PDE based Image Compression in Real-Time

H. Köstler, M. Stürmer, Ch. Freundl, U. Rüde
Lehrstuhl für Systemsimulation
Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

July 8, 2008
1. Software package for variational approaches

2. Real-time image compression
   - Image compression
   - Image decompression
   - Experimental results
Goals and applications

Provide a software package:
- Support the development of new variational approaches
- Efficient: multilevel and multigrid solvers
- Parallel
- Special hardware: Cell processor, GPU

Applications:
- Optical flow, Non-rigid image registration, Motion blur
- Tomographic image reconstruction
- Image inpainting, Image segmentation, Image denoising
- Image compression
Frame of original image sequence

(a) Uncompressed 24 bpp
Stored landmarks

(b) 10 % points left
Compression scheme

Idea for image compression \cite{GWW+05}

Some points are selected from each frame, such that they are sufficient for a good reconstruction of the original image sequence. We call these points landmarks. The rest of the points is dropped.

- B-Tree Triangular Coding\cite{DNV97}: fast recursive subdivision scheme to determine landmarks
- Huffman coding to store values
- Possible to store additional points by local variance estimation
Euler-Lagrange equations

Solve the PDE

\[(1 - c(x))Lu - c(x)(u - f) = 0 \tag{1}\]

with

\[c(x) = \begin{cases} 
1 & x \in \Omega_1 \\
0 & \text{else} 
\end{cases} \tag{2}\]

and

\[f(x) = \begin{cases} 
v(x) & x \in \Omega_1 \\
0 & \text{else} 
\end{cases} \tag{3}\]

From these equations it follows that for every landmark \(x \in \Omega_1\):
\(u = v = f\). At every other point \(x \in \Omega \setminus \Omega_1\) the PDE \(Lu = 0\) has to be solved.
Regularization

For the regularizer $L$ we use homogeneous diffusion (HD)

$$Lu = \Delta u$$

(4)

nonlinear isotropic diffusion (NID) [PM90]

$$Lu = \text{div}(g(|\nabla u|^2)\nabla u)$$

(5)

and nonlinear anisotropic diffusion (NAD)

$$Lu = \text{div} \left( g \left( \nabla u_\sigma \left( \nabla u_\sigma \right)^T \right) \nabla u \right)$$

(6)
Implementation on a PlayStation™ 3
The Cell Processor consists of a PowerPC-based general purpose core – the Power Processor Unit (PPU) – and eight Synergistic Processor Elements (SPEs), simple but very powerful co-processors.

The SPEs are SIMD only vector engines and operate on 256 kB of own Local Store.

One Cell processor is able to perform 204.8 GFlop/s in total at 3.2 GHz.

But only six SPEs are available under Linux → 153 GFlop/s.

Intel Pentium 4 (3.6 GHz): 14.4 GFlop/s.
### Decompression Runtimes

**Table:** Comparison of decompression times (in fps) on Pentium and Cell processors for different regularizers using one or three levels for a resolution of $320 \times 240$ pixels. Additionally we provide relative reconstruction errors ($e_{rr}$).

<table>
<thead>
<tr>
<th>regularizer</th>
<th>fps (Pentium)</th>
<th>1 grid level fps</th>
<th>3 grid levels fps</th>
<th>$e_{rr}$</th>
<th>color space</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>3.7</td>
<td>101</td>
<td>89</td>
<td>0.06</td>
<td>RGB</td>
</tr>
<tr>
<td>NID</td>
<td>0.55</td>
<td>48</td>
<td>35</td>
<td>0.06</td>
<td>RGB</td>
</tr>
<tr>
<td>NAD</td>
<td>0.43</td>
<td>34</td>
<td>25</td>
<td>0.05</td>
<td>RGB</td>
</tr>
<tr>
<td>NADVAR</td>
<td>101</td>
<td>63</td>
<td>-</td>
<td>0.04</td>
<td>Y’CbCr</td>
</tr>
</tbody>
</table>
Image quality I (bit rates and PSNR)

(c) Uncompressed 24 bpp

(d) MPEG-1 384 kbit/s, 32.3 dB
Image quality II (stored landmarks)

(e) 10 % (no variance)  (f) 15 % (with variance)
Image quality III

(g) PDEVC NAD 3.2 Mbit/s, 23.2 dB
(h) PDEVC NADVAR 4.4 Mbit/s, 29.3 dB
Image quality IV

(i) HD

(j) NID

(k) NAD
Image quality V

(I) MPEG  (m) NADVAR  (n) Uncompressed
Future work

- Improve compression scheme for video coding, evaluate practical use, we achieve a compression of about 1:10, current codecs about 1:100
- Better selection of points
- Use temporal information
- Color quantization
- Test different inpainting approaches
- Apply to compression of 3D medical images
- Compare to GPU implementation
Related Literature

