Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)
K. Anjyo, P. Faloutsos (Editors)

Pushing People Around

Okan Arikan[†] David A. Forsyth[‡] James F. O'Brien[†]

[†]University of California, Berkeley [‡]University of Illinois, Urbana-Champaign

Abstract

We present an algorithm for animating characters being pushed by an external source such as a user or a game environment. We start with a collection of motions of a real person responding to being pushed. When a character is pushed, we synthesize new motions by picking a motion from the recorded collection and modifying it so that the character responds to the push from the desired direction and location on its body. Determining the deformation parameters that realistically modify a recorded response motion is difficult. Choosing the response motion that will look best when modified is also non-trivial, especially in real-time. To estimate the envelope of deformation parameters that yield visually plausible modifications of a given motion, and to find the best motion to modify, we introduce an oracle. The oracle is trained using a set of synthesized response motions that are identified by a user as good and bad. Once trained, the oracle can, in real-time, estimate the visual quality of all motions in the collection and required deformation parameters to serve a desired push.

Our method performs better than a baseline algorithm of picking the closest response motion in configuration space, because our method can find visually plausible transitions that do not necessarily correspond to similar motions in terms of configuration. Our method can also start with a limited set of recorded motions and modify them so that they can be used to serve different pushes on the upper body.

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

1. Introduction

Interactive environments such as training simulations and computer games require interactive characters that react to their environments in a visually plausible way. A direct way of interacting with a character is by applying forces to him/her. In this paper, we introduce a method for synthesizing motions for a character receiving forces from its environment due to collisions or other physical actions. Our method focuses on brief impulses (pushes), rather than sustained forces. We will synthesize motions that react to such pushes and restore balance without falling down (figure 1).

Generating these motions is a difficult problem because the dynamics of recovering from a push are complicated. A forceful push leaves a person in an unbalanced state and may require taking multiple guard steps to recover. Since balance is restored after a series of maneuvers that are off balance, purely physically based methods may have difficulty synthesizing a recovery motion. Due to this complexity many computer games switch to a rag doll simulation when a character is shot or pushed, and simply let the character fall down.

We will start with a set of motions that were recorded from a real actor using *motion capture*. Some of these motions are everyday motions such as standing, walking and running, and some of them are motions of the actor being pushed by someone else while performing these everyday motions. We will call the example motions that react to a push, *response* motions. We specifically focus on responses that restore balance without falling down. When the character receives a push, we will *serve* it by transitioning from

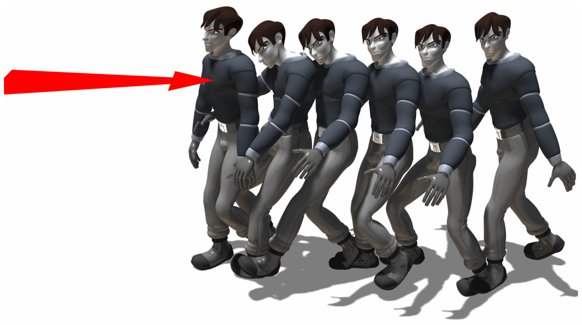


Figure 1: *This figure is a time lapsed shot of a motion synthesized using our method. Character receives a push on the chest from left (indicated as the red arrow) and takes protective steps backwards to restore balance.*

the current configuration of the character to some available response motion, just before the actor was pushed. Realistically, it is not plausible to capture enough motions so that this algorithm would work for a character being pushed any time, at any location on the body, from any direction.

Another problem for this baseline algorithm is deciding whether the character can transition from the current body configuration to the body configuration in the recorded response motion. Traditionally, to compute whether we can switch from one motion to another, the motions (joint positions / angles) are compared numerically to see if they are similar to within a user specified threshold. This method is not ideal, because the similarity threshold for transitions must be set very conservatively to guarantee good looking transitions. However, sometimes transitioning between numerically dissimilar motions can be visually plausible whereas sometimes transitioning between numerically similar motions can be detrimental to visual quality.

We introduce a parametric deformation model (Section 3.2) that takes a recorded response to a push, and modifies it so that the motion can be used for being pushed from a similar but different direction. Our deformation model is not perfect: some parameters can lead to implausible motions such as bad kinematic configurations or bad body dynamics.

When a push arrives, we need to find which response motion should serve the push and which deformation parameters must be used on the response motion. To answer these queries, we introduce an oracle (Section 5) that estimates the transitions and deformations that produce realistic motions. This oracle is trained interactively, using machine learning algorithms on a set of user identified realistic and unrealistic motions (off balance motion, impossible human configurations, bad dynamics etc.). After the training, our synthesis algorithm uses the oracle, in real-time, to generate natural looking motions that respond the pushes. We validate our synthesis algorithm by performing user studies (Section 6).

Our animated character can be pushed anywhere on the upper body, from any direction. Our method generates good looking response motions that maintain balance and respond to most pushes, but the method can fail if the user applies pushes for which no plausible response can be synthesized. Our quality oracle allows the algorithm to quickly find good response motions (with corresponding deformation parameters), or to detect that no response generates a good motion. Since we can detect when our method fails, we can use another way of handling an external force, such as switching to a physically based simulator and letting the character fall over.

2. Related Work

A simple way of generating motion is rearranging pieces from previously created motions. An important problem is finding a re-arrangement such that the synthesized motion meets certain objectives. Different algorithms [MTH00, KGP02, AF02, LCR*02] solve this problem by performing a combinatorial search. However, methods that build on rearrangement of motion pieces cannot synthesize truly novel body configurations, because every frame they synthesize is a frame in the original collection of motions they start with.

Physically based methods generate truly novel body configurations using equations of motion. Such methods typically formulate the synthesis as an optimization [WK88, FP03]. Controller based approaches, such as [BMH98, HWBO95, HP97] formulate the optimization as a parameter search for force/torque generators (controllers). Although impressive results have been obtained for systems such as bird flight [WP03], realistic full body human motion is still too complex (due to complex human anatomical structure as well as stylistic variations) for purely physics driven synthesis. Hybrid methods that start from human motion or biomechanics data have also been explored. [PW99, LP02, ALP04] provide different ways of modifying input motion data to obtain novel motions that attain different objectives.

[ZH99, ZH02] present a method utilizing a physically based simulator to track a previously recorded motion. The character they simulate can respond to external pushes; however, this method cannot generate protective steps to recover balance after a forceful hit. [FvdPT01a, FvdPT01b] presents a controller based approach for recovering balance (or falling down if not possible) after a push. Their work uses physically based models whereas we take a more data driven approach to preserve style and interactivity. [ZMCF05] introduces a physically based model where the character is ballistically simulated to joint to another motion in the database of motions. Their method is well suited towards handling motions that fall down and our method is geared towards recovery motions that maintain balance. It would be very interesting to combine two approaches as they naturally complement each other.

[GMHP04] uses machine learning for developing a generative model of human poses, and [GT95] used a learned model to approximate the physics of an animated character. [AFO03] also uses machine learning techniques to identify actions from motion sequences. Our oracle uses a similar idea in the broader context of recognizing visually realistic motions.

The details of human movement are produced by extremely complex interactions between the central nervous system, the spinal cord, various motor neurons and the musculo-skeletal system. Human observers appear to be exquisitely sensitive to some of these details, such as variations due to both anatomical structure and individual style. For identifying types of motion from light-dot displays, see [Joh73]; for identifying friends from light-dot displays, see [CK77]; for gender recognition from light-dot displays, see [BCK78, CPK78]; for the effect of rendering algorithms on the perceived reality of a motion, see [HOT98].

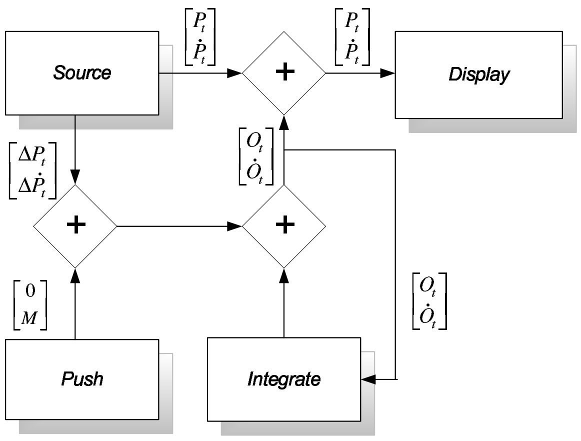

Source
Display
Push
Integrate
$\begin{bmatrix} P_t \\ \dot{P}_t \end{bmatrix}$
$\begin{bmatrix} \Delta P_t \\ \Delta \dot{P}_t \end{bmatrix}$
$\begin{bmatrix} O_t \\ \dot{O}_t \end{bmatrix}$
$\begin{bmatrix} 0 \\ M \end{bmatrix}$
+

Figure 2: *Our synthesis pipeline is organized as above. A motion source generates joint positions and velocities for every frame. An offset vector to joint positions and their velocities is maintained and integrated to die off to zero. A user push introduces additional joint velocities which are incorporated into the derivative of the offset vector. Any discontinuities due to transitions are computed and incorporated into the offset vector to create smooth motion. After the offset vector is added to the original joint positions, we display the character.*

Some large changes to a motion are innocuous while other, slight disturbances can produce a significant degradation in perceived quality. It remains mysterious how one might predict what kinds of change will prove acceptable in general, but there is good reason to believe that reactions to pushes are particularly easy to modify effectively. This is because, for somewhere between 100-250ms after the push has arrived, the central nervous system can play no part in the quality of the reaction (total proprioceptive feedback delays are of the order of 100-150ms [FC89]; total visuomotor feedback delays are of the order 200-250ms [GKM81]; a general discussion of delays appears in [MWWS93]). Furthermore, it is now well-established that feedback loops within muscles and through the spinal arc are an important part of motor control (e.g. [Mia95]). This previous work suggests that it is reasonable to model muscles as springs for a short period of time after a push.

3. Push Response

We represent body configurations with joint positions and velocities in a global coordinate frame. In order to display a body configuration, we need to convert this representation to root position and joint angles. We do this by formulating inverse kinematics as a minimization that tries to find the joint angles that produce joint positions closest to the given configuration. We also use the joint angles in the previous frame as the starting point for this numerical optimization. A motion consists of body configurations sampled at a fixed frequency (60 frames per second in our case). Let P_t denote the vector of all joint positions (if the character has 30 joints, P_t is 90 dimensional). We then define $C_t = [P_t \;\; \dot{P}_t]$ as the body configuration at time t.

3.1. Motion Transition

To handle a push, the character will transition from the body configuration at the time of the push to the beginning of a previously recorded response motion (just before the actor was pushed). At this transition point, the character starts playing frames from the response motion (we rotate and translate the response motion so that it starts from the position and orientation of the character when he was pushed). This naïve way of transitioning creates an unavoidable discontinuity in the motion (in joint angles or joint positions) at the transition point. Offline synthesis algorithms deal with this discontinuity by modifying the motion before and after the transition, so that joint positions (or angles) are continuous. A real-time algorithm cannot use this strategy, because the portion of the motion before the transition point has already been displayed to the user and hence cannot be modified.

To avoid discontinuity, we maintain an offset vector, O_t (see Figure 2). At every frame, this offset vector and its time derivative are added to the joint positions and velocities before the character is displayed: $C_t = [P_t + O_t \;\; \dot{P}_t + \dot{O}_t]$. After the offset is added, it is integrated according to the following second order ODE:

$$\ddot{O}_t = K_s O_t + K_d \dot{O}_t \qquad (1)$$

In this equation, K_s and K_d are stiffness and damping terms that pull O_t to zero. In our implementation we used $K_s = -50$ and $K_d = -10$. Prior to any transitions, the offset vector (and its time derivative) are initialized to zero. Intuitively, the offset vector is tied to the origin with a damped zero-length spring. When $O_t = \dot{O}_t = 0$, joint positions and velocities remain unchanged.

When we reach a transition point, we compute the discontinuity in joint positions (ΔP_t) and joint velocities ($\Delta \dot{P}_t$) and add the discontinuity to the offset vector: $O_t = O_t + \Delta P_t$, $\dot{O}_t = \dot{O}_t + \Delta \dot{P}_t$. Even though the joint positions before adding the offset signal contain a discontinuity, the displayed character does not. The discontinuity is absorbed by the offset vector (O_t) and decays smoothly to zero as the character follows the response motion.

3.2. Motion Deformation

A given recorded response motion corresponds to the particular push that produced it. We can make a motion look like it is responding to a push from a different direction by modifying joint positions of the original response motion. We do this by manipulating the offset vector (O_t).

When the character is pushed, we add an impulse vector to the derivative of the offset vector: $\dot{O}_t = \dot{O}_t + M$, that potentially modifies joint velocities. However, to generate visually plausible motions, M cannot be a random set of numbers. For example, if the entries in M corresponding to the

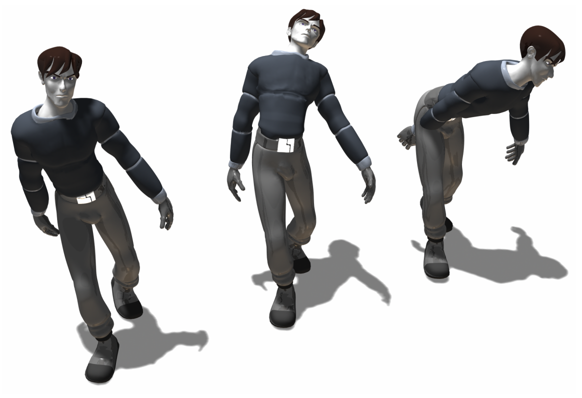


Figure 3: *Our deformation model modifies a recorded motion as a function of parameters (θ). Some of these parameters may result in good motions and some may produce bad motions. The character on the left is a frame from an original response motion where the actor was pushed from behind. The middle figure is the corresponding frame from a modified version of the same motion. The deformation parameters for the middle character produce a plausible motion, because the configuration of the character is reasonable. The character on the right is the corresponding frame that is modified with a different set of parameters. The configuration of the character is implausible, and hence the corresponding deformation parameters should be avoided.*

left shoulder and left elbow have drastically different values, these joints would move away from each other and the resulting inverse kinematics solution would exhibit unpleasant artifacts.

In practice, we only generate one impulse vector for one joint: the joint that is being pushed in the original response motion (the joint that was pushed in each recorded response was hand labelled). The components of M corresponding to other joints are then set to a fraction of this vector, depending on their distances to the joint being pushed. Nearby joints are given a larger fraction and distant joints are given a smaller fraction. For example, if the response motion was the actor being pushed on the left shoulder, we will introduce an additional velocity to the left shoulder. A smaller fraction of this velocity is added to neck and left elbow and an even smaller fraction is added to left hand. Formally, we write $M = [m_1\ m_2\ \cdots m_k]$, where m_i is the component of M corresponding to i^{th} joint, which is given as: $m_i = me^{-\omega d_i^2}$. In this equation d_i is the distance between the i^{th} joint and the position on the character being pushed, m is the impulse amount (situated at the joint being pushed) and ω is a parameter that governs how fast (as a function of distance) the impulse declines.

Once a motion is deformed using this procedure, we compute the velocity of the joint that was pushed. We collect the location being pushed and its velocity into a 6 dimensional *impulse* vector (3 dimensions for the location on the body being pushed and 3 for the velocity of this location). This vector is the handle that the user controls to deliver a push; the user indicates the position he/she wants to push and the desired velocity of this position. The synthesis algorithm should then find the response motion and deformation parameters that will make the joint being pushed move with this velocity (Section 4).

The deformation applied is governed by m (the change in the velocities at the joint that is pushed) and ω (how quickly we taper this extra velocity across the body). Let us call these parameters $\theta = [\omega\ m]$. This deformation procedure is not meant to be physically based, or to produce physically accurate results. We simply use this method to take an input response motion and modify it so that it can be used to serve a push from a direction. Not all choices of parameters produce a natural looking motion (see figure 3).

In practice, there is an envelope of non-zero parameter values θ, that yield realistic deformations and generate response motions that recover pushes from slightly different directions with different magnitudes. Section 5 will explain how we find such good parameter values.

4. Synthesis Algorithm

We create a *motion graph* that captures the possible transitions in the database of motions (including responses). We use an algorithm similar to [KGP02, AF02] to produce motion when the character is not being pushed. Care needs to be paid not to follow any edges that lead to a response motion during this traversal, because it would yield a motion of the character responding to a push without any user input. The method for traversing the motion graph is application dependent; we can make the character go to a particular spot or follow a path. In our implementation, we simply perform a random walk, so that the character idly moves about waiting for someone to push him.

When the character receives a push, we only have the current configuration of the body (C_t) and the *impulse* vector that is received from the user. We need to find a previously recorded recovery motion and the deformation parameters, such that when we synthesize a transition using the algorithm presented in Section 3, the resulting motion is visually plausible and responds to the user's push. We will denote the body configuration in the i^{th} recorded response motion just before the actor was pushed with C^i.

Let us assume that we have the following two functions:

$$\theta = F(C^i, impulse) \qquad (2)$$

$$Q = G(C_t, C^i, \theta) \qquad (3)$$

F takes a response motion (C^i) and an *impulse*, and estimates the deformation parameters (θ) that would be required to deform the response motion to meet the requested push. G takes the current configuration (C_t), response motion (C^i), deformation parameters (θ), and estimates the visual quality (Q) of the motion synthesized if we were to transition into the given response motion with the given deformation pa-

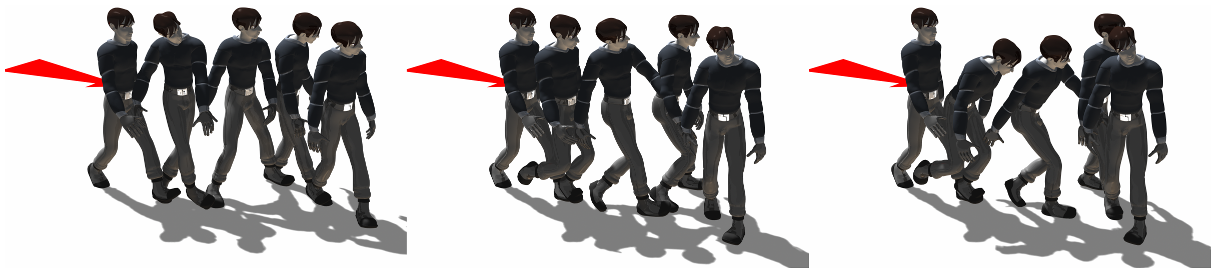


Figure 4: *The left figure is the motion synthesized using a baseline algorithm where we simply transition to the closest (in terms of configuration) recorded motion. The push direction and location is indicated with the red arrow. The closest push that the baseline algorithm can transition, is not pushed from the right direction, because the the character is pushed on the shoulder laterally. The middle figure shows the motion synthesized using our algorithm without any deformation applied. Since our oracle is not limited by numerical similarity between the configurations that we transition between, we can synthesize a recovery motion that is pushed from the correct direction. The motion on the right is synthesized using our algorithm with deformations. The oracle presented in Section 5 prevents deformations that create visual artifacts. The motion with deformations responds to the direction of the push better.*

rameters. We measure visual quality as a real valued number between 0 and 1, where $Q = 1$ means the motion is visually good and $Q = 0$ means a visually bad motion we would like to avoid. F and G together, form our oracle that estimates the visual quality of the motion before it is synthesized. We will explain how we create these functions in Section 5.

When a push arrives, we iterate over all response motions and evaluate F with the desired push to find the deformation parameters that would be required. We then evaluate the visual quality (using G) of the motion that would result if we were to transition to the response motion with the estimated deformation parameters. We find the response and deformation parameters that yield the best quality score (Q) and then generate a transition into that response (see figure 4).

Creating a transition into a response motion is equivalent to jumping from one location to another in the motion graph. After the transition, we return to synthesizing normal motions for the character (not being pushed), because the response motions join other motions in the motion graph.

An important advantage of having an oracle that estimates the quality of a motion before it is synthesized is that we can query it on any motion. For example, right after the character receives a push and transitions into a response motion, another push may arrive. We can handle such a repeated push the same way, by finding a response motion we can transition into. Notice that in the case of a repeated push, the current body configuration itself (C_t) may be a part of a previous response motion.

5. The Oracle

As described previously, to respond to an external push, we transition into a deformed version of a previously recorded response motion. This deformation is governed by the parameters θ, some settings of which may result in unrealistic motions. Moreover, we cannot expect to have a natural looking motion while transitioning from any body configuration to any recorded response motion. For example, our method would generate dynamically implausible motions if we were to transition from running to a motion of a standing person being pushed.

When a push arrives, we have two crucial questions to answer: First, which previously recorded response motions should we transition to, and second, what deformation parameters should we apply so that the resulting motion looks natural and responds to the push that the user applied.

We could try every recorded response motion and a large set of deformation parameters to see which pair generates the most visually plausible result. However, this is not feasible for a real-time system. Instead, we develop an oracle that decides which response motion and which deformation parameters we should use. We create this oracle by fitting a function to examples of good and bad motions synthesized using random transitions and random deformation parameters.

5.1. Training

The objective of the oracle is to differentiate between good and bad combinations of transition and deformation parameters. We create this oracle from an example set of already labelled good and bad transitions with deformation parameters. We approximate F and G by scattered data interpolation using a set of $< C_t, C^i, \theta, Q, impulse >$ examples. In particular, we used a nearest 10 neighbor interpolator. To generate these examples, we sample a random configuration in our set of motions (C_t), a random response motion (C^i) and random deformation parameters (θ). We then ask the user to evaluate the quality of the motion synthesized by transitioning to the selected response motion with selected deformation parameters. The user can attach any quality value between 0 and 1. This is not a binary value: the user can attach continuous values to indicate some synthesized motions are better than others. After synthesizing the motion, we compute the *impulse* automatically by looking at the velocity of the joint being pushed in the response motion.

Rather than generating many motions using these random transitions/parameters and asking user to attach a quality score, we train our oracle interactively. Before we synthesize a response motion, we first estimate its score using the already labelled examples. To do this, we first compute G by scattered data interpolation on the examples that have already been labelled. We then sample a random transition with random parameters and estimate its quality score using G. We display the synthesized motion and its estimated quality score to the user. This way, the user can see how good the oracle thinks the motion is and correct it if necessary. G is re-fitted every time the user provides a new example. We can also stop the training when the oracle has been trained on enough data so that the user mostly agrees on what the quality of the synthesized motion is. Intuitively, this oracle can be thought of as a user trained numerical similarity measure as given two motions, it estimates their visual similarity for transitioning between them.

Realistically, the dimensionality of configuration (C_t and C^i) can be high: if there are 30 joints in the body, (x,y,z) coordinates of the joint positions and their velocities would take 180 numbers. In practice, we reduce this dimensionality of the configurations to 5 using linear dimensionality reduction techniques (PCA). The dimensionality of θ is 4 (1 for ω and 3 for m). In our implementation, we trained this oracle for 3 hours of the user time with 1500 examples of good and bad push responses.

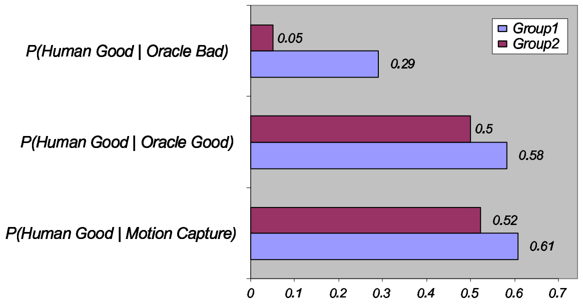
6. User Studies

Our synthesis algorithm creates motions that maximize the quality estimated by our oracle. Our algorithm, therefore, generates visually plausible motions only if the motions that our oracle thinks are visually plausible are actually visually plausible when judged by real people.

To verify that our oracle agrees with real people and that our synthesizer generates visually good motions, we performed a comparative study. Our study group consisted of 35 computer science undergraduate students (group 1), without any background on our research or motion synthesis technology. We also performed the same user study on 4 computer science graduate students (group 2) that are familiar with motion synthesis techniques, but are not involved in this research.

We first created a corpus of motions synthesized using our algorithm by using different random transitions and different random deformation parameters. For each motion, we estimated its quality using the oracle discussed in Section 5. We also included unmodified original response motions.

For 15 minutes, the subjects were displayed a random subset of these motions (rendered realistically with the same character skin as in the attached video) and were asked whether the motion looked like an actual recording of a hu-



Group1

	Oracle Good	*Oracle Bad*	*Mocap*
Human Good	807	227	240
Human Unsure	65	47	29
Human Bad	511	507	126

Group2

	Oracle Good	*Oracle Bad*	*Mocap*
Human Good	39	2	12
Human Unsure	0	0	0
Human Bad	39	38	11

Figure 5: *We performed user studies on 35 computer science undergraduate students (Group 1) and 4 graduate students (Group 2) with exposure to motion synthesis algorithms. The subjects were asked to identify if a displayed motion was an actual recording from a real person (*Human Good*) or not. We also queried our oracle and estimated the visual quality of the same motions. We labelled those motions that have an estimated quality greater than 0.5 as* Oracle Good *and the rest as* Oracle Bad. *We also displayed unmodified motion capture sequences (*Motion Capture*). The figure above contains some conditional probabilities that we computed from our studies. For example, the probability that a user would accept an actual motion capture sequence as an actual recording of a human ($P(Human\ Good|Motion\ Capture)$) is 0.52 for undergraduate students and 0.61 for graduate students. The table shows the number of motion sequences that users observed. For example, there are 807 sequences that were classified as realistic by our oracle and were also deemed realistic by users. Our user study indicates that the rate at which a human would accept a sequence that is evaluated by the oracle as good, is about the the rate at which a human would accept an original motion capture sequence as good.*

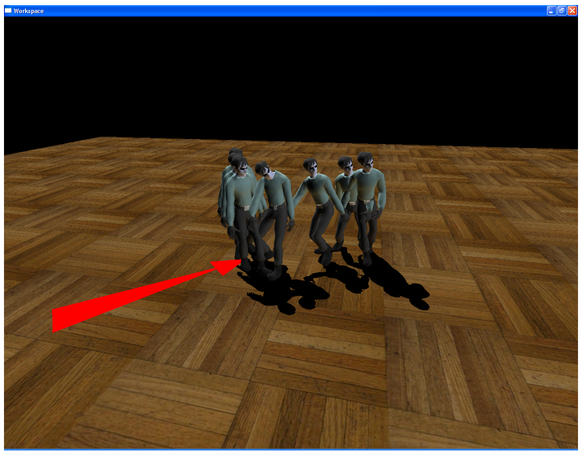


Figure 6: *This figure shows an illustration of our real-time interface. The user can push the character anytime from any direction (indicated as red arrows on the ground). See the attached video for a real-time demonstration of this interface.*

man motion ($Q > 0.5$), or an artificially synthesized motion ($Q \leq 0.5$). The users were also allowed (but discouraged) to say "unsure" if they could not make an assessment of the displayed motion.

Our findings indicate that our oracle mostly agrees with the user on good motions (both the oracle and users think the motion is good looking). However if the user believes the motion is non-human looking, our oracle disagrees almost half of the time (see figure 5). Our user study indicates that the rate at which a human would accept a sequence that is evaluated by the oracle as good, is about the the rate at which a human would accept an original motion capture sequence as good. Notice that people are not very good at recognizing captured motions (Mocap). Some of this is due to the artifacts in character modeling/skinning/lighting and some of it is due to inaccuracies in marker trajectories because of the forceful push that the actor experiences. We believe there are many factors that influence the perception of motion and understanding these factors is an important research area.

7. Results

Using our method, we can synthesize real-time motion of a synthetic character responding to pushes that may come at any time and from any direction. The attached video demonstrates our results using real-time screen capture where the user interacts with the character. The character is very reactive; the user can push the character from any direction. Our method generally generates good looking motions that do not suffer from visually disturbing kinematical and dynamical artifacts.

We have demonstrated our method on motions such as standing, walking and running. However, we think the method is applicable to other basic motions if there are examples of such motions and possible responses to external pushes while performing these motions in the database.

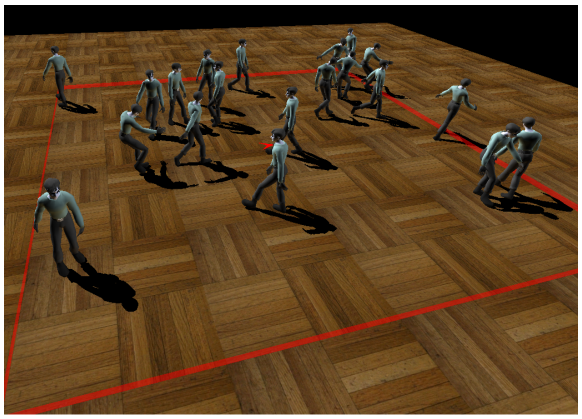


Figure 7: *Our method can be used to handle interactions in a virtual environment. This figure demonstrates 20 characters populating a space. Whenever characters bump into each other or hit the boundary of the space (indicated as the red square), they are pushed away. See the attached video for the animation.*

Our results demonstrate that our deformation model is able to enrich the input set of motions, so that a limited set of recorded response motions can be modified to serve pushes from different directions with different magnitudes. Our oracle can identify the envelope of parameters that yield visually plausible deformations as well as transitions.

The real-time screen capture in the attached video was recorded on a dual processor Athlon 2.2+GHz desktop computer. For this project, we captured 40 minutes of motion capture data. Within this data, we had the actor being pushed 200 times on various positions in the upper body, from various directions with various magnitudes. We also doubled the size of our dataset by mirroring these sequences.

An important component of the success of this method is the observation that when a real person gets pushed, he behaves passively for a very brief amount of time after the push. During this period, muscles do not yet take action for recovery and act similarly to springs. The central nervous control system then takes over and establishes balance. It is during the passive motion that we change the original response motions the most. Because the displacements to the joint positions that we introduce also behave like springs, the transition and deformation effects are not visually discernable, unless excessively large displacements are introduced (due to bad θ parameters or bad transitions).

Whenever the character receives a push, we estimate the quality of the best possible transition we can synthesize. If there are not enough response motions, the best quality that we estimate (using equation 3) may be bad. This means our algorithm can detect when it is performing poorly (i.e., synthesizing a bad looking human motion). If the estimated quality of the motion is poor, we can switch to another way of handling pushes such as switching to a ragdoll simulation or using the technique presented by [ZMCF05], however we have not yet investigated this alternative.

8. Acknowledgements

We thank the other members of the Berkeley Graphics Group and our reviewers for their helpful criticism and comments. This work was supported in part by ONR N00014-01-1-0890, California MICRO 03-067 and 04-066 and by generous support from Apple Computer, Sony Computer Entertainment America, Alias, the Hellman Family Fund, and the Alfred P. Sloan Foundation.

References

[AF02] ARIKAN O., FORSYTH D. A.: Interactive motion generation from examples. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 483–490.

[AFO03] ARIKAN O., FORSYTH D., O'BRIEN J.: Motion synthesis from annotations. *SIGGRAPH* (2003).

[ALP04] ABE Y., LIU C. K., POPOVIĆ Z.: Momentum-based parameterization of dynamic character motion. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), pp. 173–182.

[BCK78] BARCLAY C. D., CUTTING J. E., KOZLOWSKI L. T.: Temporal and spatial factors in gait perception that influence gender recognition. *Perception & Psychophysics 23*, 2 (1978), 145–152.

[BMH98] BROGAN D. C., METOYER R. A., HODGINS J. K.: Dynamically simulated characters in virtual environments. *IEEE Computer Graphics & Applications* (1998).

[CK77] CUTTING J. E., KOZLOWSKI L. T.: Recognizing friends by their walk: Gait perception without familiarity cues. *Bulletin of the Psychonomic Society 9*, 5 (1977), 353–356.

[CPK78] CUTTING J. E., PROFFITT D. R., KOZLOWSKI L. T.: A biomechanical invariant for gait perception. *Journal of Experimental Psychology: Human Perception and Performance 4*, 3 (1978), 357–372.

[FC89] FLANDERS M., CORDO P.: Kinesthetic and visual control of a bimanual task: specification of direction and amplitude. *J Neurosci 9* (1989), 447–453.

[FP03] FANG A. C., POLLARD N. S.: Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (July 2003), 417–426.

[FvdPT01a] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001* (2001), pp. 251–260.

[FvdPT01b] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics 25*, 6 (Dec. 2001), 933–953.

[GKM81] GEORGOPOULOS A., KALASKA J., MASSEY J.: Spatial trajectories and reaction times of aimed movements: Effects of practice, uncertainty and change in target location. *J. Neurophysiol. 46* (1981), 725–743.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. In *Proceedings of 2004 ACM Symposium on Interactive 3D Graphics* (2004), pp. 522–531.

[GT95] GRZESZCZUK R., TERZOPOULOS D.: Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 95* (Aug. 1995), pp. 63–70.

[HOT98] HODGINS J. K., O'BRIEN J. F., TUMBLIN J.: Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics 4*, 4 (Oct. 1998), 307–316.

[HP97] HODGINS J. K., POLLARD N. S.: Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH 1997* (1997), vol. 31, pp. 153–162.

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

[Joh73] JOHANSSON G.: Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics 14*, 2 (1973), 201–211.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 473–482.

[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. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 491–500.

[LP02] LIU C. K., POPOVIĆ Z.: Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (July 2002), 408–416.

[Mia95] MIALL R. C.: *Motor control, biological, and theoretical.* MIT Press, 1995, pp. 597–600.

[MTH00] MOLINA-TANCO L., HILTON A.: Realistic synthesis of novel human movements from a database of motion capture examples. In *Workshop on Human Motion (HUMO'00)* (2000), pp. 137–142.

[MWWS93] MIALL R., WEIR D., WOLPERT D., STEIN J.: Is the cerebellum a smith predictor? *J Mot Behav 25* (1993), 203–216.

[PW99] POPOVIĆ Z., WITKIN A. P.: Physically based motion transformation. In *Proceedings of SIGGRAPH 99* (Aug. 1999), Computer Graphics Proceedings, Annual Conference Series, pp. 11–20.

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *Proceedings of SIGGRAPH 1988* (1988), pp. 159–168.

[WP03] WU J.-C., POPOVIĆ Z.: Realistic modeling of bird flight animations. In *Proceedings of SIGGRAPH 2003* (2003), pp. 888–895.

[ZH99] ZORDAN V. B., HODGINS J. K.: Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Animation and Simulation '99* (Sept. 1999).

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH Symposium on Computer Animation* (July 2002), pp. 89–96.

[ZMCF05] ZORDAN V. B., MAJKOSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. In *To appear in SIGGRAPH 2005* (2005).