Eliminating the Back-Tracking Step in the Longest Common Subsequence (LCSS) Algorithm for Video Sequence Matching

Werner Bailer
SAMT 2008, Koblenz, 2008-12-04
Outline

- Motivation
  - detecting repeated video content
- Core Problem: distance between video sequences
- Proposed distance measure
- Optimisation
- Demo
Core Problem: Measuring Distance between Video Sequences

- **Requirements**
  - support any type of features (audio/video, discrete or continuous values, scalar/vector, custom distance functions between two feature values/vectors)
  - support partial match of sequences
  - different timing of sequences
  - support gaps and insertions in the matching parts
  - minimum length of useful match

- **Approaches**
  - Dynamic Time Warping: assign each element of sequence A to nearest element of sequence B
  - String edit distance: identical, replace, insert, delete
Distance Measure for Video Sequences (1)

- match sequences of feature vectors extracted from the input videos
- based on Longest Common Subsequence (LCSS) algorithm
  - variant of string edit distance
  - build matrix from matches
  - find longest matching subsequence which may have gaps by back-tracking
  - original LCSS algorithm assumes discrete input values, match := equality of input values
- LCSS for vectors of continuous values
  - proposed for 2D trajectories [Vlachos et al., 2002]
  - match := Euclidian distance between elements $< \varepsilon$ and offset between elements $< \delta$
Distance Measure
for Video Sequences (2)

- for matching feature sequences of videos
  - replace $\varepsilon$ by a vector of thresholds $\theta_{\text{sim}} = \{\varepsilon_1, \ldots, \varepsilon_F\}$ for $m$ features $1, \ldots, F$, which are weighted by weights $\{w_1, \ldots, w_F\}$
  - match features $f$ with appropriate distance functions
  - discard $\delta$, absolute temporal distance of feature vectors in the sequence is irrelevant
  - accept all matches longer than a minimum length $\theta_{\text{len}}$
  - introduce maximum gap $\gamma$ between two consecutive matching feature vectors
  - consequence of gap constraint:
    - not just longest common subsequence (might have gaps)
    - but all sufficiently long ($> \theta_{\text{len}}$) subsequences
  - similarity := length of match / min(lengths of input sequences)
- **1D Example**
  - input sequences top/left
  - $\theta_{\text{sim}} = 0.5$
  - $\theta_{\text{len}} = 3$
  - $\gamma = 1$

```
<table>
<thead>
<tr>
<th></th>
<th>2.3</th>
<th>4.7</th>
<th>1.2</th>
<th>3.3</th>
<th>2.2</th>
<th>1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3.4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2.4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4.6</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1.5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2.6</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

INNOVATION aus TRADITION
Distance Measure for Video Sequences (4)
Distance Measure – Optimisation: Eliminating Back-Tracking

- **proposed optimisation**
  - keep while building matrix
  - last match in each line and list of current sequences
  - for each match
    - find nearest previous match (city block distance – 2)
    - search in \((γ + 1)\) lines, add to this sequence: \(O(1)\)
Demo

- Video Browsing Tool
- Rushes Skimming