Learning Gait Parameters for Locomotion in Virtual Reality Systems

Jingbo Zhao Robert S. Allison

Department of Electrical Engineering and Computer Science

York University, 4700 Keele Street, Toronto, ON M3J 1P3, Canada

{jingbo, allison}@cse.yorku.ca

Abstract. Mechanical repositioning is a locomotion technique that uses a locomotion interface to cancel the displacement of a user for walking on the spot. This technique is especially useful for virtual reality (VR) systems that use large-scale projective displays for visualization. In this paper, we present a machine learning approach for developing a mechanical repositioning technique based on a 1-D treadmill for interacting with a unique new large-scale projective display, named as the Wide-Field Immersive Stereoscopic Environment (WISE). We also assessed the usability of the proposed approach through a novel user study that asked participants to pursue a rolling ball at variable speed in a virtual scene. Our results show that participants differ in their ability to carry out the task. We provide an explanation for the variable performance of the participants based on the locomotion technique.

1 Introduction

Locomotion techniques in VR systems include: flying using a joystick [1], leaning [2], walking-in-place (WIP) [3,4,5,6,7,8], redirected walking (RDW) [9,10] and the numerous mechanical repositioning techniques mentioned by Nilsson et al. [11]. For example, flying using a joystick, originally developed for locomotion in CAVEs, enables a participant to locomote by actively operating a joystick with a hand. As locomotion techniques became more sophisticated, more motion cues were considered and included to improve presence in VR environments. Leaning, for instance, made a step forward in terms of full body control compared to flying by sensing upper body tilt angle and using it for locomotion. The first WIP interface "walking on spot" [3] used head motion to estimate one's walking speed and more recent WIP techniques [4,5,6,7,8] consider human biomechanics by using gait parameters to estimate one's walking speed. These WIP techniques allow a user to step in-place to locomote but only accommodate the motions of stepping not the forward stride of locomotion. The more sophisticated RDW technique makes it possible for a user to perform 'real' walking (e.g., take forward steps) by consistently re-orienting the user toward the centre of a constrained space while the user wears a head-mounted display (HMD).

This circumvents the difficulty that it is often impractical to build a 1:1 virtual scene, for example, when the simulated space exceeds the size of the physical space of the VR display or the VR tracking volume. Another approach to simulate walking while constraining physical motion is mechanical repositioning. These techniques normally use a locomotion interface to cancel the effects of the user's stride, keeping them in-place while performing walking motions. The technique is especially useful for virtual reality (VR) systems that use large-scale projected displays for visualization.

A large number of locomotion interfaces [12] have been developed for mechanical repositioning in VR systems. Examples of such devices include: 1-D treadmills [13], torus treadmills [14], virtual perambulators [14], foot platforms [15], pedaling devices [16] and spheres [17]. But 1-D treadmills are still of particular interest since they can be easily obtained. Many studies have focused on developing methods for automatic speed adaptation of 1-D treadmills and these methods can be generally divided into four categories [18]: inertial-force-based controllers, position-based controllers, physiology-based controllers and gait-parameter-based controllers. For example, von Zitzewitz et al. [19] developed a force-based controller by which a participant is required to wear a mechanical harness equipped with force sensors to measure the ground reaction force between their feet and the belt of a treadmill. The acceleration of the treadmill is calculated based on Newtonian mechanics using the measured ground reaction force. Souman et al. [20] developed a position-based controller in which the walking speed is estimated from the deviation of the position of the user from a reference position on the treadmill. The physiology-based controller by Su et al. [21] used heart rate in a biofeedback mode to control the speed and the slope of a treadmill to help a user maintain heart rate. Yoon et al. [22] developed a combined gait-parameter-based and position-based controller, which estimates a user's walking speed from the peaks of foot swing velocity and the position of a user on the treadmill. Generally, the goals of such controllers are to keep a user on the treadmill by estimating a user's intended walking speed and adapting the speed of the treadmill using the speed estimate. The related parameters that can be used for such estimation include ground reaction force, the user's position on treadmill, physiological factors or gait parameters, etc.

We were interested in developing a combined gait-parameter-based and position-based controller. The first motivation for this approach was that gait parameters contain explicit cues for estimating a user's intended walking speed. However, the proposed gait-parameter-based controller only estimates the intended walking speed of a user without considering the user's position on the treadmill. To account and control for position, we introduced a position-based controller that estimates a user's position on the treadmill from the user's captured foot position data. This position estimate is used to adjust the position of a user on the treadmill. Other parameters, such as the ground reaction force and the physiological factors could also be included to improve the current design for future research. Secondly, while recent WIP techniques have used gait parameters for walking speed estimation, to the authors' knowledge, use of gait parameters for mechanical repositioning techniques has not been thoroughly investigated and offer potential for improved interaction, which has been demonstrated by recent WIP techniques. Thirdly, this paper is novel in that it uses a machine learn-

ing approach to map the defined gait parameters (gait features) into a user's intended walking speed. Machine learning can potentially produce more effective and efficient gait-parameter-based controllers but these techniques have not been investigated for real-time treadmill locomotion interfaces. The work done by Park et al. [23] is related to our approach but is intended for indoor over-ground walking speed estimation. They used features that consisted of gravity components and DFT components extracted from the acceleration data collected by a handheld device and trained regularized kernel methods to estimate a user's walking speed. Their approach estimated a user's average walking speed from data sequences collected over several seconds (2.56 or 5.12 seconds for over-ground walking). We instead aimed to estimate a user's intended walking speed within 0.53 seconds for real-time control of a 1-D treadmill using low-latency gait features extracted from the foot motion data collected by an optical motion capture system. The current paper demonstrates that it is possible to achieve the goal of an interactive real-time locomotion interface using a machine learning approach. Finally, while the techniques described in the paper are general and could be applied to a variety of head-mounted display or projective VR setups, we developed and evaluated the system in the context of a novel immersive projected VR display. This unique new display is known as the Wide-Field Immersive Stereoscopic Environment (WISE, see below). It has been set up at York University to explore locomotion techniques and other interaction techniques with large-scale projective displays and investigate human locomotion and navigation behaviors in controlled VR environments.

In the current study, we present a machine learning approach for developing a locomotion technique based on a conventional 1-D treadmill for interacting with the display. In addition, previous studies [19], [20], [22] evaluated their speed estimation algorithms by asking users to walk at their preferred speed on the treadmill. These evaluations were performed in terms of the estimated walking speed pattern within one gait cycle [19], gains of control law [20] and the error of walking speed estimation [22]. In [19], a questionnaire was also used to let participants rate the performance of their proposed method. However, these studies did not engage users in an active VR environment during evaluation. To assess our locomotion technique in a more interactive and relevant way, we present a novel user study for assessing the utility of locomotion interfaces by asking a user to pursue a rolling ball in a virtual scene. While constrained to meet needs for experimental control, such a task is representative of a broad range of navigation tasks in VR where orientation and position relative to the environment are to be monitored and controlled such as coordinated movements with simulated agents or another user’s avatar.

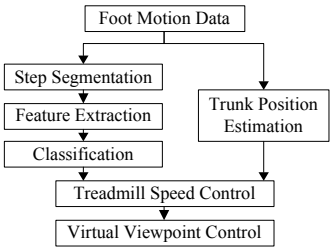

Fig. 1. Overview of the Proposed Approach.

2 Methods

We use a set of spatial-temporal features that can be extracted in real-time during actual locomotion with low latency. Inspired by Yoon et al. [22], we select the peaks of foot speed curves as one of the features for estimating intended walking speed. But contrary to their approach that models the foot speed curves as sinusoidal waves using the peaks as the amplitudes and that estimates the walking speed by calculating the average amplitude of the sinusoidal waves, our assumption is that when walking at a specific speed either over-ground or on the treadmill, the peaks of a foot speed curve should fall into a certain interval. Thus, conversely, if we can estimate the amplitude of the peaks of a foot speed curve, it should be possible to estimate the corresponding intended walking speed. The second feature that we use is the time intervals between consecutive peaks, since the time intervals are inversely proportional to step frequency. Thus, for every detected peak, the algorithm generates a 2-D feature vector. With enough data on foot speed collected from a number of participants, the problem of intended walking speed estimation can be turned into a classification task by learning the extracted features and using the learning result to estimate a user's intended walking speed. In our implementation, the classification is performed by a typical machine learning approach: the k-nearest neighbor (kNN) algorithm due to its simplicity for multi-class classification. Other machine learning algorithms [24] such as the Support Vector Machine, neural network algorithms and the K-Means etc. may also be used and further investigation is needed to compare their performance with the KNN-based approach. The result of the classification is the user's intended walking speed. The proposed technique handles two cases of locomotion behavior on the treadmill using a single workflow. First, when a user maintains current walking speed, the extracted 2-D feature vector will consistently fall into a specific interval. The estimated intended walking speed generated by the classification algorithm will be a constant speed value that maintains the speed of the treadmill. Second, when a user intends to change walking speed by increasing or decreasing foot swing speed, the extracted 2-D feature vector will jump from their previous interval to a new interval and the classification algorithm will generate an updated speed estimate that adapts the speed of the treadmill to accommodate the changed motion of the user. If the user maintains the new

walking speed, the user's locomotion behavior returns to the first case and the algorithm will generate constant speed estimates again. The second difference of our work compared with that of Yoon et al. [22] is that we only use foot position data to estimate a user's position on the treadmill, while their method directly captures the position of a user from an infrared (IR) marker attached on the waist. Our assumption is that the positions of two feet are symmetrical to the trunk in healthy gait. Thus the average of the positions of two feet can be considered as the estimate of the position of the trunk.

With the user's intended walking speed from classification and the position information estimated from a user's foot motion, the speed command for controlling the treadmill can be obtained. The actual running speed of the treadmill is queried periodically to update the speed of the virtual viewpoint in a scene. Figure 1 presents an overview of the proposed approach.

2.1 Hardware and Software of the VR System

The sensor for capturing foot motion is the NaturalPoint OptiTrack V120:Trio, which has three cameras with a resolution of 640 × 480 and a frame rate of 120 Hz. The sensor is an active IR camera that emits IR light, which is reflected back by IR markers and captured by the sensor. To capture human motion, markers are attached to body parts to be tracked. The positions of the markers are calculated by the system's supporting software (Motive 3.7) and broadcast through a network to be shared with other application software. The 1-D treadmill for the study is a commercial type LifeSpan TR5000-DT5. The size of the top surface of the treadmill belt was 51 cm wide × 142 cm long. The control of the treadmill is performed through Universal Asynchronous Receiver/Transmitter (UART) controllers by a host machine using a baudrate of 4800. The virtual environment is presented on the WISE, which is a curved and large-scale projective display designed by Christie. The images rendered on the display are cast and seamlessly merged by 8 overlapping projectors with blending and luminance calibration performed in hardware. Each projector is driven by a slave machine in a real-time rendering cluster. The rendering and the synchronization between the host machine and the slave machines are handled by the VR software Worldviz Vizard 5.0. The proposed approach was implemented using Python 2.7.

2.2 Data Collection and Buffering

To perform step segmentation and detect the peaks of foot speed, it is necessary to buffer a certain amount of data. We use three queues for buffering the timestamps of frames and the Z-positions (in depth) of the left foot and the right foot, respectively. The sizes of the queues are chosen to be 64, which gives an initial latency of about 533.3 ms for filling the buffers when the capture starts. The sizes of the queues ensure a complete step can be captured and that the initial latency for buffering is minimized.

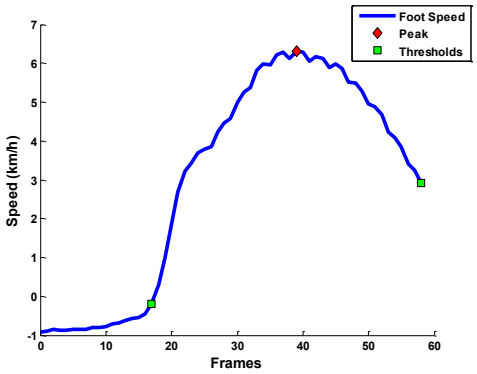

Fig. 2. An Example of Step Segmentation.

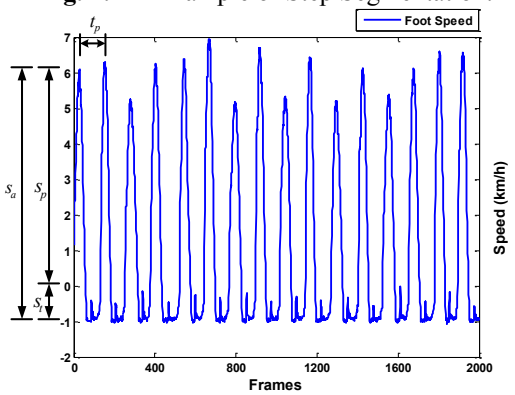

Fig. 3. Definition of the Gait Features.

2.3 Step Segmentation and Feature Extraction

We use the foot speed for step segmentation because foot speed generally shows better regularities in relation to the phases of a gait cycle than foot position does. Foot speed are calculated using the buffered position data and the calculated speeds of the left foot and the right foot are then merged together, since it is not necessary to distinguish whether a detected step is from the left foot or the right foot when estimating walking speed. A gait cycle has three important phases: the initial swing, the mid swing and the terminal swing [25]. In terms of foot speed, these phases correspond to the moments when a foot gains initial speed due to lifting; the foot reaches the maxi-

mum speed in mid-air; and the foot decelerates to a very slow speed before touching the ground, respectively.

Theoretically, successful step segmentation involves the detection of these three phases. Our dynamic thresholding technique works by first locating a step with a high threshold. From the thresholded data points of a step, the algorithm uses a gradient descent strategy to locate the data points for the initial swing and the terminal swing. The gradient descent algorithm stops whenever it reaches the minimum speed threshold for step segmentation or it detects that the gradient is ascending, indicating a different step is detected. Once the initial swing and the terminal swing are successfully located, we select the maximum speed value in the interval between these two data points and consider it as the data point for the mid swing. Use of the gradient strategy rather than using thresholding directly avoid incorrectly detecting small noisy peaks as steps.

However, during actual step segmentation, accurate detection of the terminal swing is not necessary. This is because the algorithm is able to detect the critical mid swing point before a user completes an entire step. Figure 2 shows the scenario, in which the initial and mid swing phases are detected while the terminal swing has not been buffered. The algorithm stops at the last data point which is sufficient to bracket the mid swing point. In this way, we can ensure timely detection of the mid swing and adaptation of the speed of the treadmill before the swing foot touches the treadmill belt. This minimizes the error between the user's trunk position and the reference position on the treadmill. The step segmentation algorithm is executed whenever a new data frame comes into respective buffers. Thus the same peak is repetitively detected. But each peak is used for adapting the speed of the treadmill only once.

The peak of the mid-swing during treadmill walking is not the same as the peak during over-ground walking. A major difference between over-ground walking and treadmill walking is that the peak value detected during treadmill walking is shifted downward (backwards motion) by the speed of the treadmill belt as the foot swing velocity is relative to the speed of the treadmill belt. Thus, whenever a peak is detected, it is necessary to add the current speed of the treadmill belt s_t to the detected peak s_p to yield the real amplitude of the peak s_a. This gives:

$$s_a = s_t + s_p \tag{1}$$

The second feature, the time intervals between consecutive peaks t_p, are calculated by subtracting the timestamps of adjacent detected peaks. The initial latency for measuring the time interval is approximately one and a half steps, as the algorithm needs to obtain the timestamps of two successive peaks. Figure 3 shows a foot speed curve when the treadmill runs at 1 km/h and gives an illustration of the defined gait features. These two features constitute a 2-D feature vector: $\mathrm{x} = (s_a,\ t_p)$, which can be used for training and testing in the classification step.

2.4 Classification

We divided the collected data from participants into halves: one for training and the other for testing. The class labels of extracted feature vectors y were their corresponding treadmill speeds and satisfy $\mathrm{y} \in$ {1km/h, 2km/h, 3km/h, 4km/h, 5km/h}.

Since the features were in different units: km/h for amplitudes of peaks s_a and seconds for time intervals t_p, to avoid biasing the result of classification due to the unit mismatch; we calculated the minimum and maximum values for both features in the training set and performed a linear transformation on all features in the training set to map them into the interval [0, 1] using min-max normalization. For the testing set, we used the minimum and maximum values calculated from the training set and mapped the features in the testing set into the interval [0, 1]. For classification, given a feature vector x, we wished to predict its class label y using a KD-tree version of kNN algorithm [26]. The Euclidean distance was used as the metric for similarity between feature vectors. During the testing phase, first a KD-tree was built using the training set. Then, for each feature vector x in the testing set, the algorithm found its 3 nearest neighbors. The mode of the class labels y of these neighbors was considered as the output: the estimated walking speed $\hat{s}_w$.

2.5 Control of Treadmill Speed and Virtual Viewpoint

The estimated walking speed $\hat{s}_w$ does not account for the user's position on the treadmill. To adjust the position of a user on a treadmill, we estimate a user's trunk position on the treadmill through tracked foot position data. Specifically, we calculate the average foot positions $\overline{p}_f$ and consider it as the estimated position of the trunk $\hat{p}_t$:

$$\hat{p}_t = \overline{p}_f = \frac{\sum_{i=1}^{w_{lf}} p_{lf}^i + \sum_{i=1}^{w_{rf}} p_{rf}^i}{w_{lf} + w_{rf}} \tag{2}$$

where w_{lf} and w_{rf} denote the sizes of the queues for buffering the left foot position p_{lf} and the right foot position p_{rf}. In our case, both values equal 64.
The speed value s_t' given to the treadmill for control is defined as:

$$s_t' = (\hat{p}_t - p_{ref}) \cdot k_p + \hat{s}_w \tag{3}$$

where p_{ref} is a reference position on the treadmill relative to the IR sensor and k_p is the positional gain. The positional term $(\hat{p}_t - p_{ref}) \cdot k_p$ compensates for small errors between the position of the trunk and the reference position on the treadmill while the gait-parameter term $\hat{s}_w$ accommodates abrupt speed changes.

The virtual viewpoint in a scene is controlled by querying the current speed of the treadmill s_t and using it to update the speed of the viewpoint. The frequency for querying speed is 5 Hz.

3 Experiment 1

3.1 Introduction

The purpose of Experiment 1 was to collect the data of foot motion to build the training and testing sets for the kNN algorithm. The trained classifier resulting from Experiment 1 was also used for the locomotion interface evaluated in the user study in Experiment 2. Eleven people (3 males, 8 females, age: 24.8 ± 3.1) participated in the study. Informed consent was obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University.

3.2 Procedure

In Experiment 1, the IR sensor was placed approximately 1 m in front of the treadmill and participants attached an IR marker on the toe cap of each shoe and a third IR marker on the trunk. Each participant performed two sessions of treadmill walking. In each session, the treadmill started 5 seconds after the experiment began.

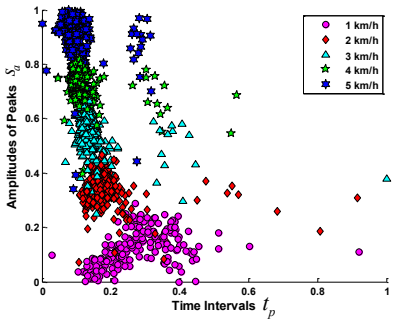

Fig. 4. Features Extracted from the Training Set.

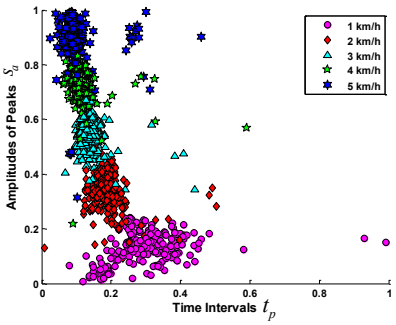

Fig. 5. Features Extracted from the Testing Set.

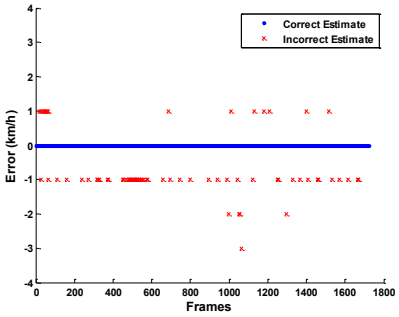

Fig. 6. Error Pattern of Misclassification

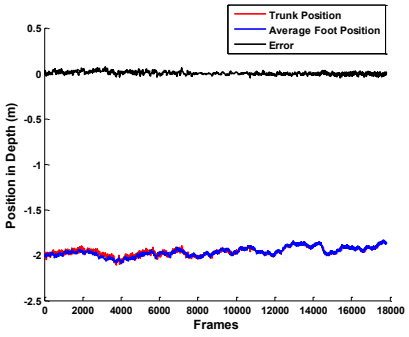

Fig. 7. Trunk Position versus Average Foot Position.

Data collection was started at the same time. The speed of treadmill started at 1 km/h, incremented by 1 km/h every 30 seconds until the speed reached 5 km/h, and stayed at 5 km/h for another 30 seconds. When the data collection finished, the treadmill

slowed down to 1 km/h and stopped in 5 seconds. Visual instructions were displayed on the WISE to give prompts to participants.

The data of 3 participants were not recorded properly due to the IR marker failure (subsequently corrected for Experiment 2 by using improved Velcro fasteners). Thus, we used the data of the first sessions of the rest of the 8 participants (3 males, 5 females, age: 23.9 ± 1.9) as the training set and the data of the second sessions of the 8 participants as the testing set. To test the efficiency of the proposed algorithm, the step segmentation and feature extraction algorithms were applied on both sets. The extracted features vectors for the training set and the testing set are plotted in Figure 4 and Figure 5, respectively. The distinct clustering of feature vectors indicates that it is possible to classify them using the kNN algorithm. During testing, a KD-Tree was built by using the extracted feature vectors from the training set. The extracted feature vectors were classified using the KD-Tree to obtain the intended walking speed estimates $\hat{s}_w$. The ground truth label of a feature vector was the true speed of the treadmill at which the foot positions were captured. We compared the result of the classification with its ground truth label to determine whether the result was correct or not.

3.3 Results

We evaluated the kNN-based algorithm in terms of the multi-class recall, the multi-class precision and the average accuracy for all 5 classes [27], as shown in Table 1 and the evaluations were performed by using the peaks of foot velocity s_a alone and using both the peaks of foot velocity s_a and the time interval t_p. For all measures, using s_a and t_p both generated better results than using s_a alone. The result demonstrates that using both features increased the classification performance.

To visualize the error pattern of misclassification, we plotted the estimated speed values for all participants, as shown in Figure 6. From it, we can observe that the proposed algorithm tended to underestimate the actual walking speed, since the number of incorrect estimates at the error level of -1 km/h was obviously larger than that of 1 km/h. In the worst case, the classification produced an estimate 3 km/h slower than

Table 1. Statistics of the Classification Performance.

Feature	Multi-class Precision	Multi-class Recall	Average Accuracy
s_a	93.4 %	94.2 %	97.6 %
$s_a + t_p$	94.1 %	94.6 %	97.8 %

Table 2. The Mean Value and the Standard Deviation of the Error between the Average Foot Position and the Captured Trunk Position.

Participant	1	2	3	4	5	6	7	8
Mean (m)	0.03	0.001	0.007	0.004	-0.004	-0.01	-0.007	-0.008
Std (m)	0.033	0.03	0.028	0.025	0.021	0.028	0.032	0.046

the actual walking speed.

To evaluate whether the average foot position is a reliable estimate of the trunk position, we calculated the mean value and standard deviation of the error between the average foot position and the captured trunk position from the testing set (Table 2). In the worst case as shown by participant 1, the mean value was 0.03 m and standard deviation was 0.033 m. Figure 7 also presents exemplar curves and shows that the average foot position closely followed the trunk position and the error between these two curves was minimal. The result demonstrates that the average foot position can be used to estimate the trunk position.

4 Experiment 2

4.1 Introduction

The purpose of Experiment 2 was to evaluate the usability of the locomotion interface through a user study using the WISE (Figure 8). Ten people (8 male, 2 females, age: 26.2 ± 4.4) participated in the study and none had participated in Experiment 1. Informed consent was obtained from all participants in accordance with a protocol approved by the Human Participants Review Subcommittee at York University.

Fig. 8. The WISE Running Experiment 2.

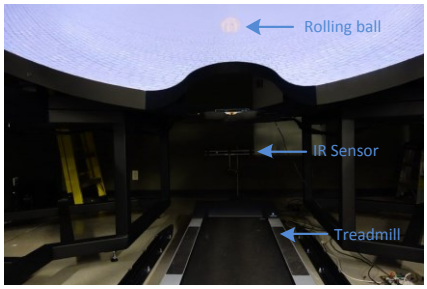

Fig. 9. The Detailed Setup for Experiment 2.

4.2 Procedure

In Experiment 2, the IR sensor was also placed approximately 1 m in front of the treadmill (Figure 9) and participants were asked to attach two IR markers on the toe caps of their shoes. We designed a virtual scene in which the participants were asked to pursue a rolling ball using the locomotion interface. The speed of the ball was unpredictably varied between 1 km/h, 2 km/h and 3 km/h with changes in speed occurring at predefined key frames. The acceleration of the ball was set to 0.176 m/s^2, which matched the acceleration of the treadmill. The virtual scene was rendered on

the WISE in monocular mode with a frame rate of 60 Hz. A KD-Tree was built by using the extracted feature vectors from the training set in Experiment 1 for speed estimation. The positional gain k_p in equation (3) was set to 3 and the reference position p_{ref} was set to 1.8 m empirically. For safety reasons, we limited the range of speed command to 0.2 km/h minimum and 3.5 km/h maximum. The participants' task was to try to maintain the distance between the virtual viewpoint and the ball to the initial distance of 5 m. To provide a visual cue for participants to judge the speed and the distance of the ball, the color of the ball was updated on a per-frame basis such that the color would gradually become red if a participant was too far away from it or blue if too close. Since the length of the treadmill belt was limited and it was impractical to increase walking speed by increasing step length, the participants were instructed to increase their step frequency and foot swing speed if they intend to accelerate and conversely decrease their step frequency and foot swing speed for deceleration. Each participant was asked to perform 4 sessions of walking using the same scene and each session lasted 230 seconds. To ensure the experiment was conducted in the same condition for all participants and all sessions, the start and the stop of the treadmill and the ball were synchronized and controlled by the experiment application software. When the researcher started the experiment, a countdown timer was shown on the display and the data collection began at the same time. The treadmill and the ball started to move simultaneously after a 10-second count. 3 seconds prior to the end of the experiment, another countdown timer was shown on the display and counted 3 seconds before the experiment stopped. The data collection was immediately

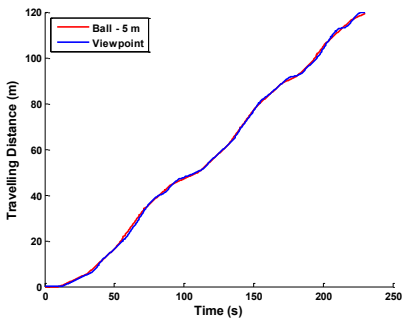

Fig. 10. An Example of Trajectories of the Ball and the Viewpoint.

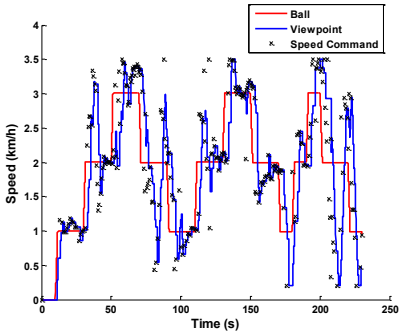

Fig. 11. An Example of Speed Curves of the Ball, the Viewpoint and the Speed Commands.

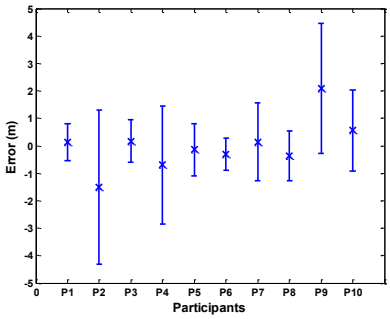

Fig. 12. Performance of the Participants.

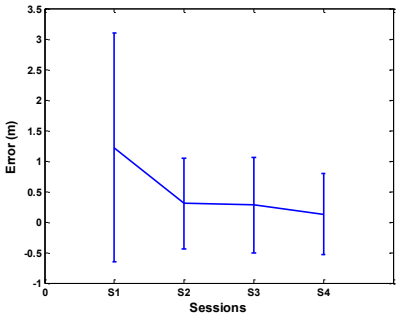

Fig. 13. The Learning Effect Shown by a Participant.

stopped at 230 seconds and the ball and the treadmill were stopped subsequently. Given such experimental conditions, theoretically, a participant should be able to control the viewpoint through the locomotion interface such that the curve of the position of the viewpoint perfectly matched the trajectory of the rolling ball while maintaining the 5 m reference distance.

4.3 Results

Figure 10 shows an example series of the trajectories of the ball and the virtual viewpoint. The 5 m distance offset was subtracted from the trajectory of the ball point-wise to show that the participant was able to maintain the 5 m distance and matched the trajectory of the viewpoint to that of the rolling ball. Figure 11 shows the speed changes of the ball and the viewpoint and the speed commands issued by the controller. We can observe that in some cases the speed of the viewpoint did not reach the value given by a speed command due to the modest maximum acceleration of the treadmill. The performance of the participants and the locomotion interface was evaluated in terms of the error of the distance between the virtual viewpoint and the ball with respect to the reference distance 5 m. This point-wise error was calculated by the following equation:

$$\varepsilon = p_b - p_v - 5 \tag{4}$$

where p_b denotes the position of the ball and p_v denotes the position of the viewpoint. We then calculated the mean value and standard deviation of the point-wise error from the 4th session of all participants. The results for all participants are shown in Figure 12 using an error plot, which shows that participants P1, P3, P5, P6 and P8 performed relatively better than the other 5 participants. The mean value of the error shows on average whether a participant is leading ahead or falling behind of the 5 m distance to the ball and the standard deviation reflects the interval in which the participant oscillates while maintaining the 5 m distance.

Several factors may lead to the error. These include a participant's attention, the ability to adapt to self-paced treadmill walking and control of gait, the accuracy of the speed estimation algorithm and the hardware limitations of the treadmill. A partici-

pant's attention on the ball is critical for the experiment. If attention is not focused on the ball and the task, then the distance and the speed of the ball cannot be judged and controlled. In addition, walking on a self-paced treadmill is different than walking over-ground. Thus participants have to quickly adapt themselves to the walking condition and learn to control the walking speed through the changing of foot swing speed and step frequency instead of step length. For the locomotion interface, although our algorithm shows that it is possible to estimate a participant's intended walking speed using the data collected from other participants, the accuracy of the speed estimation algorithm may be improved if we only use the data collected from a specific participant to classify that participant's data during the user study. A major hardware limitation of the treadmill is its low acceleration, which means it takes time for the treadmill reach the speed command sent by the host computer. Several participants reported the lack of responsiveness due to the issue.

The result of a few participants showed a clear learning effect. From Figure 13, we can observe that for a specific participant, the mean value and standard deviation gradually improved (reduced) as successive experimental sessions were performed.

5 Conclusions

In this paper, we presented a machine learning based approach for implementing a locomotion technique based on a conventional 1-D treadmill. We used the locomotion interface for interacting with the WISE and conducted experiments evaluating its usability through a novel user study. For future research, we will implement a turning strategy [28] for the locomotion interface. The improved locomotion interface can be used for the study of target interception in VR environments. Meanwhile, the speed estimation algorithm may also be used with other types of sensors, such as inertial measurement units, for estimating walking speed either over-ground or on treadmill. Finally, an empirical comparison of the proposed approach with other classic approaches of locomotion based on 1-D treadmills also is needed.

6 References

1. Robinett, W. and Holloway, R. Implementation of flying, scaling and grabbing in virtual worlds. In Proc. Symp. on I3D '92. ACM, 189-192 (1992)
2. Laviola, J. J. JR., Feliz, D. A., Keefe, D. F. and Zeleznik, R. C. 2001. Hands-free multi-scale navigation in virtual environments. In Proc. Symp. on I3D '01. ACM, 9-15 (2001)
3. Slater, M., Steed, A. and Usoh, M. The virtual treadmill: a naturalistic metaphor for navigation in immersive virtual environments. In Selected papers of EGVE '95, 135-148 (1995)
4. Templeman, J., Denbrook, P. and Sibert, L., 1999. Virtual Locomotion: Walking in Place through Virtual Environments, in Presence, 8, 6, 598-617 (1999)
5. Yan, L., Allison, R.S. and Rushton, S.K. New simple virtual walking method - walking on the Spot. In Elec. Proc. 8th Annu. IPT Symp (2004)
6. Feasel, J., Whitton, M.C. and Wendt, J.D. 2008. LLCM-WIP: Low-Latency, Continuous-Motion Walking-in-Place. IEEE Symp. on 3D UI, 97-104 (2008)

7. Williams, B., Bailey, S., Narasimham, G., Li, M., and Bodenheimer, B. Evaluation of walking in place on a Wii balance board to explore a virtual environment. In ACM Trans. on Appl. Percept. 8, 3 (2011)
8. Wendt, J. D., Whitton, M. C. and Brooks, F. P. 2010. GUD WIP: Gait-Understanding-Driven Walking-In-Place, In Proc. IEEE VR, 51-58 (2010)
9. Razzaque, S., Kohn, Z., and Whitton, M.C. 2001. Redirected walking (short paper presentation). In Eurographics (2001)
10. Razzaque, S., Swapp, D., Slater, M., Whitton M. C., and Steed, A. 2002. Redirected walking in place. In Proc. EGVE '02 (2002)
11. Nilsson, N.C., Serafin, S., Laursen, M.H., Pedersen, K.S., Sikstrom, E. and Nordahl, R. 2013. Tapping-In-Place: Increasing the naturalness of immersive walking-in-place locomotion through novel gestural input. IEEE Symp. on 3D UI, 31-38 (2013)
12. Hollerbach, J. M. Locomotion interfaces. In Handbook of Virtual Environments: Design, Implementation, and Applications. Lawrence Erlbaum Assoc., Inc. 239-254 (2002)
13. Souman, J. L., Robuffo Giordano, P., Schwaiger, M., Frissen, I., Thummel, T., Ulbrich, H., De Luca, A, Bulthoff, H. H. and Ernst, M. O. Cyberwalk: Enabling unconstrained omnidirectional walking through virtual environments, In ACM Trans. on Appl. Percept. 8, 4 (2001)
14. Iwata, H. Walking about virtual environments on an infinite floor, In Proc. IEEE VR. 286-293 (1999)
15. Iwata, H., Yano, H. and Nakaizumi, F. Gait Master: a versatile locomotion interface for uneven virtual terrain. In Proc. IEEE VR. 131-137 (2001)
16. Allison, R. S., Harris, L. R., Jenkin, M., Pintilie G., Redlick, F. and Zikovitz, D. C. First steps with a rideable computer. In Proc. IEEE VR. 169-175 (2000)
17. Medina, E., Fruland, R., and Weghorst, S. VIRTUSPHERE: Walking in a human size VR "hamster ball". In Proc. Hum. Fact. Ergon. Soc. Annu. Meet. 52, 27, 2102-2106 (2008)
18. Park, H. J., Lee, H. J., Kang, T. H. and Moon, J. I. Study on automatic speed adaptation treadmills. 15th ICCAS. 1898-1900 (2015)
19. Von Zitzewitz, J., Bernhardt, M. and Riener, R. A Novel Method for Automatic Treadmill Speed Adaptation. In IEEE Trans. on Neural Syst. and Rehabil. Eng. 15, 3, 401-409 (2007)
20. Souman, J. L., Giordano, P. R., Frissen, I., De Luca, A., and Ernst, M. O. Making virtual walking real: Perceptual evaluation of a new treadmill control algorithm. In ACM Trans. on Appl. Percept. 7, 2 (2010)
21. Su, S.W., Wang, L., Celler, B.G. and Savkin, A.V. Heart rate control during treadmill exercise. 27th Annu. Int'l. Conf. in EMBS. 2471-2474 (2005)
22. Yoon, J., Park, H. S. and Damiano, D. L. A novel walking speed estimation scheme and its application to treadmill control for gait rehabilitation. In Jrnl. Neuroeng. and Rehab. 9, 1, 1 (2012)
23. Park, J., Patel, A., Curtis, D., Teller, S. and Ledlie, J. 2012. Online pose classification and walking speed estimation using handheld devices. In Proc. ACM Conf. UbiComp. 113-122 (2012)
24. Marsland, S. Machine learning: an algorithmic perspective, Chapman & Hall/CRC. (2009)
25. Rose, J. and Gamble, J. G. Human walking. Baltimore: Williams & Wilkins (1994)
26. Maneewongvatana, S. and Mount, D. M. On the Efficiency of Nearest Neighbor Searching with Data Clustered in Lower Dimensions. In Proc. ICCS '01, 842-851 (2001)
27. Sokolova, M. and Lapalme, G. 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manage. 45, 4, 427-437 (2009)
28. Vijayakar, A. and Hollerbach, J.M. 2002. A proportional control strategy for realistic turning on linear treadmills. In Proc. 10th Symp. on HAPTICS, 231-238 (2002)