# Real-time Avatar Animation Synthesis from Coarse Motion Input

Krzysztof Pietroszek
CSU Monterey Bay
Seaside, California, USA
kpietroszek@csumb.edu

Phuc Pham
CSU Monterey Bay
Seaside, California, USA
ppham@csumb.edu

Sophia Rose
CSU Monterey Bay
Seaside, California, USA
sorose@csumb.edu

Liudmila Tahai
University of Waterloo
Waterloo, Ontario, Canada
ltahai@uwaterloo.ca

Irene Humer
California Polytechnic State Univ.
San Luis Obispo, California, USA
ihumer@calpoly.edu

Christian Eckhardt
California Polytechnic State Univ.
San Luis Obispo, California, USA
ceckhard@calpoly.edu

## ABSTRACT

We present an animation synthesis technique that produces high quality avatar animation from coarse or incomplete motion input by matching, in real-time, the input against thousands of high-quality pre-recorded motion capture templates. The technique uses subsequence multivariate Quaternion Dynamic Time Warping with the angle between quaternions as a distance measure. Our implementation of the technique using compute shaders shows performance gains orders of magnitude greater than the state-of-the-art DTW implementations.

## CCS CONCEPTS

• **Human-centered computing → Empirical studies in interaction design**; • **Computing methodologies → Procedural animation**;

## KEYWORDS

spatial input, animation synthesis, dynamic time warping

**ACM Reference format:**
Krzysztof Pietroszek, Phuc Pham, Sophia Rose, Liudmila Tahai, Irene Humer, and Christian Eckhardt. 2017. Real-time Avatar Animation Synthesis from Coarse Motion Input. In *Proceedings of VRST '17, Gothenburg, Sweden, November 8–10, 2017,* 2 pages.
https://doi.org/10.1145/3139131.3141223

## 1 INTRODUCTION

Avatar body animation is an important aspect of immersion, especially in the context of social VR, where avatar body language and facial expression complement voice communication. With the introduction of low-cost motion capture devices – such as Microsoft's Kinect, the VIVE Sensor, the Perception Neuron suit, and the Senso suit – a user's body motion can be mapped in real-time to their avatar representation in immersive systems. However, the quality of the resultant avatar animation depends on the quality of the input. Low-cost motion capture systems usually provide only a coarse input that produces low-quality avatar animation. The issues with resultant animations are caused by self-occlusion (Kinect), incomplete data (VIVE Sensor), or drift/magnetization of sensor components (Perception Neuron and Senso suits).

One approach to improving the avatar's animation is to synthesize richer animation from coarse input. Inverse Kinematics is a common technique used due to its low computational cost. Another technique proposes stitching together pre-recorded motion templates after they are matched against the coarse input for similarity [Müller and Röder 2006]. For real-time applications, the limitation of this technique is the number of templates that the input can be matched against. As a solution to this problem, we present a GPU compute shader implementation of multivariate subsequence Quaternion Dynamic Time Warping algorithm (CS-QDTW[1]), based on our previous work [Pietroszek et al. 2017]. Our implementation matches coarse motion input against thousands of pre-recorded high-quality motion templates in real-time.

### 1.1 Template Matching

Dynamic Time Warping has often been used as a baseline technique for motion capture template matching [Barbič et al. 2004; Müller and Röder 2006]. Recognition of motions can be achieved by finding similarity between N-dimensional coarse input *query* and pre-recorded motion capture *templates*. Motion capture templates can be recored as a time series of relative rotations of humanoid skeleton joints. Each joint rotation is represented as a unit quaternions $q = [w, < x, y, z >]$, where $< x, y, z >$ is a unit vector representing rotation axis and $w = cos(\frac{\Theta}{2})$, where $\Theta$ is the angle of rotation around that axis. While DTW usually uses Euclidean distance, it is not a good measure for Quaternions. This is because for $q_1 = [w, < x, y, z >]$ and $q_2 = [-w, < -x, -y, -z >]$ representing the same rotation in 3D space, Euclidean distance between $q_1$ and $q_2$ is not 0. We propose to use the angle $\alpha$ between two quaternions as a better measure of DTW distance: $\alpha(q_1, q_2) = \arccos(2 < q_1, q_2 >^2 -1)$ where $< q_1, q_2 >$ denotes an inner product of $q_1$ and $q_2$: $< q_1, q_2 >= w_1w_2 + x_1x_2 + y_1y_2 + z_1z_2$

### 1.2 Applying Matching Animations to Avatar

Matching whole body motion input against a single animation template is possible using our technique, but it would require a template for all possible combinations of body section movements. To reduce the number of templates used in matching, we perform

[1]CS-QDTW source is available at http://www.github.com/gameresearchlab/cs-qdtw


*VRST '17, November 8–10, 2017, Gothenburg, Sweden*

ACM ISBN 978-1-4503-5548-3/17/11.
https://doi.org/10.1145/3139131.3141223

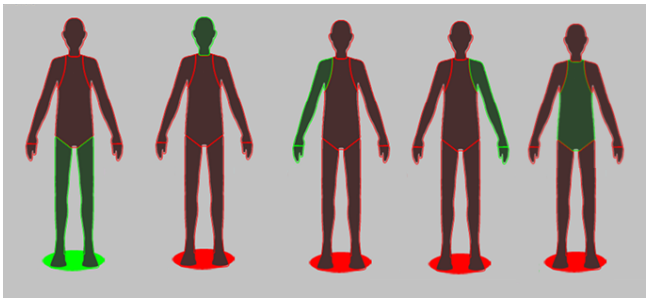

**Figure 1: Motion capture templates are matched and applied separately to legs, head, right arm, left arm, and abdomen**

the match of input separately for five body sections: legs, right hand, left hand, head, and abdomen (Figure 1).

This approach also lets us modularize the animation matching based on available input. For example, if no motion tracking for legs is available, but motion tracking is available for hands (e.g. using HTC Vive hand controllers), the method matches input using the tracking data for the hands, but retrieves the template animation for the whole body. The resulting animation represents not the actual but the *plausible* animation of legs.

### 1.3 Transitioning Between Motion Templates

We use *direct blend trees* to transition between the motion templates. Direct blend trees compute interpolation between N motion templates, where each template contributes the amount of joint rotation defined by a weight value. Transitions between two templates can be implemented by decreasing the weight of the current animation and increasing the contribution of the upcoming animation. While the transitional animation may be perceived as less realistic than the original motion capture animation, such animation transition is smooth, as opposed to animations resulting from applying the coarse motion input directly that are often perceived as jerky.

## 2 EXPERIMENT

To verify the performance of our system, we compared the state-of-the-art UCR-DTW Suite [Rakthanmanon et al. 2012] with our CS-QDTW implementation. The dataset for our experiment includes 2000 motion capture templates, where each template is a two-seconds-long time series of 6 quaternions, representing feet, hands, and head rotations recorded at a rate of 24 frames per second. The motion data used in our experiment comes from Carnegie Mellon University's Graphics Lab motion-capture database.

The goal of the experiment was to perform a continuous search for animation templates that best match spatial inputs of the length 20 frames as provided in real time by Microsoft Kinect V2. We ran the experiment on an Intel i7 workstation with 32GB of RAM and an NVIDIA GTX1070 with 1920 shader units.

## 3 RESULTS AND DISCUSSION

Table 1 summarizes the performance results. We note that the processing times for our technique are orders of magnitude faster than the processing times of UCR-Suite. Additionally, our processing times have little variance for a given number of templates – an important property for a real-time system. UCR-Suite's standard deviation is large because UCR-Suite relies on optimizations [Rakthanmanon et al. 2012] that improves the performance of matching for some, but not all, inputs. Also, multi-threaded UCR-Suite implementation of DTW, if available, would have performed better.

**Table 1: Average processing times in *ms* (standard deviation reported in parentheses) of our approach vs. UCR Suite [Rakthanmanon et al. 2012] excluding the time required to read the data from disk.**

| | 500 templates | 1000 templates | 2000 templates |
|---|---|---|---|
| CS-QDTW | 28(5)*ms* | 59(2)*ms* | 78(5)*ms* |
| UCR-Suite | 3238(2333.7)*ms* | 3235(4233)*ms* | 3404(3829)*ms* |

## 4 LIMITATIONS AND FUTURE WORK

Our experiment shows that matching coarse input against thousands of templates can be achieved in real time. However, we have not yet performed a user study verifying the recognition rate for user activities. Our pilot studies indicate that templates for legs were correctly matched in 83% of cases. If the wrong motion template is identified, the animation still appears smooth, but no longer mirrors user activity.

The latency inherent to our approach is another limitation. To match the input with a template, we must first collect long enough input. In our experiments, we collected 20 motion capture frames before attempting the match. In result, despite short processing time, the proposed approach has too high end-to-end system latency for applications such as first person shooter games or fighting simulators. However, we believe that the latency would not negatively affect the experience of social VR if synchronized with real, or imposed, latency of voice chat and face animation.

## 5 CONCLUSION

Our work presents an opportunity for real-time matching of coarse motion capture input with thousands of high-quality motion capture animation templates. Our experiment shows that by using a compute shader implementation of multivariate subsequence Quaternion Dynamic Time Warping algorithm (CS-QDTW), it is possible to match a coarse motion input against thousands of templates in real-time. The technique could find applications in immersive environments that tolerate some animation latency.

## REFERENCES

Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K Hodgins, and Nancy S Pollard. 2004. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004*. Canadian Human-Computer Communications Society, 185–194.

Meinard Müller and Tido Röder. 2006. Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 137–146.

Krzysztof Pietroszek, Pham Phuc, and Christian Eckhardt. 2017. CS-DTW: Real-time Matching of Multivariate Spatial Input Against Thousands of Templates using Compute Shader DTW. In *Proceedings of the 2017 Symposium on Spatial User Interaction*. ACM.

Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 262–270.