The Uniform Posture Map Algorithm for the Real-Time Interactive Motion Transitions of an Articulated Body

Jin Ok Kim[1], Bum Ro Lee[2], Chin Hyun Chung[2], Jun Hwang[3], and Woongjae Lee[3]

[1] School of Information and Communication Engineering, Sungkyunkwan University, 300, Chunchun-dong, Jangan-gu, Suwon, Kyunggi-do, 440-746, KOREA
jinny@ece.skku.ac.kr

[2] Department of Information and Control Engineering, Kwangwoon University, 447-1, Wolgye-dong, Nowon-gu, Seoul, 139-701, KOREA
chung@kw.ac.kr

[3] Division of Information and Communication Engineering, Seoul Women's University, 126, Kongnung2-dong, Nowon-gu, Seoul, 139-774, KOREA
wjlee@swu.ac.kr

Abstract. It is important to reuse existing motion capture data for reduction of the animation producing costs as well as for the efficiency of the producing process. Because its motion curve has no control point, however, captured data is difficult to modify interactively. Motion transition is a useful method for reusing existing motion data. It generates a seamless intermediate motion with two short motion sequences. In this paper, the Uniform Posture Map (UPM) is proposed to perform motion transitions. The UPM is organized through the quantization of various postures with an unsupervised learning algorithm; it places the output neurons with similar postures in adjacent positions. Using this property, an intermediate posture of applied two postures is generated; the generating posture is used as a key-frame to make an interpolating motion. The UPM needs fewer computational costs, in comparison with other motion transition algorithms. It provides a control parameter; an animator can not only control the motion simply by adjusting this parameter, but also produce animation interactively. The UPM prevents the generating of the invalid output neurons to present unreal postures in the learning phase; thus, it makes more realistic motion curves; finally it contributes to the making of more natural motions. The motion transition algorithm proposed in this paper can be applied to various fields such as real time 3D games, virtual reality applications, and web 3D applications.

1 Introduction

Computer systems play an important role in various fields of industry. Through the automation of tedious repetitions, they maximize the efficiency of operations and encourage workers to focus on their jobs. This is also true in the process

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2657, pp. 295–304, 2003.

of digital animation production. While animators had to draw each frame manually in the past, the introduction of the key-frame animation technology has allowed animators to devote more time to their creative work. Since all motion dynamics occurring in animation are decided according experience and intuition, the resulting animation could be extremely subjective. Such a method could be suitable for the exaggerated expressions of character animations, but not for realistic ones. Motion capture technology, however, could solve these problems. The motion-capture system records the signal of sensors attached on the articulated body over time. Because the motion data obtained from the motion capture system is a reflection of the actor's dynamics, animation that is more realistic than key frame animation can be produced. However, although such a motion capture system can record realistic articulated body motions, captured motion data lacks control points to modify motion curves; it is also impossible to control the motion in a consecutively captured frame. Since one has to capture motion separately whenever one needs similar motions with inconsiderable difference, the motion data editing has been studied from various points of view. If one could reuse existing motion data through editing, it is possible to reduce the cost of producing animation while making the production process more efficient. There are various methods of editing the captured motion [1] [2] [3] [4]. One builds a basic motion database composed of short unit motion clips. One selects two clip motions from this database, and then one makes a seamless new motion by connecting two unit motion clips by generating proper intermediate motions. Rose [5] adopted nonlinear programming to make such intermediate motion data. Nonlinear programming is a solution for nonlinear equations with energy constraints, and has a tendency of minimizing energy. Since the real behavior of an articulated body does not always minimize energy, it could generate an unnatural motion. In this paper, a form of methodology is proposed for performing a motion transition with a Uniform Posture Map (UPM) and to represent the motion tendency of an articulated body with several parameters. Because motion transition is produced by the UPM learned through real motion, more realistic intermediate motion data can be produced. Although the map learning process needs a heavy computational cost, the motion generating process is less expensive. Therefore, it is possible to produce intermediate motions in real-time.

In this paper, a model with a minimum DOF (Degree of Freedom) is defined by the posture of an articulated body. In the learning phase, one can classify the bones of an entire articulated body into four bone classes according to their properties. The DOF vector is defined as a set of DOF values included in each bone class, and four partial posture maps are generated for each class. In the synthesis phase, we generate four partial postures with four partial posture maps and assemble the partial postures into a complete posture. This posture is used to generate intermediate motion frames by interpolating the B-spline [6]. In comparison with many other heavy computational algorithms, the learning algorithm does not need heavy computational costs; additionally, an animator could control the resulting motion by adjusting only one parameter [7] [8]. Thus, our algorithm provides a real time method for an animator to produce anima-

tion interactively. Above all, our algorithm contributes to the making of a more natural motion. One creates new motion on the basis of deductive information, which is not calculated by the inductive mathematical process. Therefore, the new generated motion never exhibits unnatural behavior. This is a very important superior feature of our algorithm, because other algorithms adopt a lot of physical constraints that require more computational costs in order to make motion more natural.

2 Uniform Posture Map

The most popular algorithm, applied to the motion signal processing such as a motion transition and a motion blending, is an optimization method called Non Linear Programming (NLP), Sequential Quadratic Programming, or BFGS. Many researchers have proposed user-interactive motion editing methods with spacetime constraints [9]. However, there are several problems with this approach; one is the computation load problem [10] [11]. In this paper, we use a model with 21 bones; it is a DOF vector with 63 elements. It is a significantly high-order degree. Trying to search the error surface in a 63-dimension vector space and calculating the exact object function in each numerical step during the optimization process, it could be a horrible experience. In general, most practical NLP requires a computation load of $O(n^2)$ and $O(n^3)$; this is a critical problem in real-time-user-interactive systems [11]. Another problem is that NLP always decides its solution to minimize its energy object functions. However, the model, which moves its body by the power of its own muscles, does not always move to minimize energy. In particular, when we want to edit a choreographed motion such as a ballet, the problem would be prominent. The third problem is the convenience of motion control. To control motion easily, the motion transition algorithm has to provide an intuitive control parameter. But NLP does not have such control parameters; thus, it is difficult to control result motions through intuitive interactions. Kohonen's self-organizing map is an unsupervised learning network for solving many problems with the NLP algorithm [12] [13] [14] [15]. It consists of an input layer and an output layer, and organizes a feature map reflecting the feature of input samples in a learning phase. Fig. 1 shows the basic structure of self-organizing maps.

In general, the self-organizing map reflects a feature of the sample space correctly, and has a relatively faster learning time. But when the sample space has a complex shape, the general learning process cannot generate a proper output neuron. In particular, when the sample space has a non-convex, the part of the output neurons are sometimes formed out of the valid space, because the self-organizing learning algorithm has a tendency of preserving its neighborhood relationships [16] [17]. This could be a negligible in most of other applications, because it is not a frequent occurrence. However, the sample space of the human body's DOF is non-convex because of the mechanical restriction of the human body. This means that the calculated intermediate motion vector could be located out of an available region. Therefore, it is possible that the intermediate

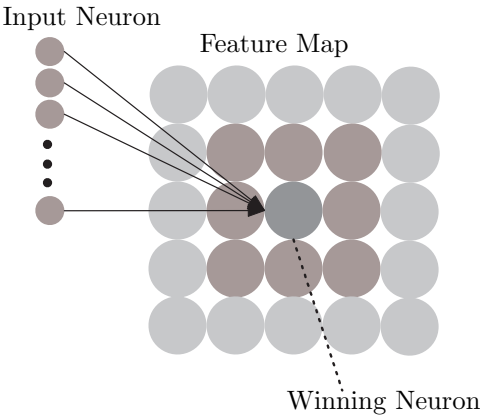
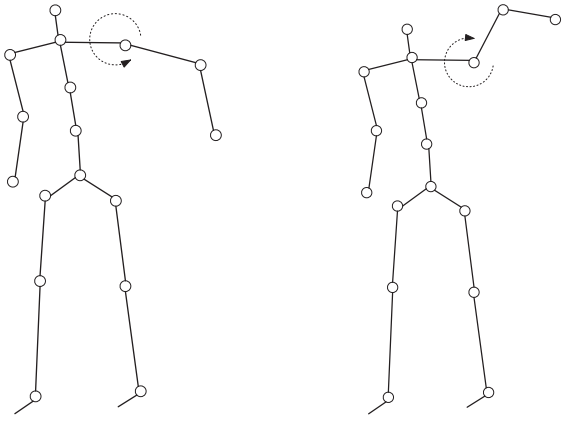


Fig. 1. Self-organizing map.

motion calculated by mathematical methods can not exist in the real world. Fig. 2 shows an example of the mechanical constraint of the human body. Fig. 2(a) shows an arm turning counterclockwise, and Fig. 2(b) shows the arm turning clockwise. The intermediate posture between two postures can be calculated by linear interpolation, but the posture is nonexistent in the real world because of the angular restriction of the shoulder joint. This fact shows that the human posture space is non-convex. Therefore, the self-organizing map could generate unreal motions in an application using human bodies, such as human character animation, posture correction systems and medical systems.

It is necessary to decide on a lot of parameters to train the self-organizing map. To get a proper parameter set, frequent learning and checking for a generated self-organizing map is required. In actual cases, the method for determining such parameters is not defined as a rule. Therefore, such parameters have to be determined by a tedious trial-and-error process. To solve these problems, we propose a Uniform Posture Map (UPM). We set its initial weights randomly and make a self-organizing map on the basis of these initial weights. For the proposed algorithm, we organize its network topology on the basis of the information of the sample vector. The only required two parameters to be determined previously are the maximum acceptable radius of an output neuron, and the size of its neighborhood. Algorithm 1 describes the UPM algorithm.

If there is an input sample with a distance from all output neurons beyond a critical value, it does not vary the existing output neurons, add a new output neuron to its map, and set the weight of new output neurons with the input sample. Therefore it could reserve its validation in a non-convex region, and overcome the drawback of self-organizing maps. It has computational merit in intermediate motion generating because it requires only the computation load,



(a) Turning counterclockwise

(b) turning clockwise

Fig. 2. An example of mechanical constraint.

$O(n)$, to calculate a distance of two points and distances between all output neuron and a certain point in the map.

3 Experiment and Result

3.1 Experiment

The motion transition system is composed of three separate phases: the data preprocess phase, the map leaning phase and the motion synthesis phase. In the data pre-processing phase, we first gather human motion-capture data with our optical motion-capture system. The data consists of the 3D positions of optical markers attached on human actors, acquired over short intervals. We map the 3D positions into the DOF of our bone set. Next, we modulate the motion data to set bounds for the angular range from 0 to 2π, and reduce some redundant DOFs by a manual data modification. In the learning phase, we divide the motion data by their property into four classes: the body, the global, the medium and the detail. We make four uniform posture maps by the UPM algorithm. Once we make the posture maps, we need not build another posture map in the next processing phase and can reuse the posture map. In the motion synthesis phase, we synthesize an intermediate posture containing the feature of two postures simultaneously: the first posture of the second motion, and the last posture of the first motion. With a generated intermediate posture, we make a new motion

Algorithm 1 The UPM (uniform posture map) Algorithm

Input: ψ (a sample posture vector)
Output: Φ (a posture map)
UPM_Algorithm(ψ)
(1) {
(2) **Weight initialization:** Set the maximum radius of output neuron, R, and the initial radius of the neighborhood, N, and the input sample as a weight of the first output neuron.
(3) Calculate the distances, D_i, between all output neurons and input sample.
(4) **if** $|D_i - R| > 0$
(5) New output sample is inserted into the map and the input sample is set as a weight of new output.
(6) **Calculation of activation:** The activation X_i of input unit i is determined by the instance presented to the network. The activation O_j of output unit j is calculated by

$$O_j = F_{\min}(D_j) = F_{\min}\left(\sum_i (X_i - W_{ji})^2\right)$$

where $F_{\min}$ is the unit function if unit j is the output node with minimum D_j of its neighborhood, and the zero function otherwise.
(7) **Weight training:** Weight is updated by

$$\Delta W_{ji} = O_j \, \eta \, (X_i - W_{ji})$$

where η is a gain term $(0 < \eta < 1)$ that decreases over time. The radius of the neighborhood also decreases over time.
(8) **if** $\eta = 0$
(9) **return** Φ.
(10) **else**
(11) Go to Step 3 and repeat the process.
(12) }

by approximation of the B-spline. Figure 3 show an overall system configuration used in our experiment.

We make a nine processed sample motion clips with 1800 frames. The applied software, such as motion classifier, and self-organizing map learning algorithm, and synthesizing, B-spline module, are implemented with Visual C++.

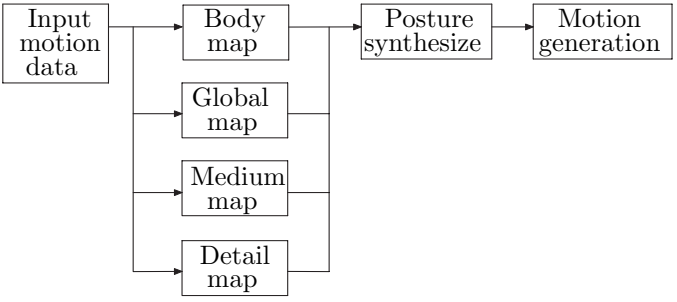
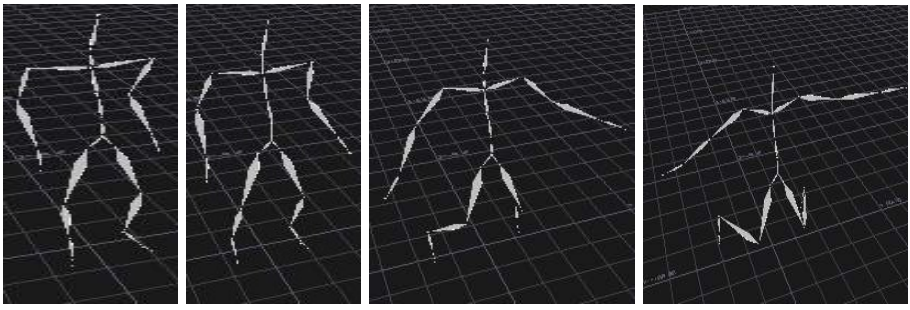


Fig. 3. Motion generation phase.

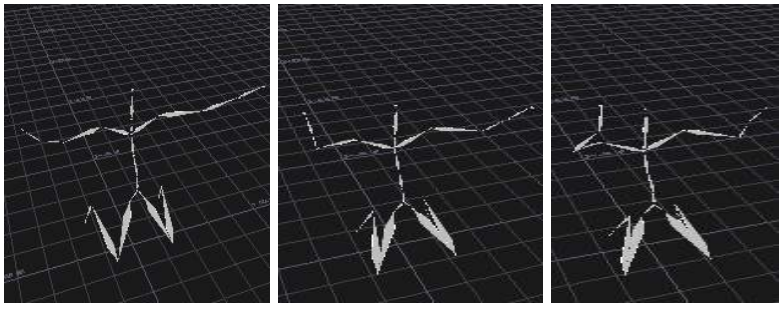
3.2 Result

We have applied our motion transition algorithm to various motion clips and demonstrated with 2 motion clip pairs in this paper - two dance motion clips, and two cocktail blending motion clips, which has 200 frames. We use 20 frames as transition interval and insert the key frame, made by our algorithm, to tenth interpolation frame. In Fig. 4, we display the motion transition result with dancing motion clips. Each frame is extracted from 5 frames. The frames, used in motion transition algorithm, are shown in Fig. 5(b) and Fig. 5(f), and synthesized intermediate frame is shown in Fig. 5(d). That is similar to Fig. 5. In interpolated motion, because each posture is generated on the basis of real posture, most of motion has a realistic posture. With careful observation, we can notice the fact that the generated intermediate posture is not based on simple linear interpolation from Fig. 4 and Fig. 5.

For example, in Fig. 5(d), the DOF of right shoulder is not numerical average of the DOFs of Fig. 5(b) and Fig. 5(f), but entire motion is not unrealistic. In General, computation times for the motion transitions are strongly dependent on the number of DOF in the involved model. In Rose' experiment [18], the motion transition of 44 DOF model generating the spacetime transition motion took 72 seconds in Pentium 100MHz System. But we've generated our intermediate 20 frames in a second. Although we have used the more powerful system, because we've used very higher dimension model with 69 DOF, it is enough to prove the merit of our motion transition algorithm in the aspect of motion generation speed. When the two motions, applied in this experiment, have a significant difference in lower part of the body motion, it appears a skidding effect between ground and foot. In this case, we must consider the additional algorithm to solve this problem [6]. But we leave that problem to feature consideration.



(a) 195th frame (b) 200th frame (c) 205th frame (d) 210th frame

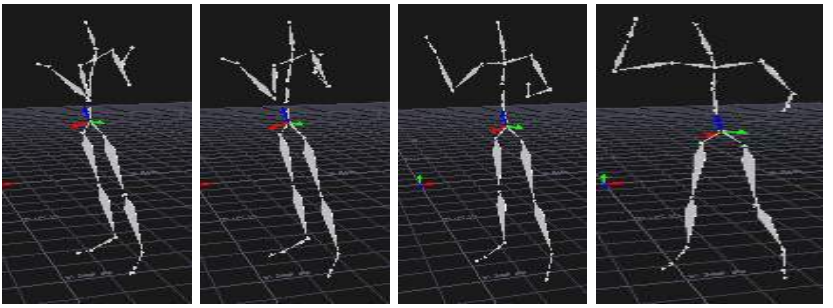


(e) 215th frame (f) 220th frame (g) 225th frame

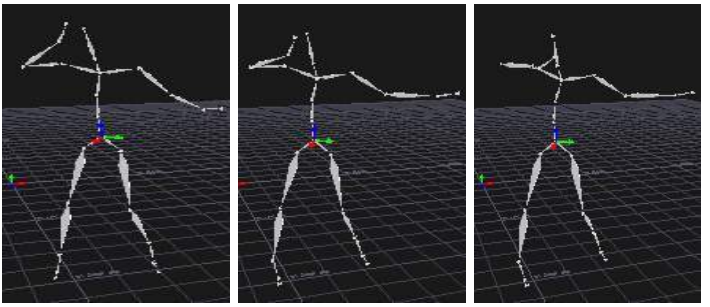
Fig. 4. Motion transition – Dance.

4 Conclusion

Many researchers have taken the effort to describe the dynamics of the articulated body by the analytic method. They have obtained excellent results in various fields. However, for the articulated body moving with its voluntary will, it is difficult to generalize the motion pattern by analytical modeling, because the motion pattern is extremely subjective and unpredictable. The learning networks overcome the restriction of analytic modeling through the deductive learning method. The UPM proposes to synthesize a new motion between existing clip motions. It requires only the computational load, $O(n)$, to generate intermediate motion curves, which, in turn, make animators generate intermediate motions in real-time. Additionally, it provides an intuitive control parameter: an animator can control motion simply by adjusting this parameter. This property of the UPM makes an animator produce animation interactively. Above all,



(a) 195th frame (b) 200th frame (c) 205th frame (d) 210th frame



(e) 215th frame (f) 220th frame (g) 225th frame

Fig. 5. Motion transition – Cocktail.

our algorithm contributes to making more natural motions. The UPM prevents unreal posture from generating in learning time; therefore, the new generated motion never exhibits unnatural behavior. This is a very important superior feature of the algorithm. In order to make motions natural, other algorithms have to adopt more physical constraints that require more computational costs. In addition, the UPM could be applied to various fields such as real time 3D games, virtual reality applications, and web 3D applications.

References

1. Jung, S.K.: Motion Analysis of Articulated Objects for Optical Motion Capture. Ph. D. dissertation, KAIST (1997) Dept. of Computer Science.
2. Wiley, D.J., Hahn, J.K.: Interpolation synthesis of articulated figure motion. IEEE Computer Graphics and Applications **17** (1997) 39–45

3. Brudelin, A., Williams, L.: Motion signal processing. In: Proc. of SIGGRAPH 95, ACM Press (1995) 97–104
4. Witkin, A., Popović, Z.: Motion warping. In: Proc. of SIGGRAPH 95, ACM Press (1995) 105–108
5. Rose, C., Bodenheimer, B., Cohen, M.F.: Verbs and adverbs: Multidimensional motion interpolation. IEEE Computer Graphics and Applications **18** (1998) 32–40
6. Zhao, J., Badler, N.I.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. ACM Transactions on Graphics **13** (1994) 313–336
7. Noser, H., Thalmann, D.: A rule-based interactive behavioral animation system for humanoids. IEEE Transactions on Visualization and Computer Graphics **5** (1999) 281–307
8. Sannier, G., Balcisoy, S., Thalmann, N.M., Thalmann, D.: A system for directing real-time virtual actors. The Visual Computer **15** (1999) 320–329
9. Gleicher, M.: Motion editing with spacetime constraints. In: Proc. of Symposium on Interactive 3D Graphics. (1997) 139–148
10. Brand, M., Hertzmann, A.: Style machine. In: Proc. of SIGGRAPH 2000, ACM Press (2000) 183–192
11. Grzeszczuk, R., Terzopoulos, D., Hinton, G.: Neuro animator: Fast neural emulation and control of physics-based models. In: Proc. of SIGGRAPH 98, ACM Press (1998) 9–20
12. Kohonen, T., Huang, T.S., Schroeder, M.R.: Self-Organizing Maps. 3 edn. Springer Verlag (2000)
13. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: Neural-gas network for vector quantization and its application to time-series prediction. IEEE Transactions on Neural Networks **4** (1993) 558–568
14. Fritzke, B.: Growing grid – a self-organizing network with constant neighborhood range and adaptation strength. Neural Processing Letters **2** (1995) 9–13
15. Baraldi, A., Alpaydin, E.: Constructive feedforward art clustering networks. IEEE Transactions on Neural Networks **13** (2002) 662–677
16. Ritter, H., Martinetz, T., Schulten, K.: Neural Computation and Self-Organizing Maps. Addition Wesley, New York (1992)
17. Gallant, S.I.: Neural Network Learning and Expert System. MIT Press (1993)
18. Rose, C., Guenter, B., Bodenheimer, B., Cohen, M.F.: Efficient generation of motion transitions using spacetime constraints. In: Proc. of SIGGRAPH 96, ACM Press (1996) 147–154