DRAFT: Interactive Dynamic Response for Games

Victor B. Zordan, Adriano Macchietto, Jose Medina, Maarc Soriano, Chun-Chih Wu*
Riverside Graphics Lab
University of California, Riverside

Abstract

Dynamic response is a technique for employing a physical reaction to an animated character. The technique utilizes a database of reactions as example motions to transition to following a dynamic simulation of an interaction. The search for the example to follow has been the stumbling block for bringing such a system into realtime applications and in this paper, we address that issue by proposing a number of speed-ups which make the approach faster than realtime and appropriate for an electronic game implementation. We accomplish this speed-up by using a supervised learning routine which trains offline on a large set of dynamic response examples and predicts online among the choices found in the database. Also, we propose a near-optimal routine which finds the alignment of the selected motion for the given scenario based on a sparse sampling with an additional speed-up over the original algorthim. With both of these changes in place, we enjoy a tremendous speed-up with inperceptable difference in the final motion compared to previous published results. Finally we offer a few additional alternatives that allow the user to choose between quality and speed based on their individual needs.

I.3.5,I.3.7Computer GraphicsPhysically based modeling, Animation

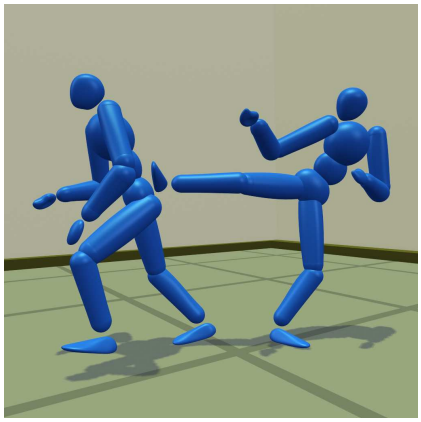

Figure 1: **The character on the left follows a physically based transition after an impact before returning to motion capture animation.**

1 Introduction

Physical simulation of articulated figures is becoming common in games and so-called *ragdoll* effects are the current benchmark for responsive characters. This paper proposes a fast, automatic method to generate more realistic response in characters following impact. In particular, we present a real-time algorithm which generates simulated physical contact and then returns to motion capture in a timely and seamless fashion to apply a more sustained reaction strategy. Our efforts draw from recent work in so-called dynamic response [Zordan et al. 2005] but are placed in making the algorithm applicable to computer games. The primary contribution of this paper is an interactive technique that combines a physical simulation which responds to the collision forces with an optimized search routine which quickly determines the best plausible 're-entry' from a reaction motion library following the impact. To accomplish our goal we utilize a classification method borrowed from machine learning which allows the system to determine the reaction based on a set of examples. Once training is performed offline, the classification can be performed online very quickly. Next, we align the example reaction with the given conditions and perform an optimized search to determine a good match for the timing of the transition to the motion sequence. These two speed-ups alone offer a tremendous speed-up over the previously published algorithm for dynamic response [Zordan et al. 2005].

The primary speed-up proposed in this research is drawn from the insight that people employ a finite number of reaction strategies when responding to an unexpected physical impact. As such, our intuition is to assess the "early warning" signs just following an impact that lead to a particular mode of response. Thus, rather than searching blindly for a good fit among a large set of examples, as done previously, we attempt to match a particular situation to the conditions seen based on experience and select from a small set of strategies. In our implementation, we produce a large number of examples to act as our training set using an exhaustive search – this becomes a pseudo-memory of experiences. We then apply a classification process over specific key attributes of these "experiences" to learn the conditions which lead to the strategy encapsulated in the motion example of each. In practice, we can define the strategy uniquely as the motion example itself and thus we do not need to label or identify the strategy in any rigorous capacity. However, we use this analogy to select a particular set of reactions which clearly depict different type of reaction strategies, thereby reducing the example set and avoiding unneccessary duplication. In the process, we drastically reduce the computation time to a prediction step of the classifier plus a narrow search to align the strategy example with the current condition in order to generate a smooth transition.

The importance of this work is reflected in modern software packages available, for example, from Havok and Natural Motion. While such commercial companies do not disclose their specific algorithms, in our approach, we set our sights on automating the generation of reaction strategies and exploring efficient search processes to allow automatically generated, sophisticated reactions within a real-time, online system. Given that systems, like Havoc's physics engine, can run many "ragdoll" character simulations simultaneously on today's machines, in this paper, we focus on optimizations which aim at speeding up the generation of a dynamic response and the related search for a satisfactory return-to clip from a motion library of reaction examples. To this end, we describe a method using a support vector machine (SVM) for classification and a number of practical optimizations that speed-up previously published work in dynamic response.

*email:vbz/macchiea/medinaj/sorianom/ccwu@cs.ucr.edu

2 Background

The goal of generating physically based responsive characters has been investigated by several groups of researchers such as [Oshita and Makinouchi 2001; Faloutsos et al. 2001; Yin et al. 2003; Komura et al. 2004; Arikan et al. 2005; Zordan et al. 2005]. Often, interaction takes into account a simulated reaction [Faloutsos et al. 2001; Zordan and Hodgins 2002; Yin et al. 2003; Mandel 2004] that gives the impression of responding *physically* to the impact. In addition, researchers have coupled reactions with transitions to new motion sequences following the impact in order to capture response behaviors [Zordan et al. 2005; Arikan et al. 2005; Komura et al. 2005]. Following these examples, we also include both a simulated reaction but introduce a novel routine for selecting the new *transition-to* behavior. In this paper, our emphasis is how to perform the search for this new behavior quickly. We contrast this work in responsive characters to several other approaches that employ physically based motion editing [Popović and Witkin 1999; Abe et al. 2004; Liu et al. 2005] in that our "edits" are meant to be computed online within a game for example, rather than modifying the motion offline. Under this consideration, many other techniques breakdown because they expect to be able to solve a space-time problem where lookahead is expected. In addition, the factor of speed is a crucial issue in the online setting.

Motion capture modifications that account for contact are highly desirable because they allow characters to react and interact with their environment, upholding and promoting consistency and richness in generated imagery. For online applications, techniques that compute character physics perpetually, such as [Faloutsos et al. 2001; Zordan and Hodgins 2002], are not as cost effective as those which compute the simulation only when required, such as [Shapiro et al. 2003; Mandel 2004; Zordan et al. 2005]. Our motivation for this choice is that we anticipate that motion capture alone will lead to better animation when impacts are not present, as well as a substantial computational savings. Shapiro et al. [Shapiro et al. 2003] transition from kinematic to dynamic models and back again based on a supervisory control scheme. Unlike their work, we give preference to the motion capture animation and use the simulation only as a vehicle for creating the reaction. Mandel also sets his focus on reaction after impact. However, he proposes specialized controllers for falling and "settling" to accomplish his transition back to mocap. In contrast, we treat the controlled simulation as a sophisticated interim system with emphasis and explicit trust placed on the mocap examples. As such, we use a minimal simulation interval, often less than a half a second long. In contrast to the dynamic response work presented by Zordan et al., a key factor in making such systems computational efficient is to streamline the search calculations. As such, in this paper, we present several novel contributions which show improvement in making the approach plausible for games today.

3 Overview

The overview of our system is outlined in Figure 2. In addition, here is a brief breakdown of our dynamic response algorithm adapted from Zordan et al. [2005].

- Initialized from mocap, interacting characters feel forces and react based on ragdoll-like forward simulation during an unexpected interaction.
- Next, an appropriate motion to follow the interaction is determined using the simulated motion based on a reaction library of motion capture examples.

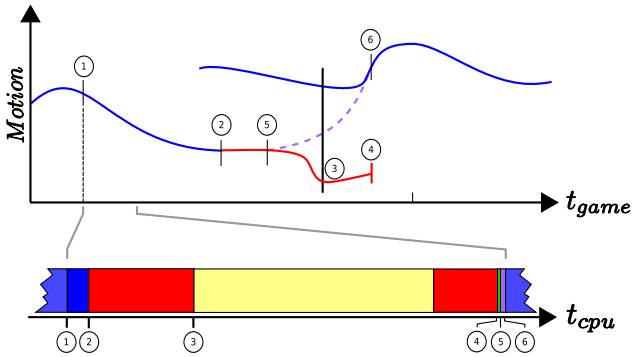

Figure 2: **Overview timing diagrams. The top *event* diagram details the various stages of a dynamic response from the game's timeclock, beginning (at Event 1) from the time of action invocation, say through a button press for an attack. Various stages are shown until the final blend is complete (Event 6.) The curves show the progression of motion from the previous motion capture (in blue on left) to the physical simulation (red) to the blend (purple) and finally back to the motion capture reaction (blue,right). Events occur chronologically based on the algorithm: Event 2 is the collision; Event 3 is the classification and search; Event 4 rewinds the system; Event 5 computes the final blend for the transition. The bottom diagram shows the stages of computation along with their relative CPU-time costs. The additional colors in the CPU diagram denote the combined cost of classifying the attributes of the new interaction and performing the optimized time-alignment (in yellow) and rewinding the system to begin the blend (green).**

- Finally, the calculated motion is smoothly blended into the target motion for the transition back to mocap.

For additional details, we refer readers to the mentioned paper. In contrast to the original algorithm, to afford our interest in making a real-time system, we opted against adding the second simulation pass described in the original algorithm. In addition, Zordan et al. point out that seventy percent of their computation time is lost to search. Our aim is to cut the search time substantially without sacrificing quality.

4 Optimizing search

As Mandel describes in his thesis [2004], search time can be controlled directly by choosing the number of reaction examples in the search database. Indeed, searching over a small set of reactions leads to faster running time but the hit in quality makes a small number of example reactions a poor choice. In contrast, if we utilize a large database of reactions and search over it exhaustively as Zordan and his colleagues propose it may produce high quality motion but at the price of interactvity. To compromise between these two techniques, we propose a twofold solution. First, we train a support vector machine to classify the choice of motion selection based on key features captured immediately after an interaction. And, second, we optimize the timing of the reaction with the given scenario by sampling to find a rough timing and fine tuning using a gradient search method as a final step.

4.1 Support Vector Machine

Before we can create a transition, we must find a motion capture sequence which closely matches some interval of the simulation trajectory following the impact. To do this, we employ a machine learning algorithm called a Support Vector Machine (SVM) to quickly classify (online) the physical motion just following an impact among the set of examples in our database. Many other researchers have worked on classifying human activities for various purposes (see [Int 2005]). In a recent example, Chai and Hodgin [2005] proposed an approach for a low-cost motion capture mechanism in which they classify human motion from a small number of control signals by employing low-resolution video cameras (webcams) and a small set of retro-reflective markers. Our classification problem is unique in that we select a motion capture example based on a synthetic, simulated human motion generated for a very short duration (only 100*msec* following initial impact.)

Among other applications SVM is a useful technique for data classification [Boser et al. 2002]. Intuitively, SVM operates by finding a partition in the space of input data. Specifically, SVM is used to fit functions which maximize the error margin between samples found in a training set. Faloutsos et al. [2001] employed SVMs in the training of preconditions for behavior controllers. We use SVM to select among a set of possible reaction *strategies* based on key characteristics sampled just following an interaction. SVM takes a set of training data to create a model which contains attribute information to predict the target class of testing data. The training data samples include both a class label and a set of features. To generate this data, we ran an exhaustive search (offline, and only once) and recorded the findings based on the desired feature vector.

Formally, let $T = (x_1, \alpha_1), ..., (x_l, \alpha_l)$ be a set of training observations where $x_i \in R_n$ are the samples of the attribute vector with n dimensions and α_i are the sample labels associated with each observation selected by the exhaustive search. The SVM is then created by solving the following optimization problem [Boser et al. 2002].

$$\begin{array}{rl} min_{w,b,\xi} & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} & y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i \ , \\ & \xi \geq 0. \end{array}$$

Training vectors x_i are mapped to a multi-dimensional space according to ϕ with separating partitions found with maximal margins by varying weights w and probability estimates b. Cost $C > 0$ is a weighting penalty for errors terms ξ_i. Further, $\phi(x_i)^T \phi(x_j)$ is defined to be kernel function $K(x_i, x_j)$ and for our purposes, we a use polynomial kernel $K(x_i, x_j) = (\gamma x_i^T x_j + t)^d, \gamma > 0$ with γ, t, d as user-defined kernel parameters. In our implementation, we employed the library LibSVM [Chang and Lin 2001] which includes a helpful instruction on using SVMs and useful defaults for several of the terms defined.

4.1.1 Feature vector

An important characteristic that contributes to the success of our work is the careful choice of the feature vector for the SVM. Because increasing the size of the feature vector requires a larger training set, keeping the length of the vector to a minimum is desirable. And, given that we want good classification, the feature vector cannot be too small or we will not be able to differentiate certain reactions (see Figure 3.)

Given the application in mind (determining the response after an interaction in a timely fashion,) it is obvious that we should include

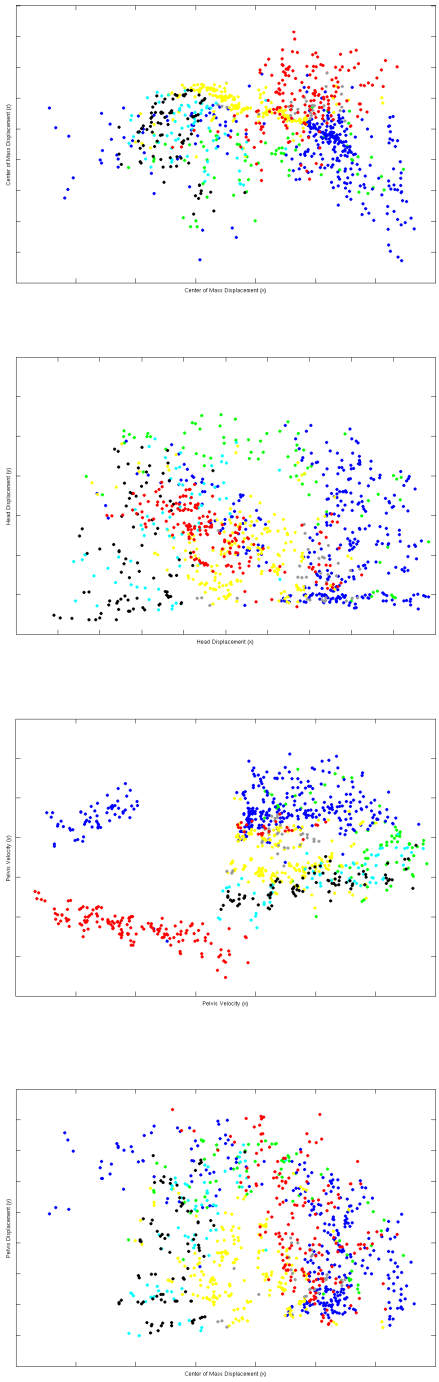

Figure 3: **Pairwise comparisons. Various scalar values taken from the simulation just after contact are compared with the colors representing unique reaction motions found by the exhaustive search routine. It is clear to see some structure in these plots but none alone are suitable for classification.**

information about the state of the character in the feature vector, but we must be selective. To this end, we collected a set of metrics from the character starting from the initial contact. In particular, we recorded the 6 degree-of-freedom state (with state derivative) for the bodies of the simulation along the spine (pelvis, stomach, chest, head) at tenth-of-a sec time slices from 0.0 to 0.3 *sec* following the collision. In addition, we recorded specific information that was valuable for balance: the state of the feet and the position and velocity of the center of mass (COM) for the same time slices. Next, we tested subsets of this data to find the feature vector that gave us the highest accuracy. Our hypothesis was that a narrow set of hand-selected features would do well given our observations of the pairwise comparisons in Figure 3 and thus we compiled several small sets of twenty or so features. Surprisingly, we found our highest accuracy by including all the recorded state information into a single vector. Interestingly, we also found by including successive time slices that the accuracy did not improve by including slices after 0.2 *sec*. But, the accuracy dropped significantly if we removed the time slice at 0.0 *sec*. More details about accuracies and specific findings appear in the results (Section 6.2.)

4.2 Optimized time alignment

The example selected from the database by the SVM classifier encapsulates the sustained reaction that the character will employ in response to the interaction. However, the timing which lines up this motion example with the simulated motion is still unknown. Recall, timing is not included in the classification step. To determine the time alignment, employing an exhaustive search will guarantee the best fit but is compute intensive. Thus, we offer an optimized routine.

We formulate the timing problem as selecting the frame indices from the simulated motion and the reaction motion capture clip that best match the two motion sequences according to the following error:

$$D_i = w_i \left(\sum_{b=1}^{n} w_{pb} \| p_b(f_{si}) - p_b(f_{mi}) \| + w_{\theta b} \| \theta_b(f_{si}) - \theta_b(f_{mi}) \| \right)$$

where w_i is the window weight over the i frames, f_s and f_m are aligned frames from the simulation and selected motion clips for each body b. The weights w_{pb} and $w_{\theta b}$ scale the linear and angular distances for each body. Importantly, an alignment step removes the global differences and facing direction between two frames for every tested window.

Optimally matching the timing is not trivial because the space is irregular with several local minima (see Figure 4.) In this graph we do, however, see an obvious bias toward the beginning of the simulation clip motion (shown at the front of the graphic, mostly in blue) and we exploit this bias in our fast timing search. To avoid a slow stochastic search, we take a sparse, uniform sampling of the timing space (we sample every ten frames along both axes) and then perform a gradient-based refinement search from the best of these. In the refinement step, we employ the BFGS algorithm from Numerical Recipes [Press et al. 1994]. In practice, in comparison to exhaustive search (applied only to for the time alignment problem,) we see tremendous speed-up over exhaustive search with slight, but unnoticeable differences in the final result. In addition, we bias the samples toward the beginning of the simulation track and cut the search if no improvement is found in subsequent sample sets.

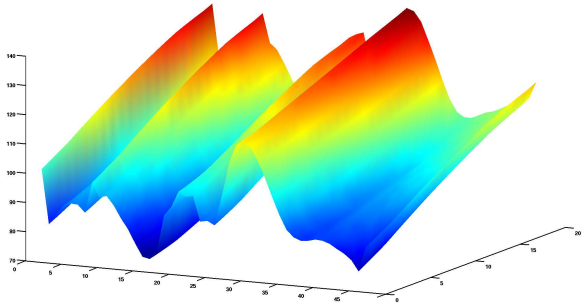

Figure 4: **Solution space for alignment. The vertical axis (and color) is the window error for time alignment found from comparing the reaction example and the simulated motion. The motion frames appear on the horizontal axes for the reaction example (front axis) and the simulation reaction sequence (into the paper). Note, the error largely increases with depth into the paper which suggests that earlier in the simulation motion example is usually better for matches in the time alignment problem.**

5 Return to motion capture

Once the timing for the selected return-to motion example is finalized, we are able to compute the remainder of the simulation and the final blend. Computing optimal transitions, for example based on physical principles, can produce pleasing results and indeed the question of how to compute such transitions is a research problem unto itself [Rose et al. 1996]. However, under our time-critical application, we opt for a straightforward interpolation to produce the transition. Our fixed-length, blended transition creates a smooth sweeping motion, in a computationally efficient manner, that brings the computed simulation following the impact to the same state as the selected motion.

5.1 Controlling the posture on the fly

One noted advantage to our approach is that the reaction target motion can be selected before computing the entire simulation. As soon as the timing is selected (Section 4.2), we know where the character is headed after the transition and can inform the simulation from that moment forward. That is, because we know ahead of time where the character should go, we can add this into the tracking controller for the duration of the simulation. In contrast, the algorithm presented by Zordan et al. [2005] requires a second simulation leg to incorporate this information. We incorporate the "lookahead" information in a very similar fashion, but in our case in the *first* simulation pass.

During the transition, a torque controller tracks a blended sequence of joint angles from the previous motion capture segment to the next, starting from the instance the time alignment step is complete (in the timeclock of the game.) Like the previous approach, the controller uses an inertia-scaled PD-servo at each joint. The desired sequence which is tracked by the controller is generated on the fly by blending. To find the desired postures, two frame samples, one from the previous motion capture example and the other from the return-to motion capture sequence, are interpolated using spherical linear interpolation (slerp) with an ease-in/ease-out (EIEO) weighting parameter that varies based on the time of the transition interval. Hand-selected springs and damping values are held fixed for the duration of the transition.

5.2 Creating a transition

Once the final simulated motion is generated, we perform a simple blend for the root position and orientation error as well as the joint angles across the transition sequence. For translation, linear interpolation is employed. For rotations, our system interpolates by slerping quaternions, again using a simple EIEO weighting in time across the transition. While this approach is extremely simple, it produces visually pleasing results in a timely fashion.

6 Implementation and Results

The character we chose includes 39 independent degrees of freedom (DOF), three each at the ankles, knees, hips, waist, back, shoulders, elbows, and neck as well as the six DOFs of the root. The dynamic simulation was generated using Open Dynamic Engine [Smith 2007] with ball and sockets for each joint of the articulation. Mass and inertial parameters are generated by ODE based on the geometrical models for the bodies shown in the figures and we use ODE's collision tools. Note, we exclude the timing of the physics in our analysis under the assumption that commercial physics engines are fast enough. Proof in modern games amply shows that this assumption is safe. Our implementation in ODE is a practical choice for us but not a recommendation for game developers. To align with the focus of our paper, speed up of the search routine is the emphasis of our timing approach.

6.1 Reaction Strategies

Our motion capture reaction library contains a variety of responses based on contact varying from light to heavy. While reactions to full contact strikes were omitted during the time of capture to avoid injury, strong pushes were performed that resulted in extreme responses without harm to the actor. In total, the library includes 110 reaction examples.

In our implementation, we experimented with the number of examples taken from this database for use in the classification. At one extreme, we could include all of the examples in order to maximize the suite of reactions - however, we would require a very large number of training observations to obtain reasonable accuracy for the SVM. To ensure a good fit, we keep the number of examples in the reaction database small instead. To select the reactions to include, we recall our idea of the small set of reaction strategies and hand select a desired set of examples. In particular, we include a single example for each of the following strategies for the animations included in this paper:

- Take a single step forward
- Take multiple steps forward
- Forward roll
- Side fall
- Step backwards
- Take multiple steps back
- Roll backwards

This set is not unique, for example we could also include a backward fall. Also, while the length of this list of actions is fairly comparable to the types of actions commonly found in a game today, recall that each entry in this list of strategy examples (as well as their left-right mirrors) is modified for timing and alignment with every computed interaction after impact forces are applied to create a purely physical interaction. In the results section, we further describe the benefits and costs related to the choice of the number of example reactions.

6.2 Results

To show our results we conducted a series of experiments to test the timing and robustness of our approach. In brief, we found the visual quality of our results were comparable to the original dynamic response algorithm but with a speed up of well over an order of magnitude.

To test our algorithm we initially trained the SVM on approximately 500 data examples. We computed these by "throwing" a heavy ball at the character while varying its angle, speed, starting height and target landmark on the character. We chose the ball because it was easy to vary in a procedural manner. We apply the training to a basic example of walking. This example was recorded separately from the reaction database. Using the reaction set described in the last subsection and an abridged attribute vector (with 20 terms) we were able to find an accuracy of 75% compared to the training set for new walking examples. If we modified the attribute vector for the SVM to include the larger set of features (See Sect. 4.1.1), we could raise that accuracy to 80% but suffer a 2x timing hit for classification. In an analysis per reaction strategy, certain responses especially those where the ball hit from the back, the SVM was 100% accurate at identifying the exact same strategy, but others were selected at lower accuracies (40%-60%). Upon inspection we found only a few examples appearing in our training set for the lowest-accuracy reactions. We then re-ran these same experiments on a more diverse training set with approximately 5500 data examples and found an accuracy of 86% and 94% for the small and large feature vectors, respectively. Several sample animations, including a comparison with the original exhaustive search, appear in the associated video. A filmstrip appears also in Figure 5 (top row).

We applied this classification SVM to new scenarios in attempt to assess the robustness of the algorithm (i.e. without retraining the SVM.) We replaced the character's motion of walking with a new animation of idle for fighting and utilize the SVM (trained on walking) to select the reaction strategy. Sample animations appear in the video and a filmstrip appears in Figure 5 (middle row). We found a comparable accuracy and quality to the exhaustive search for a small random set of animations from our training set. To assess more thoroughly, we computed a new SVM based on new training data computed specifically for the idle motion clip and found that both SVMs were equivalent in their accuracy for the high accuracy reaction strategies but an improved accuracy for some of the previously low reactions was observed. Next, we replaced the ball with a human opponent and used the "walking" SVM to compute new responses. We could not easily create an SVM for the character opponent because the training data was not easy to generate automatically, however, for the animation test cases we looked at the SVM for walking again gave comparable accuracy and produced quality motion (as in Figure 5, bottom row and the video.)

We feel it is important to note that the SVM's accuracy is not a direct measure of the quality of the final motion. While we quote accuracy percentages because it is a common metric in the assessment of SVMs, in our application, the classification of the SVM can be "wrong" (because it does not match the original system) but still produce a suitable animation. For example, we found "errors" such as choosing to step once rather than taking multiple steps. Also, with more carefully placed training examples, we found that we could continue to improve the accuracy of the SVM, however, we

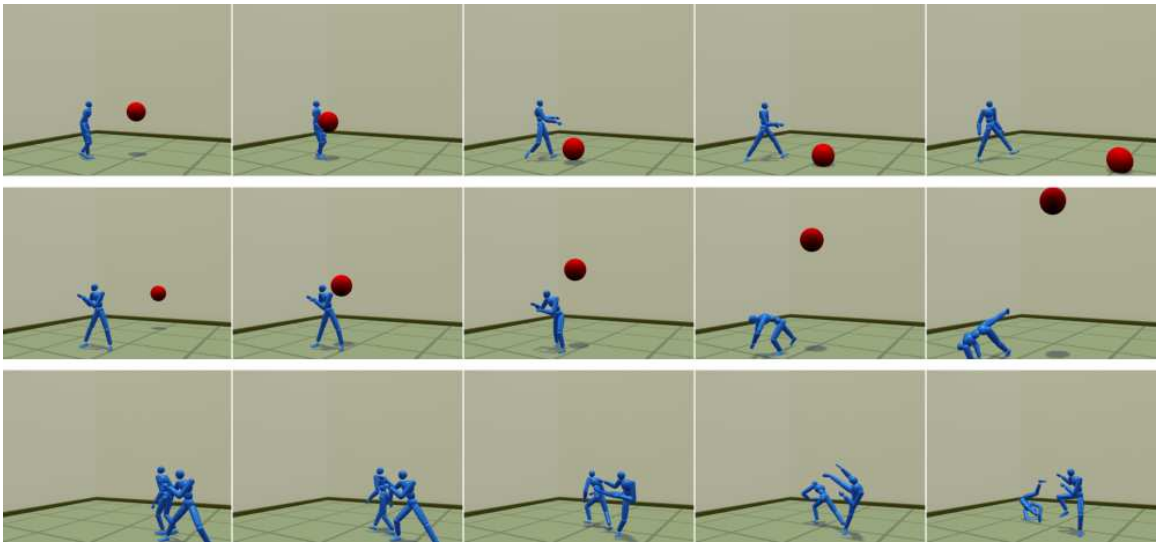

Figure 5: **Animation filmstrips. Three examples of dynamic response from our system (view left to right).**

reaction	original (*sec*)	optimized (*sec*)
step forward	1.82	0.08
step backward	1.96	0.06
forward roll 1	1.74	0.07
forward roll 2	1.95	0.07

Table 1: Timing comparisons. **Time cost of CPU is shown between the unoptimized and optimized versions for three reaction strategies with the last being shown for two example motions. The figures shown are composite timings for the reaction selection and timing alignment steps. All timing tests were run on an AMD Athlon X2 5000+ processor.**

felt the results coming out of the SVM were already reasonable with the data set of 500.

Another important measure, next to accuracy/quality, is computation time. Table 1 summarizes our timing speed-up to the original algorithm. Notably, after training, the prediction using SVM classification was a negligible part of the CPU cost (1-2 *msec*.) and varied based on the size of the attribute vector. This resulted in the largest part of our time savings. For the time alignment component, the original CPU-cost was between 0.21 to 0.37 *sec*, of this approximately 0.04 *sec* was a fixed cost associated with orienting and positioning the test motion repeatedly before testing each time-alignment fit which carried over in the optimized timing alignment presented. In aggregate, we see that the optimizations presented gave an approximately 20x speed-up on average. Given that the game timeclock advances about one second, our timing statistics show that we have moved from slower-than real-time to a substantially faster than real-time with the improvements we have introduced, making the approach much more practical for games.

7 Conclusions

In this paper we describe an approach that computes dynamic response to unanticipated interactions *faster than real-time* for game and other online applications. To accomplish this goal, we introduce a two machine learning methods that speed up 1) the selection of an appropriate motion capture clip from a set and 2) the optimal alignment of that motion, both based on the conditions before and just following the specific interaction. Based on our findings, our approach is scalable in the size of the reaction set given that the selection process using SVM does not lead the calculation time, based on our findings. Instead, we found the upper bound on the number of reactions is derived from the need for an exorbitant number of observations to train the SVM. (This may be handled with multi-resolution clustering methods.) However, for our implementation, we experimented with a much smaller set of example reactions, analogous to a single example each for a hand-picked set of strategies, and we feel we found a good compromise between the number of training examples needed and the accuracy of the SVM. One alternative for future work might be to group examples into strategies, thereby keeping the classification size small while including a more diverse set of reactions. Finally, we show that our system is capable of generalizing to new situations - the SVM we trained for a ball hitting a walking character was used to generate motion for two characters interacting (Figure 1).

We conclude with a couple of remarks related to the "dynamic response" approach in general based on our experience with this project. First, we believe that dynamic response is best suited for situations with large disturbances and that other techniques are better for small disturbances [Zordan and Hodgins 2002; Yin et al. 2003; Komura et al. 2004]. In our experimentation, the small disturbance reactions lead to less pleasing motion particularly for parts of the body not associated with the interaction (e.g. the feet slide too much.) Second, we skipped the second simulation pass described in the original algorithm in lieu of a faster algorithm. Given the timing speed-up we saw, we could have included it, but felt it complicated our story, especially since it seems to make minor perceptual improvements. However, in general, we believe the choice to include it or not should be made on a case-by-case basis depending on the sensitivity of the audience and the computation time available.

There are continuing trends toward more physically simulated characters and the game industry is already moving beyond 'rag-doll'-type effects for characters. Through real-time techniques like the one described here, physics-based characters can become more useful and believable and through such advances, we hope to continue to improve the responsiveness of game characters and the immersiveness of games in general.

References

ABE, Y., LIU, C. K., AND POPOVIĆ, Z. 2004. Momentum-based parameterization of dynamic character motion. In *2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 173–182.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Pushing people around. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 59–66.

BOSER, B., GUYON, I., AND VAPNIK, V. 2002. A training algorithm for optimal margin classifiers. In *In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144-152. ACM Press.*

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

CHANG, C.-C., AND LIN, C.-J. 2001. *LIBSVM: a library for support vector machines.* Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, 251–260.

INTERNATIONAL WORKSHOP ON HUMAN ACTIVITY RECOGNITION AND MODELING. 2005. Segmentation and Classification of Human Activities, HAREM.

KOMURA, T., LEUNG, H., AND KUFFNER, J. 2004. Animating reactive motions for biped locomotion. In *Proc. ACM Symp. on Virtual Reality Software and Technology (VRST '04)*.

KOMURA, T., HO, E. S., AND LAU, R. W. 2005. Animating reactive motion using momentum-based inverse kinematics. *Compute Animation and Virtual Worlds 1*, 16, 213–223.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (Aug.), 1071–1081.

MANDEL, M., 2004. Versatile and interactive virtual humans: Hybrid use of data-driven and dynamics-based motion synthesis. *Master's Thesis, Carnegie Mellon University*.

OSHITA, M., AND MAKINOUCHI, A. 2001. A dynamic motion control technique for human-like articulated figures. *Computer Graphics Forum (Eurographics 2001) 20*, 3, 192–202.

PLAYTER, R. 2000. Physics-based simulation of running using motion capture. In *Course notes for SIGGRAPH 2000*.

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

PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 1994. *Numerical Recipes in C*. Cambridge University Press, New York.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of ACM SIGGRAPH 1996*, 147–154.

SHAPIRO, A., PIGHIN, F., AND FALOUTSOS, P. 2003. Hybrid control for interactive character animation. In *Pacific Graphics 2003*, 455–461.

SMITH, R., 2007. Open dynamics engine. www.ode.org.

YIN, K., CLINE, M. B., AND PAI, D. K. 2003. Motion perturbation based on simple neuromotor control models. In *Pacific Graphics*.

ZORDAN, V. B., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 89–96.

ZORDAN, V. B., MAJKOWSKA, A., CHIU, B., AND FAST, M. 2005. Dynamic response for motion capture animation. *ACM Trans. Graph. 24*, 3, 697–701.