第 28 回日本バーチャルリアリティ学会大会論文集（2023 年 9 月）

物理キャラクタの歩行スタイルとバランスの設計環境

Design Environment for Different Style Locomotion and Balance of Physically Simulated Characters

藍愷 [1)]，長谷川晶一 [2)]
Kai LAN, Shoichi HASEGAWA

1,2) 東京工業大学 工学院（〒 226-8503 神奈川県横浜市緑区長津田町 4259,lankai1992@hotmail.com）

概要: この研究ではリアルタイムで物理キャラクターの歩行スタイルを創作できるデザイン環境を提案した。このデザイン環境は歩行バランス制御の方法と動作編集するための UI で構成される。ユーザーは、各歩行状態におけるキーフレームのポーズを 3D 視点で編集することができる。キーフレームには歩行周期において最も重要な 4 つの状態で構成される標準歩行スタイルを用意した。ユーザーはこの 4 つの状態に基づいて新しいスタイルを作成できる。

キーワード： 物理キャラクタ、二足歩行、ユーザーインターフェース

1. Introduction

In the past two decades, research on the locomotion control policy of physics-simulated characters has been a popular topic. Many locomotion control methods for physically simulated characters have been proposed, but the use of these methods requires professional programming and scripting knowledge. It is still difficult for social VR users and creators to use these methods to create locomotion motions, such as different styles of walking motion. According to the survey, 39.4% of social VR users will make their own original characters for use in the virtual environment. These users also need to create their own character's walking motions. However, currently, there is still a lack of a design environment that can intuitively create physically simulated character walking motions in real-time. For these reasons, we present a design environment to allow creators to create different types of physics-based walking characters easily and enable creators to modify the walking style of the characters in real-time.

2. Related Work

Research on physics-simulated characters has been going on for many years. The latest survey[1] offers a complete examination of the state of research on physics-simulated character animation and control techniques. Here we present some of the research that inspired our study. We classify the studies by the way in which the walking styles were generated.

Key pose editing: Divide a walking cycle into several key poses. Edit the key poses in the walk cycle as edit the keyframes of an animation, which is typically used in physics characters driven with joint torques. One of the most famous and has become the basis of many works is the SIMBICON, conducted by Yin et al[2]. They introduce a generic framework for biped locomotion control based on pose control graphs. The default walking motion is represented by four states. Their framework allows biped locomotion control with various gaits and styles. Coros et al[3] extended the framework of SIMBICON. They use virtual forces to compensate for gravity, which allows low-gain PD control and in turn, helps allow for natural, highly compliant motion.

To allow creators or animators to create different styles of walking motion, both [2] and [3] implemented a simple user interface. The UI of [2] allows users to choose the joint and edit the target angle of the joint in each state of the pose-control graph. However, such an editing method is indirect perceptual. A small change to the target angle of a joint will affect the entire walking style, and it is difficult for the user to predict the consequences of the operation. Also, small manipulations of some joints' target angles can create an unbalanced walking style. The UI of [3] improved some issues with a 2D target pose graph. However, their UI can only edit movements in two planes and cannot edit the walking style in which the character's joints rotate in the direction of the z-axis (the axis facing directly above). When editing certain styles, such as parade steps, the feet cannot be fully extended, and it isn't easy to maintain balance. The model for physics simu-

lations of the walking motion and displaying the target posture is not separated, so in order to view the target posture, the simulation must be started at the same time. When the physics character loses balance and falls down during walking, it is not possible to view the desired motion. Our work in the user interface is most close to those two research.

Automatic generation of walking motions: Muscle-based physics characters could generate walking motions automatically through optimization based on biological constraints. [4][5] presented a biologically-motivated control policy that can be used to automatically generate 3D creature-like walking and running controllers of different speeds. However, the optimization process takes about 10 hours. It's not fit for real-time editing.

Using reference motion: Take the reference motion as a sequence of states for the character over time. Then have the actuator generate a force to track the reference motion's trajectory[2]. In recent studies, the optimization techniques used to develop controllers are based on deep learning[6][7]. But the training also takes a long time.

3. System Overview

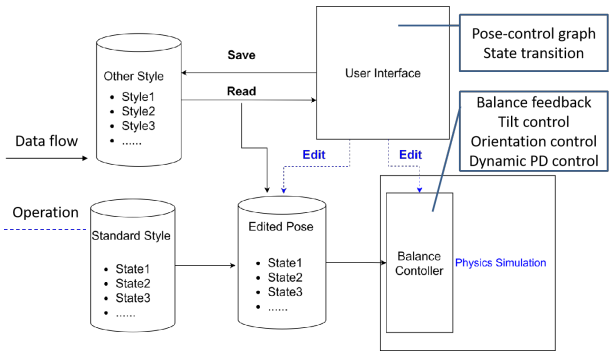


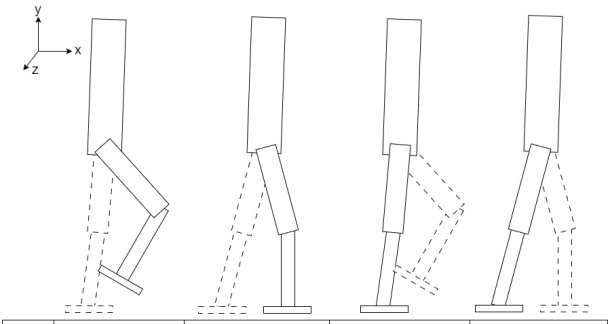
図 1: System Overview

The proposed system consists of two main parts. The **control strategy** and an **user interface**. The control strategy integrated a pose-control graph, state transition, balance control, tilt control, dynamic PD control, and orientation control. Controllers can be adjusted using a small number of parameters. The user interface allows the user to adjust the target pose in a third-person 3D view. At the same time, it allows the user to observe stationary walking motion and simulated dynamic walking motion. As shown in Figure 1, We introduce a list *Edited Pose*, which is used to store temporary target poses. It stores the target pose that the character actually executes in each state. At the beginning of the program, the edited pose list first stores the pose of the standard style. In each walking cycle, the character executes the motion of the next state after the end of the last state. When the user changes the target pose of a state through the user interface, the target pose of this state in the edited pose list will be updated, and the character will use the new target pose when its state reach this updated state. This way, users can see changes in walking style while editing.

4. Control Strategy

4.1 Pose-control Graph

In our strategy, the user specifies the number of states of a walking cycle and sets the target foot position, direction, and execution time of each state, and then a cycle of walking motion can be obtained. The stance foot is switched when the execution time is finished, or the swing foot contacts the floor. We prepared a default standard walk style consisting of 4 states. Users can create other walk styles based on the standard style. As shown in Figure 2, in this standard style, state 0 means the swing pose for the right foot and the support pose for the left foot. The rest may be deduced by analogy. If more states are needed, the user must specify which states correspond to lift, land, support, and back kick. These states will be used for balance control in the balance controller.



State	0	1	2	3
Right	Lift	Land	Support	Back Kick
Left	Support	Back Kick	Lift	Land

図 2: The pose of each foot in each state and the function it performs

4.2 Foot Placement Strategy

To make the physically simulated character stably walk on the ground while keeping our goal of real-time adjustment, we need to limit the hyperparameter numbers and make them as few as possible. For such reason, we constructed a simple and intuitive foot placement strategy that consists of **Balance Feedback**,**Tilt Control**, and **Orientation Control**. Since the pose-control graph only decides the target pose of the character in each state, the target pose may not fit the ground. Four lasers are used at the front and rear of the character to detect the

height and slope angle of the ground. The detected tilt angle of the upper body decides the real landing position of the swing foot. The detected slope angle of the ground, the tilt angle, and the target forward orientation of the character decide the real landing orientation of the swing foot.

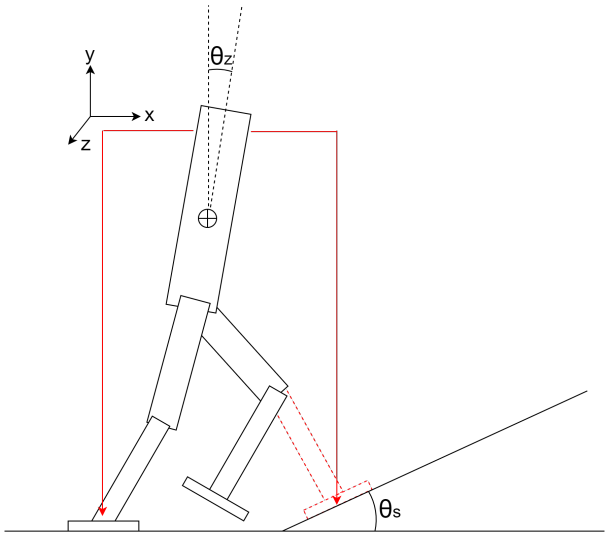


図 3: Character in sagittal plane

4.2.1 Balance Feedback

The basic idea of balance feedback is to obtain the inclination angle of the character's body when the character is walking, and then control the balance according to this angle. If the character leans forward in the forward direction, the swing foot's landing point will be farther away, or in other words, take a larger step. In Figure 3, $[\theta_x, \theta_y, \theta_z]^T$ is the inclination angle of the character's upper body.

4.2.2 Orientation Control and Tilt Control

Due to the conservation of angular momentum, when the physically simulated character takes a quick step, it will cause the body to rotate. To keep the character walking in the right direction, we adjust the direction of the next landing foot or push the character in the right direction through the support foot. This is just like what humans do when they walk.

4.3 Dynamic PD Control

PD(proportional-derivative) control is used to control the torque of the joint. In our system, when we need to move the joint quickly to the required angle, we set a large spring and damping gain, but this will make the joint stiff. When the foot touches the ground, because the foot and the ground can not be completely fit, the stiff ball joint of the ankle to perform the action to the target angle will make the character suddenly jump up. To solve this problem, we dynamically control the spring and damping gain of the ball joint of the ankle. When the contact force between the ground and the swing foot is less than 200N, the spring and damping gain are set to small values. When the contact force is greater than 200N, which means the swing foot becomes the support foot, the spring and damping gain are set to large values.

5. User Interface

From the previous research, the walking styles may be affected by the stride length, width, and walking speed. We also consider that the foot posture in the swing state will affect the walking style. Our user interface should enable users to edit those parameters, and thus we use an edit window to edit the target pose of each state. Users may need to refer to the target poses of the states before and after the state is edited. To allow users to see the characters' poses in other states intuitively, we use a round UI (The images representing the poses of each state of the character form a circle) to show the poses in all states. Lastly, a window is used to show the whole walking motion and physics simulated result. The constructed user interface is shown in Figure 4.

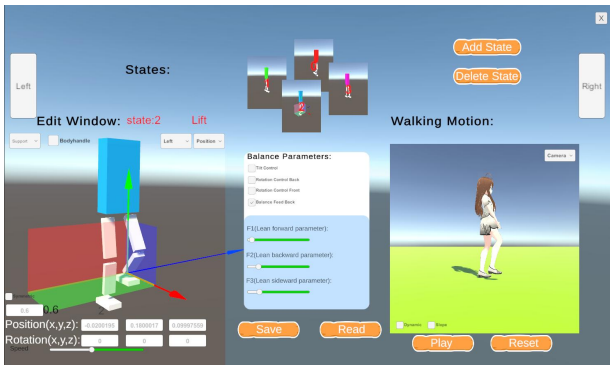


図 4: User interface of the proposed system

6. Implementation

The system was implemented in Unity. Physics engine Springhead is used for Physics simulation. The character models for editing and simulation are the same, consisting of an upper body and two legs. The hip and ankle joints are made of ball joints. The knee joints are made of hinge joints. We set the weight of the character to 60kg. The IK calculation of the character and the interpolation of intermediate poses for different target poses use the functions of VGentEditor. When an end-effector needs to go to another pose on its way to one pose, the intermediate pose is interpolated by the minimum-jerk model. In this way, a natural character movement is achieved.

As mentioned earlier, the user edits the walking motion by simply specifying the pose of the feet relative to the body for each state. Once the target pose is obtained, it is converted to the target angle of the joint using IK. The Jacobian Inverse Kinematics is used for the calculation of the target angle.

7. Evaluation

We conducted experiments to evaluate the robustness of the balance controller of our proposed UI. Considering that the virtual environment shared by social VR users is generally a room with a flat floor, the terrain rarely changes. We test the stability of the physical character walking on flat ground, encountering a slope, and being hit by a flying ball. In the experiment, when the character walking in the standard walking style is hit from the front and the back by a ball weighing 3kg with a speed of 6.7m/s, the character can recover from impact and maintain a stable walk, but when the body is hit from the side by this ball, the character may not easy to maintain stability. Using a combination of all controllers, the character can climb slopes with a maximum inclination angle of 3 degrees. Even if the character's feet slip relative to the ground due to insufficient friction. The character can still maintain balance on the slope.

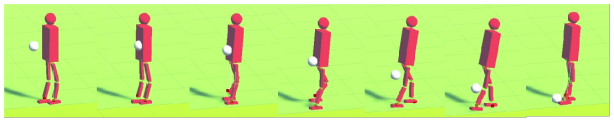


図 5: The ball hits the character from the back of the character

To test if our proposed UI could do trial and error in real-time, we asked 10 participants to create three kinds of walking styles (Cowboy, Kicking, and Parade Step) using our UI and see if they could complete it in a short time. The average editing time for cowboy style is 7.2 minutes, for parade style is 5.0 minutes, and for kicking style is 8.0 minutes. We also asked participants to try to complete the three walking styles using the UI from the two previous studies, but no participant was able to complete any of those walking styles within 15 minutes using their UI.

8. Conclusion and Future Work

This research proposed a design environment that allows creators to create different walking styles of physically simulated characters in a few minutes with direct perception. The creators could do trial and error and observe the effect on the character's motion in real-time.

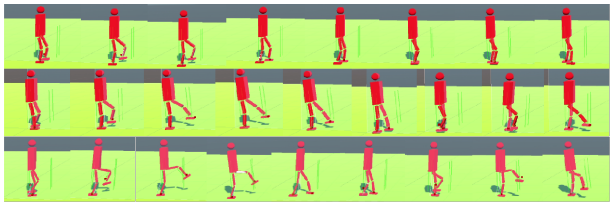


図 6: Created walk style using our system. Cowboy style (top), Parade style (middle), Kicking style (bottom)

The experiments also show the robustness of the system. However, our experiments only show balance stability on some occasions, because the balance control of the character is based only on the balance feedback system. The balance feedback is parameter sensitive. Small changes in parameters may cause unbalanced walking. The following improvements can be made in future research. 1. Comparing our research with the SUS(System Usability Scale) scores of previous studies to judge whether our system is creator-friendly. 2. Add more methods for balance control, such as methods based on the nonlinear inverted pendulum model to calculate the next landing position. 3. Combining this system with other physics-based character motion editors to allow users to edit the character's full body movements and use them in VR.

参考文献

[1] Geijtenbeek: Interactive character animation using simulated physics: A state-of-the-art review, Wiley Online Library, vol.31, No.8, pp.2492–2515, 2012.

[2] Yin, KangKang: SIMBICON, vol.26, No.3, pp.105–es, 2007.

[3] Coros, Stelian: Generalized biped walking control, ACM Transactions On Graphics (TOG), vol.29, No.4, pp.1–9, 2010.

[4] Wang, Jack M: Optimizing locomotion controllers using biologically-based actuators and objectives, ACM Transactions On Graphics (TOG), vol.31, No.4, pp.1–11, 2012.

[5] Geijtenbeek, Thomas: Flexible muscle-based locomotion for bipedal creatures, ACM Transactions on Graphics (TOG), vol.32, No.6, pp.1–11, 2013.

[6] Peng, Xue Bin: Deeploco, ACM Transactions on Graphics (TOG), vol.36, No.4, pp.1–13, 2017.

[7] Peng, Xue Bin: Amp, ACM Transactions on Graphics (TOG), vol.40, No.4, pp.1–20, 2021.