Lecture 3: Introduction to Audio, Video & Image Coding Techniques (II) – Coding Techniques

Dr. Jian Zhang

NICTA & CSE UNSW
COMP9519 Multimedia Systems
S2 2007
jzhang@cse.unsw.edu.au
3. Introduction to audio, video & image coding techniques (II)

- 3.1 Transform Coding
- 3.2 Still Image Coding Standard (JPEG)
- 3.3 Subband Coding
- 3.4 Temporal Redundancy and Prediction for Video Coding
- 3.5 Motion Estimation & Compensation Algorithms
3.1.1 Basic Transform coding

- **Block Based Transform Coding**
  - Transform coding is performed by taking an image and breaking it down into sub-image (block) of size nxn. The transform is then applied to each sub-image (block) and the resulting transform coefficients are quantized and entropy coded.

![Diagram of Transform Coding Process]
3.1.1 Basic Transform coding

- Consider the following block of data

\[
[F] = \begin{bmatrix}
5 & 6 & 8 & 10 \\
6 & 6 & 5 & 7 \\
4 & 5 & 3 & 6 \\
8 & 7 & 5 & 5 \\
\end{bmatrix}
\]

- A 2-Dimensional transform can be carried out in a separable way (i.e. first down the columns and then along the rows).
3.1.1 Basic Transform coding

- The 1-Dimensional transform is calculated according to $[C'] = [T][F]$;
- Where $[T]$ is the transform matrix

$$[C'] = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 5 & 6 & 8 & 10 \\ 6 & 6 & 5 & 7 \\ 4 & 5 & 3 & 6 \\ 8 & 7 & 5 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} 11.5 & 12.0 & 10.5 & 14.0 \\ -0.5 & 0.0 & 2.5 & 3.0 \\ 1.5 & 1.0 & 2.5 & 1.0 \\ -2.5 & -1.0 & 0.5 & 2.0 \end{bmatrix}$$

4 Coefficients contain 94% of total energy
3.1.1 Basic Transform coding

- A 2-dimensional transform can be extended according to: $[C] = [C'][T] = [T][F][T]^T$

\[
[C] = \frac{1}{2} \begin{bmatrix}
11.5 & 12.0 & 10.5 & 14.0 \\
-0.5 & 0.0 & 2.5 & 3.0 \\
1.5 & 1.0 & 2.5 & 1.0 \\
-2.5 & -1.0 & 0.5 & 2.0
\end{bmatrix}
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
24.0 & -0.5 & 1.5 & -2.0 \\
2.5 & -3.0 & 0.0 & -0.5 \\
3.0 & -0.5 & -0.5 & 1.0 \\
-0.5 & -3.0 & 0.0 & -1.5
\end{bmatrix}
\]

93% of energy now in one term that is in position (0,0)
3.1.1 Basic Transform coding

- Discrete Cosine Transform
  - For a 2-D input block $U$, the transform coefficients can be found as $Y = CUC^T$
  - The inverse transform can be found as $U = C^TYC$
  - The $N\times N$ discrete cosine transform matrix $C = c(k,n)$ is defined as:

$$c(k,n) = \begin{cases} 
\frac{1}{\sqrt{N}} & \text{for } k = 0 \text{ and } 0 \leq n \leq N - 1, \\
\sqrt{\frac{2}{N}} \cos \frac{\pi(2n+1)k}{2N} & \text{for } 1 \leq k \leq N - 1 \text{ and } 0 \leq n \leq N - 1.
\end{cases}$$
### 3.1.1 Basic Transform Coding

- **8x8 2-D DCT**

Original 8x8 pixel image

\[
\begin{bmatrix}
543.87 & 22.53 & 50.40 & -21.20 & -0.62 & -4.91 & -22.37 & -12.54 \\
-76.02 & 8.84 & -40.14 & 30.71 & 4.25 & 2.03 & 15.01 & 6.62 \\
83.64 & 32.46 & 5.76 & -2.03 & -2.68 & -34.57 & 20.62 & -2.62 \\
32.91 & 23.27 & -10.25 & 34.64 & -4.21 & 29.63 & -39.55 & -11.59 \\
15.62 & -17.05 & 30.56 & -27.60 & 8.62 & -19.00 & 21.84 & 3.71 \\
12.27 & 20.77 & -31.24 & 36.16 & -2.39 & -17.86 & 2.94 & -10.00 \\
11.49 & 5.52 & 11.37 & -12.96 & -0.92 & 14.98 & -3.26 & 9.73 \\
7.64 & 1.15 & -5.81 & 9.25 & -3.84 & -15.97 & 2.03 & -1.61 \\
\end{bmatrix}
\]

8x8 2-D DCT coefficients
3.1.1 Basic Transform coding

• The distribution of 2-D DCT Coefficients

Ref: H. Wu
3.1.2 Optimal Transform

- An Optimal transform should achieve:
  - Completely de-correlate the data
  - Maximize the amount of energy packed into the lowest order coefficients
  - Given a 1-D general data column vector \( X \) with the mean vector, the covariance matrix (2D) is defined as:

\[
COV(X) = E[(X - \bar{X})(X - \bar{X})^T]
\]

- Where \( E(\bullet) \) is the expectation operator. Given an orthogonal transform represented by the matrix \([T]\) and the corresponding transform coeff. Vector by \([Y]=[T][X]\), we have:

\[
COV(Y) = E[(TX - \bar{X})(XT - \bar{X})^T] = TCOV(X)T^T
\]
3.1.3 Transform Efficiency

- De-correlation and energy packing effects (Eg. DCT)
- The transform domain covariance matrix $\text{COV}(Y_{\text{DCT}})$ is generated by using the 1-D 4-point discrete cosine transform (DCT)

$$COV(Y_{\text{DCT}}) = T_{\text{DCT}} \cdot COV(X) \cdot T_{\text{DCT}}^T$$

Ref: H. Wu
3.1.3 Transform Efficiency

- From the previous slides, there are two distinctive effects by the DCT:
  - All off-diagonal elements of the transform domain covariance matrix are significantly reduced, indicating that the transform coefficients are effectively de-correlated as a result of the transform;
  - The majority of the data energy (as represented by the variance) has been transferred to a few low-order coefficients after the transform.
- The most desirable transform domain covariance matrix will have a diagonal form (with values corresponding to the eigenvalues of the COV matrix) which achieves 100% decorrelation efficiency for transform coefficients.

\[
[T][COV(X)][T]^T = \Lambda
\]
3.1.2 Optimal Transform

- The optimal Transform exists – eg. Karhunen-Loeve Transform (KLT), but has some limitations
  - The KLT has the maximum energy packing capability with completely de-correlation for signal in transform domain
  - KLT involves the estimation of the COV, its diagonalisation and the contraction of the basis vectors
  - The basis vectors (transform matrix) are not fixed, and must be generated for images with different correlation characteristics.
  - The transform remains of more theoretical than practical interest as far as image coding is concerned.
  - Due to the lack of a fast implementation, it is not used practically.
3.1.4 Discrete Cosine Transform

- A transform is calculated by finding the correlation between the input sub-image and a series of basis vectors.
- The transform has separable property (2D transform = 1D transform along rows and then down the columns).
- The DCT has the advantage for video coding of good redundancy reduction with performance close to the KLT [K.R.Rao]

\[
c(k, n) = \begin{cases} 
\frac{1}{\sqrt{N}} & \text{for } k = 0 \text{ and } 0 \leq n \leq N - 1, \\
\sqrt{\frac{2}{N}} \cos \frac{\pi (2n + 1)k}{2N} & \text{for } 1 \leq k \leq N - 1 \text{ and } 0 \leq n \leq N - 1.
\end{cases}
\]
3.1.4 Discrete Cosine Transform

- In the case of the DCT, the basis vectors (functions) are a series of harmonically related cosine functions.
- Basis Vectors for N=8 DCT

<table>
<thead>
<tr>
<th>$c(k,n)$</th>
<th>$n=0$</th>
<th>$n=1$</th>
<th>$n=2$</th>
<th>$n=3$</th>
<th>$n=4$</th>
<th>$n=5$</th>
<th>$n=6$</th>
<th>$n=7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k=0$</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>$k=1$</td>
<td>0.49</td>
<td>0.42</td>
<td>0.28</td>
<td>0.10</td>
<td>−0.10</td>
<td>−0.28</td>
<td>−0.42</td>
<td>−0.49</td>
</tr>
<tr>
<td>$k=2$</td>
<td>0.46</td>
<td>0.19</td>
<td>−0.19</td>
<td>−0.46</td>
<td>−0.46</td>
<td>−0.19</td>
<td>0.19</td>
<td>0.46</td>
</tr>
<tr>
<td>$k=3$</td>
<td>0.42</td>
<td>−0.10</td>
<td>−0.49</td>
<td>−0.28</td>
<td>0.28</td>
<td>0.49</td>
<td>0.10</td>
<td>−0.42</td>
</tr>
<tr>
<td>$k=4$</td>
<td>0.35</td>
<td>−0.35</td>
<td>−0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>−0.35</td>
<td>−0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>$k=5$</td>
<td>0.28</td>
<td>−0.49</td>
<td>0.10</td>
<td>0.42</td>
<td>−0.42</td>
<td>−0.10</td>
<td>0.49</td>
<td>−0.28</td>
</tr>
<tr>
<td>$k=6$</td>
<td>0.19</td>
<td>−0.46</td>
<td>0.46</td>
<td>−0.19</td>
<td>−0.19</td>
<td>0.46</td>
<td>−0.46</td>
<td>0.19</td>
</tr>
<tr>
<td>$k=7$</td>
<td>0.10</td>
<td>−0.28</td>
<td>0.42</td>
<td>−0.49</td>
<td>0.49</td>
<td>−0.42</td>
<td>0.28</td>
<td>−0.10</td>
</tr>
</tbody>
</table>
3.2 Still Image Coding Standard (JPEG)

- Joint Photographic Experts Group.
- Formed in 1986 by ISO and ITU-T (CCITT) and Become an international standard (IS) in 1991.
- General purpose, applicable to almost all continuous-tone still image application.
- Lossy (DCT based) vs Lossless (2D DPCM).
- 8/12 bits sample precision up to 65535 lines and 65535 pixels per line.
- Huffman vs arithmetic coding.
- Sequential, progressive & hierarchical modes.
- Compression ratio of 10:1 to 50:1.
- Colour-space independent: up to 255 colour components, components can be sub-sampled and interleaved, YUV is better than RGB.
3.2 Still Image Coding Standard (JPEG)

- 3.2.1 JPEG DCT-Based Encoding
- 3.2.2 Coding of DCT Coeff.
- 3.2.3 JPEG DCT-based coding
3.2.1 JPEG DCT-Based Encoding

Ref: H. Wu
3.2.1 JPEG DCT-Based Encoding

- Recommended JPEG quantisation matrix $N_{i,k} =$

  For luminance:
  \[
  \begin{bmatrix}
  16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
  12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
  14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
  14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
  18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
  24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
  49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
  72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \\
  \end{bmatrix}
  \]

  For chrominance:
  \[
  \begin{bmatrix}
  17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\
  18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\
  24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\
  \end{bmatrix}
  \]

- Quantisation of coefficients $Y_{j,k}$ with a quality factor $Q_s$
  \[
  \hat{Y}_{j,k} = Q[Y_{j,k}] = \text{round}\left( \frac{Y_{j,k}}{N_{j,k} \times Q_s} \right)
  \]

- High-frequency coefficients and colour components can be quantised more.
### 3.2.2 Coding of DCT Coefficients (DC)

- DC coefficient is coded differentially as (size, amplitude). There are 12 categories of size.

\[
\Delta DC_i = DC_i - DC_{i-1}
\]

<table>
<thead>
<tr>
<th>Size</th>
<th>[Coeff]</th>
<th>[Code]</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>2+0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>010</td>
<td>3+1</td>
</tr>
<tr>
<td>2..3</td>
<td>2</td>
<td>011</td>
<td>3+2</td>
</tr>
<tr>
<td>4..7</td>
<td>3</td>
<td>100</td>
<td>3+3</td>
</tr>
<tr>
<td>8..15</td>
<td>4</td>
<td>101</td>
<td>3+4</td>
</tr>
<tr>
<td>16..31</td>
<td>5</td>
<td>110</td>
<td>3+5</td>
</tr>
<tr>
<td>32..63</td>
<td>6</td>
<td>1110</td>
<td>4+6</td>
</tr>
<tr>
<td>64..127</td>
<td>7</td>
<td>11110</td>
<td>5+7</td>
</tr>
<tr>
<td>128..255</td>
<td>8</td>
<td>111110</td>
<td>6+8</td>
</tr>
<tr>
<td>256..511</td>
<td>9</td>
<td>1111110</td>
<td>7+9</td>
</tr>
<tr>
<td>512..1023</td>
<td>10</td>
<td>11111110</td>
<td>8+10</td>
</tr>
<tr>
<td>1024..2047</td>
<td>11</td>
<td>111111110</td>
<td>9+11</td>
</tr>
</tbody>
</table>

Final code: 01100
### 3.2.2 Coding of DCT Coefficients (DC)

**Huffman Coding of DC Coefficients**

<table>
<thead>
<tr>
<th>S S S S</th>
<th>DIFF Values</th>
<th>Additional Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>-1,1</td>
<td>0,1</td>
</tr>
<tr>
<td>2</td>
<td>-3,-2,2,3</td>
<td>00,01,10,11</td>
</tr>
<tr>
<td>3</td>
<td>-7,-4,4,7</td>
<td>000,...,0111,1000,111</td>
</tr>
<tr>
<td>4</td>
<td>-15,-8,8,15</td>
<td>00000,...,01111,10000,11111</td>
</tr>
<tr>
<td>5</td>
<td>-31,-16,16,31</td>
<td>000000,...,011111,100000,111111</td>
</tr>
<tr>
<td>6</td>
<td>-63,-32,32,63</td>
<td>.......</td>
</tr>
<tr>
<td>7</td>
<td>-127,-64,64,127</td>
<td>.......</td>
</tr>
<tr>
<td>8</td>
<td>-255,-128,128,-255</td>
<td>.......</td>
</tr>
<tr>
<td>9</td>
<td>-511,-256,256,-511</td>
<td>.......</td>
</tr>
<tr>
<td>10</td>
<td>-1023,-512,512,-1023</td>
<td>.......</td>
</tr>
<tr>
<td>11</td>
<td>-2047,-1024,1024,-2047</td>
<td>.......</td>
</tr>
</tbody>
</table>

**Huffman Table for Luminance DC Coefficient Differences**

<table>
<thead>
<tr>
<th>Category</th>
<th>Code Length</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>010</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>101</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1110</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>11110</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>111110</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>1111110</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>11111110</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>111111110</td>
</tr>
</tbody>
</table>
3.2.2 Coding of DCT Coefficients (AC)

- AC coefficients are re-arranged to a sequence of (run, level) pairs through a zigzag scanning process.
- Level is further divided into (Size Categories, Amplitude).
- Run and size are then combined and coded as a single event (2D VLC).
  - An 8-bit code ‘RRRRSSSS’ is used to represent the nonzero coefficients.
    - The SSSS is defined as size categories from 1 to 11.
    - The RRRR is defined as run-length of zeros in the zig-zag scan or number of zeros before a nonzero coefficient.
    - The composite value of RRRRSSSS is then Huffman coded.
  
  Ex: 1) RRRRSSS=11110000 represents 15 run ‘0’ coef. and followed by a ‘0’ coef.
    2) Multiple symbols used for run-length of ‘0’ coef. exceeds 15
    3) RRRRSSS=00000000 represents end-of-block (EOB)
3.2.2 Coding of DCT Coefficients (AC)

Zig-Zag scan

<table>
<thead>
<tr>
<th>Zero Run</th>
<th>Category</th>
<th>Code length</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td>01</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>4</td>
<td>1011</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>5</td>
<td>11010</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>6</td>
<td>11111000</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>7</td>
<td>11111000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>11000</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
<td>1110001</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>7</td>
<td>1111001</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>9</td>
<td>11111011</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>11011</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>8</td>
<td>11111000</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>6</td>
<td>1111010</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>9</td>
<td>11111011</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>6</td>
<td>111011</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>7</td>
<td>111010</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>7</td>
<td>111011</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>8</td>
<td>11110001</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>8</td>
<td>11110110</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>9</td>
<td>11111100</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>9</td>
<td>111111000</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>9</td>
<td>11111110</td>
</tr>
</tbody>
</table>

Composite values
3.2.2 Coding of DCT Coefficients (AC)

1. Code the amplitude (sign and values) of the nonzero AC coefficient with VLC that is similar to the DC code table.

2. The Huffman tables for AC coef. are defined in Annex K of the JPEG standard.

3. Support for arithmetic coding is included (but is patented).

4. Arithmetic coding can give a 8.8%-17.4% improvement in compression over Huffman coding (mean of entropy coding) with fixed table.

<table>
<thead>
<tr>
<th>Category (SSSS)</th>
<th>AC Coefficient Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>−1,1</td>
</tr>
<tr>
<td>2</td>
<td>−3, −2, 2, 3</td>
</tr>
<tr>
<td>3</td>
<td>−7, ..., −4, 4, ..., 7</td>
</tr>
<tr>
<td>4</td>
<td>−15, ..., −8, 8, ..., 15</td>
</tr>
<tr>
<td>5</td>
<td>−31, ..., −16, 16, ..., 31</td>
</tr>
<tr>
<td>6</td>
<td>−63, ..., −32, 32, ..., 63</td>
</tr>
<tr>
<td>7</td>
<td>−127, ..., −64, 64, ..., 127</td>
</tr>
<tr>
<td>8</td>
<td>−255, ..., −128, 128, ..., 255</td>
</tr>
<tr>
<td>9</td>
<td>−511, ..., −256, 256, ..., 511</td>
</tr>
<tr>
<td>10</td>
<td>−1023, ..., −512, 512, ..., 1023</td>
</tr>
<tr>
<td>11</td>
<td>−2047, ..., −1024, 1024, ..., 2047</td>
</tr>
</tbody>
</table>
3.2.3 JPEG DCT-Based decoding
3.2.3 JPEG DCT-Based decoding

- **Lossless JPEG**
  - Pixel based prediction
  - Prediction error entropy coded
  - Compression of around 2:1

- **Progressive JPEG**
  - Gradual build-up of an Image
    - Spectral selection
      - A subset of coefficients is transmitted during each pass for all blocks. One usually starts with the low frequency coef. and moves towards to the high freq.
    - Successive approximation
      - The most selected coef. are coded and transmitted during the first pass. In each subsequent pass, one extra bit is coded and transmitted to increase the precision of the coef.
3.3 Subband Coding

- The fundamental concept behind Subband Coding is to split up the frequency band of a signal and then to code each subband using a coder and bit rate accurately matched to the statistics of the band.

- This makes each subband at the lower sampling rate.

- At the receiver, the subband are resampled, fed through interpolation filters and added to reconstruct the image.

- With an appropriate choice of filters, perfect reconstructions can be achieved.
3.3.1 Analysis/Synthesis Stages

- Analysis
  - A signal is first filtered to create a set of signals, each of which contains a limited range of frequencies. These signals are called subbands.
  - Since each subband has a reduced bandwidth compared to the original fullband signal, they may be downsampled. That is, a reduced number of samples may be taken of the signal without causing aliasing.

- Synthesis—Reconstruction is achieved by:
  - Upsampling the decoded subbands.
  - Applying appropriate filters to reverse the subbanding process.
  - Adding the reconstructed subbands together.
3.3.1 Analysis/Synthesis Stage

- 2D dimensional decomposition structure
3.3.1 Analysis/Synthesis Stage

- The formation of subbands does not create any compression in itself.
  - The same total number of samples is required to represent the subbands as is required to represent the original signal.

- The subbands can be encoded efficiently:
  - The significance of the different spatial frequencies are not uniform, and this fact may be exploited by different bit allocations to the various subbands
  - The subbands are then encoded using one or more coders. Different bit rates or even different coding techniques may be used for each subband.
3.3.1 Analysis/Synthesis Stage

Ref: H. Wu
3.4 Temporal Redundancy and Prediction for Video Coding

- In video coding, it is necessary to explore the compression in the spatial domain (spatial redundancy in intra-picture) and in time domain (temporal redundancy in inter-picture).

- A video sequence is a series of images with limited motion between adjacent images in most of the time.
3.4.1 Temporal Statistical Redundancy

- Temporal redundancy between pixels of adjacent frames of video sequence
- The pixel differences at the same spatial location between consecutive frames are typically small.

Susie frame 40

Susie frame 41

Susie: abs(frame 40 – frame 41)
3.4.2 Inter-frame Encoder

- Conditional Replacement
  - Inter-frame coders produce differential signals that are effectively zero in the non-changing parts of the picture and non-zero only in the moving areas.
  - It is needed to transmit differential signal values only for the moving areas of the picture.
  - We call this technique – Conditional frame replacement or Inter-frame DPCM.
3.4.2 Inter-frame Encoder

**Encoder**
- Frame $x(n)$
- Error image $e(n)$
- Dequantised error image $e'(n)$

**Transmission or Storage Media**

**Decoder**
- Reconstructed frame $x'(n)$

**Reconstructed frame $\hat{x}(n-1)$**

**Step 1:** Calculate the difference between the current and previous frames;
**Step 2:** Quantise and encode the difference image.
**Step 3:** Add the dequantised (residual) image to the previous frame to reconstruct the current frame of image.
3.5 Motion Estimation & Compensation Algorithms

- The football has moved a limited distance but its shape remains almost constant between two adjacent frames.
- To reduce these temporal redundancies, several motion compensation methods can be applied.
3.5 Motion Estimation & Compensation Algorithms

- 3.5.1 Block based Motion Estimation
- 3.5.2 Full search algorithm
- 3.5.3 Motion Compensation
- 3.5.4 Fast Motion Estimation Algorithm
- 3.5.5 Motion Estimation to sub-pixel accuracy
3.5.1 Block Based Motion Est.

- A common approach in predictive encoder is to attempt to compensate for motion which occurs between frames using block matching.

- Using this approach, the reconstruction of the previous frame is searched for the best match to the current frame on a block by block basis. The location of this matching block (called the motion vector) and the prediction error (residual) is then transmitted.
3.5.1 Block Based Motion Est.

- Each picture is divided into block-based subimages for motion estimation rather than using each pixel as a unit.

- To reduce computational and storage requirements, a limited search area is defined to the position around the current block.

- The range of possible displacement in the X and Y directions is +/- 16.
  - 5 bits to specify the Hor/Ver displacement

- The total motion vector overhead is 10 bits/pixel (more than the entropy of the orig. image)

- 16x16 Macroblock is defined for ME.
3.5.1 Block Based Motion Est.

- Block base search

Reference Frame

<table>
<thead>
<tr>
<th>Motion Vector</th>
<th>w = search range</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

Search Window

Position of Current Block

Current Frame

Current Block

16x16 -- Macroblock
3.5.2 Full Search Algorithm

Search Window Size: $W = +/- \ 16$
3.5.2 Full Search Algorithm

- For each Macroblock (MB)
  - Total search steps = 33x33 = 1089
  - Total pixel operations = 1089x256=278784
  - For CCIR Rec.601 video (704 pixels x 576 pixels)
    - Pixel operation/frame = 441.6 million/frame
    - Pixel operation/second=11.0 billion/second
      (25 frame/second)
- Conclusion:
  Motion Estimation is Highly Computationally Intensive !!!!!!
3.5.3 Motion Compensation

“Previous frame (No. 1)”
3.5.3 Motion Compensation
“Current original frame (No. 2)"
3.5.3 Motion Compensation

“Motion compensated prediction (No. 2)"
3.5.3 Motion Compensation
“Motion compensated prediction error (No. x)”
3.5.4 Fast Motion Estimation Techniques

- Full search motion estimation is computationally complex.
- Several sub-optimum fast search techniques have been developed.
- Many work on the assumption that block matching will improve monotonically as the search moves closer to the optimum point.
- Since they do not examine all of the candidate blocks, the choice of matching block might not be as good as that chosen by a full search. However, the quality-cost trade-off is usually worthwhile.
3.5.4 Fast Motion Estimation Techniques

**Two-dimensional logarithmic search (TDL)**

- This and the following techniques are quadrant monotonic searches.
- Quadrant monotonic assumes that the value of the distortion function increases as the distance from the point of minimum distortion increases.
- Special case of the principle of locality: not only locality to optimal block, but also distance from optimal block.
- TDL is a multi-stage search which successively reduces the search area during each stage until the search area is trivially small.
- Search positions: \(2 + 7 \log_2 w\). Example: for \(w=16\), 30 positions are searched, compared with 1089 for the full search.
- The processing of a search step relies on the previous search steps; therefore, the search steps cannot be performed in parallel.
3.5.4 Fast Motion Estimation Techniques

**Two-dimensional logarithmic search (TDL)**

**Step 1**: The block at the centre of the search area and four blocks at distance $s$ from the centre on the X and Y axes are searched for a best match.

**Step 2**: If the position of best match is the centre, halve the step size ($s/2$). Else, if the best match is in one of the four outer positions, then it becomes the new centre point ([cx,cy]) for the next stage.

**Step 3**: If the step size $s$ is 1, then all nine blocks around the centre are examined, and the best match chosen for the target block. Otherwise, blocks at positions ([cx,cy], [cx+s,cy], [cx-s,cy], [cx,cy+s], and [cx,cy-s]) are searched, and the algorithm goes to stage 2.

**Note**: The points [0,+4], [+4,+4], [+6,+4] are the minima at each stage, and finally [+7,+4] is chosen as the matching block.

For step size $s$ and search window size $w$, the step update algorithm is given by

$$s_0 = 2^\left\lfloor \log_2 w \right\rfloor - 1, \quad s_n = s_{n-1}/2$$
3.5.4 Fast Motion Estimation Techniques “Three Step Search (TSS)”

- Very similar to TDL search, and developed around the same time.
- The three step search tests eight points around the centre instead of four, with the position of minimum distortion becoming the new centre.
- After each stage the step size is reduced.
- For step size $s$ and search window size $w$, the step update algorithm is given by

$$s_0 = 2^{\left[\log_2 w\right]-1}, \quad s_n = 2^{\left[\log_2 s_{n-1}\right]-1}.$$
3.5.4 Fast Motion Estimation Techniques

Hierarchical and sub-pixel block matching
### 3.5.4 Fast Motion Estimation Techniques


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Maximum number of search points</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>Full search</td>
<td>(2w+1)^2</td>
<td>225</td>
</tr>
<tr>
<td>Ghanbari</td>
<td>2+7log₂ w</td>
<td>961</td>
</tr>
<tr>
<td>PHS</td>
<td>1+4log₂ w</td>
<td>3969</td>
</tr>
<tr>
<td>SR</td>
<td>3+2w</td>
<td></td>
</tr>
<tr>
<td>KR</td>
<td>1+6log₂ w</td>
<td></td>
</tr>
</tbody>
</table>

Project – Fast motion estimation

- Diamond Search algorithm
  - A two step gradient-based diamond search algorithm was proposed. It is based on the observation, that 53% (in larger motion case) – 99% (in small motion case) of the motion vectors are enclosed in a circular area with a radius of 2 pixels around the zero motion vector.

- Large diamond search pattern (LDSP) is used for the gradient-based coarse search. When the centred search position of the LDSP shows the minimum SDA, the small diamond search pattern (SDSP) is chosen for fine search
3.5.4 Fast Motion Estimation Techniques

- Issues with fast motion estimation algorithms
  - Full search motion estimation is computationally complex.
  - Fast search algorithms only work based on the assumption that block matching will improve monotonically as the search moves closer to the optimum point, which may not be the case in general.
  - All block matching based motion estimation algorithms assume pixel motion within the block is uniform, which will not work if the block involves object deformation or varying motion speeds.
  - Overlapped block matching algorithm and hexagonal matching algorithms have been introduced to address these two issues.
3.5.5 Motion Estimation to Sub-pixel Accuracy

- After a single pixel accuracy of motion estimation, it can be extended to sub-pixel accuracy by performing bi-linear interpolation between pixels in the search area.

- This can lead to a worthwhile improvement in the performance of motion compensated prediction.
3.5.5 Motion Estimation to Sub-pixel Accuracy