# Interactive Human Motion Control Using a Closed-form of Direct and Inverse Dynamics

Zhiyong Huang[1]
Nadia Magnenat Thalmann[2]
Daniel Thalmann[1]

*1) LIG - Computer Graphics Lab*
*Swiss Federal Institute of Technology*
*Lausanne, CH-1015 Switzerland*
*fax: +41-21-693-5328*
*email: {huang, thalmann}@di.epfl.ch*

*2) MIRALab-CUI, University of Geneva*
*24 rue du Général Dufour*
*Geneva, CH -1211 Switzerland*
*fax: +41-22-320-2927*
*email: thalmann@cui.unige.ch*

**ABSTRACT**

In this paper, we propose the use of inverse dynamics in a closed-form with direct dynamics for interactive motion control of a human skeleton. An efficient recursive algorithm based on Newton-Euler formulae is used to calculate the force and torque produced by joint actuator in order to fulfill a desired motion. The resulting force and torque are then used in direct dynamics to make the final motion with external force and torque. Armstrong-Green algorithm is used for direct dynamic simulation. To decrease the errors in numerical integration, we use fourth-order Runge-Kutta method instead of Euler method. Inverse dynamic functions calculate the required force and torque at every small time interval in the process of direct dynamic simulation. In this way, it will correct errors at each time interval. The direct and inverse dynamic functions are integrated in the software TRACK with direct and inverse kinematics functions that provide a more powerful way for human animation.

## 1. Introduction

Today, there are two trends in research in motion control: new methodologies of general control and applications of specific and functional methods to the models. The first approach consists of finding new ways of using the existing laws either directly or after simplification for motion control. [1] The major methods include keyframing,[2] functional method,[3] inverse kinematics,[3] direct and inverse dynamics,[4][5][6] or optimal control.[7][8] The second approach corresponds to methods coming from various disciplines. In human animation, recent research includes walking,[9] motion of flexible torso and spine,[10] jumping,[11] ballroom dances,[12] etc. The goal of all these applications is looking for realism of the desired motion. To improve motion realism requires use of a more complex motion control method on a more realistic model. It motivates our efforts in this paper to use dynamics for motion control of our human skeleton.

Because of its physics-based nature, direct dynamics can give the best quality motions. The classic example is dropping of a pinpointed chain. A human is an actuated system that uses muscles to convert stored energy into time-varying forces and torques acting on the joint, contrary to a passive object like a pinpointed chain that is only driven by external force and torque during motion. The problem is how to find the time-varying forces and torques before using the direct dynamics. There has been some work in this aspect recently.[6][8][24]. The Spacetime Control in [8] has given an impressive sequence of a dynamic lamp. The method describes a formation of animation as a constrained multi-point boundary-value problem, thus allowing the user to specify initial, intermediate and final configurations. The SQR method used to solve nonlinear optimization requires considerable computational effort. There is one point in [8] that we think is very helpful in our work, and that is motion planning done in joint coordinates, or configuration space, instead of Cartesian space. Our work is also similar to [6] in the use of both direct and inverse dynamics, but we think the Newton-Euler is more computationally efficient than the D'Alembert's principle of virtual work which can be calculated in almost real time. An efficient algorithm is given by Luh, Walker and Paul [23] in robotics. Our human skeleton model is different from the robotics model in two major parts: in the human body, there are one to three DOF at one joint, and where no translation DOF in our model, so we adapted the formulations to our specific purpose. Armstrong-Green method[5] is a very efficient algorithm based on Newton-Euler formulation. By using it together with inverse dynamics in a closed-form, we provide a way for human skeleton motion control.

This paper is organized in two main sections: modeling and motion control. In order to make this paper self-containing, the computational scheme for inverse dynamics based on the Newton-Euler formulation is given in the Appendix. This appendix is an adaptation from the formulation in [23].

## 2. Modeling

A human skeleton has about 206 bones in total.[21] However, in applications, human skeleton models are quite different from the easiest five-link planar model[13] to multiple link 3D model [14]. We have made a simplified model of the human skeleton with rigid articulated figures connected by one to three DOF at each joint. This model is shown in Figure 1. It contains 49 joints with 88 degrees of freedom, DOF. We do not represent joints inside head, not considering the facial animation. The number of links in the spine is decreased to 4 by neglecting the vertebrae in thorax totally and abdomen partly, that is less detail than the representation in [10], where the interest is in motion of the spine.

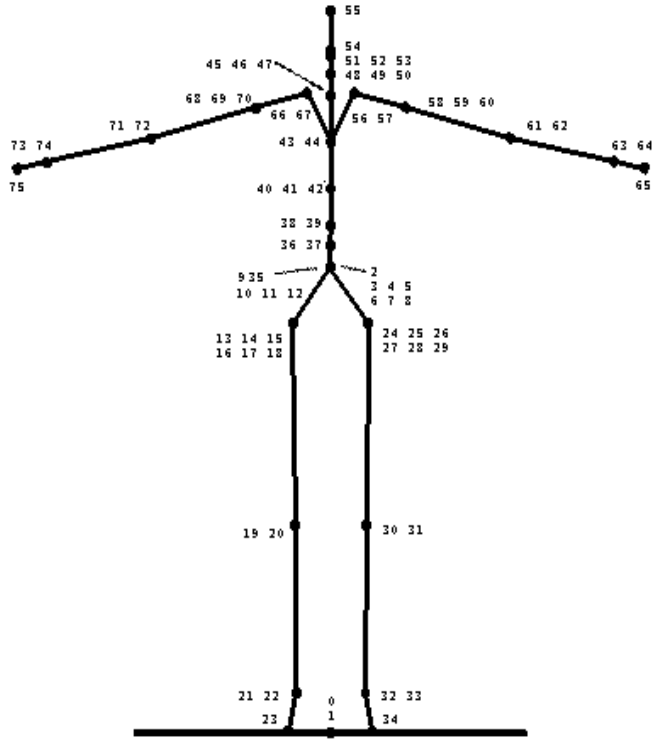
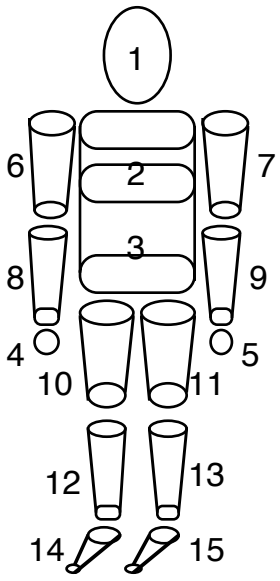



Figure 1. Human Skeleton Model without Hands

In order to use dynamic simulation, the volume of the human should be modeled. We approximate it with 15 solid primitives, cylinder, sphere ellipsoid, and truncated cone. It is easy to calculate the geometric and physical properties with this approximation. The mass of each volume is derived from the Biomechanical experiment [18]. In Figure 2(a), a skeleton is associated with solid, and its mass is listed in Figure 2(b). All solid primitives are rigid. They are linked by joints in a hierarchical structure.




| Name | Type | Mass (Segment / Total) |
|---|---|---|
| head and neck | ellipsoid | 0.081 |
| up_torso | cylinder | 0.216 |
| down_torso | cylinder | 0.281 |
| r_hand | sphere | 0.006 |
| l_hand | sphere | 0.006 |
| r_upper_arm | frustum | 0.028 |
| l_upper_arm | frustum | 0.028 |
| r_forearm | frustum | 0.016 |
| l_forearm | frustum | 0.016 |
| r_thigh | frustum | 0.100 |
| l_thigh | frustum | 0.100 |
| r_leg | frustum | 0.0465 |
| l_leg | frustum | 0.0465 |
| r_foot | frustum | 0.0145 |
| l_foot | frustum | 0.0145 |

(a) Skeleton with Solid (b) The Mass of Each Solid

Figure 2. Representation of Body Volume

The motion can be described by the position and orientation of each solid. To describe its motion, we define a body coordinate frame for each of them with the origin at the hinge point to another solid. The body coordinate frame is moving together with the solid. All geometric and physical properties defined in this frame do not change for the rigidity of solid. For this reason, most formulae are represented in the body frame for simplicity.

Our human body model is different from [5], where between two solids, there is only one coordinate frame. The advantage of one frame connection is its simplicity when updating the structure of the structure in dynamic simulation, but it is very difficult to represent the different number of DOF at each joint, and the value range of each DOF. In our human body modeling, between two links, there are one to three DOF coordinate frames. The frame number is equal the DOF number of the joint. The associated link only can rotate around the Z axis of the DOF frame. It adds more complexity to update the structure. In Figure 3, the detail of our model is illustrated from thorax to right arm. It is the same for other parts.

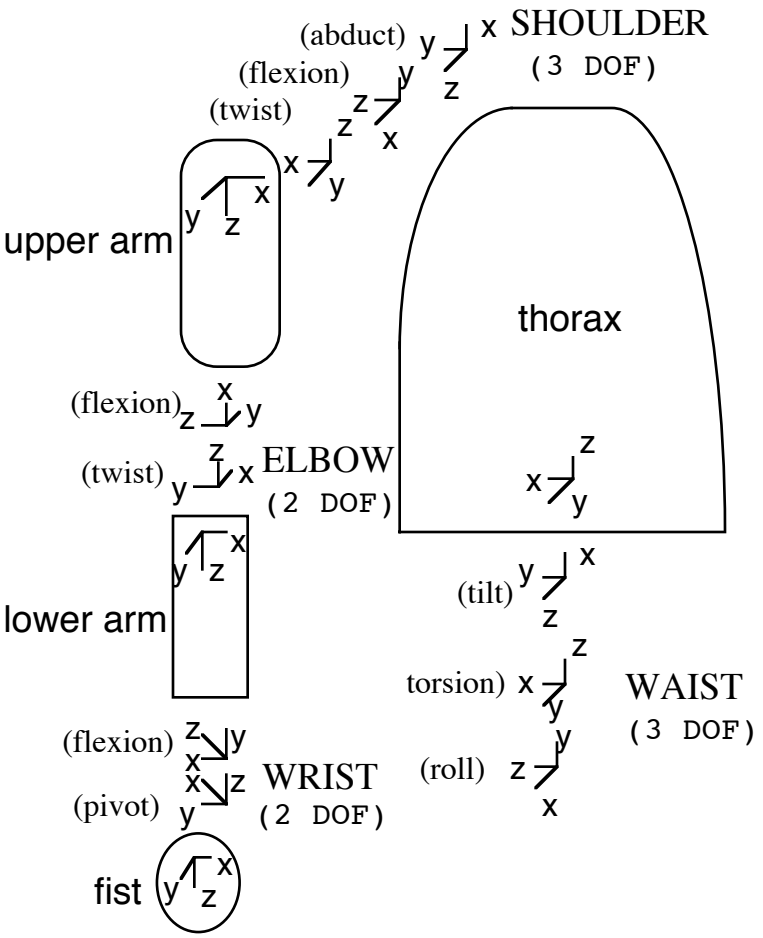


Figure 3. Right Arm and Thorax Model, The origin of all DOFs of same joint is one point that is the same point of the origin of the next body frame. Each DOF only can rotate around its Z axis.

The model is introduced in [15], represented by a kind of homogeneous hierarchy named H3D with neutral node node_3D to retain the geometric, topological and display information. A node_3D can be combined with other type nodes, such as DOF, SOLID, FIGURE, FREE and SCENE for applications. One can refer [15] for more detail description.

## 3. Motion Control

Since the introduction of physically-based modeling in computer graphics in the late 1980s [25], it provides a new research direction for computer animation. Its purpose is to produce more realistic animation by force and torque equations from the physical laws. With the latest high speed workstation, they can be simulated in real time. Among the work of motion control of articulated figure [5][6][24], we are most interested in Armstrong-Green algorithm described detail in [5] for its computational efficiency and the simplicity of the equations. To use it in our work, we have to solve two problems, the first is adjusting the equations for the differences of our model, and the second, to get the time-varying torque produced by muscle at the actuated joint to perform the desired motion. It is the major problem since the introduction of physically-based modeling. In this paper, we use a kind of inverse dynamics formulation based on Newton-Euler equation to get the time-varying force and torque for direct dynamic simulation. The details are described in the following parts.

### *3.1 Direct Dynamic Simulation*

The most popular algorithm in dynamic simulation is Armstrong-Green algorithm, based on Newton-Euler formulations (see [5] for details). To get a simplified form in order to avoid the inversion of matrices larger than three by three, two hypotheses have be made. The first assumes the linear relationship between the linear acceleration of the link and the amount of angular acceleration it undergoes. The second is the linear relationship between the linear acceleration and the reactive force on its parent link. The algorithm can be divided to two opposite processes, inbound and outbound. The inbound calculates some matrices and vectors from the geometric and physical structure of the system, and propagates force and torque along each link from leaves to the root of the hierarchy. The outbound calculates kinematics quantities, the linear and angular acceleration of each link, then to get the linear and angular velocity by numerical integration for updating the whole structure. It is a recursive algorithm along the whole time of dynamic simulation. The kinematic results of the former time step are used as an initial values of the next step calculation.

Euler method is used for numerical integration in [5] at each step. In our work, we find this is not enough when simulating the motion for a long time period. The errors due to the numerical integration will accumulate from step to step. At the end, unrealistic motion

will happen. To get a higher precision, we use fourth-order Runge-Kutta method for numerical integration. The general algorithm is described in detail in [26]. To get the angular velocity for each step, we use the inbound-outbound process four times to calculate the change rate of angular velocity. By doing so, we find the stability of algorithm is improved.

On the other hand, considering that the model we use is not one coordinate frame connection structure, we can not directly change the rotation matrix as [5] in each step. Instead, we need to distribute the rotation vector to each DOF coordinate frame of the joint by solving the following linear equation,

$$z_1\theta_1 + z_2\theta_2 + z_3\theta_3 = \omega \qquad \text{.....(3.1)}$$

where $z_i$ is the rotation axis as shown in the Figure 3. It is represented in the body frame. $\omega$ is the current angular velocity of joint relative to its parent link coordinate frame. It is also represented in the current body coordinate frame. In a very small time step, the rotation of the link can be regarded as planar rotation. The normal of the plane has the same direction as $\omega$. $\theta_i$ is the change of the ith DOF that should be solved from the equation 3.1. One will notice the equation 3.1 is for the joint having 3 DOF. For the joint with one or two DOF, we should update the angular velocity each time from the solution of the equation 3.1. To find the actuator joint force and torque for direct dynamic simulation, we need use the inverse dynamic formulations described in the next section.

## *3.2 Inverse Dynamics*

The inverse dynamics problem is to find at each joint the force and torque that generate the desired motion of the structure. It is a fundamental problem of robotics to get the force and torque for each DOF motor that will drive the robot arm to perform a defined task [23]. Various forms of robot arm motion equations are derived mainly from Lagrange-Euler and Newton-Euler formulations. The motion equations are equivalent to each other in the sense that they describe the dynamic behavior of the same physical robot manipulator. However the structure of these equations may differ as they are obtained for various reasons and purposes. Among many formulations, we select a kind of recursive formulation based on Newton-Euler equations for their computational efficiency. It is described in [22][23] and is successfully used in robotics control. The best advantage is that the computation time is linearly proportional to the number of joints of the robot arm and independent of the robot arm configuration. It is based two basic equations,

$$F = ma \qquad \text{.....(3.2)}$$
$$N = J\alpha + \omega \times J\omega \qquad \text{.....(3.3)}$$

The first is Newton's second law that describes the linear motion of the rigid body. $F$ is the total vector force on the body with mass $m$. The second one is Euler's equation that governs the rotation of the rigid body. $N$ is the total vector torque on the body with the

inertia matrix $J$ about its center of mass. The linear acceleration $a$, angular acceleration $\alpha$, and angular velocity $\omega$ define the kinematics of the rigid body.

To construct the formulations for the human body model described in part 2, we use the same way in robotics, writing a series equations for each link, using constrained forces and torque to guarantee their connection. The details of this formulation for inverse dynamics are listed in the Appendix. We can see it has two opposite process. The forward recursion starts from the inertial coordinate frame to the end-effector coordinate frame to forward propagate kinematics information. The backward recursion propagates the forces and moments exerted on each link from the opposite direction.

Using inverse dynamics in a closed from with direct dynamic simulation is a key for getting the desired motion. In Figure 4, we describe the closed-form control with frames. The desired motion is represented by joint variables $q, \dot{q}$ and $\ddot{q}$. Here we use the same notation as robotics. In Figure 4, the `time_step` is the period to use inverse dynamics. It is about 10 times of the period of Armstrong-Green algorithm `small_time_step` from our experiment. This is what we mean by closed-form control.

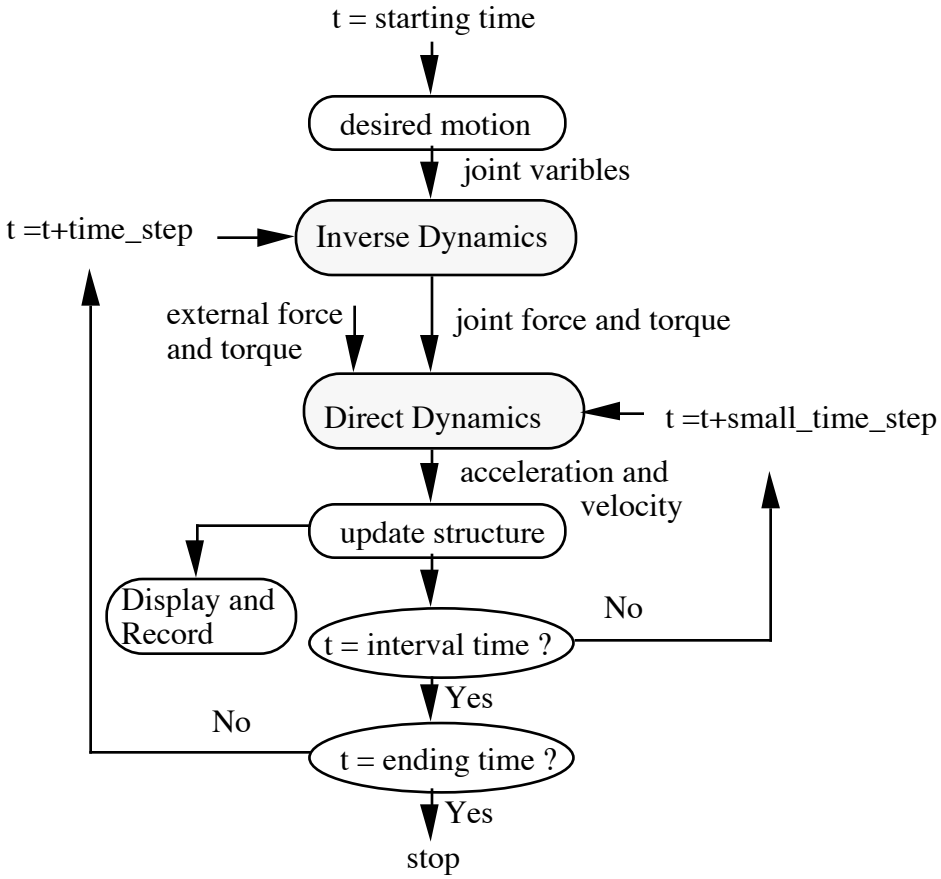



Figure 4. The closed-form control with inverse dynamics

## 4. Evaluation and Results

In this part, we want to show some examples of motion control method described in the section 3. The first example uses direct dynamics with only gravity and supporting force. The supporting force acts on the center of gravity of the whole body located at the abdomen. Its value equals the gravity of the whole body, but with opposite direction. In Figure 5, we show some moments by snapshot from SGI workstation. One will find the motion of arms and legs is like a pendulum because we set all the DOF of them free to check the correctness of the resulting motion.

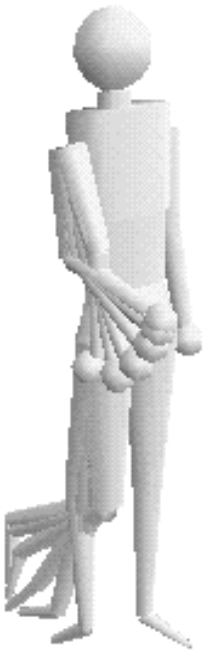


Figure 5. The dynamic motion under gravity and supporting force

In the second example, we use inverse dynamics to get the joint torque of the upper body with root of thorax to keep it still in the gravity field. The motion control is described in Figure 4. The upper body keep still as shown in Figure 6 in dynamic simulation. In order to show the value and direction of joint force and torque clear, we do not display the solids representing the volume. `G` is the gravity of each solid which unit is [Newton], and `ti` is the internal torque from inverse dynamics which unit is [Newton][Meter].

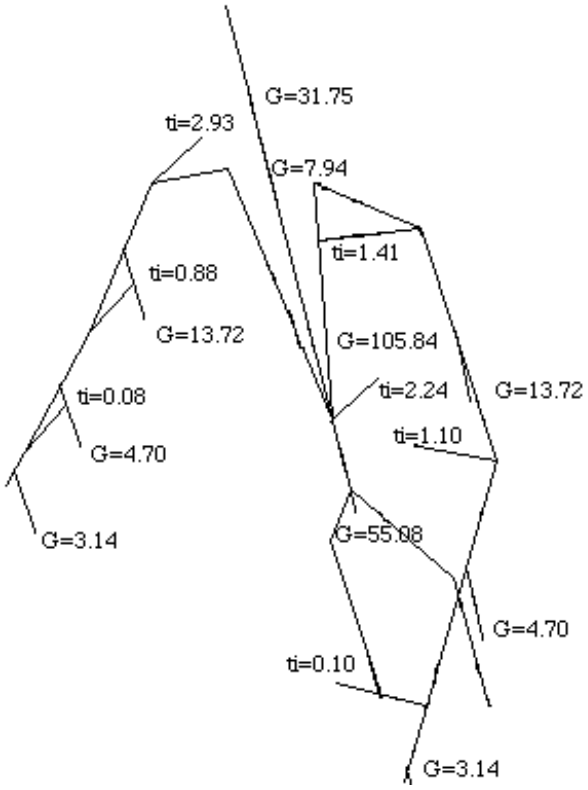
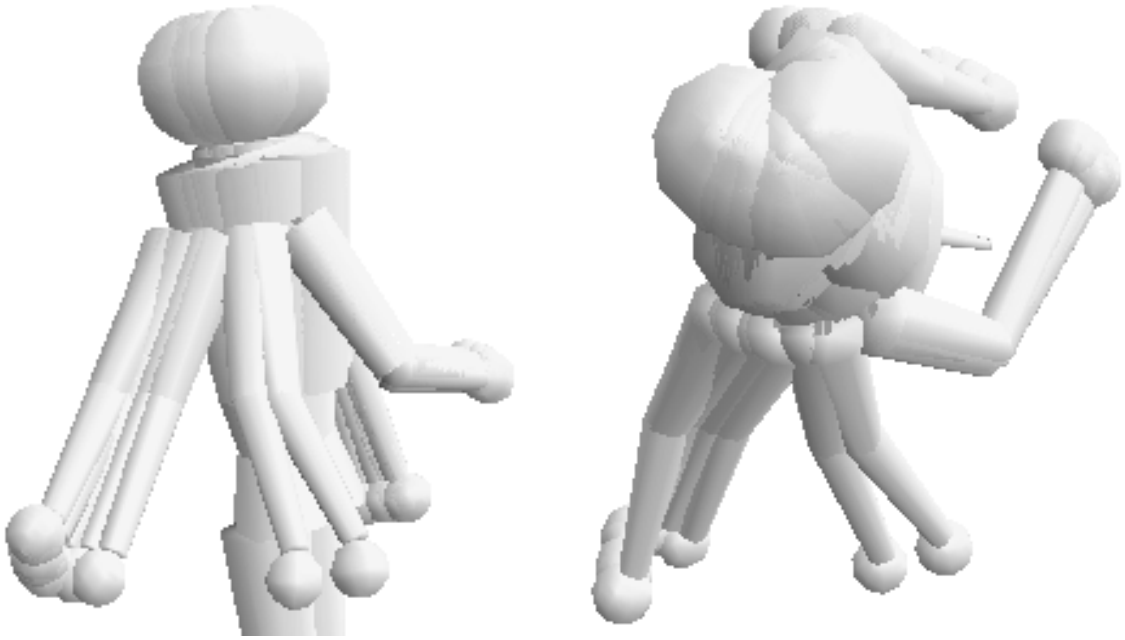


Figure 6. Keep upper body still in gravity field

We want to show a swing motion in the final example. It is similar to the motion as forehand hit tennis ball. In Figure 7, we show series postures of the motion in two views. The time step for simulation is 0.001 seconds, and the total time is 1.8 seconds.



(a ) side view (b) top view

Figure 7. Swing arm by dynamics with 0.04 second time step in display

The joint force and torque is calculated from inverse dynamics to drive the motion. In Figure 8, the joint torque of shoulder is drawn with curves.

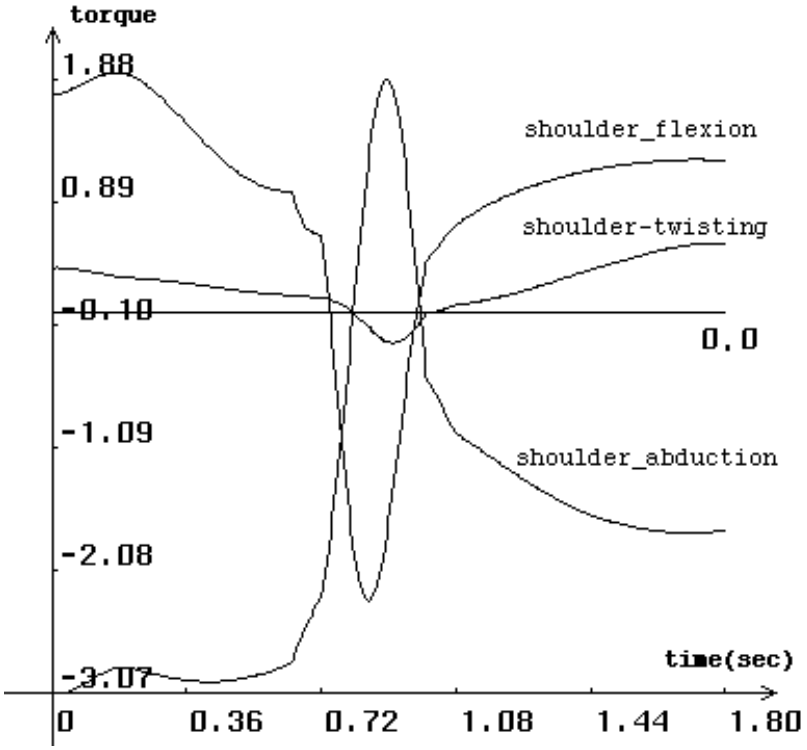



Figure 8. The joint torque from inverse dynamics

Color Plate 1 : Part of the interface layout of the TRACK system

## 5. Implementation

The software has been implemented in ANSI C on SGI Indigo 2. The Lig 5D Toolkit [20] is used for the user interface. The dynamic functions are integrated in TRACK, a human motion control software developed in the Computer Graphics Lab, EPFL. [16] The resulting motion can be represented by multiple tracks, that is the same representation of motion produced by other functions.

## 6. Conclusion and Future Work

Our approach seems very promising to implement complex motion. In the future, we will try to dynamically animate a game with information from the sensor of synthetic actor, and use a more sophisticated control strategy. We should also further consider interaction between the actor and the environment.

## 7. Acknowledgments

We will thank our colleagues Dr. Ronan Boulic, Mr. Shen Jianhua, Mr. Wu Yi and Mr. Walter Maurel for helpful discussions, and also Mr. Hans-Martin B. Werner for checking the text. The research was supported by the Swiss National Science Research Foundation and Esprit Project Humanoid.

## 8. Appendix : Recursive Newton-Euler Formulation

Table 1. Summary of Notation

---

All vectors with sub index i are expressed in the body coordination frame of the ith link.
The following items except $m_i$ are vectors.

$q_{ij}, \dot{q}_{ij}, \ddot{q}_{ij}$ : the first and second derives of generalized variable for the jth DOF of the ith joint.

$\omega^i$ : angular velocity of the ith link.

$\alpha^i$ : angular acceleration of the ith link.

$a^i$ : linear acceleration of the ith link at the origin of its local coordination frame.

$a_c^i$ : linear acceleration of the ith link at the center of mass.

$g$ : gravitational acceleration vector in the inertial coordination frame.

$g_i$ : gravitation of the ith link.

$f_i$ : force exerted on the ith link by the (i-1)st link.

$n_i$ : moment exerted on the ith link by the (i-1)st link.

$\tau_i$ : joint generalized actuator torque at the ith joint.

$p_i$ : vector from the origin of ith link to the origin of (i+1)st link.

$s_i$ : vector from the origin of ith link to its center of mass.

$z_{ij}$ : rotation vector of the jth DOF of the ith joint.

$m_i$ : mass of link i.

$F_i$ : total external force exerted on the ith link.

$N_i$ : total external torque exerted on the ith link.

The following items are 3 by 3 matrix.

$J_i$ : inertia matrix of the ith link.

$A_i^j$ : pure rotational matrix from the jth link coordination frame to the ith.

Integer

$nd_i$ : number of DOF at joint i.

---

Table 2. Recursive Newton-Euler Algorithm

---

Initialization

$\omega^0 = \alpha^0 = v^0 = 0$

$f_{n+1} = force\ required\ at\ the\ end-effector$

$n_{n+1} = moment\ required\ at\ the\ end-effector$

Forward recursion

*For* $i = 1,\ldots,n$ *do*:

$$\omega^i = A_i^{i-1}\omega^{i-1} + \sum_{j=1}^{nd_i} z_{ij}\dot{q}_{ij}$$

$$\alpha^i = A_i^{i-1}\alpha^{i-1} + \sum_{j=1}^{nd_i} z_i\ddot{q}_{ij} + A_i^{i-1}\omega^{i-1} \times \sum_{j=1}^{nd_i} z_{ij}\dot{q}_{ij}$$

$$a^i = A_i^{i-1}(a^{i-1} + \alpha^{i-1} \times p_{i-1} + \omega^{i-1} \times (\omega^{i-1} \times p_{i-1}))$$

$$a_c^i = a^i + \alpha^i \times s_i + \omega^i \times (\omega^i \times s_i)$$

$$g_i = A_i^0(m_i g) \quad /* \quad g = (0.0, 0.0, -9.8) \quad */$$

$$F_i = m_i a_c^i$$

$$N_i = J_i\alpha^i + \omega^i \times J_i\omega^i$$

Backward recursion

*For* $i = n,\ldots,1$ *do*

$$f_i = F_i + A_i^{i+1} f_{i+1} - g_i$$

$$n_i = N_i + A_i^{i+1} n_{i+1} + s_i \times (F_i - g_i) + p_i \times A_i^{i+1} f_{i+1}$$

$$\tau_i = n_i \cdot z_i$$

*end*;

---

## 9. References


1. D. Thalmann and N. Magnenat-Thalmann, *Motion Control of Human Animation*, Siggraph'93 Course Notes: Computer Animation.
2. S. N. Stteketee and N. Badler, *Parametric Keyframe Interpolation In Incorporating Kinetic Adjustment and Phrasing Control,* Proceedings of Siggraph'85.
3. N. Badler, P. Lee, C. Philips and E. Otani, *The Jack interactive human model*, Concurrent Engineering of Mechanical Systems, First Annual Symposium on Mechanical Design in a Concurrent Engineering Environment, University of Iowa, Iowa City, IA.
4. C. W. Reynolds, *Computer Animation with Scripts and Actors*, Proceedings of Siggraph'82.
5. W. W. Armstrong and M. W. Green, *The dynamics of Articulated Rigid Bodies for Purposes of Animation*, Proceedings of Graphics Interface'85.
6. P. M. Isaacs and M. F. Cohen, *Controlling Dynamics Simulation with Kinematic Constraints*, Proceedings of Siggraph'87.
7. L. S. Brotman and A. N. Netravali, *Motion interpolation by optimal control.*, Proceedings of Siggraph'88.
8. A. Witkin and M. Kass, *Spacetime Constraints*, Proceedings of Siggraph'85.
9. R. Boulic, D. Thalmann and N. Magnenat-Thalmann, *A Global Human Walking Model with Real-Time Kinematic Personification*, The Visual Computer, Vol. 6, No 6, December 90.
10. N. Badler and G. Monheit, *A Kinematic Model of the Human Spine and Torso*. IEEE Computer Graphics & Applications, Vol. 11, No. 2, March 1991.

11. K. Arai, *Keyframe Animation of Articulated Figures Using Partial Dynamics*, Models and Techniques in Computer Animation, edited by Magnenat-Thalmann N and Thalmann D, Springer-Verlag 1993.
12. R. Lake and M. W. Green, *Dynamic Motion Control an Articulated Figure Using Quaternion Curves*, First Conf. on CAD and Graphics, Hangzhou, 1991.
13. H. Hemami and R. L. Farnsworth, *Posture and gait stability of a planar five link biped by simulation*, IEEE trans. on Automatic Control, AC-22, 1977.
14. G. Monheit and N. Badler, *A Kinematic Model of the Human Spine and Torso*, Computer Graphics and Applications, Vol. 11, No. 2, March, 1991.
15. R. Boulic and O. Renault, *3D Hierarchies for Animation, New Trends in Animation and Visualization*, edited by Magnenat-Thalmann N and Thalmann D, JOHN WILEY & SONS, 1991.
16. R. Boulic, Z. Huang, N. Magnenat-Thalmann, and D. Thalmann, *A Unified Framework for the Motion Manipulation of Articulated Figures with the TRACK System*, Second Conf. on CAD and Graphics, Beijing, 1993.
17. N. Magnenat-Thalmann, D. Thalmann, *The Direction of Synthetic Actors in the Film Rendez-vous a Montreal*, IEEE Computer Graphics and Applications, Vol. 7, No. 12, 1987.
18. A. Seireg. and R. Arvikar, *Biomechanical Analysis of the Musculoskeletal Structure for Medicine and Sports* , Hemisphere Publishing Corporation, NY 1989.
19. R. Farouki and V. Rajan, *On the numerical condition of polynomials in Bernstein form*, Computer Aided Geometric Design, 4(3), 1987.
20. R. Turner, E. Gobbetii, F. Balaguer, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann, *An Object-Oriented Methodology using Dynamic Variables for Animation and Scientific Visualization*, Proceedings of Computer Graphics International'90, Springer-Verlag.
21. J. V. Basmajian, *Grant's METHOD of ANATOMY*, Tenth Edition, The Williams & Wilkins Company, 1980.
22. K. S. Fu, Gonzalez, and C. S. G. Lee, *Robotics, Control, Sensing, Vision, and Intelligence*, McGraw-Hill, inc. 1987, p. 111.
23. J. Y. S. Luh, M. W. Walker and R. P. C. Paul, *On-line computation control of mechanical manipulator*. Trans. ASME J. Dynamic Systems, Measurement, and Control, 120:69-76, 1980.
24. J. Park, D. Fussell, M. Pandy, and J. C. Browne, *Realistic Animation using Musculotendon Skeletal Dynamics and Suboptimal Control*, Eurographics Worshop on Computer Animation, 1992.
25. R. Barzel, *Physically-Based Modeling for Computer Graphics: A Structured Approach*, Academic Press, Inc, 1992
26. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C, The Art of Scientific Computing*, Second Edition, Cambridge Press 1992.