Character Animation Seminar

Motion Graphs

03.07.08

Pascal Pohl
Introduction

- Motion Capturing:
  - Actor plays different scenes
  - Gained information used to animate joints of the virtual character

- Last talk: Motion synthesis from annotations
  - Interactive tool to generate animations
  - Process controlled by annotations
Introduction

- Motion Capturing leads to good animations but is expansive
- How to make the data more flexible?
Related issue

- Motion capture
- Manual animation
- Inverse kinematics
The idea of Motion Graphs

  - They created new connections between the different motion capture clips in order to generate new animation
  - Organized as directed graph:
    - every edge $\leftrightarrow$ piece of motion
    - nodes represent connections of these motion clips
    - nodes serve as choice points
The idea of motion graph

• How to build a Motion Graph:
  – All motion clips in the database correspond to edges of the motion graph
  – Create nodes at the beginning and end of every motion clip
  – Result is a simple directed graph, split clips to get more nodes
Construction of motion graphs

- To get a more useful graph more connections are needed
- How to find such connections?
Construction of Motion Graphs

- Motion data consists of clips (set of successive frames) defined by parameters (position of root joint, quaternions for other joints, constraints and additional labels)
- Similar postures in different motion clips are represented as transition nodes
- Every motion that includes such a posture gets connected to that node
- How do we find these postures?
How to evaluate the distance?

- Naive way with simple vector norms would fail:
  - The motions are defined to a rigid 2D coordinate system
  - Discard pose position and overall orientation in Cartesian space
  - More information is needed then from one frame (speed, accelerations)
How to evaluate the distance?

• The character is seen as a cloud of joints in relative Cartesian space

• Use labels to identify some categories of motions (running, sneaking, crawling...) (motion from annotation)
Calculate Differences

- To compare frames, calculate the minimal weighted sum of squared distances

$$\min_{\theta, x_0, z_0} \sum_i w_i \| p_i - T_{\theta, x_0, z_0} p_i' \|^2$$
How to evaluate the distance

- Measure differences of frames by distances between the point clouds
- Use windows of frames to compare motion clips. Gives additional parameters to the evaluation (velocity or accelerations)
How to evaluate distance

- Can't decide if this frame is taken from a walking or running scene
- Also if posture is not exactly the same in both motions, no information about velocity and accelerations included in one frame
Walking scene

frame 1

frame 2

frame 3
Running scene

frame 1

frame 2

frame 3
Comparison of the frame windows

walking clip

running clip
Find good „spots“ for transitions

- Calculate differences for every pair of frames in the database
- Use results as a 2D error function and calculate local minima

entry (i,j) contains error between frame Ai and Bj, white means a low error
Find good „spots“ for transitions

- These local minima not necessarily lead to good transitions (just better then other points)
- Improvement by using a threshold, accepting only values below that threshold
- Often variable threshold, controlled by the user according to the scene. (fast or slow, smooth or frantic).
- Threshold deals between quality of blends and connectivity of the graph
Finalize the graph

- Graph must be pruned, could contain sinks and dead ends
- Both must be deleted to be able to perform unlimited motions
- Labels must be checked, node may have no confirming labels on all entering and leaving edges
Finalize the graph

- Dead ends: when entering a dead end, no further motion is possible

- Sinks: Still endlessly long motion synthesis possible (by loops), but some nodes are no longer reachable
Welding clips (Creating new transition)

Original motion

new transitions

Original motion
Creating new transition

- Poses are similar, but not close enough to enable seamless junctions
- Walking on graph requires filtering between clips at new connection points
Creating Transitions

- If \( D( A_i, B_j ) \) meets the thresholds, motion \( B_j \) must first be translated to same point as \( A_i \)

- Necessary to do some additional work:
  Perform linear interpolation between the frames \( A \) and \( B \)
Creating Transitions

- Linear interpolation may hurt constraints of the primary frames, so correct them (delete or add)
- Build new set of labels for the new transition:

\[ L_1 \cup L_2 \]
Graph construction completed

• We have the finalized graph.
  How do we extract new motions?
How to get motion from the graph

- From a graph walk to motion (random walks): Since the graph is constructed out of pieces of motion, not much has to be done:
  - Place motions at the right position => multiply translations for every edge, starting at the origin
  - Regard constraints, may be violated through interpolation
Search for motion

• How to find motions according to user's inputs?
• Compute function

\[ f(w) = f([e_1, \ldots, e_n]) = \sum_{i=1}^{n} g([e_1, \ldots, e_{i-1}], e_i) \]

to sum up the errors of the used edges in the graph walk.

• User supplies function \( g(e, w) \) (additional error) and a halting condition to stop the walk

• Goal is to find a complete path that minimizes \( f \)
Search for motion

- Naive way to find path is depth-first search -> too expansive
- Better: use branch and bound strategy:
  - $w$: Path from origin to actual position, $v$: next edges
  - $f(w)$ is lower bound to $f(w+v)$ for any $v$
  - Pursue optimal path $w_{opt}$, stop branch if $f(w+v) > f(w_{opt})$
  - Add threshold $\varepsilon$ (user controlled) so that a path is good enough if $f(w) \leq \varepsilon$
- Further improvement: search for $n$ frames but only retain the first $m$ and start at point $m$ again.
Search for motion
Search for motion

- Choice of g leads to different results
- Given a start and an ending frame, the result can be very different from the expected path
- Sometimes more freedom (just specifying the sort of motion) yields to a better result
- Too much freedom varies the result to much

Character turns on one place to find a better alignment to the desired path (figure from L. Kovar and M. Gleicher and F. Pighin (2002))
Path synthesis

- Aim: generate motion along a given path (in space)
- Now a concrete function is used for \( g(e,w) \)
- Differences between the actual and the wanted position of the character must be computed to get a close locomotion to the desired path
Path synthesis

- Project root joint on the floor -> received points form a linear curve
- \( w_i := \) path to the \( i \)-th frame
- \( s_{wi} := \) arc length from the starting point
- Leads to the following function:
  \[
g(w, e) = \sum_{i=1}^{n} \| P'(s(e_i)) - P(s(e_i)) \|^2
\]
- Completeness condition: Path length of \( P' \) is sufficient long
Take away

- The use of motion graphs adds more control to animation processes based on motion capture
- Animations generated by the help of a directed graph
- Changing the weight function leads to different results
- Cannot extrapolate from data
- Extensively cited (Lee J. et al. (2002) avatars)

- **Uses motion graphs**
  - In the lowest layer they use motion graphs to organize the motion clips

- **Improvements**
  - More complexity added to the graph
  - Second layer with clusters of similar postures included
  - Clusters get connected to cluster-trees
  - Provides a better interface to user control
  - Offers different kinds of control (sketch, choice or vision)
Motion capture

Lower layer: motion graph

Create cluster of similar motion clips

Build Cluster trees

Higher layer: cluster forest
Conclusion

• Motion graphs
  – Higher connectivity gives more control to the user
  – Preserves good quality of motion capture
  – Interactive control possible
  – Still quiet restricted to the recorded motions
  – Thresholds must be set by hand (different motion needs different fidelity)
  – No further informations can be saved in the motion clips and considered when synthesizing motion (such as physics or cinematic constraints)