GENERATING HUMAN INTERACTIVE BEHAVIOURS USING THE WINDOWED VITERBI ALGORITHM

Yue Zheng, Yulia Hicks, Dave Marshall
Cardiff University, Cardiff, UK, CF24 3AA

Darren Cosker
Department of Computer Science, Bath University, Bath, UK, BA2 7AY

Keywords: Windowed Vitebi Algorithm, Generating interactive behaviours, Dual HMM.

Abstract: In this paper, we propose a new approach for generating interactive behaviours for virtual characters, namely the windowed Viterbi algorithm, capable of doing so in real-time. Consequently, we compare the performance of the standard Viterbi algorithm and the windowed Viterbi algorithm within our system. Our system tracks and analyses the behaviour of a real person in video input and produces a fully articulated three dimensional (3D) character interacting with the person in the video input. Our system is model-based. Prior to tracking, we train a collection of dual Hidden Markov Model (HMM) on 3D motion capture (MoCap) data representing a number of interactions between two people. Then using the dual HMM, we generate a moving virtual character reacting to (the motion of) a real person. In this article, we present the detailed evaluation of using the windowed Viterbi algorithms within our system, and show that our approach is suitable for generating interactive behaviours in real-time.

1 INTRODUCTION

In recent years, motion capture technology has come into common use in the computer graphics and computer vision area. Using motion capture equipment, real person behaviours can be recorded and exported as MoCap data. Many researchers have become interested in producing virtual worlds and populating them with virtual characters using MoCap data (Zordan and Horst, 2003), (Meredith and Maddock, 2005) and (Wen et al., 2006). There has been also a limited amount of research into enabling virtual characters with the ability to produce intelligent behaviour on the basis of visual analysis of the scene, which mainly was conducted in the computer vision area. Johnson et al. (Johnson et al., 1998) developed a system capable of producing a 2D silhouette of a virtual person interacting with a real person in video and demonstrated it working with a handshake behaviour. Jebara et al. (Jebara and Pentland, 2002) developed a system capable of producing a dynamic human face together with speech reacting to the events around it. However, as Jebara states himself, the system exhibited only limited intelligent behaviour. Both of the above systems automatically learnt the intelligent behaviours from observed video data and represented them using HMMs (Rabiner, 1989), which are commonly used for representing temporal dynamics of the data. In our previous research (Zheng et al., 2006), we tracked a 3D person in real video using a HMM trained on 3D MoCap data, and then used the tracked person as the input to generate interactive behaviours for virtual characters. However, our system was able to only post process the data as opposed to real-time processing.

In this paper, we propose a new approach for generating interactive behaviours, namely the windowed Viterbi algorithm (Pilu, 2004), capable of doing so in real-time. Consequently, we compare the performance of the standard Viterbi algorithm and the windowed Viterbi algorithm within our system. The advantage of the standard Viterbi algorithm (Rabiner, 1989) is that the generated behaviours look very similar to the real behaviours. However, when we use this method, it requires the full observation sequence before the processing starts, thus making real-time processing impossible. When we use the windowed Viterbi method instead, it does not require the full ob-

Zheng Y., Hicks Y., Marshall D. and Cosker D. (2008).
GENERATING HUMAN INTERACTIVE BEHAVIOURS USING THE WINDOWED VITERBI ALGORITHM.
In *Proceedings of the Third International Conference on Computer Graphics Theory and Applications*, pages 353-358
DOI: 10.5220/0001097503530358

servation sequence before the processing starts, thus it can be used in a real-time system. Moreover, we still can obtain realistic generated interactions behaviours, as we show in our results section.

The organization of the paper is as follows: in Section 2, we describe our model of interactive behaviours. Then we explain the behaviour generating part in Section 3. The experiments together with evaluation are presented in Section 4, followed by our conclusion and future work in Section 5.

2 MODEL OF INTERACTIVE BEHAVIOURS

In this work, our dual HMM was trained to represent several types of interactive behaviours involving two people. In particular, we model the following behaviours: a handshake between two people; one person pulling another person and one person pushing another person. The model is trained on the 3D MoCap data of two real people. In our experiments we use PhaseSpace motion digitizer system to capture several sets of motion data in 3D space with 30 markers, therefore each pose is represented by a 90-dimensional vector. Such data is always constrained by physical and dynamical factors, thus we reduce its dimensionality using Principal Component Analysis (PCA), keeping 95% of the eigenenergy (leaving 11 dimensions), and then train our HMM on a number of such vectors.

A dual HMM, such as we use here, has two sets of states. The first set of states models the poses for the first person (A), and the other set of states models the poses for the second person (B). Each state in the model is modelled with a Gaussian. The first HMM is defined as:

$$\lambda_A = (T_A, B_A, \pi_A) \quad (1)$$

where T_A is the state transition probability matrix, B_A is the observation probability distribution and π_A is the initial state probability distribution.

The second HMM is defined as

$$\lambda_S = (T_A, B_S, \pi_A) \quad (2)$$

It has the same transition matrix as λ_A. The means and covariances of the Gaussians are calculated from the data set representing the motion of the second person (B).

Using this dual HMM, we can estimate a sequence of 3D poses for a virtual character given a sequence of 3D poses of the first person (A) as described in the next section.

3 GENERATING INTERACTIVE BEHAVIOURS

In our system, we track 3D articulated motion of a real person in a video sequence using annealed particle filtering (Deutscher et al., 2000), after which we use the obtained 3D data in conjunction with the dual HMM and the standard Viterbi algorithm to generate the responsive behaviour for a virtual character. Finally, the virtual character performing the generated motion can be placed back into the original video sequence, as shown in Figure 1. In this article, we focus our attention on generating interactive behaviours using the windowed Viterbi algorithm in the latter part of our system.

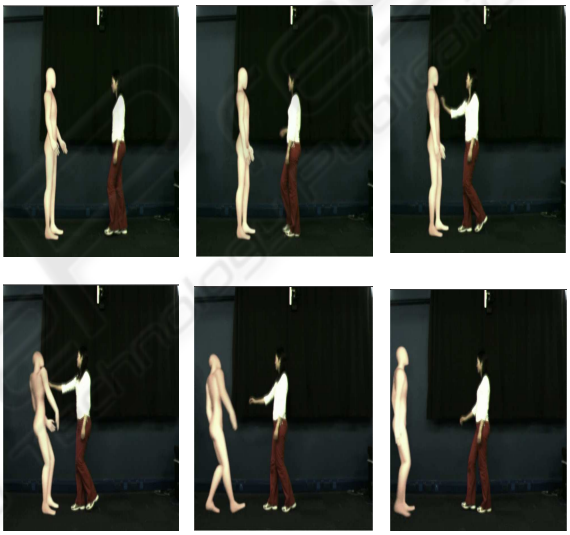

Figure 1: Interactions with virtual character (Pushing).

In the following sections we describe briefly the theory of the windowed Viterbi algorithm, and the process of generating interactive behaviours given a sequence of the poses for the first person (A).

3.1 The Windowed Viterbi Algorithm

The standard Viterbi algorithm (Rabiner, 1989) is able to find the single best state sequence for given observation sequence. It requires the full observation sequence before the processing starts, which fully prevents real-time processing. In order to achieve the benefits of the Viterbi without having to wait for the entire full state sequence, we use the windowed Viterbi algorithm (Pilu, 2004) and (Rybski and Veloso, 2005), that can obtain the best state sequence using only the part of full state sequence at each time.

In a real-time system, the windowed Viterbi algorithm will cause a small delay in the state estimate.

In our experiments, we capture our training data at 30 fps, and we choose the length of 10 as the window size. Thus the delay time is 0.33 second. It is not noticed in most of the generated motion. However, it is possible to avoid this delay by adjusting the time correspondences between interacting people in the training data.

At any given time t, we take a time slice of T samples and apply the Viterbi procedure to it. Of the sequence of states $Q = \{q_1, q_2, ...q_T\}$ defining the best path for said time slice we retain only the second state q_2.

In summary, the windowed Viterbi algorithm can be described in the following way (Pilu, 2004).

1. Select the initial state probability π for the N states.
$$\pi_{\frac{N+1}{2}} = 1, \pi_{i \neq \frac{N+1}{2}} = 0 \quad (3)$$
And select the length of the window T as well.
2. The best path $\{q_1, q_2, ..., q_T\}$ for the window T can be obtained using the standard Viterbi processing.
3. Retain the second state q_2 as the output at time t, and let
$$\pi_{q_2} = 1, \pi_{i \neq q_2} = 0 \quad (4)$$
for processing of the next window.
4. Repeat steps 2-3 (slice trellis forward) until the end state is reached.

3.2 Estimating Output Behaviours

Now it is possible, given as input a sequence of 3D poses of a person, to generate a corresponding sequence of poses for the virtual character using the methods described in the previous section. To achieve this goal, we defined a trellis data structure. Figure 2(a) illustrates an example of an initial trellis structure (Cosker et al., 2004). Each column of the trellis structure represents motion data of person B corresponding to the motion of our virtual character. We assign an error value (the Mahalanobis distance) to each element in each column, and then we work backward through the trellis to choose the motion data for the virtual character with the lowest cost for each time t.

Figure 2(b) shows how errors are calculated according to the trellis. C_t^j is the data vector for person B at time t, b_{input} is the new input signal (the tracking result from previous section) and b_a is the data vector for person A, which has the same location as the data vector for person B. For state 1 ($t = 1$) of the trellis structure, we only have the distance between the new input signal and person A data (denoted as D_4). For other states $t = 2, ..., M$ (M is the number of frames of the new input signal), we calculate the distances between C_t^j in a column at time t and C_{t-1}^j in a column at time $t-1$ (denoted as D_1, D_2 and D_3). Thus, error can be obtained as $E = D_4$ for state 1 and $E = (D_1 + D_2 + D_3) \times D_4$ for other state. When errors are calculated, we work backwards through the trellis and choose the motion data for the virtual character in each column with the lowest error at time t. Through this process, the interactive behaviour for virtual character is obtained.

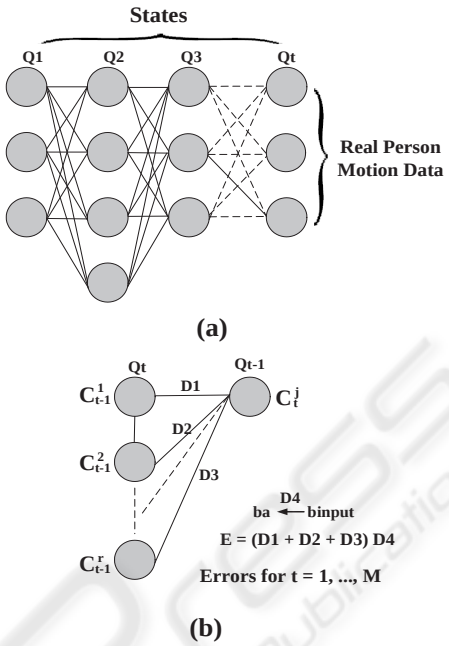

Figure 2: (a). Initial Trellis data structure for generating motion data. (b). Error Calculation for the generating motion data.

4 RESULTS

In the following experiments we generate interactive behaviours using the windowed Viterbi algorithm, and compare the performance of the standard Viterbi algorithm and the windowed Viterbi algorithm. We do so by calculating the Euclidean distance between original motion and the generated motion generated by both approaches. We also present the results of visual inspection of the generated motion by ten independent observers.

Selected frames from the generated video sequences are shown in Figures 3, 4 and 5.

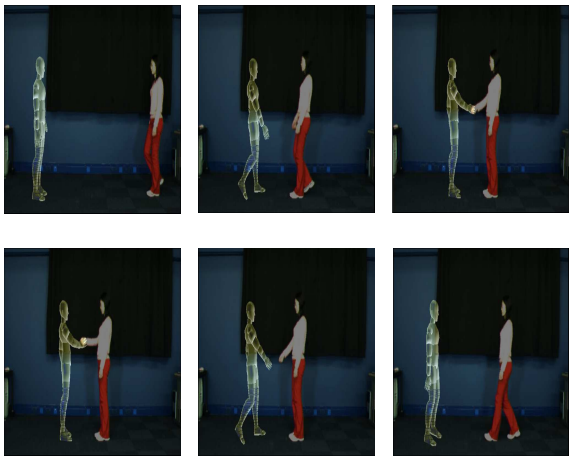

Figure 3: Interactions with virtual character (Handshake).

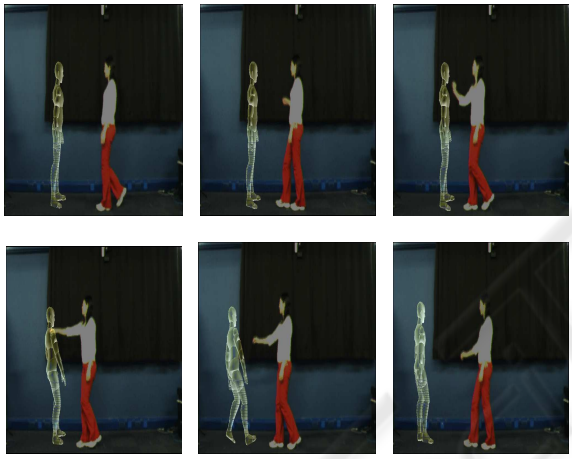

Figure 4: Interactions with virtual character (Pushing).

4.1 Assessing the Accuracy of the Generated Behaviour

We trained a dual HMM on 3D MoCap data representing three different types of motion of two people: handshaking, one person pulling another person and one person pushing another person, totalling 10 sequences around 1600 pose vectors. Then given a sequence of MoCap data extracted from video sequence for the first person (A), we generated interactive motion for the second person (B) using the windowed Viterbi algorithm (we choose the length of 10 as the window size that was determined experimentally.). Figures 6, 7 and 8 show the Euclidean distance between original motion and the generated motion when using the standard Viterbi algorithm and the windowed Viterbi algorithm on three different types of motion.

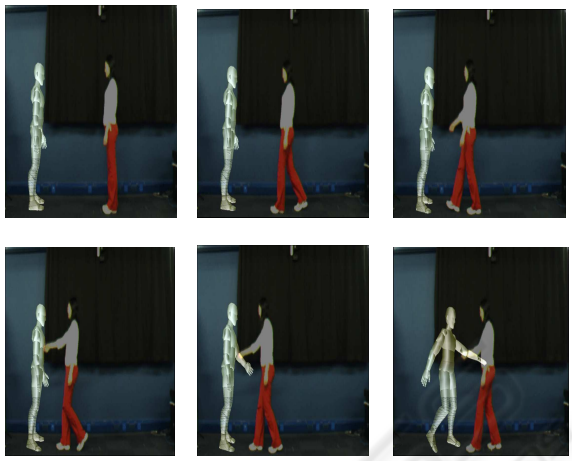

Figure 5: Interactions with virtual character (Pulling).

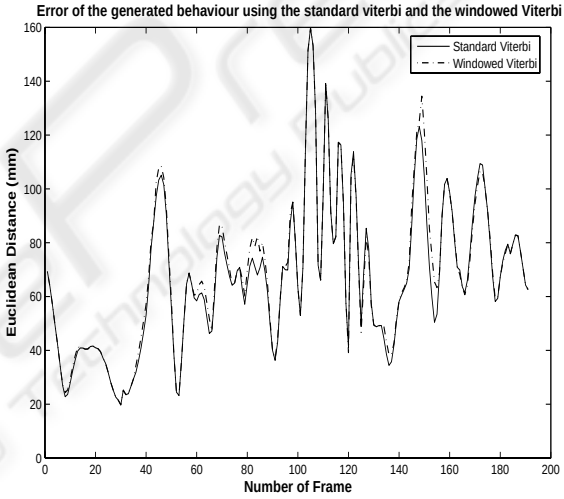

Figure 6: Error of the generated behaviour (in mm). Shaking hands behaviour. The error of generated behaviour using the standard Viterbi algorithm is shown in solid line, the dash line is for using the windowed Viterbi algorithm.

From these figures, we make the following conclusions.

1. In above figures, the error of generated behaviour using the standard Viterbi algorithm is shown in solid line. The dash line shows the error of generating behaviour using the windowed Viterbi algorithm. It is easy to see the errors of generating behaviours using both algorithms are similar. It means the generated motion using the windowed Viterbi algorithm is as good as the motion generated using the standard Viterbi, it also looks real and natural in all of the generated sequences.

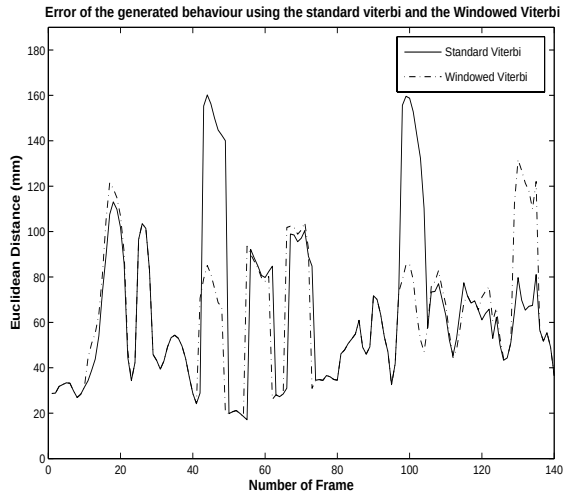

Figure 7: Error of the generated behaviour (in mm). Pulling behaviour. The error of generated behaviour using the standard Viterbi algorithm is shown in solid line, the dash line is for using the windowed Viterbi algorithm.

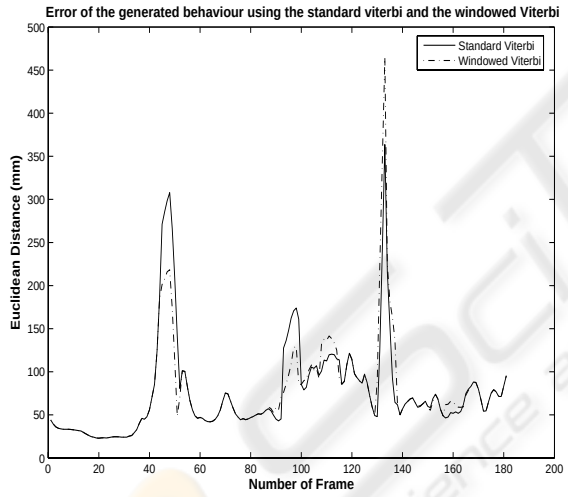

Figure 8: Error of the generated behaviour (in mm). Pushing behaviour. The error of generated behaviour using the standard Viterbi algorithm is shown in solid line, the dash line is for using the windowed Viterbi algorithm.

2. From the above three figures, it is possible to see that the generated motion using the windowed Viterbi algorithm has a small delay compared with the generated motion using the standard Viterbi algorithm. However, this delay is not noticeable in the generated motion sequences.
3. Although the error for the standard Viterbi algorithm is similar in most of the frames, in some frames the windowed Viterbi algorithm produces smaller error. After inspecting the generated motion sequences visually, we discovered that such frames corresponded to sudden changes in the motion. We conclude that the windowed Viterbi algorithm can cope with sudden changes in motion better as it uses less history of motion. In future, we are planning to take advantage of this observation by developing a generated algorithm with adaptive window size.

Table 1: Evaluation Results. The generated motions are generated using the windowed Viterbi algorithm.

Motion	*Comments*
Shaking hands	2/10 - generated person did not touch another at the beginning of shaking hands 2/10 - generated motion is wobbly
Pushing	2/10 - generated motion is floaty
Pulling	no comments

4.2 Visual Evaluation

Our goal was to evaluate how convincing the generated motion was from the point of view of an independent observer. For this purpose, we generated six test video sequences, three of which showed original MoCap data collected from two people performing handshake, pushing, and pulling actions. In the remaining three sequences the MoCap data of the second person was substituted with motion data generated using the windowed Viterbi algorithm described in this article. All videos showed only the motion of certain points on the body, not the whole body, which is acceptable for visual evaluation according to Johansson's (Johansson, 1973) experiments in psychology. Each of the above video sequences was shown to ten independent observers. The observers were told that the videos showed the motion data of two people performing some action and were asked to identify which action that was. They were also asked to comment if they noticed anything strange or unusual about the motion of the people. All ten subjects were able to identify the actions in all six videos correctly. The comments of the independent observer are shown in Table 1. Table 2 shows our previous evaluation results which the results motion were generated using the standard Viterbi algorithm.

From these results, we conclude that the generated behaviours using the windowed Viterbi algorithm looked very similar to the real behaviours as eight out of ten people did not notice anything unusual about the generated behaviours.

Table 2: Evaluation Results. The generated motions are generated using the standard Viterbi algorithm.

Motion	*Comments*
Shaking hands	5/10 - generated motion is floaty
Pushing	3/10 - original motion is wobbly 1/10 - generated person did not touch another
Pulling	no comments

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a new approach for generating interactive behaviours for virtual characters using the windowed Viterbi algorithm. We also compared the performance of the standard Viterbi algorithm and the windowed Viterbi algorithm.

To this end, we trained a dual HMM representing interactive behaviours of two people. Then we used a sequence of 3D poses of a person in conjunction with dual HMM to generate the responsive behaviour for a virtual character using the windowed Viterbi algorithm. From the analysis of the results and evaluation experiments, it is clear that the windowed Viterbi algorithm can generate very similar behaviours to the real behaviours. In addition, the windowed Viterbi method does not require the full observation sequence before the processing starts, thus it can be used in a real-time system.

In our new work, we take advantage of the windowed Viterbi algorithm in a new approach to modelling motion with a goal of obtaining better tracking and generating results.

REFERENCES

Cosker, D., Marshall, D., Rosin, P. L., and Hicks, Y. (2004). Speech driven facial animation using a hidden markov coarticulation model. *IEEE ICPR*, 1:314–321.

Deutscher, J., Blake, A., and Reid, I. (2000). Articulated Body Motion Capture by Annealed Particle Filtering. *IEEE CVPR Proceedings*, 2:126–133.

Jebara, T. and Pentland, A. (2002). Statistical Imitative Learning from Perceptual Data. *Proceedings of the 2nd International Conference on Development and Learning*.

Johansson, G. (1973). Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14(2):201–211.

Johnson, N., Galata, A., and Hogg, D. (1998). The acquisition and use of interaction behaviour model. *IEEE CVPR Proceedings*, pages 866–871.

Meredith, M. and Maddock, S. (2005). Adapting motion capture data using weighted real-time inverse kinematics. *Comput. Entertain.*, 3(1):5–5.

Pilu, M. (2004). Video stabilization as a variational problem and numerical solution with the viterbi method. *IEEE CVPR*, 1:625–630.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *IEEE Proceedings*, 77(2):257–286.

Rybski, P. E. and Veloso, M. M. (2005). Robust real-time human activity recognition from tracked face displacements. *Proceedings of the 12th Portuguese Conference on Artificial Intelligence*, pages 87–98.

Wen, G., Wang, Z., Xia, S., and Zhu, D. (2006). From motion capture data to character animation. *VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 165–168.

Zheng, Y., Hicks, Y., Cosker, D., Marshall, D., Mostaza, J. C., and Chambers, J. A. (2006). Virtual friend: Tracking and generating natural interactive behaviours in real video. *IEEE ICSP*.

Zordan, V. B. and Horst, N. C. V. D. (2003). Mapping optical motion capture data to skeletal motion using a physical model. pages 245–250.