In *Computer Graphics and Applications*,
(*Proc. of Pacific Graphics '95*, Seoul, Korea, August 21–24, 1995),
S.Y. Shin and T. Kunii (Eds.), World Scientific, pp. 118–132.

Pseudo Dynamic Keyframe Animation with Motion Blending on the Configuration Space of a Moving Mechanism

Joo-Haeng Lee and *Myung-Soo Kim*
Dept. of Computer Science
POSTECH
Pohang 790-784, South Korea.

Abstract

This paper presents an effective way of generating and controlling dynamic motions of a multi-linked body system while interpolating given boundary conditions. The basic idea is based on smoothly blending two dynamic motion curves in the *Configuration space* of a moving mechanism. Given a time interval $[t_1, t_2]$, two motion curves $C_1(t)$ and $C_2(t)$, $t_1 \leq t \leq t_2$, are generated by physical simulation so that they satisfy the boundary conditions at $t = t_1$ and $t = t_2$, respectively. The two curves are then smoothly blended on the Configuration space so that the blended motion curve satisfies the boundary conditions at $t = t_1$ and $t = t_2$. Higher degree blending functions provide extra degrees of freedom in controlling the generated motion in the middle of the time interval $[t_1, t_2]$. It is straightforward to extend the motion blending technique to keyframe animation. Compared with the conventional animation techniques for a multi-linked body system, the motion blending technique achieves not only computational efficiency but also pseudo dynamic motions by incorporating the merits of each of the previous methods within one system.

Keywords: Keyframe animation, dynamics, Euler-Lagrange Equation, Configuration space, phase space, motion blending

1 Introduction

In commercial animation movies, the motion of a character is usually generated by keyframe animation technique. However, this technique is quite laborious and requires considerable talent in order to generate natural motion. There are relatively few computer animators who can produce animation of a quality comparable to that of a Walt Disney animation movie. An immediate goal of computer animation research is to develop appropriate animation tools which enable even the novice to produce natural character motion. Since most end users of computer animation softwares are graphic artists, it is also important to develop animation tools which are easy to use and understand. Keyframe animation, in spite of its other shortcomings, is popular in such softwares primarily because it is easy to use.

Although many techniques have been developed for character animation, the most powerful tool currently is still the straightforward method of motion recording, *rotoscoping*. For a human-like character with arms and legs, the easiest way to generate a walking motion is to record the actual human walking data and postprocess them in accordance with what appears most appropriate to the character. Simple postprocessing techniques include scaling and local modifications, etc. One can easily manipulate the motion data once the data have been fitted with spline curve segments, but the stepping motion of a character introduces singularities on the motion curve. For these postprocessing steps, the actual data taken from a real human motion plays an important role as a standard natural motion.

The approach of this paper is based on a similar strategy. We start with some natural dynamic motions of a moving mechanism and modify them by using a blending technique to generate a pseudo dynamic motion. The initial motion data can be obtained from any of the previous animation techniques. One simple way of generating them is by doing forward dynamic simulations on the moving mechanism. Dynamic simulation usually produces natural dynamic motion; however, it is very difficult to make the resulting motion satisfy both the initial and final boundary conditions simultaneously. This is the reason why the SC (Spacetime Constraint) method is promising in computer animation since the SC method generates motion which satisfies all the constraints including the boundary conditions [5, 7, 10, 14]. However, the SC method is based on a non-linear optimization technique, which is computationally quite intensive since the optimization is done on the motion curve itself. Moreover, further modifications to the generated motion require repeated recomputing of the entire optimization. The process of making an animation movie usually involves a lengthy sequence of repetition and local modification. As a result, it is very important to devise a new method which may reuse some of the previous motion data and, at the same time, support local modifications efficiently.

The motion blending method suggested in this paper overcomes the drawback of physical dynamic simulation and generates a motion curve which satisfies the boundary conditions at both initial and final keyframes. This is done by blending two dynamic motion curves, each satisfying one of the two boundary conditions; the resulting blended motion curve satisfies all the boundary conditions. Many efficient methods have been developed for forward dynamic simulation [2, 3, 4, 6, 12, 13]; based on simulating the law of physics, they produce quite natural dynamic motions. The blended motion curve generates slightly less natural dynamic motion; however, this is inevitable when the motion curve must satisfy all the boundary conditions. The SC method tries to minimize awkwardness by optimizing the external energy consumed during the motion.

The input motion curves of the motion blending method are not only limited to those generated by forward dynamic simulation. Any pair of motion curves can be used for blending as long as each boundary condition is satisfied by at least one of them. It is also allowed for only one curve to satisfy both initial and final conditions, while the other satisfies neither. For example, the motion curve generated by the SC method can be blended with any other motion curve for local and/or global modifications. This capability would make the SC method more powerful as an animation tool by providing a way to easily modify and control the generated motion curve.

This method can also be generalized to the case of blending three or even more motion curves, in which case, the blending function is required to have a higher dimension. Furthermore, by using a high degree blending function with extra degrees of freedom, we can easily make the motion blending method switch into a variant of the SC method, in which the extra degrees of freedom become the parameters to be optimized in the non-linear optimization process. By choosing an appropriate set of basic motion curves to be blended and the parameters for the extra degrees of freedom in the blending function, we can significantly reduce the size of non-linear optimization for the SC method. The motion curve thus generated can also be used as an initial solution for the more precise SC method. By using a suboptimal solution as an initial solution, the non-linear optimization process may converge very quickly.

The rest of this paper is organized as follows. In §2, the underlying geometry for motion blending

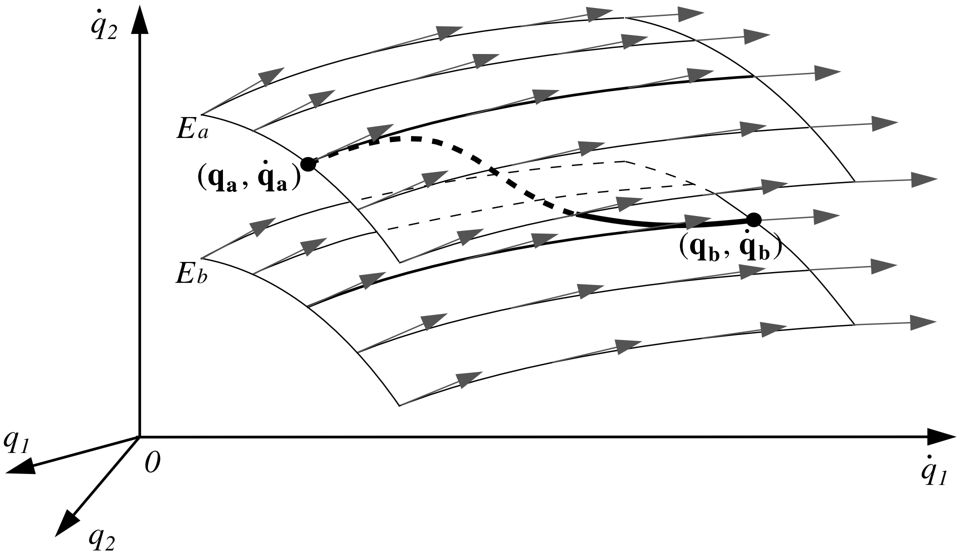


Figure 1: Phase Flows on Energy Level HyperSurfaces

is explained. It is based on the concept of phase flow on an energy level hypersurface embedded in the phase space. In §3, the motion blending technique is applied to the keyframe animation of (i) a ball of mass m and (ii) a multi-linked open-chain. Finally, we conclude this paper in §4.

2 Phase Flows on Energy Level Hypersurfaces

In this paper, we interpret the motion of a moving mechanism as a geometric curve in the Configuration space of the mechanism. In classical mechanics [1], the Configuration space is parametrized by generalized coordinates $\mathbf{q} = (q_1, \ldots, q_n) \in R^n$, and the *phase space* of the mechanism is parametrized by $(\mathbf{q}, \dot{\mathbf{q}}) = (q_1, \ldots, q_n, \dot{q}_1, \ldots, \dot{q}_n) \in R^{2n}$. The dynamic motion curve following the law of physics is given by the integral curve $\mathbf{q}(t)$ which satisfies the following Euler-Lagrange Equation:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} = \Gamma, \tag{1}$$

where L is the Lagrangian and Γ is a generalized external force to the mechanism. In the above equation, we can observe that the equation holds even if we change t into $-t$ and $\dot{\mathbf{q}}$ into $-\dot{\mathbf{q}}$. That is, by reversing the time and the generalized velocity vector, we can also generate a natural dynamic motion that satisfies the Euler-Lagrange Equation. After we compute the solution, by reversing the time and the velocity vector back to the originals, the reversed motion curve becomes a solution to the original Euler-Lagrange Equation. In this way, we generate a natural dynamic motion curve which satisfies the final conditions at the end of the curve. Dynamic simulation performed in this way is called *backward simulation*.

When there is no external force exerted on the mechanism, the total energy is preserved and the phase flow curve lies on the same energy level hypersurface (see Figure 1). When the boundary conditions given at the initial and final keyframes produce different levels of energy, we need to exert external forces onto the system so that the phase flow curve smoothly deviates from one energy level hypersurface and flows to the other energy level hypersurface. When the transition is made in such a way that the intermediate energy level changes monotonically, it is clear that the total external force exerted is optimal. However, the monotonicity of energy level provides only a sufficient condition for the motion curve to become an optimal solution. In most cases, an optimal motion curve produces the energy transition in an oscillating fashion. The energy consumption mainly depends on the given keyframe boundary conditions. For example, consider the problem of interpolating two keyframes (p_i, v_i) of a point p, $i = 1, 2$, where the two positions p_1 and p_2 are at a long distance and the two velocities are $v_1 = v_2 = 0$. The motion curve $p(t)$ should have a high speed in the middle of the curve so that it can travel over a long distance from p_1 to p_2. Though the kinetic energy is 0 at the two boundary keyframes, the high speed in the middle of the curve introduces a high kinetic energy. Therefore, even an optimal motion curve would experience an energy jump in the middle of the curve.

The energy minimization problem as discussed above presents difficulties, mainly because the energy level surface is a hypersurface in the phase space (parametrized by $(\mathbf{q}, \dot{\mathbf{q}})$), not in the Configuration space of the moving mechanism. Instead of directly manipulating the hypersurface in the $2n$-dimensional phase space, we project two phase flow curves (each corresponding to one of the two boundary conditions) into the Configuration space. It is clear that the two projected motion curves $C_1(t)$ and $C_2(t)$ maintain their respective energy. The motion blending technique generates a motion curve $C(t)$ in the Configuration space by blending the two motion curves $C_1(t)$ and $C_2(t)$ as follows:

$$C(t) = \gamma_{C_1(t), C_2(t)}(f(t)), \quad \text{for } t_1 \leq t \leq t_2,$$

where $\gamma_{C_1(t), C_2(t)}(s)$, $0 \leq s \leq 1$, is the geodesic path which connects two points $C_1(t)$ and $C_2(t)$ in the Configuration space. We assume the Configuration space is a *complete* Riemannian manifold in which the geodesic between any two points is defined [8, 11].

One may wonder why we do not blend the two phase flow curves directly in the phase space. By doing this in the phase space, it becomes quite difficult to make the resulting blended phase curve $(\mathbf{q}(t), \dot{\mathbf{q}}(t))$ satisfy the necessary condition between the two components $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$:

$$\frac{d}{dt}\mathbf{q}(t) = \dot{\mathbf{q}}(t).$$

However, when we first construct the motion curve $C(t)$ in the Configuration space, the corresponding phase curve is automatically generated as $(C(t), C'(t))$ in the phase space. With an appropriate blending function, the phase curve satisfies the two boundary conditions.

Given a phase curve $(C(t), C'(t))$, we can examine the change of total energy $T(C'(t)) + U(C(t))$ and see if the energy level change is monotonic, where $T(C'(t))$ is the kinetic energy and $U(C(t))$ is the potential energy of the motion. If the total energy change is almost monotonic, the energy consumption would also be almost optimal. Otherwise, when there is a big energy level jump, we may try to use other blending functions to generate a motion curve that consumes less energy. When the blending functions are formulated with extra degrees of freedom, an optimization can be performed over the parameters for the extra degrees of freedom. Nevertheless, in many test examples, we have not observed much improvement by using different blending functions. This means that the motion blending curves are usually very close to the optimal solutions and the energy level jumps are somewhat inevitable due to the given keyframe conditions.

3 Motion Blending

3.1 Ball of Mass m

To illustrate the basic idea of motion blending technique, we first consider a simple example of throwing a ball of mass m. Assume the ball has position $p(t)$ and velocity $v(t)$ at time t, and $p_i = p(t_i)$ and $v_i = v(t_i)$ for $i = 1, 2$. The Lagrangian $L(p, v)$ is given by:

$$L(p, v) = T(v) - U(p),$$

where $T(v)$ is the kinetic energy and $U(p)$ is the potential energy. They are given by:

$$T(v) = \frac{1}{2}m < v, v >, \quad \text{and} \quad U(p) = -m < g, p >,$$

where g is the gravitational force and $< v_1, v_2 >$ denotes the inner product of two vectors v_1 and v_2. When there is no external force, the Euler-Lagrange Equation (1) reduces to:

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial v} - \frac{\partial L}{\partial p} &= 0 \\ \frac{d}{dt}mv + mg &= 0 \\ m\dot{v} + mg &= 0 \\ \dot{v} = -g & \end{aligned}$$

The velocity $v(t)$ is given by:

$$v(t) \;=\; v_1 + \int_{t_1}^{t} \dot{v}dt \;=\; v_1 + \int_{t_1}^{t} -g dt \;=\; v_1 - g(t - t_1).$$

The forward simulation produces a path $C_1(t)$:

$$C_1(t) \;=\; p_1 + \int_{t_1}^{t} (v_1 - g(t - t_1))\, dt \;=\; p_1 + v_1(t - t_1) - \frac{1}{2}g(t - t_1)^2.$$

Similarly, the backward simulation produces a path $C_2(t)$:

$$C_2(t) \;=\; p_2 + v_2(t - t_2) - \frac{1}{2}g(t - t_2)^2.$$

The motion blending generates a path $C(t)$:

$$C(t) = (1 - f(t))C_1(t) + f(t)C_2(t),$$

where $f(t)$ is a blending function which satisfies the following conditions:

$$f(t_1) = 0, \quad f(t_2) = 1, \quad f'(t_i) = 0, \quad \text{for } i = 1, 2.$$

One can easily check that the motion blending curve $C(t)$ satisfies:

$$C(t_i) = C_i(t_i) \quad \text{and} \quad C'(t_i) = C_i'(t_i), \quad \text{for } i = 1, 2.$$

To avoid a jump in the acceleration of the motion blending curve at each keyframe, we also need to enforce the following second derivative condition to the blending function:

$$f''(t_1) = 0 \quad \text{and} \quad f''(t_2) = 0.$$

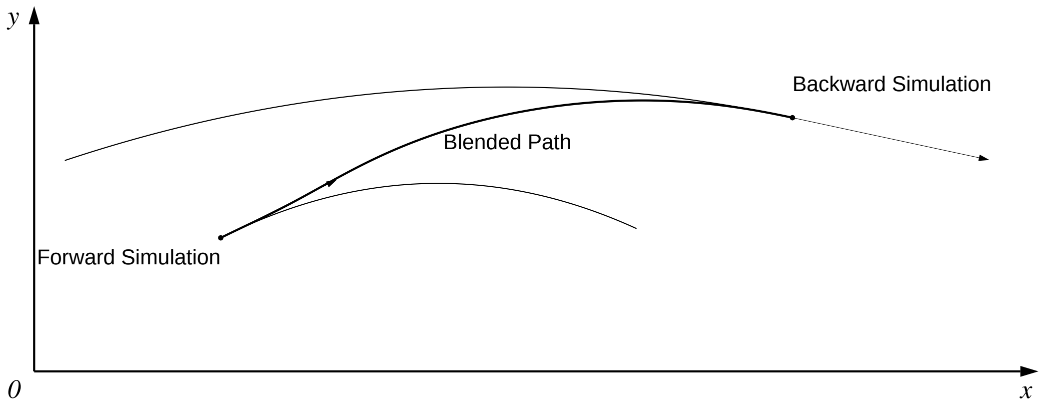


Figure 2: Throwing a Ball of Mass m

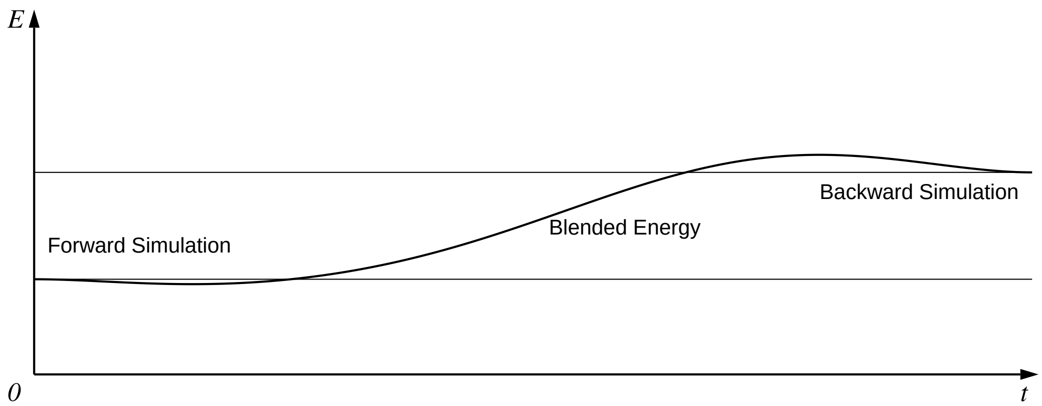


Figure 3: Total Energy Graph

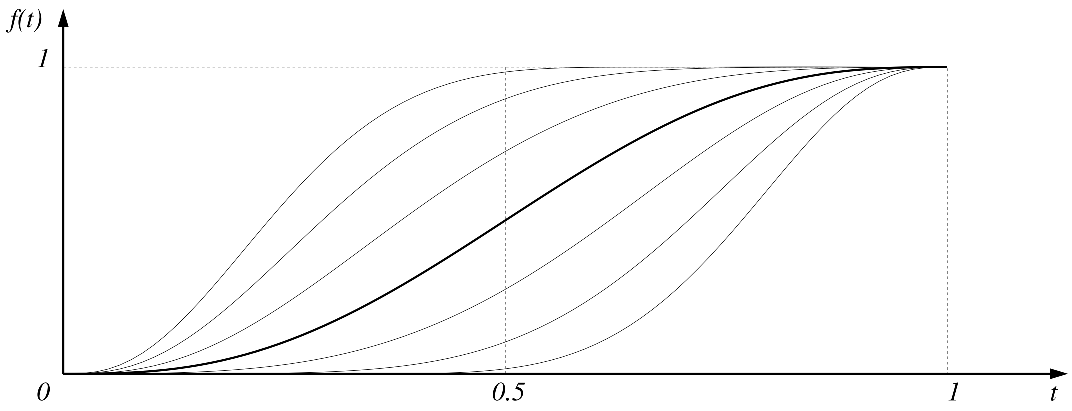


Figure 4: Graphs of Blending Functions

Figure 2 shows the result of a motion blending for throwing a ball of mass m in the xy-plane. Figure 3 shows the total energy change of the ball under motion. The total energy jumps significantly in the middle of the motion. The energy jump is due to the following velocity property of the motion blending curve:

$$C'(t) = (1 - f(t))C_1'(t) + f(t)C_2'(t) + f'(t)(C_2(t) - C_1(t)).$$

The first part of the velocity: $(1 - f(t))C_1'(t) + f(t)C_2'(t)$ essentially blends the velocities of two component curves and generates no larger speed than $\|C_1'(t)\|$ or $\|C_2'(t)\|$ when $0 \leq f(t) \leq 1$. The second part: $f'(t)(C_2(t) - C_1(t))$ becomes large when the position difference: $C_2(t) - C_1(t)$ has a large magnitude in the middle of the curve. The slope of the blending function $f(t)$ usually becomes larger in the middle of the time interval $[t_1, t_2]$. (Figure 4 shows a set of blending functions which interpolate different values at $t = 0.5$.) The velocity component $f'(t)(C_2(t) - C_1(t))$ becomes much larger in the middle of the curve and the kinetic energy increases by the square of the amount of the velocity component. As a result, when we have extra degrees of freedom in generating a family of blending functions, it is better to choose $f(t)$ for which the derivative $f'(t)$ changes almost inverse proportionally to the magnitude $\|C_2(t) - C_1(t)\|$.

Although a better choice of the blending function $f(t)$ may reduce the maximum magnitude of energy jump, the improvement mainly depends on the two boundary conditions. Even if the kinetic and potential energies are similar at both initial and final keyframes, a big energy jump occurs when the boundary conditions are given in such a way that the forward and backward simulation motion curves are much different (see Figures 5-6). We may generate many other motion curves by changing the blending functions and selecting the one that has an optimal energy change. Figures 7–8 show the motion blending paths and their energy graphs, respectively, which result from different blending functions. The original path and energy graph are drawn with dark curves. Also shown are the path and engergy graph of the optimal solution obtained by the SC method. They are drawn with light curves. We can try to reduce the maximum energy jump by scaling down the velocity at the maximum energy peak at time $t = t_3$ and repeating the motion blending procedure for each of the time subintervals $[t_1, t_3]$ and $[t_3, t_2]$. However, we have observed new energy peaks occurring on both of the two time subintervals (see Figure 9).

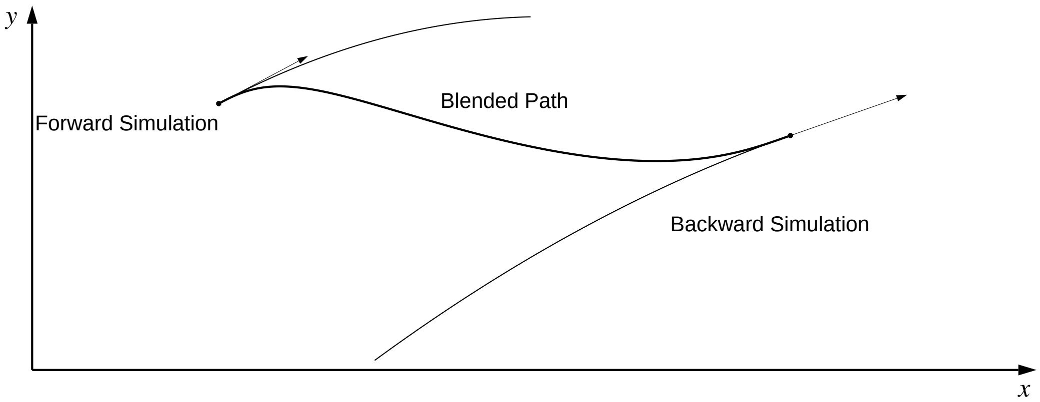


Figure 5: Example with High Intermediate Speed

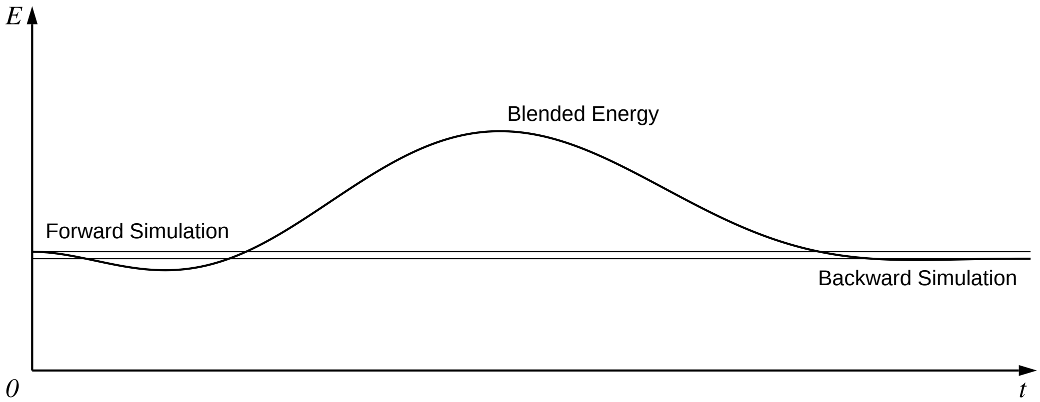


Figure 6: Total Energy Graph

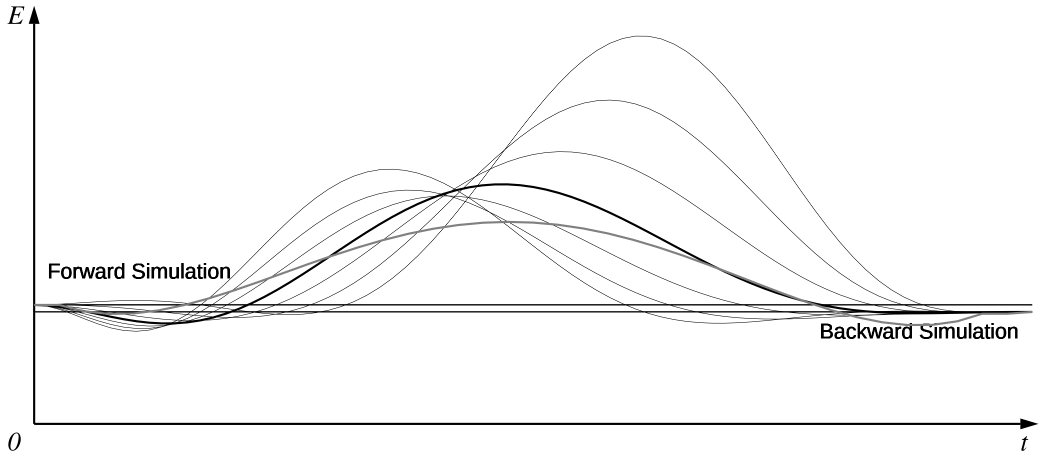


Figure 7: Paths Resulting From Different Blending Functions

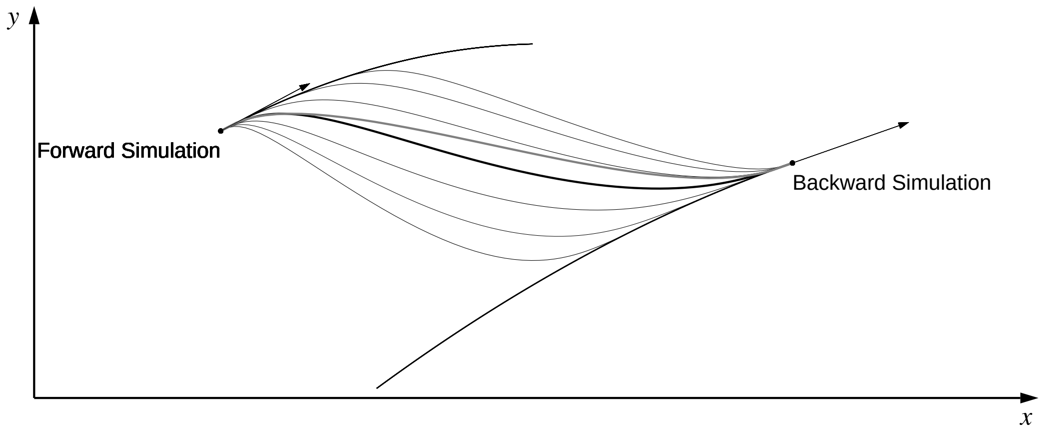


Figure 8: Different Energy Graphs

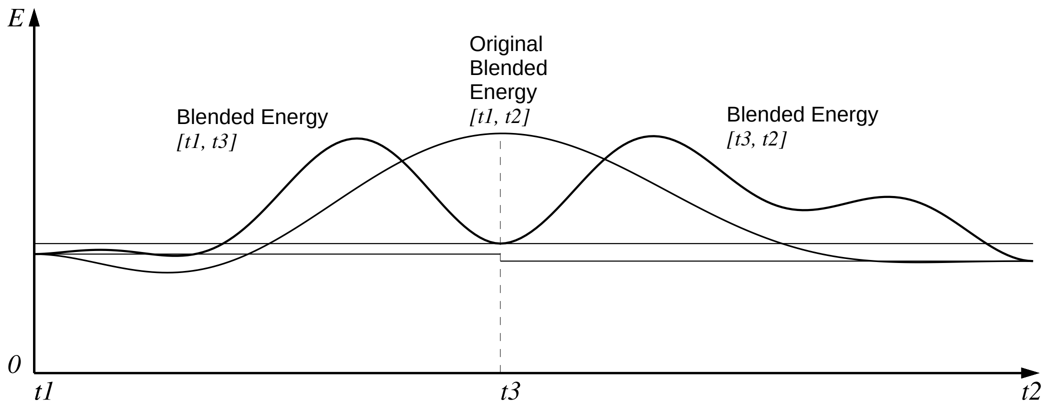


Figure 9: New Energy Peaks

3.2 Open-Chain Multi-Linked Body System

We consider how the motion blending technique can be applied to the case of open-chain multi-linked body system. There are many ways to formulate the multi-link body kinematics and dynamics [2, 6, 9, 12, 13]. For the implementation here, we use the explicit formulas developed in Reference [9] for the Lagrangian and the kinetic and potential energy terms. The formulas are very compact using Lie group and Lie algebra notations. Although they are somewhat difficult to read, the compact formulations significantly reduce numerical errors in the dynamic simulation.

Given an open-chain multi-link body system with n joints with joint angles: $\theta = (\theta_1, \ldots, \theta_n)$, the total kinetic energy $T(\theta, \dot{\theta})$ is defined by:

$$T(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta},$$

where $M(\theta) \in R^{n\times n}$ is the manipulator inertia matrix (see [9] for more details). The total potential energy $U(\theta)$ is given by:

$$U(\theta) = \sum_{i=1}^{n} m_i g h_i(\theta),$$

where m_i is the mass of the i^{th} link, g is the gravitational force, and $h_i(\theta)$ is the height of the center of mass of the i^{th} link. Combining the two terms, the Lagrangian is given by:

$$\begin{aligned} L(\theta, \dot{\theta}) &= \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} - U(\theta) \\ &= \frac{1}{2}\sum_{i,j=1}^{n} M_{ij}(\theta)\dot{\theta}_i\dot{\theta}_j - U(\theta). \end{aligned}$$

The equation of motion is given by substituting the above equation into the Euler-Lagrange Equation (1):

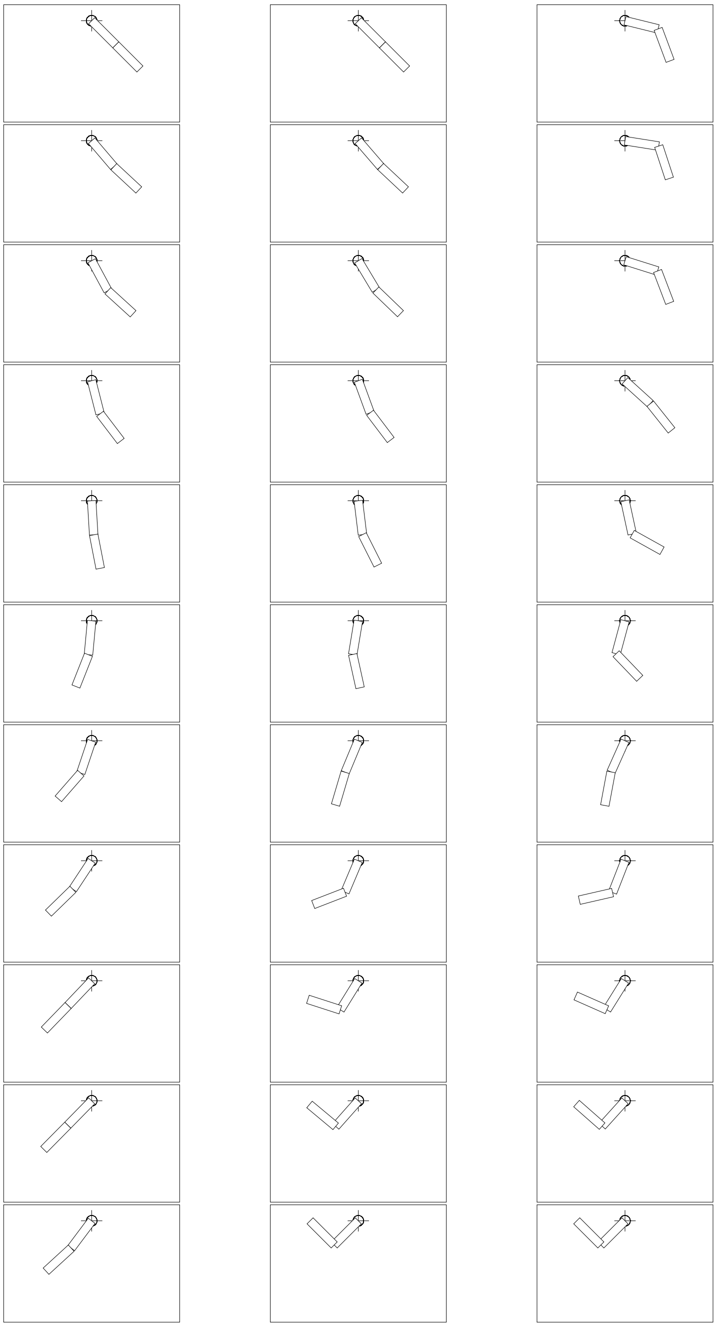
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} = \Gamma_i, \quad \text{for } i = 1, \ldots, n,$$

where Γ_i represents the actuator torque and other nonconservative generalized forces acting on the i^{th} joint. The equation of motion can be written as follows:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + N(\theta, \dot{\theta}) = \tau,$$

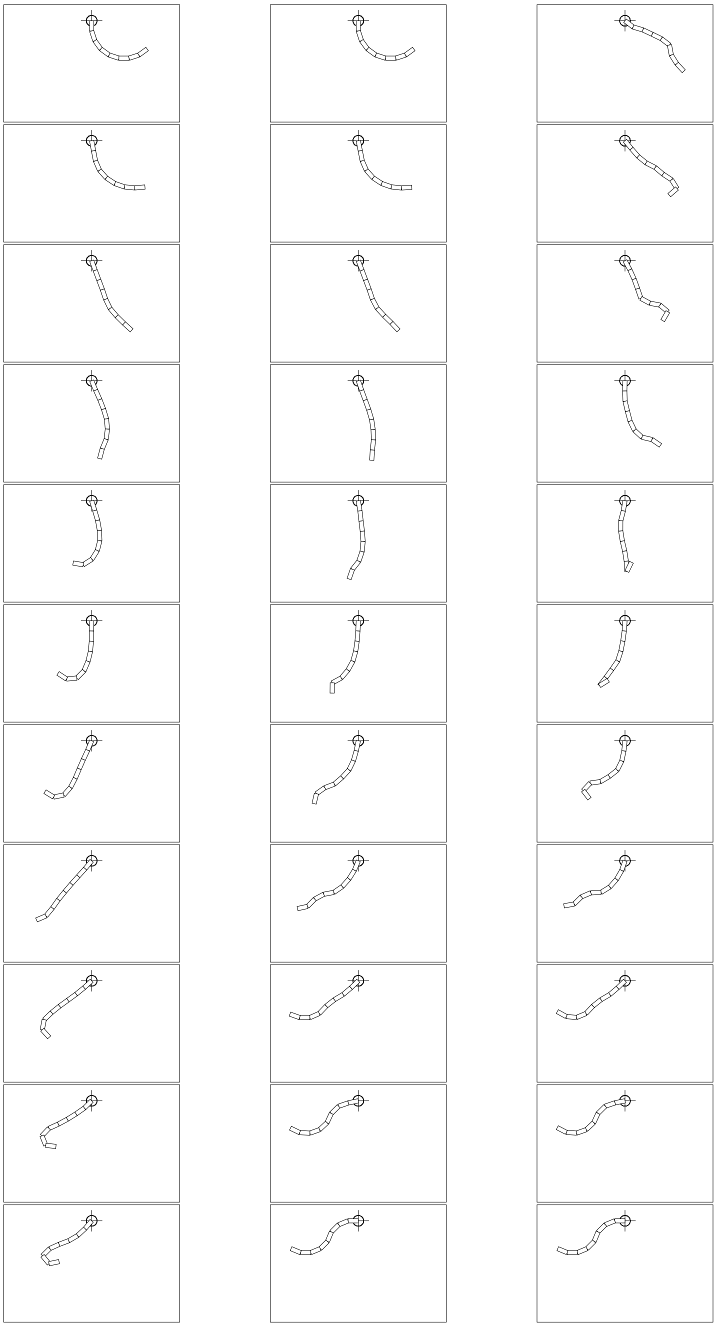
where the matrix $C(\theta, \dot{\theta}) \in R^{n\times n}$ is the Coriolis matrix for the manipulator, τ is the vector of actuator torques, and $N(\theta, \dot{\theta})$ includes gravity terms and other forces which act at the joints (see [9] for more details).

The physical forward simulation is done by computing the solution $\ddot{\theta}$ of the above linear system at each time step, which gives the second derivatives of the joint angles with respect to time. This means that we need to integrate the second derivatives to obtain the angular velocity and the angle of each joint. After they are determined, the matrices $M(\theta)$, $C(\theta, \dot{\theta})$, and $N(\theta, \dot{\theta})$ are updated in the linear system. Now, for the new linear system, we need to solve the solution $\ddot{\theta}$ at time $t + \Delta t$ for the forward simulation and at time $t - \Delta t$ for the backward simulation. The same procedure is repeated and the simulation results are blended using an appropriate blending function. In this case, the motion blending is applied to the joint space of the multi-linked body system, which is the Configuration space of the system.



(a) (b) (c)

Figure 10: Blended Motion of Two-Segmented Body



(a) (b) (c)

Figure 11: Blended Motions of Eight-Segmented Body

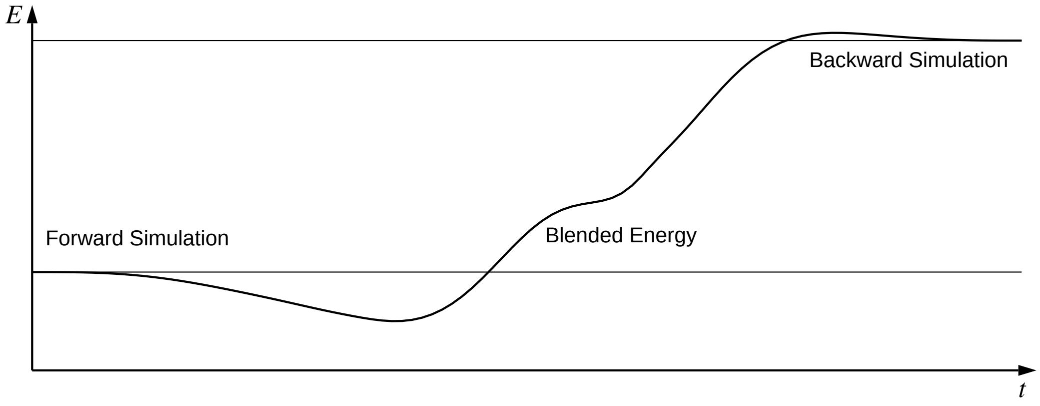


Figure 12: Energy Graph for Two-Segmented Body

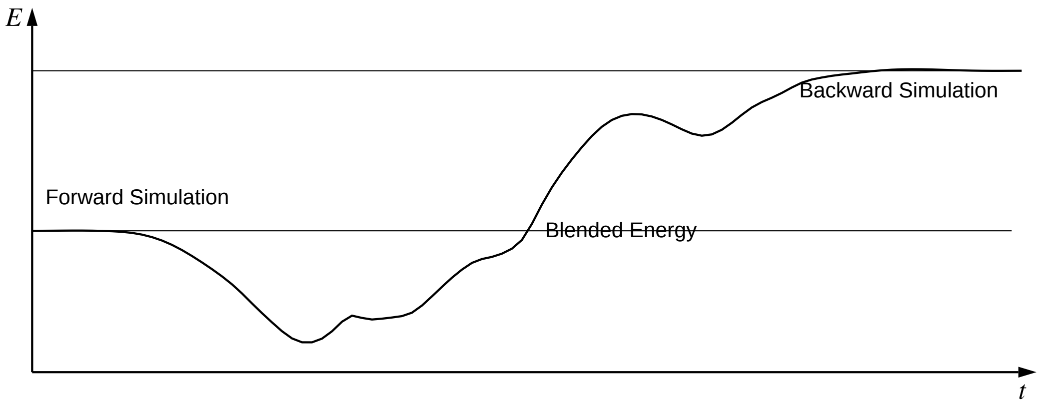


Figure 13: Energy Graph for Eight-Segmented Body

Figures 10–11 show two animation sequences of a multi-linked body system which are generated by the motion blending technique. In each animation sequence, the left column shows the result of a forward simulation, the right column shows the result of a backward simulation, and the middle column shows the motion blending result of the left and right columns. Figures 12–13 show the total energy graphs for the two animation sequences. The blended motion looks pseudo dynamic and quite natural in the real animation; the motion also satisfies both the initial and final keyframe conditions. The motion blending result can be easily modified by using other blending functions while satisfying the boundary conditions. But, in these examples, we have included neither the collision detection nor the angular constraint for each pair of consecutive links.

4 Conclusion

In this paper, a new technique is proposed that blends two physically simulated dynamic motions of a moving mechanism. The result is a pseudo dynamic motion of the moving mechanism that satisfies the given keyframe boundary conditions. The motion blending technique is a hybrid method that combines the merits of previous methods in that:

1. A purely kinematic keyframe animation system can be easily upgraded so that it can also include a certain level of dynamic realism.
2. A forward physical simulation method can be easily adapted so that it generates a motion which satisfies both the initial and final boundary conditions.
3. An optimal motion curve generated by the Spacetime Constraint method can be easily modified without repeating the time-consuming non-linear optimization process.
4. The space of blending functions provides a low dimensional space to which the Spacetime Constraint method can be applied. The space has a much lower dimension compared with the space of motion curves to which the conventional Spacetime Constraint method has been applied.

These advantages are obtained at slight expense to motion realism. However, the resulting motion is not far from an optimal solution. The mathematical tools provided by the phase space and the kinetic and potential energy terms can also play an important role in analyzing the behavior of various other motion curves. They are not limited to the motion blending curves we have discussed in this paper.

References

[1] Arnold, V.I., *Mathematical Methods of Classical Mechanics*, Second Ed., Springer-Verlag, New York, 1989.

[2] Armstrong, W.W., and Green, M.W., "The dynamics of articulated rigid bodies for the purposes of animation," *Proceedings of Graphics Interface '85*, Canadian Information Processing Society Toronto, Ontario, Canada, PP. 407-415.

[3] Barzel, R., *Physically-Based Modeling for Computer Graphics*, Academic Press, San Diego, CA, 1992, pp. 14–24.

[4] Badler, N., Barsky, B., Zeltzer, D., *Making Them Move*, Morgan Kaufmann, San Mateo, CA, 1991, pp. 269–271.

[5] Cohen, M., “Interactive Spacetime Control for Animation,” *Computer Graphics (Proc. of SIGGRAPH '92)*, Vol. 26, No. 2, pp. 293–302, 1992.

[6] Hoffmann, C., and Hopcroft, J., “Simulation of Physical Systems from Geometric Models,” *IEEE Trans. on Robotics and Automation*, Vol. 3, No. 3, pp. 194–206, 1987.

[7] Liu, Z., Gortler, S., and Cohen, M., “Hierarchical Spacetime Control,” *Computer Graphics (Proc. of SIGGRAPH '94)*, pp. 35–42, 1994.

[8] Milnor, J., *Morse Theory*, Princeton Univ. Press, Princeton, 1963.

[9] Murray, R., Li, Z., and Sastry, S., *A Mathematical Introduction to Robotic Manipulation*, CRC Press, Boca Raton, Florida, 1994.

[10] Ngo, J., and Marks, J., “Spacetime Constraints Revisited,” *Computer Graphics (Proc. of SIGGRAPH '93)*, pp. 343–350, 1993.

[11] Spivak, M., *A Comprehensive Introduction to Differential Geometry*, Vol. I, Publish or Perish, Inc., Berkeley, 1970.

[12] Wilhelms, J., “Using Dynamic Analysis for Realistic Animation of Articulated Bodies,” *IEEE Computer Graphics and Applications*, Vol. 7, No. 1, pp. 12–27, 1987.

[13] Wittenburg, J., *Dynamics of Systems of Rigid Bodies*, B.G. Teubner, Stuttgart, 1977.

[14] Witkin, A., and Kass, M., “Spacetime Constraints,” *Computer Graphics (Proc. of SIGGRAPH '88)*, Vol. 22, No. 4, pp. 159–168, 1988.