Motion Retiming using CUDA and Smooth Surfaces

Innfarn Yoo, Michel Abdul, Illia Ziamtsov, Raymond Hassan, Bedrich Benes
College of Technology, Purdue University, 401 N. Grant St., West Lafayette, IN 47907
yooi@purdue.edu, michel.abdul@gmail.com, iziamtso@purdue.edu, rhassan@purdue.edu, bbenes@purdue.edu

High Performance Computer Graphics Laboratory

PURDUE UNIVERSITY

NVIDIA CUDA™ TEACHING CENTER

Abstract

Motion retiming is used to edit character animations to match a given time. It is a non-trivial task to retime the motion of a set of joints since spatio-temporal correlation exists among them. It is especially difficult in the case of motion capture when there are forward kinematic keys on every frame to define the motion. We present a novel approach to **motion retiming that exploits the proximity of joints as a way of preserving the motion coherence**. **Our framework** that allows users to **intuitively and interactively retime motion using CUDA**. The free-form smooth surface located on the timeline that can be interactively deformed to move the action of a particular joint to a certain time. The animation is retimed by manipulating successive smooth surfaces and the final animation is generated by CUDA and resampling the original motion by time spans.

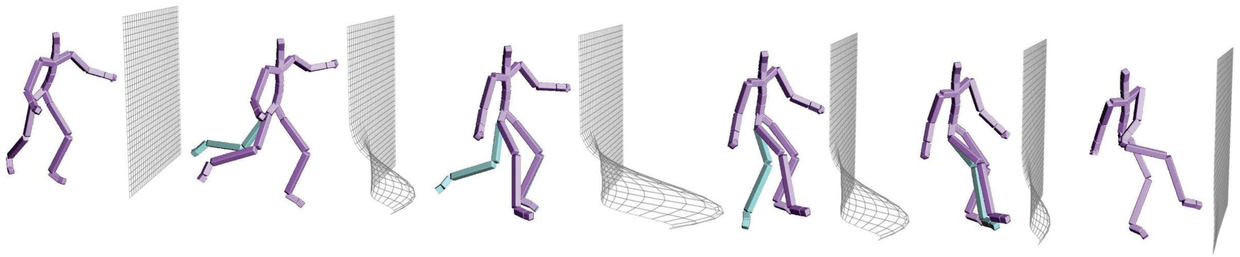

Figure 1. Retimed soccer kick motion. The cyan character represents the original motion and the purple character represents the resulting motion. The surfaces shown in front of the characters represent the smooth surfaces for each frames.

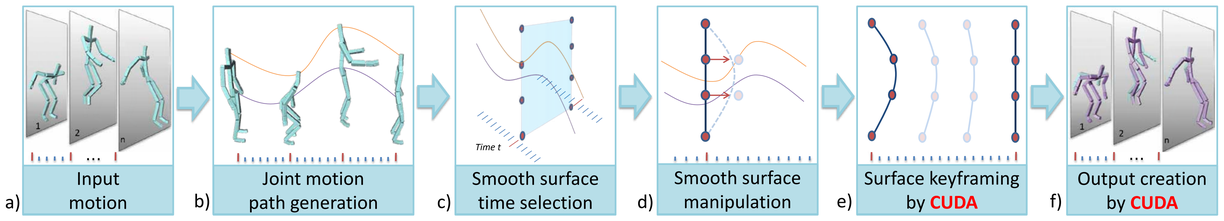

Figure 2. System Pipeline. The input of the framework is a character motion (a) from which the joint motion projected path curves are generated (b). The user sets a smooth surface at a specific time for retiming (c) and adjusts it by manipulating control points (d). The user creates keyframes of smooth surfaces at different times (e) and the system generates a new motion (f).

1. Motivation

Retiming animation in an intuitive way poses a difficult problem because of the need to maintain the **spatial and temporal coherence of the different parts of the animated character**. We observe that the spatial proximity of joints translates into temporal coherence of their motion. **Our framework can be** applied for retiming any motion sequence, it **preserves global motion coherence**, it preserves **smoothness of the joint curves**, and it is **simple to control**.

2. Projected Motion Curves

The first step in the process of motion retiming is to create joint paths from existing motion data. Since the generated joint paths have **4 dimensions (x, y, z, and t), the visual-ization and intuitive control of joints' timing is not a trivial task.** To tackle this problem, we reduce dimensions of the motion using a **projection onto the time direction**.

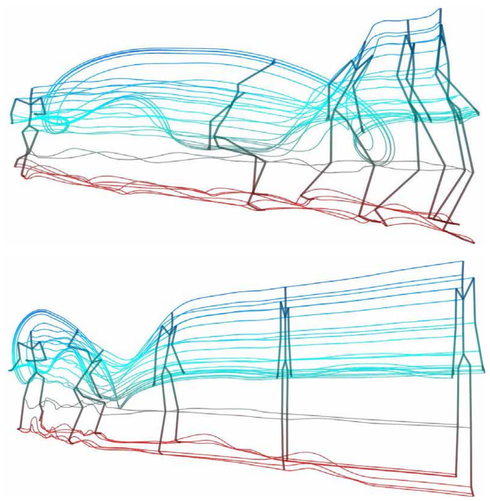

Figure 3. Original motion curves (up) and the projected motion curve (bottom).

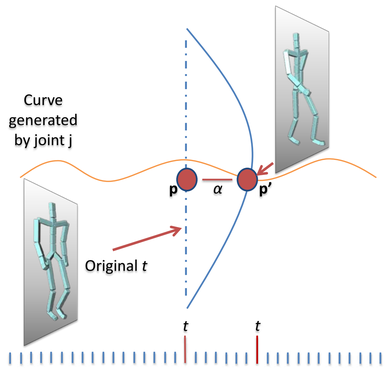

Figure 4. Smooth surface editing. The user moves the control points of the surface. New motion parameters are sampled from the intersection between the surface and the joint paths.

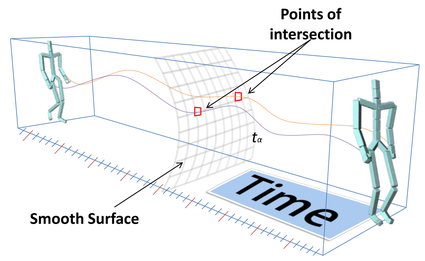

Figure 5. Intersection points between the smooth surface and the joint paths.

4. Smooth Surface Control

we introduce smooth surfaces that that allows one to manipulate the timing of each joint with respect to the joints' topological information (see Figures 4 and 5). Initially, a smooth surface is a bounded and conceptually flat surface that represents a specific time step t on the projected spatial domain. **Our method allows users to manipulate the surface as a free-form deformation, and it can be understood as time being bent in the spatial domain**.

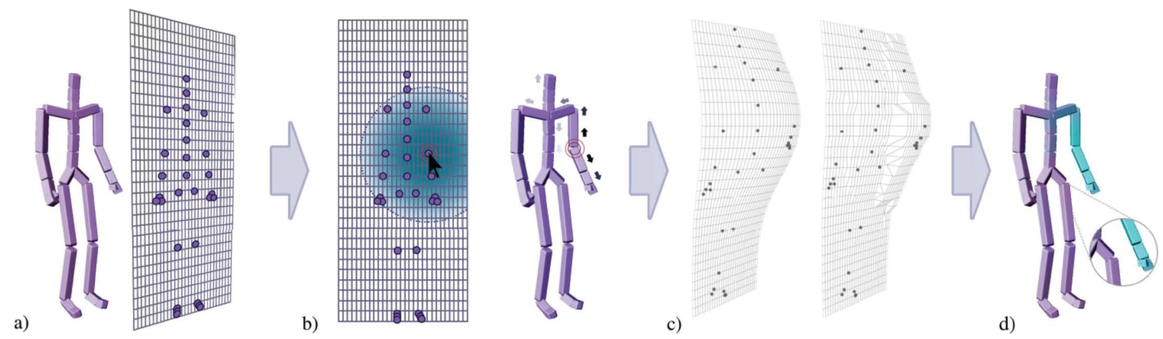

Figure 6. Topological distance also considered in the smooth surface. The user clicks on the elbow joint and the topological distance from the elbow to the rest of the joints is computed.

5. Output Motion Generation

We compute the joint paths with smooth surfaces using CUDA, and also can be calculate the new time t'. The new motion can be also generated by feeding the new time t'. This means that we are replacing the rotation of the joint at time t with the rotation of the joint at time t, effectively applying a retiming operation to the joint.

6. Results

The CUDA implementation calculates **new time t' for all joints immediately**, and allow to create the **retimed motion in Real-Time**. CUDA generates the result **x times faster** than CPU implementation.

	CPU	GPU	GPU Speedup
CPU vs NVIDIA GTX 870M	1.3488 ms	0.1570 ms	**8.59 times**
CPU vs NVIDIA Quadro K5000	1.8663 ms	0.0631 ms	**29.6 times**
CPU vs NVIDIA Tesla K40	1.3701 ms	0.0424 ms	**32.3 times**

Table 1: Performance comparison between CPU and GPU implementation

ACKNOWLEDGMENTS
We would like to thank NVIDIA for providing graphics hardware. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.