Adding physical like reaction effects to skeleton-based animations using controllable pendulums

Ahmad Abdul Karim[1,2], Thibaut Gaudin[2], Alexandre Meyer[1], Axel Buendia[2,3], and Saida Bouakaz[1]

[1] Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
[2] Spir.Ops Artificial Intelligence, Paris, France
[3] CNAM − CEDRIC, 292, rue St Martin, 75003 Paris, France

Abstract. We propose a system capable in real time of adding controllable and plausible oscillating physical like reaction effects in response to external forces (perturbations). These oscillating effects may be used to modify a motion or to customize it in a cartoon like way. The core of our system is based on several connected 3D pendulums with a propagating reaction. Our pendulums always return to a preferred direction that can be fixed in advance or can be modified during the motion by external predefined data (like keyframe). Our pendulums are fully controllable concerning reaction time and damping, leading to a fully deterministic system. And they are easy to implement even without any previous knowledge about physics equations. Our system is applicable on articulated body with predefined motion data (manually set or captured) or procedural animation, and is even capable of simulating cloth.

1 Introduction

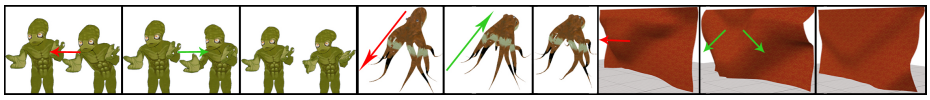


Fig. 1: Different implementations of our system. Red arrows: external perturbation. Green arrows: our system immediate response.

There is a big focus in computer graphics to modify/edit any articulated body animation (traditionally based on predefined motion data like keyframe or motion capture), and adapt it to match any environment or character. An articulated body has a large number of degrees of freedom, many constraints of length or angles, as well as many possible self-collisions during animation. Thus, the physics that govern its movement is computationally expensive, numerically

imprecise, and often difficult to predict and to control. A recent survey by Welbergen *et al.* [1] gives a good overview of the different methods and paradigms used and most importantly the trade offs between animation control and motion naturalness. In this context, there is a crucial demand for real-time methods providing physically plausible, but controllable effects [2].

We propose an original system, to our knowledge, that adds physical like reaction effects to any skeleton-based object, in real-time with a full user control using our 3D pendulums. The effects we seek to obtain are based on damped oscillatory motions that propagate through an articulated chain. The effect may be visually plausible like a rope moving in the wind, or a body reacting to external forces, and may also be more cartoon like which is shown in the accompanied video by giving a dancing like effect. In our system each bone of the articulated body is animated by a 3D pendulum. The pendulum is guided by a spring-damper that pulls it toward a customizable direction. Our approach has two objectives. Firstly, we ensure body length constraints between two joints working only on the angle between bodies. Secondly, we make a predictable real-time system in which we can control the reaction time to reach a user-defined direction and also the regime (critical or underdamped) of the oscillations around this direction. Our pendulums have three degrees of control: reaction time, damping and target direction. This concept of 3D pendulums may be applicable in a multitude of scenarii with some of them illustrated in Figure 1. We must emphasize that our system is better suited for acyclic bodies and that we do not address the balancing problem in the case of the biped example. Our system is really easy to implement, as we will see in section 3, with no need for a full physics simulation, nor any kind of complex calculations (like the inertia matrix). The mesh of the 3D model is animated with the classical linear blend skinning technique [3].

2 Related Work

Our pendulums oscillate visually like real 3D pendulums by computing their movements with a mass-spring approach. Mass-spring system is one of the simplest approach to simulate physically-based animation [4] as they offer an intuitive and flexible way to model mechanical systems. Furthermore, they can be used in derived fashion as to animate large crowds of humans and robots in a believable way in Pixar's movie WALL-E [5].

Editing motion of an articulated body (producing or modifying an existing animation) is an important topic in computer animation. For instance, some methods aim at warping the time of an existing motion [6], or combining/blending existing motions often represented in an animation graph [7], others propose the use of signal processing tools to modify an animation [8].

In parallel of these approaches, an important aspect is to add physical reactions to animations, like blending an existing motion with a physical response [9]. Zordan *et al.* in [10, 11] use a Proportional-Derivative (PD) controller to drive the dynamic body motion toward a re-entry in motion capture data. Our method may be related to a PD controller, and also to the MRAC controller

that uses the Adaptive Control proposed by [12] and used in [13]. All of these controllers are like our pendulum bringing a bone to a preferred direction but we differ in several essential ways: the other methods always try to return to a preferred direction (or position) even if there is no external perturbation. That introduces a delay in the produced animation between the target pose (keyframe or motion capture data) and the response of the PD controller like in [14]. On the other hand, our system is a superposed layer over the motion data and only reacts when there is an external perturbation. Our system plays exactly the motion data with no delay, and only adds the physical like reaction effects when needed. Another difference is the controllability; our system is designed to be temporally controlled (control over the reaction time). Temporal control in the case of the PD controller is hard to achieve and demands some hand tuning of the gain constants. In [15] they show the ability to temporally control PD controllers (using adaptive calculation of the gain constants), but it involves some heavy calculations of the inertia matrix of each joint on each keyframe, with specific calculations in the case of an external perturbation (calculating the re-entry key frame). Finally, all the mentioned controllers are always critically damped. On the other hand, our pendulums can be critically damped or underdamped while maintaining the temporal control. MRAC controller is designed to be temporally controlled but it calculates and adapts its gain constants on each frame using forces and torques calculations, while our pendulums do not need any adaptive pass once the user sets the reaction time and the damping, plus the ability to modify them in real time.

We differ from other systems of skeleton-driven deformations like [16] in that we concentrate only on deforming the skeleton of the articulated body without any specific treatment to the mesh. Our mesh is animated by the classical linear blend skinning [3]. We illustrate anecdotally that a rigid tissue may be represented by a tree of bones animated by our approach with a simple, large time-step, controllable computation. Our goal is not to simulate realistic fabric like in recent techniques [17].

3 3D pendulums

In our system, a bone of an articulated body is animated as a pendulum with a configurable preferred direction as illustrated in Figure 2(a). A pendulum is an anchored bar, with a fixed length L, attracted to its target direction by a spring, which pulls the pendulum toward this direction as described in Section 3.1. This idea allows to easily add plausible oscillations to any animation with a temporal control, as explained in Section 3.2. In Section 3.3 and 3.4 we present our linear algorithm that deals with a tree of pendulums or a skeleton, by propagating the motion of a single pendulum to its father and sons.

3.1 3D pendulum principle

We design a pendulum $\overrightarrow{V}$ as a rotating bar attracted to its preferred direction by two springs: one spring on each 2D plane XY and ZY as illustrated

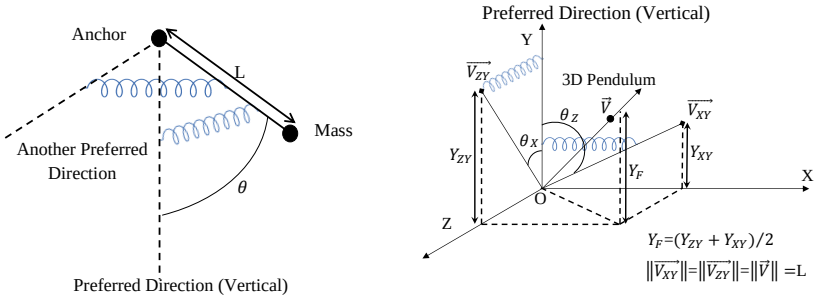


Fig. 2: (a) 3D pendulum (b) A 3D pendulum composed of two 2D pendulums with Y as their preferred direction

in Figure 2(b). We choose this scheme with two springs instead of one spring to avoid spiral rotation motion around the target direction. The computation of pendulum $\overrightarrow{V}$ motion is done using its projections $\overrightarrow{V_{XY}}$,$\overrightarrow{V_{ZY}}$ independently. During the motion, after calculating the two new spring positions in 2D $\overrightarrow{V_{XY}}$ and $\overrightarrow{V_{ZY}}$, the 3D position $\overrightarrow{V}$ is obtained by combining them and ensuring that $\|\overrightarrow{V}\| = \|\overrightarrow{V_{XY}}\| = \|\overrightarrow{V_{ZY}}\| = L$. In the current implementation we omit the twist component around the axe of the 3D pendulum $\overrightarrow{V}$ which is the third degree of freedom, we plan to add it in a future work. It is interesting to notice that by using the target direction $\overrightarrow{-Y}$, we can give the impression of gravity that always pulls the bodies toward the ground.

3.2 Time Based control for spring dampers

Let m be a mass connected to a spring with stiffness constant k. This mass oscillates around a rest position x_0 with a viscous damper that has a damping coefficient c. Based on Newton second law of physics the acceleration is $\ddot{x} = -(k(x-x_0)+c\dot{x})/m$ where x is the current position of the mass, $\dot{x}$ its velocity. We integrate this motion using the Verlet scheme [4] which was numerically stable during our experiment described in Section 4. Giving a random position x to the mass, it oscillates around the rest value x_0, seeking to minimize the error $(x - x_0)$ until reaching zero. This oscillation depends directly on the constants (k, c, m). In order to timely control the movement of this spring damper we use the *Settling Time* T_s principle. It is the time required for the mass position x to reach its max amplitude inside a given error interval (See Figure 3(a)) and remains inside it. This interval is symmetrical around x_0.

$$T_s = -\frac{ln(tolerance fraction)}{\zeta * w_0} \tag{1}$$

Where the *tolerance fraction* is the needed error interval shown in Figure 3(a), w_0 is the natural frequency and ζ is the damping of the ordinary differential

equation governing a damped harmonic oscillator:

$$m\ddot{x} + c\dot{x} + k(x - x_0) = 0$$

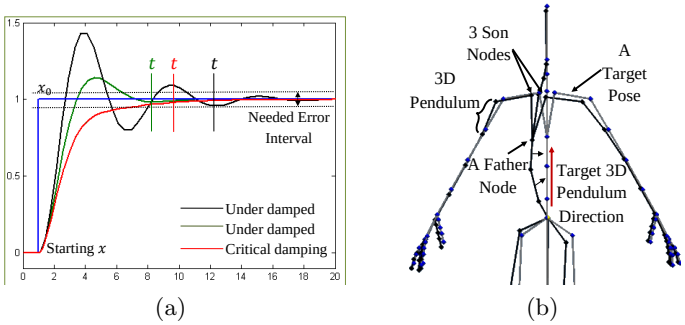
or

$$\ddot{x} + 2 * \zeta * w_0 * \dot{x} + w_0^2 * (x - x_0) = 0$$

with

$$\zeta = \frac{c}{2mw_0}, w_0 = \sqrt{\frac{k}{m}} \tag{2}$$

By fixing the tolerance fraction to 5% in equation (1) and by using the user provided settling time and damping (critically damped or underdamped), the spring damper constants k and c are calculated from equation (2), achieving total control over the curve of the spring damper while maintaining its dynamic aspect.



(a) (b)

Fig. 3: (a) Spring oscillation under different damping (b) 3D pendulums Tree in Black, Skeleton Vertical Target Pose

Figure 3(a) illustrates springs oscillating under different damping values. They oscillates around their x_0 until full stop, with their respective settling time. The third spring damper is a critical spring damper which converges toward x_0 faster than the others, and without oscillation.

3.3 Tree of 3D pendulums

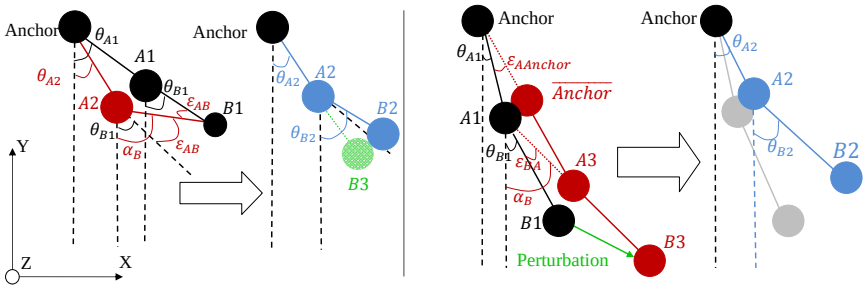
The skeleton of an articulated body is a tree of connected joints (articulations). By connecting several 3D pendulums and by defining the target direction for each one of them, the final result is a tree of pendulums that maps the articulated structure, as shown in Figure 3(b). Some of the 3D pendulums act as father node for several others. When they move, the anchor points of their children moves. In order to have a visually believable reaction, these 3D pendulums need to interact with each other. We define two strategies used in conjunction to achieve this goal:

Father Pursuit strategy and Son Pursuit strategy. In the accompanying video we show the similarities between the motion of a chain of pendulums fully-physically simulated and our chain of 3D pendulums that incorporate these strategies. For simplicity, these strategies will be described in a 2D plane.

3.3.1 Father Pursuit Strategy. The objective of this strategy is to propagate the motion of the father 3D pendulum toward its children, thus they need to incorporate this movement in their own motion. Figure 4(a) illustrates two connected pendulums P_A,P_B, A,B are the positions of each mass, L_A,L_B are the lengths of the bars, and θ_A,θ_B are the errors that each pendulum seeks to minimize. In this example the preferred direction of the pendulums are identical (the dashed -$\overrightarrow{Y}$).

The update system is a top-down system scheme, starting from the anchor toward the leaf. First, on time t_1 (in black) the error that we try to minimize is θ_{A1} in P_A and θ_{B1} in P_B. Now, on time t_2 (in red):

1. P_A moves, its spring damper tries to minimize the error, and has a new position $A2$.
2. P_B: the angle ε_{AB} between the two vectors $\overrightarrow{B1A1}$ and $\overrightarrow{B1A2}$ is added to its own error, $\alpha_B = \theta_{B1} + \varepsilon_{AB}$.
3. P_B: letting the spring damper integrate its equations, we obtain a new angle value θ_{B2} which contains the new pursuit error.
4. P_B: based on L_B the new position $B2$ (in blue) is calculated.



(a) (b)

Fig. 4: (a) Father Pursuit Strategy (b) Son Pursuit strategy

Without this process, the new position of P_B would have been $B3$ (in green), which is not correct and would have produced a non-logical disconnected motion. This Father Pursuit process is extended to every pendulum in the chain. A third pendulum P_C follows the motion of its father P_B, and so on. By extending this process in 3D we have a totally plausible physical chain of 3D pendulums (as seen in the accompanying video). Each one reacting to its father movement while oscillating around its target direction.

3.3.2 Son Pursuit Strategy. The objective of this strategy is to reflect the perturbation that can occur on the son level, to reflect it on its father. It occurs when the mass of P_B takes a perturbation as seen in Figure 4(b) (in green). The perturbation is regarded as a change in the position, as if we only take the final position resulted of an impulse applied to a rigid body.

1. The perturbation induces its full impact as if the mass P_B was not attached (in red).
2. P_B's mass has two positions: $B1$ (the old one) and $B3$ (the new one). Inverting the previously detailed computation of the father induced error ε_{AB}, we calculate the child error ε_{BA}, the angle between $\overrightarrow{A1B3}$ and $\overrightarrow{A1B1}$ and adding it to θ_{B1}, we obtain $\alpha_B = \theta_{B1} + \varepsilon_{BA}$.
3. The mass of P_A should follow, as it is being pulled by its son now. The new position $A3$ is calculated easily by choosing on the line $A1B3$ the point $A3$ where $\|A1B1\| = \|A3B3\|$.
4. This process propagates toward the anchor.
5. The new positions are recalculated based on the fixed anchor position.

With this scheme, all the errors that the spring dampers need to minimize because of a perturbation are calculated in a bottom-up way starting from the son that took the perturbation toward the anchor.

3.3.3 Final workflow. In a tree of pendulums, calculation cycles may occur when two nodes are influencing each others in an endless loop (father influencing its son, then the son influencing its father, and so on.). To avoid this kind of loops, our update system is inspired of Featherstone's divide and conquer algorithm [18, 19] which eliminates these calculation cycle problems and breaks computation into two main linear passes: a bottom-up then a top-down pass through the articulated body tree. We adopt this paradigm, only the calculations differ as follow:

1. For each 3D pendulum perturbed in the tree: resolve this perturbation by applying it on its mass then calculate the errors ε in a bottom-up iteration toward its ancestors according to the Son Pursuit strategy.
2. For each father 3D pendulum integrate all the children errors (ε_1,ε_2 *etc.*) to its own error θ.
3. Start the standard top-down pass starting from the anchor toward the leaf according to the Father Pursuit strategy.

In the previous step 2, there are many ways to calculate the integration:

- Summing up all the perturbation errors coming from its children: it is the method used to produce all of our results. It is the simplest method, and the one we chose after testing.
- Calculating an average: this method will perturb the father node in the same direction of the previous method, but with less amplitude. It can be useful when the application decides that the father node should be less affected by its children.

– Doing a weighted average based on:
  - The mass: the heavier son has more influence on its father.
  - The importance of each branch: by assigning predefined priorities on the children.

We can imagine many other possibilities based on specific application needs. Our system is quite easy to implement and the actual calculations in each strategy require only basic knowledge in 3D vector math. None knowledge in physics systems is required; we do not compute the inertia matrix nor we use the notion of force. In the same time we can use physics principles to enhance the end result like in the case of the father pendulum integrating its children errors based on the inertia matrix.

3.4 3D model Skeleton Vs. 3D pendulums tree

In the following section 4 we use a skinned 3D models, and we construct the pendulums tree based on a predefined skeleton. Starting from the bind pose (rest pose) of the skeleton, we create a 3D pendulum for each skeletal connection (bone) with the same length and with its rest preferred direction calculated from the bone rest pose orientation. By maintaining the parenting hierarchy of the base skeleton, we have a pendulums tree that maps this skeleton perfectly. While playing a motion data, we will modify the target direction of each corresponding pendulum thus reproducing exactly the motion data. If the 3D pendulums start to react to an external perturbation, each of the 3D pendulums orientation and position is applied to the corresponding bone.

4 Applications and Results

In this section, we present several ways to use the 3D pendulums tree: add physical effects to lifeless models like an octopus, modifying pre-defined motion data with physics reactions, and anecdotally a cloth simulation (which is normally a closed-loop problem). In all cases, the pendulums reaction is totally controllable: reaction time, damping and target direction. The results were computed on an Intel Core 2 Duo 2GHz, 2 GB RAM, with and ATI X1400, 256 MB. Our experimentation does not manage collisions, but we can easily imagine a system that creates an impulse (change in the position) on each 3D pendulum in reaction to a contact.

4.1 Adding physical reaction effects to any skeleton-based bodies

On Figure 5, we use our system on a lifeless octopus model. By adding some simple procedural animation to its tentacles (pulling only the root node of each tentacle toward the center at random intervals) the rest of the model reacts in a passive way, modifying the animation and adding plausible physics effects. The octopus model consists of 150 joints and the computation time of our superposed physical effects is only 0.3 ms, which is negligible in today's standards.

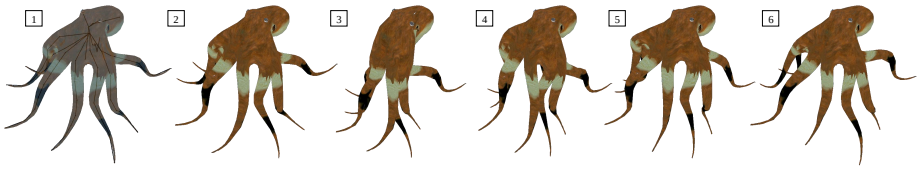


Fig. 5: From top left: [1] the model with its 3D pendulums, [2] rest pose, [3] we pull all the tentacles toward the center, [4] reacting, [5] tentacles overshooting (underdamped regime), [6] return to rest pose

We can also use our system on animated models. In that case on each frame, the motion data takes control of the skeleton changing the preferred direction of each pendulums. With no external perturbations, the 3D pendulums rigorously follow the animation data, while when an external perturbation occurs, our system reacts to this perturbation while trying to return to the needed target pose. With such a technique, our system adds a plausible physical reaction effects over a predefined animation data, as a superposed layer of animation. These reactions can furthermore be customized by making a section of the body more rigid, more flexible, changing the reaction time, or tuning the damping. This gives the end user a powerful tool to modulate the reaction of the body in a very intuitive and easy way.

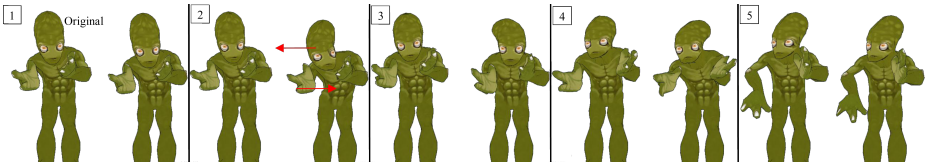
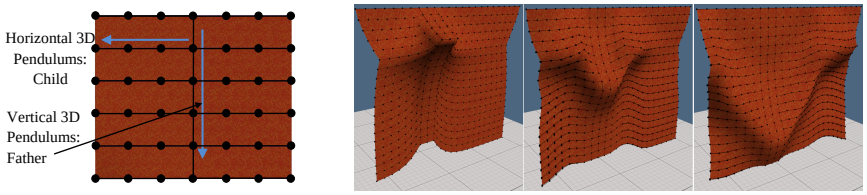


Fig. 6: From top Left: [1] Original (on the left) and our simulated articulated body (on the right), [2] Two perturbations, [3] to [5] Reaction and returning to Original keyfarme

By playing only the animation data on our test machine, for the previous model in Figure 6 with 92 joints, the average computation time for each frame is 0.06 ms. When playing the same animation using our pendulums and two perturbations, the computation time rises to an average of 0.46 ms, which stays negligible. This overcost consists of reading the motion capture data in order to change the pendulums target direction, integrating the perturbation and doing the main integration (based on the explained workflow).

4.2 Cloth Simulation

Although cloth is a closed loop problem, we are capable of giving the impression of an animated cloth by simply creating several vertical 3D pendulums that cover



(a) (b)

Fig. 7: (a) A cloth represented as a tree of pendulums (b) Cloth being pulled in the middle with a visual representation of the 3D pendulums

the cloth, plus attaching several horizontal 3D pendulums to each vertical one (one vertical is shown in Figure 7(a)). By doing a weighted average between the positions of all horizontal pendulums activated by their vertical father, weighted based on the distance between each horizontal pendulum and its vertical father, we compute the final cloth position. We have a fully reactive cloth without any tearing problems, and the cloth dimensions are always ensured vertically and horizontally (a constraint that we always ensure in the calculations), while giving total control over the reaction time. We are not aiming to compete against more general, visually and physically accurate cloth simulators that are better suited to simulate actual human cloth for example. We are just proposing a less sophisticated, but stable and fast enough method that can gives the effect of a reactive cloth. In Figure 7(b), only the three middle vertical 3D pendulums with their horizontal children's are active and being simulated, because they are the ones that have been perturbed, thus optimizing the calculation time. It is constructed using about 1500 3D pendulums. The average calculation time of these pendulums with the post calculations for the final cloth is around 5 ms.

5 Conclusion and Future Work

Our system is linear, straightforward and based on simple 3D pendulums. It is capable of adding physical like reaction effects to skeleton-based body very easily. Plus it is highly customizable: we can control reaction time, target direction and damping of the motion. The current system does not enforce angular constraints. We need to incorporate them in our future work in order to simulate real joint constraints, that exists in most skeleton-based bodies. Beside that, our future work will be to couple this system with a balance solver: that should give us not only the physical reactions to external perturbations, but also the realism that comes from the different strategies that the articulated body uses in order to regain balance. Our system can be used to simulate hair which is an acyclic system and fits neatly in the domain that the 3D pendulums system can simulate.

References

1. H. Welbergen van, B.J.H. Basten van, A. Egges, Z.M. Ruttkay, and M. H. Overmars. Real time character animation: A trade-off between naturalness and control. In *Eurographics - State-of-the-Art-Report*, Munich, 2009.
2. Ronen Barzel, John F. Hughes, and Daniel N. Wood. Plausible motion simulation for computer graphics animation. In *Proceedings of the Eurographics workshop on Computer animation and simulation*, 1996.
3. James Gain and Dominique Bechmann. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph.*, 27:107:1–107:21, November 2008.
4. Matthias Müller, Jos Stam, Doug James, and Nils Thürey. Real time physics: class notes. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 1–90. ACM, 2008.
5. Paul Kanyuk. Brain springs: Fast physics for large crowds in wall-e. *IEEE Computer Graphics and Applications*, 29:19–25, 2009.
6. Eugene Hsu, Marco da Silva, and Jovan Popović. Guided time warping for motion editing. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 45–52. Eurographics Association, 2007.
7. Paul S. A. Reitsma and Nancy S. Pollard. Evaluating motion graphs for character animation. *ACM Trans. Graph.*, 26, October 2007.
8. Armin Bruderlin and Lance Williams. Motion signal processing. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, 1995.
9. Okan Arikan, David A. Forsyth, and James F. O'Brien. Pushing people around. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, 2005.
10. Victor Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. In *ACM SIGGRAPH 2005 Papers*, 2005.
11. Victor Zordan, Adriano Macchietto, Jose Medina, Marc Soriano, and Chun-Chih Wu. Interactive dynamic response for games. In *Sandbox '07: Proceedings of the 2007 ACM SIGGRAPH symposium on Video games*, 2007.
12. Yoan D. Landau. *Adaptive control : the model reference approach / Yoan D. Landau.* Dekker, New York :, 1979.
13. Evangelos Kokkevis, Dimitri Metaxas, and Norman I. Badler. User-controlled physics-based animation for articulated figures. In *Proceedings of the Computer Animation*, CA '96, Washington, DC, USA, 1996. IEEE Computer Society.
14. Victor B. Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH / Eurographics Symposium on Computer animation*, pages 89–96. ACM Press, 2002.
15. Brian Allen, Derek Chu, Ari Shapiro, and Petros Faloutsos. On the beat!: timing and tension for dynamic characters. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, 2007.
16. Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. In *Proceedings of 29th annual conference on ComputerGraphics&InteractiveTechniques*, SIGGRAPH, 2002.
17. Pascal Volino, Nadia Magnenat Thalmann, and François Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Transaction on Graphics*, 28, 2009.
18. Roy Featherstone. A divide-and-conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics. part 1:basic algorithm. 1999.
19. Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel o(log(n)) calculation of rigid-body dynamics. part 2:trees,loops,& accuracy. 1999.