

PD Controllers and Physically-based Human Animation

Abstract

Human animation has been an integral part of a diverse range of media such as games, movies and virtual trainings. However, realistic human animations are difficult to achieve because of the complicated structure of human body and our extreme familiarity with human motions. This survey focuses on the dynamic approach to animate human motions, and in particular, discusses the techniques that are used to generate physically-base human animations using Proportional-Derivative controllers (PD controllers). The three primary systems discussed in this survey are: the Limit Cycle Control (LCC); Simple Biped Locomotion Control (SIMBICON), and Simulating Biped Behaviors from Human Motion Data (SIMMOCAP). All of the systems use PD controller in the simulation of bipedal walking. LCC focuses on cyclic motions like walking and running. It solves the control perturbations on top of a handcrafted nominal control to keep system track the nominal state cycle and therefore keep balance of the simulated character. SIMBICON directly incorporates balance in their controller design and uses a feedback error learning to achieve robust balancing. The system is not limited to cyclic motion and can handle transitions according to a predefined finite state machine map. SIMMOCAP rectifies the motion capture data via optimization so that the simulated character can follow the rectified motions robustly. Interpolations and transitions of the rectified motions can be used to simulate new animations. The three systems are compared in terms of simulation robustness, computational cost and motion diversity. Their advantages and disadvantages are discussed and the pros and cons of PD controller based human simulations in general are analyzed in this paper.

1. Introduction

1.1 Importance of character animation

Human animation has been an integral part of diverse range of media. Research work in human animation has played a major role in feature animations, computer games and special effects in movies. In addition, human animation is important in simulation and training to better prepare users to handle equipments in various situations. Further more, it is used in the study of medical analysis to enhance the design of equipments that aid people with disabilities. There is a strong reason to believe that in the future, human animation will play an even more important role in human computer interaction, exerting deeper effects in our lives.

Realistic human animation is a difficult problem. The human body is a complicated structure (206 bones for adults on average), which is capable of performing actions ranging from subtle actions like a sigh to dynamic actions like a flip. A typical skeleton used in human animation has over 60 degrees of freedom (DOF). On the other hand, having been observers of human motions throughout our lives, humans are experts in distinguishing different types of human motions. Just by looking at the style of a walking motion, we can tell if the person was sad or happy at that moment. Therefore, in order to achieve a high fidelity human animation, many aspects of human motion have to be studied carefully. Unfortunately, this is a very complicated and long term effort. One common simplification to this problem is to consider the human body as a rigid skeleton that drives the deformable skin. Animations are generated by changing the joint angles over time. Figure 1¹ illustrates a typical skeleton model for human animation.

There are two common approaches used in the generation of human animations: kinematic approach and dynamic approach. In general, kinematic approach directly manipulates the joint angles and considers few or none of the physical properties of the animation generated [2, 14, 1], whereas dynamic approach treats the human model in a

¹ Figures and tables in this paper are captured and edited from the following sources: Figure 1: [7], Figure 2: [13], Figure 3-12: [15], Figure 13-16: [31], Figure 17-21, Table 1: [25].

physically-based fashion and uses torque and force to simulate the behaviors [10, 5, 18, 6]. Other studies [32] which combined the two approaches showed success by trading off optimality with computational cost.



Figure 1. Human skeleton model.
Deformable skin and joint hierarchy.

1.2 Kinematic approach

Kinematic approach animates the character through manipulation of joint angles. It focuses on the joint angles and root transformation changes over time rather than the force and torque changes. Keyframing is one of the traditional methods. A key artist arranges the poses of the character at key moments of the animation. Later on, the assistant artist adds the in-between frames by interpolating the key frames. Keyframing gives artists the ultimate control of the animation result, but usually requires time and talent. Moreover, the result animation does not guarantee physical realism and the in-between motions often need manual adjustments to eliminate artifacts like foot skates. Motion capture on the other hand uses calibrated cameras to triangulate marker positions on a human performer to drive the virtual character. The result animations are usually realistic and exact. However, remapping the animation to a character of a different size or configuration, and modifying the animation to satisfy different constraints are difficult.

Given the advantages and disadvantages of keyframing and motion capture, researchers have come up with various methods to better control the animation under user specified constraints. Inverse kinematics (IK) solves the skeleton joint angles so that one or more joints satisfy the given the position and orientation constraints [8, 27, 26]. Welman’s survey [27] gives an overview of various computational algorithms for numerical solutions. These algorithms work well with full body skeleton and can handle sophisticated constraints. However, they suffer from computational expenses and ill-condition problems [19]. Analytical approaches, on the other hand, find a closed-form solution to IK systems. However, such approaches usually only handle relatively simple joint structures [12]. Moreover, IK solutions do not guarantee naturalness by itself. Interpolation methods solve the naturalness problem by interpolating existing natural poses to generate new ones [2]. Multi-target interpolation is used to create parameterized motions [28]. Rose et al. combined it with linear blend transitions to create the Verbs and Adverbs system for parameterized motions [23]. Motion graph technique takes a different approach by building a graph out of small motion clips for the creation of new animations. The edges of the graph correspond to motion clips and nodes indicate how these clips can attach. Some of the edges correspond to original data and others are synthesized transitions (Figure 2). The assumption of motion graph technique is that there exist enough redundancy and smooth transitions between motion clips from different animations, such that a concatenation of selected motion clips can produce the animation that satisfies user needs. Kovar et al. uses branch and bound graph search together with beam search technique to perform such selection with reasonable efficiency [14].

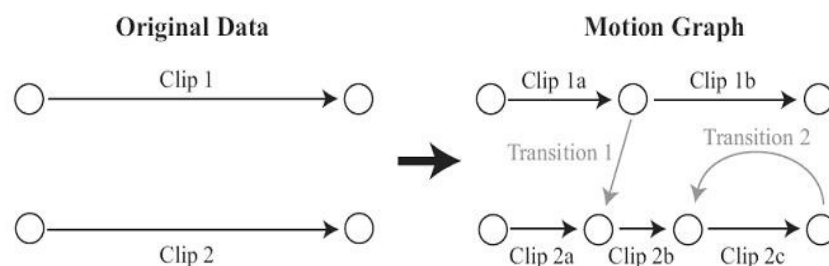


Figure 2. Building a motion graph from two motion clips. While preserving the transitions in the original motion clips, the motion graph introduces new transition within and between the original clips.

Most kinematic approaches leverage the high fidelity of motion capture data and are capable of generating life-like animations. However, the resulting animation is limited by the diversity of the existing data set. Therefore, in order to produce a general purpose animation generator, one has to sample a very rich motion capture data set and design a sophisticated algorithm to produce the desired animation. Moreover, since these approaches do not model physics explicitly, there is no guarantee of physical realism in the generated animations.

1.3 Dynamic Approach

Dynamic approach to human animation focuses on the realism of the generated animation. It builds an internal physical model, and simulates the motion of it under the laws of motion. Instead of directly providing joint angles and root translation in the kinematic approach, the system solves the torque and force for each joint and integrates them to get joint angles and velocities. In this way, physical realism is guaranteed, but finding the torques that maintain the desired animation over time is a difficult task. Two popular solutions are controller based and optimization based methods.

Hodgins et al. [10] came up with carefully designed joint controllers based on finite state machine and proportional-derivative servos. They demonstrated this approach on several types of motion, including running, bicycling, and vaulting. Laszlo et al. [16] applied limit cycle control to joint controllers to make sure that the generated motion follows the designed repetitive nominal motion. To adapt an existing controller to a new character, Hodgins et al. [9] computed new controller parameters by first applying approximate scaling strategies and then tuning the result with simulated annealing. Finally, Faloutsos et al. [4] composed their new controller by machine learning. They trained Support Vector Machine (SVM) to learn the conditions under which controllers for different actions can be composed, providing a meta-controller that switches control strategies to transition between actions.

Constrained optimization in character animation dates back to the eighties, when Witkin et al. demonstrated this approach with a jumping Luxo lamp [29]. In general, this approach requires user specified start and end poses as well as a physically-based

objective function. The optimizer minimizes the objective function and computes the details of the motion while satisfying the pose and physical constraints across all frames. Optimal energy movement and intuitive control give this method great appeal. However, for complex characters the system is high dimensional and highly nonlinear, preventing the optimization from converging to a solution. One way to simplify the problem is to reduce the optimization space. Popović et al. [21] presented an interactive simulation system that allows user to drag a rigid body, like a hat, and place it at a desired location. The system solves the required physical parameters and simulates the result motion. Safonova et al. [24] exploited the fact that most dynamic motions are intrinsically low dimensional and constructed a low dimensional space that captures the properties of the desired behavior from motion captured data. Then optimization was solved in this reduced space efficiently while all physical constraints were satisfied. Fang et al. [6] proposed a way to enhance the optimization efficiency by considering only constraints and objective functions that lead to linear time first derivatives. They showed examples of swinging and leaping motions for characters with varying DOFs. Realistic motion that preserves physical properties can also be obtained from editing motion capture data [22].

In general, dynamic approaches automatically generate motions that are physically realistic, relaxing the user from worrying about the interaction details between the character and environment. In the controller based case, once the controllers are properly designed, the system works for various tasks under many environments. In the optimization based case, once the user specifies the start and end poses and a physically-based objective function, the animation details are generated automatically. However, satisfying the physical constraints does not guarantee naturalness. In controller based case, most of the generated motions are robot-like, lacking details of smoothness and personality in human motion. In optimization based case, whether natural human motions are indeed optimal or not is still under discussion. It might be true for highly dynamic motions, but for low energy motions, motion styles might have more impact than dynamics does.

2. Definition of PD Controllers

The topic we are interested in follows the controller based idea from the dynamic approach. We focus on the physical simulation of biped locomotion (Figure 4). The model is a rigid body with 12 rotational joints. Most of the joints have 1 DOF, except that the hips have 3 and the ankles have 2. Mass and inertia parameters are taken from [30] and they reflect realistic human statistics.

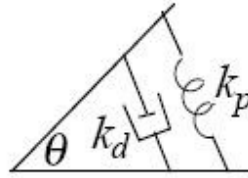


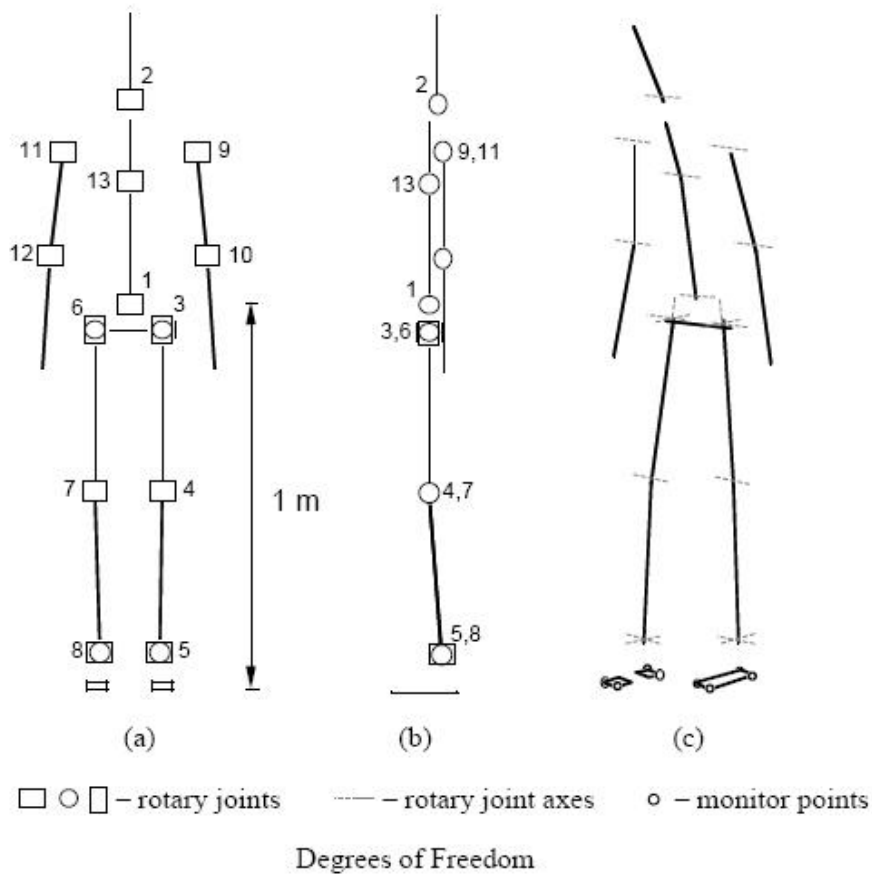
Figure 3. Rotational PD controller.

In the implementations of different systems, each joint of the human model is driven towards some desired angles by joint actuators following Proportional-Derivative (PD) control law. Figure 3 illustrates a rotational PD controller:

$$\tau = k_p \cdot (\theta_d - \theta) - k_d \cdot \dot{\theta} \quad (1)$$

τ is the control torque applied to the joint, θ_d is the desired joint angle, θ is the current joint angle, $\dot{\theta}$ is the joint angular velocity and k_p and k_d are the proportional and derivative control gains.

k_p and k_d specify the strength of a rotational spring and damper pair. This simple model captures the key factors of realistic joint motion while abstracting away from the details of muscle structure and functionality. The actuator gains are constant and are considered part of the model specification.



- | | |
|--|--|
| 1 - waist pitch (sagittal plane) | |
| 2 - neck pitch (sagittal plane) | |
| 3:0 - left hip roll (coronal plane) | 6:0 - right hip roll (coronal plane) |
| 3:1 - left hip pitch (sagittal plane) | 6:1 - right hip pitch (sagittal plane) |
| 3:2 - left hip yaw (transverse plane) | 6:2 - right hip yaw (transverse plane) |
| 4 - left knee pitch (sagittal plane) | 7 - right knee pitch (sagittal plane) |
| 5:0 - left ankle pitch (sagittal plane) | 8:0 - right ankle pitch (sagittal plane) |
| 5:1 - left ankle roll (coronal plane) | 8:1 - right ankle roll (coronal plane) |
| 9 - left shoulder pitch (sagittal plane) | 11 - right shoulder pitch (sagittal plane) |
| 10 - left elbow pitch (sagittal plane) | 12 - right elbow pitch (sagittal plane) |
| | 13 - mid-back pitch (sagittal plane) |

Figure 4. (a) front view, (b) left side view, (c) perspective view with monitor points shown for foot contact.

3. Simulation Systems with PD Controllers

In a PD controller based simulation system, users specify target joint angles. Ideally the PD controllers will drive the biped model to the target pose by solving equation (1) for each joint. The control poses could also come from motion captured data directly. Ideally, this will allow the bipedal model to follow the motion capture data and produce the desired animation. However, PD controllers alone can only track the desired poses for a very short period of time and eventually fall over due to the loss of balance. There are two main reasons behind this failure. Firstly, when users design the target joint angles, they are not aware of the underline dynamical model, which makes the control poses difficult to achieve during the simulation. In the case of tracking motion capture data, the dynamic model of the human performer is different from the bipedal model. Secondly, the interaction with the environment deviates the simulation pose from target poses. Given the time limit in simulation and torque limits on the joints, the system is not able to recover from the deviation and will diverge to the falling over.

Given the fact that simple PD controllers fail to track target poses exactly, the three systems (LCC [16], SIMBICON [31], SIMMOCAP [25]) discussed in this survey looked at this problem from different perspectives and proposed novel solutions.

3.1 LCC

As stated above, one of the problems that cause the falling over is that the simulation pose deviates from the target poses and cannot be recovered within the given time and torque limits. In order to fix the problem before it is too late, Laszlo et al. [16] introduced Limit Cycle Control to the simulation system. In this design, the system focuses on cyclic motions such as walking, because such motions have a nominal cyclic trajectory in the state space. Aperiodic motions such as sitting down and standing up are not discussed.

For stable cyclic motions, their state follows a closed loop in the state space over time. Little external perturbations damp rapidly in order to maintain its stability, which is called passive stability. However, for unstable motions like walking, the system cannot

rely on this passive self damping. The goal of Limit Cycle Control is to actively introduce external control to the system to keep it to a fixed cycle (Figure 5). Limit Cycle Control avoids dealing with complicated non-linear dynamics, but instead presents the motion with a well-behaved discrete dynamical system.

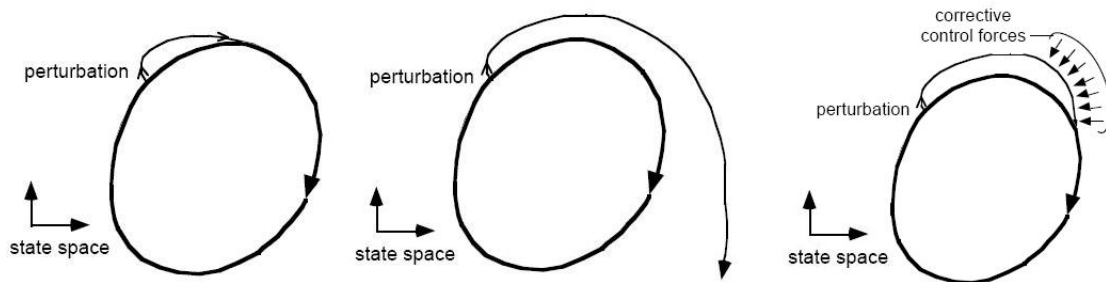


Figure 5. Limit Cycle Stability.

Left: Passively stable. Middle: Passively unstable. Right: Active stable

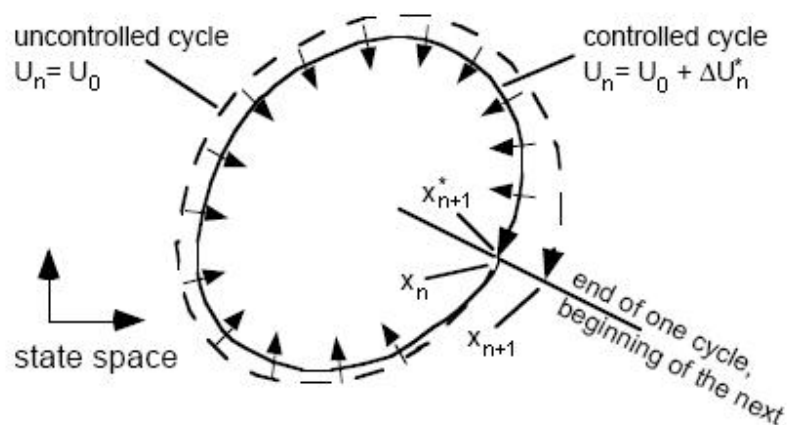


Figure 6. Discrete system

- x_n initial state of the n^{th} cycle and final state of the $(n-1)^{\text{th}}$ cycle
- x_{n+1} final state of the uncontrolled n^{th} cycle
- x_{n+1}^* final state of the controlled n^{th} cycle and initial state of the $(n+1)^{\text{th}}$ cycle
- u_0 nominal cyclic control
- Δu_n computed control perturbation for n^{th} cycle

The discrete dynamic system that we wish to model can be expressed in the following form:

$$x_{n+1} = g(x_n, u_0 + \Delta u_n) \quad (2)$$

where x_n is the initial state of the n^{th} periodic limit cycle, x_{n+1} is the state of next cycle, u_0 is the periodic nominal control and Δu_n is an applied control perturbation at cycle n . u_0 alone usually fails to maintain the stability of the system, therefore, Δu_n is introduced to drive the system state to a desired value x_d . Figure 6 illustrates the discrete dynamic system.

By introducing this discrete dynamical system, we can perform linear approximation when control perturbation Δu_n is relatively small. This results to the equation (3)

$$\Delta x_{n+1} = J \Delta u_n \quad (3)$$

where J is a Jacobian relating the change in state space after one cycle to the control perturbation. Therefore, if we know the difference between our next state and the desired state, we will be able to approximate the control perturbation to cover that difference by the following equation:

$$\Delta u_n = J^{-1} \Delta x_{n+1} \quad (4)$$

where $\Delta x_{n+1} = x_d - x_{n+1}^0$. x_d is the desired state and $x_{n+1}^0 = g(x_n, u_0)$ is the state achieved when only nominal control u_0 is applied.

We can further constrain that

$$\Delta u_n = \sum_{j=1}^N k_{nj} \Delta \hat{u}_j = K_n \bullet \Delta \hat{U} \quad (5)$$

where $\Delta \hat{u}_j$ are a number of fixed perturbations and k_{nj} are the coefficients. K_n and $\Delta \hat{U}$ are their matrix form. In another word, instead of finding the perturbation vector, we are looking for the perturbation scalars of a set of fixed perturbations served as the principal components. These fixed perturbations are carefully tuned by hand.

The computation of perturbation depends on the initial state, so each cycle, x_{n+1}^0 needs to be simulated and the Jacobian J needs to be recomputed. This is a very costly step in every motion cycle, so instead of working in the full state space, LCC projects the state space into a reduce space of Regulation Variables (RVs). These RVs incorporate the key element(s) of the desired cycle and reflect movements or positions relative to the world frame. Moreover, they are easy to control. In LCC, the following RVs are used (Figure 7):

- i. The swing center of mass (swing-COM) vector, describes the position of the center of mass with respect to the current swing foot.
- ii. The stance center of mass (stance-COM) vector, describes the position of the center of mass with respect to the current stance foot.
- iii. The up vector, is a vector that points along the length of the torso, from waist to chest. It measures the uprightness of the body.

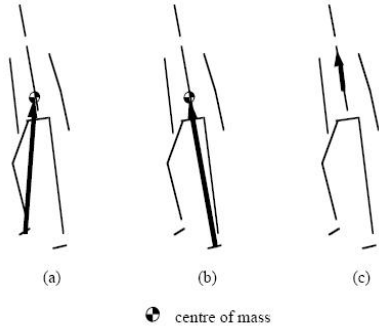


Figure 7. Regulation Variables
(a) swing-COM vector,
(b) stance-COM vector, (c) up vector

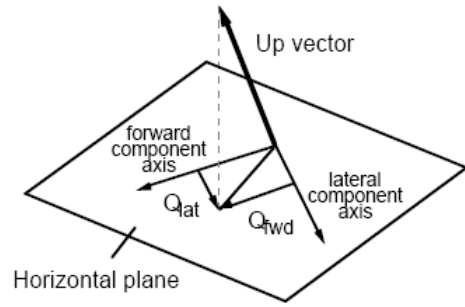


Figure 8. Up vector projection

The RVs are two components scalar values, resulted from the projection of the 3D vectors to the horizontal plane. The two components correspond to the values in forward and lateral directions respectively. Figure 8 illustrates the up vector projection.

Replacing equation (2) - (5) with a function that relates state variable x and chosen RV Q , such that $Q = \gamma(x)$, we have:

$$Q_{n+1} = h(Q_n, u_0 + K_n \cdot \Delta \hat{U}) \quad (6)$$

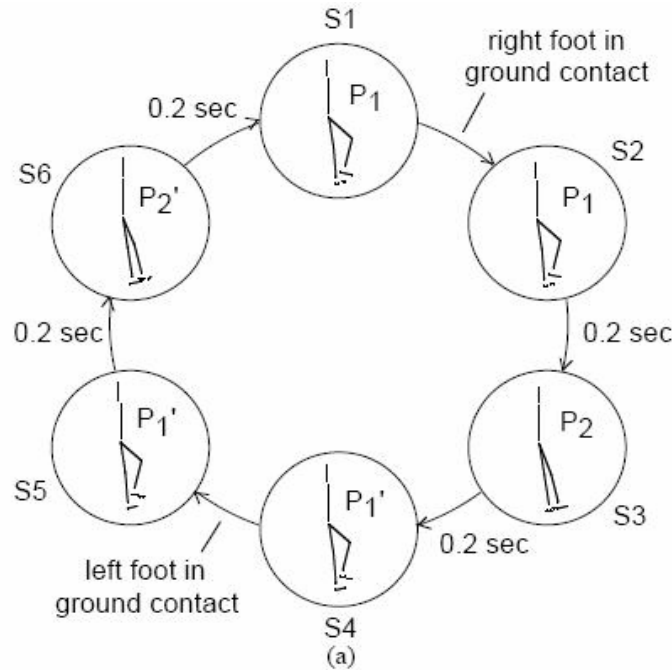
LCC chooses to approximate the response of the system about the nominal operating point u_0 , which yields:

$$\Delta Q_{n+1} = JK_n \quad (7)$$

Finally the control perturbation scalars can be computed:

$$K_n = J^{-1} \Delta Q_{n+1}^d \quad (8)$$

where, ΔQ_{n+1}^d is the difference between desired RVs and RVs resulting from nominal control in current cycle.



DOF:	1	2	3:0	3:1	4	5:0	5:1	6:0	6:1	7	8:0	8:1	transition info
S1	0	0	0	-50	60	-5	0	0	-10	0	0	0	0 R
S2	0	0	0	-50	60	-5	0	0	-10	0	0	0	.2
S3	0	0	0	-20	0	5	0	0	-10	0	0	0	.2
S4	0	0	0	-10	0	0	0	0	-50	60	-5	0	0 L
S5	0	0	0	-10	0	0	0	0	-50	60	-5	0	.2
S6	0	0	0	-10	0	0	0	0	-20	0	5	0	.2

(b)

(a) State diagram (right foot dashed). Poses P_i and P_i' are left-right symmetric.

(b) Pose table. All DOFs are in degrees relative to reference position. Transition information is given as time (seconds) followed by an optional foot sensor type (left or right) for sensor-based transitions.

State	Description
S1	Right foot placed on ground Left leg begins forward swing (knee bent for ground clearance)
S2	Right leg propels body forward Left leg continues forward swing (knee bent for ground clearance)
S3	Left leg extends in anticipation of ground contact
S4	Left foot placed on ground Right leg begins forward swing (knee bent for ground clearance)
S5	Left leg propels body forward Right leg continues forward swing (knee bent for ground clearance)
S6	Right leg extends in anticipation of ground contact

Figure 9. PCG for a walk cycle

In the case of bipedal walking, a base Pose Control Graph (PCG) is constructed to represent all phases in a walk cycle (Figure 9). The PCG specifies the maximum time or condition to transition from one state to its following state. A careful design of each pose

and transition time can produce a few cycles of balanced walk. Even though it falls eventually, the base PCG provides the nominal control u_0 for every cycle of the walk simulation (Figure 10).

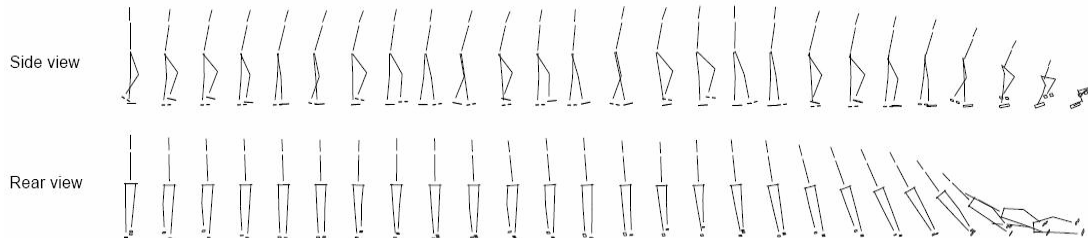


Figure 10. Side view and rear view of the unbalanced motion only using base PCG.

Since Q^{nom} (nominal RV state produced by u_0 only) and J change in every cycle, it is necessary to construct a new discrete system model every step. Therefore, for every cycle, LCC samples $h(K_n)$ for linearization around u_0 and computes J^{-1} to find out the perturbation scalars. The results are used in the actual simulation for current walk cycle. This process iterates. Figure 11 illustrates the procedure within one cycle.

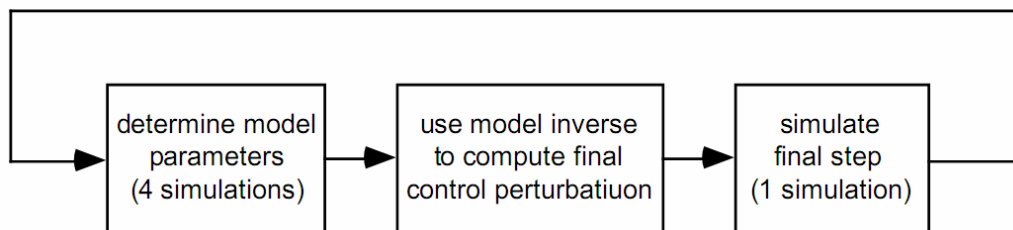


Figure 11. Simulation steps for one cycle.

Now that LCC is able to produce robust simulation of the designed motion in base PCG, it is time to introduce some variations to the simulation in order to allow additional user controls. If we have designed two versions of the system that can produce fast and slow walk, the interpolation of these base controllers can be used to simulate walking with intermediate speed. If we introduce another parameter in the base PCG to model hip twist, we are also able to control the direction of the walk in 3D. The interpolation of

base controllers can also be applied with varying weights across time, producing motion transition effects. Figure 12 shows the simulation result of LCC.

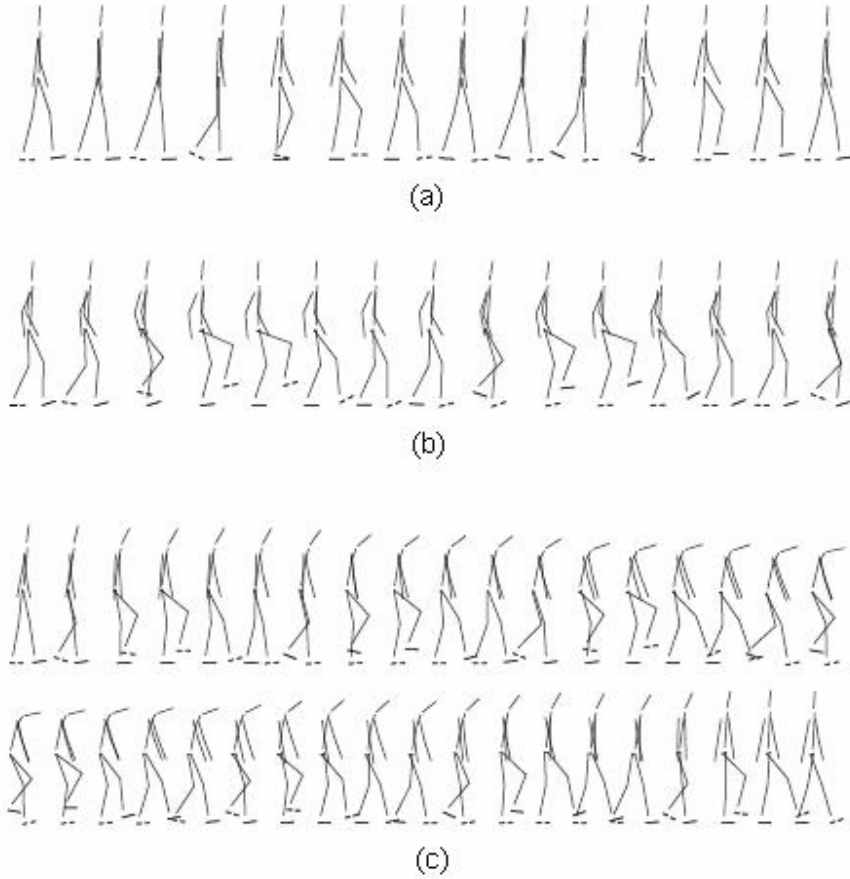


Figure 12. LCC simulation result. (a) normal walk. (b) walk with a different style. (c) walk with a duck. (b) and (c) are achieved by interpolating the base controllers.

3.2 SIMBICON

The key idea in SIMBICON [31] is to introduce balance control strategy into the controller design. This control strategy has three main elements: a finite state machine representing the pose control graph, torso and swing-hip control, and balance feedback.

Similar to LCC, SIMBICON uses finite state machine with each state being the target pose to abstract the motions for simulation (Figure 13). In any given state, the joints apply torques computed by the PD controllers in the form of $\tau = k_p \cdot (\theta_d - \theta) - k_d \cdot \dot{\theta}$.

Note that the poses in the finite state machine only represent the desired set of joint angles and are typically not actually achieved during simulation. For example, in state 1 of Figure 13, the swing leg usually extend forwards in actual simulation, but in the target state, it is extended backwards in order to introduce the effect of moving swing leg backward and down, bringing it into contact with the ground. This is different from the LCC system, which tries to make the system follow the poses in the control graph as exact as possible.

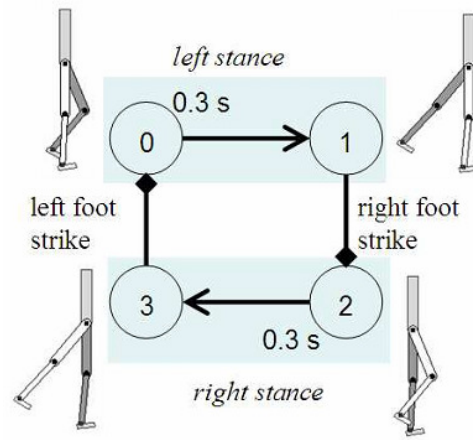


Figure 13. Finite state machine for walking

In SIMBICON, the stance hip and swing hip are handled separately. The system computes the torso torque τ_{torso} and swing hip torque τ_B according to the desired joint angles expressed in world coordinates. The total torque on the stance hip joint must be realizable, so the stance hip torque τ_A is set to $-\tau_{torso} - \tau_B$, such that $\tau_{torso} + \tau_B + \tau_A = 0$.

The last component of the control strategy is to apply the following balance feedback strategy to the swing foot placement.

$$\theta_d = \theta_{d0} + c_d d + c_v v \quad (9)$$

where θ_d is the target angle for the PD controller, θ_{d0} is the default target angle from the finite state machine, d is the horizontal distance from the stance ankle to the center of mass (COM) and v is the velocity of the center of mass. In SIMBICON, the position of COM is approximated by the midpoint of the hips. The combination of (d, v) provides

complete information about the current phase of a gait cycle. Figure 14 illustrates the balance control strategy.

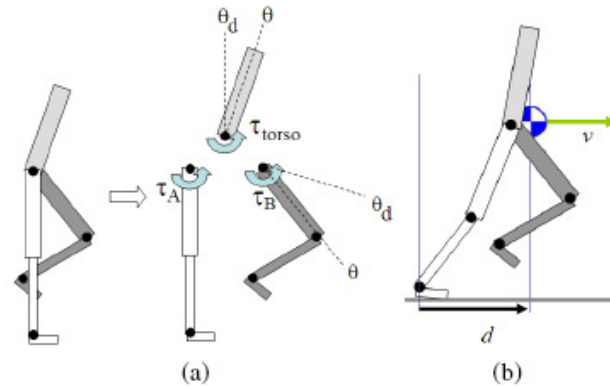


Figure 14. Balance control strategy. (a) Torso, stance-hip and sw torque relationships. (b) Balance feedback control.

Now that the system is set up, the following parameters need to be tuned in order to make the system run robustly: a). the number of states of the FSM and the state-transition duration; b). the balance feedback gains, c_d and c_v ; c). the target joint angles for the pose in each state. The choice of the state number for each motion reflects phases of different locomotion. In SIMBICON, walking has four states, running has two states and skipping with eight states is the most complicated. The values of balance feedback gains require some trial-and-error tuning. The target joint angles are just from intuitive design of poses that represent the motion in consideration. The most important parameters for each state are the state duration and target angles for the swing hip and swing knee, because they determine the next foot placement, which is vital to keep balance. SIMBICON is also able to mimic the motion capture data. In order to achieve so, the input motion capture data have to be preprocessed to find a proper cycle frequency and representative poses. The target poses are computed from the rectified motion capture data and the PD controllers take a slightly different form of $\tau = k_p \cdot (\theta_d - \theta) - k_d \cdot (\dot{\theta} - \dot{\theta}_d)$ in order to take into consideration of the nominal angular velocity $\dot{\theta}_d$. Nonetheless, the system is only able to mimic the motion capture data and is not able to perfectly imitate them for a number of reasons. First, the original motion capture data have much higher DOFs to

display subtle motions that are not considered in the bipedal model simulation. Second, the physical system of the person being captured is different from the bipedal model used in the simulation. Finally, in order to maintain balance during simulation, the system foregoes tracking of the stance hip angles.

SIMBICON framework has been applied to both 2D and 3D models. Figure 15 shows the details of the models. The 7-link 2D biped has a 70 kg trunk, 5kg upper legs, 4 kg lower legs and 1 kg feet. The respective lengths are 48 cm, 45 cm, 45 cm and 20 cm. The 3D model is the same as used in LCC [16]. Figure 16 shows the simulation result in both 2D and 3D cases.

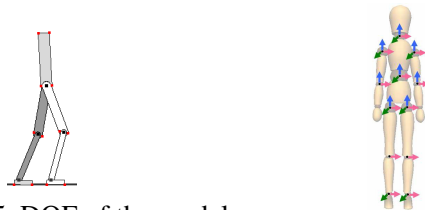


Figure 15. DOF of the models.
 Left: 2D model with 6 internal DOFs, 9 DOFs in total.
 Right: 3D model with 28 internal DOFs, 34 DOF in total.

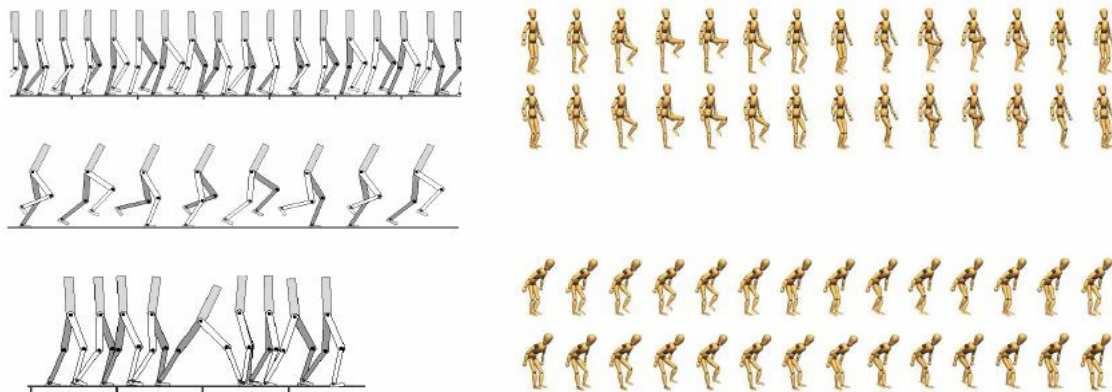


Figure 16. SIMBICON results. Left column are 2D simulations of normal walk, run, and big step. Right column are 3D simulations to imitate high knee walk and bend forward walk from motion capture data. In each group, the top row is the motion capture data and bottom row is the imitation result.

3.3 SIMMOCAP

Directly learning a biped behavior from motion capture data is difficult, because the bipedal model is drastically simplified from the human performer. Therefore, SIMMOCAP [25] system proposed an optimization approach to solve this problem by rectifying the motion capture data before feeding it to simulation.

The rectified target motion $\widehat{m}(t)$ is a modification of input motion data $m(t)$ in the following form $\widehat{m}(t) = m(t) + d(t)$, where $m(t) = (\theta_1(t), \dots, \theta_n(t))$, n is the number of joints and $\theta_i(t)$ is the angle of joint i at time t . $d(t)$ is the displacement map and $d(t) = (d_1(t), \dots, d_n(t))$, where each $d_i(t)$ is represented in a parametric form with bell-shaped basis functions:

$$d_i(t) = \sum_{j=1}^m h_{ij} B_j(t; c_j, w_j) \quad (10)$$

where m is the number of node points and the sinusoidal function

$$B_j(t; c_j, w_j) = \begin{cases} \frac{1}{2} \left[1 + \cos\left(\frac{\pi}{w_j}(t - c_j)\right) \right], & \text{if } c_j - w_j < t < c_j + w_j \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

has a peak at node point c_j , extends from $c_j - w_j$ to $c_j + w_j$ and has zero value outside the interval. The node points $\{c_1, \dots, c_m\}$ are spaced non-uniformly and each node point corresponds to either a takeoff, kick down of a stance foot or the halfway of a swing phase (Figure 17). The optimization algorithm takes $m(t)$ as input and tries to solve $(h_{11}, \dots, h_{nm}, w_1, \dots, w_m)$ of the motion displacement.

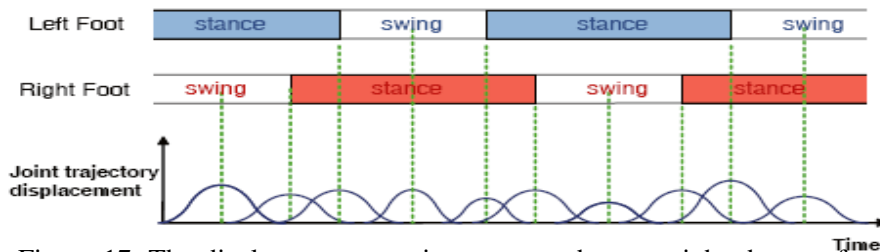


Figure 17. The displacement map is represented as a weighted sum of sinusoidal basis functions. The basis functions are distributed along the time axis synchronized with stance and swing phase.

The PD controllers used in SIMMOCAP is in the form of $\tau = k_p \cdot (\theta_d - \theta) - k_d \cdot (\dot{\theta} - \dot{\theta}_d)$, where θ_d is the joint angle and $\dot{\theta}_d$ is the joint angular velocity from the rectified motion $\hat{m}(t)$. The simulation starts with the first frame of the original input motion $m(t)$ and $\tilde{m}(t)$ is the output motion.

The goal of the optimization is to minimize the difference between $m(t)$ and $\tilde{m}(t)$. The object function is:

$$E = \int_0^T \left(\left(\frac{t}{T} \right)^2 + c \right) \cdot \text{diff}(m(t), \tilde{m}(t)) dt, \quad (12)$$

where $\text{diff}(m(t), \tilde{m}(t))$ compares the mass points on the skeleton in the two poses. The system assumes that masses reside on the end joints of the limbs and the two sets of mass points obtained from $m(t)$ and $\tilde{m}(t)$ are first normalized so that they have the same average x-coordinates and then their difference is computed as the squared sum of the Euclidean distance multiplied by the mass. The weight term $\left(\left(\frac{t}{T} \right)^2 + c \right)$ allows the optimized motion to deviate from the target motion in the short term in order to achieve a long term tracking.

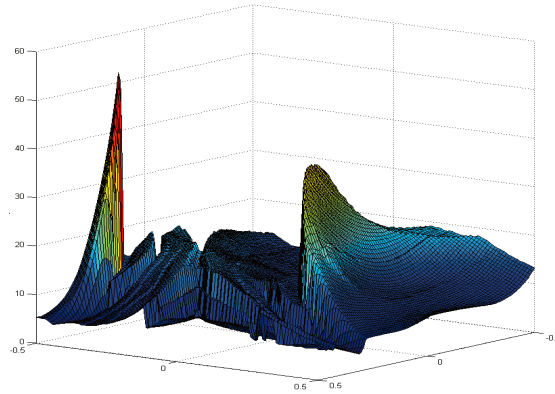


Figure 18. The objective function has many local minima. This plot shows a sampling of the objective function by varying two parameters: the heights of displacement maps for the right ankle and the right knee.

This objective function has many local minima (Figure 18). Therefore, a good optimization method is needed in order to find the minimum value of the function fast and robustly. SIMMOCAP uses downhill simplex method with randomly chosen initial

parameters. Such process is done repeatedly and the minimum among the result local extreme value is selected as the solution. According to the authors of SIMMOCAP, such optimization method is slow but robust. In order to reduce the optimization time, SIMMOCAP uses an incremental technique inspired by space time window [3]. In addition, to reduce the dimensionality of the bipedal model, the upper body PD servos are excluded from the optimization, resulting that the displacement maps are computed only for six joints (hips, knees, ankles) in the lower body to keep balance. SIMMOCAP took about 15 minutes to optimize 410 frames of a walking motion that includes a start, six steps and a stop. This gives us a glance at the optimization speed of the system.

Given the rectified motions from optimization, two types of controllers are built in SIMMOCAP in order to actuate the PD controllers to produce different types of motions: stationary controllers for maintaining walking, running, and standing behaviors; transitioning controllers that handles the transition between these activities. Figure 19 shows an example of the finite station machine representing different motion behaviors and transitions between them.



Figure 19. An example of finite state machines that govern transitioning between motor controllers. The nodes correspond to motor controllers and the edges show the transition possibilities between controllers.

Given any novel state of the model, SIMMOCAP uses a simple regression technique to select the nearby sample states from rectified motions and combine them to output the state for the next time step. The distance measurement of the state space includes the following features:

1. Joint angles and velocities. The weighted angle and velocity values on all the joints are included in the feature vector. The weights on lower body joints are much higher than those in upper body.
2. Root position, velocity and direction. The height of the root from horizontal plane, the horizontal and vertical velocities of the root are included in this feature vector. The signed direction of the spine link with respect to its upright position is also included in the feature vector.
3. Foot position and velocity. The position and velocity of the feet are included in the feature vector. They are expressed relative to root position and the direction of the spine link defines the up axis.
4. Ground contact. Contact with the environment is an important feature of the motion. This feature vector contains four Booleans indicating the contact (one) or none-contact (zero) of toe and heel regions of both feet.

The above four features are weighted, squared and summed to compute the distance for any motion frame. Table 1 gives an example of the weights used in various motions.

Controllers	# of frames	KNN	Feature weights			
			POS	TOR	ANG	ROO
Stand	50	6	15	2	0.1	1
Balance	790	3	15	2	15	1
BalanceRecover	1200	3	15	2	15	1
NormalWalk	410	6	15	2	0.1	1
NormalStop	186	6	15	2	0.1	1
SoldierStart	182	6	15	2	0.1	1
SoldierWalk	655	6	15	2	0.1	1
SoldierStop	182	6	15	2	0.1	1
Walk2Run	305	6	15	2	0.1	1
Run2Walk	227	6	15	2	0.1	1

Table 1. Motor controller parameters. KNN is the k-nearest neighbors, The feature weights from left to right are feet position (POS), torso up direction (TOR), joint angles (ANG), the y-position and y-velocity of the root (ROO). The weights for other features are 1.

Given a feature vector $F(t)$ of the current simulation state $P(t)$, the controller computes the target state $P(t + \Delta t)$ for the next time step. The controller first searches for the k -nearest neighbor samples (P_1, \dots, P_k) according to their feature vector distances (F_1, \dots, F_k) , and then computes the target state by combining the associated states with weights inversely proportional to the distances:

$$P(t + \Delta t) = \begin{cases} \frac{\sum_i w_i P_i}{\sum_i w_i}, & \text{if } d_i > \varepsilon \text{ for } \forall i, \\ P_{\text{argmin}(d_i)}, & \text{otherwise,} \end{cases} \quad (13)$$

where $d_i = \text{distance}(F(t), F_i)$, $w_i = \frac{1}{d_i}$, and ε is a small constant. If the distance to any sample is below ε , choose the nearest sample in order to avoid division by zero. In order to efficiently index the k -nearest neighbors, data are stored in a kd -tree and ANN library [20] is used in the search.

The controller developed so far has no guarantee that it will produce stationary cycles without falling over. SIMMOCAP proposed an adaptive refinement to prevent the falling over. During the simulation, once an unsuccessful situation is detected, i.e. the model is about to fall or deviates too much from training data set, the system roll back to the time when the nearest cycle of motion begins. At this point, the system selects a cycle of motion that starts with the configuration most close to the current configuration and kinematically blends the current configuration smoothly into this selected clip using displacement mapping. This warped motion is fed to the optimization module to produce the corresponding rectified motion. The rectified motion is added to the training set and it will guide the simulation of the model from then on. In this way, the controller can be refined until no more falling over is observed for the desired simulation.

The transitioning controllers are trained in a similar way as stationary controllers. The training data come from motion capture or blending. Figure 20 shows the transition between controllers. Finally, Figure 21 shows the result of SIMMOCAP bipedal motion simulation.

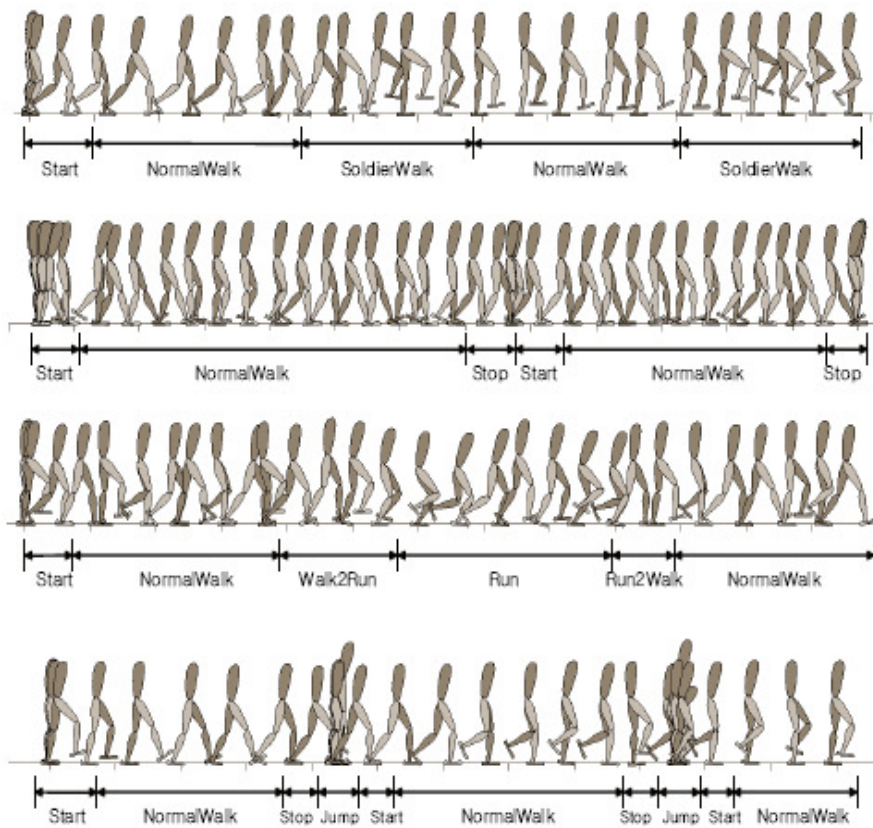


Figure 20. Transition between controllers.

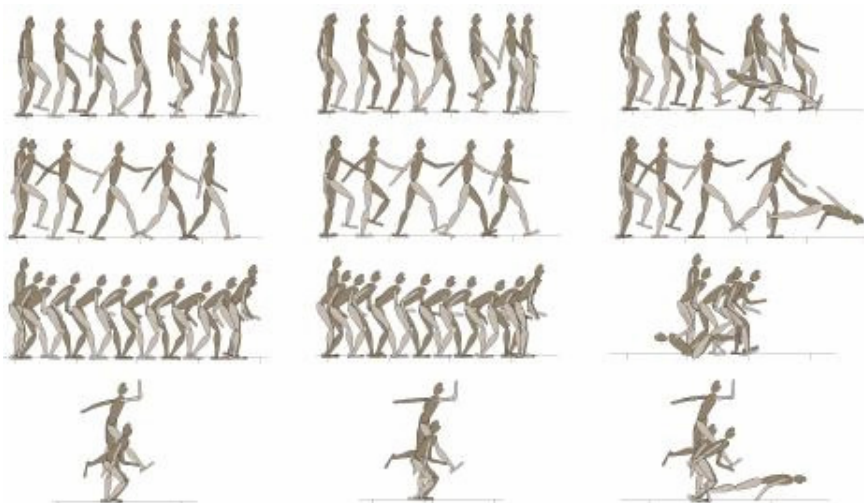


Figure 21. SIMMOCAP results. Left column is original motion capture data. Middle column is the simulation result of rectified motion. Right column is the result of direction simulation of original data, which falls over easily. The four motions are normal walk, soldier walk, sneak walk and jump.

3.4 System comparison

The three systems discussed so far all use PD controllers as their main control method to produce physically-based biped locomotion, yet each proposes a different strategy to make the simulation robust. In this section, we are going to compare their performance in terms of simulation robustness, computational cost and the diversity of motions that they can generate.

LCC solves the balance problem by introducing external perturbations to the base controller to make sure that it follows a nominal trajectory in the state space. The robustness of the system relies on the robustness of the base PCG and the assumption that control perturbations and RV states are linear. However, so far, LCC only provides a trial-and-error way to create the PCGs with heuristics. Moreover, if the motions are highly dynamic, the relationship between control perturbations and RV states is not necessarily linear.

The main computational cost in LCC is spent in sampling simulations and calculating the Jacobian matrix inverse. As the number of DOFs increases to achieve more realistic human modeling, the size of the Jacobian matrix increases as square of that, resulting a significant time increase in matrix inversion.

LCC is limited to generating only cyclic motions like walking and running, because it needs a nominal looping trajectory. It is capable of controlling speed and direction of the locomotion and can introduce some motion variation by interpolating the base PCGs.

SIMBICON solves the balance problem by incorporating balance into the controller design. By separating target angles into world and local coordinates, SIMBICON runs more independently of the current pose and environmental changes. Carefully choosing c_d and c_v in balance feedback is also the key to system robustness. However, similar to LCC, this is again a trial-and-error process.

SIMBICON is computationally efficient with only two dynamic equations to solve in every simulation step. It runs 5 times faster than real-time for 2D simulation and 1.2 times faster than real-time for 3D simulation.

SIMBICON is capable of generating a variety of motions and transiting between them. Also, SIMBICON is capable of simulating the impact of a reasonable external impulse. The impulse changes the current state and the balance controller recovers that modification later on.

SIMMOCAP solves the balance problem by rectifying the motion capture data kinematically via optimization, so that the rectified motion can be tracked in simulation robustly. The robustness relies on the convergence of the optimization method and a good sampling of motion capture data. Even though the paper showed successful cases using the downhill simplex method, there is no explicit analysis on its converging conditions.

SIMMOCAP is the most computational expensive among the three systems. To rectify a 400-frame walk motion, it takes about 15 minutes. Its adaptive refinement strategy, which improves the robustness of the system, requires generating and rectifying interpolated motion on the fly. These operations isolate the system from real-time applications. Notice that all the experiments of SIMMOCAP are in 2D, which limits the optimization to a much smaller space.

SIMMOCAP is capable of generating a variety of motions and transiting between them. However, in order for the system to produce realistic tracking, it requires a good sampling of motions in configuration space. It would be difficult to imagine that the controller trained with walking motion could simulate hopping motion naturally.

In summary, table 2 shows the comparison among the three systems in locomotion related bipedal simulation with a 3-star (best) rating.

System Name	Key Idea	Robustness	Computational Efficiency	Motion Diversity
LCC	Nominal cyclic motion Control perturbation	★	★★	★
SIMBICON	Balance controller	★★★	★★★★	★★★★
SIMMOCAP	Rectify motion capture data	★★★	★	★★★★

Table 2. System comparison on a 3-star (best) scale.

4. Conclusion and Future Work

Physically-based human animation is a difficult task. The complex human body structure lives in a high dimensional space and there is no simple general control technique to model it. Having been observers of human motions throughout our lives, humans are experts in distinguishing different types of human motions, which sets a high criterion for producing realistic natural human motions. Apart from the dynamics that are governed by laws of physics, realistic human motions have nuances, like style, which are very important to its naturalness. Even though attempts have been made to incorporate them into human motion generation [11, 17], no general solution exists so far.

PD controller based approaches discussed in this survey simplifies the problem by working with a human model with reduced DOFs and actuating joints using PD controllers to track the desired poses. Each of the three systems in this survey came up with a novel solution to the problems in PD controller based approaches and produced robust human motion simulation. Their advantages and disadvantages give user room for choices depending on the type of the application.

However, even the state-of-art PD controller based simulations lack desired motion smoothness and naturalness. None of the systems shows sound robustness in their simulation either. In order to make PD controller based human simulation more popular and robust, the following questions need answering.

1. More than locomotion: None of the systems demonstrated the capability of performing interactive motions like catching a ball, moving an object. These high non-cyclic motions often deviate largely from the nominal control or motion capture data. This presents a challenge to PD controller based simulation systems.
2. Simple local controller: In the three systems, most of the PD controllers are simple local controllers that track desired joint angles without considering what other controllers are doing. However, the dynamic system of human motion is much more complex and requires coordination in order to function naturally.

Therefore, there is need for explicit modeling of such coordination among the PD controllers.

3. **Stability condition:** In LCC, the stability of simulation relies on tracking nominal trajectory by actively applying perturbations. In SIMBICON, the balance control relies largely on the desired torso angle solved from equation 9. If these conditions are not satisfied, the character will fall and the simulation has to be reset. There is need for predicting such stability conditions and avoid this from happening. SIMMOCAP addressed this issue by adaptive refinement. However, given the extremely complex nature of their objective function, the optimization stability condition is in question and was not analyzed in the paper.
4. **Scalability:** Ultimately, we would like to simulate a character with many more DOFs. In that case, either LCC or SIMMOCAP will suffer from their expensive computational cost. SIMBICON also needs to modify their balance strategy to take more joints into account. It is unclear yet if a similar simple feedback equation exists to produce reasonable results. Further more, in all the three systems, the parameters are manually tuned. As the complexity of the system increases, there is need for automatic parameter tuning.
5. **Tracking:** A more philosophical question arises with PD controller base simulation. In its nature, PD controllers track the desired joint angles. However, in real life, do we really move by tracking some nominal motions in our mind?

In summary, there is still plenty of room for research work to be done in order to make PD controller based simulation more robust and produce more life-like human motions. One of the possible approaches is to explore the intersection between physical simulation and motion capture data. Instead of modifying motion capture data for simulation, like SIMMOCAP did, one may consider simulating the simple bipedal model first and extracting mass center and end effectors position and orientation information, and then mapping them to a corresponding animation sequence. Methods like Motion Graph, IK, and interpolation can be used to modify the animation to meet the joint requirements from the simulation for physic realism. Meanwhile, the details of the human motion from motion capture are also preserved.

References

- [1] O. Arikan, D. Forsyth, and J. O'Brien 2003. Motion synthesis from annotations. In *Proceedings of the ACM Trans. Graph.* ACM, 402-408.
- [2] N. Badler, R. Bindiganavale, J. Granieri, S. Wei, and X. Zhao 1994. Posture interpolation with collision avoidance. In *Proceedings of the Proceedings of Computer Animation*, Geneva, Switzerland, pp. 13-20.
- [3] M. Cohen 1992. Interactive spacetime control for animation. In *Proceedings of the SIGGRAPH 92*, 293-302.
- [4] P. Faloutsos, M. van de Panne, and D. Terzopoulos 2001. Composable Controllers for Physics-Based Character Animation. In *Proceedings of the ACM SIGGRAPH 2001*, 251-260.
- [5] P. Faloutsos, M. van de Panne, and D. Terzopoulos, 2001. The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers and Graphics* 25 933-953.
- [6] A. Fang, and N. Pollard 2003. Efficient synthesis of physically valid human motion. In *Proceedings of the ACM Trans. Graph.* ACM, 417-426.
- [7] A. Garcia-Rojas, 2005. An Ontology of Virtual Humans: Incorporating Semantics into Human Shapes, The 2nd European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (EWIMT 2005), pp. 7- 14.
- [8] M. Girard, and A.A. Maciejewski 1985. Computational modeling for the computer animation of legged figures. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* ACM, 263-270.
- [9] J. Hodgins, and N. Pollard 1997. Adapting Simulated Behaviors For New Characters. In *Proceedings of the SIGGRAPH 97*, 153-162.
- [10] J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien 1995. Animating Human Athletics. In *Proceedings of the SIGGRAPH 95*, 71-78.
- [11] E. Hsu, K. Pulli, and J. Popovi 2005. Style translation for human motion. In *Proceedings of the ACM Transactions on Graphics*, 1082-1089.
- [12] IKAN, Inverse Kinematics using ANalytical Methods, <http://cg.cis.upenn.edu/hms/software/ikan/ikan.html>
- [13] L. Kovar, 2004. Automated methods for data-driven synthesis of realistic and controllable human motion, Ph.D. Thesis, University of Wisconsin-Madison.
- [14] L. Kovar, M. Gleicher, and F. Pighin 2002. Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* ACM Press.
- [15] J. Laszlo, 1996. Controlling bipedal locomotion for computer animation, Master's Thesis, University of Toronto.
- [16] J. Laszlo, M. van de Panne, and E. Fiume 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* ACM Press.

- [17] K. Liu, A. Hertzmann, and Z. Popovi 2005. Learning physics-based motion style with nonlinear inverse optimization. In *Proceedings of the ACM Transactions on Graphics*, 1071-1081.
- [18] K. Liu, and Z. Popovi 2002. Synthesis of Complex Dynamic Character Motion from Simple Animations. In *Proceedings of the ACM Transactions on Graphics*, 408-416.
- [19] A. Maciejewski, 1990. Dealing with the Ill-Conditioned Equations of Motion for Articulated Figures. *IEEE Computer Graphics & Applications* 10 63-71.
- [20] D. Mount, and S. Arya, Ann: Library for approximated nearest neighbor searching, <http://www.cs.sunysb.edu/algorithm/implement/ann/distrib/index1.html>
- [21] J. Popovi, S. Seitz, M. Erdmann, Z. Popovi, and A. Witkin 2000. Interactive Manipulation of Rigid Body Simulations. In *Proceedings of the ACM SIGGRAPH 2000*, 209-218.
- [22] Z. Popovi, and A. Witkin 1999. Physically Based Motion Transformation. In *Proceedings of the Proceedings of SIGGRAPH 99*, 11-20.
- [23] C. Rose, M. Cohen, and B. Bodenheimer, 1998. Verbs and Adverbs: Multidimensional Motion Interpolation, *IEEE Computer Graphics and Applications*, pp. 32-41.
- [24] A. Safonova, J. Hodgins, and N. Pollard 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *Proceedings of the ACM Transactions on Graphics*, 514-521.
- [25] K.W. Sok, M. Kim, and J. Lee 2007. Simulating Biped Behaviors from Human Motion Data. In *Proceedings of the 34th annual conference on Computer graphics and interactive techniques* ACM Press.
- [26] D. Tolani, A. Goswami, and N. Badler, 2000. Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs. *Graphical Models* 62 353-388.
- [27] C. Welman, 1989. Inverse kinematics and geometric constraints for articulated figure manipulation, Master's thesis, Simon Fraser University.
- [28] D. Wiley, and J. Hahn, 1997. Interpolation Synthesis of Articulated Figure Motion, *IEEE Computer Graphics and Applications*, pp. 39-45.
- [29] A. Witkin, and M. Kass 1988. Spacetime Constraints. In *Proceedings of the SIGGRAPH 88*, 159-168.
- [30] W. Wooten, and J. Hodgins 1995. Simulation of Human Diving. In *Proceedings of the Graphics Interface '95*, 1-9.
- [31] K. Yin, K. Loken, and M.v.d. Panne 2007. SIMBICON: Simple Biped Locomotion Control. In *Proceedings of the 34th annual conference on Computer graphics and interactive techniques* ACM Press.
- [32] V. Zordan, A. Majkowska, B. Chiu, and M. Fast 2005. Dynamic response for motion capture animation. In *Proceedings of the SIGGRAPH '05* ACM, 697-701.