A Procedural Approach to Solving Constraints of Articulated Bodies

*J. Won Lee, †Nakhoon Baek, *Dongho Kim and *James K. Hahn

*Department of Computer Science, The George Washington University, Washington D.C., United States
†School of Electronic & Electrical Eng., Kyungpook National University, Taegu, Republic of Korea

Abstract
Realistic motions of articulated bodies are usually generated by using physically-based animation methods such as constrained dynamics. However, these methods involve heavy computations and complicated numerical methods. We present an alternative way of solving constraints of articulated bodies. Our objective is not physically correct motions but visually plausible animation. In our method, each object of the constrained body is first moved according to their physical parameters and external forces, without considering any constraints. Then the objects are translated and rotated to satisfy the given constraints. Instead of strict simulation of physical laws, we suggest procedural formulations for solving constraints. This formulation has the power of generating visually plausible motions as presented in our example animation sequences. Since our method is free from complex numerical methods, it is fast enough to be used in real-time applications such as virtual reality, computer games and real-time simulations. Numerical stability is another merit of our method. This procedural approach can be an alternative to strict physically-based animation methods.

1. Introduction

In the area of computer animation, we have many methods for generating realistic motions of various kinds of objects. Among them, articulated body animation is regarded as one of the most important research topics, since many real world objects can be modeled as articulated bodies. So far, most of works on articulated body animation are based on the physically-based modeling and dynamics simulation.[1, 2, 3, 4, 5, 6, 7]

These physically-based modeling methods have their own pros and cons. For example, constrained dynamics method can be used to generate realistic motions of articulated bodies. It is based on Newtonian physics, and thus has the power of producing physically correct motions. However, it uses constraint equations along with the equations of motions, and usually results in heavy computations. Additionally, physical quantities of objects including forces and accelerations are often directly used to control motions of articulated bodies, even though it is not an intuitive way of control. For example, users find it difficult to place an object at a specific location by controlling the forces applied on it. Thus, currently, the constrained dynamics method is not so widely used even though it is one of the best methods for realistic articulated body animation.[8]

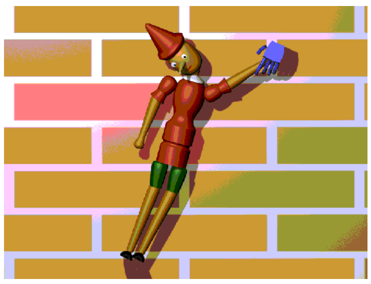


Figure 1: *Interactive control of an articulated body*

An alternative to physically-based modeling is the paradigm of *procedural methods*.[9] In 1980's, some research have focused on mimicking physical phenomena rather than strictly simulating physical laws.[10, 11, 12] Although these approaches were motivated basically by the the lack of sufficient computing power, they achieved visual plausibility and also provided easy control of complex phenomena. Even today, we still do not have sufficient computing power to simulate physically-based models of complicated phenomena in real time. In virtual reality environments and computer games, for example, it is a requirement to display the motion of objects in real time, even at the expense of displaying physically incorrect motions.

This paper presents a method to solve constraints procedurally to interactively calculate motions of articulated bodies. Our objective is not necessarily to generate physically correct motions but *visually plausible motions*. Although it is not directly derived from Newtonian dynamics, our method provides an efficient and numerically stable way of generating visually plausible motions. Its calculation procedure is based on the positions and orientations of the objects rather than dynamics properties such as forces and accelerations. Hence it additionally provides an easy way of controlling the motions via specifying the desired positions and orientations. Figure 1 shows an example of interactive control of an articulated body. It is especially suitable for presenting dragging effects, which often occur when the user moves a selected portion of the articulated body.

Section 2 is a brief review of related works including the traditional physically-based paradigm and the procedural paradigm. In Sections 3 and 4, we present a detailed description of our procedural method and how to apply it to articulated bodies and various joint constraints, respectively. Section 5 demonstrates some examples of image sequences generated by the method. Finally, conclusions and future work are given in Section 6.

2. Previous Works

Since articulated body animation is one of the major topics in computer animation, there has been much research devoted to it. They can be classified into two categories: *kinematics-based methods* and *dynamics-based methods*. Both have their advantages and disadvantages.

Kinematics-based methods are relatively easy to implement and good for interactively controlling the motions. However, since they involve positions, orientations, and velocities, it is difficult to apply physical laws involving accelerations. Inverse kinematics method is one of kinematics-based methods for articulated body animation. Girard and Maciejewski applied an inverse kinematics technique for motions of running and walking humans.[1] Badler et al. developed an inverse kinematics-based algorithm for solving multiple constraints concurrently, and applied it for articulated bodies.[2] Currently, several inverse kinematics systems are available and some run at interactive speeds.[5] Kinematics methods and inverse kinematics methods can be used to generate realistic active motions of articulated bodies. However, it is difficult to incorporate external and internal forces to generate realistic passive motions.

In the case of dynamics-based methods, the constrained dynamics method is widely used for articulated body animation. The constrained dynamics method uses a system of equations, which consists of equations of motions and constraint equations. The systems of equations are usually too complex to be solved efficiently, and much works are focused on the effective way of solving these systems of equations. Among them, Armstrong and Green presented a recursive formation for the constrained dynamics method and introduced a linear time algorithm for constrained dynamics equations of articulated bodies.[13] Presently, the constrained dynamics methods are usually solved through one of the two numerical techniques: coordinate reduction technique and Lagrange multiplier technique. Currently a linear time solution for Lagrange multiplier technique is available.[14]

Isaacs and Cohen introduced the inverse dynamics method as a way of controlling the motions of articulated bodies.[15] In this method, inverse forces are calculated to satisfy user-specified accelerations. Westenhofer and Hahn presented a motion control system that integrates kinematics-based controls into a constrained dynamics system.[16] Constrained dynamics method and inverse dynamics method are sufficient to generate realistic and physically correct motions of articulated bodies. However, solving the systems of constrained dynamics equations is hard to perform interactively, even though there are theoretically linear-time solutions. Thus the dynamics-based method is not widely used for real-time applications such as virtual reality and computer games.

In 1980's, some researchers mimicked dynamics behaviors of objects using only relatively simple mathematical equations. For example, Fournier and Reeves[10] and Peachey[11] both successfully expressed ocean waves through combining simple trigonometric equations rather than using fluid dynamics formulations. Weil also succeeded in presenting complex shapes of cloth objects using simple catenary functions.[12]

Currently, these procedural methods are being revisited due to their simplicity and their visually plausible results. Milenkovic introduced a position-based formation for non-articulated objects.[17] Using this formulation, he demonstrated that small sphere particles contained in an hour-glass shape can be animated in a way similar to traditional constrained dynamics simulations.

Gascuel[18] introduced the displacement constraint to solve constraints of articulated objects quickly. The idea of separating constraint solving part from equation of motion is similar with our procedural method. However, their method is iterative-based method and has separating velocity adjusting procedure.

Barzel introduced a *fake* dynamics technique, which can be classified as a kind of procedural kinematic method.[8, 19] This technique was successfully used to mimic the dynamics behavior of ropes and springs in the animation film "Toy Story." In our knowledge, there has not been any procedural method for articulated bodies.

3. Basic Idea

3.1. Overview

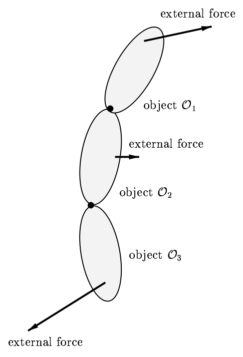
Generation of an articulated body motion means deciding the position and orientation of each object in the articulated body for each time instant. From the dynamics point of view, the current geometric configurations (positions, orientations, etc.) and physical parameters (velocities, accelerations, etc.) are calculated from the configurations of the previous time instant. This calculation process has two requirements:

1. The motion of each object should be generated according to the equations of motions, in which external forces are involved.
2. The final geometric configurations of objects should satisfy constraints due to the joints of the articulated body.

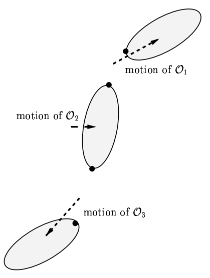
The constrained dynamics method starts from a system of equations, which explicitly express the above requirements. Equations of motions and constraint equations are usually integrated into a system of equations, which is usually solved by a relatively complex numerical method.

In contrast, the basic idea of our procedural method is separating the whole process into two stages each of which concentrates on one of the two above requirements. In the *update stage*, positions and orientations of objects making up the articulated body are updated by solving the equations of motions. The animator can specify the position and orientation of an object explicitly, if desired. Notice that any constraint equation is not considered at this time, as shown in Figure 2.(b).

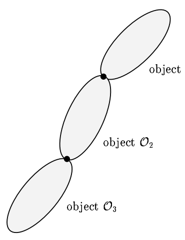
In the *adjustment stage*, we adjust the positions and orientations of each object to satisfy the constraints, as shown in Figure 2.(c). During this adjustment process, a procedural calculation of required transform for each object is used rather than the original constraint equations. Notice that our goal is the visual plausibility rather than physically correct motion, and thus the constraint equations are not solved explicitly.



(a) original configuration


motion of $\mathcal{O}_1$
motion of $\mathcal{O}_2$
motion of $\mathcal{O}_3$

(b) update stage



(c) adjustment stage

Figure 2: *Overview of the procedural constraint solving method*

In comparison with traditional constrained dynamics methods, our procedural method has two advantages:

1. It is faster than any constrained dynamics methods since our adjustment equations make it possible to satisfy the constraints without considering complex physical properties such as accelerations and velocities.
2. It can be integrated into a direct manipulation system in which an object is selected to change

its position and orientation interactively, since our method solves the constraints based on positions and orientations. In constraint dynamics, inverse dynamics is required to control positions or orientations.

The update stage is straightforward. We can use any dynamics methods to update positions and orientations of objects, since the constraint equations are excluded during this update step. In our implementation, we use Euler's integration method for this purpose, mainly due to its simplicity.

The strictly physically-based modeling often includes friction forces and drag equation from the fluid dynamics. In our implementation, we add a damping term to approximate them. Letting $\mathbf{x}^i$, $\mathbf{v}^i$ and $\mathbf{a}^i$ be the position, velocity and acceleration of an object at the i-th time step, Euler integration of Newton's law is expressed as follows:

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \mathbf{v}^i(\Delta t) \tag{1}$$

and

$$\mathbf{v}^{i+1} = \mathbf{v}^i + \mathbf{a}^i(\Delta t) \tag{2}$$

where Δt is the time interval between animation frames.

Combining Equations (1) and (2), the position can be evaluated with the following single equation:

$$\mathbf{x}^{i+1} = \mathbf{x}^i + (\mathbf{x}^i - \mathbf{x}^{i-1}) + \mathbf{a}^i(\Delta t)^2.$$

Since $(\mathbf{x}^i - \mathbf{x}^{i-1})$ corresponds to the velocity, we multiply a damping constant to it. Thus, the final equation for $\mathbf{x}^{i+1}$ is:

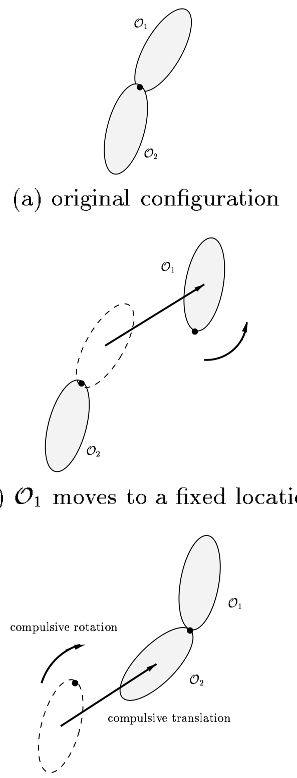
$$\mathbf{x}^{i+1} = \mathbf{x}^i + k(\mathbf{x}^i - \mathbf{x}^{i-1}) + \mathbf{a}^i(\Delta t)^2,$$

where $k \in [0, 1]$ is the damping constant. By controlling the value of k, we can control the visual illusions of frictions and/or motions in fluid such as water. Details of the adjustment process will be explained in the following sections.

3.2. Fixed position solution for two-object articulated bodies

In the adjustment stage, positions and orientations of objects are adjusted to satisfy the joint constraints. We formulated this adjustment process to reflect the characteristics of constraint forces. In the case of constrained dynamics, constraint forces should satisfy the following two characteristics:

1. The constraint forces applied to the objects connected by a joint have same magnitudes but opposite directions.
2. Constraint forces should be workless.



(a) original configuration

(b) $\mathcal{O}_1$ moves to a fixed location

(c) constraint solving

Figure 3: *Compulsive translation and rotation*

Our adjustment process aims to mimic the constraint forces as much as possible. Especially, we hope to represent the motions of an articulated body when the user drags a point on the body.

In our approach, constraints are solved by translating and rotating the objects to satisfy the constraints. In the update stage, objects are moved due to the external forces and/or user inputs without considering any constraints. Thus the movements usually break the joint constraints of articulated bodies. The major role of the constraint solving is to decide the translational and rotational motions that satisfy the given constraints. We call these translations and rotations due to the constraints *compulsive translations* and *compulsive rotations*, respectively. *Compulsive motion* will be used to refer to both compulsive translation and compulsive rotation.

To formulate the equations of compulsive translation and rotation, we will start from a simplest case. Suppose that an articulated body has only two objects and the position and orientation of an object $\mathcal{O}_1$ is *fixed* after the update stage. This situation often occurs when the user drags $\mathcal{O}_1$ to a specific location.

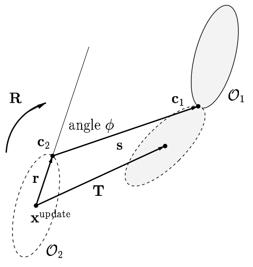


Figure 4: *Solving the constraint through moving the object*

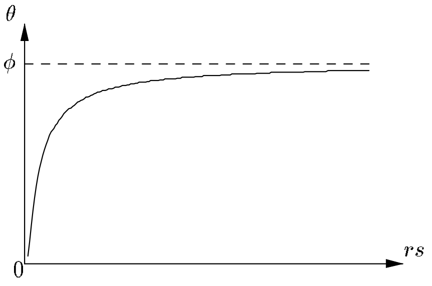


Figure 5: *The graph of* $\theta = \phi\, e^{-\frac{h}{rs}}$

Then another object $\mathcal{O}_2$ should move closer to the dragged object, as shown in Figure 3.

Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be the position of the constraint point on $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. Our objective is moving $\mathbf{c}_2$ to $\mathbf{c}_1$ by applying compulsive translation and compulsive rotation onto $\mathcal{O}_2$, as shown in Figure 4. The arm vector $\mathbf{r}$ for $\mathbf{c}_2$ is calculated as:

$$\mathbf{r} = \mathbf{c}_2 - \mathbf{x}^{\mathrm{update}},$$

where $\mathbf{x}^{\mathrm{update}}$ is the position of $\mathcal{O}_2$ after the update stage. Now the compulsive translation vector $\mathbf{T}$ and compulsive rotation matrix $\mathbf{R}$ should satisfy the following equality condition:

$$\mathbf{r} + \mathbf{s} = \mathbf{R}\,\mathbf{r} + \mathbf{T}, \qquad (3)$$

where the vector $\mathbf{s}$ equals to $\mathbf{c}_1 - \mathbf{c}_2$.

Notice that the constraint force should be workless. In other words, we should move $\mathcal{O}_2$ along the shortest path to minimize the compulsive motion of $\mathcal{O}_2$. In the case of rotation, $\mathbf{R}$ can be specified with the rotation axis $\mathbf{A}$ and the rotation angle θ. We can intuitively calculate the rotation axis $\mathbf{A}$ from the cross product of $\mathbf{r}$ and $\mathbf{s}$:

$$\mathbf{A} = \frac{\mathbf{r} \times \mathbf{s}}{|\mathbf{r}||\mathbf{s}|}.$$

Calculation of the rotation angle θ is indirectly derived from rotational dynamics. For the rotating object $\mathcal{O}_2$, torque τ with initial configuration can be calculated as follows:

$$\tau = \mathbf{r} \times \mathbf{f},$$

where $\mathbf{f}$ is the linear force applied at $\mathbf{c}_2$. This imaginary force $\mathbf{f}$ will move $\mathbf{c}_2$ to $\mathbf{c}_1$. Assuming $\mathbf{f}$ generates constant acceleration $\mathbf{a}$, it is possible to approximate $\mathbf{f}$ as follows:

$$\mathbf{f} = m_2\,\mathbf{a} = \frac{2\,m_2}{(\Delta t)^2}\,\mathbf{s},$$

where m_2 is the mass of $\mathcal{O}_2$ and Δt is the time interval between animation frames. Now, the magnitude of torque τ can be expressed as:

$$\tau = \frac{2\,m_2}{(\Delta t)^2} r\, s \sin\phi, \qquad (4)$$

where s is the length of $\mathbf{s}$ and ϕ is the initial angle between $\mathbf{r}$ and $\mathbf{s}$. Letting α be the angular acceleration, τ also can be expressed as:

$$\tau = I_2\,\alpha, \qquad (5)$$

where I_2 is the moment of inertia for $\mathcal{O}_2$. From Equations (4) and (5), the angular acceleration α can be approximated as follows:

$$\alpha = \frac{2\,m_2}{I_2(\Delta t)^2} r\, s \sin\phi. \qquad (6)$$

Theoretically, the rotation angle θ can also be approximated from the above angular acceleration. However, Equation (6) is available only for the initial configuration since the angle ϕ varies along with the rotation of $\mathcal{O}_2$ due to the angular acceleration α. Thus we only use the characteristic physical parameters to build up our rotation angle calculation formula.

Our starting point for approximating θ is the simple observation that θ is a value between 0 and ϕ. Additionally, the angle θ is influenced by parameters r and s. Thus it is natural to use exponential function as follows:

$$\theta = \phi\, e^{-\frac{h}{rs}},$$

where the constant h is equivalent to $\frac{2\,m_2}{I_2(\Delta t)^2}$. As shown in Figure 5, this formulation shows that the rotation angle θ is nonlinearly proportional to r and s, while its value is bounded in $(0, \phi)$. When r and/or s are increased, the rotation angle θ approaches to ϕ, while θ goes to near 0 with small r or s values. When r or s is 0, it means no rotation at all and thus the angle θ is trivially 0. The constant h is a user-controllable parameter, which decides the ratio of θ and ϕ for given r and s.

Now we have the formulations for the rotation axis and the rotation angle. Thus the rotation matrix $\mathbf{R}$ in

Equation (3) can be calculated. The compulsive translation vector $\mathbf{T}$ is calculated from Equation (3) as follows:

$$\mathbf{T} = (\mathbf{I} - \mathbf{R})\mathbf{r} - \mathrm{s},$$

where $\mathbf{I}$ is the 3-by-3 identity matrix.

In this way, we showed that the compulsive translation vector and the compulsive rotation matrix can be calculated from the given geometric configuration. The object is then moved in order to satisfy the constraint. It is the final step of the adjust stage. In the next subsection, we will show another case in which neither of the objects has fixed location.

3.3. Moving objects solution for two-object articulated bodies

Suppose that an articulated body with two objects is moving freely. After the update stage, each object has its own position, and often does not satisfy the constraint. In the previous subsection, we presented a simpler example in which an object is fixed at a specific location. In contrast, this subsection focuses on the case in which the articulated body moves freely. Only the joint constraint restricts its motion.

The central idea in this case is calculating the position of the constraint point. Then we solve two simple cases of fixed constraint point. In other words, the original problem of moving two-object articulated body is transformed to two separate problems of fixed position cases, as shown in Figure 6.

After the update stage, we have two positions of constraint points for each object. Our objective is calculating the position of the new coincident constraint point from these positions. Notice that the constraint force should be workless. Thus, it is natural to select the new coincident constraint point $\mathbf{c}$ to be located along the line segment connecting the two given positions $\mathbf{c}_1$ and $\mathbf{c}_2$.

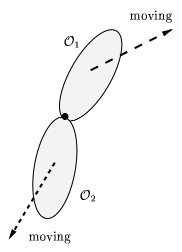
Another characteristic of the constraint force is that it is applied to the objects with the same magnitude but opposite direction. When forces with the same magnitude are applied to objects, the linear movement of each object is inversely proportional to its mass. Letting the masses of $\mathcal{O}_1$ and $\mathcal{O}_2$ be m_1 and m_2, it is intuitive that

$$m_1(\mathbf{c}_1 - \mathbf{c}) = m_2(\mathbf{c} - \mathbf{c}_2).$$

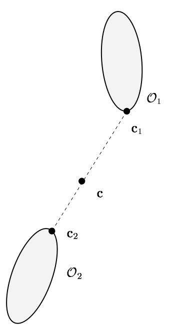
Since the new coincident constraint point is located on the line segment $\overline{\mathbf{c}_1\mathbf{c}_2}$, we can easily derive

$$\mathbf{c} = \frac{m_1\,\mathbf{c}_1 + m_2\,\mathbf{c}_2}{m_1 + m_2}.$$

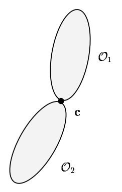
Now the two objects of the articulated body move to this coincident constraint point, as presented in the previous subsection.



(a) original configuration



(b) calculating $\mathbf{c}$



(c) final configuration

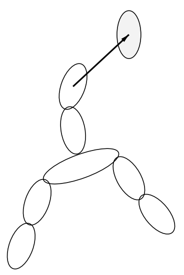
Figure 6: *Constraint solving for a free-moving articulated body*

4. Extensions

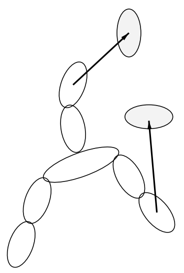
4.1. Tree-like articulated bodies

Since an object of an articulated body is connected to its adjacent objects, propagation of forces from its neighbors affects itself. This propagation process makes the motions of articulated bodies realistic. Constraint dynamics methods usually achieve the propagation process by solving equations of all constraints simultaneously. Even though we have some linear time solutions for articulated bodies, formulations of such equations are complicated and their solutions usually require sophisticated numerical computations.[20, 14]

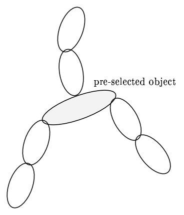
In this subsection, we extend our procedural method to general tree-like articulated bodies, which consist of multiple objects. Constraint solving for tree-like articulated bodies can be classified into three categories,



(a) dragging a single object

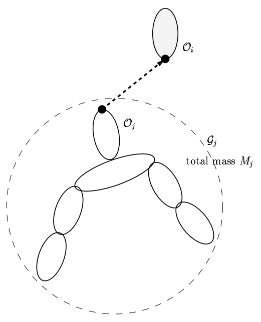


(b) dragging multiple objects

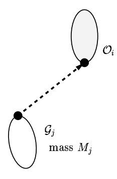


(c) free movement

Figure 7: *Constraint solving for tree-like articulated bodies*



(a) $\mathcal{O}_i$ and $\mathcal{O}_j$



(b) $\mathcal{O}_i$ and $\mathcal{G}_j$

Figure 8: *Object grouping*

as shown in Figure 7. For the first case, the position of only a single object is fixed. This case is basically similar to the single object fixed case in Section 3.2, while the number of objects in the articulated body is more than two. The next case is multiple objects being dragged, where the positions of more than one object are specified. Finally, we also have the free-moving case in which there are no external constraints. Each of these cases is presented in this subsection.

The basic idea of extending the procedural method to tree-like articulated bodies is grouping adjacent objects in order to regard them as a single object, as shown in Figure 8. Suppose that an articulated body has n objects, $\mathcal{O}_1$, $\mathcal{O}_2$, $\cdots$, $\mathcal{O}_n$. As an example, suppose that the position of $\mathcal{O}_i$ is fixed, and $\mathcal{O}_i$ is connected to $\mathcal{O}_j$ with a joint constraint. When the objects $\mathcal{O}_j$, $\mathcal{O}_{j+1}$, $\cdots$, $\mathcal{O}_k$ are all connected together, we group these objects into $\mathcal{G}_j$. Then we simplify this situation as a two-object articulated body whose objects are $\mathcal{O}_i$ and $\mathcal{G}_j$. For efficient calculation, we simply assume that the geometric shape of $\mathcal{G}_j$ is identical to $\mathcal{O}_j$, but the mass of $\mathcal{G}_j$ is the total mass of $\mathcal{O}_j$, $\mathcal{O}_{j+1}$, $\cdots$, $\mathcal{O}_k$. Then, we can calculate the compulsive motion required for $\mathcal{O}_j$, and we apply the same idea for the next object $\mathcal{O}_{j+1}$ through grouping $\mathcal{O}_{j+1}$, $\mathcal{O}_{j+2}$, $\cdots$, $\mathcal{O}_k$.

When there are multiple objects whose positions are fixed, we cannot simply apply the same idea. A possible solution can be a relaxation process, which is similar to Weil's idea for the cloth modeling.[12] For each fixed object, we apply the above grouping method to the entire articulated body with ignoring other fixed objects. Although objects may move from one place to another at each step, they will reach a stable state after a number of iterations. Of course, we should check impossible cases, which result in infinite loops.

The last case can be occurred when there is no fixed object of the tree-like articulated body after the update stage. The central idea in this case is fixing a pre-selected object. For example, we can select the torso of a human-like articulated body as its pre-selected one. At the adjustment stage, we first calculate the position of this pre-selected object. Suppose that the pre-selected object $\mathcal{O}_i$ has its neighbors $\mathcal{O}_j$, $\mathcal{O}_{j+1}$, $\cdots$, $\mathcal{O}_{j+k}$. We apply the two object articulated body solutions for each pair of $\mathcal{O}_i$ and its neighbor. Now we have k locations for each pair, and the final location of $\mathcal{O}_i$ is calculated as the weighted sum of these locations. After fixing the pre-selected object $\mathcal{O}_i$, it is straightforward to calculate the locations of other objects.

4.2. Other kinds of constraints

In constraint dynamics, handling various kinds of joints is an important issue.[14] Our procedural method works well with various joint constraints, since the constraints can be explicitly expressed procedurally. To demonstrate the power of our method, we present approaches to handle joint-angle limit constraints, multiple positional constraints and contact constraints.

The joint-angle limit constraint is widely used in articulated body motions. It is a kind of an inequality constraint, and defines the acceptable range of angles between connected objects. In constrained dynamics, the inequality equations are usually solved by linear complementary method or quadratic programming, which require heavy computations.[14]

These difficulties are due to the fact that the constrained dynamics methods have to calculate accelerations even to limit an angle in a pre-defined range. In contrast, our procedural method does not calculate any acceleration, and we can express this kind of constraint in an explicit procedural form. For an articulated body, a pair of objects will be processed using two-object case solutions. After fixing the two objects, we check whether the angle between them violates the pre-specified joint-angle constraint. When it violates the constraint, we simply limit the angle to an extreme value of the given constraint. That is all that is required to satisfy the joint-angle limit constraint.

Multiple positional constraints provide useful tools for interactive control of articulated bodies. For example, user may want to drag one hand of a human-like figure while its feet are fixed on the floor. In this case, we have three positional constraints: one for the hand and one for each foot. In inverse kinematics, an optimization method was already proposed.[2]

However, using our procedural method, it is possible to speed up its calculation without using any optimization method. Notice that the multiple positional constraints are equivalent to the multiple fixed object case of tree-like articulated bodies. As explained in Section 4.1, we can satisfy the multiple positional constraints by a relaxation process.

In dynamics-based simulations, contact constraints are one of the hard-to-solve problems.[21] They usually require complicate computations involving quadratic programming or Danzig's algorithm to solve the contact problem.[14] Contact points are calculated using collision detection techniques and checking relative velocities of the objects. Since most implementations use discrete time steps, the objects are usually penetrating each other when the collision is detected. Thus, solving contact constraint is equivalent to removing the penetration, in most cases.

In Hahn's method, the penetration is eliminated by backing up the penetrating object with its relative velocity but along the opposite direction.[22] Our idea is similar to this backing up method. However, we directly move the position of the object while Hahn uses relative velocity for the same purpose.

Suppose that an object (a bouncing ball, for example) penetrates a stationary object (the floor). After detecting the intersection, we first search for a vertex that has penetrated the deepest among the vertices of the penetrating object. Then the penetrating object is compulsively translated along the surface normal direction of the penetrated object so that the two objects are just touching. When several objects are colliding simultaneously, the above translation is applied to each pair of objects.

5. Examples

In this section, we present examples of articulated body motions to demonstrate the power of our procedural method. Figure 9 demonstrate the interactive position control of articulated bodies. User selects the uppermost object of the chain-shape body and drags it to the desired location. Figure 10 is another example of the same chain-shape body, whose mass and damping terms are changed. It is easy to find the differences in motions due to the change of physical parameters.

Figure 11 shows the motion of a human-like articulated body. User can select an object and move the object, and then other objects follow the selected object. User can generate motions similar to that of a marionette whose hand is dragged.

6. Conclusions and Future Works

We presented a procedural approach for motions of articulated bodies. Our aim was to generate visually plausible animation sequences rather than physically correct motions. Our method does not solve any systems of equations and achieves interactive control of the motions with numerical stability. This procedural approach can be an alternative to dynamics simulation, especially for real-time applications such as virtual reality environment and computer games.

The procedural approaches in computer animation are a relatively new and promising area and there are many unsolved problems. We plan to extend the procedural approach to cooperate with collisions. Integrating our procedural method with existing motion control methods is also an interesting problem.

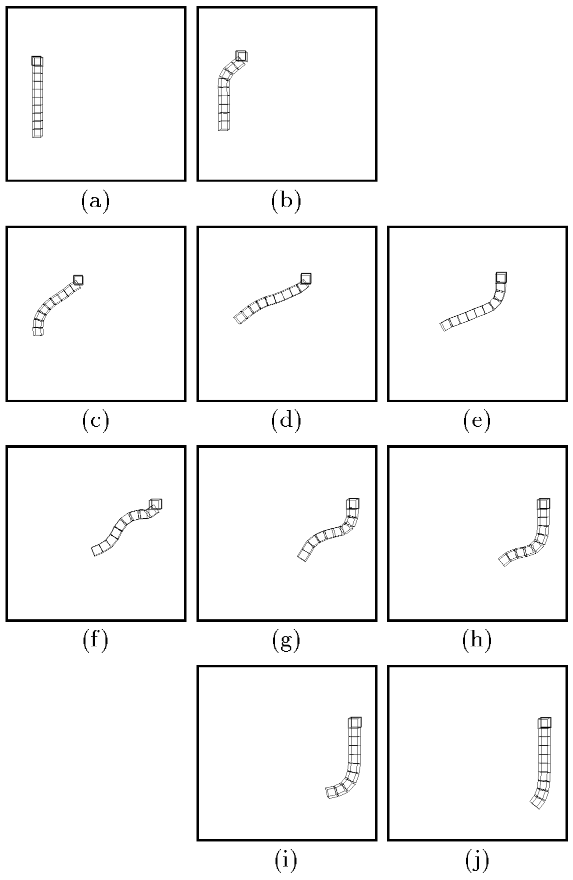


Figure 9: *Example of dragging a chain: the uppermost object is dragged interactively*

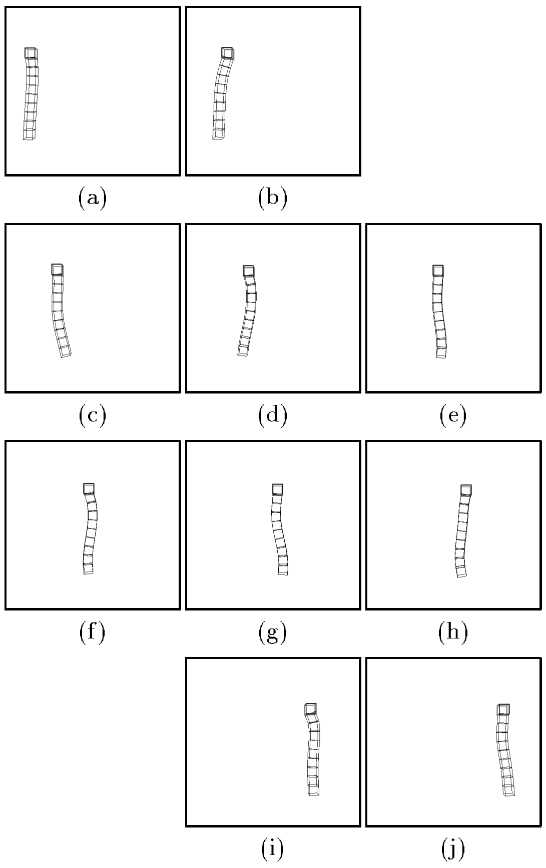


Figure 10: *Another Example of dragging a heavy chain*

References

1. M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. *SIGGRAPH'85*, 19:263–270, 1985.
2. N. I. Badler, K. H. Manoochehri, and G. Walters. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics & Applications*, 7(6):28–38, 1987.
3. M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *SIGGRAPH'90*, 24:29–38, 1990.
4. M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *SIGGRAPH'91*, 25:349–358, 1991.
5. C. Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. Master's thesis, Simon Frasier University, 1993.
6. J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.
7. J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. *SIGGRAPH'95*, pages 71–78, 1995.
8. R. Barzel. Faking dynamics of ropes and springs. *IEEE Computer Graphics & Applications*, 17(3):31–39, 1997.
9. A. Watt and M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice.* Addison-Wesley Publishing Company, 1992.
10. A. Fournier and W. T. Reeves. A simple model of ocean waves. *SIGGRAPH'86*, 20:75–84, 1986.
11. D. R. Peachey. Modeling waves and surf. *SIGGRAPH'86*, 20:65–74, 1986.
12. J. Weil. The synthesis of cloth objects. *SIGGRAPH'86*, 20:49–54, 1986.
13. W. W. Armstrong and M. W. Green. The dy-

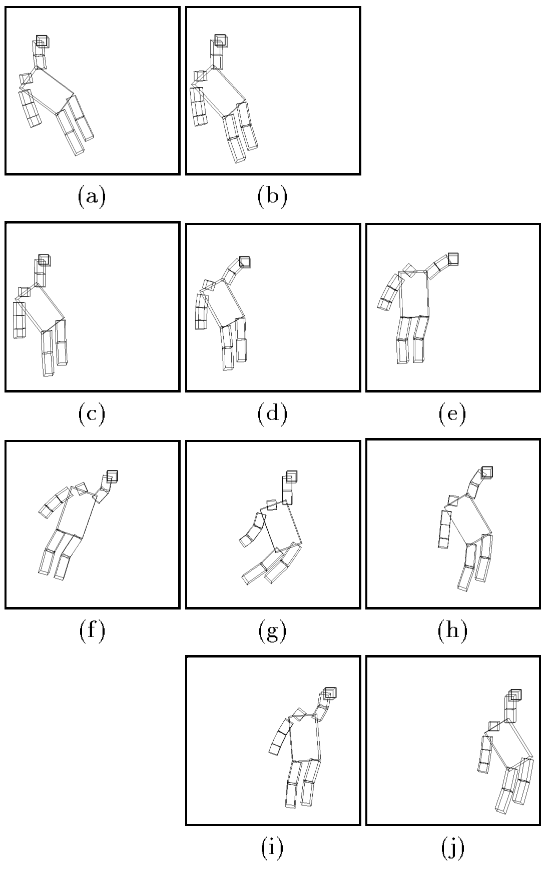


Figure 11: *Example of dragging a human-like shape*

namics of articulated rigid bodies for purposes of animation. *The Visual Computer*, 1(4):231–240, 1985.

14. D. Baraff. Linear-time dynamics using Lagrange multipliers. In *Proceedings of the ACM Conference on Computer Graphics*, pages 137–146, New York, 1996. ACM.

15. P. M. Isaacs and M. F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *SIGGRAPH'87*, 21:215–224, 1987.

16. B. Westenhofer and J. K. Hahn. Using kinematic clones to control the dynamic simulation of articulated figures. *Pacific Graphics Proceedings*, 1996.

17. V. Milenkovic. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. *SIGGRAPH'96*, pages 129–136, 1996.

18. J.-D. Gascuel and M.-P. Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, 10(4):191–204, March 1994.

19. R. Barzel, J. F. Hughes, and D. Wood. Plausible motion simulation for computer graphics animation. In *Eurographics Workshop on Animation and Simulation*, pages 183–197. Springer-Verlag, 1996.

20. P. Schröder and D. Zeltzer. The virtual erector set: Dynamic simulation with linear recursive constraint propagation. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):23–31, 1990.

21. D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. *SIGGRAPH'94*, pages 23–34, 1994.

22. J. K. Hahn. Realistic animation of rigid bodies. *SIGGRAPH'88*, 22:299–308, 1988.