2D Biped Walking Control

Yiran Wang
University of California, Davis
wyrwang@ucdavis.edu

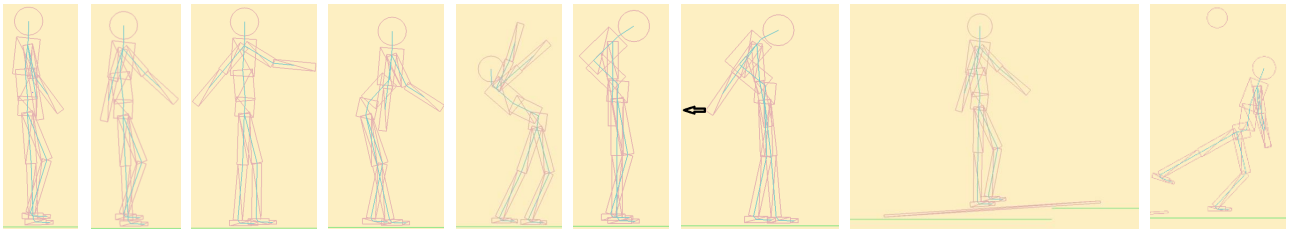


Figure 1: A part of a physics-based biped walking controller discussed in [1] is implemented and applied to create a 2D simulated character.

ABSTRACT

In this project, a part of a physics-based biped walking controller discussed in [1] is implemented and applied to create a 2D simulated character model, which results in realistic, stable and balanced walking motions. SIMBICON, the main part I implement in this project, is based on the idea of using a simple finite state machine with certain key poses and developing feedback laws for robust locomotion. A 2D physics engine, BOX2D, is used to simulate the character and a graphical user interface, openFrameworks, is used to display the animation and to allow for user interactions. People can interact with the character as well as other objects and the control strategy can generate balanced walking motions in real-time, which demonstrate the robustness to pushes in different directions, unexpected slopes, and unexpected user-designed kinematic changes.

KEYWORDS

biped walking, controller, 2D character

1 INTRODUCTION

Algorithm approaches of animated motions are supposed to be more general than other approaches such as keyframing. Controller-based algorithm approaches have the potential to handle user-interactions and can adapt motions continually. Some algorithm approaches take physics into account and these physics-based approaches of animation have the advantage that they can create motions as it occurs in the real world. The motions are achieved by applying gravity and other forces on the skeletons. The by-products of working with physical simulation will make the animation looks more natural. However, they could also be very challenging since in a physics-based animation, we can only control forces and torques rather than control motions directly.

Walking is a fundamental motion of many character animation. Another significant challenge to implement a physics-based biped walking control is the unstableness of biped characters. They need to be aware of balance and a good balance strategy for biped motions is a long-standing topic in computer animation as well as robotics. Biped walking motions also involve problems such as joint limit constrains and contact impacts. For a generalized biped walking controller, the transitions between different gaits or motions should also be taken into accounts.

In this project, a part of a physics-based biped walking controller proposed in [1] is studied and implemented in C++. The main part I implemented is the SIMBICON controller. A character is created and its walking is achieved using this controller. An animation is provided and there are a human-like character, a slope, and balls with different size in that animation, where people can interact with the character and other objects.

2 RELATED WORK

Since the importance of biped walking, there are many papers work on the simulation and control of walking.

A generalized biped walking control for physics-based simulation was proposed in [1]. It integrated several biped walking methods and generalized well across different gaits, motions, characters' proportions and skills.

The work above integrated the inverted pendulum model(IPM) method to achieve balance. A physics-based approach to use IPM on biped walking animation was proposed in [2]. The IPM was utilized to adjust the desired motion trajectories and it demonstrated a good generalization across walking gaits, motions and characters' proportions.

The work in [1] was also based on the SIMBICON controller which introduced in [3]. It is the main part of what this project works on. The works of Raibert, Hodgins, and their colleagues introduced robust hopping and running motions [4-6], which decompose the control of hopping height, torso pitch and speed. The foot placement is a key element to achieve balance. Paper [3] developed the idea. Its feedback mechanism can achieve continuous adaptations in a motion cycle and it used the position and velocity of the mass to compute the desired joint angles.

3 CONTROL FRAMEWORK

The control strategy of SIMBICON introduced in [3] can be described as a simple finite state machine and the modifications to achieve locomotion and balance.

3.1 Finite State Machine

The finite state machine is used to describe the poses of body representing target angles. Figure 2. shows the four states for a walking cycle. Since a walking cycle is left-to-right symmetric, so the states also show symmetry. The states include lifting the swing foot forwards and striking the swing foot on the ground.

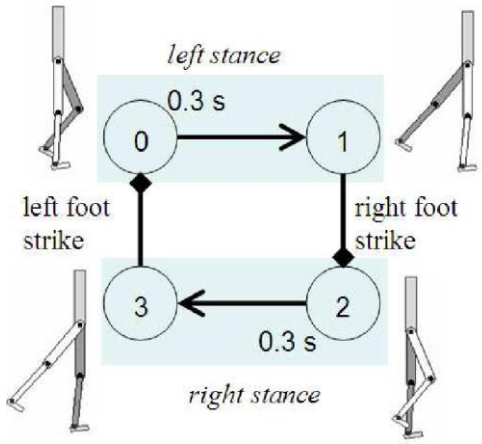


Figure 2: Finite state machine for walking. [3]

For each state, each joint has its own desired target angle with respect to its parent link, except the torso and swing leg which are respect to the world frame. Proportional derivative(PD) controllers are used to compute the torques needed to apply on the links to achieve the desired joint angles. The transitions between a stance state to a foot strike state occurs after a fixed time and when the foot contact the ground, the foot strike states transits to a stance state.

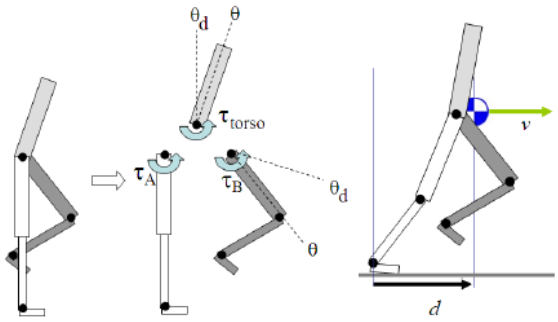
3.2 Achievement of locomotion and balance

If all the target joint angle is set with respect to their parent links, it cannot make the character to walk forwards and backwards. To achieve locomotion, the target angles for torso and swing leg are set with respect to the world frame. Then required torques will be computed and the torque on the torso is realized by the stance leg.

To keep balance of the character, the placement of the swing foot is adjusted by a feedback strategy. The feedback term shown below will modify the target angle of the swing hip continually.

$$\boldsymbol{\theta_d = \theta_{d0} + c_d d + c_v v}$$

θ_d is the modified target angle and θ_{d0} is the fixed angle given in the state. d is the distance between the stance ankle and the center of mass. v is the velocity of center of mass. c_d and c_v are gain parameters. From the feedback term we can see that if the velocity is zero and the character has the position shown in Figure 3(b), d is not zero and then θ_d will be modified and torques will be applied the character to step forward to keep the character form failing.



(a) (b)

Figure 3: (a)The target joint angles for torso and swing leg; (b) The distance between swing ankle and center of mass, and velocity. [3]

4 IMPELEMENTATION

This project was implemented in C++. The biped walking controller is applied to a human-like character created in Box2D. Box 2d is a physics engine used for simulation rigid bodies. The 2D character is shown Figure 4. It has 15 internal DOFS and 18 DOFs in total. The simulation time step is set to 0.0005s.

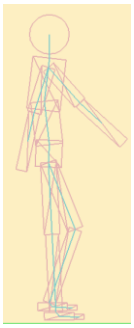


Figure 4: the character created in Box2D.

Box2D is responsible for combining the current positions, joint angles, velocities and the torques computed by the walking controller and then generated the corresponding next positions joint angles and velocity, which are feed back to the controller to compute required torques. For the initial state, a small velocity is set to the character to make it start walking.

A graphical user interface, openFrameworks, is used to display the animation. For each updates in openFrameworks, it will check the time since the last time to display the character. If the time is larger than 0.01s, the program for computing the position for each body will iterate required times and then the bodies of the character will be shown on the screen and we can see the animation. The openFrameworks is also used to handle the interactions with users. It checks the action of mouse and if the mouse performs a defined operation, it will detect that operation,

5 Results and Discussions

The walking motions of the character is achieved in this project. The two accompanying videos with give a better display of these results.

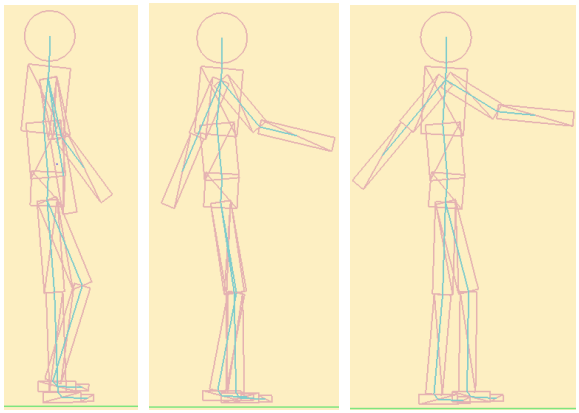


Figure 5: Different phases of a walking cycle.

5.1 Different walking motions

The character can achieve different walking motions including walking forwards, walking backwards and step-in-place. The walking speed changed by the character itself automatically to achieve balance requirement. Figure 5 shows the different phases of a walking cycle. The default direction is walking forwards and the character only walks backwards to achieve balance. If the velocity is zero and the character is in balance, it will keep stepping in-place. I did not implement a transition between walking and standing in this project.

The shape of the body and the trajectories of arms and feet can be modified manually to achieve different walking styles. And different shapes will result in different walking performance, such as the walking speed or the robustness to disturbance, since the place of the center of mass is different and gravity has a great influence on the motions. Some walking character with different body shapes are shown in Figure 6.

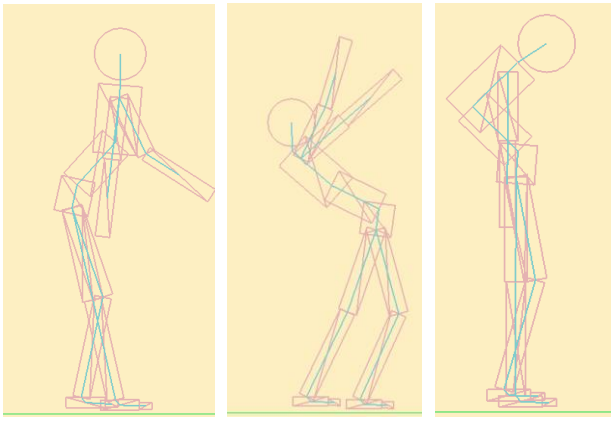


Figure 6: Different walking styles.

5.2 Interactions with people

The character can interact with people. We can drag the character and other objects to any positions. The program will find which body is being dragged. Then Box2d will generate appropriate force to that body and the character will response to the drag. Some motions with drag are shown in Figure 7. This could be more clear in the video.

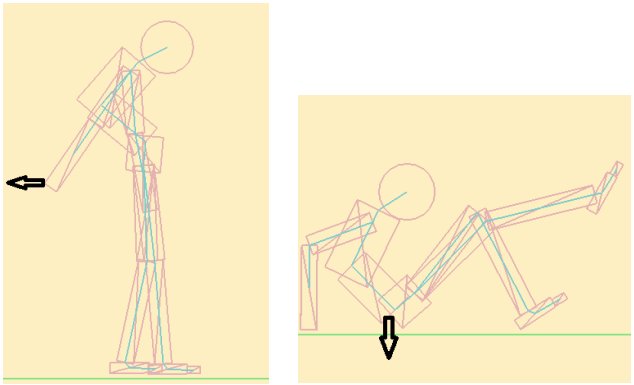


Figure 7: Motions with drag.

5.3 Interactions with environment

The character can interact with environment such as walking on a slope and collision with balls. When the character is walking on a slope, the duration time of the foot strike state will change since this state will transit to the nest state at the time the foot contact with the ground. The trajectory of the swing foot will change since the swing to need to move on the height direction. This will also influence the position of center of mass, which will influence the ability to keep balance. In this project, the character can walk on a slope with small angle, which is shown in Figure 8.

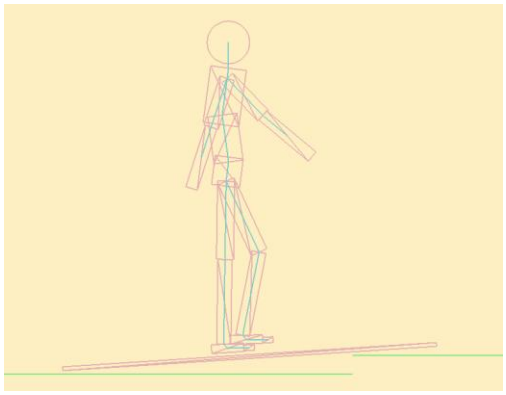


Figure 8: Walking on a slope.

The character is also robust to the collision with small balls. The collision is achieved in the similar way of drags. When collision of two bodies happens, the forces will be applied on both of the bodies. If the collision happens on our character, the force will introduce a change in velocity. The balance controller will be applied to check the current state and try to make the character recover from disturbance.

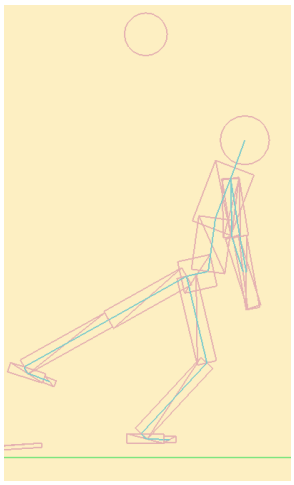


Figure 9: Character collides with a small ball and then fails down .

5.4 Robustness

As has shown above, the biped walking controller is robust to small disturbances. The character is able to recovery to from disturbance of drag, collision and environment. However, if the disturbance is big enough, the character may fail down and cannot recover by itself, shown in Figure 9. A drag by user could help the character to stand up again.

4 CONCLUSIONS

In this project, a part of a physics-based biped walking controller discussed in [1] is implemented and a 2D human-like character is created in Box2d. The character can perform walking forwards, walking backwords and step-in-place motions. People can interact with the created characters and other objects. The character is robust to small disturbance such as drag by user, collision with small objects and different environments.

REFERENCES

[1] Coros S, Beaudoin P, van de Panne M. Generalized biped walking control[C]//ACM Transactions on Graphics (TOG). ACM, 2010, 29(4): 130.

[2] Tsai Y Y, Lin W C, Cheng K B, et al. Real-time physics-based 3d biped character animation using an inverted pendulum model[J]. IEEE transactions on visualization and computer graphics, 2010, 16(2): 325-337.

[3] Yin K K, Loken K, van de Panne M. Simbicon: Simple biped locomotion control[C]//ACM Transactions on Graphics (TOG). ACM, 2007, 26(3): 105.

[4] Hodgins J K, Pollard N S. Adapting simulated behaviors for new characters[C]//Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 1997: 153-162.

[5] Raibert M H, Hodgins J K. Animation of dynamic legged locomotion[C]//ACM SIGGRAPH Computer Graphics. ACM, 1991, 25(4): 349-358.

[6] Hodgins J K. Biped gait transitions[C]//Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. IEEE, 1991: 2092-2097.

APPENDIX

The executable file is at \proj279\bin\proj279.exe.
To build the source code, the openFrameworks source code need be downloaded first and a plugin “openFrameworks plugin” need to be add to VS 2015 version.
The include directories need to be changed to specify the current dir where the code is.