Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2009)
E. Grinspun and J. Hodgins (Editors)

Interactive Synthesis of Human-Object Interaction

Sumit Jain† and C. Karen Liu†

Georgia Institute of Technology

Abstract

We present an interactive motion editing tool for creating dynamic scenes with human and object interaction. Our approach solves for an optimal control problem that leverages methods for physics-based rigid body control and kinematics-based human motion editing. Because the humans and the objects are coupled via physical contacts, our interface allows the animator to directly change the trajectories of humans or objects and simultaneously render the effect of the edits on the entire scene. Consequently, the animator can efficiently create complex interaction that requires precise synchronization, such as juggling. Our framework is generic to the choice of human motion editing method, as long as the differential information of the motion can be computed.

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

1. Introduction

A realistic depiction of human activities often involves complex interaction with the environment. In character animation, the most interesting and entertaining scenes, such as juggling seven balls or pitching and hitting a baseball, usually require the character to have meticulous control and skillful manipulation with passive objects. To create such a dynamic scene with complex human-object interaction, the passive objects must move according to the laws of physics while synchronizing temporally and spatially with human activities. Likewise, the character's movement must appear both controlled and agile when manipulating the objects.

Most existing techniques in computer graphics only provide a partial solution to this problem. Motion capture editing techniques have demonstrated the ability to synthesize realistic human motion. However, complex scenarios with human-object interaction are often difficult to acquire from the real world, because the motion quality directly depends on the capability of the performer. Even if one manages to capture both the human activity and the object motion in the scene, modifying such an animation is exceedingly challenging because any small modification can cause a dynamic inconsistency between the human and the objects. In contrast, physical simulation is a powerful tool to synthesize realistic passive objects [Bar89, Bar91]. Modifying the simulation results can also be achieved effectively using optimal control techniques [PSE*00]. Nevertheless, the dynamic controllers to date still fall short of producing most human activities, let alone handling precisely controlled interaction with objects.

Many state-of-the-art commercial animation tools provide a physics simulator to generate object motion, but require the animator to manually align and synchronize the simulated results with the character motion. Since the animator has to adjust the objects and the characters in two separate steps using two different methods, it takes a tremendous amount of manual editing and parameter tuning to synthesize dynamic human-object interaction. In light of the limitations of the current tools, we introduce a new animation technique that allows the animator to intuitively create complex scenes by simultaneously editing the objects and the characters. The animator can directly adjust the trajectories of the objects or of the characters while the correct dynamic coupling is maintained by the system. Our method interleaves the techniques of human motion editing and rigid body control. Through physical contacts, the characters provide active control forces to objects while the objects provide kinematic constraints to the characters.

Based on the previous work on interactive control of rigid bodies [PSE*00], we include the parameters of the character motion as part of the control variables for simulating rigid bodies. Using a differential update procedure, our method

† {sumit,karenliu}@cc.gatech.edu

computes the new parameters of the character motion that achieves the desired update of object motion. Our framework is generic to any human motion editing algorithm whose output motion can be expressed as a differentiable function of the motion parameters. We describe two simple motion editing algorithms to demonstrate the generality of our framework.

Our approach provides a few key advantages. First, the input human motion need not be precise because it will be modified according to the desired interaction with the objects. Consequently, input human motion can be approximately captured or animated by amateurs. Second, animating human-object interaction can be done by simply editing the objects. The system will automatically produce the character motion via coupled dynamic equations of motion. This allows the animator to focus on the high-level semantics of the animation, rather than adjusting detailed joint angles or control parameters. Third, solving the problem of human motion editing and rigid body control simultaneously is in fact simpler than solving them individually. Through interaction, the character simplifies the control of rigid bodies by providing active control to an under-actuated system, whereas the objects simplify the human motion editing by providing kinematic constraints to an under-determined optimization.

2. Related Work

Editing motion capture data is an active research area as the number of applications grows in the entertainment industry. Among the most difficult motions to modify is the physical interaction with the environment. To achieve physical effects of human-environment interaction from motion capture data, researchers have proposed methods for synthesizing motions that respond to external forces [AOF05, YPvdP05, ZMCF05], manipulate sustained contacts [LCL06], or interact with other characters [SKSY08, KHKL09]. These methods directly capture human-environment interaction and make minimal modification on the acquired motion data. Consequently, a large dataset of motion is often required to capture sufficient variability in the interaction. This approach becomes impractical when applied to human-object interaction, because dynamic interaction with passive objects is difficult to capture and the amount of data that covers all possible interactions is astronomical. Our method circumvents these issues by allowing the animator to intuitively edit the input motion after acquisition and using physics simulation to achieve variability in motion.

Kinematic methods for editing motion sequences have been widely studied and applied in character animation. Whether it is modifying a single input sequence [WP95, BW95, Gle97, LS99] or interpolating multiple sequences [RCB98, PB02, KG04, MK05], these method effectively change the input data according to user-specified constraints. Combined with planning algorithms that compute the trajectories of objects, kinematics methods can synthesize manipulation tasks of objects [KKKL94, YKH04]. These methods only work when the objects are fully actuated by the human (e.g. an object in a force-closure grasp). To manipulate dynamically varying passive objects, it is very difficult, if possible, to specify the character's kinematic constraints consistent with the object motion under the laws of physics. Our method does not need to explicitly set kinematic constraints on the character because we enforce the dynamic coupling between the character and objects.

Researchers in robotics have designed various controllers for robot interaction and manipulation of objects in a dynamic environment [AR87, And89, SA93, RK94] (among many). Because these controllers are carefully engineered for special tasks, modifying the resulting motion requires tuning of a large number of unintuitive control parameters. Consequently, it is difficult to adapt these methods to the purpose of computer animation where the animator needs to make frequent changes to the objects and the characters. Khatib and Sentis [Kha87, SK05] used operational-space formulation to control multiple operational tasks. Abe and Popović [AP06] applied the idea of operational-space control to computer animation and demonstrated that simple manipulation tasks can be achieved by directing the character towards prescribed positions or trajectories, such as the desired motion of manipulated objects. Our method takes one step further by allowing the user to directly modify the object trajectories at any given time. However, our system does not model the dynamic responses of the human induced by the impact of the objects.

Our work also builds upon prior art on control of passive dynamic systems. Researchers have applied various techniques to control rigid bodies [PSE*00, TJ08], smoke and water [TMPS03, FL04], and deformable objects [WMT06, BP08]. Our method is inspired by the interactive control interface for manipulating rigid body simulations [PSE*00]. This framework used a few physical parameters to control the simulation of passive objects. Our method extends their work to edit both human and object motion simultaneously. By coupling human motion in the dynamic equations of motion, the optimal control of simulated rigid bodies becomes more feasible, as human motion provides many additional control variables to the optimization problem.

3. Overview

We present an interactive animation interface to create dynamic scenes of human-object interaction. Based on an optimal control framework, our approach simultaneously synthesizes dynamically coupled character and object motions. The interface allows the animator to directly edit the trajectories of the characters or of the objects to create interesting interaction such as juggling.

Our work is built upon the previous work on interactive

control of simulated rigid bodies [PSE*00], reviewed in Section 4. We extend this framework to handle interaction between rigid bodies and humans in Section 5. Our framework is generic to any human motion editing algorithm, as long as the differential information of the motion can be computed. We describe two simple motion editing algorithms to demonstrate the generality of our optimal control framework. Finally, we describe the features that facilitate the process of making animation, including trajectory editing and constraints (Section 6).

4. Review: Optimal control for rigid body simulation

We give a brief overview of the optimal control problem for rigid body simulation. For complete details, we refer the reader to [PSE*00, Pop01].

The state of the rigid body is represented as $\mathbf{q} \equiv (\mathbf{x}, \mathbf{r}, \mathbf{v}, \omega)^T$, where $\mathbf{x} \in \mathbb{R}^3$ is the position, $\mathbf{r}$ is a quaternion representing the orientation, $\mathbf{v} \in \mathbb{R}^3$ is the linear velocity and $\omega \in \mathbb{R}^3$ is the angular velocity.

The time evolution of the rigid body in a free-flight is governed by the following ordinary differential equation:

$$\frac{d}{dt}\mathbf{q}(t) = \mathbf{F}(t, \mathbf{q}(t), \mathbf{u}) \tag{1}$$

where $\mathbf{F}$ is derived from Newton's laws of motion and $\mathbf{u}$ denotes the set of parameters to control the simulation, e.g. initial position and velocity of the body. Integrating this equation from time t_0 to t_f, we get:

$$\mathbf{q}(t_f) = \mathbf{q}(t_0) + \int_{t_0}^{t_f} \mathbf{F}(t, \mathbf{q}(t), \mathbf{u})dt \tag{2}$$

The above equation does not incorporate collisions since a collision introduces discontinuity in the simulation and needs to be handled separately. The states just before and after a collision at time t_c, $\mathbf{q}(t_c^-)$ and $\mathbf{q}(t_c^+)$ respectively, are related by the following equation:

$$\mathbf{q}(t_c^+) = \mathbf{q}(t_c^-) + \mathbf{I}(\mathbf{q}(t_c^-), \mathbf{u}) \tag{3}$$

where $\mathbf{I}$ is the impulse of the collision. For example, in the case of collision of a sphere with a static plane, the resulting velocity of the sphere $\mathbf{v}^+$ after the collision is given by:

$$\mathbf{v}^+ = \mathbf{v}^- - (1+r)(\mathbf{v}^-.\hat{\mathbf{n}})\hat{\mathbf{n}} \tag{4}$$

where r is the coefficient of restitution and $\hat{\mathbf{n}}$ is the normal of the plane. In this case, impulse $\mathbf{I} = -(1+r)(\mathbf{v}^-.\hat{\mathbf{n}})\hat{\mathbf{n}}$. For the general case of collision of two rigid bodies, refer to [Pop01].

To add more degrees of freedom to the simulation, physical parameters like coefficient of restitution and normal vector are included in the control vector $\mathbf{u}$, making the impulse directly dependent on $\mathbf{u}$. We group these parameters including the initial state and denote them by $\mathbf{u}_r$. Therefore, the current set of control parameters $\mathbf{u} = \{\mathbf{u}_r\}$.

Now, the state of the rigid body at any time can be abstractly written in terms of a simulation function as:

$$\mathbf{q}(t) = S(t, \mathbf{u}) \tag{5}$$

This simulation function is composed of free flight functions between the collisions (Equation 2) and collision resolution functions at the collisions (Equation 3). In an interactive framework, the user changes the state of the rigid body at time t_i, $\delta\mathbf{q}(t_i)$. The optimal control problem aims at computing the required change in control variables $\delta\mathbf{u}$ such that the user changes are realized:

$$\delta\mathbf{q}(t_i) = \frac{\partial S(t_i, \mathbf{u})}{\partial \mathbf{u}}\delta\mathbf{u} \tag{6}$$

This is a linear equation in the unknowns $\delta\mathbf{u}$. Solving simultaneously for n such user changes with many control parameters is an under-constrained problem. Therefore, the user can weight the change in control variables leading to a quadratic objective function $\delta\mathbf{u}\mathbf{D}\delta\mathbf{u}$. The resulting optimization problem can be written as:

$$\begin{aligned} &\arg\min_{\delta\mathbf{u}} \quad \delta\mathbf{u}\mathbf{D}\delta\mathbf{u} \\ &\text{subject to} \quad \left\{ \delta\mathbf{q}(t_i) = \tfrac{\partial S(t_i,\mathbf{u})}{\partial \mathbf{u}}\delta\mathbf{u},\ i = 1, \cdots, n \right. \end{aligned} \tag{7}$$

Once $\delta\mathbf{u}$ is solved, the existing control vector $\mathbf{u}$ is updated by taking a small step:

$$\mathbf{u} \leftarrow \mathbf{u} + \varepsilon\delta\mathbf{u} \tag{8}$$

5. Human motion editing

To incorporate human motion in the simulation, we extend the optimal control framework described in Section 4 by including human motion parameters as part of control vector $\mathbf{u}$. The change of human motion is dynamically coupled with the change of object motion via Equation 6.

We represent an articulated character's skeleton in reduced coordinates with 1 to 3 rotation degrees of freedom (DOFs) for each joint and 6 DOFs for the root of the hierarchy including global translation and rotation. In our examples, the joint configuration is represented by a total of 42 DOFs. The value of j^{th} DOF h_j at time sample t_k is represented as $h_j(t_k)$. The joint configuration of the character at t_k is represented as $\mathbf{h}(t_k)$. The input motion $\mathbf{H}$ for a character is represented as a sequence of these DOFs in time $\{\mathbf{h}(t_0), \mathbf{h}(t_1), \cdots, \mathbf{h}(t_{n-1})\}$.

Unlike rigid bodies, we do not physically simulate human motion. Instead, we use a kinematic method to edit the input motion $\mathbf{H}$. Our framework can work with any algorithm for human motion editing, as long as we can represent a joint angle configuration $\mathbf{h}$ as a differentiable function of the motion parameters included in $\mathbf{u}$. We first introduce a primitive local joint angle editing method, and then we enhance it with a linear motion blending technique.

5.1. Spline-based joint angle editing

Since human motion is usually smooth, we use a sparse representation for time sequences of DOFs rather than using samples at every frame. In our implementation, we fit cubic splines to each of the DOFs. This gives us an advantage of representing DOFs in a continuous and differentiable manner while approximating the samples well with much less number of control points (representing the spline). We place the control points for each of the DOF at every 10-15 frames of the original motion. We denote the control points for DOF h_j as $\mathbf{c}_j$. The set of control points for all the DOFs $\mathbf{h}$ is denoted as $\mathbf{c}$. Note that for any time t, the value of the DOF depends on a local neighborhood of control points (four in our case).

The human-object interaction happens when objects are in contact with human body parts. Because the human motion can introduce contacts, remove contacts, and determine the normal direction and the velocity at contact, we include the control points of human motion $\mathbf{c}$ in the control vector $\mathbf{u}$ for rigid body simulation (Equation 5), along with control parameters $\mathbf{u}_r$, i.e. $\mathbf{u} = \{\mathbf{c}, \mathbf{u}_r\}$. The impulse equation (Equation 3) can be re-written as:

$$\mathbf{q}(t_c^+) = \mathbf{q}(t_c^-) + \mathbf{I}(\mathbf{q}(t_c^-), \mathbf{q}_h(t_c^-, \mathbf{u}), \mathbf{u}) \tag{9}$$

where $\mathbf{q}_h$ is the state (position, velocity etc.) of the human body part in contact at the time of collision t_c. Given the joint configuration $\mathbf{h}(t_c^-; \mathbf{c})$, $\mathbf{q}_h$ can be computed using the transformation hierarchy of the human skeleton. Our current method does not synthesize passive reaction of the human due to the impact from a collision.

5.2. Linear motion blending

Motion editing method described in the previous section produces only local and small-scale changes in each DOF. To synthesize long-duration or contact changes, we use a motion blending technique to continuously interpolate a set of motion clips.

Suppose we have two motion clips, $\mathbf{H}$ and $\mathbf{H}'$, as input. We begin by selecting a set of corresponding keyframes in $\mathbf{H}$ and $\mathbf{H}'$. The first and last keyframes in $\mathbf{H}$ define the time interval $[t_s, t_e]$. Based on the selected keyframes, $\mathbf{H}'$ is aligned and time warped such that the time of the corresponding events coincide with that in $\mathbf{H}$.

We introduce a parameter α to interpolate between motions $\mathbf{H}$ and $\mathbf{H}'$ in time interval $[t_s, t_e]$, yielding a new motion $\mathbf{H}^\alpha$:

$$\mathbf{H}^\alpha(\alpha, \mathbf{c}, \mathbf{c}') = (1-\alpha)\mathbf{H}(\mathbf{c}) + \alpha\mathbf{H}'(\mathbf{c}'). \tag{10}$$

where $\mathbf{c}$ and $\mathbf{c}'$ are the control points for motion $\mathbf{H}$ and $\mathbf{H}'$ respectively. These control parameters from human motion $\mathbf{H}^\alpha$ are added to the control vector $\mathbf{u}$, i.e. $\mathbf{u} = \{\alpha, \mathbf{c}, \mathbf{c}', \mathbf{u}_r\}$. Given desired user changes, these parameters are computed in the optimization (Equation 7) such that the interpolated motion satisfies user edits. To facilitate larger changes in α, we appropriately scale it in the optimization process. To ensure the smooth transition in and out of interval $[t_s, t_e]$, we blend the interpolated motion $\mathbf{H}^\alpha$ with $\mathbf{H}$ or $\mathbf{H}'$ (as desired) outside this interval.

This interpolation scheme can be applied to multiple motion clips. We assume that these motions are stylistically different but performing similar actions in the interpolated time interval. In addition, interpolation changes will occur while editing only if the user edits can be interpolated by the input motion clips. For example, if the input motion clips include a leaning forward sequence, the juggler will interpolate that action to reach forward as the animator drags the hand further away from him. On the other hand, if the animator drags the hand in the sideways direction, there is no interpolated motion that lies close to that change. Therefore, the original motions will be modified without α being changed.

This interpolation can cause certain artifacts in the blended motion at contacts such as feet slippage. This can be corrected as a post processing operation on the synthesized motion.

6. Interactive control interface

Our interface provides two features to allow the animator to efficiently edit complex dynamic scenes with human-object interaction.

6.1. Trajectory editing

The animator can directly pose the human by dragging any parts of the body in the Cartesian space. Our system will automatically update the joint trajectories and the object motion accordingly. The control points $\mathbf{c}$ (part of the control vector $\mathbf{u}$) are adjusted for each DOF such that the $\delta\mathbf{q}_h$ specified by the animator is satisfied. We include the following linear constraint in the optimization (Equation 7):

$$\delta\mathbf{q}_h = \frac{\partial \mathbf{q}_h(t_i, \mathbf{u})}{\partial \mathbf{u}} \delta\mathbf{u} \tag{11}$$

The animator can also change the state of the objects such as position and orientation at any time instance. The constraint for this manipulation is given by Equation 6. This change in the state depends on the control variables of human motion through Equation 9. The computed change $\delta\mathbf{u}$ may violate any prior edits made by the animator; therefore, constraints can be added to preserve these while editing as discussed in the next section (Section 6.2).

In practice, these two types of trajectory editing provide different advantages to facilitate the animating process. Editing object trajectories allows the animator to directly manipulate in the task space, significantly simplifying the process of animating human with a large number of DOFs. On the other hand, the ability to pose human directly can be very

useful for refining the animation. Our system provides this precise control while shielding the animator from tuning any unintuitive physical parameters.

6.2. Constraints

The optimal control framework (Equation 7) is generic to any constraint formulated as a function of $\delta\mathbf{u}$. To facilitate more precise control in motion synthesis, our framework allows the animator to enforce the following types of constraints:

Grasp constraint: Apart from passively colliding with the objects, the animator can choose to grasp the object once the collision has been detected. During the grasping, the state of the object is defined by the state of the human body part it is in contact with i.e. $\mathbf{q}(t) = \mathbf{q}_h(t,\mathbf{u})$, $t_c \leq t \leq t_f$. After a user-specified grasping duration, the object follows free flight motion as in Equation 2. The initial state of this free flight segment is given by the state of the grasping body part $\mathbf{q}_h(t_f,\mathbf{u})$, which depends on human control variables $\mathbf{c}$.

State constraint: Similar to the constraints between rigid bodies described in [PSE*00], the animator can choose to constrain the state of the object or the human body part at some specified time t_i. The constraint is written as in Equation 6. This constraint can be used to fix the position of an object in space or control the change in its position or velocity as directed by the user.

Collision constraint: The animator can choose to maintain a collision between a human body part and the object by constraining the relative change in positions. The timing of the collision t_c and the contact position in the world coordinates are free to change as they depend on the control vector $\mathbf{u}$. The constraint can be written as:

$$\begin{aligned} 0 &= \delta\left(\mathbf{q}(t_c(\mathbf{u})) - \mathbf{q}_h(t_c(\mathbf{u}),\mathbf{u})\right) \\ &= \left(\left(\mathbf{F} - \frac{\partial \mathbf{q}_h}{\partial t}\right)\frac{\partial t_c(\mathbf{u})}{\partial \mathbf{u}} + \frac{\partial \mathbf{q}}{\partial \mathbf{u}} - \frac{\partial \mathbf{q}_h}{\partial \mathbf{u}}\right)\delta\mathbf{u} \end{aligned} \quad (12)$$

where $\mathbf{F}$ is defined in Equation 1 and $\frac{\partial \mathbf{q}}{\partial \mathbf{u}}$ and time Jacobian $\frac{\partial t_c(\mathbf{u})}{\partial \mathbf{u}}$ are computed as described in [PSE*00]. Terms $\frac{\partial \mathbf{q}_h}{\partial t}$ and $\frac{\partial \mathbf{q}_h}{\partial \mathbf{u}}$ are computed using the transformation hierarchy of the human skeleton. If desired, the time of collision can be fixed by zeroing out the time Jacobian term in the above equation.

7. Results

We now present our results for interactive synthesis of various human-object interactions. An optimization (QP) is solved at each step to compute the change in control variables. The number of variables being solved for ranges from four to around two hundred. Our interface runs at interactive rates on a single core of Intel Core2 Duo 2.8 GHz. We demonstrate our control framework through the following examples making use of one or more editing techniques described in Section 6. We refer the reader to the supplementary results video for complete animations and editing demonstrations.

Soccer ball juggling. We motion captured a naive subject mimicking the juggling of a soccer ball with his feet, knees and head. We interactively edited the motion by editing the trajectory of the soccer ball involved in a sequence of collisions with the body parts of the virtual character. In the process, we drastically changed the trajectory of the ball while the system computed appropriate changes in the velocity and position of the body part in contact with the ball. The change in human motion is subtle in this example because a little change in direction and speed of the human motion at the contact can effectively adjust the object trajectories. The final demo took 15-20 minutes to create. Figure 1 compares the input motion of the character and the initial trajectory of the soccer ball with the interaction synthesized using our framework.

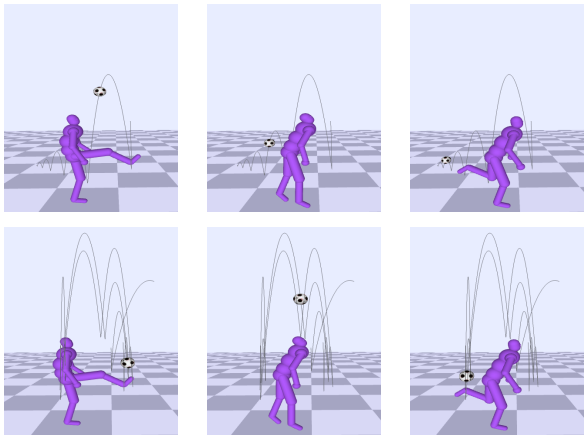


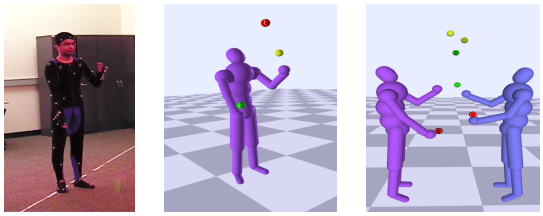
Figure 1: *Comparison of position and trajectory of the soccer ball at different time instances in the input motion sequence (top row) and the motion synthesized using our interactive framework (bottom row).*

Hand juggling three balls. The input was a crude motion with the subject arbitrarily pretending to juggle. We enhanced this input human motion to proficiently interact with three juggling balls. In our final sequence, the juggler was able to juggle three balls while handling accidental collisions. In addition, he was also able to drastically change his routine motion to save a ball. We used the following editing features described in Section 6.

- *Collision constraints:* We edited each contact of the ball and the hand sequentially. However, manipulating the trajectory of the ball can easily affect the previously edited contacts or cause uncalled-for collisions with other balls. We used a floating collision constraint such that an edited contact between the hand and the ball can be preserved during later editing.

- *Grasp constraints:* In addition to impulsive contacts, we manipulated the trajectory of the objects by applying a grasp constraint and releasing it at user-defined time instances.
- *Human trajectory edits:* Since the trajectories of the balls were fairly close to each other, collisions were inevitable. In the process, we made some minor adjustments by directly dragging the human body parts to avoid some collisions.
- *Motion blend:* To add interesting interaction in the sequence, we "forced" the juggler to make a bad throw. The resulting sequence is an interpolated motion between a normal catch and a forward-leaning catch.

It took 30-40 minutes of user interaction to synthesize the full sequence. Figure 2(a) and Figure 2(b) compare a snapshot of the captured motion with the interaction synthesized by our framework.



(a) (b) (c)

Figure 2: *Comparison of the original motion capture and the human-object interaction synthesized by our interactive framework.*

Multiple characters interacting via objects. Our interface can also synthesize dynamic scenes with multiple characters. Because these characters are interacting with one another through the contacts with objects, changing one character's motion will modify the object motion, which in turn changes other characters. We demonstrate a baseball pitching and batting example. We interactively repositioned the batter, while the pitching motions and blending factor were both changed accordingly, resulting in a new interpolated pitching motion and baseball trajectory. We also synthesized a sequence of two characters juggling six balls together. The editing complexity for this example was increased due to presence of many balls in the air at the same time. We edited both the trajectories of the balls and hands to maintain proper grasps and avoid uncalled-for collisions. Figure 2(c) shows a snapshot from the synthesized sequence.

8. Conclusions

We presented an intuitive animation interface for creating dynamic scenes with complex human-object interaction. Our approach solves for the problems of human motion editing and rigid body control simultaneously via dynamic equations that couple the movement of the characters and the objects. As a result, the animator can directly edit the trajectories of characters or objects while the correct dynamic coupling is maintained. Our method is generic to various human motion editing algorithms, as long as the joint configuration can be expressed as a differentiable function of the motion parameters. We have demonstrated that two different human motion editing methods, spline-based joint editing and linear motion blending, can be easily applied to the proposed optimal control framework.

Our results show that joint trajectories and blending factors are automatically modified as the animator changes the object trajectories. Our primitive motion editing methods produce smooth recovery motion, but do not guarantee the modified human motion to be realistic or natural when the user makes large deviation from the original motion. Furthermore, our current implementation is not able to handle the situations when the animator specifies an interaction that cannot be achieved by the input motion. One future direction is to incorporate physics-based human motion editing methods or leverage more motion data.

Our framework also suffers from inherent limitations of differential updates. Since constraints are linearized in the change of control vector, large steps may cause divergence in updates. Owing to non-linearity of the problem, this approximation results in slow drift in the constraints (Section 6.2) which may eventually get violated after repeated updates. Our current implementation limits us to work in continuous domain for differential updates. A potential future direction is to incorporate a discrete search for human motion editing similar to the method proposed by Kim *et al.* [KHKL09].

Kinematics-based human motion editing simplifies the dynamic relation between the object and the human. The main drawback, however, is that the human does not respond to the impact of collision. We could instead use a physically simulated character by treating the character as an articulated rigid-body system actuated by some control variables. The linear approximation in differential updates would result in extremely small step sizes for updates (Equation 8) due to a highly nonlinear relation between object motion and human actuators. Furthermore, simulating human motion after each update would be computationally expensive. Designing new algorithms to compute the differential updates efficiently and accurately for a dynamically controlled character can be a challenging future direction.

References

[And89] ANDERSON R. L.: Dynamic sensing in a ping-pong playing robot. *IEEE Trans. Robotics and Automation 5*, 6 (Dec. 1989), 723–739. 2

[AOF05] ARIKAN O., O'BRIEN J. F., FORSYTH D. A.: Pushing people around. In *Eurographics/SIGGRAPH Symposium on Computer Animation* (2005), pp. 59–66. 2

[AP06] ABE Y., POPOVIĆ J.: Interactive animation of dynamic manipulation. In *Eurographics/SIGGRAPH Symposium on Computer Animation* (2006). 2

[AR87] ABOAF C. G. A. E. W., REINKENSMEYER D. J.: *Task-Level Robot Learning: Ball Throwing*. Tech. Rep. AIM-1006, MIT Artificial Intelligence Laboratory, Dec. 1987. 2

[Bar89] BARAFF D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *SIGGRAPH* (July 1989), vol. 23, pp. 223–232. 1

[Bar91] BARAFF D.: Coping with friction for non-penetrating rigid body simulation. In *SIGGRAPH* (July 1991), vol. 25, pp. 31–40. 1

[BP08] BARBIC J., POPOVIC J.: Real-time control of physically based simulations using gentle forces. *ACM Trans. Graph 27*, 5 (2008), 163. 2

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *SIGGRAPH* (Aug. 1995), pp. 97–104. 2

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 441–448. 2

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *Symposium on Interactive 3D Graphics* (Apr. 1997), pp. 139–148. 2

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 559–568. 2

[Kha87] KHATIB O.: A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation RA-3*, 1 (Feb. 1987), 43–53. 2

[KHKL09] KIM M., HYUN K. L., KIM J., LEE J.: Synchronized multi-character motion editing. *ACM Trans. on Graphics (SIGGRAPH) 28*, 3 (Aug. 2009). 2, 6

[KKKL94] KOGA Y., KONDO K., KUFFNER J., LATOMBE J.-C.: Planning motions with intentions. In *SIGGRAPH* (July 1994), pp. 395–408. 2

[LCL06] LEE K. H., CHOI M. G., LEE J.: Motion patches: Building blocks for virtual environments. *ACM Trans. on Graphics (SIGGRAPH) 25*, 3 (July 2006), 898–906. 2

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH* (Aug. 1999). 2

[MK05] MUKAI T., KURIYAMA S.: Geostatistical motion interpolation. *ACM Transactions on Graphics 24*, 3 (July 2005), 1062–1070. 2

[PB02] PULLEN K., BREGLER C.: Motion capture assisted animation: Texturing and synthesis. *ACM Trans. on Graphics (SIGGRAPH) 21*, 3 (July 2002), 501–508. 2

[Pop01] POPOVIĆ J.: *Interactive Design of Rigid-Body Simulations for Computer Animation*. PhD thesis, Carnegie Mellon University, July 2001. 3

[PSE*00] POPOVIĆ J., SEITZ S. M., ERDMANN M., POPOVIĆ Z., WITKIN A. P.: Interactive manipulation of rigid body simulations. In *SIGGRAPH* (July 2000), pp. 209–218. 1, 2, 3, 5

[RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (Sept. 1998). 2

[RK94] RIZZI A. A., KODITSCHEK D. E.: Further progress in robot juggling: Solvable mirror laws. In *ICRA* (1994), pp. 2935–2940. 2

[SA93] SCHAAL S., ATKESON C. G.: Open loop stable control strategies for robot juggling. In *ICRA (3)* (1993), pp. 913–918. 2

[SK05] SENTIS L., KHATIB O.: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics 2*, 4 (2005), 505–518. 2

[SKSY08] SHUM H., KOMURA T., SHIRAISH M., YAMAZAKI S.: Interaction patches for multi-character animation. *ACM Trans. on Graphics (SIGGRAPH) 27*, 5 (Dec. 2008), 114:1–114:8. 2

[TJ08] TWIGG C. D., JAMES D. L.: Backward steps in rigid body simulation. *ACM Trans. Graph 27*, 3 (2008). 2

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. *ACM Trans. on Graphics (SIGGRAPH) 22*, 3 (July 2003), 716–723. 2

[WMT06] WOJTAN C., MUCHA P. J., TURK G.: Keyframe control of complex particle systems using the adjoint method. In *ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2006), pp. 15–23. 2

[WP95] WITKIN A., POPOVIĆ Z.: Motion warping. In *SIGGRAPH* (Aug. 1995). 2

[YKH04] YAMANE K., KUFFNER J. J., HODGINS J. K.: Synthesizing animations of human manipulation tasks. *ACM Trans. Graph 23*, 3 (2004), 532–539. 2

[YPvdP05] YIN K., PAI D. K., VAN DE PANNE M.: Data-driven interactive balancing behaviors. In *Pacific Graphics* (Oct. 2005). 2

[ZMCF05] ZORDAN V. B., MAJKOWSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. *ACM Trans. on Graphics (SIGGRAPH) 24*, 3 (July 2005), 697–701. 2