SOLUTIONS - DIGITAL IMAGE PROCESSING

E 4 08/ ISE 4.33/2014 2007

QUESTION 1

(a) (i) Because the inverse matrix is obtained directly from the complex conjugate and transpose of the forward matrix. **[3]**

(ii) It will exhibit circular symmetry as well. **[3]**

(b) (i) The 2-D Hadamard transform is defined as

$$H(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\left[\prod_{i=0}^{n-1}(-1)^{(b_i(x)b_i(u)+b_i(y)b_i(v))}\right] \text{ or}$$

$$H(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)(-1)^{\sum_{i=0}^{n-1}(b_i(x)b_i(u)+b_i(y)b_i(v))}$$

$$f(x,y)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}H(u,v)\left[\prod_{i=0}^{n-1}(-1)^{(b_i(x)b_i(u)+b_i(y)b_i(v))}\right] \quad \textbf{[3]}$$

(ii) Find the Hadamard Transform of the following image

$$\begin{bmatrix}5&6&8&10\\6&6&5&7\\4&5&3&6\\8&7&5&5\end{bmatrix}.$$

[Hint: You may use the recursive relation property of the Hadamard matrix $H_{2N}=\begin{bmatrix}H_N & H_N\\ H_N & -H_N\end{bmatrix}$ knowing that the 2×2 Hadamard matrix is $H_2=\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}$].

By using the recursive relationship we find that

$$H_4=\frac{1}{\sqrt{2}}\begin{bmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{bmatrix}.$$

By using the fact that the transform is separable and symmetric we may use the following formula

$$H(u,v)=H_4\begin{bmatrix}5&6&8&10\\6&6&5&7\\4&5&3&6\\8&7&5&5\end{bmatrix}H_4^T$$

in which case we get

$$H(u,v)=\frac{1}{2}\begin{bmatrix}96&-8&-2&6\\-2&-6&-12&0\\10&-2&-12&0\\12&4&-2&-2\end{bmatrix} \quad \textbf{[4]}$$

(c) (i) Find the covariance matrix of the population $\underline{g}$. The eigenvalues of $\underline{C_f}$ are found to be $\lambda_1=0.1051$, $\lambda_2=0.16$, $\lambda_2=0.3349$. Therefore the covariance matrix of the population $\underline{g}$ is

$$\underline{C_g}=\begin{bmatrix}0.1051&0&0\\0&0.16&0\\0&0&0.3349\end{bmatrix} \quad \textbf{[4]}$$

(ii) In the case of reconstruction using 1 image the error is $0.1051+0.16=0.2651$. In the case of reconstruction using 2 images the error is 0.1051. **[3]**

QUESTION 2

(a) Histogram equalization of the original $s = T(r) = \int_0^r p_r(w)dw = \int_0^r 4w^3 dw = r^4,\ 0 \le r \le 1$

Histogram equalization of the desired $v = G(z) = \int_0^z 2w dw = z^2$

The two equalized images can be considered the same and therefore, $r^4 = z^2 \Rightarrow z = r^2$

[4]

(b) $\mu = \frac{1}{MN}\sum_{n=0}^{255} nh(n)$ with $h(n)$ the number of pixels in $f(x,y)$ with value n.

$$\sigma^2 = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f(x,y)-\mu)^2 = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)^2 - \mu^2 = \frac{1}{MN}\sum_{n=0}^{255} n^2 h(n) - \mu^2$$ [4]

(c) For each of the image disturbances listed below propose an appropriate enhancement technique.

(i) Salt and pepper noise.

The proper technique for this is median filtering. The median m of a set of values is such that half the values in the set are less than m and half are greater than m. Median filtering means that each pixel is replaced by the median of the grey level in the neighborhood of that pixel.

Median filters are non linear filters because for two sequences $x(n)$ and $y(n)$

$$\text{median}\{x(n)+y(n)\} \neq \text{median}\{x(n)\} + \text{median}\{y(n)\}$$

Median filters are useful for removing isolated lines or points (pixels) while preserving spatial resolutions. They perform very well on images containing binary (salt and pepper) noise but perform poorly when the noise is Gaussian.

Their performance is also poor when the number of noise pixels in the window is greater than or half the number of pixels in the window. [4]

(ii) Uncorrelated zero mean additive noise.

The proper technique is a low-pass spatial mask. A spatial mask operating on an image can produce a smoothed version of the image (which contains the low frequencies) if the coefficients of the mask have certain properties as for example if they are all positive.

The drawback is that the mask removes the high frequency information of the image as well. [4]

(iii) Low contrast where edge detail should be enhanced but region information should be preserved.

The proper technique is high boost filtering. A high pass filtered image may be computed as the difference between the original image and a low-pass filtered version of that image as follows:

High-pass=Original-Low-pass

Multiplying the original image by an amplification factor denoted by A, yields a high boost filter:

High-boost=(A) (Original)-Low-pass

$=(A-1)$ (Original)+Original-Low-pass

$=(A-1)$ (Original)+High-pass

The general process of subtracting a blurred image from an original as given in the first line is called un-sharp masking. A possible mask that implements the above procedure could be the one illustrated below.

[4]

QUESTION 3

(a) (i) Wiener filter in space

$$\mathbf{W} = \mathbf{R}_{\mathbf{fy}}\mathbf{R}_{\mathbf{yy}}^{-1} = \mathbf{R}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}}(\mathbf{H}\mathbf{R}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}} + \mathbf{R}_{\mathbf{nn}})^{-1}$$

Estimate for the original image is

$$\hat{\mathbf{f}} = \mathbf{R}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}}(\mathbf{H}\mathbf{R}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}} + \mathbf{R}_{\mathbf{nn}})^{-1}\mathbf{y}$$

Note that knowledge of $\mathbf{R}_{\mathbf{ff}} = E\{\mathbf{f}\mathbf{f}^{\mathbf{T}}\}$ and $\mathbf{R}_{\mathbf{nn}} = E\{\mathbf{n}\mathbf{n}^{\mathbf{T}}\}$ is assumed.

In frequency domain

$$W(u,v) = \frac{S_{ff}(u,v)H^*(u,v)}{S_{ff}(u,v)|H(u,v)|^2 + S_{nn}(u,v)}$$

$$\hat{F}(u,v) = \frac{S_{ff}(u,v)H^*(u,v)}{S_{ff}(u,v)|H(u,v)|^2 + S_{nn}(u,v)}Y(u,v)$$

The noise variance has to be known, otherwise it is estimated from a flat region of the observed image.

In practical cases where a single copy of the degraded image is available, it is quite common to use $S_{yy}(u,v)$ as an estimate of $S_{ff}(u,v)$. This is very often a poor estimate.

[5]

(ii) If $S_{nn}(u,v) = 0 \Rightarrow W(u,v) = \frac{1}{H(u,v)}$ which is the inverse filter. If $S_{nn}(u,v) \to 0$

$$\lim_{S_{nn}\to 0} W(u,v) = \begin{cases} \dfrac{1}{H(u,v)} & H(u,v) \neq 0 \\ 0 & H(u,v) = 0 \end{cases}$$

which is the pseudo-inverse filter.

[5]

(b) (i) The noise is white with variance σ_n^2 and therefore the autocorrelation function of the noise is

$$R_{nn}(k,l) = \underset{(x,y)}{E}\{n(x,y)n(x+k,y+l)\} = \sigma_n^2\delta(k,l)$$

and the power spectrum of the noise is σ_n^2. In that case

$$S_{ff}(u,v) = \frac{(1-\rho_1^2)(1-\rho_2^2)}{(1-\rho_1^2)(1-\rho_2^2) + \sigma_n^2[1+\rho_1^2 - 2\rho_1\cos(u)][1+\rho_2^2 - 2\rho_2\cos(v)]}$$

[5]

(ii) Develop a method of estimating ρ_1 and ρ_2 from $g(x,y)$.

$$R_{gg}(k,l) = R_{ff}(k,l) + R_{nn}(k,l) = \rho_1^{|k|}\rho_2^{|l|} + \sigma_n^2\delta(k,l)$$

If we choose three pairs of values for the parameters k,l as $(k,l) = (1,0)$, $(k,l) = (0,1)$ and $(k,l) = (0,0)$ we have the following relationships:

For $(k,l) = (1,0)$: $R_{gg}(1,0) = \rho_1$ where $R_{gg}(1,0)$ is estimated from the available data.

For $(k,l) = (0,1)$: $R_{gg}(0,1) = \rho_2$ where $R_{gg}(0,1)$ is estimated from the available data.

For $(k,l) = (0,0)$: $R_{gg}(0,0) = 1 + \sigma_n^2 \Rightarrow \sigma_n^2 = R_{gg}(0,0) - 1$ where $R_{gg}(0,0)$ is estimated from the available data

[5]

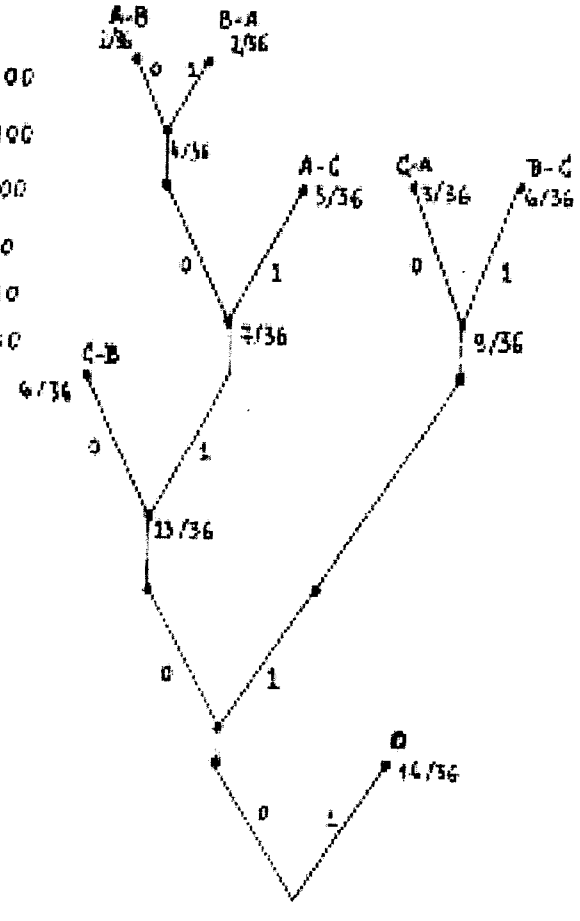
QUESTION 4

(i)

Symbol	Probability
A	1/6
B	2/6
C	3/6

Differential Coding Symbol	Probability
A-A = 0	1/36
A-B	2/36
A-C	5/36
B-A	2/36
B-B = C	4/36
B-C	6/36
C-A	5/36
C-B	6/36
C-C = 0	9/36

Differential Coding Symbol	Probability	Huffman
0	14/36	1
A-B	2/36	00100
B-A	2/36	10100
A-C	5/36	1100
C-A	5/36	010
B-C	6/36	110
C-B	6/36	000



(ii)

Entropy $= \sum -p_i \ln p_i = 2.4524$

$L_{avg} = \sum p_i \ell_i = 2.528$

redundancy = Entropy − L_{avg}

(b)

(i) The lossless compression method within JPEG is fully independent from transform-based coding. It uses differential coding to form prediction residuals that are then coded with either a Huffman coder or an arithmetic coder. The prediction residuals usually have a lower entropy; thus, they are more amenable to compression than the original image pixels.
In lossless JPEG, one forms a prediction residual using previously encoded pixels in the current line and/or the previous line. The prediction residual for pixel x in Figure is defined as $r = y - x$ where y can be any of the following functions:
$y = 0$, $y = a$, $y = b$, $y = c$, $y = a + b - c$, $y = a + (b - c)/2$, $y = b + (a - c)/2$, $y = (a + b)/2$
Note that, pixel values at pixel positions a, b, and c, are available to both the encoder and the decoder prior to processing x. The particular choice for the y function is defined in the scan header of the compressed stream so that both the encoder and the decoder use identical functions. Divisions by two are computed by performing a one-bit right shift.

The prediction residual is computed modulo 2. This residual is not directly Huffman coded. Instead, it is expressed as a pair of symbols: the category and the magnitude. The first symbol represents the number of bits needed to encode the magnitude. Only this value is Huffman coded. The magnitude categories for all possible values of the prediction residual are shown in Table 2.2. If, say, the prediction residual for x is 42, then from Table 2.2 we determine that this value belongs to category 6; that is, we need an additional six bits to uniquely determine the value 42. The prediction residual is then mapped into the two-tuple (6, 6-bit code for 42). Category 6 is Huffman coded, and the compressed representation for the prediction residual consists of this Huffman codeword followed by the 6-bit representation for the magnitude. In general, if the value of the residual is positive, then the code for the magnitude is its direct binary representation. If the residual is negative, then the code for the magnitude is the one's complement of its absolute value. Therefore, codewords for negative residual always start wish a zero bit.

Compression is based on the issue that small categories will occur with significantly higher probabilities

(ii) $a = 100$, $b = 191$ and $x = 180$. Let $y = (a + b)/2$; then $y = 145$, and the prediction residual is $r = 145 - 180 = -35$. -35 belongs to category 6. The binary number for 35 is 100011, and its one's complement is 011100. Thus, -35 is represented as (6,011100). If the Huffman code for six is 1110, then -35 is coded by the 10-bit codeword 1110011100. Without entropy coding, -35 would require 16 bits.