MOTION CONTROL USING EXTENDED GENERALIZED COORDINATE TRANSFORMATIONS

Jihun Park, Donald Fussell, and James C. Browne

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-92-25 April 1992

IK 42-25

Motion Control using Extended Generalized Coordinate Transformations

Jihun Park, Donnald Fussell, and James C. Browne
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

Abstract

This paper presents a new methodology for specification and control of the motion of an articulated rigid body for the purposes of animation by inverse dynamics. The approach is to formulate the problem as a coordinate transformation from the joint space of the body to a user-defined space which is chosen for convenience in constraining the motion. Constraints are applied to the resulting coordinate transformation equations. It is sufficiently general so that it can be applied to all common types of control problems, including closed loop as well as open loop mechanisms. In order to prove this, we provided a set of simple examples of extended coordinate transformations. Then we also provided a new approach to simulate a closed loop mechanism, which is using extended coordinate transformation technique. The method is formulated in detail and is demonstrated by animating the motion of an inchworm.

1 Introduction

It is by now well known that producing realistic animated motion of natural objects requires modeling the dynamics which determine key aspects of that motion [6, 28, 29]. Controlling very complex motions such as walking animals is a particularly challenging task. Ideally, an animation system allows the animator to directly specify the macroscopic features of the motion such as the intermediate goal configurations of a moving animal. The direct specification of the kinematic features of the walking motion itself, however, is not at all desirable, since it is essentially impossible to achieve realism this way. Instead, a dynamic model of the animal is built which determines these details. It is desirable that such models be specified in terms of parameters which allow effective and intuitive control by the animator of the relevant features of the motion.

Dynamic models of articulated bodies typically have linear or tree-structured topologies. In either case, they are *open loop* mechanisms. That is, they can be modeled as links connected by joints, where the links at the ends of the structure have a free, unconstrained end. However, creatures moving on the ground cannot always be modeled as open loop mechanisms since the free ends of the links are often constrained by the forces involved when they touch the ground. In this case, a *closed-loop* model of the moving creature is needed. Closed-loop models occur in many other situations besides locomotion. A human holding a cup or lifting a barbell also requires a closed-loop model.

Given a model, we still need to determine values for the dynamic parameters such as the appropriate torques on the joints of the body for a forward dynamic simulation which will give proper motion for a particular specified animation. There are two basic approaches to determining these parameters. One is to use heuristic algorithms which generate torque values for each motion increment and then verify that they cause the proper effects. Another is to use inverse dynamics to specify constraints on the dynamic parameters which will cause the specified motion to occur.

Either type of model can be specified in terms of the *generalized coordinate system* or *joint space* of the model itself. This space has a dimension for each degree of freedom (DOF) of each of the joints in the model. The forces exerted at each joint determine the motion of the mechanism in its working space, which is normally Cartesian space with six degrees of freedom in the case of a single open kinematic loop. We can think of the working space as the space of kinematic parameters input to the system, for which we must compute the forces in joint space needed to achieve the specified result using inverse dynamics. These dynamic parameters can be applied using a forward dynamic simulator to generate the resulting motion.

We can specify the relationship between joint space and working space as a coordinate transformation from n-dimensional joint space to 6 dimensional Cartesian space. Since for most articulated creatures n is much larger than 6, a number of unconstrained degrees of freedom are left in the transformation. If in such cases a particular spatial configuration of the articulated body has been specified, there will be a large number of solutions to the transformation equations which produce this result. This redundancy makes motion control very difficult.[21]

The usual approach to this problem is to add constraint functions, and if the system remains underdetermined, to use optimization techniques to find the required joint angle velocities for given end-effector positions and orientations. From this information the joint angle values are obtained by integrating. This approach does not lead to an easy interface for animator specified parameters to be input to the system.

Any minimal set of coordinates to specify a motion is called a set of generalized coordinates[3]; generalized refers to the fact that they need not be only positional or only angular coordinates. By *extended generalized coordinates* we mean that we allow any function, not only position or angle, meaningful for our motion control to be modeled as a coordinate dimension. Our approach to the problem is to replace working space with an arbitrary user-designed *animation space* with up to n dimensions. Typically, this user-designed space will contain working space as a subspace, with additional dimensions defined to represent features of the system which are to be constrained by input from the animation specification. We refer to techniques based on transformations between generalized coordinate systems as *extended generalized coordinate transformation* methods. Based on this generalized coordinate transformation and differential relationships between joint space and animation space, we introduce a dynamic formulation which is based on the General Principle of D'Alembert and virtual work. This approach gives a well-structured matrix (closed) form for the control equations. This formulation is particularly useful in the analysis of dynamic system models. We have used this dynamic formulation for several physical simulations including an open loop serial chain (figure 5), an inchworm (figure 6) and a pin-pointed dropping chain.

The remainder of the paper is organized as follows. In section 2 we review related work and in section 3 we introduce our approach. In section 4 we provide motion equations for animation of linear chain topologies. Section 4.1 contains a new formulation of the geometry of the serial structure of an articulated body. In section 4.2 we describe the extended generalized coordinate transformation technique and in section 4.3 we provide an example of extended coordinate transformation technique and shows that the control works, and in section 4.4 we introduce a dynamic formulation which is easy to use for analysis purposes. In section 4.5 we show three ways to simulate closed loop systems, and in section 5 we describe specific dynamic simulation results.

2 Related Work

Much work has been done on the dynamic simulation of moving creatures or articulated bodies in the past several years. The most common approaches model a real creature as an articulated body consisting of joints connected by links, possibly also including springs and dampers [29, 28]. In [20], a specialized spring-and-damper body for modeling the sliding motions of snakes and worms was developed.

The motions of these models are determined in two basic ways. Forward dynamics based systems provide realistic simulations of the motion of figures but are difficult for an animator to control since the specification of unknown joint torques is required. Lagrangian [6], Gibbs-Appell [27], Armstrong's Newton-Euler [2] and Featherstone's [19] formulations of the dynamics have been used in such systems. Iterative approaches, particularly the latter two, provide the greatest computational efficiency among extant simulation methods.

The basic problem in forward dynamics systems is finding a set of torque functions to control the body. This requires either a heuristic approach or an inverse dynamic (constraint-based) approach. In [16], a system allowing geometric constraints on the joints and specification of either accelerations or torques on the joints for open-loop systems is described. Witkin and Kass [30] give a method which solves geometric constraint equations on the joints along with constraints on the control forces for the entire span of the simulation at once. [31] presents an iterative method to solve inverse kinematics with a given set of constraints, and [18] discusses an iterative method to find a path satisfying kinematic and dynamic constraints.

In [17], a method based on D'Alembert's principle and virtual work which allows inverse dynamic solutions for both open and closed loop systems was developed. An articulated body is defined in terms of a generalized coordinate system with a dimension for each DOF of each joint in the figure. Constraint equations involving one or more of these DOFs can be specified. Lagrangian multipliers for each kinematic constraint equation are used to represent the unknown forces required to satisfy the constraints. Other techniques for handling closed-loop systems have been developed for robotics. Freeman[10, 12] has worked on two dimensional serial chains and Stewart platforms. His approach is to add selected joint positions and angles to the working space to form a new space with the same dimensionality as joint space. A constrained coordinate transformation from generalized coordinates (joint space) to this target space provides the solution to torques required.

3 Our Approach

Our approach is an extension of the generalized coordinate technique [11]. We relax restrictions on *generalized* working space to define an *animation space* which is allowed to have any number of DOF less than or equal to that of joint space and to allow any function of the kinematic parameters to form a dimension in animation space. These newly added dimensions represent kinematic constraints imposed on the dynamic model by the animation system in which it is embedded. Their values are specified as inputs to the dynamic model, which then solves for the torques required to satisfy them. This provides a very natural mechanism for designing the control of a dynamic body model to fit into an animation environment. Our methodology is general enough to handle open and closed loop mechanisms and most other common types of control problems.

We will deal with highly redundant systems. In a redundant system whose degree of freedom is slightly greater than that of animation space, we can usually control motion by adding constraints using Lagrangian multipliers. But because the articulated body is highly redundant, we can not control the *shape* of the body by Lagrangian multipliers alone. We use Lagrangian multipliers in calculating pseudo inverse matrices, but we also extended animation space to actively control the shape of the body. To better understand our approach, consider an example of animation space, U. Let $\vec{u} \in U$,

$$\vec{u} = \begin{pmatrix} x \\ y \\ z \\ (\phi_2 - a)^2 \\ (\phi_4 - b)^2 \end{pmatrix} \tag{1}$$

where a and b are constants, ϕ_i is i-th joint angle and we want to restrict joints 2 and 4 fixed to fixed angles a and b respectively.

4 Motion Control Equations

In this section, we derive motion control equations for our animation and introduce our new formulation. First we formulate the geometry equations for an open-loop serial kinematic chain. Then we present differential equations for the functions used in an animation. We show a specific example of the derivation of geometry equations in sub-section 4.1. The other differential derivations are based on the functions we use in a particular animation. Next we introduce a general dynamic formulation based on the gradient and Hessian tensor in sub-section 4.2. Then in sub-section 4.5, we extend the dynamic equations by transforming the coordinates.

4.1 Geometry of an Open-loop Kinematic Chain

In controlling an articulated body, we frequently need open-loop geometry. We use a formulation for an open-loop chain based on screw theory [4] for simplicity. The notation follows [10, 13, 26].

Define

$$\begin{aligned} Screw[axis, s, \phi] &\equiv Rot[axis, \phi]Trans[axis, s] \\ &\equiv Trans[axis, s]Rot[axis, \phi] \end{aligned} \tag{2}$$

where $Rot[axis, \phi]$ means rotation by ϕ degrees along *axis* and $Trans[axis, s]$ means translation by s along *axis*. Here the sequence of translation and rotation does not matter. The homogeneous transformation matrix between two adjacent link coordinates is given by

$$T^i_{i+1} = Screw[x_i, a_{i,i+1}, \alpha_{i,i+1}]Screw[z_i, s_{i,i+1}, \phi_{i+1}] \tag{3}$$

A homogeneous transformation matrix T for the geometry of the entire mechanism can be defined as [23]

$$T = \begin{pmatrix} \vec{n} & \vec{o} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

where $\vec{a}$ is an approach vector, $\vec{o}$ is an orientation vector, $\vec{n} = \vec{o} \times \vec{a}$ and $\vec{p}$ is a position vector, all in three dimensional space.

We assign joint indices in ascending order from the base of the chain, with the local reference coordinate for each link assigned to its lower indexed joint. Then

$$\vec{a}_i = [T^0_i]_{\vec{n}} \tag{5}$$
$$\vec{s}_i = [T^0_{i-1}Screw[x_{i-1}, 0, \alpha_{i-1,i}]]_{\vec{a}} \tag{6}$$
$$\vec{r}_i = [T^0_i]_{\vec{p}} \tag{7}$$
$$\vec{c}_i = [T^0_i Trans[\vec{a_{i,ci}}, |\vec{a_{i,ci}}|]]_{\vec{p}} \tag{8}$$

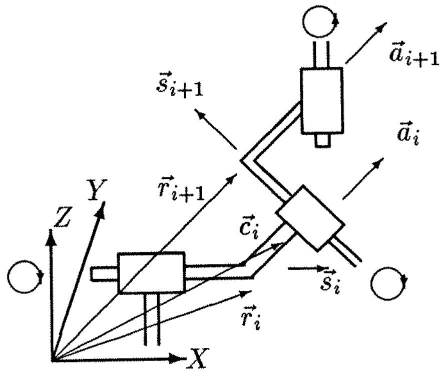
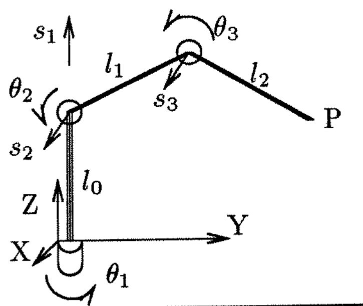


Figure 1: Coordinates of a Chain

where x_i is the x axis of ith local coordinate, $\alpha_{i-1,i}$ is the twist angle between $\vec{s}_{i-1}$ and $\vec{s}_i$, and $|\vec{a_{i,ci}}|$ is the distance from the origin of the ith local coordinate at link i to the mass center of link i. $a_{i,i+1}$ is the offset along x_i from the origin of the i-th coordinate to the origin of the (i+1)-th coordinate, ϕ_i is joint angle along z_i, $s_{i,i+1}$ is offset along z_i from the i-th coordinate to the (i+1)-th coordinate. Note that $a_{i,j}$ is a scalar, $\vec{a}_i$ is a vector, $s_{i,i+1}$ is a scalar, $\vec{s}_i$ is a vector.

The meaning of this equation is straightforward. $\vec{a}_i$ represent the x axis of the ith local coordinate in terms of the base coordinates, $\vec{s}_i$ is the z axis of the ith local coordinate in terms of the base coordinates, and $\vec{r}_i$ is a distance vector from the base coordinate system origin to the origin of the ith local coordinate system. In order to find $\vec{c}_i$, which is a distance vector from the base coordinate system origin to the mass center of the ith link, translating by $|\vec{a}_{i,ci}|$ along $\vec{a}_{i,ci}$. These relationships are illustrated in Figure 1.

From these geometric equations, we can obtain the differential kinematic equations needed to control the articulated body. In order to clarify the above formulation, consider the following example. Figure 2 shows a three degree of freedom kinematic chain and its link parameter table. In



index i, i+1	ϕ_{i+1}	$\alpha_{i,i+1}$	$a_{i,i+1}$	$s_{i,i+1}$
0,1	ϕ_1	0	0	l_0
1,2	ϕ_2	90	0	0
2,3	ϕ_3	0	l_1	0
3,4	0	0	l_2	0

Figure 2: Example of a Serial Chain

the solution, c and s in scalar form mean cosine and sine respectively. So c_{ijk} means $cos(\phi_i+\phi_j+\phi_k)$. s_{ij} means $sin(\phi_i+\phi_j)$.

$$T_1^0 = Screw[x_0, a_{0,1}, \alpha_{0,1}]Screw[z_0, s_{0,1}, \phi_1]$$

$$= \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{aligned} T_2^1 &= Screw[x_1, a_{1,2}, \alpha_{1,2}]Screw[z_1, s_{1,2}, \phi_2] \\ &= \begin{pmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} T_3^2 &= Screw[x_2, a_{2,3}, \alpha_{2,3}]Screw[z_2, s_{2,3}, \phi_3] \\ &= \begin{pmatrix} c_3 & -s_3 & 0 & l_1 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} T_4^3 &= Screw[x_3, a_{3,4}, \alpha_{3,4}]Screw[z_3, s_{3,4}, \phi_4] \\ &= \begin{pmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \vec{a}_1 &= [T_1^0]_{\vec{n}} = \begin{pmatrix} c_1 & s_1 & 0 \end{pmatrix}^T \\ \vec{a}_2 &= [T_2^0]_{\vec{n}} = \begin{pmatrix} c_1c_2 & s_1c_2 & s_2 \end{pmatrix}^T \\ \vec{a}_3 &= [T_3^0]_{\vec{n}} = \begin{pmatrix} c_1c_{23} & s_1c_{23} & s_{23} \end{pmatrix}^T \end{aligned}$$

$$\begin{aligned} \vec{s}_1 &= [T_0^0 Screw[x_0, 0, \alpha_{0,1}]]_{\vec{a}} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \\ \vec{s}_2 &= [T_1^0 Screw[x_1, 0, \alpha_{1,2}]]_{\vec{a}} = \begin{pmatrix} s_1 & -c1 & 0 \end{pmatrix}^T \\ \vec{s}_3 &= [T_2^0 Screw[x_2, 0, \alpha_{2,3}]]_{\vec{a}} = \begin{pmatrix} s_1 & -c1 & 0 \end{pmatrix}^T \\ \vec{s}_4 &= [T_3^0 Screw[x_3, 0, \alpha_{3,4}]]_{\vec{a}} = \begin{pmatrix} s_1 & -c1 & 0 \end{pmatrix}^T \end{aligned}$$

$$\begin{aligned} \vec{r}_1 &= [T_1^0]_{\vec{p}} = \begin{pmatrix} 0 & 0 & l_0 \end{pmatrix}^T \\ \vec{r}_2 &= [T_2^0]_{\vec{p}} = \begin{pmatrix} 0 & 0 & l_0 \end{pmatrix}^T \\ \vec{r}_3 &= [T_3^0]_{\vec{p}} = \begin{pmatrix} l_1c_1c_2 & l_1s_1c_2 & l_1s_2 + l_0 \end{pmatrix}^T \\ \vec{r}_4 &= [T_4^0]_{\vec{p}} = \begin{pmatrix} l_2c_1c_{23} + l_1c_1c_2 \\ l_2s_1c_{23} + l_1s_1c_2 \\ l_2s_{23} + l_1s_2 + l_0 \end{pmatrix}^T \end{aligned}$$

4.2 Coordinate Transformations and Differential Relationships

If we are trying to represent a coordinate in terms of another coordinate, we need to derive a set of equations relating one coordinate to another. In robotics, this usually means transforming from the generalized coordinate space of the joints, where each dimension represents one DOF of a joint, to Cartesian space, which is the normal type of working space. Thus each object can be represented by a set of positions and orientations. In special circumstances, the working space can be spherical or cylindrical [3]. Freeman and Tesar[10, 13] allow the working space to be a generalized coordinate system consisting of Cartesian space positions and orientations and joint angles. This generalized system is required to have the same dimension as joint space.

Usually the coordinate transformation from joint space to Cartesian space is aimed to give output in Cartesian space when the motion control is done in joint space. We may interpret this in reverse, where if we want some output in Cartesian space, we calculate the corresponding effective change in joint space. We extend the idea of generalized coordinate transformations by including *implicit* output functions[11]. By implicit, we mean that the meaning of the function is usually not as obvious as other position or orientation functions. For example if we want to fix joint i, then the function can be $(\phi_i - a)^4$ where a is joint angle(figure 5). Please note that there are many ways of controlling the constraint. We can use any arbitrary function in terms of joint angles and positions and orientations in the ordinary working space to control the motion of the articulated body. That is to say, we create a new *animation* space whose dimensions are defined by these functions. So our animation space can include motion constraint space as well as positions and orientations which are for motion control.

Let $\vec{u}$ denote a vector in the created m dimensional space and $\vec{\phi}$ a vector in joint space having n degrees of freedom. Then

$$\vec{u} = \begin{pmatrix} f_1(\phi_1, \phi_2, \cdots, \phi_n) \\ f_2(\phi_1, \phi_2, \cdots, \phi_n) \\ \vdots \\ f_m(\phi_1, \phi_2, \cdots, \phi_n) \end{pmatrix}. \tag{9}$$

We can find some relationship between $\vec{u}$ and $\vec{\phi}$ such that

$$\dot{\vec{u}} = [G^u_\phi]\dot{\vec{\phi}} \tag{10}$$

where G is a gradient or Jacobian tensor. The tensor $[G^u_\phi]$ consists of

$$[G^u_\phi] = [\frac{\partial \vec{u}}{\partial \phi_1}\frac{\partial \vec{u}}{\partial \phi_2}\cdots\frac{\partial \vec{u}}{\partial \phi_n}] = [g^u_1 g^u_2 \cdots g^u_n]. \tag{11}$$

g^u_i denotes the effect in $\vec{u}$ space of a change in joint i.

Up to now we we have discussed first order differential relationships. But we also frequently need second order differential relationships. An acceleration vector of $\vec{u}$ space, $\ddot{\vec{u}}$, can be expressed in terms of a gradient tensor and Hessian tensor such that

$$\ddot{\vec{u}} = [G^u_\phi]\ddot{\vec{\phi}} + [\dot{G}^u_\phi]\dot{\vec{\phi}} = [G^u_\phi]\ddot{\vec{\phi}} + \dot{\vec{\phi}}^T [H^u_{\phi\phi}]\dot{\vec{\phi}}. \tag{12}$$

Let us assume $\vec{u}$ space is defined by a set of m functions f^i_1, f^i_2, $\cdots$, f^i_m of the ϕ_i. Then the G and H tensors are given by

$$[G^u_\phi] = \begin{pmatrix} \frac{\partial f_1}{\partial \phi_1} & \frac{\partial f_1}{\partial \phi_2} & \cdots & \frac{\partial f_1}{\partial \phi_n} \\ \frac{\partial f_2}{\partial \phi_1} & \frac{\partial f_2}{\partial \phi_2} & \cdots & \frac{\partial f_2}{\partial \phi_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial \phi_1} & \frac{\partial f_m}{\partial \phi_2} & \cdots & \frac{\partial f_m}{\partial \phi_n} \end{pmatrix} \tag{13}$$

and

$$[H^{f_i}_{\phi\phi}] = \begin{pmatrix} \frac{\partial^2 f_i}{\partial\phi_1\partial\phi_1} & \frac{\partial^2 f_i}{\partial\phi_1\partial\phi_2} & \cdots & \frac{\partial^2 f_i}{\partial\phi_1\partial\phi_n} \\ \frac{\partial^2 f_i}{\partial\phi_2\partial\phi_1} & \frac{\partial^2 f_i}{\partial\phi_2\partial\phi_2} & \cdots & \frac{\partial^2 f_i}{\partial\phi_2\partial\phi_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f_i}{\partial\phi_n\partial\phi_1} & \frac{\partial^2 f_i}{\partial\phi_n\partial\phi_2} & \cdots & \frac{\partial^2 f_i}{\partial\phi_n\partial\phi_n} \end{pmatrix} \tag{14}$$

$$\dot{\vec{\phi}}^T [H^u_{\phi\phi}]\dot{\vec{\phi}} = \begin{pmatrix} \dot{\vec{\phi}}^T \left[H^{f_1}_{\phi\phi}\right] \dot{\vec{\phi}} \\ \dot{\vec{\phi}}^T \left[H^{f_2}_{\phi\phi}\right] \dot{\vec{\phi}} \\ \vdots \\ \dot{\vec{\phi}}^T \left[H^{f_m}_{\phi\phi}\right] \dot{\vec{\phi}} \end{pmatrix} \tag{15}$$

Here f can be interpreted to be a very general function. The position and orientation functions of an end-effector are well-known cases of f, which are commonly used in robotics. We can introduce additional functions as long as the total does not exceed the number of degrees of freedom of the articulated body.

The derivation of the differential relations between the joint space and animation space dimensions is based on [10, 13]. These relations describe the effective change of every position (including the base of local coordinates) of a serial kinematic chain as a result of a change in a joint.

Let $\vec{p}$ be an arbitrary position on the jth link of the kinematic chain. Then $[G^p_\phi]_n$ is a differential relation in terms of joint n which can be represented as

$$[G^p_\phi]_n = \begin{cases} \vec{s}_n \times (\vec{p} - \vec{r}_n) & \text{if } n \leq j \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

The Hessian tensor in terms of position $\vec{p}$ can be described as

$$[H^p_{\phi\phi}]_{m;n} = \begin{cases} \vec{s}_m \times (\vec{s}_n \times [\vec{p} - \vec{r}_n]) & \text{if m} \leq \text{n} \leq \text{j} \\ \vec{s}_n \times (\vec{s}_m \times [\vec{p} - \vec{r}_m]) & \text{if n} < \text{m} \leq \text{j} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where m and n denote joint angle indices.

Similarly, orientational differential relations are derived as [10, 13]

$$[G^{jk}_\phi]_n = \begin{cases} \vec{s}_n & \text{if } n \leq j \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$\left[H^{jk}_{\phi\phi}\right]_{i;n} = \begin{cases} \vec{s}_i \times \vec{s}_n & \text{if } i < n \leq j \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

The gradient tensor $[G^{jk}_\phi]$ contains orientational differential relations between every pair of joints in the articulated body. The gradient and Hessian tensors of the other functions, which we use in our motion control, can be obtained in a similar way.

Now we have both direct and differential relationships between $\vec{u}$ space and $\vec{\phi}$ space. For example, geometry (position and orientation) is a common direct relationship. For control, we can assign some values to the vectors $\dot{\vec{u}}$ and $\ddot{\vec{u}}$ in the newly created space. By the constraints thus imposed on $\dot{\vec{u}}$ and $\ddot{\vec{u}}$, the articulated body is constrained in motion. The simplest case is assigning the value zero. If we assign zero to the velocity and acceleration functions of the end-effector position, we convert an open loop chain mechanism to a closed loop.

Usually the dimension of $\vec{u}$ will be smaller than that of joint space. In this case, we need to apply optimization techniques, as is commonly done in robotics [3, 21]. In our simulation, we used a joint velocity minimization technique.

4.3 Example and Basic Concept of Transformations and Differential Relationships

In this section we work on a three equal segment(length was set to 1) planar linkage. This section is related with the next sections that the readers may need to refer next sections while reading. We set DOF of animation space equal to that of joint space in order to avoid non square matrices. But Lagrangian multiplier method or Singular Value Decomposition method is necessary when we have non square matrices. Our example animation space and joint space was set as follows.

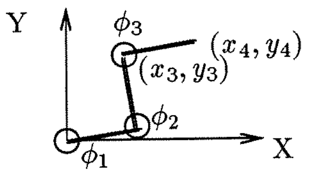


Figure 3: Example of a Serial Chain

$$\begin{aligned}
\vec{u} &= \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} x_4 \\ y_4 \\ \phi_3^2 \end{pmatrix} \\
\vec{\phi} &= \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix}
\end{aligned}$$

Let us work on a general function, f_i, $(i = 1, 2, 3)$, for animation space.

$$\begin{aligned}
\dot{f}_i &= \frac{\partial f_i}{\partial \phi_1}\frac{d\phi_1}{dt} + \frac{\partial f_i}{\partial \phi_2}\frac{d\phi_2}{dt} + \frac{\partial f_i}{\partial \phi_3}\frac{d\phi_3}{dt} \\
&= g_1^{f_i}\dot{\phi}_1 + g_2^{f_i}\dot{\phi}_2 + g_3^{f_i}\dot{\phi}_3 \qquad (20) \\
\dot{\vec{u}} &= \begin{pmatrix} g_1^{f_1} \\ g_1^{f_2} \\ g_1^{f_3} \end{pmatrix}\dot{\phi}_1 + \begin{pmatrix} g_2^{f_1} \\ g_2^{f_2} \\ g_2^{f_3} \end{pmatrix}\dot{\phi}_2 + \begin{pmatrix} g_3^{f_1} \\ g_3^{f_2} \\ g_3^{f_3} \end{pmatrix}\dot{\phi}_3 \\
&= g_1^u\dot{\phi}_1 + g_2^u\dot{\phi}_2 + g_3^u\dot{\phi}_3 \\
&= [G_\phi^u]\dot{\vec{\phi}} \qquad (21)
\end{aligned}$$

From equation (20),

$$\begin{aligned}
\ddot{f}_i &= g_1^{fi}\ddot{\phi}_1 + g_2^{fi}\ddot{\phi}_2 + g_3^{fi}\ddot{\phi}_3 + \dot{g}_1^{fi}\dot{\phi}_1 + \dot{g}_2^{fi}\dot{\phi}_2 + \dot{g}_3^{fi}\dot{\phi}_3 \\
\dot{g}_j^{fi} &= \frac{\partial g_j^{f_i}}{\partial \phi_1}\frac{d\phi_1}{dt} + \frac{\partial g_j^{f_i}}{\partial \phi_2}\frac{d\phi_2}{dt} + \frac{\partial g_j^{f_i}}{\partial \phi_3}\frac{d\phi_3}{dt}
\end{aligned}$$

$$
\begin{aligned}
&= h^{f_i}_{j1}\dot\phi_1 + h^{f_i}_{j2}\dot\phi_2 + h^{f_i}_{j3}\dot\phi_3 \\
\ddot{f}^i &= [g^{f_i}_1 g^{f_i}_2 g^{f_i}_3]\ddot{\vec\phi} + \dot{\vec\phi}^T \begin{pmatrix} h^{f_i}_{11} & h^{f_i}_{12} & h^{f_i}_{13} \\ h^{f_i}_{21} & h^{f_i}_{22} & h^{f_i}_{23} \\ h^{f_i}_{31} & h^{f_i}_{32} & h^{f_i}_{33} \end{pmatrix} \dot{\vec\phi} \\
\ddot{\vec u} &= [g^{f_i}_1 g^{f_i}_2 g^{f_i}_3]\ddot{\vec\phi} + \begin{pmatrix} \dot{\vec\phi}^T \left[H^{f_1}_{\phi\phi}\right] \dot{\vec\phi} \\ \dot{\vec\phi}^T \left[H^{f_2}_{\phi\phi}\right] \dot{\vec\phi} \\ \dot{\vec\phi}^T \left[H^{f_3}_{\phi\phi}\right] \dot{\vec\phi} \end{pmatrix} \\
&= [G^u_\phi]\ddot{\vec\phi} + \dot{\vec\phi}^T \left[H^u_{\phi\phi}\right] \dot{\vec\phi} \qquad (22)
\end{aligned}
$$

If we solve inverse kinematics, from equation (21) and (22)

$$
\begin{aligned}
\dot{\vec\phi} &= [G^u_\phi]^{-1}\dot{\vec u} = [G^\phi_u]\dot{\vec u} \\
\ddot{\vec\phi} &= [G^u_\phi]^{-1}\left(\ddot{\vec u} - \dot{\vec\phi}^T [H^u_{\phi\phi}]\dot{\vec\phi}\right) \\
&= [G^u_\phi]^{-1}\ddot{\vec u} - \dot{\vec\phi}^T([G^u_\phi]^{-1} \otimes [H^u_{\phi\phi}])\dot{\vec\phi} \qquad (23)
\end{aligned}
$$

where $\otimes$ is explained later. From Figure 3, we can easily check

$$
\begin{aligned}
x_4 &= c_1 + c_{12} + c_{123} \\
y_4 &= s_1 + s_{12} + s_{123} \\
g^{x_4}_1 &= -s_1 - s_{12} - s_{123} \\
g^{x_4}_2 &= -s_{12} - s_{123} \\
g^{x_4}_3 &= -s_{123} \\
g^{y_4}_1 &= c_1 + c_{12} + c_{123} \\
g^{y_4}_2 &= c_{12} + c_{123} \\
g^{y_4}_3 &= c_{123} \\
g^{\phi_3^2}_1 &= g^{\phi_3^2}_2 = 0 \\
g^{\phi_3^2}_3 &= 2\phi_3 \\
h^{x_4}_{11} &= -c_1 - c_{12} - c_{123} \\
h^{x_4}_{21} &= -c_{12} - c_{123} \\
h^{x_4}_{31} &= -c_{123} \\
h^{x_4}_{12} &= -c_{12} - c_{123} \\
h^{x_4}_{22} &= -c_{12} - c_{123} \\
h^{x_4}_{32} &= -c_{123} \\
h^{x_4}_{13} &= -c_{123} \\
h^{x_4}_{23} &= -c_{123} \\
h^{x_4}_{33} &= -c_{123} \\
h^{y_4}_{11} &= -s_1 - s_{12} - s_{123} \\
h^{y_4}_{21} &= -s_{12} - s_{123}
\end{aligned}
$$

$$\begin{aligned}
h_{31}^{y_4} &= -s_{123} \\
h_{12}^{y_4} &= -s_{12} - s_{123} \\
h_{22}^{y_4} &= -s_{12} - s_{123} \\
h_{32}^{y_4} &= -s_{123} \\
h_{13}^{y_4} &= -s_{123} \\
h_{23}^{y_4} &= -s_{123} \\
h_{33}^{y_4} &= -s_{123} \\
h_{11}^{\phi_3^2} &= h_{21}^{\phi_3^2} = h_{31}^{\phi_3^2} = h_{12}^{\phi_3^2} = 0 \\
h_{22}^{\phi_3^2} &= h_{32}^{\phi_3^2} = h_{13}^{\phi_3^2} = h_{23}^{\phi_3^2} = 0 \\
h_{33}^{\phi_3^2} &= 2
\end{aligned}$$

$$[G_\phi^u]^{-1}\dot{\vec{\phi}}^T[H_{\phi\phi}^u]\dot{\vec{\phi}} = \dot{\vec{\phi}}^T\left([G_\phi^u]^{-1}\otimes[H_{\phi\phi}^u]\right)\dot{\vec{\phi}}$$

$$= \begin{pmatrix} \dot{\vec{\phi}}^T\left[a_{11}[H_{\phi\phi}^{f_1}] + a_{12}[H_{\phi\phi}^{f_2}] + a_{13}[H_{\phi\phi}^{f_3}]\dot{\vec{\phi}}\right] \\ \dot{\vec{\phi}}^T\left[a_{21}[H_{\phi\phi}^{f_1}] + a_{22}[H_{\phi\phi}^{f_2}] + a_{23}[H_{\phi\phi}^{f_3}]\dot{\vec{\phi}}\right] \\ \dot{\vec{\phi}}^T\left[a_{31}[H_{\phi\phi}^{f_1}] + a_{32}[H_{\phi\phi}^{f_2}] + a_{33}[H_{\phi\phi}^{f_3}]\dot{\vec{\phi}}\right] \end{pmatrix}$$

$$\text{where}[G_\phi^u]^{-1} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\begin{aligned}
[G_\phi^u] &= \begin{pmatrix} -s_1 - s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \\ c_1 + c_{12} + c_{123} & c_{12} + c_{123} & c_{123} \\ 0 & 0 & 2\phi_3 \end{pmatrix} \\
\left[H^{f_1}\right] &= \begin{pmatrix} -c_1 - c_{12} - c_{123} & -c_{12} - c_{123} & -c_{123} \\ -c_{12} - c_{123} & -c_{12} - c_{123} & -c_{123} \\ -c_{123} & -c_{123} & -c_{123} \end{pmatrix} \\
\left[H^{f_2}\right] &= \begin{pmatrix} -s_1 - s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \\ -s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \\ -s_{123} & -s_{123} & -s_{123} \end{pmatrix} \\
\left[H^{f_3}\right] &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
\left[G_\phi^u\right]^{-1} &= \frac{M}{D} \\
D &= -2\phi_3 c_{12} s_1 - 2\phi_3 c_{123} s_1 \\
M &=
\end{aligned}$$

$$\begin{pmatrix} 2\phi_3(c_{12} + c_{123}) & 2\phi_3(s_{12} + s_{123}) & s_3 \\ -2\phi_3(c_1 + c_{12} + c_{123}) & -2\phi_3(s_1 + s_{12} + s_{123}) & -s_{23} - s_3 \\ 0 & 0 & s_2 + s_{23} \end{pmatrix}$$

Let

$$\dot{\vec{u}} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$$

then

$$\begin{aligned}
\dot{\vec{\phi}} &= [G^u_\phi]^{-1}\dot{\vec{u}} \\
&= \frac{1}{D}\begin{bmatrix} 2\phi_3(c_{12}+c_{123}+s_{12}+s_{123}) \\ -2\phi_3(c_1+c_{12}+c_{123}+s_1+s_{12}+s_{123}) \\ 0 \end{bmatrix} \\
\dot{\phi}_3 &= \mathbf{0}
\end{aligned}$$

If

$$\begin{aligned}
\dot{\vec{\phi}} &= \begin{bmatrix} 1 & 10 & 0 \end{bmatrix}^T \\
\ddot{\vec{u}} &= \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T
\end{aligned}$$

then by equation (23)

$$\begin{aligned}
\ddot{\phi}_1 &= \frac{-121 - 121cos(\phi_3) - cos(\frac{\phi_3}{2})cos(\phi_2 + \frac{\phi_3,2}{)}}{Dcos(\frac{\phi_3}{2})sin(\phi_2 + \frac{\phi_3}{2})} \\
\ddot{\phi}_2 &= \frac{243 + 242cos(\phi_3) + 244cos(\frac{\phi_3}{2})cos(\phi_2 + \frac{\phi_3}{2})}{2Dcos(\frac{\phi_3}{2})sin(\phi_2 + \frac{\phi_3}{2})} \\
\ddot{\phi}_3 &= \mathbf{0}
\end{aligned}$$

As you can see, we can control $\dot{\vec{u}}$ and $\ddot{\vec{u}}$ to fix ϕ_3 instead of directly controlling joint space.

Now we have both direct and differential relationships between $\vec{u}$ space and $\vec{\phi}$ space. For example, geometry (position and orientation) is a common direct relationship. For control, we can assign some values to the vectors $\dot{\vec{u}}$ and $\ddot{\vec{u}}$ in the newly created space. By the constraints thus imposed on $\dot{\vec{u}}$ and $\ddot{\vec{u}}$, the articulated body is constrained in motion. If we assign zero to the velocity and acceleration functions of the end-effector position, we convert an open loop chain mechanism to a closed loop.

Usually the dimension of $\vec{u}$ will be smaller than that of joint space. In this case, we need to apply local optimization techniques, as is commonly done in robotics [3, 21]. In our simulation, we used joint velocity minimization and joint torque minimization technique.

4.4 Basic Dynamic Equations

Using the gradient and Hessian tensors we have derived, we can formulate dynamics equations for a dynamic animation. This section is heavily due to [10, 13, 26]. The dynamic equation is based on the Generalized Principle of D'Alembert and virtual work, that is to say, the derivation of the dynamic equation is based on the virtual work of inertial loads. We can use other techniques based on inertial power or Lagrangian methods to derive exactly the same set of equations. Please be ware that the dynamic equation is based on three dimensional space. In order to derive dynamic equations we need to derive kinematic differential equations in terms of positions and orientations. If we have external forces/torques, then we also have to compute differential relations for each corresponding positions. All necessary terms are included in a giant dynamic equation (27).

In order to provide a better feel for our large system of dynamics equations, we provide a very loose derivation of a simplified equation. Let τ_i^I be the i-th joint torque, F_l be the force acting on the l-th link centroid, τ_l the moments acting on the l-th link, M_l the mass of link l, x the position of the l-th link centroid, I_l the moments of inertia of the l-th link and ψ_l the angle of the l-th link. Then by the Generalized Principle of D'Alembert,

$$\sum_i \tau_i^I \delta\phi_i + \sum_{l=1}^{L} [F_l \delta x_l + \tau_l \delta\psi_l] = 0 \tag{24}$$

$$\begin{aligned} F_l &= -M_l \ddot{x}_l \\ \tau_l &= -I_l \ddot{\psi}_l \\ \delta x_l &= \sum_i g_{li}^x \delta\psi_i \\ \delta\psi_l &= \sum_i g_{li}^\psi \delta\psi_i \\ \ddot{x}_l &= \sum_i [g_{li}^x \ddot{\phi}_i + h_{li}^x \dot{\phi}_i^2] \\ \ddot{\psi}_l &= \sum_i [g_{li}^\psi \ddot{\phi}_i + h_{li}^\psi \dot{\phi}_i^2] \end{aligned}$$

Substituting these equations and eliminating $\delta\phi$, we get an equation of the form

$$\tau_i^I = I_i^* \ddot{\phi} + P_i^* \dot{\phi}^2 \tag{25}$$

Proceeding in this fashion, we obtain the following complete set of equations.

$$\begin{aligned} \vec{\tau}_\phi^I = \; & /\text{*inertial\_force*}/ \\ & [I_{\phi\phi}^*]\ddot{\vec{\phi}} \\ & /\text{*coriolis and centripetal force*}/ \\ & +\dot{\vec{\phi}}^T [P_{\phi\phi\phi}^*]\dot{\vec{\phi}} \\ & /\text{*external forces/moments*}/ \\ & -\sum_{j=1}^{n}([G_\phi^{p_j}]^T \vec{f}^{p_j} + [G_\phi^{jk}]^T \vec{m}^{jk}) \\ & /\text{*gravitational forces*}/ \\ & -\sum_{i=1}^{n}[G_\phi^{c_i}]^T \vec{f}_g^{c_i} \\ & /\text{*damping*}/ \\ & -\sum_{r=1}^{n} c_r [G_\phi^{c_r}]^T [G_\phi^{c_r}] \dot{\vec{f}}^{c_i} \\ & /\text{*spring load*}/ \\ & -\sum_{q=1}^{n}\{k_q [G_\phi^{c_q}]^T [G_\phi^{c_q}] \\ & +((\tau^u)^T \otimes [H_{\phi\phi}^q]^T\} \dot{\vec{f}}^{c_i} \end{aligned} \tag{26}$$

where

$$\begin{aligned}
\left[I^*_{\phi\phi}\right] &= \sum_{j=1}^{n}\{mass_j[G^{c_j}_{\phi}]^T[G^{c_j}_{\phi}] \\
&\quad +[G^{jk}_{\phi}]^T[II^j][G^{jk}_{\phi}]\} && (27) \\
\left[II^j\right] &= [T^j][I^j][T^j]^T && (28) \\
\left[P^*_{\phi\phi\phi}\right] &= \sum_{j=1}^{n}\{mass_j([G^{c_j}_{\phi}]^T \otimes [H^{c_j}_{\phi\phi}]) \\
&\quad +([G^{j}_{\phi}]^T[II^j] \otimes [H^{j}_{\phi\phi}]) \\
&\quad +([G^{j}_{\phi}]^T[II^j]([G^{j}_{\phi}]^T \times [G^{c_j}_{\phi}]))\} && (29)
\end{aligned}$$

and $[I^j]$ is an inertia tensor in local coordinates.

This is the inverse dynamic equation. We can easily obtain the forward dynamic equation from it such that

$$\begin{aligned}
\ddot{\vec{\phi}} &= [I^*_{\phi\phi}]^{-1}(\vec{\tau}^I_{\phi} - \dot{\vec{\phi}}^T[P^*_{\phi\phi\phi}]\dot{\vec{\phi}} \\
&\quad +\text{external force} + \text{gravitational force} \\
&\quad +\text{damping force} + \text{spring force}). && (30)
\end{aligned}$$

The multiplication $a \otimes b$ of two tensors a and b of dimension $n \times m$ and $n \times n \times m$, respectively results in tensor c of dimension $n \times n \times n$ and is performed as follows.

for i , j, k from 1 to n

$$c_{jki} = \sum_{l=1}^{m} a_{il} b_{jkl}$$

Basically $\otimes$ has the power of normal matrix multiplication in a two dimensional array. It is a bit different from the tensor multiplication operator in [10, 13] due to the indexing used in our formulation, but has the same power. For more detail about the dynamics and a further explanation of the notation used here, see [10, 13, 26] .

4.5 Closed Loop Mechanisms

There are three methods for simulating the closed loop mechanism : soft constraints , hard constraints and simulation by extended coordinate transformation technique.

4.5.1 Simulation by the Extended Generalized Coordinate Transformation Technique

By this method closed loop mechanisms are easily handled by including corresponding constraint equations at animation space and by doing motion control appropriately as was shown at previous section. Consider Figure 3. Please notice that this model is the same with a human sitting on a chair. If we want to simulate the human, then we need to make an animation space. Because this example is highly constrained we need to be careful. If there are lots of redundancy, it may be easier to decide an animation space. Let us make animation space and its control as follows.

$$\vec{u} = \begin{pmatrix} x_3 \\ y_3 \\ \phi_3 \end{pmatrix}$$

$$\dot{\vec{u}} = \begin{pmatrix} 0 \\ 0 \\ \dot{\phi}_3 \end{pmatrix} \ddot{\vec{u}} = \begin{pmatrix} 0 \\ 0 \\ \ddot{\phi}_3 \end{pmatrix} \tag{31}$$

so that we can constrain $\dot{x_3} = \ddot{x_3} = \dot{y_3} = \ddot{y_3} = 0$ to fix (x_3, y_3). By differential kinematic equations derived so far, we can calculate corresponding joint velocity and acceleration values. Then with this, we do inverse dynamic simulation. We still need the simulation if there are external forces acting on a body except the forces caused by hip constraints. By this way, we can simulate human sitting on a chair. Because this example is really simple, the animation becomes straight forward.

4.5.2 Soft Constraints Technique

Soft constraints method models external forces to constraint the motion by using stiff springs and dampers. Consider Figure 3. In order to fix a hip of a human body which corresponds to (x_3, y_3), we need to use vertical and horizontal direction stiff spring and dampers. Dampers should work opposite direction than that of springs to restrain wild action. We do forward dynamic simulation in terms of time by including external forces caused by springs and dampers. During simulation, the hip is moving a very small amount which causes springs and dampers reaction forces. Due to the reaction forces, the hip moves to opposite direction. But by the small amount of displacement, the hip get opposite direction forces due to springs and dampers. But this approach is only effective for forward dynamic simulation (so can not be applied in this paper approach) and is very expensive in terms of computation time because we need to make the integration time step very small especially wild motion is involved. For some tip, even this simple simulation is not easy because we need to carefully select spring and damper coefficients, and integration time step, and the most difficult part is the control of the body. We used a weak variant of an optimal control method to get a best set of controls for the human movement. Also a good model of human foot is very important if the simulation(animation) includes human jumping or walking.

4.5.3 Hard Constraints Technique

In this method, we use kinematic constraints to derive the resulting constrained closed loop mechanism. In terms of this approach the motion equation usually becomes messy. Our dynamic model carrys enough information that we can model hard constraints in a closed form equation and is shown in this sub-section. Other dynamic formulation, for example recursive algorithms[9], usually do not compute enough differential kinematic relations and can not include hard constraints straight forwardly because their equation is not in a closed form. We have to derive kinematic constraints for a set of joint angles and then has to simplify corresponding dynamic equation.

Consider the closed loop mechanism(Figure 3) where the hip of an open loop mechanism is fixed on the chair. Then the velocity and acceleration of the hip are both zero. If we want to derive dynamic equation using the recursive algorithm[9] then we need to derive differential kinematic equation for hip position. Then as a result, we know velocity and acceleration of joints 1 and 2 have some relation(zero). By applying these kinematic equations to recursive dynamic equation, we get a new dynamic equation for closed loop mechanism. Joint torque 1 and 2 means reaction forces acting at an ankle and knee in this case. But you can see that this process is pretty messy if DOF of a mechanism becomes higher and higher. But the method presented in this paper is pretty neat.

Let us make an animation space for the closed loop mechanism the same with subsection 4.5.1. in order to include hip constraints. We then derive differential kinematic equations $[G^u_\phi]$ and $[H^u_{\phi\phi}]$

and derive the dynamic equation in terms of animation space. Then we get $\vec{\tau}^I_u$. The meaning of $\tau^I_{x_3}$ and $\tau^I_{y_3}$ becomes hip reaction forces in horizontal and vertical direction, and $\tau^I_{\phi_3}$ is the same with $\tau^I_{\phi_3}$ which is derived in joint space $\vec{\phi}$. Then we discard $\tau^I_{x_3}$ and $\tau^I_{y_3}$, and use only $\tau^I_{\phi_3}$ for forward dynamic simulation. Please note that we are very flexible in selecting an animation space. We can include any function if meaningful for our animation and use it to constrain the motion. The *constraint* do not mean fixed constraint in geometry. It can be any thing.

Now we shall work in general case. Basically the open loop case and closed loop case can be handled in the same way in terms of our extended generalized coordinate transformation technique. The only difference is the constraints assigned as inputs to the system. If the dimensions of $\vec{\phi}$ and $\vec{u}$ are the same, the problem is easy. Usually the dimension of $\vec{u}$ is smaller than that of $\vec{\phi}$, so we need to use optimization techniques [3, 8, 21] and need to use different approaches to solve the closed loop equation. We have tried several optimization techniques, including joint velocity optimization with weights and weighted joint torque optimization. Among these weighted joint velocity optimization gave the best results in our simulations. Also we can easily adjust joint weights to modify the resulting motion.

Let $f : \Phi \longrightarrow U$ and $\vec{u} \in U$, $\vec{\phi} \in \Phi$.

$$\vec{u} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{pmatrix} \tag{32}$$

$$\vec{\phi} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{pmatrix} \tag{33}$$

Then $[G^u_\phi]$ and $[H^u_{\phi\phi}]$ are derived from the differential relationships. If f_i is a linear combination of ϕ_i in $\vec{\phi}$, then $[H^{f_i}_{\phi\phi}]$ becomes zero.

In order to solve in terms of $\vec{\phi}$, we need to decide redundant and nonredundant joint sets. Then we solve equations in terms of non-redundant joints by eliminating redundant joints. We then solve for the unknown endeffector forces and eliminate the forces from the equation. Thus we can do forward dynamic simulation in terms of nonredundant joints. The angular values of redundant joints can be calculated from nonredundant joints. If we want to derive the equations in terms of $\vec{u}$, the relationship between $\vec{\phi}$ and $\vec{u}$ becomes as follows. From

$$\begin{aligned} Kinetic\_Energy_{\vec{u}} &= Kinetic\_Energy_{\vec{\phi}} \\ \frac{1}{2}\dot{\vec{\phi}}^T[I^*_{\phi\phi}]\dot{\vec{\phi}} &= \frac{1}{2}\dot{\vec{u}}^T[I^*_{uu}]\dot{\vec{u}} \\ \dot{\vec{\phi}} &= [G^\phi_u]\dot{\vec{u}} \\ \ddot{\vec{\phi}} &= [G^\phi_u]\ddot{\vec{u}} + \dot{\vec{u}}^T[H^\phi_{uu}]\dot{\vec{u}} \\ \dot{\vec{\tau}}^I_\phi &= [G^u_\phi]^T\dot{\vec{\tau}}^I_u \\ \dot{\vec{\tau}}^I_u &= [G^\phi_u]^T\dot{\vec{\tau}}^I_\phi \end{aligned}$$

We get

$$\begin{aligned}
[I^*_{uu}] &= [G^\phi_u]^T[I^*_{\phi\phi}][G^\phi_u] && (34)\\
\vec{\tau}^I_u &= [I^*_{uu}]\ddot{\vec{u}} + \dot{\vec{u}}^T[P^*_{uuu}]\dot{\vec{u}} - [G^\phi_u]^T\vec{\tau}_L && (35)
\end{aligned}$$

where

$$\begin{aligned}
\left[G^\phi_u\right] &= \left[G^u_\phi\right]^{-1} && (36)\\
\left[H^\phi_{uu}\right] &= -\left[G^\phi_u\right]^T\left(\left[G^\phi_u\right]\otimes\left[H^u_{\phi\phi}\right]\right)\left[G^\phi_u\right] && (37)\\
[P^*_{uuu}] &= \left[G^\phi_u\right]^T\{(\left[G^\phi_u\right]^T\left[P^*_{\phi\phi\phi}\right])\\
&\quad -([I^*_{uu}]\otimes\left[H^u_{\phi\phi}\right])\}\left[G^\phi_u\right] && (38)
\end{aligned}$$

and $\vec{\tau}_L$ includes load terms which are similar to that of equation (19). More specifically, the equation

$$\tau^I_u = [I^*_{uu}]\ddot{\vec{u}} + \dot{\vec{u}}^T[P^*_{uuu}]\dot{\vec{u}} \tag{39}$$

is of the form

$$\begin{pmatrix} \tau^I_{f_1}\\ \tau^I_{f_2}\\ \vdots\\ \tau^I_{f_m}\end{pmatrix} = \begin{pmatrix} I^*_{f_1f_1} & I^*_{f_1f_2} & \cdots & I^*_{f_1f_m}\\ I^*_{f_2f_1} & I^*_{f_2f_2} & \cdots & I^*_{f_2f_m}\\ \vdots & \vdots & \vdots & \vdots\\ I^*_{f_mf_1} & I^*_{f_mf_2} & \cdots & I^*_{f_mf_m}\end{pmatrix}\begin{pmatrix}\ddot{f}_1\\ \ddot{f}_2\\ \vdots\\ \ddot{f}_m\end{pmatrix} +$$
$$\left(\begin{array}{c} \begin{pmatrix}\dot{f}_1\\ \dot{f}_2\\ \vdots\\ \dot{f}_m\end{pmatrix}^T \begin{pmatrix} P^*_{f_1f_1f_1} & P^*_{f_1f_1f_2} & \cdots & P^*_{f_1f_1f_m}\\ P^*_{f_1f_2f_1} & P^*_{f_1f_2f_2} & \cdots & P^*_{f_1f_2f_m}\\ \vdots & \vdots & \vdots & \vdots\\ P^*_{f_1f_mf_1} & P^*_{f_1f_mf_2} & \cdots & P^*_{f_1f_mf_m}\end{pmatrix}\begin{pmatrix}\dot{f}_1\\ \dot{f}_2\\ \vdots\\ \dot{f}_m\end{pmatrix}\\ \dot{\vec{u}}^T[P^*_{f_2uu}]\dot{\vec{u}}\\ \vdots\\ \dot{\vec{u}}^T[P^*_{f_muu}]\dot{\vec{u}}\end{array}\right) \tag{40}$$

If f_i is a position-related term, and that position is fixed in the closed loop case, then we need to set $\dot{f}_i = \ddot{f}_i = 0$. The corresponding $\tau^I_{f_i}$ term is computed to be a reaction force (torque) at that position. If f_i is just ϕ_i, then the $\tau^I_{f_i}$ can be the torques at the desired input. But if f is an arbitrary function, it is not easy to get a physical meaning.

4.5.4 Mechanism used for Inchworm Animation

Now we will introduce the specific mechanism used in inchworm modeling. The mechanism is very important because some redundant mechanisms produce many singular configurations. We used two DOF joints at each links because these joints give fewer singular configurations than the three degree of freedom ball joints which were used in [14]. The mechanism used for modeling an inchworm is shown in Figure 4. The inchworm has a very flexible body with a great many degrees of freedom. We map the inchworm body to a finite segmented articulated body. Before creating a $\vec{u}$ space, we need to check the degrees of freedom of the mechanism so that we may decide the dimension for our space. The open loop case is pretty easy, and the closed loop case is determined by Grübler's equation [15]. Let us assume one DOF at each joint, which means we assume there are two joints at each segment. Then the DOF for the closed loop is determined by the equation

$$closed\_loop\_d.o.f = 6(n-1) - 5j \tag{41}$$

where n is the number of links and j is the number of one DOF links[15]. From this, we can decide the upper limit on the dimension of $\vec{u}$ space.

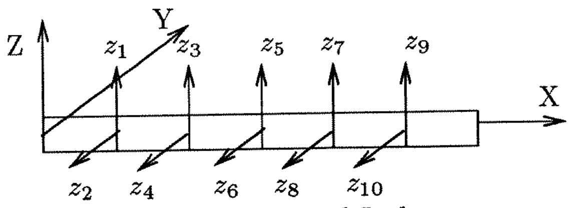


Figure 4: Segmented Inchworm

5 Dynamic Simulation

Using the motion equations developed above, we have done a dynamic simulation of an open loop serial chain, an inchworm and a pin-pointed dropping chain. The pin-pointed dropping chain is a famous simple example which we used to check the correct operation of our system. For inverse dynamic simulations, we used extended coordinate transformation method to get differential kinematic equations for the control of the motion.

The inchworm example given in Section 4.2 illustrates the basic steps of our method of motion control. These steps are:

1. Define a model of the object to be controlled for animation.

2. Formulate the equations for motion control in the extended general coordinate transformation basis.

3. Solve the inverse dynamics problem to obtain torque functions.

4. Execute the forward dynamics simulation of motion, both open loop and closed loop steps, using the information contained in the torque functions. Please note that we still need forward dynamic simulation because we need to include external forces/torques during simulation.

The core of dynamic simulation is as follows:

a. Calculate kinematic information with current $\vec{\phi}$.

b. Determine $\dot{\vec{u}}$ and $\ddot{\vec{u}}$.

c. Calculate $\dot{\vec{\phi}}$, $\ddot{\vec{\phi}}$ using $\dot{\vec{u}}$, $\ddot{\vec{u}}$.

d. Calculate torque set by inverse dynamics.

e. Do forward dynamic simulation.

f. Obtain a new $\vec{\phi}$, and go to step a.

5.1 Dynamic Simulation of Open Loop Constrained Mechanism

A dynamic simulation on a simple open loop mechanism is shown in Figure 5.

The mechanism has ten DOF and the control is only position based. So we have seven redundant DOF in this example. The redundancy is solved, as was discussed previously, with a joint velocity optimization technique used in robotics.[3, 8, 21] This optimization technique uses a Lagrange multipliers, which adds artificial constraints by reducing the null space. But we stress that this technique alone cannot control the shape of the body. This is why we use extended generalized coordinate transformations for our animation. The left simulation is normal movement, but the right simulation is constrained such that the first segment is fixed. As can be seen from Figure 5, the segment is completely disabled. But when a position is given which cannot be reached with the constrained mechanism, the fixed segment eventually moves to reach it. This is a kind of inverse kinematic problem which was given unreachable position as an input. This is a very simple control example of our extended generalized coordinate transformation technique. We set $f_i = \phi_2^2$ and controlled by $\dot{f}_i = \ddot{f}_i = 0$.

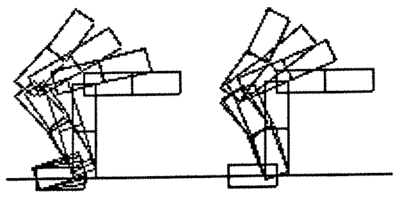


Figure 5: Normal and Constrained Motion

5.2 Dynamic Simulation of an Inchworm

The main purpose of using inverse dynamics is to get a set of torque functions which can be used for forward dynamic simulation. An inchworm has legs only at front and rear. It lacks middle pairs of legs, so it moves by extending the front part of the body while the rear grasps the ground, and then pulling the rear forward while the front grasps the ground[1, 5]. When an inchworm extends the front part, it can be modeled as an open loop mechanism. When it pulls the rear forward, it can be modeled as a closed loop mechanism. In the case of the closed loop, there is only a single direction sliding motion at the rear part of the worm.

The inchworm model controls its own motion, subject to the intermediate goal position constraints provided by the animator. Motion planning is done by the worm to find the best body positions, favorable torques etc. For this we used a heuristic which checks a finite number of potential directions to move at each moment and selects the best move among them. (The easiest way to animate inchworm is let end of an inchworm to reach a point on a ground. If there are any external forces, the shape may be affected. But because it can be simply asked to reach the goal position, the movement continues. In this process, we can calculate slope($\dot{\vec{u}}$) of displacement from current position to the goal position. $\ddot{\vec{u}}$ can be calculated by finite difference. In terms of pulling rear part of the inchworm, we may constrain it to stay on the ground.) This step needs to be done only once to get a set of torque functions. The forward simulation based on this set of torque functions proceeds quite rapidly. In forward simulation we interpolate the torque values obtained from the inverse dynamics.

Because we simulate an articulated body with high DOF, it is possible to fall into near singular configurations. We tried several methods for handling this including singular value decomposition(SVD)[25] and the damping technique [7]. Some optimization schemes cannot be used with the singular value decomposition method, so we incorporated damping factors in calculating the pseudo-inverse Jacobian matrix. The damping approach involves adding some values at places needed for calculating the inverse. These adjustments can cause small errors, which we have not found to be significant. The damping technique has proven quite effective in handling singular configurations.

To change steps, we need to change the base of the worm from one end to the other. We need only two DOF at each segment for motion, but the position of a segment is specified with three DOF. Since there is a segment length constraint between joints, we can change steps with only two degrees of freedom. Figure 6 shows several frames from the dynamic simulation of an inchworm. A videotape of an "inchworm race" has been produced to better illustrate this simulation. In the simulation, we used two additional constraint functions to control the shape of the body. These

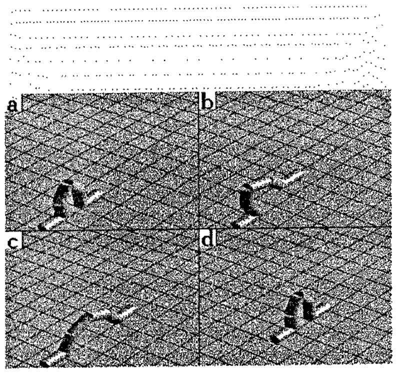


Figure 6: Dynamic Simulation of an Inchworm

functions are of the form $f(\vec{\phi}) = (\phi_i - \alpha)^2$ where α is a function of time.

For numerical integration, we tried Runge-Kutta fourth order adaptive size control and the Stoer-Bulrish method[25]. Both methods seem to have the same computing power. We used the Stoer-Bulrish method for these results. For a normal inchworm we assumed 0.005m and 0.001Kg for each segment.

The system is implemented in C. The core of the dynamic routines is about 4000 lines. The computation time varies depending on the number of degrees of freedom. The inchworm simulation takes about three seconds on IBM RS/6000 320 for one step of the inverse dynamic computation. We store only key frame data and display after the whole computation.

6 Conclusion

In this paper, we have provided a new technique for inverse dynamic simulation of both closed and open loop systems. The technique is based on an extension of generalized coordinate transformations, which allows the use of an animation space with arbitrary functions of kinematic parameters defining its dimensions in place of the normal Cartesian workspace. This provides a flexible technique for designing the control interface for many types of articulated body models. Using this technique, we have given three approaches for the analysis of a closed loop, and we have used the constraint control technique on both closed and open loop mechanisms. We used a method based on the generalized principle of D'Alembert and virtual work for our dynamic control.

The complexity of our algorithm is $O(n^3)$ (with some optimization of the tensor multiplication), which is expensive compared to the linear complexity of some iterative dynamic formulations [3]. But usually linear complexity formulations, including Newton-Euler, are not well-suited to the design of control interfaces. Our dynamic formulation is good for the analysis of mechanisms, and we can use linear complexity algorithms after the initial analysis for the forward dynamic simulation of the models. A particularly attractive structural feature of this formulation is that the equations for models with a given branching topology are quite similar. This makes it very easy to derive equations for new models once some experience with the method is obtained.

Our initial experience with this technique indicates that it is a very promising method for the development and analysis of open and closed loop dynamic models as well as for inverse dynamic control. Further research experience will be required to understand how to determine good functions for generating the desired motion and how well the approach adapts to more complex modeling situations. The next step is to go from moving inchworms to walking humans, insects, octopi, Martians, etc. Since our animation space dimensions are not simple geometric quantities, it can be difficult to get a set of control functions by intuition. As the dimension of $\vec{u}$ space grows, these problems become more difficult.

We have also experimented with the *finite difference method of two point boundary value problem* of Witkin & Kass[30] and are now working on a human jumping animation using musculo-tendon-skeletal dynamics and weak variant of an *Optimal Control* technique. *finite difference method of two point boundary value problem* involves making a continuous time system completely discrete getting a set of nonlinear equations. An index function is used to organize these nonlinear equations so that they have some physical meaning. This approach achieves computational efficiency by giving up accuracy in modeling the physical world. Optimal control finds the best motion which satisfies all boundary conditions. This technique is very computation intensive because it does forward dynamic simulation while finding the best motion trajectory. We have produced a worm-racing movie generated using the [30]'s method to provide a comparison with the same animation using our technique.

7 Acknowledgements

We would like to thank Jane Wilhelms for her papers on dynamic animation, Bob Freeman for his discussions on robotics, Jesse Driver and A.T. Campbell for their support in taking images. This work was also supported in part by a grant from IBM Research Initiation Grant Program to support Interdisciplinary Computer Science Research.

References

[1] Americana. *The Encyclopedia Americana*, chapter Measuring Worm. Grolier Inc., 1990.

[2] W. Armstrong, M. Green, and R. Lake. Near-real-time control of human figure models. *IEEE Computer Graphics and Applications*, pages 52–61, June 1987.

[3] H. Asada and J. Slotin. *Robot Analysis and Control.* John Wiley and Sons, New York, NY, 1985.

[4] R. Ball. *Theory of Screws.* Cambridge University Press, Cambridge, England, 1990.

[5] Britannica. *The New Encylopedia Britannica*, chapter Measuring Worm. Encylopedia Britannica Inc., 1990.

[6] A. Bruderlin and T. Calvert. Goal-directed, dynamic animation of human walking. *Computer Graphics*, 23(3):233–242, July 1989.

[7] S. Chan and P. Lawrence. General Inverse Kinematics with the Error Damped Pseudoinverse. In *IEEE CH2555-1*, 1988.

[8] K. Cleary. *Decision Making Software for Redundant Manipulators.* PhD thesis, University of Texas at Austin, 1990.

[9] J. Craig. *Introduction to Robotics Mechanics and Control.* Addison-Wesley, Reading, MA, 1986.

[10] R. Freeman. *Kinematic and Dynamic Modeling, Analysis and Control of Robotic Mechanisms.* PhD thesis, University of Florida, 1985.

[11] R. Freeman. *Discussions with Dr. Freeman.* 1990.

[12] R. Freeman and D. Tesar. Dynamic Modeling of Serial and Parallel Mechanisms / Robotic Systems : Part II - Applications. In *Trends and Developments in Mechanisms, Machines and Robotics, 20th Biennial Mechanisms Conference*, 1988.

[13] R. Freeman and D. Tesar. Dynamic Modeling of Serial and Parallel Mechanisms / Robotic Systems : Part I - Methodology. In *Trends and Developments in Mechanisms, Machines and Robotics, 20th Biennial Mechanisms Conference*, 1988.

[14] A. Hayashi, J. Park, and B. Kuipers. Toward Planning and Control of Highly Redundant Manipulators. In *Fifth IEEE International Symposium on Intelligent Control*, Sep. 1990.

[15] K. Hunt. *Kinematic Geometry of Mechanisms.* Cambridge University Press, Cambridge, 1978.

[16] P. Isaacs and M. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics*, 21(4):215–224, July 1987.

[17] P. Isaacs and M. Cohen. Mixed methods for complex kinematic constraints in dynamic figure animation. *Visual Computer*, pages 296–305, April 1988.

[18] P. Lee, S. Wei, J. Zhao, and N. Badler. Strength guided motion. *Computer Graphics*, 24(4):253–263, August 1990.

[19] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics*, 24(4):29–38, August 1990.

[20] G. S. Miller. The motion dynamics of snakes and worms. *Computer Graphics*, 22(4):169–178, August 1988.

[21] D. N. Nenchev. Redundancy resolution through local optimization : A review. *Journal of Robotic Systems*, 6(6):769–799, 1989.

[22] M. Panne, E. Fiume, and Z. Vranesic. Reusable motion synthesis using state-space controllers. *Computer Graphics*, 24(4):225–234, August 1990.

[23] R. P. Paul. *Robot Manipulators : Mathematics, Programming and Control.* MIT Press, Cambridge, MA, 1982.

[24] C. Phillips, J. Zhao, and N. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics*, 24(2):245–250, 1990.

[25] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, England, 1986.

[26] M. Thomas and D. Tesar. Dynamic modeling of serial manipulator arms. *Transactions of the ASME*, 104:218–228, Sep. 1982.

[27] J. Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications*, 7(6):12–27, June 1987.

[28] J. Wilhelms. Dynamic animation : Interaction and control. *Visual Computer*, Dec. 1988.

[29] J. Wilhelms. *Making them Move*, chapter Dynamic Experiences. Morgan Kaufmann Publishers, 1991.

[30] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[31] J. Zhao and N. Badler. Real Time Inverse Kinematics with Joint Limits and Spatial Constraints. MS-CIS 89-09, University of Pennsylvania, January 1989.