An Architecture for Motion Estimation in the Transform Domain

J. Lee, N. Vijaykrishnan, M. J. Irwin, W. Wolf
Embedded & Mobile Computing Design Center
Pennsylvania State University
Electrical Engineering
Princeton University
http://www.cse.psu.edu/~mdl
Outline

1. Motivation
2. Motion Estimation in the Transform Domain
3. Proposed Algorithm for Efficient Matrix Computation
4. Proposed Architecture
5. Simulation Results
6. Conclusions
Outline

1. Motivation

2. Motion Estimation in the Transform Domain

3. Proposed Algorithm for Efficient Matrix Computation

4. Proposed Architecture

5. Simulation Results

6. Conclusions
Motivation

- Great demand for video applications
  - HDTV
  - Mobile Phone Video Communications
  - Video Streaming on the Internet

- Core algorithms of the standard-compliant video coders
  - DCT (Discrete Cosine Transform) removes spatial redundancies
  - ME and MC (Motion Estimation and Motion Compensation) removes temporal redundancies
  - Entropy Coding removes statistical redundancies
Motivation

- Importance of motion estimation algorithm
  - One of the key algorithms in video coding standards
  - The most computationally demanding algorithm of a video encoder
    (60% ~ 80% of the total computation time is consumed)
  - High impact on the visual quality of reconstructed images
  - Open to competition
Video encoder architecture

a) ME in the spatial domain

b) ME in the transform domain

- DCT
- +
- Quantizer
- Zig-Zag Scan
- Inverse Quantizer
- Run Length Encoder
- Inverse DCT
- VLC Encoder
- Buffer
- Bitstream

Video

Transform Domain ME

Frame Memory

Spatial Domain ME
Outline

1. Motivation

2. Motion Estimation in the Transform Domain

3. Proposed Algorithm for Efficient Matrix Computation

4. Proposed Architecture

5. Simulation Results

6. Conclusions
ME in the Transform Domain

The $N \times N$ discrete cosine transform matrix $T = \{t(k,n)\}$ is given by

$$t(k,n) = \begin{cases} 
1, & k = 0, \ 0 \leq n \leq N - 1 \\
\sqrt{\frac{2}{N}} \cos \frac{\pi(2n+1)k}{2N}, & 1 \leq k \leq N - 1, \ 0 \leq n \leq N - 1 
\end{cases}$$

The 1-D DCT of a sequence $x(n)$, $0 \leq n \leq N - 1$ is given by

$$y(k) = \sum_{n=0}^{N-1} t(k,n)x(n) \quad 0 \leq k \leq N - 1$$
ME in the Transform Domain

The 2-D DCT $F(u, v)$ of 2-D image $f(x, y)$ is given by

$$F(u, v) = c(u)c(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x, y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

$$c(k) = \begin{cases} \sqrt{\frac{1}{8}} & \text{for } k = 0 \\ \sqrt{\frac{2}{8}} & \text{for } k = 1, 2, \ldots, 7 \end{cases}$$

The case of $N = 8$ is considered
ME in the Transform Domain

Properties of DCT transform

\[ DCT(f) \cong \hat{f} = T f T^t \]

\[ T \cdot T^t = I \]

\[ DCT(ab) = DCT(a)DCT(b) = \hat{a}\hat{b} \]
ME in the Transform Domain

Shifting Matrix-Based Algorithm

Previous Blocks

\[ f_{\text{pred}} \]

\[ f_0 \]

\[ f_1 \]

\[ f_2 \]

\[ f_3 \]

Current Blocks

\[ f_{\text{curr}} \]

\[ \Delta x, \Delta y \]

MV

Displacement Matrix

\[ D_n = \begin{bmatrix} 0 & I_n \\ - & - & - & - \\ 0 & 0 \end{bmatrix} \]

\[ f_{\text{pred}} = \sum_{i=0}^{3} V_i f_i H_i \]

\[ V_0 = D_{8-\Delta x} \quad V_1 = D_{8-\Delta x} \quad V_2 = D_{\Delta x}^t \quad V_3 = D_{\Delta x}^t \]

\[ H_0 = D_{8-\Delta y}^t \quad H_1 = D_{\Delta y} \quad H_2 = D_{8-\Delta y}^t \quad H_3 = D_{\Delta y} \]
ME in the Transform Domain

\[ \hat{f}_{\text{pred}} = DCT(f_{\text{pred}}) \]

\[ = DCT(\sum_{i=0}^{3} V_i f_i H_i) = \sum_{i=0}^{3} \hat{V}_i \hat{f}_i \hat{H}_i \]

\[ \hat{V}_i = TV_i T^t \quad \hat{f}_i = Tf_i T^t \quad \hat{H}_i = TH_i T^t \]
Outline

1. Motivation
2. Motion Estimation in the Transform Domain
3. Proposed Algorithm for Efficient Matrix Computation
4. Proposed Architecture
5. Simulation Results
6. Conclusions
Algorithm for Efficient Matrix Computation

\[
\hat{f}_{\text{pred}} = \sum_{i=0}^{3} \hat{V}_i \hat{f}_i \hat{H}_i
\]

\[
= \sum_{i=0}^{3} \sum_{m=0}^{7} \sum_{n=0}^{7} \hat{f}_i(m,n) \hat{V}_i(m) \hat{H}_i(n)
\]

\[
= \sum_{m=0}^{7} \sum_{n=0}^{7} \sum_{i=0}^{3} \hat{f}_i(m,n) \hat{V}_i(m) \hat{H}_i(n)
\]

\(\hat{f}_i(m,n)\): matrix entry in the \(m_{th}\) row and \(n_{th}\) column of the block \(\hat{f}_i\)

\(V_i(m)\): the \(m_{th}\) column of matrix \(V_i\)

\(H_i(n)\): the \(n_{th}\) row of matrix \(H_i\)
Algorithm for Efficient Matrix Computation

\[ \hat{f}_{\text{pred}} = \sum_{m=0}^{7} \sum_{n=0}^{7} \sum_{i=0}^{3} f_i(m,n) \hat{V}_i(m) \hat{H}_i(n) \]

\[ = \hat{f}_0(0,0) \hat{V}_0(0) \hat{H}_0(0) + \hat{f}_1(0,0) \hat{V}_1(0) \hat{H}_1(0) + \hat{f}_2(0,0) \hat{V}_2(0) \hat{H}_2(0) + \hat{f}_3(0,0) \hat{V}_3(0) \hat{H}_3(0) \]

\[ + \hat{f}_0(0,1) \hat{V}_0(0) \hat{H}_0(1) + \hat{f}_1(0,1) \hat{V}_1(0) \hat{H}_1(1) + \hat{f}_2(0,1) \hat{V}_2(0) \hat{H}_2(1) + \hat{f}_3(0,1) \hat{V}_3(0) \hat{H}_3(1) \]

\[ + \hat{f}_0(1,0) \hat{V}_0(1) \hat{H}_0(0) + \hat{f}_1(1,0) \hat{V}_1(1) \hat{H}_1(0) + \hat{f}_2(1,0) \hat{V}_2(1) \hat{H}_2(0) + \hat{f}_3(1,0) \hat{V}_3(1) \hat{H}_3(0) \]

\[ + \ldots + \hat{f}_0(7,7) \hat{V}_0(7) \hat{H}_0(7) + \hat{f}_1(7,7) \hat{V}_1(7) \hat{H}_1(7) + \hat{f}_2(7,7) \hat{V}_2(7) \hat{H}_2(7) + \hat{f}_3(7,7) \hat{V}_3(7) \hat{H}_3(7) \]
Outline

1. Motivation
2. Motion Estimation in the Transform Domain
3. Proposed Algorithm for Efficient Matrix Computation

4. Proposed Architecture

5. Simulation Results
6. Conclusions
Proposed Architecture

Schematic diagram of multiplier element
Proposed Architecture

Schematic diagram of processing element
Motion estimation architecture

ROM Module
(Seven DCT-Transformed Shifting Matrices)

Pipelined Calculation
(Wavefront Movement)

DCT Coefficients
(Frame Memory)

One of the Computational Wavefronts

Pipelined Calculation
(Wavefront Movement)
Proposed Architecture

\[ \hat{f}_{pred} = \sum_{i=0}^{3} \hat{V}_i \hat{f}_i \hat{H}_i = \sum_{m=0}^{7} \sum_{n=0}^{7} \sum_{i=0}^{3} \hat{f}_i(m,n)\hat{V}_i(m)\hat{H}_i(n) \]

\[ \hat{f}_{pred}^k = \hat{f}_{pred}^{k-1} + \hat{f}_i(m,n)\hat{V}_i(m)\hat{H}_i(n) \]
The diagram shows a network of nodes labeled with functions and values. The nodes are connected with arrows indicating the flow of information. The functions include:

- $f_0(0,0)$
- $f_0(1,0)$
- $f_B(0,0)$
- $f_B(0,1)$
- $f_B(0,2)$
- $f_B(0,3)$
- $H_B(0,0)$
- $H_B(0,1)$
- $H_B(0,2)$
- $H_B(0,3)$
- $H_0(0,0)$
- $H_0(0,1)$
- $H_0(0,2)$
- $H_0(0,3)$

The network consists of MUL and PE nodes, indicating multiplication and processing elements, respectively. The arrows connect these nodes, showing the direction of data flow.
Outline

1. Motivation
2. Motion Estimation in the Transform Domain
3. Proposed Algorithm for Efficient Matrix Computation
4. Proposed Architecture
5. Simulation Results
6. Conclusions
Simulation Results

Table 1. Average number of non-zero DCT coefficients per block in the P frame

<table>
<thead>
<tr>
<th></th>
<th>QP</th>
<th>3</th>
<th>9</th>
<th>15</th>
<th>21</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>foreman</td>
<td>QCIF</td>
<td>18.3</td>
<td>5.7</td>
<td>3.0</td>
<td>1.8</td>
<td>1.2</td>
</tr>
<tr>
<td>news</td>
<td>QCIF</td>
<td>11.6</td>
<td>5.7</td>
<td>4.0</td>
<td>2.9</td>
<td>2.1</td>
</tr>
<tr>
<td>paris</td>
<td>CIF</td>
<td>15.3</td>
<td>7.7</td>
<td>5.1</td>
<td>3.8</td>
<td>2.9</td>
</tr>
</tbody>
</table>

(H.263 video encoder is used for the experiment)
Simulation Results

a) Only DC coefficient  
b) The top-left $2 \times 2$ DCT coefficients

c) The top-left $4 \times 4$ DCT coefficients  
d) All the DCT coefficients

Motion predicted images in the DCT domain (QP=1)
### Simulation Results

#### Table 2. Comparison of the transform domain approach and the spatial domain approach

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Frame Memory</th>
<th>Processing Time/Each Search Location</th>
<th>IDCT</th>
<th>Data Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform Domain</td>
<td>$\frac{P_H \times P_v}{N^2}$ 9 bits</td>
<td>155 + $82 m$</td>
<td>Not Needed</td>
<td>Algorithm Independent</td>
</tr>
<tr>
<td>Spatial Domain</td>
<td>$101376 \times 99$ bits</td>
<td>$N64N$</td>
<td>Needed</td>
<td>Algorithm Dependent</td>
</tr>
</tbody>
</table>

---

**Block size:** $8 \times 8 N (8 \times 8)$

**Image size:** CIF $352 \times 288$

**CIF = 352 \times 144**, CIF = 352 × 288

$m$ ♯ of the non-zero DCT coefficients per block, used for $f_{pred}$ estimation
Outline

1. Motivation
2. Motion Estimation in the Transform Domain
3. Proposed Algorithm for Efficient Matrix Computation
4. Proposed Architecture
5. Simulation Results
6. Conclusions
Conclusions

- Benefits of the Proposed Algorithm and Architecture
  1) Better PSNR and compression ratio
  2) Higher throughput rate
  3) Reduction of the frame memory and the memory accesses
  4) Easy to reconfigure to different ME algorithms
  5) Capable of supporting power-aware video encoding
Thank you!
DCT Coding Example for 8x8 Block
Discrete Cosine Transform

\( C(u, v) \)  \hspace{1cm} 2-D DCT Coefficients

\( f(x, y) \)  \hspace{1cm} 2-D Image Data

\[
C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x, y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}
\]

\[
\alpha(u) = \begin{cases} 
\sqrt{\frac{1}{8}} & \text{for } u = 0 \\
\sqrt{\frac{2}{8}} & \text{for } u = 1, 2, \ldots, 7 
\end{cases}
\]
Quantization

$C_q(u, v) = \text{Integer Round} \left( \frac{C(u, v)}{Q_F} \cdot Q(u, v) \right)$

$Q_F : \text{Quality Factor}$

$Q_{\text{PRECISION}} = 50$

<Example of Quantization Table>

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>11</td>
<td>10</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>51</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>58</td>
<td>60</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>57</td>
<td>69</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
<td>87</td>
<td>80</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
<td>109</td>
<td>103</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
<td>104</td>
<td>113</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
<td>121</td>
<td>120</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
<td>100</td>
<td>103</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>
Inverse Quantization & IDCT

- Inverse Quantization

\[ C_{iq}(u, v) = C_q(u, v) \times Q(u, v) \times \frac{QF}{Q_{\text{PRECISION}}} \]

- Inverse Discrete Cosine Transform

\[ \hat{f}(x, y) = \sum_{u=0}^{7} \sum_{v=0}^{7} \alpha(u)\alpha(v)C_{iq}(u, v)\cos\frac{(2x + 1)u\pi}{16}\cos\frac{(2y + 1)v\pi}{16} \]
### <8x8 Block Image Data>

<table>
<thead>
<tr>
<th>173</th>
<th>172</th>
<th>174</th>
<th>175</th>
<th>176</th>
<th>178</th>
<th>176</th>
<th>179</th>
</tr>
</thead>
<tbody>
<tr>
<td>170</td>
<td>172</td>
<td>172</td>
<td>178</td>
<td>176</td>
<td>178</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>176</td>
<td>172</td>
<td>176</td>
<td>175</td>
<td>181</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>172</td>
<td>180</td>
<td>176</td>
<td>170</td>
<td>176</td>
<td>177</td>
<td>177</td>
</tr>
<tr>
<td>172</td>
<td>171</td>
<td>176</td>
<td>176</td>
<td>171</td>
<td>173</td>
<td>180</td>
<td>179</td>
</tr>
<tr>
<td>170</td>
<td>174</td>
<td>176</td>
<td>175</td>
<td>171</td>
<td>174</td>
<td>179</td>
<td>177</td>
</tr>
<tr>
<td>165</td>
<td>168</td>
<td>173</td>
<td>175</td>
<td>172</td>
<td>178</td>
<td>179</td>
<td>181</td>
</tr>
<tr>
<td>165</td>
<td>170</td>
<td>174</td>
<td>174</td>
<td>174</td>
<td>180</td>
<td>176</td>
<td>179</td>
</tr>
</tbody>
</table>

### <DCT Coefficients>

| 1398.9 | -21.1 | -0.7 | -8.4 | -3.6 | 3.9 | 1.2 | -3 |
| 5.0 | 4.0 | 1.2 | 3.1 | 3.8 | -0.9 | -2.1 | 3 |
| -1.0 | -5.8 | -3.8 | 3.9 | 1.8 | -5.0 | 1.9 | -2.1 |
| -0.9 | 2.2 | 1.7 | -0.5 | 0.5 | 2.1 | 0.5 | 1.1 |
| -0.6 | 3.1 | -0.3 | -0.3 | -0.6 | 0.9 | 4.9 | 1.2 |
| -0.5 | 0.2 | 0.3 | -0.6 | -1.3 | 0.3 | 2.2 | 3 |
| 1.1 | 3.3 | 1.4 | 1.0 | -1.2 | -2.1 | -2.9 | 1.1 |
| 0.4 | -3.4 | 1.7 | 0.8 | 2.2 | -0.2 | -2.9 | 0.7 |

### <Quantized DCT Coefficients>

| 87 | -2 | 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 | 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 |

### <Inverse DCT Coefficients>

| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |
| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |
| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |
| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |
| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |
| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |
| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |
| 168 | 171 | 175 | 175 | 173 | 173 | 177 | 180 |