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

Ahmad Abdul Karim, Thibaut Gaudin, Alexandre Meyer, Axel Buendia, Saïda Bouakaz

▶ **To cite this version:**

Ahmad Abdul Karim, Thibaut Gaudin, Alexandre Meyer, Axel Buendia, Saïda Bouakaz. Adding physical like reaction effects to skeleton-based animations using controllable pendulums. Lecture Notes in Computer Science, 2011, Transactions on edutainment VI, 6758, pp.111-121. 10.1007/978-3-642-22639-7_12 . hal-01126420

HAL Id: hal-01126420
https://hal.science/hal-01126420v1

Submitted on 3 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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. These pendulums always return to a preferred direction that can be fixed in advance or can be modified during the motion by external predefined data (such as keyframe). Our pendulums are fully controllable, concerning reaction time and damping, and the results are completely deterministic. They are easy to implement, even without any prior knowledge of physical simulations. Our system is applicable on articulated body with predefined motion data (manually set or captured) or procedural animation.

1 Introduction

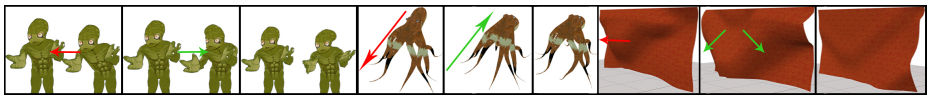


Fig. 1: Some example uses of our system. Red arrows represent external perturbations. Green arrows represent the system's response

There is a lot of research into developing convenient methods for adapting existing articulated body animations to suit other environments and characters. The joints in a given articulated body can typically have many degrees of freedom, many constraints of length or angles, and additionally are generally required not to allow the body to self-penetrate. 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.* [15] 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 [3].

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. They may also be made more cartoonish 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 user-definable target direction. Our approach has two objectives. Firstly, we ensure body length constraint between any two joints by only working on the angle between the 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 scenarios 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 [8].

2 Related Work

Our pendulums oscillate visually like real 3D pendulums by computing their movements with a mass-spring approach. A mass-spring approach is a very simple way to simulate physics-based animation [13], as it offers an intuitive and flexible means of modeling a mechanical system. In Pixar's movie WALL-E [10], they used a mass-spring system in a derived fashion to animate large crowds of humans and robots in a believable way.

Editing the 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 [9], or combining/blending existing motions (often represented in an animation graph) [14]. Others propose the use of signal processing tools to modify animations [4].

Parallel of these approaches, an important aspect is to add physical reactions to animations, like blending an existing motion with a physical response [2]. Zordan *et al.* in [18, 17] 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 [11]. All of these controllers are like our pendulum bringing a bone to a preferred direction but we differ in several essential ways. Firstly, the other methods always try to return to a preferred direction (or position), even if there is no external perturbations. This introduces a delay in the produced animation between the target pose (keyframe or motion capture data) and the response of the PD controller (as seen in [19]). On the other hand, our system is a superimposed 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 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 [1] 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. An MRAC controller is designed to be temporally controlled but it calculates and adapts its gain constants on each frame using forces and torque calculations. Our pendulums do not need any adaptive pass once the user sets the reaction time and damping. Additionally, they can be modified in real time.

We differ from other systems of skeleton-driven deformations like [5] 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 [8]. With no intention to compete against realistic fabric simulation seen in [16], we show a rigid tissue represented by a tree of bones animated by our approach with a large time-step and controllable computation.

3 3D pendulums

In our system, a bone of an articulated body is animated as a pendulum with a configurable target 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. This spring pulls the pendulum toward this direction (described in Section 3.1). This idea allows the system to easily add plausible oscillations to any animation with a temporal control (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 in

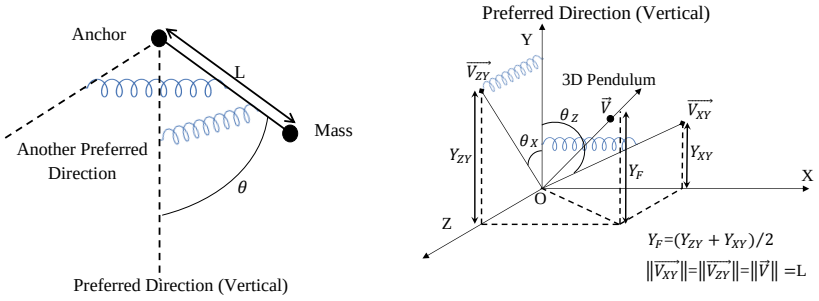


Fig. 2: (a) 3D pendulum (b) A 3D pendulum composed of two 2D pendulums with Y as their preferred direction

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's 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, and $\dot{x}$ is its velocity. We integrate this motion using the Verlet scheme [13] 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 achieve temporal control on the spring damper movement, 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:

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

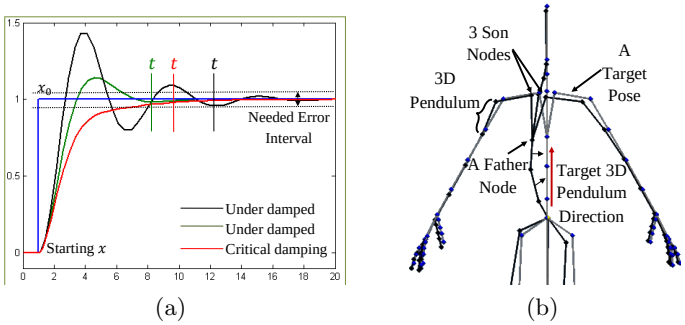
or

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

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 with target direction in Grey

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

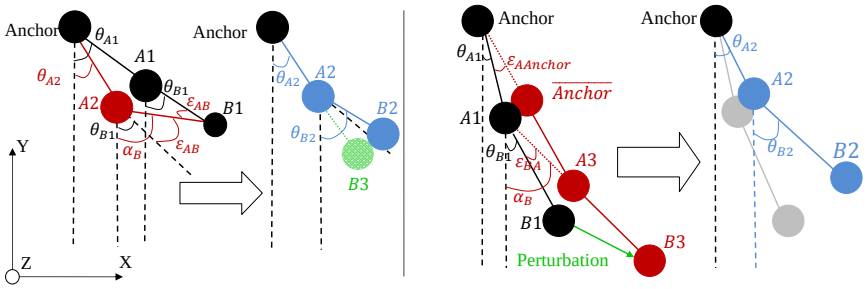
The skeleton of an articulated body is a tree of connected joints (articulations). By connecting several 3D pendulums and by defining the target direction for each one of them, the final result is a tree of pendulums that map the articulated structure, as shown in Figure 3(b). Some of the 3D pendulums act as a father node for several others. When they move, the anchor points of their children move. 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's 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 these kinds of loops, we use an update system inspired by Featherstone's divide and conquer algorithm [6, 7]. This algorithm eliminate any cyclic calculation problems and breaks the computation into two main linear passes. The first is a bottom-up pass through the articulated body tree, and the second is carried out from the top to the bottom. We adopt this paradigm completely. Only the calculations differ, as listed below:

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: the father node will be perturbed in the same direction as the previous method, but with less amplitude. It is useful when the application decides that the father 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: assigning predefined priorities on the children.

We can imagine many other possibilities based on a specific application's needs. Our system is quite easy to implement and the actual calculations in each strategy require only basic knowledge of 3D vector math. No prior knowledge of physics systems is required; we do not compute the inertia matrix nor we use the notion of force. At the same time we can use physics principles to enhance the end result like in the case of the father pendulum integrating its children's errors based on the inertia matrix.

3.4 3D model Skeleton Vs. 3D pendulums tree

In the following section we demonstrate our system with skinned 3D models, using a predefined skeleton to construct the pendulums tree. 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 preferred rest direction calculated from the bone rest pose orientation. By maintaining the hierarchy of the base skeleton, we have a pendulums tree that maps this skeleton perfectly. While playing motion data, we modify the target direction of each corresponding pendulum, mimicking the base animation exactly. If the 3D pendulums start to react to an external perturbation, each of the 3D pendulums orientation and position is applied to its corresponding bone.

4 Applications and Results

In this section, we present several ways to use the 3D pendulums tree: adding 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 pendulum's reaction time, damping, and target direction is totally controllable. The results were computed on an Intel Core 2 Duo 2GHz, 2 GB RAM, with an 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 to counter any penetrations that occur.

4.1 Adding physical reaction effects to any skeleton-based bodies

In 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 superimposed physical effects is only 0.3 ms.

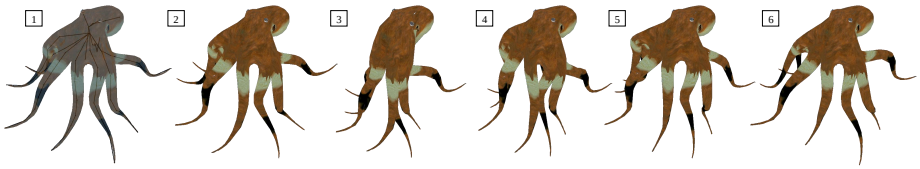


Fig. 5: From top left: [1] the model with its 3D pendulums, [2] rest pose, [3] we pull all the tentacles toward the center, [4] reacting, [5] tentacles overshooting (underdamped regime), [6] return to rest pose

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. When an external perturbation occurs, our system reacts to this while continuously trying to return to the desired target pose. With such a technique, our system adds plausible physical reaction effects to predefined animation data, as a superimposed animation layer. 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 easy and intuitive way.

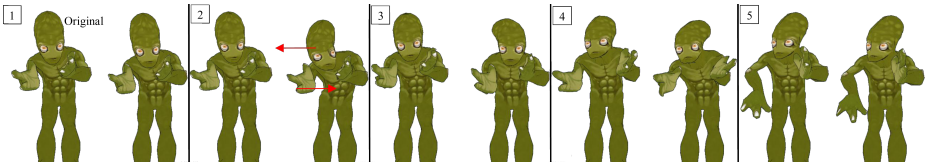
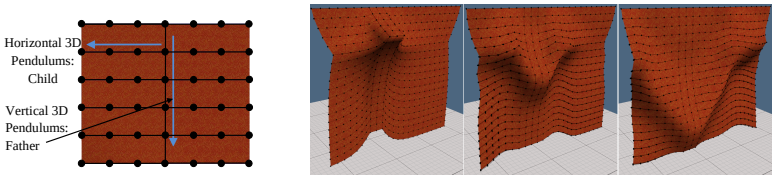


Fig. 6: From top Left: [1] Original (on the left) and our simulated articulated body (on the right), [2] Two perturbations, [3] to [5] Reaction and returning to the original keyfarme

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 added cost is the result of reading the motion capture data in order to change the pendulum's target direction, integrating the perturbation, and then performing the main integration (as previously described).

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. This results in a fully reactive cloth, without any tearing problems, that maintains its horizontal and vertical dimensions, 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 relatively fast method that can plausibly simulate the motion of reactive cloth. In Figure 7(b), an external perturbation is applied to the middle three vertical pendulums. In order to optimize calculation time, those three vertical pendulums are the only ones actively being simulated (with the horizontal children of each one of them). The mesh is simulated using approximately 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. Additionally 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 into our future work in order to simulate real-life joint constraints that exists in most skeleton-based bodies. In addition to this, we are investigating coupling the system with a balance solver. This work would provide a body with the ability to actively work to maintain its balance, in opposition to the passive reactions described in this paper. Our system could also be used to mimic hair, which is an acyclic system and fits neatly in the domain that the 3D pendulums system can simulate.

References

1. Allen, B., Chu, D., Shapiro, A., Faloutsos, P.: On the beat!: timing and tension for dynamic characters. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (2007)
2. Arikan, O., Forsyth, D.A., O'Brien, J.F.: Pushing people around. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (2005)
3. Barzel, R., Hughes, J.F., Wood, D.N.: Plausible motion simulation for computer graphics animation. In: Proceedings of the Eurographics workshop on Computer animation and simulation (1996)
4. Bruderlin, A., Williams, L.: Motion signal processing. In: Proceedings of the annual conference on Computer Graphics & Interactive Techniques. SIGGRAPH (1995)
5. Capell, S., Green, S., Curless, B., Duchamp, T., Popović, Z.: Interactive skeleton-driven dynamic deformations. In: Proceedings of 29th annual conference on ComputerGraphics&InteractiveTechniques. SIGGRAPH (2002)
6. Featherstone, R.: A divide-and-conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics. part 1:basic algorithm (1999)
7. Featherstone, R.: A divide-and-conquer articulated-body algorithm for parallel o(log(n)) calculation of rigid-body dynamics. part 2:trees,loops,& accuracy (1999)
8. Gain, J., Bechmann, D.: A survey of spatial deformation from a user-centered perspective. ACM Trans. Graph. 27, 107:1–107:21 (November 2008)
9. Hsu, E., da Silva, M., Popović, J.: Guided time warping for motion editing. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (2007)
10. Kanyuk, P.: Brain springs: Fast physics for large crowds in wall-e. IEEE Computer Graphics and Applications 29, 19–25 (2009)
11. Kokkevis, E., Metaxas, D., Badler, N.I.: User-controlled physics-based animation for articulated figures. In: Proceedings of the Computer Animation (1996)
12. Landau, Y.D.: Adaptive control : the model reference approach / Yoan D. Landau. Dekker, New York : (1979)
13. Müller, M., Stam, J., James, D., Thürey, N.: Real time physics: class notes. In: SIGGRAPH '08: ACM SIGGRAPH 2008 classes. pp. 1–90. ACM (2008)
14. Reitsma, P.S.A., Pollard, N.S.: Evaluating motion graphs for character animation. ACM Trans. Graph. 26 (October 2007)
15. van, H.W., van, B.B., Egges, A., Ruttkay, Z., Overmars, M.H.: Real time character animation: A trade-off between naturalness and control. In: Eurographics (2009)
16. Volino, P., Magnenat Thalmann, N., Faure, F.: A simple approach to nonlinear tensile stiffness for accurate cloth simulation. ACM Transaction on Graphics (2009)
17. Zordan, V., Macchietto, A., Medina, J., Soriano, M., Wu, C.C.: Interactive dynamic response for games. In: Sandbox: Proceedings of the ACM SIGGRAPH symposium on Video games (2007)
18. Zordan, V., Majkowska, A., Chiu, B., Fast, M.: Dynamic response for motion capture animation. In: ACM SIGGRAPH 2005 Papers (2005)
19. Zordan, V.B., Hodgins, J.K.: Motion capture-driven simulations that hit and react. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer animation (2002)