count3 + count4;
compression_ratio = HW / count;

for k=1:2:H
     for H=1:2w
     kblk = (k-1)/2 + 1;
```

Can be replaced by

```
Count = sum(sum(
sum(sum(LLg(x,y)~=0))
+ sum(sum(LHg(x,y)~=0))
+ sum(sum(HLg(x,y)~=0))
+ sum(sum(HHg(x,y)~=0))
```

```
lblk = (l-1)/2 + 1
fr(k, l) = (LLg(kblk, lblk) + LHg(kblk, lblk + HLg(kblk, lblk)
           + HHg(kblk, lblk))/2;
fr(k, l+1) = (LLg(kblk, lblk) + LHg(kblk, lblk) - HLg(kblk, lblk)
             - HHg(kblk, (blk1))/2;
fr(k+1, l) = (LLg(kblk, lblk) - LHg(kblk, lblk) + HLg(kblk, lblk)
             - HHg(kblk, (blk1))/2;
fr(k+1, l+1) = (LLg(kblk, lblk) - LHg(kblk, lblk) - HLg(kblk, lblk)
               + HHg(kblk, lblk)))/2;
                                        end ; end

fr = round(fr);
fr = uint8(fr)
  save 'fr.m' -> imwrite(fr, 'outfile', 'jpg');

MSE = mean2((f - fr).^2);
PSNR = 10 * log10(255 * 255 / MSE);
```

Alternate solution (Using separable computation of Haar transform)

7. [outfile, compression_ratio, PSNR] = waveletCoding (infile, Q)

```
inimg = imread('infile'); inimg = double(inimg);   % suppose infile is the path
[M,N]=size(inimg); M1= floor(M/2); N1=floor(N/2);
L2 = inimg(:,1:2:2*N1) inimg(:,2:2:N);
H2 = inimg(:,1:2:2*N1) inimg(:,2:2:N);
LL = (L2(1:2:2*M1,:) + L2(2:2:M,:))/2;
LH = (L2(1:2:2*M1,:) - L2(2:2:M,:))/2;
HL = (H2(1:2:2*M1,:) + H2(2:2:M,:))/2;
HH = (H2(1:2:2*M1,:) - H2(2:2:M,:))/2;
count1(M1,N1)=zeros(M1,N
count2(M1,N1)=zeros(M1,N
count3 = count1;
count4 = count1;
nonzero=0;
% Quantize
for i=1:M1
  for j=1:N1
    LL(i,j) = floor(LL(i,j)/Q)*Q + Q/2;
    LH(i,j) = floor(abs(LH(i,j))+Q/2)* sgn(LH(i,j));
    HL(i,j) = floor(abs(HL(i,j))+Q/2) * sgn(HL(i,j));
    HH(i,j) = floor(abs(HH(i,j))+Q/2) * sgn(HH(i,j));
if(LL(i,j)~=0) count1(i,j) = count(i,j)+1; end if(LH(i,j)~=0) count2(i,j)=1; end
if(HL(i,j)~=0) count3(i,j)=1 end if(HH(i,j)~=0) count4(i,j)=1; end count=[count1,count2,count3,coun
for i=1:4 nonzero = sum(sum(count[i]) compression_ratio = double((M*N)/nonzero
  +nonzero; end end
```

-1

```
% reconstruct
output = zeros(M,N); L2 = zeros(M,N1); H2 = zeros(M,N1);
L2(1:2:M*2,:) = (LL+LH)/2;
L2(2:2:M*2,:) = (LL-LH)/2
H2(1:2:M*2,:) = (HL+HH)/2;
H2(2:2:M*2,:) = (HL-HH)/2;
output(:,1:2:N*2) = L2+H2;
output(:,2:2:M*2) = L2-H2;
output = uint8(output); imwrite(output, outputfile, '');
MSE = mean2((inimg-output).^2);   % suppose outfile is the path
PSNR = 10*log10(255*255/MSE);
```