ECE 634: Digital Video Systems
Transform coding: 2/16/17

Professor Amy Reibman
MSEE 356
reibman@purdue.edu

http://engineering.purdue.edu/~reibman/ece634/index.html

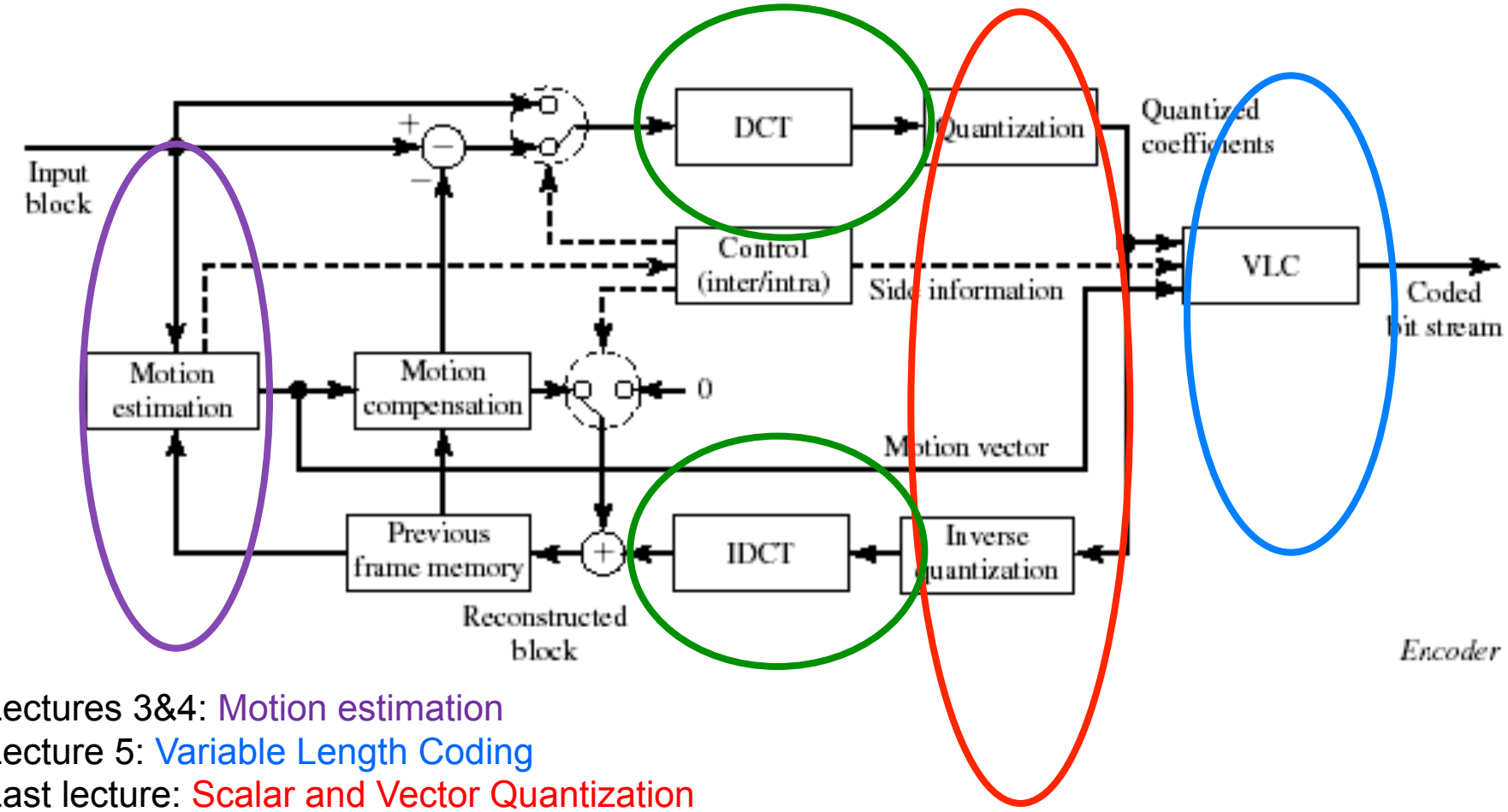
Background reading (Transform coding)

- R. J. Clarke, Transform Coding of Images, Academic Press, London, 1985, chapters 3&4
- W. K. Pratt, Digital Image Processing (2nd ed), Wiley, 1991, chapter 8
- Wang, Ostermann, Zhang, Video Processing and Communications, Prentice Hall 2002, Section 9.1

Transform Coding

- What, how, which, why, how good?

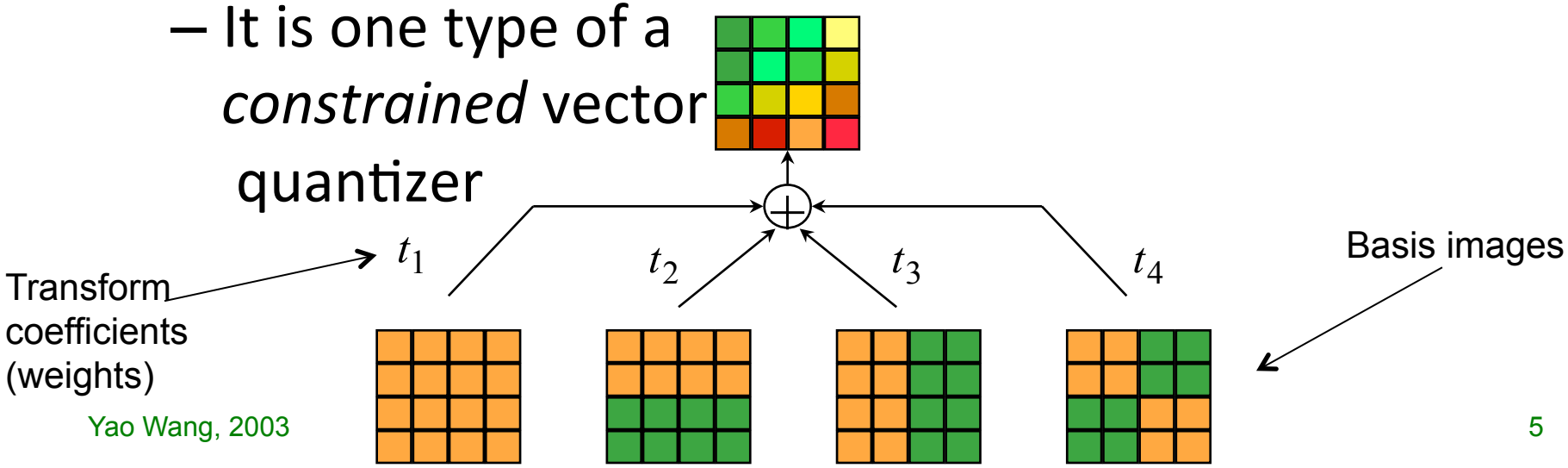
Encoder Block Diagram of a Typical Block-Based Video Coder (Assuming No Intra Prediction)



Lectures 3&4: Motion estimation
Lecture 5: Variable Length Coding
Last lecture: Scalar and Vector Quantization
This lecture: DCT, wavelet and predictive coding

Transform Coding

- Motivation:
  - Represent a vector (e.g. a block of image samples) as the *superposition* of some typical vectors (block patterns)
  - Quantize and code the coefficients
  - It is one type of a *constrained* vector quantizer



General Linear Transform

- Basis vectors (or blocks):

$$[\mathbf{U}] = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$$

- Inverse transform represents a vector or block as the superposition of basis vectors or blocks

$$\text{inverse transform:} \quad \mathbf{s} = \sum_{k \in \mathcal{N}} t_k \mathbf{u}_k = [\mathbf{U}]\mathbf{t}$$

- Forward transform determines the contribution (weight) of each basis vector

$$\text{forward transform:} \quad \mathbf{t} = [\mathbf{U}]^{-1}\mathbf{s} = [\mathbf{V}]\mathbf{s}$$

Example

$$U = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \qquad V = U^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

$$s = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 1 \end{bmatrix} \qquad t = Vs = \begin{bmatrix} 9 \\ 3 \\ 3 \\ 5 \end{bmatrix}$$

$$s = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 1 \end{bmatrix} = Ut = \frac{9}{4}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + \frac{5}{4}\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

Unitary Transform

- Unitary (orthonormal) basis:
  - Basis vectors are orthogonal to each other and each has length 1

$$\langle \mathbf{u}_k, \mathbf{u}_l \rangle = \sum_{n \in \mathcal{N}} u^*_{k;n} u_{l;n} = \delta_{k,l} = \begin{cases} 1 & \text{if } k = l, \\ 0 & \text{if } k \neq l, \end{cases}$$

$$[\mathbf{U}]^H [\mathbf{U}] = [\mathbf{U}][\mathbf{U}]^H = [\mathbf{I}]_N$$

- Transform coefficient associated with a basis vector is simply the projection of the input vector onto the basis vector
- Can also be thought of as an approximation

$$\text{forward transform:} \quad t_k = \langle \mathbf{u}_k, \mathbf{s} \rangle \quad \text{or} \quad \mathbf{t} = [\mathbf{U}]^H \mathbf{s} = [\mathbf{V}]\mathbf{s}$$

$$\text{inverse transform:} \quad \mathbf{s} = \sum_{k \in \mathcal{N}} t_k \mathbf{u}_k = [\mathbf{U}]\mathbf{t} = [\mathbf{V}]^H \mathbf{t}.$$

Example

$$U = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \qquad V = U^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

$$s = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 1 \end{bmatrix} \qquad t = Vs = \frac{1}{2}\begin{bmatrix} 9 \\ 3 \\ 3 \\ 5 \end{bmatrix}$$

$$s = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 1 \end{bmatrix} = Ut = \frac{9}{4}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + \frac{5}{4}\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

Example: Approximations

$$U = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \qquad V = U^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

$$s = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 1 \end{bmatrix} \quad t = Vs = \frac{1}{2}\begin{bmatrix} 9 \\ 3 \\ 3 \\ 5 \end{bmatrix} \qquad s = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 1 \end{bmatrix} = Ut = \frac{9}{4}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + \frac{5}{4}\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

$$\hat{s} = \frac{9}{4}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.75 \\ 2.25 \\ 0.75 \\ 2.25 \end{bmatrix}$$

Separable transforms for images

- Apply a 1-D transform on the rows, and then a 1-D transform on the columns (and continue, if signal has more than 2 dimensions..)
- Basis images: $u_k g_l^T$

Transform design

- What are desirable properties of a transform for image and video?
  - Nearly decorrelating – improves efficiency of scalar quantizer
  - High energy compaction – a few large coefficients to send
  - Easy to compute (few operations)
  - Separable – compute 1-D transform first on rows, then on columns
- What size transform should we use?
  - Entire image? Small?
  - 2-D (on an image) or 3-D (incorporating time also)?

Karhunen Loève Transform (KLT)

- Optimal transform
- Requires statistics of the input source
  - Known covariance function

- Coefficients are completely uncorrelated
- The best energy compaction
  - Sort coefficients from largest to smallest expected squared magnitude; then the sum of the energies of the first M coefficients is as large as possible
- No computationally efficient algorithm

- We'll derive it later

Other Transform Bases

- Optimal transform
  - Karhunen Loève Transform (KLT): Depends on the signal statistics
- Suboptimal transforms – **many** available!
  - Discrete Fourier Transform (DFT): complex values; discontinuities
  - Discrete Cosine transform (DCT): nearly as good as KLT for common image signals
  - Hadamard and Haar: basis functions contain only +1,0,-1

DCT vs. DFT

- DFT is complex; DCT is real
- Consider continuous signal, sampled over window of length N
- When applying DFT, transform-domain is not the isolated segment. Instead, that sampled signal gets repeated. If left side is not same value as right side, severe discontinuities
- DCT equivalent to “folding” signal and applying DFT

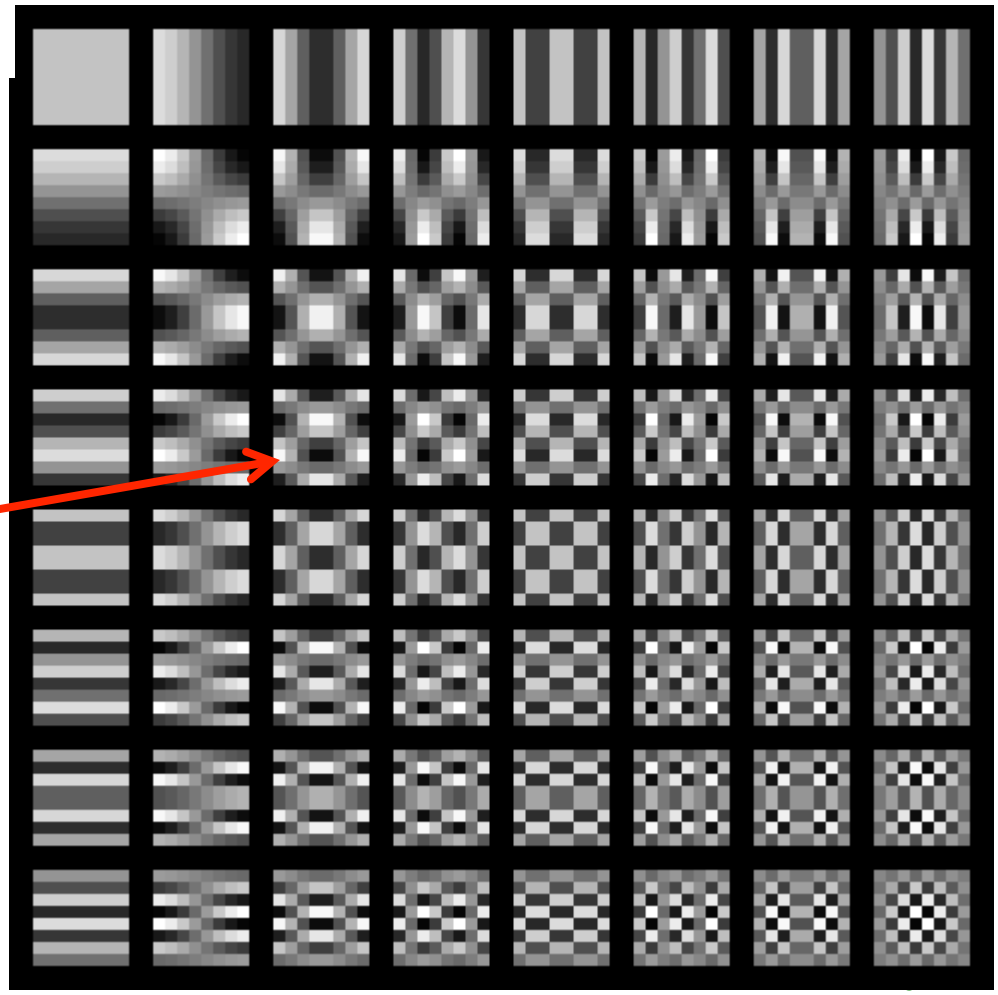
Discrete Cosine Transform: 8*8 Basis Images

inverse transform: $\mathbf{s} = \sum t_k \mathbf{u}_k = [\mathbf{U}]\mathbf{t}$

forward transform: $\mathbf{t} = [\mathbf{U}]^{-1}\mathbf{s} = [\mathbf{V}]\mathbf{s}$

Example in Matlab:

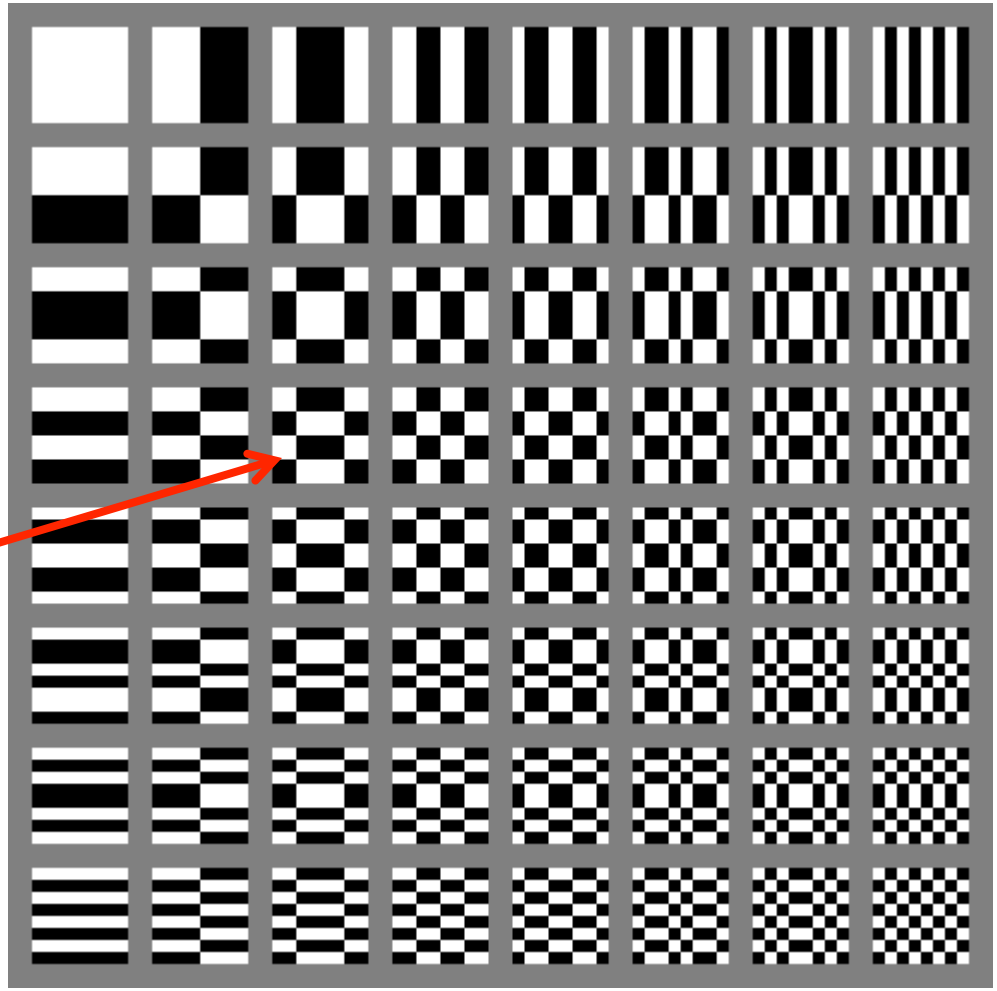
```
D=dctmtx(8);
X=zeros(8);
X(4,3)=1;
Basis=D' *X*D;
```



Hadamard Transform: 8*8 Basis images

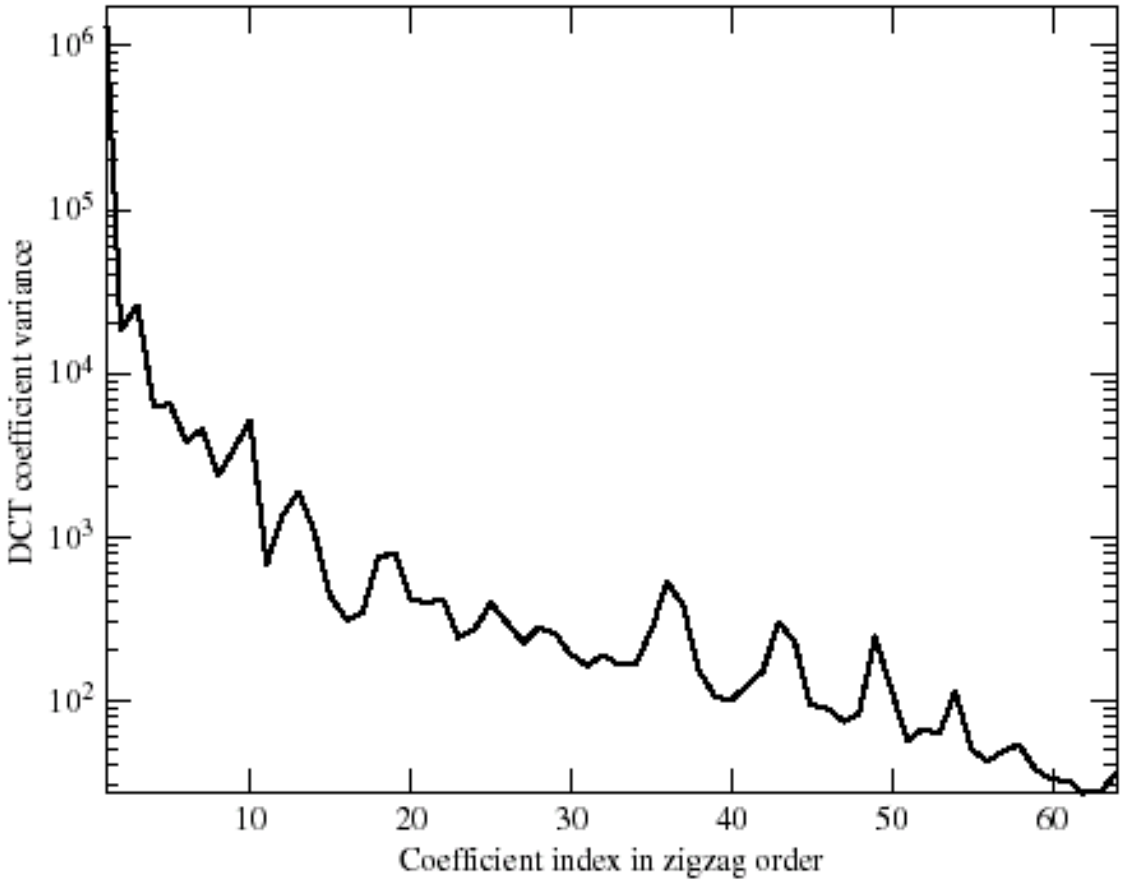
Example in Matlab:

```
D=hadamard(8);
reindex=[1,8,4,5,2,7,3,6];
D(reindex,:)=D;
X=zeros(8);
X(4,3)=1;
Basis=D' *X*D;
```



Energy/Variance Distribution of DCT Coefficients in Typical Images

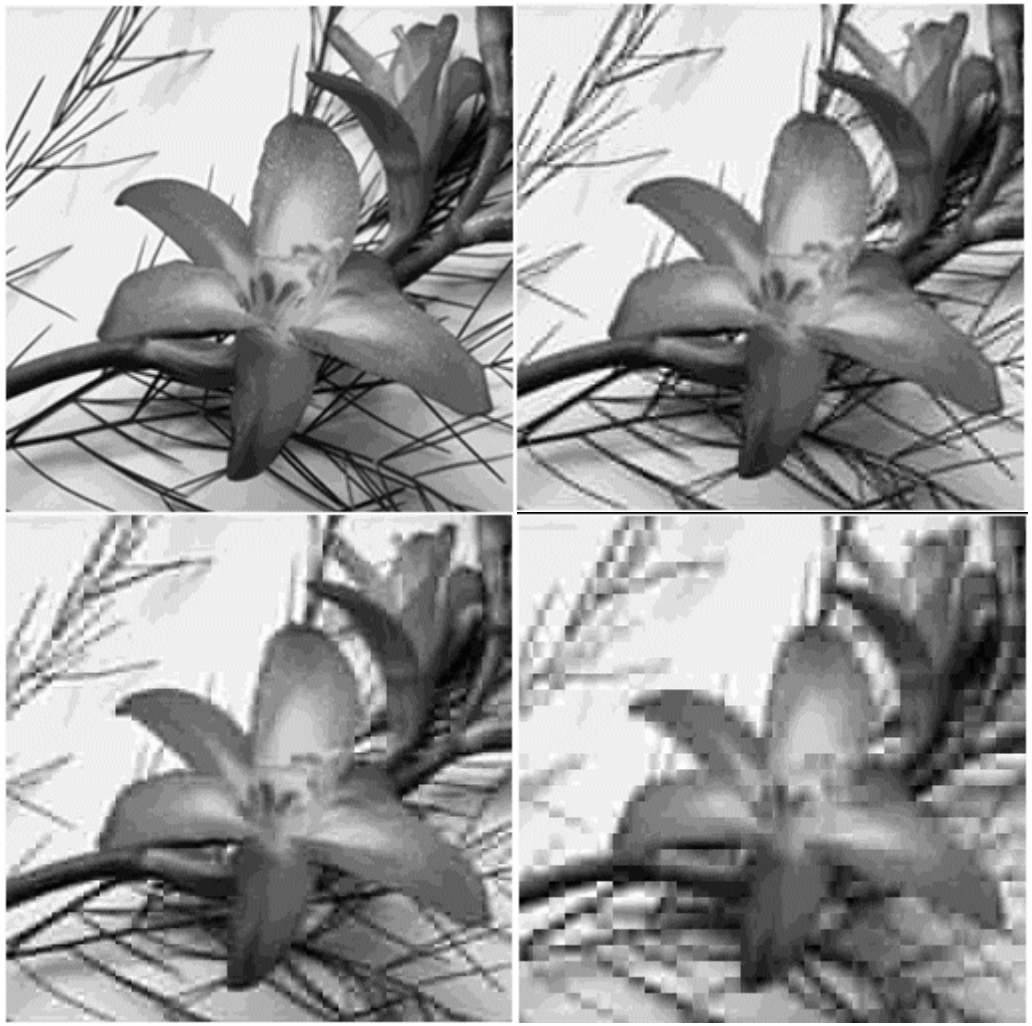
On a log scale



Zigzag order; low horizontal and low vertical are earlier

Images Approximated by Different Number of DCT Coefficients

Original



With 16/64 Coefficients

With 8/64 Coefficients

With 4/64 Coefficients

Bit allocation for transform coding (outline)

- Assumptions:
  - 1-D transform; Scalar quantization with MMSE quantizer; Fixed-length coding
- Goal: what is the average distortion and bit-rate per sample

- Computing distortion
- Optimal bit allocation among the N transform coefficients
  - High rate approximation
- Performance improvement relative to pixel-based coding
- Examples

- Optimal transform design

Distortion (MSE) in Transform Coding: A statistical analysis

- Distortion in sample (image) domain

$$D_s = \frac{1}{N} E\{\|\mathcal{S} - \hat{\mathcal{S}}\|^2\} = \frac{1}{N} \sum_{n \in \mathcal{N}} D_{s,n} \qquad D_{s,n} = E\{(\mathcal{S}_n - \hat{\mathcal{S}}_n)^2\}.$$

- Distortion in coefficient (transform) domain

$$D_t = \frac{1}{N} E\{\|\mathcal{T} - \hat{\mathcal{T}}\|^2\} = \frac{1}{N} \sum_{k \in \mathcal{N}} D_{t,k} \qquad D_{t,k} = E\{(\mathcal{T}_k - \hat{\mathcal{T}}_k)^2\}.$$

- With a unitary transform, the two distortions are equal

$$D_s = \frac{1}{N} E\{\|\mathcal{S} - \hat{\mathcal{S}}\|^2\} = \frac{1}{N} E\{\|[\mathbf{V}]^H (\mathcal{T} - \hat{\mathcal{T}})\|^2\}$$

$$= \frac{1}{N} E\{(\mathcal{T} - \hat{\mathcal{T}})^H [\mathbf{V}][\mathbf{V}]^H (\mathcal{T} - \hat{\mathcal{T}})\} = \frac{1}{N} E\{\|\mathcal{T} - \hat{\mathcal{T}}\|^2\} = D_t.$$

Modeling Distortion Due to Coefficient Quantization

- How much distortion is introduced by quantizing the k-th transform coefficient?
- Use a high-resolution approximation of scalar quantization
  - MMSE quantizer; each coefficient is quantized with high rate; pdf in each quantization bin is nearly flat

One coefficient

$$D_{t,k}(R_k) = \epsilon_{t,k}^2 \sigma_{t,k}^2 2^{-2R_k}$$

Average over all coefficients

$$D_{\mathrm{TC}} = D_s = D_t = \frac{1}{N} \sum_{k \in \mathcal{N}} \epsilon_{t,k}^2 \sigma_{t,k}^2 2^{-2R_k}.$$

$\epsilon_{t,k}^2$ Depends on the pdf of the k-th coefficient.

Optimal Bit Allocation Among Coefficients

- How many bits to use for each coefficient?
  - Can be formulated as an constrained optimization problem:

Minimize: $$D_{\mathrm{TC}} = D_s = D_t = \frac{1}{N}\sum_{k\in\mathcal{N}} \epsilon_{t,k}^2 \sigma_{t,k}^2 2^{-2R_k}.$$

Subject to: $$\sum_{k\in\mathcal{N}} R_k = RN$$

  - The constrained problem can be converted to unconstrained one using the Lagrange multiplier method

Minimize: $$J(R_k, \forall k \in \mathcal{N}) = \sum_{k\in\mathcal{N}} \epsilon_{t,k}^2 \sigma_{t,k}^2 2^{-2R_k} + \lambda\left(\sum_{k\in\mathcal{N}} R_k - RN\right)$$

Derivation and Result

If we let $(\partial J / \partial R_k) = 0$, we obtain

$$\frac{\partial D_{t,k}}{\partial R_k} = -2 \ln 2 D_{t,k} = -(2 \ln 2)\epsilon_{t,k}^2 \sigma_{t,k}^2 2^{-2R_k} = -\lambda, \quad \forall k \in \mathcal{N}$$

Multiply to obtain:

$$\lambda^N = (2 \ln 2)^N \left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2 \right) 2^{-2 \sum_k R_k} = (2 \ln 2)^N \left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2 \right) 2^{-2NR}$$

$$\lambda = (2 \ln 2) \left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2 \right)^{1/N} 2^{-2R}.$$

Substitute into first equation:

$$R_k = R + \frac{1}{2} \log_2 \frac{\epsilon_{t,k}^2 \sigma_{t,k}^2}{\left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2 \right)^{1/N}}.$$

Result: all distortions are equal!

$$D_{\mathrm{TC}} = D_t = D_{t,k} = \left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2 \right)^{1/N} 2^{-2R}.$$

Implication of Optimal Bit Allocation

- Bit rate for a coefficient proportional to its variance (energy)

$$R_k = R + \frac{1}{2}\log_2 \frac{\epsilon_{t,k}^2 \sigma_{t,k}^2}{\left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2\right)^{1/N}}.$$

Geometric mean

- Distortion is equalized among all coefficients and depends on the geometric mean of the coefficient variances

$$D_{\mathrm{TC}} = D_t = D_{t,k} = \left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2\right)^{1/N} 2^{-2R}.$$

Transform Coding Gain Over PCM

- Distortion for PCM if each sample is quantized to R bit:

$$D_{\text{PCM}} = D_{s,n} = \epsilon_s^2 \sigma_s^2 2^{-2R}$$

- Transform Coding Gain over PCM: $G_{\text{TC}} = \dfrac{D_{\text{PCM}}}{D_{\text{TC}}}$.

Arithmetic mean

$$G_{\text{TC}} = \frac{\epsilon_s^2 \sigma_s^2}{\left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2\right)^{1/N}} = \frac{\epsilon_s^2}{\left(\prod_k \epsilon_{t,k}^2\right)^{1/N}} \frac{\frac{1}{N}\sum \sigma_{t,k}^2}{\left(\prod_k \sigma_{t,k}^2\right)^{1/N}},$$

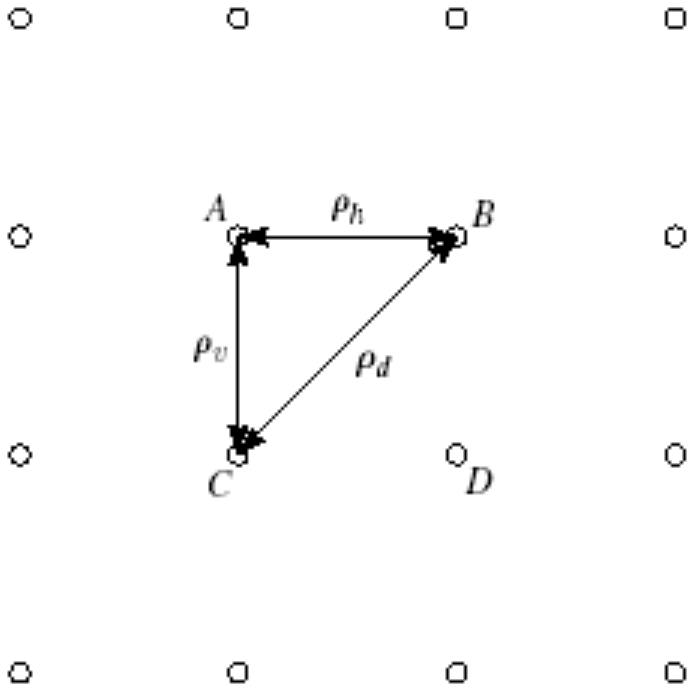
- For Gaussian source
  - each sample is Gaussian, so that coefficients are also Gaussian, all the same

$\epsilon_{t,k}^2$

$$G_{\text{TC,Gaussian}} = \frac{\sigma_s^2}{\left(\prod_k \sigma_{t,k}^2\right)^{1/N}} = \frac{\frac{1}{N}\sum \sigma_{t,k}^2}{\left(\prod_k \sigma_{t,k}^2\right)^{1/N}}.$$

Example

- Determine the optimal bit allocation and corresponding TC gain for coding 2x2 image block using 2x2 DCT. Assuming the image is a Gaussian process with inter-sample correlation as shown below.



$$\rho_h = \rho_v = \rho$$

$$\rho_d = \rho^2$$

Covariance between coefficients in Transform Coding

- $Cov(S) = E[(S-\mu_S)\,(S-\mu_S)'] = E(SS^T)-\mu_S\mu_S'$
- $Cov(T) = E[(T-\mu_T)\,(T-\mu_T)'] = E(TT')-\mu_y\mu_y'$
- Assume zero mean
- $T=[V]S$
- $Cov(T)=Cov([V]S)=E([V]\,SS^T[V]')$
  $= [V][Cov(S)][V]'$

Example Continued (Convert 2x2 into 4x1)

- Covariance matrix (assume zero mean)

$$[\mathbf{C}]_s = E\left\{ \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} [A \quad B \quad C \quad D] \right\} = \begin{bmatrix} C_{AA} & C_{AB} & C_{AC} & C_{AD} \\ C_{BA} & C_{BB} & C_{BC} & C_{BD} \\ C_{CA} & C_{CB} & C_{CC} & C_{CD} \\ C_{DA} & C_{DB} & C_{DC} & C_{DD} \end{bmatrix}$$

$$= \sigma_s^2 \begin{bmatrix} 1 & \rho_h & \rho_v & \rho_d \\ \rho_h & 1 & \rho_d & \rho_v \\ \rho_v & \rho_d & 1 & \rho_h \\ \rho_d & \rho_v & \rho_h & 1 \end{bmatrix}.$$

- DCT basis images

$$\frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, \quad \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \quad \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

- Equivalent transform matrix

$$[\mathbf{U}] = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Example Continued

$$[\mathbf{C}]_t = [\mathbf{V}][\mathbf{C}]_s[\mathbf{V}]^H \longrightarrow$$

$$\sigma_{t,k}^2 = \{(1+\rho)^2, (1-\rho^2), (1-\rho^2), (1-\rho)^2\}\sigma_s^2$$

$$\sigma_t^2 = \left(\prod_k \sigma_{t,k}^2\right)^{1/4} = (1-\rho^2)\sigma_s^2 : \qquad G_{\mathrm{TC}} = \frac{\sigma_s^2}{\sigma_t^2} = \frac{1}{1-\rho^2}$$

$$R_k = R + \frac{1}{2}\log_2 \frac{\epsilon_{t,k}^2 \sigma_{t,k}^2}{\left(\prod_k \epsilon_{t,k}^2 \sigma_{t,k}^2\right)^{1/N}}. \longrightarrow$$

$$R_k = \{4.64, 2, 2, -0.64\}. \qquad \text{(for R=2)}$$

Optimal transform

- Approach 1: Minimize the MSE introduced if we omit a coefficient
  - Start with smallest coefficient, then next, etc
- Approach 2: Minimize the correlation between different transform coefficients
- Approach 3: Maximize the transform coding gain

$$G_{\mathrm{TC}} = \frac{D_{\mathrm{PCM}}}{D_{\mathrm{TC}}}.$$

- All lead to same answer (assuming Gaussian source): Karhunen Loeve Transform (KLT)

Optimal Transform: Design

- If source is Gaussian, the optimal transform is the Karhunen-Loeve transform, which depends on the covariance matrix between samples
  - Basis vectors are the eigenvectors of the covariance matrix, the coefficient variances are the eigenvalues

$$[\mathbf{C}]_s \phi_k = \lambda_k \phi_k, \quad \text{with} \quad \langle \phi_k, \phi_l \rangle = \delta_{k,l}. \qquad \sigma_k^2 = \lambda_k.$$

  - The determinant is the product of the coefficient variances

$$\prod_{k \in \mathcal{N}} \sigma_{t,k}^2 = \det[\mathbf{C}]_t = \det[\mathbf{C}]_s.$$

  - The distortion for a given rate R: $D_{\mathrm{TC}} = \epsilon_{\mathrm{Gaussian}}^2 (\det[\mathbf{C}]_s)^{1/N} 2^{-2R}$ exceeds the RD bound by a constant factor

  - The transform coding gain for the KLT:

$$G_{\mathrm{TC,KLT}} = \frac{\epsilon_s^2}{(\prod_k \epsilon_k^2)^{1/N}} \frac{\sigma_s^2}{(\prod_k \lambda_k)^{1/N}} = \frac{\epsilon_s^2}{(\prod_k \epsilon_k^2)^{1/N}} \frac{\sigma_s^2}{(\det[\mathbf{C}]_s)^{1/N}}$$

What would be necessary to implement in a real system?

- Estimate the data covariance matrix
  - For both row and column!
- Compute eigenvectors to generate basis vectors
- Associate this KLT with the image
  - Either when stored, or transmitted

- But the relative gains for typical images is small

Example gains: 1st order Markov source, rho=0.91

- Define efficiency to be the amount of source energy contained in all coefficients up to and including coefficient i

	1	2	3	4	5	6	7	8
KLT	79.5	91.1	94.8	96.7	97.9	98.7	99.4	100
DCT	79.3	90.9	94.8	96.7	97.9	98.7	99.4	100
Hadamard	79.3	89.3	92.7	95.5	96.7	97.9	99	100
DST	73.6	84.3	92.5	95.0	97.4	98.4	99.4	100

From R. J. Clarke, *Transform coding of images*, Prentice Hall, 1985. Chapter 3.

Properties of KLT

- The optimal transform for Gaussian sources
- Nearly optimal transform for non-Gaussian sources

- Minimal approximation error for K<N coefficients among all unitary transforms
- KLT has highest energy compaction
- Coefficients are uncorrelated

- Requires a stationary source with known covariance matrix – most sources vary spatially and temporally
- No fast algorithms – and not signal **in**dependent

Example

- Determine the KLT for the 2x2 image block in the previous example

$$[\mathbf{C}]_s = \sigma_s^2 \begin{bmatrix} 1 & \rho_h & \rho_v & \rho_d \\ \rho_h & 1 & \rho_d & \rho_v \\ \rho_v & \rho_d & 1 & \rho_h \\ \rho_d & \rho_v & \rho_h & 1 \end{bmatrix}$$

Determine the eigenvalues by solving: $\det([\mathbf{C}]_s - \lambda[\mathbf{I}]) = 0$

$$\lambda_k = \{(1+\rho)^2, (1-\rho^2), (1-\rho^2), (1-\rho)^2\}\sigma_s^2.$$

(same as the coefficient variances with DCT)

Determine the eigenvectors by solving $([\mathbf{C}]_s - \lambda[\mathbf{I}])\phi_k = 0$

Resulting transform is the DCT
(because this is a 2*2 example!!)

Example: JPEG Image Coder

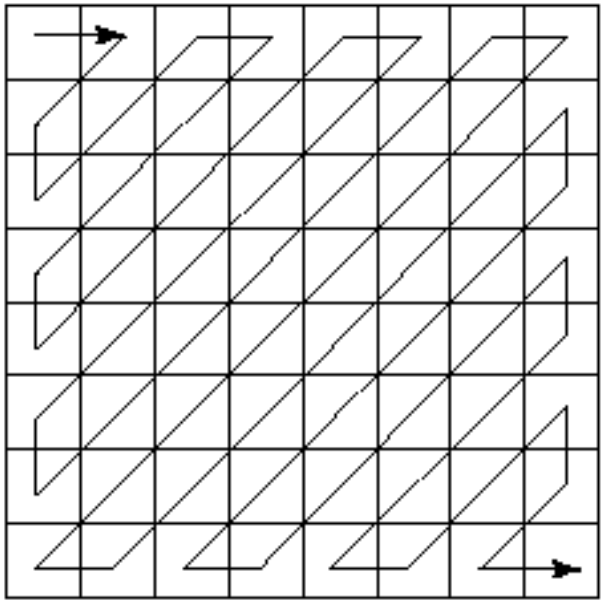
- Joint Photographic Expert Group
- Uses 8x8 DCT
- Each coefficient is quantized using a uniform quantizer
- Step sizes vary based on coefficient variances and their visual importance
- Quantized coefficients are converted into binary bitstreams using runlength coding plus Huffman coding

JPEG: a bit more detail

Perceptual based quantization matrix:

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

Zig-zag ordering of DCT coefficients:



Runlength coding example:

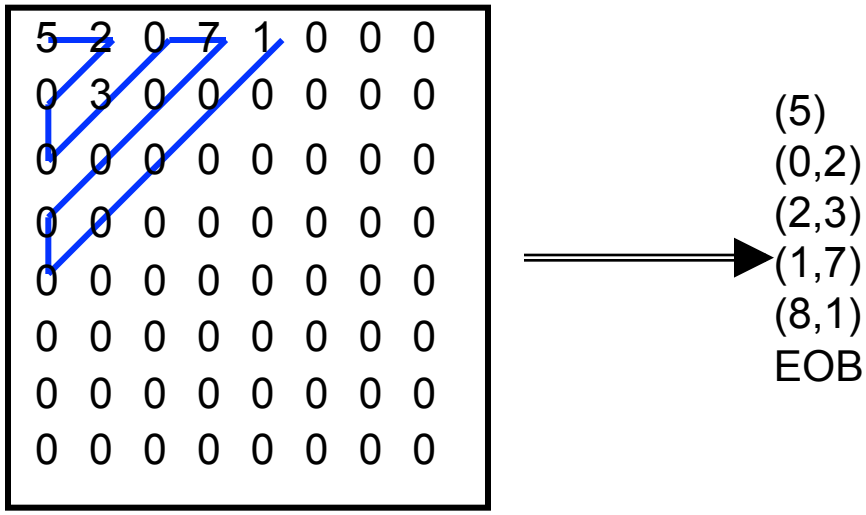
DCT coefficients: [5 0 0 2 3 0 0 4 0 0 0 0 0 0 1 0 0 0 0 0 … 0]

Coding symbols: 5, (2,2), (0,3), (2,4), (6,1), EOB

Run-length coding

5	2	0	7	1	0	0	0
0	3	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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Run-length coding



- Map 2-D quantized DCT coefficients into 1-D set of pairs
  - Uses "zigzag" scan
  - Number of consecutive zeros followed by coefficient value
- Works well, because there are typically lots of zeros