Optimal Linear Interpolation Coding for Server-based Computing

Fei Li and Jason Nieh

Network Computing Laboratory
Columbia University

April 30, 2002

ICC 2002
Content

1. Server-based computing
2. Coding requirements
3. Interpolation algorithms
   1. Optimal linear interpolation
   2. 2DLI – 2D linear interpolation
4. Experimental results
5. Conclusion
Server-based computing
Server-based computing

Lightweight client sends user input to server
Server-based computing

Server executes application logic and encodes display updates
Server-based computing

Server sends encoded display updates back to client
Server-based computing

- **Server-based computing**
  - **advantages:**
    - reduced administrative costs
    - better resource utilization
  - **mechanism:**
    - screen updates are encoded at the server side and delivered to clients
  - **problem:**
    - existing screen update approaches do not support multimedia applications effectively
Coding requirements

- Coding requirements for pixel-based coding approach
  (different from traditional image/video coding)
  - complexity
    - for video with window size of 352 * 240: at least 2Mpixels/sec.
  - independent on applications
    - same coding method on video, console, web page, screen dump, etc.
Coding requirements

- Take advantages of
  - inter-frame redundancy
    - not much similarity existing between two adjacent frames (even less than 30% for lecture video)
    - much less for adjacent update regions
  - intra-frame redundancy
    - neighboring pixel values are the same or vary at a constant rate
Coding requirements
Interpolation algorithms

- Interpolation
  - linear decoding time
  - how about encoding complexity?
  - how about compression ratio (depends on intra-frame redundancy)?
  - can it be lossy?
Optimal linear interpolation (OLI)

- **Formula**
  - a segment function to describe the curve:
    - $f(x) = \sum(tao_i(a_i \cdot x + b_i))$
    - $tao_i = 0, \text{ if } x_i \leq x < x_i+1$
    - $tao_i = 1, \text{ else}$
  - $S = \text{union}(s_i, \text{delta}(s_i))$

- **Criteria**
  - recovering more exact pixel values is preferred
  - recursively selecting pixel values
  - lossy and lossless
Optimal linear interpolation (OLI)

- Encoding complexity
  - intrinsic exponential computational complexity: knapsack problem

- Reason
  - computing for selecting re-sampled pixels under threshold takes exponential complexity
Lossless linear interpolation (2DLI)

- **Formula**
  - \[ x_i = x_0 + i \times \text{delta } x \]

- **Procedure**
  - find independent pixels
  - index dependent pixels: horizontally interpolated or vertically interpolated
  - record independent pixels and index
  - further compression using gzip

- **Properties**
  - linear encoding time
  - lossless
  - greedy
Lossless linear interpolation (2DLI)

- x-axis and y-axis interpolation to find dependent pixel values
- index matrix
- independent pixels
Lossless linear interpolation (2DLI)

- **Sample**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

Independent pixels with delta

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Gzip encoding

Index matrix

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Note: red: independent
black: interpolated
index: 2: independent
1: vertical interpolated
0: horizontal interpolated
Experimental results

- **Test-bed**
  - CPU: AMD 1000MHz
  - RAM: 256MB

- **Images categories**
  - smoothed-toned image
  - desktop (screen-dump)
  - web-page
  - lecture video (to test network collaboration)

- **Comparison on**
  - JPEG, gzip (gunzip), hexitle (VNC), 2DLI
Experimental results (average compression ratio)

![Bar chart showing compression ratios for different image formats: JPEG, Gzip, Hextile, and 2DLI. The chart indicates that JPEG has the highest compression ratio for smooth-toned images.]
Experimental results (average compression ratio)
Experimental results (average compression ratio)

![Graph showing compression ratio for JPEG, Gzip, Hextile, and 2DLI. The graph indicates that 2DLI has the highest compression ratio, followed by Hextile, Gzip, and JPEG.]
Experimental results (average compression ratio)

![Diagram showing compression ratios for JPEG, Gzip, Hextile, and 2DLI for lecture video. The diagram displays a comparison of compression ratios across different methods, with 2DLI having the highest compression ratio.]
Experimental results (en/decoding complexity Mpixels/sec)

smooth-toned image coding complexity

- JPEG
- JPEG gzip
- gzip gunzip
- Hextile
- Hextile 2DLI
- 2DLI

Series1
Experimental results (en/decoding complexity Mpixels/sec)

web page coding complexity

<table>
<thead>
<tr>
<th>Method</th>
<th>Coding Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEG</td>
<td>1</td>
</tr>
<tr>
<td>JPEG</td>
<td>1</td>
</tr>
<tr>
<td>gzip</td>
<td>4</td>
</tr>
<tr>
<td>gunzip</td>
<td>4</td>
</tr>
<tr>
<td>Hextile</td>
<td>12</td>
</tr>
<tr>
<td>Hextile</td>
<td>12</td>
</tr>
<tr>
<td>2DL</td>
<td>4</td>
</tr>
<tr>
<td>2DL</td>
<td>4</td>
</tr>
</tbody>
</table>

web page

April 30, 2002

ICC 2002
Experimental results (en/decoding complexity Mpixels/sec)
Experimental results (en/decoding complexity Mpixels/sec)

lecture video coding complexity

- JPEG
- gzip
- gunzip
- Hextile
- Hextile
- 2DLI
- 2DLI

Coding complexity
Conclusion

- 2DLI achieves a good combination of high compression ratio with low coding time

- Much better compression ratio than JPEG, gzip or VNC on web pages, screen dumps, and lecture video

- Compression ratio is second only to JPEG on smooth-toned images, but with much lower coding time

- Further research: higher compression ratio with system approaches, like process migration with X window, to reduce coding complexity at the server side