Dynapack: Space-Time compression of the 3D animations of triangle meshes with fixed connectivity

Lawrence Ibarria, Jarek Rossignac
College of Computing and GVU Center
Georgia Tech, Atlanta
Animated meshes

- Dynamic Product inspection
- Virtual Reality Engines
- Interactive 3D movie streaming
Dynapack Overview

- Quantize dataset to integers
- Encode first Key Frame
- Traverse mesh generating next key-frame
- Proceed with next frame
Quantization

- Quantization is lossy, but not always noticeable

<table>
<thead>
<tr>
<th>Original 32 bits float</th>
<th>Quantized to 13 bits per coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quantized to 11 bits per coordinate</th>
<th>Quantized to 7 bits per coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Encoding First Frame

- Connectivity will remain constant through all the animation
- We use Edgebreaker to encode the connectivity
  - Other approaches may easily be used instead
- We use the standard Edgebreaker for the first frame
- We use a modified Edgebreaker for subsequent frames
Dynapack Algorithm (recursive version)

dynapack(c) {
    IF c == –1 THEN RETURN;  #compression of a component of a frame
    IF NOT c.t.m THEN {
        IF NOT c.v.m THEN {
            encode(c.v.g( f ) – predict(c, f ) )  #return if a border is reached
            c.v.m := TRUE;
        }
        c.t.m := TRUE;  #if triangle c.t not yet visited
        encode residue coordinates
        c.v.m := TRUE;
        c.t.m := TRUE;  #mark the tip vertex as visited
        dynapack(c.r);  #mark the triangle as visited
        dynapack(c.r);  #try to go to the right neighbor
        dynapack(c.l);  #try to go to the left neighbor
    }
}

#compression of a component of a frame
#return if a border is reached
#if triangle c.t not yet visited
#if tip vertex not yet visited
#encode residue coordinates
#mark the tip vertex as visited
#mark the triangle as visited
#try to go to the right neighbor
#try to go to the left neighbor
Predictors

- Requirements
  - Simple
  - Small number of points
- Predictors
  - Space only: Parallelogram Predictor
  - Time only: Location on the previous frame
  - Space-Time ELP: Extended Lorenzo Predictor
    - Predicts perfectly translations
  - Space-Time Replica
    - Perfect predictor for translations, rotations, scaling
Space-Only Predictor

\[
predict(c,f) = c.n.v.g(f) + c.p.v.g(f) - c.o.v.g(f)
\]

- The old parallelogram
- Uses one frame at a time
- Does not exploit temporal coherence
Time-only predictor

\[
predict(c,f) = c.n.v.g(f-1)
\]

- Expects the vertex to stay in the same place as in the previous frame
- Does not exploit coherence between neighbors along the surface
Extended Lorenzo Predictor (ELP)

\[
predict(c,f) = c.v.g(f-1) + (c.n.v.g(f) - c.n.v.g(f-1)) + (c.p.v.g(f) - c.p.v.g(f-1)) - (c.o.v.g(f) + c.o.v.g(f-1))
\]

Exploits both space and time coherence
Replica Predictor

- \( \text{predict}(c, f) = \text{c.o.v.g}(f) + aA' + bB' + cC' \)

- Expresses a vertex in coordinate system of neighbor triangle
- Exact predictor for rigid body transforms and scaling

\[
\begin{align*}
a &= \frac{A \cdot D + B \cdot B - B \cdot D + A \cdot B}{A \cdot A + B \cdot B - A \cdot B + A \cdot B} \\
b &= \frac{A \cdot D + A \cdot B - B \cdot D + A \cdot A}{A \cdot B + A \cdot B - B \cdot B + A \cdot A} \\
c &= D \cdot \frac{A \times B}{\|A \times B\|^2} + \sqrt{\|A \times B\|}
\end{align*}
\]

\[
\begin{align*}
A' &= \text{c.p.v.g}(f) - \text{c.o.v.g}(f), \\
B' &= \text{c.n.v.g}(f) - \text{c.o.v.g}(f), \\
C' &= \frac{A' \times B'}{\sqrt{\|A' \times B'\|^2}}.
\end{align*}
\]
Results: sub-sampled head

- Space only is very poor
- Other 3 are similar at 13 bits
- Replica is the best for cruder quantization

<table>
<thead>
<tr>
<th>Head Shaping</th>
<th>7 Bit</th>
<th>9 Bit</th>
<th>11 Bit</th>
<th>13 Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space only</td>
<td>3.07</td>
<td>4.94</td>
<td>6.98</td>
<td>9.16</td>
</tr>
<tr>
<td>Time Only</td>
<td>0.80</td>
<td>1.13</td>
<td>1.52</td>
<td>2.02</td>
</tr>
<tr>
<td>ELP</td>
<td>0.61</td>
<td>0.96</td>
<td>1.42</td>
<td>2.05</td>
</tr>
<tr>
<td>Replica</td>
<td>0.60</td>
<td>0.94</td>
<td>1.39</td>
<td>2.02</td>
</tr>
</tbody>
</table>
Results: Chicken Crossing

- ELP and Replica are much better than the other two.
- They yield similar results.

<table>
<thead>
<tr>
<th>Chicken Crossing</th>
<th>7 Bit</th>
<th>9 Bit</th>
<th>11 Bit</th>
<th>13 Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space only</td>
<td>1.90</td>
<td>3.37</td>
<td>5.20</td>
<td>7.19</td>
</tr>
<tr>
<td>Time Only</td>
<td>1.78</td>
<td>3.29</td>
<td>5.03</td>
<td>6.91</td>
</tr>
<tr>
<td>ELP</td>
<td>1.37</td>
<td>1.79</td>
<td>2.28</td>
<td>3.01</td>
</tr>
<tr>
<td>Replica</td>
<td>1.37</td>
<td>1.83</td>
<td>2.35</td>
<td>2.91</td>
</tr>
</tbody>
</table>
Conclusions

- Effective compression of 3D animation frames
- Trivial implementation
- Needs only previous frame
  - Small foot-print
  - Suitable for out-of-core compression/decompression of large sets
  - Perfect for streaming animations from the Internet
  - Perfect for live compression
- Currently limited to constant connectivity
- Not limited to any kind of deformations