Performance-based Control Interface for Character Animation

Satoru Ishigaki
Georgia Tech

Timothy White
Georgia Tech

Victor B. Zordan
UC Riverside

C. Karen Liu
Georgia Tech

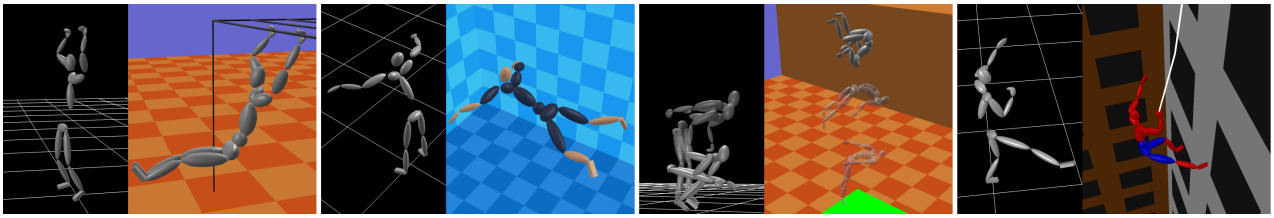

Figure 1: *Our system allows the user to directly control a virtual character in a wide array of situations.*

Abstract

Most game interfaces today are largely symbolic, translating simplified input such as keystrokes into the choreography of full-body character movement. In this paper, we describe a system that directly uses human motion performance to provide a radically different, and much more expressive interface for controlling virtual characters. Our system takes a data feed from a motion capture system as input, and in real-time translates the performance into corresponding actions in a virtual world. The difficulty with such an approach arises from the need to manage the discrepancy between the real and virtual world, leading to two important subproblems 1) recognizing the user's intention, and 2) simulating the appropriate action based on the intention and virtual context. We solve this issue by first enabling the virtual world's designer to specify possible activities in terms of prominent features of the world along with associated motion clips depicting interactions. We then integrate the prerecorded motions with online performance and dynamic simulation to synthesize seamless interaction of the virtual character in a simulated virtual world. The result is a flexible interface through which a user can make freeform control choices while the resulting character motion maintains both physical realism and the user's personal style.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Character animation, Motion capture

1 Introduction

Intuitive and expressive control interfaces for controlling virtual characters can dramatically improve the end-user experience in applications such as video games, virtual reality, and communication tools. For such online applications, the user's intention must be accurately identified and translated to an action executed by the virtual character. Traditional gaming interfaces identify the user's intention via a small set of agreed-upon control signals, such as a specific button command or keystroke. Once the intention is identified, a pre-recorded motion is played, and the user has little or no control over the execution of the action. As we move towards more expressive input devices (e.g. accelerometers or pressure sensors), the mapping between the user's intention and a character's action becomes more literal. In the limit, real-time motion capture enables a one-to-one correspondence between a user and a virtual character, allowing her to directly control the character's activity through online performance.

Performance-based control gives the user ultimate freedom to direct the virtual character in an intuitive manner. Moreover, such interfaces enable the virtual character to behave intelligently, as the user's performance reveals how humans take strategic actions based on situation and stylistic variation. However, in spite of its immense potential, this type of "literal" control interface is plagued with specific issues related to the discrepancy between the user and the virtual character, and their respective environments. We use a virtual playground scenario as an example to illustrate a few common issues in Figure 2. When considering performance-based control interfaces for practical use, two challenging subproblems arise. They are: 1) robust, online recognition of user intention; and 2) appropriate (visually pleasing) modification to the performance which both manages differences and carries out the recognized intent.

We propose a new control interface for navigating and manipulating virtual worlds based on motion reconstructed in real-time from a human performance. Our control interface intelligently maps online motion capture data to a virtual character's action, such that the user can directly control the virtual character using his/her own body movement. The proposed system leverages a physical simulation, a small set of offline action examples, and an optimal integration process to synthesize the character's motion. When interacting with an object in the the virtual world, the simulation ensures that the global movement of the character obeys the laws of physics, rather than directly tracking the performance. In addition, we propose a novel method for deriving active control from the performance to give intuitive control over the simulation. For local joint configurations, we integrate the online performance with offline example motions that demonstrate specific desired interactions with virtual artifacts. Recognizing *when* and knowing *how* to integrate the online and offline motion capture poses are particularly challenging because of the online nature of the input motion and real-time computation requirements. Furthermore, the identification needs to be robust against different styles of motion and different body types of users. We propose efficient algorithms to

ACM Reference Format
Ishigaki, S., White, T., Zordan, V., Liu, C. 2009. Performance-based Control Interface for Character Animation. *ACM Trans. Graph. 28*, 3, Article 61 (August 2009), 8 pages. DOI = 10.1145/1531326.1531367 http://doi.acm.org/10.1145/1531326.1531367.

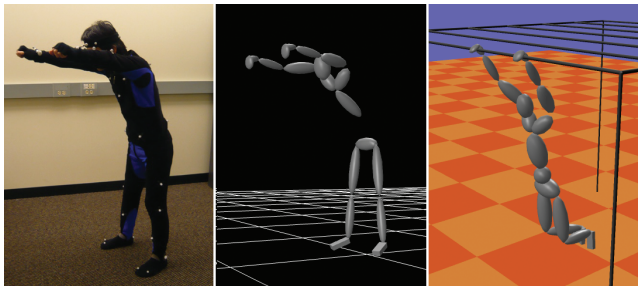

Figure 2: *Left: The user's online performance. Middle: Real-time reconstruction of the user's pose. Right: The corresponding character's pose synthesized by our system. This monkey bar example illustrates several issues observed when directly mapping the user's performance onto the character. First, the lower body is not realistic due to the limitations of the real world that hinder the user's performance. Second, the performance does not accurately maintain virtual contact with the feature because no actual contacts are established in the real world. Third, the global movement of the user is physically inconsistent with the dynamics of the virtual world.*

recognize the appropriate moments for integration and which body parts to include.

By correcting and enhancing the online performance with physical simulation and offline example motions, our interface provides freeform control by which the user can make a decision about the strategy appropriate for a given scenario, while the virtual character's motion maintains physical realism and resembles the user's personal style. We demonstrate the power of our approach with a testbed implemented in a mini-adventure game where the character can navigate the world with a large set of actions including climbing, walking, swinging, jumping, swimming and more.

2 Related work

Novel interfaces for interacting with characters have been proposed through the use of different input devices such as a simple mouse [Laszlo et al. 2000; Zhao and van de Panne 2005], a tablet PC [Thorne et al. 2004], video cameras [Chai and Hodgins 2005], Nintendo Wiimotes [Shiratori and Hodgins 2008], a motion capture system [Shin et al. 2001], or a floor pressure sensor [Yin and Pai 2003]. In these systems, the ratio of input degrees of freedom (DOFs) to output DOFs is often few to many. A representative example of this line of work is proposed by Chai and Hodgins [Chai and Hodgins 2005], where a custom camera set-up is used to record a small set of motion markers, and the output of their system is a full body character. Their contribution centers around accurately determining a motion example in a library which matches the input markers. Shin et al. presented a performance animation system that directly maps the detailed user performance to a virtual avatar [Shin et al. 2001]. Our work also depends on a high DOF motion input provided by a motion capture system. However, we focus on appropriate modification of the online motion to control virtual characters in different dynamic situations, rather than directly using the performer's motion to drive an avatar. Consequently, our method allows the character to achieve movements beyond the actual performance of the user.

A few researchers have addressed the use of physics in the designing of custom interfaces. Laszlo et al. show that a small set of input keys can be used to generate full-body character motion for 2D-planar characters with several DOFs [Laszlo et al. 2000]. Their insight is that by exploiting the dynamics of the character's motion, a fewer number of inputs are required – effectively increasing the bandwidth through the control of the simulation. Inspired by their work, our simple simulation embeds much of the state of the character in the few parameters related to the root. Shiratori and Hodgins present an interface for controlling a simulated character using Nintendo Wii-motes [Shiratori and Hodgins 2008]. They perform the mapping of the input to high-level control values in the behavior control system. They claim that through the use of a physical model, players will be more forgiving of delays related to the interface. Similarly, we observe that users of our system can learn to time their motions correctly due to the consistency of the physics in the underlying world.

One can think of our work as a performance-based motion editing system for realistic character animation. Unlike previous motion editing techniques that depend on an animator to oversee the modifications that are to take place [Witkin and Popović 1995; Gleicher 1997; Popović and Witkin 1999; Lee and Shin 1999], or a large motion capture database with high-level user control [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002](among many), our approach relies on the performer to use her own body movement to modify an existing motion. Because the performance is not only providing detailed motion trajectories, but also revealing how humans make intelligent decisions based on the external situations, our method is able to produce drastically different output motions from the prerecorded examples.

Performance-based motion editing approaches have also been investigated by several researchers. Oore et al. proposed a layered approach to animation using a novel interface consisting of two trackers [Oore et al. 2002]. Similarly, Dontcheva et al. presented a performance animation system based on mapping between the movement of a designed widget tool and the character's DOFs [Dontcheva et al. 2003]. Neff et al. described a method that determines correlations between the DOFs, effectively reducing the control space [Neff et al. 2007]. Igarashi et at. interpolated spatial keyframes by moving the cursor in 3D space, allowing the user to create expressive character animation in real-time [Igarashi et al. 2005]. Our problem domain is somewhat different from many other interfaces because the user's whole body is the input representation. Although we do not need to tackle the problem of mapping between low DOF input device and the high DOF character representation, our problem poses a unique challenge that requires a novel algorithm to tease out the meaningful part of the online performance and seamlessly integrate it with the existing motion.

One challenge in our work is to accurately recognize the performance as an imitation of a known behavior. Much previous work in computer vision has focused on recognizing human motion from time-sequential images [Yamato et al. 1992; Darrell and Pentland 1993]. A unique requirement of our system is to recognize the intention of the performer's motion (automatically) in a very short time window following its initiation. Other researchers have addressed matching motion capture queries to close examples inoffline settings [Chai and Hodgins 2005; Müller et al. 2005; Keogh et al. 2004]. We propose a new recognition algorithm that shares similar ideas with previous work but allows the behaviors to be identified in real-time without use of the full input motion sequence.

3 Algorithm overview

Our system takes as input a pose stream recorded by a motion capture device, and produces, in real time, a character animation that maps the user's movements to the virtual world. To use the input motion for control of the virtual character, we need to modify the real-time input motion so that the virtual character's actions reflect the performer's intention, preserve her motion style, and sat-

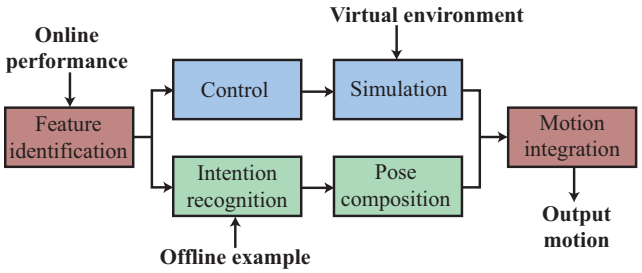

Figure 3: *Overview of the system.*

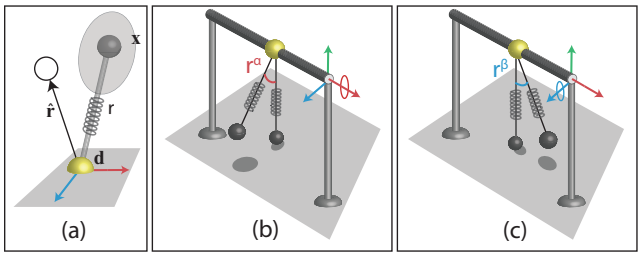

Figure 4: *Active body control via an intentional contact.*

isfy physical constraints in the virtual world. The designer of the virtual world plays a key role in this process by predefining the features of the world that the user can interact with and associating these features with motion sequences of possible interactions.

The overall system, shown in Figure 3, comprises both a dynamic process and a kinematic process to produce the virtual character's motion. First, we cycle through the features of the environment to check if the input motion matches their conditions for engagement. Once the interacting feature is identified, the dynamic process simulates the global motion of the virtual character by combining active control inferred from the user's performance with dynamic influences from the virtual environment. In tandem, the kinematic process produces realistic context-appropriate poses by matching the user's motion with the example interaction. Finally we formulate an optimization process to combine the global motion from the dynamic process and the poses from the kinematic process to produce an output motion that conforms to kinematic constraints in the virtual environment.

4 Feature identification

The virtual world is built upon a set of user-specified features, each of which consists of a geometric description **G**, a set of engage conditions **C**, an example motion **Q**, and some dynamic parameters **D**. To determine whether the user intends to interact with a particular feature, we need to first check whether the character's motion satisfies the engage conditions **C**. If the conditions are satisfied, the system will return a set of intended contact points **p** on the feature along with the character's end-effectors **d** that engage them. For example, if the user's hands are closed and in the proximity of the monkey bar, the system will determine that the user is attempting to interact with the monkey bar, and return the closest points on the monkey bar from the hands as the contacts **p** and the markers on the hands as the corresponding end-effectors **d**.

5 Dynamic simulation

We employ a simplified rigid body model to simulate the dynamics of the character. Our system simulates the global motion of the virtual character, namely the position of the center of mass (COM), $\mathbf{x}$ and the global orientation of the body θ. The active control of the virtual character is done through *intentional* contacts with the environment. As opposed to a generic contact, the character can actively manipulate such intentional contacts to generate appropriate reaction forces for achieving a certain task.

We model such an "intentional" contact as a (two-dimensional revolute) universal joint connected to the COM by a linear damped spring (Figure 4(a)). The joint is actuated by two angular proportional-derivative (PD) controllers with individual gains and damping coefficients defined by the virtual feature, as part of its dynamic parameters **D**. The roll and pitch rotations of the joint, α and β, can be precisely controlled if the PD servos are user-actuated (Figure 4(b) and (c)). The linear spring acts like a elastic pendulum if the joint is unactuated. In addition, the user can also actively control the desired length of the linear spring. The gain and the damping coefficient of the linear spring are roughly related to the muscle strength of the character.

Once the character establishes an intentional contact with a virtual feature, active control is derived from the relative movement of the end-effector and COM, $\mathbf{r}_j = \mathbf{d}_j - \mathbf{x}$. For each intentional contact j, the desired angles and lengths for the actuators are taken from the user's relative movement $\hat{\mathbf{r}}_j = \hat{\mathbf{d}}_j - \hat{\mathbf{x}}$. When the relative movement of end-effector $\mathbf{r}_j$ on the character is different from $\hat{\mathbf{r}}_j$, the joint and the linear spring will generate forces to move the character's COM to the desired position relative to the intentional contact. The equations of motion for the COM can be written as:

$$m\ddot{\mathbf{x}} = \sum_j (\mathbf{f}_j^l + \mathbf{f}_j^\alpha + \mathbf{f}_j^\beta) + m\mathbf{g} \tag{1}$$

$$\mathbf{f}^l = (k_s(\|\hat{\mathbf{r}}\| - \|\mathbf{r}\|) - k_d\|\dot{\mathbf{r}}\|)\frac{\mathbf{r}}{\|\mathbf{r}\|} \tag{2}$$

$$\mathbf{f}^\alpha = (k_s^\alpha(\hat{r}^\alpha - r^\alpha) + k_d^\alpha(\dot{\hat{r}}^\alpha - \dot{r}^\alpha))\frac{\vec{\mathbf{u}}}{\|\mathbf{r}\|} \tag{3}$$

$$\mathbf{f}^\beta = (k_s^\beta(\hat{r}^\beta - r^\beta) + k_d^\beta(\dot{\hat{r}}^\beta - \dot{r}^\beta))\frac{\vec{\mathbf{v}}}{\|\mathbf{r}\|} \tag{4}$$

where r^α and r^β are the angles from $\mathbf{r}$ to the pitch axis $\vec{\mathbf{i}}$ and to the roll axis $\vec{\mathbf{j}}$ respectively. $\vec{\mathbf{u}}$ is a vector normal to $\mathbf{r}$ and the pitch axis ($\vec{\mathbf{u}} = \mathbf{r} \times \vec{\mathbf{i}}$). Similarly, $\vec{\mathbf{v}}$ is defined as $\vec{\mathbf{v}} = \mathbf{r} \times \vec{\mathbf{j}}$. $\mathbf{f}^\alpha$ and $\mathbf{f}^\beta$ are in effect the forces due to torques about the related hinges at the point of the intentional contact. By matching the relative movement of the end-effectors and the COM, the user can intuitively control a simulated virtual character from completely different physical conditions. For example, when interacting with the monkey bar feature, the character anchors its hands on the bar and swings its COM about the bar as the user moves her arms back and forth in the air with her COM relatively fixed. Although the user performs a completely different movement, the relative motion between the end-effectors and COM is consistent in the two scenarios.

When the character is constrained by any intentional contacts, the orientation is not affected by the actuator controls. Instead, control forces are applied directly to the COM. In addition, collisions are applied as impulses to the center of mass, inducing no rotation change. Empirically, we found it easier for our players to control the character using this simplification. Furthermore, when the position of the COM and the end-effectors are simultaneously constrained, the global orientation is largely determined and does not play a significant role in the output poses. The orientation, however,

is affected by contact with other virtual objects when the character is unconstrained (i.e. free fall). In this case, both the character's global position and orientation are passively affected by gravity and other external forces via collisions. The inertial parameters are derived from the user's pose for each time instance. Thus, the user can control spin by manipulating her own limbs in an intuitive way. To ease landing, we apply an imaginary force to control the pitch and roll rotation such that the character can land on its feet. For all the features we have implemented and tested, landing is the only situation when such a trade-off between physical correctness and user controllability is necessary.

6 Kinematic composition

For certain features, our system leverages an offline example motion sequence to provide detailed, accurate poses that correct and enhance the online performance. To integrate the poses from these two motion streams properly, we need to infer the intention of the user from the online performance. In addition, we must determine which part of the user's motion needs to be modified and which part can be mapped to the virtual character literally.

6.1 Intention recognition

At each time instance, the system first updates the constraint state of each end-effector on the user. In our system, the only constraint in the real world is the floor, which is recognized during the calibration of the system. For example, if the user is standing on the floor, the end-effectors on the feet are constrained while those on the hands and head are unconstrained. We define S_f as a set of unconstrained end-effectors and S_c as a set of constrained ones. If the unconstrained end-effectors of the user are imitating the example motion $\mathbf{Q}$ of the virtual feature, we integrate the user's pose with $\mathbf{Q}$. Otherwise, the virtual character's motion directly follows the user's pose $\hat{\mathbf{q}}$.

Our system recognizes the imitation behavior via a simple but robust analysis in the low dimensional space, invariant to users' different styles in movements or different skeletal dimensions. We use Principal Component Analysis (PCA) to project the position and the velocity of end-effectors to a low-dimensional space. Specifically, we compare the relative positions $\hat{\mathbf{r}}_j$ and the relative velocities $\dot{\hat{\mathbf{r}}}_j$ of the user's unconstrained end-effectors ($j \in S_f$) against those in the example motion. For each pose $\check{\mathbf{q}}^i$ in the example motion $\mathbf{Q}$, we compose a feature vector, $\check{\mathbf{w}} = [\check{\mathbf{r}}_j, \dot{\check{\mathbf{r}}}_j],\ j \in S_f$, which contains relative positions and relative velocities of the unconstrained end-effectors. Because the COM position and the yaw rotation are subtracted from the end-effectors, the feature vector is invariant to user's global position and orientation.

To measure the similarity, we apply PCA to the feature matrix $\mathbf{W} = [\check{\mathbf{w}}^1, \cdots, \check{\mathbf{w}}^N]$ derived from the example motion, where N denotes the number of frames in $\mathbf{Q}$. We define matrix $\mathbf{A}$ as the first m principal components of the feature matrix $\mathbf{W}$. To compare the similarity of the current user's pose $\hat{\mathbf{q}}$ to the example motion, we project the feature vector $\hat{\mathbf{w}}$ derived from $\hat{\mathbf{q}}$ to the low dimensional space via $\mathbf{A}^T$, and back project into the high-dimensional space using $\mathbf{A}$, resulting in error e_1 as:

$$e_1 = \|\hat{\mathbf{w}} - \mathbf{A}(\mathbf{A}^T\hat{\mathbf{w}})\|^2 \tag{5}$$

Of course, an intuitive way to determine whether the user is imitating the example motion is to compute the similarity of the user's pose and the example motion. However, due to the inherent differences between the user and the performer of the example motion, often time the user's motion is obviously different from the example motion in spite of the attempt at imitation. To improve the usability of our system in practice while still being able to accurately discern the user's intention, we add a second condition that measures the consistency of the user. This "consistency" condition allows the user's pose to be dissimilar from the example motion, as long as the deviation is nearly constant over time. The condition can be measured by the following equations.

$$e_2 = \|\Delta\hat{\mathbf{w}} - \mathbf{A}(\mathbf{A}^T\Delta\hat{\mathbf{w}})\|^2, \quad \Delta\hat{\mathbf{w}} = \hat{\mathbf{w}}^t - \hat{\mathbf{w}}^{t-1} \tag{6}$$

If the reconstruction errors e_1 and e_2 are smaller than their respective threshold values, the system determines that the user intends to perform the activity described by the example motion.

6.2 Pose composition

If the intention recognition procedure concludes that the user is intentionally imitating the example motion of a particular feature, the system synthesizes an intermediate pose $\bar{\mathbf{q}}$ from the user's current pose $\hat{\mathbf{q}}$ and the example motion $\mathbf{Q}$. To do so, we must compare the constraint state of the end-effectors on the user against those on the virtual character. Based on the constraint state we partition the end-effectors into constrained and unconstrained sets for the user, S_c^u and S_f^u, as well as for the virtual character S_c^v and S_f^v. The composition algorithm follows the *integration rule* defined as follows. For every end effector in $S_c^u \cap S_f^v$, namely, the end-effector is constrained on the user and unconstrained on the virtual character, the joint angles on the branch of that end-effector are synthesized from $\mathbf{Q}$. For all other joints, the joint angles come from $\hat{\mathbf{q}}$.

To compute the optimal pose from the example motion $\mathbf{Q}$, we propose a fast optimization in the low-dimensional space of the example motion derived from PCA. Optimizing in the low-dimensional space allows us to compute a pose that maintains the characteristics and the correlations of the example motion while the output motion can be interpolated or extrapolated according to the user's control motion. A pose $\mathbf{q} \in R^n$ can be mapped back and forth to its low dimensional representation $\mathbf{s} = [s_1, ..., s_m] \in R^m$ via the matrix $\mathbf{B}$ associated with each example motion $\mathbf{Q}$.

$$\mathbf{s} = \mathbf{B}^T(\mathbf{q} - \check{\mathbf{q}}_{avg}), \qquad \mathbf{q} = \mathbf{B}\mathbf{s} + \check{\mathbf{q}}_{avg} \tag{7}$$

where $\check{\mathbf{q}}_{avg}$ is the mean pose of the example motion $\mathbf{Q}$. $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_m]$ consists of the first m eigenvectors of the covariance matrix of $\mathbf{Q}$.

We formulate the optimization to solve for the low-dimensional representation of a pose $\mathbf{s}$ that minimizes the weighted sum of three objectives.

$$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmin}}\ \ w_P G_P(\mathbf{s}) + w_E G_E(\mathbf{s}) + w_S G_S(\mathbf{s}) \tag{8}$$

We define the objective function from three user-weighted terms defined in the following manner. G_P ensures the unconstrained end effectors move in a way similar to the user:

$$G_P = \sum_{j \in S_f^u} \|\mathbf{f}_j(\mathbf{B}\mathbf{s} + \check{\mathbf{q}}_{avg}) - \hat{\mathbf{r}}_j\|^2 \tag{9}$$

where function $\mathbf{f}_j$ evaluates the Cartesian coordinates of the relative end-effector j from a pose.

G_E measures the similarity of the solution to a representative pose of the example motion $\mathbf{Q}$, denoted as $\check{\mathbf{q}}_{rep}$. We first select k most similar poses to the user's current pose $\hat{\mathbf{q}}$ from $\mathbf{Q}$. The representative pose $\check{\mathbf{q}}_{rep}$ is then computed as the weighted average pose of

these k poses. G_E penalizes the deviation from $\check{\mathbf{q}}_{rep}$ in the low-dimensional space, scaled by the corresponding eigenvalue λ_i associated with each axis. These scaling factors balance the influence of each component in the low-dimensional space.

$$G_E = \sum_{i=1}^{m} \|(s_i - \mathbf{b}_i^T(\check{\mathbf{q}}_{rep} - \check{\mathbf{q}}_{avg}))\|^2 \frac{1}{\lambda_i} \tag{10}$$

Finally, G_S minimizes the differences between the current pose and the previous pose to increase the smoothness of the motion.

$$G_S = \|(\mathbf{B}\mathbf{s} + \check{\mathbf{q}}_{avg}) - \mathbf{q}_{t-1}\|^2 \tag{11}$$

Once the optimal pose is reconstructed by $\mathbf{B}\mathbf{s}^* + \check{\mathbf{q}}_{avg}$, we follow the integration rule defined earlier to obtain the intermediate pose $\bar{\mathbf{q}}$. When the character begins or ends the interaction with a feature, there could be potential popping artifacts in the motion. To enforce smooth transitions in/out an interaction, we linearly blend $\bar{\mathbf{q}}$ with the previous pose $\mathbf{q}_{t-1}$ for a small window of time.

7 Motion integration

We utilize a final optimization process to solve for the output pose that meets the simulated global motion $\mathbf{x}$ and the intentional contacts $\mathbf{p}$, while matching the intermediate pose $\bar{\mathbf{q}}$ as closely as possible. We also add inequality constraints to prevent the character from penetrating the features in the environment.

$$\underset{\mathbf{q}_t}{\text{argmin}} \ \|\mathbf{q}_t - \bar{\mathbf{q}}\|^2 \quad \text{subject to} \quad \begin{cases} \mathbf{p}_{com} = \mathbf{x} \\ \mathbf{p}_{ef} = \mathbf{p} \\ \mathbf{p}_{pen} \leq \mathbf{p} \end{cases} \tag{12}$$

where $\mathbf{p}_{com}$, $\mathbf{p}_{ef}$, and $\mathbf{p}_{pen}$ are functions of $\mathbf{q}_t$ which evaluate the position of the center of mass, the end-effectors of the interactive contacts, and the locations of the penetrations, respectively.

We formulate an iterative process to solve this nonconvex optimization. Our formulation efficiently yields a local optimum biased toward the example motion $\mathbf{Q}$, in a way similar to the algorithm proposed by Grochow et al. [Grochow et al. 2004]. We first transform the joint space to the space formed by the eigenvectors of the covariance matrix of $\mathbf{Q}$, spanned by the columns of $\mathbf{B}$. At the first iteration, we only solve for one degree of freedom along the eigenvector corresponding to the largest eigenvalue. As we expand the space of degrees of freedom by including successive eigenvectors in the following iterations, the residual of constraints and objectives is gradually eliminated. The domain of the degrees of freedom is equivalent to the original joint space once all the eigenvectors are included. In practice, the residual approaches zero very quickly after three to five eigenvectors are added to the domain. Consequently, our method produces a natural pose with less computational time compared to previous optimization-based inverse kinematics methods.

8 Interaction with virtual features

Our system provides a generic template for designing new features in the virtual world. The designer can easily assemble, reuse, and modify any features to create an interesting game world. To design a new feature, the designer only needs to provide the following information.

Geometric description G. Each feature consists of a set of geometric primitives describing the area in which intentional contacts can be established.

Engage conditions C. An engage condition is a boolean function of the character's current pose. The default engage condition determines whether an intentional contact can be established based on the distance between specified end-effectors $\mathbf{d}$ and the contact area described in $\mathbf{G}$.

Example motion Q. The designer can optionally provide a motion sequence demonstrating the interaction with the virtual feature. Some features, such as rock climb and trampoline, do not require any example motion.

Dynamic parameters D. The dynamic parameters of a feature include the coefficients of the active control and additional parameters for simulating the dynamics of the feature itself (e.g. the spring coefficients for the trampoline). These parameters determine the reaction forces the character receives from the feature during interaction. The coefficients of the control are determined empirically based on the feature design and player controllability.

8.1 Swing on monkey bars

- **G** : Each monkey bar is represented by a cylinder on which the intentional contacts can be established.
- **C** : The default condition computes the closest distance from the end-effector on the hand to the cylinder. If the hand is closed and in the proximity of the cylinder, the engage conditions are met.
- **Q** : The example motion only contains one cycle of swinging with two hands on the bar. This motion was directly downloaded from the CMU motion capture database ([CMU]).
- **D** : We set coefficients for the linear spring based on the desired strength of the arm and the flexibility of the character's upper body. The joints (PD servos) at the intentional contacts only allow the user to actively control the swing about the pitch axis (forward/backward) by tracking the angular velocity of the user. We leave the sideway swing passive ($k_s^\beta = 0$).

We demonstrate that the user can manipulate the monkey bars in unlimited ways beyond the simple swing provided by the example motion, such as alternating arms to move forward and backward, traverse sideways, do chin-up, hang with one arm while punching or kicking, or moving more vigorously to increase the forward swing speed (Figure 5).

8.2 Climb a rope/pole

- **G** : A pole is represented by a cylinder. A rope is represented by a set of rigid cylinders connected by passive joints.
- **C** : The same engage conditions as the monkey bar feature.
- **Q** : One cycle of upward rope climbing.
- **D** : The coefficients for the active control are the same as the monkey bar feature. The dynamics of rope is simulated as a constrained mass-spring system.

We show that the user can direct the character to climb not only up and down on the rope, but also use the rope to swing to reach different places. In a complex scene with multiple ropes, the user can take advantage of other virtual objects to create interesting motions. For example, the user can kick the wall nearby to gain some momentum and then swing from rope to rope.

The rope feature can be extended to a more entertaining scenario where the character has a special power to shoot out a web and use it to swing across buildings (Figure 1 right).

8.3 Walk on the floor

- **G** : A floor is represented by a plane.
- **C** : If a foot is in the proximity of the plane, the engage condition is met.
- **Q** : One cycle of walking motion is provided.
- **D** : The spring and damping coefficients are set to high values so that the character can closely follow the integrated poses.

Walking presents a unique challenge because the discrepancy between the virtual character and the user originates from the spatial limitation of the real world, rather than the differences in contact constraints. Due to the working range of the mocap system, the user's movement is restricted while the character can roam freely in the virtual world. In the walking case, we interpret that the end-effectors on the user's feet are constrained while those on the character's feet are unconstrained. Therefore, as long as both arms are imitating walking, our algorithm will start integrating the example motion and produce full body walking motion for the character. Our interface allows the user to have full control over the speed, direction, size of stride, and style of the walk.

8.4 Rock climb

- **G** : Each rock is represented by a single contact point and a radius defining the range at which contact is allowed.
- **C** : For a hand to engage the feature, it has to be closed and in the proximity of the rock. For a foot to engage the feature, it has to be in the proximity and above the face of the rock.
- **Q** : No example motion is used.
- **D** : For the hands, the spring and joints are the same as with the monkey bar. For the feet, they are the same as the floor.

8.5 Jump on a trampoline

- **G** : A trampoline is represented by a thin box.
- **C** : The same conditions used for walking.
- **Q** : No example motion is used.
- **D** : The coefficients for the active control are the same as the walking case. The trampoline is modeled as a damped spring attached to the floor. If the character jumps when the trampoline is compressed, the spring converts its potential energy to the character's kinetic energy at the take off.

We create a virtual trampoline to demonstrate the user's control over the dynamics of the character. By jumping at the appropriate moment, the user can make the character jump higher and higher. Many complex acrobatic motions can be easily produced using this interface. For example, to perform somersaults or twists, the user uses her arm movement to provide the desired angular momentum as the character takes off. The user then couches down or tucks in her arms to increase the angular velocity of the character. As the character is coming down, the user straightens her body again to prepare for landing.

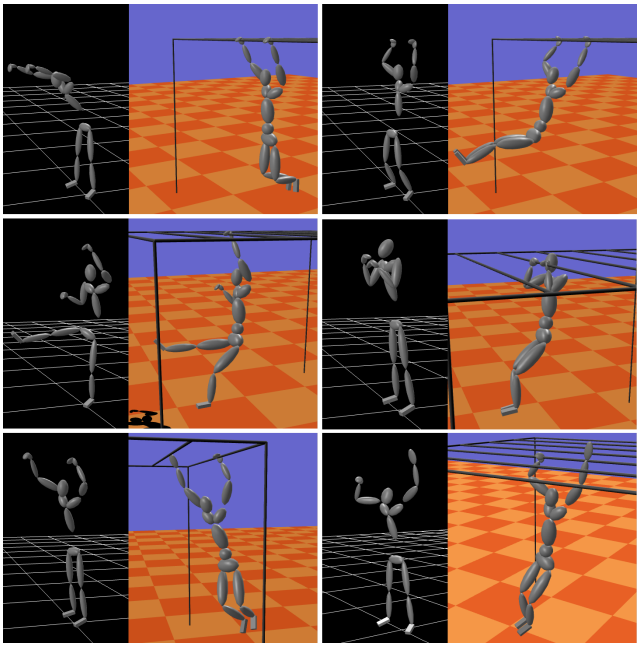

Figure 5: *Comparisons of the user's real-time motion (black background) and the modified character motion in the virtual world. Our system allows the user to interact with features in a free form manner. Top Row: The virtual character combines physics and an example motion to perform realistic swinging even though the user is constrained to the ground. Middle Left: Hanging by one hand while kicking. Middle Right: Chin-ups. Bottom Left: Traversing from bar to bar. Bottom Left: Traversing sideways.*

8.6 Swim in the water

- **G** : The swimming feature is represented as a box.
- **C** : The COM of the character must be within the confines of the box.
- **Q** : One cycle of the breast-stroke swimming motion is used.
- **D** : We apply buoyancy in the water to cancel the gravity. The drag and lift forces are computed from the movement of the hands relative to COM.

While in the water, the user can mimic a swimming motion with only her hands and the virtual character will begin swimming. By adjusting the speed and angle of her hands, the user can control the swimming speed and the direction.

9 Experiments

Our system runs at real-time on a single 3.4Ghz Pentium 4 CPU. We evaluate the applicability of our system by conducting two experiments. First, we test whether our system is robust when handling users with different body types and motion styles. Second, we explore the possibility of using our algorithm to develop a low-cost control interface.

9.1 Different users

We composed the features described in section 8 to design a mini-adventure game. Unlike the conventional platform games where the player can only advance in a few limited ways, our game allows

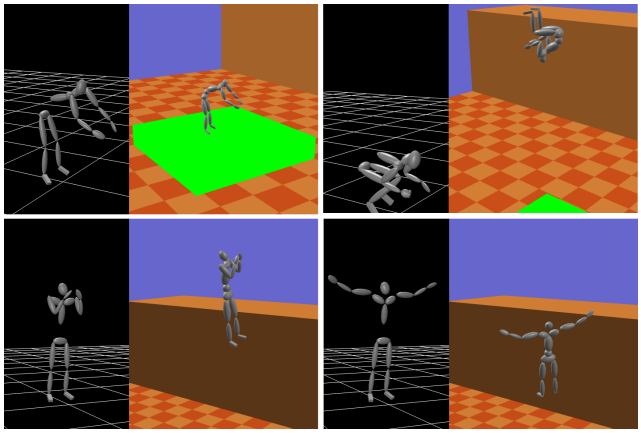

Figure 6: *The trampoline demonstrates the user's control over the dynamics of the character. Top Left: The user leans forward while jumping to generate forward angular momentum. Top Right: Tucking into a ball increases angular velocity, allowing the virtual character to flip. Bottom Row: Controlling the angular velocity by pulling in (left) and extending the arms (right).*

the user to freely manipulate any feature in the virtual world and discover creative strategies to accomplish tasks.

We tested our adventure game on six users, with age ranges from 10 to 31, weight ranges from 29.5 to 90.7 kg, and height ranges from 137 to 188 cm (Figure 7). Our system is robust even when the user's body type is significantly different from the subject performing in the example motion. As long as the user's end-effectors move in a reasonable way relative to the COM, the recognition algorithm can accruately determine the intention of the user.

Most users can successfully manipulate all the features in the adventure game with minimal instruction. Occasionally, some users found it difficult to precisely grasp the feature based on the projected 2D view of the virtual world. Another common mistake observed when manipulating monkey bar or rope features is to let go the grasp before a new contact is established. After falling a couple of times, most users quickly learned to perform more precisely when establishing contacts with the virtual world. We observed that users came up with many interesting, unexpected ways to interact with the features, such as kicking the side pole of the monkey bar to gain momentum and holding onto the edge of the wall to stop a fall.

9.2 Reduced marker set

The full marker set includes 46 retro-reflective markers placed on the user in a standard way. Based on the wrist joint angle reconstruction from the mocap system, the hand grasps are very difficult to identify. To circumvent this issue, we place an additional marker in the middle of each palm and use the visibility of this marker along with the reconstructed wrist angle to accurately determine whether the hand is closed or open.

Since our main algorithm only depends on the movement of end-effectors relative to the COM, our system in theory should produce reasonable results if the position of each end-effector and the root are provided in real-time. To this end, we design a reduced marker set that only places markers on the feet, hands, and the waist (Figure 7 left).

One difficulty with this reduced marker set is that we cannot use any commercial software provided by the motion capture system to label the markers according to the body parts due to the lack of underlying calibrated skeleton. Our solution is to design a unique pattern of markers for each end-effector as an identifier. This method allows us to robustly and efficiently identify the location of the end-effectors and the root. However, in order to form a pattern, this method requires multiple markers on each end-effector (6 on the hand, 3 on the foot, and 4 on the waist). We attached the markers to a pair of gloves, a pair of socks, and a belt, largely reducing the preparation effort for using our system.

Figure 7: *Left: We tested our system on a reduced marker set consists of 22 markers placed at the hands, feet, and the waist of the user. Middle and Right: Our system performs well when tested by different users.*

By using this reduced marker set, the user is able to control the virtual character in the same way as the full marker set. Our system was able to correctly recognize the user's intention and extract the appropriate poses from the example motion. The quality of the output motion, however, is significantly reduced in those features that heavily rely on the online user performance, such as chin-up on the monkey bar. Without complete poses from the real-time performance, the character can not perform actions largely different from the example motion. One potential solution is to employ a large dataset of motion sequences and integrate our physical model with the approach proposed by Chai and Hodgins [Chai and Hodgins 2005].

10 Discussion

We introduce an intuitive control interface for navigating and manipulating virtual worlds using real-time human motion performance. Our system leverages physical simulation to overcome the issues due to the discrepancy between the real world and the virtual world. We also describe an algorithm that accurately recognizes the user's intention in real-time. Based on the user's intention, our system integrates the online performance and offline example motions to produce realistic motion while preserving the personal style of the user. Our algorithm is robust against users with different motion style and body types.

Our system has several limitations. First and foremost, a full motion capture system is required in the described implementation. While cost is currently prohibitive for professional grade systems to be placed in domestic households, it is likely that in the near future we will see greater deployment of motion capture systems for use as performance capture devices. Further, low cost options for input are already becoming available and the potential for wide distribution only depends on the proper use and design. Our preliminary experiments with a reduced marker set suggest that low cost input devices, such as Wiimotes, can potentially replace the full motion capture system. Second, the current output device can also be largely improved. Our system projects the rendered virtual world onto a wall in front of the user. The virtual camera is set

behind the character's head to give the user a first-person view of the world. Often times, the user has a hard time perceiving distances in the world from the 2D projection, substantially affecting the user's ability to manipulate the virtual features. To improve the sense of immersion, more sophisticated output devices, such as a head-mounted display, can be integrated with our system. Another possibility is to let the user control the camera angle using body gestures.

There are several exciting directions for future work following directly from the described implementation. First, although it is fairly easy to design a new feature using our template, it still requires some coding and parameter tuning. One short-term future direction is to design a user-friendly interface that allows the player to create a new virtual feature using her own performance. The system would automatically adjust the physical parameters based the player's demonstration. Second, the current system does not support multiple character interaction. There are no major obstacles against including multiple characters from the respect of the motion capture performance, but displaying the characters with a shared display would likely be infeasible due to the limitations described. Also, the interplay between the two characters poses a new host of interesting problems. Finally, although our system is posed as an online gaming interface, but it has potential to be used as a performance-based synthesis tool for generating character animation. As such, a promising long-term direction is to extend our system to an offline animation tool for creating special effects.

To the best of our knowledge, the proposed interface represents one of the premiere investigations into the use of performance as a full-body, interactive interface. With our implementation and mini-adventure game testbed, we have shown the flexibility and power of the performance-control paradigm. Given the potential for short-term advances in hardware, we anticipate that performance has the potential for becoming a practical game interface in a short time horizon. We encourage others to strongly consider the use of motion performance in other interface problem domains.

Acknowledgements

We would like to thank Brian Davidson, Daniel French, Cameron Jasen, and Cassie Turk for their help with evaluation of the system. We are grateful to Marc Soriano for help with motion capture implementation. This work was supported by NSF CAREER award CCF 0742303.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Synthesizing constrained motions from examples. *ACM Trans. on Graphics 21*, 3, 483–490.

CHAI, J., AND HODGINS, J. K. 2005. Performance animation from low-dimensional control signals. *ACM Trans. on Graphics 24*, 3, 686–696.

CMU Motion Capture Database, http://mocap.cs.cmu.edu/.

DARRELL, T., AND PENTLAND, A. 1993. Space-time gestures. In *CVPR*, 335–340.

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layered acting for character animation. *ACM Trans. on Graphics 22*, 3, 409–416.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Symposium on Interactive 3D Graphics*, 139–148.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. 2004. Style-based inverse kinematics. *ACM Trans. on Graphics 23*, 3, 522–531.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. Spatial keyframing for performance-driven animation. In *Eurographics*, 107–115.

KEOGH, E., PALPANAS, T., ZORDAN, V., GUNOPULOS, D., AND CARDLE, M. 2004. Indexing large human-motion databases.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Trans. on Graphics 21*, 3, 473–482.

LASZLO, J., VAN DE PANNE, M., AND FIUME, E. L. 2000. Interactive control for physically-based animation. In *ACM SIGGRAPH*, 201–208.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *ACM SIGGRAPH*.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Trans. on Graphics 21*, 3, 491–500.

MÜLLER, M., RÖDER, T., AND CLAUSEN, M. 2005. Efficient content-based retrieval of motion capture data. *ACM Trans. on Graphics 24*, 3, 677–685.

NEFF, M., ALBRECHT, I., AND SEIDEL, H.-P. 2007. Layered performance animation with correlation maps. *Computer Graphics Forum 26*, 3, 675–684.

OORE, S., TERZOPOULOS, D., AND HINTON, G. 2002. Local physical models for interactive character animation. *Computer Graphics Forum 21*, 3, 337–326.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *ACM SIGGRAPH*, 11–20.

SHIN, H. J., LEE, J., GLEICHER, M., AND SHIN, S. Y. 2001. Computer puppetry: An importance-based approach. *ACM Trans. on Graphics 20*, 2, 67–94.

SHIRATORI, T., AND HODGINS, J. K. 2008. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Trans. on Graphics 27*, 5, 123:1–123:9.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: an interface for sketching character motion. *ACM Trans. on Graphics*, 424–431.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion warping. In *ACM SIGGRAPH*.

YAMATO, J., OHYA, J., AND ISHII, K. 1992. Recognizing human action in time-sequential images using hidden markov model. In *ICCV*, 379–385.

YIN, K., AND PAI, D. 2003. FootSee: an interactive animation system. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, 329–338.

ZHAO, P., AND VAN DE PANNE, M. 2005. User interfaces for interactive control of physics-based 3d characters. In *Symposium on Interactive 3D graphics and games*, 87–94.