IMAGE PROCESSING (RRY025)

LECTURE 17

TRANSFORMS/IMAGE COMPRESSION III

Karhunen-Loeve Transform (KLT)

- The DCT is better than DFT for compressing information into a few components - is there some theoretically optimum method of compression? Answer is the Karhunen-Loeve Transform (KLT), also known as the Hotelling transform or Principal Component Analysis (PCA).

- The KLT analyzes a set of vectors or images, into basis functions or images *where the choice of the basis set depends on the statistics of the image set* - depends on image covariance matrix.

Consider a set of vectors (corresponding for instance to rows of an image)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix}$$

First form mean vector of population

$$\mathbf{m_x} = E[\mathbf{x}]$$

where $E[\quad]$ is the 'expectation' or mean operator.

- We define the *covariance matrix*

$$\mathbf{C_x} = E\left[(\mathbf{x} - \mathbf{m_x})(\mathbf{x} - \mathbf{m_x})^{\mathbf{T}}\right]$$

which equals the **outer product** of the vector $\mathbf{x} - \mathbf{m_x}$ with itself. Hence if $\mathbf{x}$ is a length N vector, C_x is a NxN symmetric matrix such that $C_x{}^T = C_x$.

- Multiplying the above out we get

$$\mathbf{C_x} = E\left[\mathbf{x}\mathbf{x}^{\mathbf{T}}\right] - E\left[\mathbf{x}\mathbf{m_x}{}^{\mathbf{T}}\right] - E\left[\mathbf{m_x}\mathbf{x}^{\mathbf{T}}\right] + E\left[\mathbf{m_x}\mathbf{m_x}{}^{\mathbf{T}}\right]$$

$$\mathbf{C_x} = E\left[\mathbf{x}\mathbf{x}^{\mathbf{T}}\right] - E\left[\mathbf{m_x}\mathbf{m_x}{}^{\mathbf{T}}\right]$$

Example

- Compute the covariance matrix of a set of 4 vectors

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

- then $\mathbf{m_x} = \frac{1}{4}\begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$ and so $\mathbf{m_x}\mathbf{m_x}{}^{\mathbf{T}} = \frac{1}{16}\begin{bmatrix} 9 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix}$

To calculate $\mathbf{C_x}$ also need to know $E[\mathbf{x}\mathbf{x}^{\mathbf{T}}]$, the mean of the outer products of each vector with itself.

$$\mathbf{x_1}\mathbf{x_1}{}^{\mathbf{T}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{x_2}\mathbf{x_2}{}^{\mathbf{T}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{x_3}\mathbf{x_3}{}^{\mathbf{T}} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{x_4}\mathbf{x_4}{}^{\mathbf{T}} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

then $E[\mathbf{x}\mathbf{x}^{\mathbf{T}}] = \frac{1}{4}\begin{bmatrix} 3 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

- It follows that $\mathbf{C_x} = E[\mathbf{m_x m_x^T}] - \mathbf{E}[\mathbf{xx^T}]$ equals,

$$\mathbf{C_x} = \frac{1}{16}\begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

- The diagonal elements of $\mathbf{C_x}$ show that the mean variance of each pixel about its mean is the same, 3/16.

- The non-zero off-diagonal elements $\mathbf{C_x}$ show that the pixels are not independent. Pixels 1 and 2 are positively correlated (so when pixel 1 is greater than its average value then usually pixel 2 is also greater than its average value) but pixels 2 and 3 are negatively correlated (so if pixel 2 is more than its average then pixel 3 is usually less than its average) .

- Given the covariance matrix C_X the KLT transformation matrix A is defined such that after transformation of each vector via;

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{x_m})$$

the covariance matrix of the vectors $\mathbf{y}$ which we denote C_y is diagonal - so that each pixel of the output vector set is uncorrelated with every other pixel.

- We can show that for any $\mathbf{A}$ the covariance of the original vectors $\mathbf{x}$ and the transformed vectors $\mathbf{y}$ are related via.

$$\mathbf{C_y} = \mathbf{A C_x A^T}$$

- It can be shown that if we choose the rows of $\mathbf{A}$ to be the eigenvectors of C_x then C_y is diagonal. Hence this choice of transformation matrix defines the Hotelling transform.

- To see this choice of $\mathbf{A}$ is the correct one. Consider matrix $\mathbf{A}$ whose rows are the normalized (total vector length 1) eigenvectors of $\mathbf{C_x}$, $\mathbf{e_1}$, $\mathbf{e_2}$... placed in order of decreasing eigenvalue $\lambda_1 > \lambda_2 > \lambda_2$... Then

$$C_y = \begin{bmatrix} -- & e_1 & -- \\ -- & e_2 & -- \\ -- & e_3 & -- \end{bmatrix} C_x \begin{bmatrix} | & | & | \\ e_1 & e_2 & e_3 \\ | & | & | \end{bmatrix}$$

- Multiplying the last two matrices first it follows that

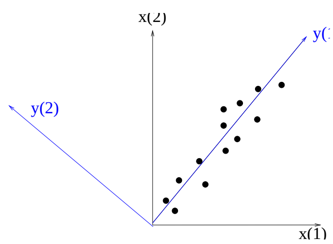
$$C_y = \begin{bmatrix} -- & e_1 & -- \\ -- & e_2 & -- \\ -- & e_3 & -- \end{bmatrix} \begin{bmatrix} | & | & | \\ \lambda_1 e_1 & \lambda_2 e_2 & \lambda_3 e_3 \\ | & | & | \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

where we use the fact that the eigenvectors of a symmetric matrix like C_x are all mutually orthogonal.

Geometrical Interpretation

- From its definition the transformation matrix $\mathbf{A}$ corresponds to a rotation of the coordinate system. A given input vector $\mathbf{x}$ is transformed to an output vector $\mathbf{y}$ via $\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{x_m})$

- For set of length-2 vectors $\mathbf{x}$, plot coordinates $x(1)$ and $x(2)$. Hotelling transform finds rotated coordinate system with first axis $y(1)$ is aligned so that the data has the largest variation, and in which there is no-correlation between $y(1)$ and $y(2)$.



- For a set of 3 pixel images as in earlier example, allocate each image in the set a point in 3D space, now find a rotated coordinate system so that the new y(1) -axis is along the direction of maximum variance, new y(2) -axis along an orthogonal direction with next highest variance etc. New y(1), y(2), y(3) axes define basis vectors of KLT.

Optimum Compression

- It can be shown theoretically that the ID transform using the KLT matrix **A**, is optimum in packing as much information (largest coefficients) as possible into as few components as possible.

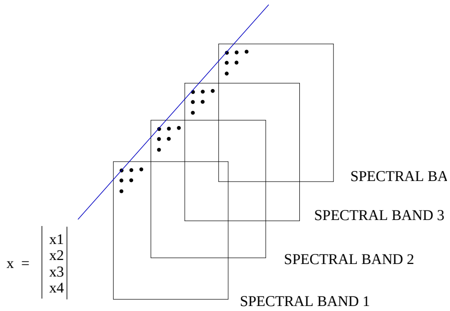
- A 2D version is (theoretically) possible by considering a set of 256 by 256 pixel images as a set of length 65526 (= 256 x256) images. From this set a 65526 x 65526 covariance matrix can be calculated its eigenvectors found and a transforms matrix calculated (not practical in reality!!).

- However can form analytic solutions for KLT transform matrix if we assume covariance between two adjacent pixels in ρ and covariance between two pixels n_{pix} apart is $\rho^{n_{pix}}$. But no fast transform exists (and we find for $\rho > 0.9$ that the basis functions are close to Cosine).

- A practical application to compression however is if we consider data cubes (i.e two planes of space and say another of frequency band), where the 'depth' of the cube is relatively small, and compress in that third dimension. In this case consider each vector formed at each pixel as an image and the collection of these 1D vectors over the image as the 'image set'.

KLT/PCA of Remote Sensing Images

- KLT (often called Principal Component Analysis, PCA) is used for both compressing and sumarising information in multi-spectral remote sensing data. Say we have 6 wavelengths. Can form a large set of length 6 vectors from this. Work out PCA transformation matrix and apply it to spectral data at each pixel.



- Note different from previous examples where the vectors formed from x,y coordinates of different features within 2D images. Here vectors formed at each pixel from the spectral variation. Transformation to principal components. concentrates information into first few image planes, making task of say automated analysis for landform classification easier. Would also be useful for compression (only need to star and transmit most significant principle component images).

Transform Coding

- **Type of Transform** Have considered in previous lectures different types of image transform, eg DFT, Hadamard, Cosine, KLT. If adjacent pixels highly correlated - transforms concentrate power into a few components,

```
    Input                   DCT transform

 5  4  3  4                57 -2  5   2
 4  3  5  5                10  3  1   0
 4  3  4  4                -7  3  1  -4
 2  2  2  3                 1  4  0  .1
```

- Theoretically if we model image as a noise like signal with high degree of covariance (ρ, similarity) between adjacent pixels, can ask which transform is best at packing energy into a few components. Answer is KLT, but for this need to know or assume statistics of interpixel correlation, and no fast transforms exist. It was found that for $\rho > 0.9$, typical of images basis functions almost identical to DCT so this was adopted in first generation compressors (like JPEG)

- In fact that most images are well modelled as noise like signals is not correct(!), In particular image statistics vary over image from object to object. Hence later in 1990's found wavelet transforms to be more efficient (basis of JPEF2000).

Block Coding

- **Block Transforms** Can divide NxN image up into blocks of size MxM. Do cosine transform on each block separately, compress, transmit, reform image, Cosine transform takes $M^2 logM$ per block and there are $(N/M)^2$ blocks. Final number of operations is

$$N^2 LogM$$

- So for speed make M as small as possible? Must be a limit since compression comes from inter-pixel redundancy.

- If covariance between adjacent pixels is 0.9, and covariance depends only on distance between pixel, find $0.9^8 \approx 0.5$, only weakly correlates. Using $M = 8$ is reasonable.

- Can think of block DCT as a 'windowed DCT' an crude approximation to a wavelet with sorting data as a function of both image position - and spatial frequency at that position.

- Biggest advantage of block transforms, is they can be made adaptive to image, in regions with lots of high spatial frequencies (block on 'Scotsmans kilt') keep the high spatial frequencies, in smooth regions can be more aggressive in eliminating high spatial frequencies .

Choosing Components -Zonal Mask

Transform Compression works by only sending part of the transform but which parts to send? How does transmitter tell receiver which components are being sent (gives overhead and reduces compression).

- **Zonal Mask** Simplest, just transmit transform coefficients within a masked region of transform. This mask is usually the same for all of the individual blocks Various options for selecting region.
- **Fixed zonal mask**, send top left corner of transform, geometry of mask embedded in application software, so for a given amount of compression (communicated in file header, decoder knows the geometry of transmitted regions and sequence in which these are transmitted, usually zig-zag).
- **Zonal Bit allocation** - as above but vary number of bits used to represent each coefficient with position in transform (accomplished using a re-quantizer optimised to statistics of each coefficient.
- **Zonal mask based on variance** - Look at variance in value of coefficients at different u,v positions over all the blocks, select for a given compression the appropriate number of highest variance coefficients to transmit (used in m-file example **shodct**).

Fixed Zonal Mask

1	1	1	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Zonal Bit allocation

8	7	6	4	3	2	1	0
7	6	5	4	3	2	1	0
6	5	4	3	3	1	1	0
4	4	3	3	2	1	0	0
3	3	3	2	1	0	0	0
2	2	1	1	1	0	0	0
1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0

Zonal Mask based on inter-block variance

1	1	0	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Zig-zag coding Sequence

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Choosing Components - Threshold Mask

- **Threshold Mask** Send different transform coefficients for each block. Is adaptive to image structure (sends different coefficients for a block on Scotsman kilt, compared to a block on background sky) - this factor increases compression for fixed quality or increases quality for fixed compression factor. On the other hand there is some overhead in telling the decoder which coefficients were transmitted, which decreases compression ratio. Variants include
- **Fixed Number** Sending for each block the a fixed number of the N_{send} largest transform coefficients, method has fixed compression ratio. Send by scanning block in a zig-zag pattern, setting unselected coefficients to zero and using a type of run length code to indicate number of zeros before each non-zero value.
- **Variable number** Send for each block all coefficients above some threshold (usually as in JPEG threshold depends on position within block, in a pattern that represents human vision system sensitivity to different spatial frequencies.

- JPEG uses (for intensity) a threshold mask like

JPEG Threshold Matrix Z(u,v)

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

- Form $T'(u,v) = round{-}down[T(u,v)/Z(u,v)]$ where $T(u,v)$ is the transform of the block. Here $Z(u,v)$ acts as variable threshold mask, transform values below this value become zero and are not sent. The $Z(u,v)$ also controls the accuracy of the coefficients that are sent. After putting into an intermediate run length code at the receive end the $T'(u,v)$ are multiplied by $Z(u,v)$ placed in matrix before the inverse DCT is performed.

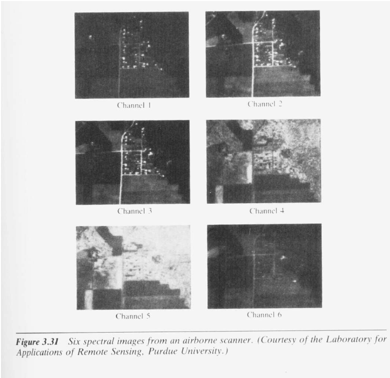


Figure 1: Input images

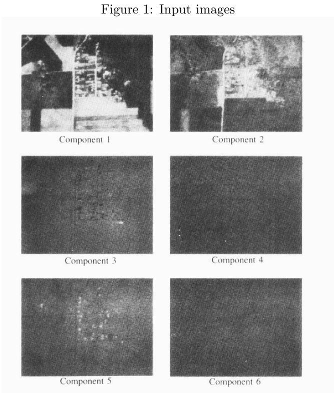


Figure 2: Principal Component Images