DOI: 10.1111/cgf.13134

EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Performance-Based Biped Control using a Consumer Depth Camera

Yoonsang Lee[1] and Taesoo Kwon[2]

[1]Kwangwoon University [2]Hanyang University

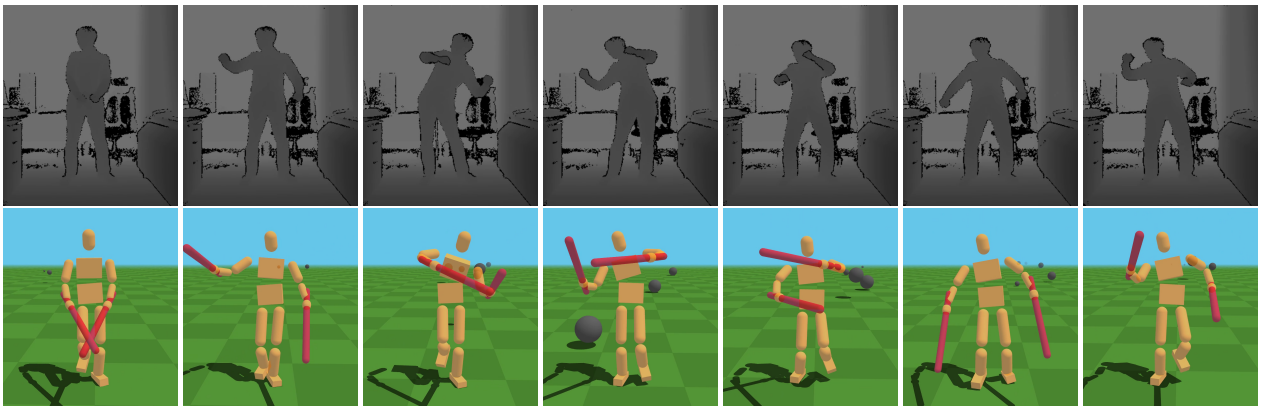

Figure 1: *A user controls a physically simulated character by moving his or her body parts. Top: The input depth data of user poses. Bottom: The simulated character imitating the user's actions.*

Abstract

We present a technique for controlling physically simulated characters using user inputs from an off-the-shelf depth camera. Our controller takes a real-time stream of user poses as input, and simulates a stream of target poses of a biped based on it. The simulated biped mimics the user's actions while moving forward at a modest speed and maintaining balance. The controller is parameterized over a set of modulated reference motions that aims to cover the range of possible user actions. For real-time simulation, the best set of control parameters for the current input pose is chosen from the parameterized sets of pre-computed control parameters via a regression method. By applying the chosen parameters at each moment, the simulated biped can imitate a range of user actions while walking in various interactive scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

The development of motion sensing input devices has facilitated the advance of performance-based control techniques. The Wiimote, an accelerometer-based game controller introduced in 2006, was commercially successful and has had a large impact on how games are played. Recent devices such as Microsoft Kinect and Leap Motion provide users with more freedom of movement and controllability by tracking their bodies and hands. These devices have been adopted by many games, which offer new levels of interactivity, and enhance the gameplay experience.

Another important factor for an immersive experience is physical realism. Physics simulations of ragdolls, collisions, and explosions are widely adopted in games to allow natural interactions between the entities, the users, and the virtual world. Controlling physically simulated bipeds is a promising technique because it has the potential to bring physical realism into the motions of game characters, although such a possibility has yet to be explored thoroughly in current games. Combining these two factors, the performance-based control of a simulated biped would be a powerful tool for creating immersive games and VR applications.

However, the output motion from current off-the-shelf depth cameras is somewhat noisy and inaccurate to be directly used for physically simulated characters. Output motion is particularly

problematic for locomotion control because it often requires reliable contact information for both left and right feet, as well as coordinated movement from the whole-body to be accurately captured. Also, consumer depth cameras are designed to work in a limited space, such as a small room. Even walking a few steps at normal speed is not easy in the space.

We propose a performance-based biped control system that allows users to make their characters walk forward and mimic their actions. Our controller takes a real-time stream of user poses from a depth camera and synthesizes a stream of target poses based on it. By tracking the target poses, the biped can imitate user actions while moving forward at modest speed and maintaining balance. The controller directly transfers the upper-body portion of the input pose to the target pose to reproduce the user's upper-body movement. At the same time, our lower-body controller computes the target angles of leg joints which allow the biped to maintain balance and move forward, even though the user is in a small space. This is a challenging problem because a controller that works for one input motion is not guaranteed to work robustly for a different and unpredictable input motion. Naturally, users would want to change their pose freely in interactive control scenarios, for example, by bending their knees and changing their upper-body poses to avoid obstacles or to attack enemies.

To deal with this problem, we parameterize our controller over a set of modulated reference motions. Each modulated reference motion is obtained by editing a single normal walking motion, and a set of modulated reference motions are generated to cover the range of possible user actions. A set of ten control parameters is optimized for each of modulated reference motions. During real-time simulation, our system calculates a set of parameters for the current input pose using a regression method with the precomputed controller parameters. This parameterization enables a stable biped walking under modest but unpredictable variations of user actions. We demonstrate the effectiveness of our approach by comparing the parameterized controller and the baseline controller without parameterization, and several interactive control demos.

2. Related Work

Computer puppetry systems can produce high-quality animations based on a performer's motions in real time. A computer puppetry system works by mapping a user pose to a corresponding character pose in an on-line manner using various devices that track human motions. Some research results focused on producing high quality animations for movies and broadcast performances [Stu98, SLSG01] using optical or magnetic mocap devices. Such performance-based motion prototyping has been proven effective especially for novice animators who do not have the necessary skills and time for key-framing. Others studied the potential of low-cost input devices to interactively control human characters [OTH02, LCR*02, TBvdP04, JPG*14]. Such approaches were generally better suited for virtual reality applications and game-like scenarios. Oore et al. presented a tangible input device and interface for intuitive control of characters having more degrees of freedom than those of the input device [OTH02]. Yin et al. studied an animation interface that uses a foot pressure sensor pad to interactively control avatars [YP03]. Terra and Metoyer proposed an approach that allows the user to perform key-frame timing using simple 2D input devices such as a mouse or a stylus [TM04]. Chai and Hodgins employed video cameras and a small set of retro-reflective markers in stereo video, and reconstructed 3D poses using a local linear model to build a low-cost mocap system [CH05]. Grochow et al. presented an inverse kinematics solver based on a Gaussian process latent variable model (GPLVM) learned from human poses, and demonstrated its robustness using a real-time motion capture system with missing markers [GMHP04]. Slyper and Hodgins created a performance animation system that uses five low-cost accelerometers sewn into a shirt. The low-dimensional and noisy signals from the system were supplemented by a database of captured human motions to animate the upper-body of an avatar according to the motion of the performer [SH08b]. Tautges et al. used four accelerometers attached to the hands and feet of a human actor, and a large number of captured motion sequences to reconstruct the full-body motion of the performer using a lazy-learned neighborhood graph [TZK*11].

Using natural user interface to control characters is becoming more widely accepted since the advent of many consumer-grade, real-time motion capture devices such as Microsoft Kinect, Leap Motion, Nintendo Wiimote and many others. Some researchers focused on creating applications to extend the capabilities of such devices. Seol et al. presented a real-time motion puppetry system to animate non-human characters with various skeleton structures using human motion input [SOL13]. Rhodin et al. estimated wave properties such as amplitude, frequency, and phase from user motions, and used these properties to robustly control various characters for common cyclic animations from a sparse set of example motions [RTIK*14, RTK*15]. Vögele et al. presented a method to interactively generate detailed mesh animations using a statistical shape model learned from existing mesh sequences based on inputs from a Kinect sensor [VHKK12].

Physics simulation has also been frequently used for interactive character animation, because of its potential for truly responsive and realistic animation [HWBO95, LvdPF00, ZvdP05, YLvdP07, dSAP08, MLPP09, MZS09, CBvdP09, dLMH10, MdLH10, KH10, YL10, WP10, MPP11, ABDLH13, HL14, HRL15, LvdPY16]. The physics-based control of characters is generally regarded as more difficult because of the inherent instability of the floating-base underactuated system. The instability, however, is a key ingredient for reproducing natural responses to perturbations due to changes in the environment. Many recent publications on physically simulated characters demonstrated dramatic improvements in motion quality, versatility, and robustness. Similar to our work, some of them used feed-back controllers for simulated characters to track captured reference motions [SKL07, dSAP08, MLPP09, LKL10, KH10, LPKL14, LvdPY16]. However, these controllers were demonstrated using only low DOF input devices such as keyboards and joysticks, and it is not clear whether the results can be generalized to high-dimensional, free-form inputs from natural user interfaces.

Our work is most closely related to approaches for constructing reusable parameterized controllers. Regression models have been used to map between states, tasks and control parameters, and allowed the development of robust controllers for bipeds walking

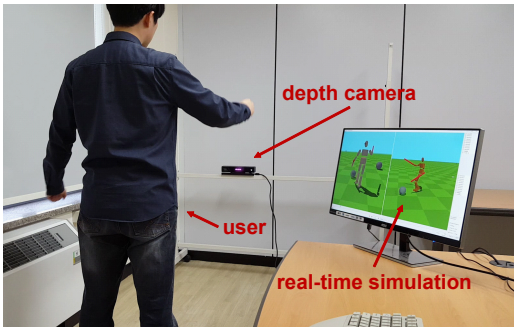

Figure 2: *The depth camera in front of the user captures the user's pose and our system controls a simulated biped as shown on the monitor.*

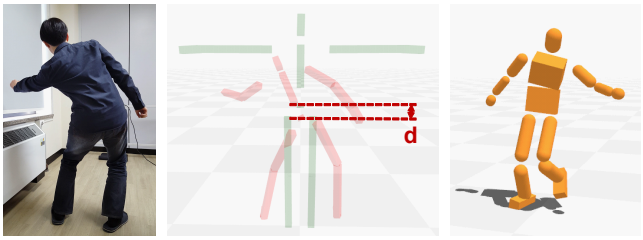

Figure 3: *The user pose (left) is converted to the target pose (right). Left: A user is bending his knees while moving his arms. Center: The difference in the pelvis height (d) of the user pose (red) and the upright-standing pose (green) is applied to the lower-body part of the target pose. Right: The resulting target pose makes the simulated biped mimic the user's upper-body pose and knee-bending while walking.*

over terrain [LYvdPG12] and performing various stunts [LvdPY16] and quadrupeds jumping over obstacles [CKJ*11] as well as humanoid robots [dSBK*14,HL16]. We also use a similar method for performance-based control.

Below, we briefly survey prior works relevant to performance-based control of simulated avatars. Ha et al. demonstrated that a wide range of natural motions can be reconstructed using ground reaction forces from force sensors and arm motions from a hand tracking device [HBL11]. Ishigaki et al. introduced a performance-based control interface for creating physically-plausible motions that preserve the style of example motions using dynamic simulation of a simplified rigid body model [IWZL09]. However, the model lacked the ability to maintain balance, and the pitch and roll were controlled using an imaginary force when necessary. Nguyeny et al. proposed a framework that unifies kinematic playback of motion capture and dynamic simulation for interactively controlling complex interactions [NWB*10]. However, a trade-off between physical correctness and kinematic control must be made. Vondrak et al. recovered biped controllers capable of implicitly simulating the human behavior observed from video and replaying it under other environments [VSHJ12]. Shiratori et al. used two sets of Wiimotes, and mapped the amplitude, phase and frequency of the sensor inputs to a physically simulated character [SH08a]. Three performance interfaces (legs, wrists and joystick interface) were tested on a set of tracks containing turns and pits. Our goal is different from this work in that a user is provided with direct, free-form control of upper-body motion, as well as indirect control of the lower-body based on the motion of the performer.

3. Controller

Our controller takes a stream of real-time user poses from a depth camera as input (Figure 2). Based on the input poses, it synthesizes a stream of target poses and calculates joint torques to track the target poses. The target poses are generated at the same rate as the input (30 Hz), and the joint torques are computed at a higher rate (900 Hz). A biped character is physically simulated by integrating the joint torques using a forward dynamics simulator.

The target poses need to be generated in a way that the biped can mimic user actions while walking and maintaining balance simply by tracking them. To build a target pose from each input pose, the upper-body joint orientations are first copied from the input pose into the target pose for freeform manipulation of the upper-body. The time-varying upper-body poses can cause changes in the center of mass (CM) position of the character and possible instability associated with the changes. Thus, the remaining degrees of freedom are used for balancing the character.

Specifically, a stepping strategy is used to regulate the CM of the character. We choose to automatically generate the stepping motions based only on the captured pelvis height, and ignore other aspects of the captured lower body motion. This simple approach works well, and the user does not need to walk around the limited capture space from the user-interface point of view. Some users may prefer to directly use the captured whole-body motion, but the inherent discrepancies in the underlying dynamics between the user and the rigid-body biped model would make the control problem unnecessarily difficult. Also, the contact information from the input data is not reliable due to the input noises. Even a small delay in foot contact timing can easily disrupt the locomotion stability. There can be some noises in the upper-body motion captured from the depth camera, but they have relatively small effects on simulation stability in our experiments.

Our lower-body control algorithm closely resembles an existing algorithm proposed by Lee et al. [LKL10]. The controller in [LKL10] takes reference motion data as input, and modifies it to maintain balance by using a few feedback rules. It also synchronizes the contact timings of the reference and the simulated motions. We slightly modify the controller so that the knee bending angles can be adjusted to imitate a user. A captured normal walking motion is used as the reference motion for the lower-body part of the character (*base motion*). To reflect a user's knee bending, we measure the height of the pelvis from the ground for each real-time input pose (Figure 3). The difference between the pelvis height of the input pose and the precomputed upright-standing pose is applied to the current pose of the *base motion* using an analytic inverse kinematics solver. Note that we use the same skeletal structure for the captured reference pose and the input pose, and thus applying the offset to the reference pose does not cause any problems. Then, the target pose is built by combining the upper-body portion

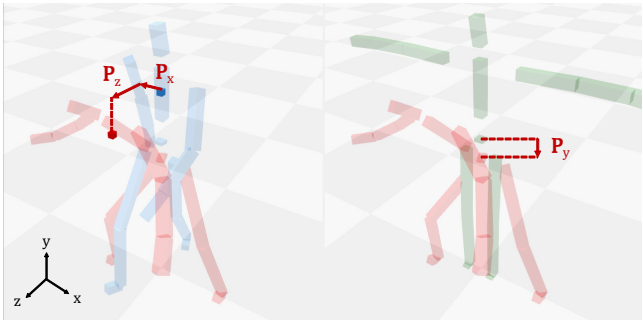

Figure 4: *The parameters for controller parameterization. Left:* $\mathbf{P}_x$ *and* $\mathbf{P}_z$ *are the horizontal offset of the upper-body CM of the input pose (red cube) from that of the current pose of the base motion (blue cube). Right:* $\mathbf{P}_y$ *is the offset of the pelvis height of the input pose (red) from that of the standing pose (green).*

of the input pose and the lower-body portion of the current lower-body reference pose. As a result, if a user bends his or her knees, the biped similarly bends its knees while walking. Once a target pose is generated, our controller calculates all the joint torques for tracking the target pose. The algorithm for computing the joint torques is almost identical to that in [LKL10].

4. Parameterization

Our controller uses ten feedback parameters θ that determines how to move each leg in a given situation to maintain balance [LKL10]. Although one set of the parameter values works robustly for small variations in the reference motion to be tracked, the parameters needs to be adjusted depending on the style of the reference motion. For example, the parameters are hand-tuned for each reference motion in the original work.

However, we cannot use the same approach because there is no fixed reference motion in our case. The user in front of a depth camera can move freely, for example, swing his or her arms, and the next action is unpredictable. Even though the control algorithm takes into account the changes in CM position due to the upper-body movement, a single set of control parameters cannot cover enough variations in the upper-body poses. A set of hand-tuned or optimized parameters for one reference action works well only for user actions that are similar to the reference action. To deal with a wider range of input poses, we parameterize our controller so that a parameter set corresponds to a style of user action. We prepare a repertoire of possible user actions, and assign a parameter set for each style.

4.1. Parameters

Parameters $\mathbf{P}$ for controller parameterization should be carefully selected to compactly represent the possible space of user actions. The dimensionality of the parameter space should be low to avoid the curse of the dimensionality problem that occurs when there are insufficient training data. In our case, the process of preparing the training data is time-consuming because it involves a complex non-linear optimization.

We choose the following parameters (Figure 4):

- **Horizontal offset of the upper-body center of mass (CM) ($\mathbf{P}_x$, $\mathbf{P}_z$)** of the user with respect to that of the current pose of *base motion* aligned horizontally with the pelvis position of the user. It aims to summarize the user's various upper-body actions in terms of balance control. The horizontal CM position is important for balanced walking because it directly affects footstep locations. We use only the CM of the upper-body parts because the lower-body CM of the simulated biped is mostly determined by the lower-body controller. The *xy*-plane is the lateral plane, and the *yz*-plane is the sagittal plane of our simulated biped.

- **Vertical offset of the pelvis ($\mathbf{P}_y$)** of the user with respect to the pelvis height of the upright-standing pose. It aims to compactly represent the user's lower body motion. As stated in the previous section, our lower-body controller can adjust the pelvis height of the biped according to that of the user.

A parameter set $\mathbf{P}^i = (\mathbf{P}^i_x, \mathbf{P}^i_y, \mathbf{P}^i_z)$ represents a reference motion of which upper-body CM and pelvis height are expressed by $\mathbf{P}^i$ (Figure 5). To find a set of optimal feedback parameters θ^{i*} for an arbitrary $\mathbf{P}^i$, each reference motion that corresponds to $\mathbf{P}^i$ is constructed starting from the *base motion*. Specifically, we modify the joint orientations of the *base motion* to match a given $\mathbf{P}^i$. We apply a Jacobian transpose inverse kinematics method to upper-body joints to adjust the upper-body CM position of a reference pose, and the orientations of leg joints are modified to vertically offset the pelvis via an analytic inverse kinematics solver. This editing process is conducted for every frame of the *base motion* and the resulting reference motion represents the given $\mathbf{P}^i$.

4.2. Sampling the Parameter Space

To prepare sets of controller parameters θ, we first need to sample the space of parameter $\mathbf{P}$. A regular grid of points in the space is created to synthesize a set of reference motions. An optimization process searches for the best feedback parameters θ^{i*} for each point $\mathbf{P}^i$, which is described in the next section.

The size of the sampling grid defines the range of possible user actions to be covered by our controller. We need to choose a reasonable range for each dimension of $\mathbf{P}$. Large values of $\mathbf{P}_x$, $\mathbf{P}_y$, and $\mathbf{P}_z$ correspond to walking with excessive knee bending and upper-body stooping, which is unlikely to appear in the user input.

We use $[0, 0.2]$, $[-0.2, 0]$, and $[-0.2, 0.2]$ (in the meter unit) for the ranges of $\mathbf{P}_x$, $\mathbf{P}_y$, and $\mathbf{P}_z$, respectively (Figure 5). The range of $\mathbf{P}_x$ contains only non-negative values because we use the same set of feedback parameters for a negative-valued $\mathbf{P}_x$ which indicates that the lateral leaning of the upper-body has the same magnitude but in the opposite direction. This greatly reduces the number of samples, and shortens the computation time. We sample three points in the x and y directions, and five points along the z direction, and thus the total number of points is $3 * 3 * 5 = 45$.

We first sort the 45 sample points based on the Euclidian distance from the origin $\mathbf{P}^0 = (0, 0, 0)$. Starting from the nearest point to $\mathbf{P}^0$, we sequentially optimize the sets of feedback parameters

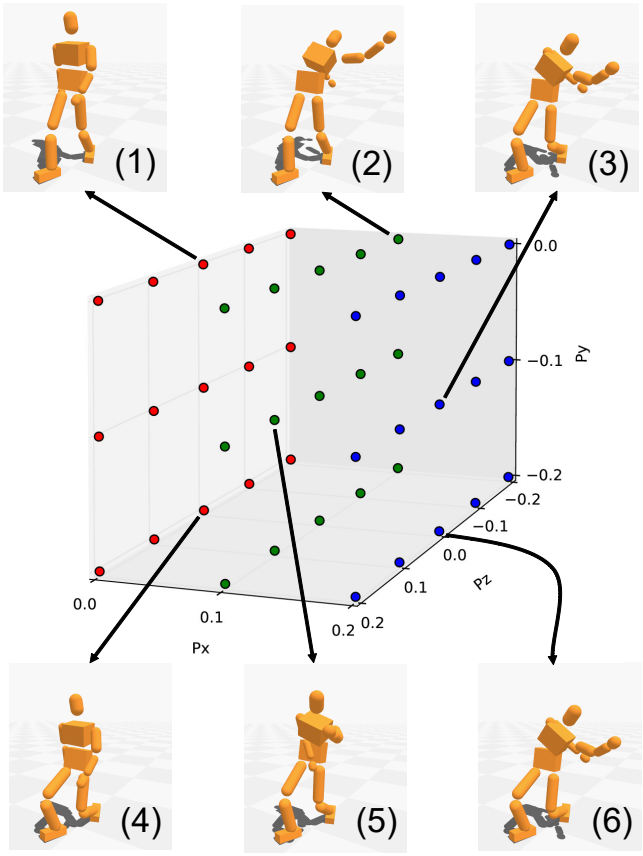

Figure 5: *Parameterization space. Each point* $\mathbf{P}^i$ *in the space indicates a reference motion used in the optimization process. The modulated reference poses (1), (2), (3), (4), (5), and (6) correspond to the first frames of the reference motions indicated by the sample parameters* $\mathbf{P}^1 = (0,0,0)$, $\mathbf{P}^2 = (0.1,0,-0.2)$, $\mathbf{P}^3 = (0.2,-0.1,0)$, $\mathbf{P}^4 = (0,-0.2,0)$, $\mathbf{P}^5 = (0.1,-0.1,0.1)$, *and* $\mathbf{P}^6 = (0.2,-0.2,0)$, *respectively. The points are rendered using three different colors based on their x values for a clean visualization.*

θ^i for each grid point $\mathbf{P}^i$ in the sorted order. Finding robust controller parameters for $\mathbf{P}^i$ farther from $\mathbf{P}^0$ is more challenging than for those that are closer to $\mathbf{P}^0$ because the former requires walking with a more pronounced upper-body lean and crouched knees. To improve the optimization performance, we provide initial guesses θ^{i0} for each optimization of $\mathbf{P}^i$ using other parameters θ^{j*} that have already been optimized. We choose an initial guess θ^{i0} from θ^{j*} of nearby k_1 points via a k-nearest neighbor regression.

Even when the size of the parameterization space is reasonable, the optimizer may fail to find robust feedback parameters for a sample point. In such a case, we use a heuristic search algorithm that repeatedly scales the distance between the sample point and the origin by 0.8 until a set of feedback parameters is successfully found.

5. Optimization

Given a new reference motion represented by $\mathbf{P}^i$, the controller feedback parameters θ^i are optimized by the objective function proposed in [LLK*15]:

$$E(\theta) = w_1 E_{\text{balance}} + w_2 E_{\text{track}} + w_3 E_{\text{param}}, \quad (1)$$

where E_{balance} penalizes the falling down of the biped, E_{track} penalizes the deviation of the simulated poses from the reference poses, and $E_{\text{param}} = \|\theta\|$ penalizes large parameter values to prevent abrupt changes in reference motions, which often make the simulation unstable. We set the weight values $w_1 = 2000$, $w_2 = 1.0$, and $w_3 = 1.0$.

An evaluation of the objective function requires a simulation of 20 seconds in duration: ten seconds for the given reference motion and ten seconds for the mirrored motion in the sagittal plane of the biped. Evaluating both the reference motion and the mirrored motion allows us to handle the left and right leaning of the upper-body symmetrically using a single set of optimized parameters θ^{i*}. It also enhances the robustness of the controller by optimizing for the two different initial conditions, starting with the left foot and the right foot. The result of the objective function is set as the sum of the results for the first ten seconds and the next ten seconds.

We use a Covariance Matrix Adaptation (CMA) method to find a set of optimal parameters θ^{i*}. The size of the CMA population is 24. The optimization for each $\mathbf{P}^i$ is terminated when the number of iterations exceeds 200 or the result of the objective function is smaller than 2000, to shorten the computation time. An evaluated value larger than 3000 is regarded as a failure case, and the heuristic search algorithm is executed as described in Section 4.2.

6. Real-time Simulation

During the real-time simulation, a user stands up in front of a depth camera and freely move his or her body parts (Figure 2). Our controller generates target poses based on the input poses to imitate the user's actions in the simulated environment, as described in Section 3. Only a carefully selected set of feedback parameters leads to robust walking of a simulated biped.

We compute feedback parameters θ^{cur} at the input pose frequency. First, the parameterization parameter $\mathbf{P}^{\text{cur}}$ is extracted from the current input pose by measuring the position of the upper-body CM and the height of the pelvis. Then, we compute θ^{cur} using a multivariate regression of the precomputed controller parameters. Specifically, precomputed parameters θ^{j*} of k_2 nearest points are interpolated using a k-nearest neighbor scheme with inverse distance weighting (IDW). The resulting θ^{cur} makes the biped imitate user poses while walking and maintaining balance.

7. Results

We use a Microsoft Kinect (version 2) in our experiments. A user input pose is built from global joint orientation data acquired using Kinect for Windows SDK 2.0. The frame rate of both the input poses for real-time simulation and the reference motions used in the optimization are 30 Hz. Although a new pose is mostly obtained from the Kinect camera within 1/30 seconds, we reuse the same

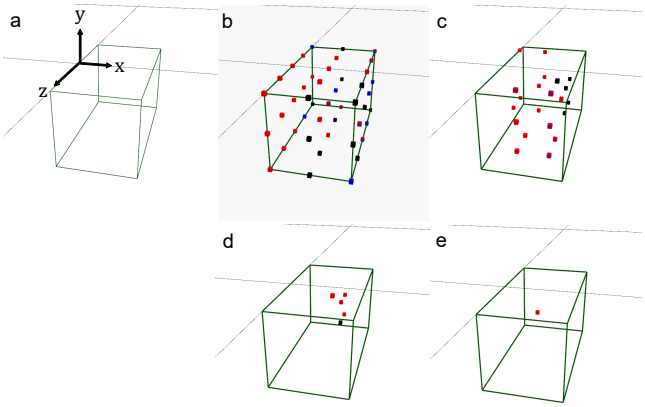

Figure 6: *Optimization of parameter space. The range of the initial parameter space is rendered as the green cube (a). Minimum evaluated values of the first 45 sample points (b), of 21 sample points scaled by* 0.8 *(c), of 5 sample points scaled by* 0.8^2 *(d), of 1 sample point scaled by* 0.8^4 *(e). Successful points (evaluated value from* 2000 *to* 3000*) are rendered in color from red (value of* 2000 *or less) to purple (up to value of* 3000*) and failed points (evaluated value more than* 3000*) are rendered in color from blue (value of* 3000*) to black (value of* 10000 *or more).*

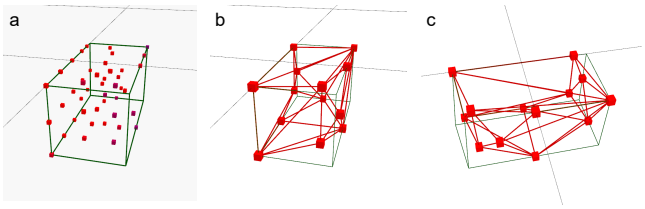

Figure 7: *"Shrunk" parameter space. (a) All successful 45 sample points. (b)(c) The convex hull of the successful points.*

Figure 8: *Comparison of the parameterized controller (row 1, 3) and the baseline controller (row 2, 4). Row 1, 2: Simulation of the reference motion for* $\mathbf{P} = (0.05, -0.05, -0.15)$*. Row 3, 4: Simulation of the reference motion for* $\mathbf{P} = (0.05, -0.05, 0.05)$*.*

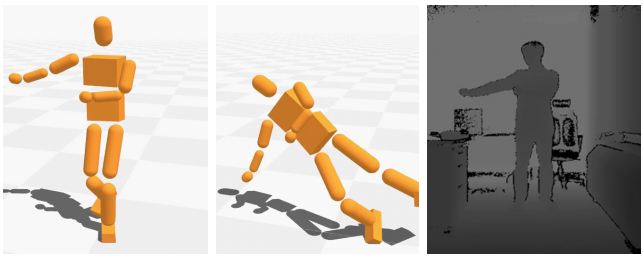

Figure 9: *Comparison of controllers with real-time pose input. Left: The parameterized controller. Middle: The baseline controller. Right: The input depth data of the real-time user pose.*

input pose from the previous frame when the input pose is delayed occasionally.

We use GEAR [GEA] to simulate our biped character with a simulation time step of $1/900$ seconds. The number of neighbors for both the initial guess of the optimization (k_1) and the real-time interpolated parameters (k_2) are 4. Using 12 cores on a Xeon-based workstation (Intel X5650@2.67 GHz), the parameter optimization for all 45 sample points takes approximately 35 hours, including the time spent in the heuristic search algorithm for failure cases.

Shape of the Optimized Parameter Space. The optimizer tries to find the controller parameters for all sample points in the range of $[0, 0.2]$, $[-0.2, 0]$, and $[-0.2, 0.2]$ of the parameter space, but sometimes it fails and resorts to the heuristic search algorithm. Because we repeatedly scaled down the distance between such samples and the origin as stated in Section 4.2 (Figure 6), the final parameter space is smaller than the initial one (Figure 7). We visualize the successful points as a convex hull to show the overall shape of the final space that can be covered by our parameterized walking controller. Note that the convex hull is used only to visualize the shape of the space, and it does not guarantee a successful simulation of a sample point in it. Figure 7 shows that our controller is more unstable when the user leans the upper-body backward, or when the user leans and bends knees at the same time.

Comparison of Parameterized and Baseline Controller. We compared the robustness of our parameterized controller and the controller optimized only for the *base motion* (*baseline controller*).

In the first comparison, we used the automatically generated reference motions corresponding to multiple sets of parameters $\mathbf{P}$ as the input (Figure 8). The reference motions contained various combinations of upper-body leaning and knee bending. We found the parameterized controller can deal with a much wider range of modulated reference motions. For example, the parameterized controller generated stable walking simulation for a motion with $\mathbf{P} = (0.05, -0.05, -0.15)$, but the baseline controller failed even for a motion with a smaller change: $\mathbf{P} = (0.05, -0.05, 0.05)$.

Our second comparison used real-time user poses from a depth camera as input (Figure 9). The baseline controller easily failed when the user simply moved their arms to the right side or slightly bent their knees. The parameterized controller maintained balance

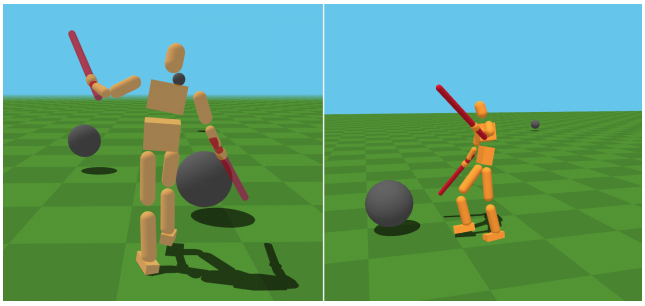

Figure 10: *Demo game #1. A user need to block or avoid rolling stones to survive.*

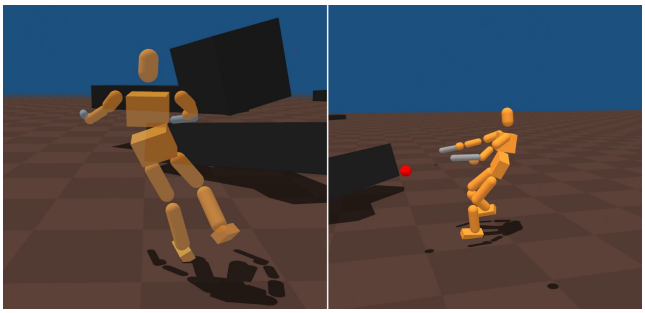

Figure 11: *Demo game #2. A user has to use their guns to protect themselves from the boxes.*

during the more complex movement of the user's upper-body and knees.

Demo Games. The performance-based interface and physical simulation can improve immersion in the virtual environment and the gameplay experience. We designed two simple game demonstrations to make the best use of our performance-based biped controller.

In the first game, a user needs to survive from the endless stones rolling at the game character (Figure 10). The user can avoid stones by leaning the body or blocking the stones using the two sticks in the character's hand. Each stick is 80 cm long and weighs 0.1 kg. The stones are randomly generated with a radius range of $[5,30]$ cm, a mass range of $[0.02,4]$ kg (which is proportional to their volume), and an initial speed range of $[5,10]$ m/s. Without any additional game elements such as other skills or items, the game demonstrates its own gameplay using performance-based control and a physics simulation.

In the second game, the character encounters huge boxes randomly thrown toward it (Figure 11). The user can shoot guns to push away the boxes. We use the Kinect's grasp detection to shoot the gun. The speed of a bullet is 20 m/s, and a box weighs between 1 kg to 20 kg randomly. The games are controlled based on user performance and optionally, hand gestures, and are actually fun to play.

Computation Time. Because the entire optimization process includes a large number of time-consuming optimizations, we carefully determined the terminating criteria for CMA optimization to reduce the computation time. The evaluated value of the objective function (Equation 1) 2000 was experimentally chosen to make walking simulation almost as stable as when the optimal objective value was used. In most cases, stability improvement was negligible below that value despite consuming too much computation time and the value was found in 200 iterations. Using these criteria, the entire computation process takes about 35 hours using a single workstation. The detailed time is given in Table 1.

	Computation time (hours)	Avg. # of CMA iteration
Initial points (24/45)	21.5	157.4
Scaled by 0.8 (16/21)	11.5	146.9
Scaled by 0.8^2 (4/5)	2	137.4
Scaled by 0.8^3 (0/1)	0.6	200
Scaled by 0.8^4 (1/1)	0.2	68

Table 1: *Computation time for optimizing the parameter sample points. Starting from* 45 *points, we repeatedly scaled down the failed points by* 0.8 *until a successful solution is found. (a/b) denotes the number of successful points (a) and total points (b) at each stage.*

8. Discussion

Using our approach, a user can freely swing his or her arms while controlling a walking biped. Simply tracking the user pose from a depth camera would result in falls of the simulated biped. Real-time interactive control is made possible by using a regression on the precomputed set of parameters and a balancing mechanism that compensates for the deviation between the simulated character and the user.

In the accompanying video, the biped exhibits some jerky upper-body motions due to the noise in the Kinect data. We leave them as they are because handling input noise is not a goal of this paper, and the jerkiness does not affect the stability of our simulated biped. It should also be noted that the skeleton data from the current generation depth camera is not suitable for synthesizing quality animations. There are a lot of problems such as noises, incorrect labels, unnatural rolls of limbs, and over-extended elbows and knees. Better performing input devices would improve the motion quality, and also encourage the development of other interesting real-world applications.

We assumed that the space for input motions can be parameterized using only a small number of quasi-static quantities such as the deviation of the CM position from the reference motion. Although this assumption was made due to limited computational resources, the proposed 3D parameter space is effective for controlling a consistently walking biped which mimics user actions. However, our controller may not show good results for some cases when there are abrupt changes in the upper-body pose, as shown in the accompanying video. Additional dimensions for the parameter space

would help the controller deal with such cases. For example, the velocity and acceleration of the CM are important for balancing, and the external forces could also be parameterized to improve the foot-step planning for balance recovery. However, the training data for such parameters can only be generated in the form of a short motion clip, and thus obtaining steady controllers would require time-varying controller parameters such as the approach proposed by Liu et al. [LvdPY16]. Also, the increased search space would require more effective methods such as a better regression method that shows a good performance with a small number of samples, or an adaptive sampling of the parameter space. The hybrid use of pre-computed searches and online searches such as those in [MdLH10, HRL15] would also be an interesting future research topic.

In our demonstrations, the biped only exhibits forward walking at the same speed. We chose normal walking because we experimentally found it was the most stable one, and focused on designing the controller that mimics user performance while walking consistently. Because the controller proposed by Lee et al. [LKL10] can control various styles of walking, turning, and spinning as shown in their original demonstration, our performance-based controller can be extended to control those skills in principle. Walking speed can also be adjusted by kinematically editing the reference motions. However, to support these skills, a user interface needs to be extended to control the additional parameters (turning angles, forward speed, etc.). Designing such an interface using a Kinect sensor is non-trivial because of the limited capture space, the limited recognition range of forward-facing direction, and the increased dimensionality of the parameter space. Dealing with these issues would be an interesting topic for future research.

We use the stepping strategy built on top of a single normal walking motion for balance recovery. However, it is well-known that a human uses a combination of multiple strategies including hip, ankle and stepping strategies depending on the amplitude and duration of the perturbation. Also, the strategy depends on the style of the motion that the user is performing. For example, when practicing a Kendo skill, the agility and speed rely highly on the coordinated use of the arms holding the sword and the legs for stepping. A trajectory library approach to automatically generating and controlling lower-body motions based on the input upper-body motion would also be promising.

Acknowledgements

We thank the anonymous reviewers for helpful feedback. This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03930746) and the MSIP (NRF-2014R1A1A1038386), and by the Research Grant of Kwangwoon University in 2016.

References

[ABDLH13] AL BORNO M., DE LASA M., HERTZMANN A.: Trajectory optimization for full-body movements with complex contacts. *IEEE Trans. Vis. Comput. Graph 19*, 8 (2013), 1405–1414. 2

[CBvdP09] COROS S., BEAUDOIN P., VAN DE PANNE M.: Robust task-based control policies for physics-based characters. *ACM Transactions on Graphics 28*, 5 (2009), 1–9. 2

[CH05] CHAI J., HODGINS J. K.: Performance animation from low-dimensional control signals. *ACM Transactions on Graphics 24*, 3 (2005). 2

[CKJ*11] COROS S., KARPATHY A., JONES B., REVERET L., VAN DE PANNE M.: Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics 30*, 4 (2011). 3

[dLMH10] DE LASA M., MORDATCH I., HERTZMANN A.: Feature-based locomotion controllers. *ACM Transactions on Graphics 29*, 4 (2010), 131. 2

[dSAP08] DA SILVA M., ABE Y., POPOVIĆ J.: Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics 27*, 3 (2008), 1–10. 2

[dSBK*14] DA SILVA B. C., BALDASSARRE G., KONIDARIS G., , BARTO A.: Learning parameterized motor skills on a humanoid robot. In *Proceedings of the 2014 International Conference in Robotics and Automation (ICRA)* (2014). 3

[GEA] GEAR: Geometric Engine for Articulated Rigid-body simulation. http://www.cs.cmu.edu/~junggon/tools/gear.html. 6

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (2004), 522–531. 2

[HBL11] HA S., BAI Y., LIU C. K.: Human motion reconstruction from force sensors. In *Symposium on Computer Animation* (2011), Bargteil A. W., van de Panne M., (Eds.), pp. 129–138. 3

[HL14] HA S., LIU C. K.: Iterative Training of Dynamic Skills Inspired by Human Coaching Techniques. *ACM Transactions on Graphics 34*, 1 (2014). 2

[HL16] HA S., LIU C. K.: Evolutionary optimization for parameterized whole-body dynamic motor skills. In *Proceedings of the 2016 International Conference in Robotics and Automation (ICRA)* (2016). 3

[HRL15] HÄMÄLÄINEN P., RAJAMÄKI J., LIU C. K.: Online control of simulated humanoids using particle belief propagation. *ACM Transactions on Graphics 34*, 4 (2015), 81:1–81:13. 2, 8

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *SIGGRAPH* (1995), pp. 71–78. 2

[IWZL09] ISHIGAKI S., WHITE T., ZORDAN V. B., LIU C. K.: Performance-based control interface for character animation. *ACM Transactions on Graphics 28*, 3 (2009). 3

[JPG*14] JACOBSON A., PANOZZO D., GLAUSER O., PRADALIER C., HILLIGES O., SORKINE-HORNUNG O.: Tangible and modular input device for character articulation. *ACM Transactions on Graphics 33*, 4 (2014), 82:1–82:12. 2

[KH10] KWON T., HODGINS J.: Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), pp. 129–138. 2

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3 (2002), 491–500. 2

[LKL10] LEE Y., KIM S., LEE J.: Data-driven biped control. *ACM Transactions on Graphics 29(4)* (2010). 2, 3, 4, 8

[LLK*15] LEE Y., LEE K., KWON S.-S., JEONG J., O'SULLIVAN C., PARK M. S., LEE J.: Push-recovery Stability of Biped Locomotion. *ACM Transactions on Graphics 34*, 6 (2015), 180:1–180:9. 5

[LPKL14] LEE Y., PARK M. S., KWON T., LEE J.: Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics 33*, 6 (2014), 218:1–218:11. 2

[LvdPF00] LASZLO J., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), SIGGRAPH '00, pp. 201–208. 2

[LvdPY16] LIU L., VAN DE PANNE M., YIN K.: Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics 35*, 3 (2016). 2, 3, 8

[LYvdPG12] LIU L., YIN K., VAN DE PANNE M., GUO B.: Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Transactions on Graphics 31*, 6 (2012), 154. 3

[MdLH10] MORDATCH I., DE LASA M., HERTZMANN A.: Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics 29*, 4 (2010), 71. 2, 8

[MLPP09] MUICO U., LEE Y., POPOVIĆ J., POPOVIĆ Z.: Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics 28*, 3 (2009), 1–9. 2

[MPP11] MUICO U., POPOVIĆ J., POPOVIĆ Z.: Composite control of physically simulated characters. *ACM Transactions on Graphics 30*, 3 (2011), 16. 2

[MZS09] MACCHIETTO A., ZORDAN V., SHELTON C. R.: Momentum control for balance. *ACM Transactions on Graphics 28*, 3 (2009), 1–8. 2

[NWB*10] NGUYENY N., WHEATLAND N., BROWN D. F., PARISE B., LIU C. K., ZORDAN V. B.: Performance capture with physical interaction. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), pp. 189–195. 3

[OTH02] OORE S., TERZOPOULOS D., HINTON G.: A desktop input device and interface for interactive 3d character animation. In *Proceedings of the 2002 Graphics Interface Conference* (2002), pp. 133–140. 2

[RTIK*14] RHODIN H., TOMPKIN J., IN KIM K., VARANASI K., SEIDEL H.-P., THEOBALT C.: Interactive motion mapping for real-time character control. *Comput. Graph. Forum 33*, 2 (2014), 273–282. 2

[RTK*15] RHODIN H., TOMPKIN J., KIM K. I., DE AGUIAR E., PFISTER H., SEIDEL H.-P., THEOBALT C.: Generalizing wave gestures from sparse examples for real-time character control. *ACM Transactions on Graphics 34*, 6 (2015). 2

[SH08a] SHIRATORI T., HODGINS J. K.: Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics 27*, 5 (2008), 123:1–123:9. 3

[SH08b] SLYPER R., HODGINS J. K.: Action capture with accelerometers. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 193–199. 2

[SKL07] SOK K. W., KIM M., LEE J.: Simulating biped behaviors from human motion data. *ACM Transactions on Graphics 26*, 3 (2007), 107. 2

[SLSG01] SHIN H. J., LEE J., SHIN S. Y., GLEICHER M.: Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2 (2001), 67–94. 2

[SOL13] SEOL Y., O'SULLIVAN C., LEE J.: Creature features: Online motion puppetry for non-human characters. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2013), SCA '13, pp. 213–221. 2

[Stu98] STURMAN D. J.: Computer puppetry. *IEEE Comput. Graph. Appl. 18*, 1 (1998), 38–45. 2

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion Doodles: An Interface for Sketching Character Motion. *ACM Transactions on Graphics 23*, 3 (2004). 2

[TM04] TERRA S. C. L., METOYER R. A.: Performance timing for keyframe animation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), SCA '04, pp. 253–258. 2

[TZK*11] TAUTGES J., ZINKE A., KRÜGER B., BAUMANN J., WEBER A., HELTEN T., MÜLLER M., SEIDEL H.-P., EBERHARDT B.: Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics 30*, 3 (2011), 18:1–18:12. 2

[VHKK12] VÖGELE A., HERMANN M., KRÜGER B., KLEIN R.: Interactive steering of mesh animations. In *Proceedings of the 2012 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012). 2

[VSHJ12] VONDRAK M., SIGAL L., HODGINS J., JENKINS O.: Video-based 3d motion capture through biped control. *ACM Transactions on Graphics 31*, 4 (2012). 3

[WP10] WU J.-C., POPOVIĆ Z.: Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics 29*, 4 (2010), 72. 2

[YL10] YE Y., LIU C. K.: Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics 29*, 4 (2010), 74. 2

[YLvdP07] YIN K., LOKEN K., VAN DE PANNE M.: Simbicon: simple biped locomotion control. *ACM Transactions on Graphics 26*, 3 (2007), 105. 2

[YP03] YIN K., PAI D. K.: FootSee: An Interactive Animation System. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 329–338. 2

[ZvdP05] ZHAO P., VAN DE PANNE M.: User interfaces for interactive control of physics-based 3d characters. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (2005), I3D '05, pp. 87–94. 2