Proceedings of the 1999 IEEE/RSJ
International Conference on
Intelligent Robots and Systems

Efficient Algorithm for Extended Operational Space Inertia Matrix

Kyong-Sok Chang

Robotics Lab., Computer Science Dept.
Stanford Univ., Stanford, CA 94305, U.S.A.
kcchang@cs.stanford.edu

Oussama Khatib

Robotics Lab., Computer Science Dept.
Stanford Univ., Stanford, CA 94305, U.S.A.
khatib@cs.stanford.edu

Abstract

This paper describes an efficient recursive algorithm for the computation of the extended operational space inertia matrix of an n*-link branching (tree-like) redundant robotic mechanism with multiple operational points. The proposed algorithm behaves linearly with respect to* n *in practice. Therefore, as the number of links increases, this algorithm performs significantly better than the existing* $O(n^3)$ *symbolic method. The experimental results of this algorithm are presented using real-time dynamic simulation.*

1 Introduction

The extended operational space formulation [9] is an approach for the dynamic modeling and control of a complex branching (tree-like) redundant mechanism (Figure 1) at its task or end-effector level. This formulation is particularly useful for dealing with simultaneous tasks of multiple end-effectors since its basic structure provides dynamic decoupling among end-effectors' tasks and the complex internal dynamics in their associated null space.

In order for this formulation to be usable in real-time control of a complex n-link mechanism, however, the complexity $O(n^3)$ of the existing symbolic method [9] is not acceptable when n is large. This $O(n^3)$ complexity comes from the explicit inversion operation of the joint space inertia matrix of size $O(n^2)$, required for the computation of the extended operational space inertia matrix.

In this paper, we propose an efficient recursive algorithm for the computation of the extended operational space inertia matrix of an n-link branching redundant robotic mechanism with m operational points. Since m can be considered as a small constant in practice, we obtain the linear running time of $O(n)$ for this algorithm.

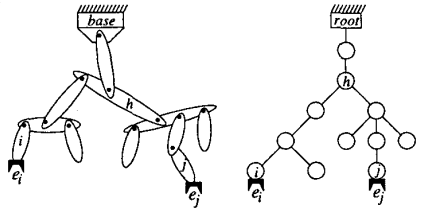

Figure 1: A branching robot with a tree-like topology. In its corresponding tree structure, each link becomes a node and each joint becomes an edge of the tree.

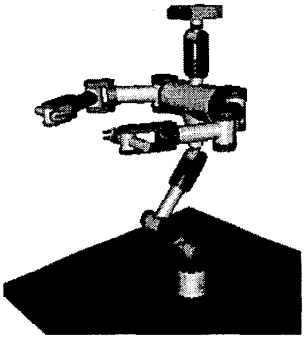

Figure 2: Robot.

The next section provides background material describing a modified spatial notation and basic spatial kinematic and dynamic quantities using this notation. In the third section, an efficient recursive algorithm is developed based on these spatial quantities and its $O(n)$ complexity is proved. Finally, real-time simulation results with a basic humanoid redundant robotic mechanism with $n = 24$ and $m = 2$ (Figure 2) are presented to illustrate the efficiency of the proposed algorithm.

2 Background

This section summarizes the *unified spatial notation* used throughout the paper. Also, some of the spatial quantities, which are essential for developing the proposed algorithm in the next section, are presented using this notation.

0-7803-5184-3/99/$10.00 © 1999 IEEE

2.1 Unified Spatial Notation

Spatial notation, introduced by Featherstone [1], has been widely used in the modeling of kinematics and dynamics of complex robotic mechanisms [1, 8, 4, 5, 6, 7]. In this subsection, we introduce the *unified spatial notation* which combines various versions of existing spatial notations in order to utilize the results from various researchers in a unified way.

In spatial notation, each quantity incorporates the appropriate linear and angular components and results in a concise form (6×1 vector or 6×6 matrix). For example, a spatial acceleration, $\ddot{\mathbf{x}}_i$, and a spatial force, $\mathbf{f}_i$, of link i are defined as:

$$\ddot{\mathbf{x}}_i = \begin{bmatrix} \dot{v}_i \\ \dot{\omega}_i \end{bmatrix} \quad \text{and} \quad \mathbf{f}_i = \begin{bmatrix} f_i \\ n_i \end{bmatrix}$$

where v_i, ω_i, f_i, and n_i, are 3×1 linear velocity, angular velocity, force, and moment vectors expressed in frame i, respectively.

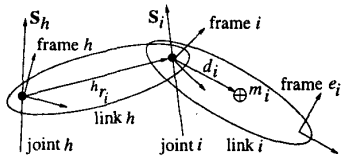

Figure 3: Basic notation.

In Figure 3, ${}^h r_i$ is a 3×1 position vector from the origin of frame h to the origin of frame i expressed in frame h, d_i is a 3×1 position vector from the origin of frame i to the center of mass of link i expressed in frame i, and m_i is the mass of link i.

A spatial inertia matrix, $\mathbf{I}_i$ is a 6×6 symmetric positive definite matrix defined for each individual link i in its own frame:

$$\mathbf{I}_i = \begin{bmatrix} M_i & H_i^T \\ H_i & I_i^* \end{bmatrix}; \quad \begin{array}{l} M_i = m_i \mathbf{1}_3 \\ H_i = M_i \widehat{d}_i \\ I_i^* = I_{c_i}^* - M_i \widehat{d}_i \widehat{d}_i \end{array} \tag{1}$$

where M_i, H_i, and I_i^* are the 0^{th}, 1^{st}, and 2^{nd} moments of inertia of link i, respectively. Notice that I_i^* is the inertia tensor at the origin of frame i. Also, $\mathbf{1}_3$ is a 3×3 identity matrix and $I_{c_i}^*$ is the inertia tensor of link i in frame c_i located at the center of mass of link i with the same orientation as frame i. $\widehat{d}_i$ is the cross-product operator associated with d_i shown in Figure 3. A cross-product operator associated with a 3×1 vector, $p = [p_x \; p_y \; p_z]^T$, is an anti-symmetric matrix defined as:

$$\widehat{p} = p\times = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix}$$

The 6×6 spatial transformation matrix, ${}^h_i\mathbf{X}$, transforms a spatial quantity from one frame (i) to the other (h):

$${}^h_i\mathbf{X} = \begin{bmatrix} {}^h_i R & \mathbf{0} \\ \widehat{{}^h r_i}\, {}^h_i R & {}^h_i R \end{bmatrix} \tag{2}$$

where ${}^h_i R$ is the 3×3 rotation matrix which transforms a quantity expressed in frame i to the same quantity expressed in frame h and $\widehat{{}^h r_i}$ is the cross-product operator associated with ${}^h r_i$ shown in Figure 3. For example, the spatial transformations of accelerations and forces between frames i and j are:

$$\ddot{\mathbf{x}}_i = {}^h_i\mathbf{X}^T \ddot{\mathbf{x}}_h \tag{3}$$

$$\mathbf{f}_h = {}^h_i\mathbf{X}\, \mathbf{f}_i \tag{4}$$

Note that the origin of each frame is located at the joint and any variable without the reference frame number (front superscript) is expressed in its own frame. Also, if link i is a leaf (outermost) link, end-effector frame e_i is located at the tip (operational point) of link i (Figure 1). These conventions will be assumed throughout the paper.

2.2 Supporting Spatial Quantities

In this subsection, we present some of the essential spatial quantities using the unified spatial notation to support the proposed algorithm developed in the next section.

The force propagator, ${}^h_i\mathbf{L}$, propagates a spatial force from link i to its parent link h across the actuated joint i in a dynamically consistent manner [5]:

$$\mathbf{f}_h = {}^h_i\mathbf{L}\, \mathbf{f}_i \tag{5}$$

where $\mathbf{f}_h$ is the resulting spatial force of link h when the spatial force of link i, $\mathbf{f}_i$, is propagated across joint i. Force propagation is physically valid only if the spatial force is propagated in inward (tip-to-base) direction.

Similarly, the acceleration propagator defined as ${}^h_i\mathbf{L}^T$ propagates a spatial acceleration of link h to its child link i across the actuated joint i in a dynamically consistent manner [1, 5]:

$$\ddot{\mathbf{x}}_i = {}^h_i\mathbf{L}^T \ddot{\mathbf{x}}_h \tag{6}$$

where $\ddot{\mathbf{x}}_i$ is the resulting spatial acceleration of link i when the spatial acceleration of link h, $\ddot{\mathbf{x}}_h$, is propagated across joint i. Acceleration propagation is physically valid only if the spatial acceleration is propagated in outward (base-to-tip) direction.

The force propagator, ${}^h_i\mathbf{L}$, and the acceleration propagator, ${}^h_i\mathbf{L}^T$, are defined as [1, 5]:

$$ {}^h_i\mathbf{L} = {}^h_i\mathbf{X}\left[\mathbf{1}_6 - \mathbf{S}_i\overline{\mathbf{S}}_i\right]^T \quad (7) $$

$$ {}^h_i\mathbf{L}^T = \left[\mathbf{1}_6 - \mathbf{S}_i\overline{\mathbf{S}}_i\right]{}^h_i\mathbf{X}^T \quad (8) $$

where $\mathbf{1}_6$ is a 6×6 identity matrix and the general joint model, $\mathbf{S}_i$, is a $6 \times n_i$ matrix with full column rank, n_i, when joint i has n_i degrees of freedom ($n_i \leq 6$) [1, 5]. Its columns (unit vectors) make up a basis for the motion space of joint i. Notice that this matrix is constant since it is expressed in its own frame. For example, $\mathbf{S}_i = [0\ 0\ 0\ 0\ 0\ 1]^T$ for a revolute joint around z-axis and $\mathbf{S}_i = [0\ 1\ 0\ 0\ 0\ 0]^T$ for a prismatic joint along y-axis.

$\overline{\mathbf{S}}_i$ is the generalized inverse of $\mathbf{S}_i$ weighted by the corresponding inertia matrix and defined as:

$$ \overline{\mathbf{S}}_i = \mathbf{K}_i^{-1}\,\mathbf{S}_i^T\,\mathbf{I}_i^A $$

The articulated-body inertia matrix of link i, $\mathbf{I}_i^A$, introduced by Featherstone [1], relates the spatial force and acceleration of a link, taking into account the dynamics of the rest of the articulated body [1, 8, 5]. Using the force propagator (7) and the acceleration propagator (8), the articulated-body inertia matrix of link h, $\mathbf{I}_h^A$, can be written as:

$$ \mathbf{I}_h^A = \mathbf{I}_h + \sum_i \left[{}^h_i\mathbf{L}\,\mathbf{I}_i^A\,{}^h_i\mathbf{L}^T\right] \quad (9) $$

where $\mathbf{I}_h$ is the spatial inertia matrix (1) of link h and i represents the index of each child link of link h. This recursive equation shows that the articulated-body inertia of a link is the sum of its own spatial inertia and the dynamically consistent projection of the articulated-body inertia of each child link. Note that $\mathbf{I}_h^A = \mathbf{I}_h$ if link h is a leaf link.

$\mathbf{K}_i$ is the $n_i \times n_i$ full rank matrix projecting $\mathbf{I}_i^A$ onto the motion space of joint i with n_i degrees of freedom:

$$ \mathbf{K}_i = \mathbf{S}_i^T\,\mathbf{I}_i^A\,\mathbf{S}_i $$

Notice that the force propagator (7) has the same dynamic property as the dynamically consistent null space projection matrix for redundant robotic systems [2, 3]. Both quantities guarantee that the resulting (propagated or projected) quantity does not produce any coupling effect in their corresponding motion (operational) space.

A 6×6 inertia matrix, $\mathbf{\Omega}_i$ relates the spatial acceleration of link i and the spatial force of the same link at the joint [8, 4, 5]:

$$ \ddot{\mathbf{x}}_i = \mathbf{\Omega}_i\,\mathbf{f}_i \quad (10) $$

Finally, the 6×6 operational space inertia matrix, $\mathbf{\Lambda}_{e_n}$, of a single open-chain mechanism defined as [2]:

$$ \ddot{\mathbf{x}}_{e_n} = \mathbf{\Lambda}_{e_n}^{-1}\,\mathbf{f}_{e_n} \quad (11) $$

can be related to $\mathbf{\Omega}_n$ using Equations (2), (3), (4), (10), and (11):

$$ \mathbf{\Lambda}_{e_n}^{-1} = {}^n_{e_n}\mathbf{X}^T\,\mathbf{\Omega}_n\,{}^n_{e_n}\mathbf{X} \quad (12) $$

where end-effector frame e_n is at the tip of leaf link n.

3 Efficient Recursive Algorithm

This section describes an efficient recursive algorithm to compute the extended operational space inertia matrix. We will develop this algorithm from the basic analysis of the physical properties of and the relationships among forces, accelerations, and inertia matrices. Also, the proposed algorithm is shown to be of complexity $O(n)$ in practice.

3.1 Analysis of Extended Operational Space Inertia Matrix

The extended operational space inertia matrix, $\mathbf{\Lambda}_e$, of an n-link N-degree-of-freedom branching redundant mechanism with m operational points is defined as [9]:

$$ \mathbf{\Lambda}_e^{-1} = \mathbf{J}_e\,\mathbf{A}^{-1}\,\mathbf{J}_e^T \quad (13) $$

where $\mathbf{\Lambda}_e$ is an $6m \times 6m$ symmetric positive definite matrix, $\mathbf{J}_e$ is the $6m \times N$ Jacobian matrix, and $\mathbf{A}$ is the $N \times N$ joint space inertia matrix. Note that m cannot be greater than n and since each joint can have only up to 6 degrees of freedom, $N = O(n)$ and the size of $\mathbf{A}$ is $O(n^2)$.

As in the case of a single operational point (11), $\mathbf{\Lambda}_e^{-1}$ relates the forces at the end-effectors to the accelerations at the end-effectors:

$$ \ddot{\mathbf{X}}_e = \mathbf{\Lambda}_e^{-1}\,\mathbf{F}_e \quad (14) $$

where $\ddot{\mathbf{X}}_e$ and $\mathbf{F}_e$ are $6m \times 1$ vertically concatenated vectors of the accelerations and forces of each end-effector:

$$ \ddot{\mathbf{X}}_e = \begin{bmatrix} \ddot{\mathbf{x}}_{e_1} \\ \vdots \\ \ddot{\mathbf{x}}_{e_m} \end{bmatrix} \quad \text{and} \quad \mathbf{F}_e = \begin{bmatrix} \mathbf{f}_{e_1} \\ \vdots \\ \mathbf{f}_{e_m} \end{bmatrix} \quad (15) $$

Also, since $\mathbf{\Lambda}_e^{-1}$ is symmetric, it can be expressed in terms of its 6×6 block matrix components as:

$$ \mathbf{\Lambda}_e^{-1} = \begin{bmatrix} \mathbf{\Lambda}^{-1}_{e_1,e_1} & \cdots & \mathbf{\Lambda}^{-1}_{e_1,e_m} \\ \vdots & \ddots & \vdots \\ \mathbf{\Lambda}^{-T}_{e_1,e_m} & \cdots & \mathbf{\Lambda}^{-1}_{e_m,e_m} \end{bmatrix} \quad (16) $$

From Equations (14), (15), and (16), the additive property of the coupling effect on the i^{th} end-effector (of leaf link i) can be written as:

$$\ddot{\mathbf{x}}_{e_i} = \sum_{j=1,\ldots,m} \left[\ddot{\mathbf{x}}^*_{e_i,e_j} \right] \quad (17)$$

$$\ddot{\mathbf{x}}^*_{e_i,e_j} = \mathbf{\Lambda}^{-1}_{e_i,e_j} \mathbf{f}_{e_j} \quad (18)$$

where $\ddot{\mathbf{x}}^*_{e_i,e_j}$ is the coupling acceleration on the i^{th} end-effector by the force of the j^{th} end-effector. This additive property of the coupling effect shows that the resulting acceleration of an end-effector is not only dependent on its own force, but also on the forces of all other end-effectors in the system.

Notice that when the j^{th} end-effector produces the only non-zero force in the system, we can isolate the coupling effect on the i^{th} end-effector by the force of the j^{th} end-effector. This can be written, from Equations (17) and (18), as:

$$\ddot{\mathbf{x}}_{e_i} = \ddot{\mathbf{x}}^*_{e_i,e_j} = \mathbf{\Lambda}^{-1}_{e_i,e_j} \mathbf{f}_{e_j} \quad (19)$$

when $\mathbf{f}_{e_j} \neq 0$ and $\mathbf{f}_{e_k} = 0$ for all $k \neq j$.

Then, similarly to Equation (10), a 6×6 inertia matrix, $\mathbf{\Omega}_{i,j}$ relates the spatial acceleration of link i and the spatial force of the link j at the corresponding joints. This relationship can be written as:

$$\ddot{\mathbf{x}}_i = \mathbf{\Omega}_{i,j} \mathbf{f}_j \quad (20)$$

Also, similarly to Equation (12), the 6×6 block inverse operational space inertia matrix, $\mathbf{\Lambda}^{-1}_{e_i,e_j}$, can be related to $\mathbf{\Omega}_{i,j}$ using Equations (2), (3), (4), (19), and (20):

$$\mathbf{\Lambda}^{-1}_{e_i,e_j} = {}^{i}_{e_i}\mathbf{X}^T \, \mathbf{\Omega}_{i,j} \, {}^{j}_{e_j}\mathbf{X} \quad (21)$$

where end-effector frames e_i and e_j are at the tips of leaf links i and j. Note that this relationship is necessary since the inertial properties are desired at the tips instead of at the joints.

3.2 Derivation of Recursive Algorithm

In this subsection, we will develop a recursive algorithm by separately analyzing the inertial effects of the block diagonal matrices, $\mathbf{\Lambda}^{-1}_{e_i,e_i}$, and of the block off-diagonal matrices, $\mathbf{\Lambda}^{-1}_{e_i,e_j}$ $(i \neq j)$, of $\mathbf{\Lambda}^{-1}_{e}$ in Equation (16).

Each block diagonal matrix, $\mathbf{\Lambda}^{-1}_{e_i,e_i}$, is the inertia matrix that would occur if link i is the only leaf link with an end-effector.

With this physical insight, $\mathbf{\Lambda}^{-1}_{e_i,e_i}$ can be computed using a trivial extension of the Force Propagation Method, an $O(n)$ recursive algorithm to compute the 6×6 inverse operational space inertia matrix of a single open-chain mechanism defined as [8, 4, 5]:

$$\mathbf{\Omega}_i = \mathbf{S}_i \, \mathbf{K}_i^{-1} \, \mathbf{S}_i^T + {}^{h_i}_{i}\mathbf{L}^T \, \mathbf{\Omega}_{h_i} \, {}^{h_i}_{i}\mathbf{L} \quad (22)$$

where link i is the only child link of its parent link h_i and $\mathbf{\Omega}_{root} = 0$.

Using the relationships from (10) and (20), the Force Propagation Method (22) can be extended immediately for a branching robot by replacing $\mathbf{\Omega}_i$ with $\mathbf{\Omega}_{i,i}$. This extension enables the outward recursion to pass through all children instead of a single child:

$$\mathbf{\Omega}_{i,i} = \mathbf{S}_i \, \mathbf{K}_i^{-1} \, \mathbf{S}_i^T + {}^{h_i}_{i}\mathbf{L}^T \, \mathbf{\Omega}_{h_i,h_i} \, {}^{h_i}_{i}\mathbf{L} \quad (23)$$

where h_i is the parent link of link i. Note that this recursion starts from the root link with $\mathbf{\Omega}_{root,root} = 0$ and ends at the leaf links with end-effectors.

Then, the block diagonal matrices, $\mathbf{\Lambda}^{-1}_{e_i,e_i}$, can be computed by transforming $\mathbf{\Omega}_{i,i}$ of leaf links i using Equation (21).

The block off-diagonal matrices, $\mathbf{\Lambda}^{-1}_{e_i,e_j}$ $(i \neq j)$, may be regarded as cross-coupling inertias that are a measure of the inertia coupling to the i^{th} end-effector from the force of the j^{th} end-effector via the *nearest common ancestor* of leaf links i and j. A *nearest common ancestor* of links i and j is the first common link in two paths: one from link i to the root link and the other from link j to the root link. For example, in Figure 1, link h is the nearest common ancestor of leaf links i and j.

From this physical property of block off-diagonal matrices, we can conceptually view $\mathbf{\Lambda}^{-1}_{e_i,e_j}$ as an inertial quantity which propagates the spatial forces from the j^{th} end-effector to link h (the nearest common ancestor of leaf links i and j) and then propagates the resulting spatial accelerations of link h to the i^{th} end-effector.

With this conceptual view, we will develop a recursive algorithm to compute $\mathbf{\Lambda}^{-1}_{e_i,e_j}$ by finding the propagation of the spatial force from the j^{th} end-effector to link h and the propagation of the resulting spatial acceleration from link h to the i^{th} end-effector. Then, $\mathbf{\Lambda}^{-1}_{e_i,e_j}$ can be computed by relating the resulting spatial acceleration of link h to the propagated spatial force from the j^{th} end-effector.

First, using Equations (4), (5), and (7), we can propagate the force $\mathbf{f}_{e_j}$ at the j^{th} end-effector to any of its ancestor h:

$$\mathbf{f}_h = {}^{h}_{j}\mathbf{L}^* \, {}^{j}_{e_j}\mathbf{X} \, \mathbf{f}_{e_j} \quad (24)$$

$${}^{h}_{j}\mathbf{L}^* = \prod_{k} \left[{}^{h_k}_{k}\mathbf{L} \right] \quad (25)$$

where link k is the descendent links of link h in the path from link h to link j and link h_k is the parent link of link k. ${}^{h}_{j}\mathbf{L}^{*}$ results a compound propagation of the spatial force from link j to link h [5].

Similarly, using (3), (6), (8), and (25), the spatial acceleration $\ddot{\mathbf{x}}_h$ of link h can be propagated to the end-effector of any of its descendant leaf link i:

$$\ddot{\mathbf{x}}_{e_i} = {}^{i}_{e_i}\mathbf{X}^T \, {}^{h}_{i}\mathbf{L}^{*T} \, \ddot{\mathbf{x}}_h \tag{26}$$

Now, combining Equations (20), (24), and (26), we can relate $\ddot{\mathbf{x}}_{e_i}$ and $\mathbf{f}_{e_j}$ as:

$$\ddot{\mathbf{x}}_{e_i} = {}^{i}_{e_i}\mathbf{X}^T \, {}^{h}_{i}\mathbf{L}^{*T} \, \mathbf{\Omega}_{h,h} \, {}^{h}_{j}\mathbf{L}^{*} \, {}^{j}_{e_j}\mathbf{X} \, \mathbf{f}_{e_j} \tag{27}$$

Then, from Equations (19), (20), (21), and (27), we can derive $\mathbf{\Omega}_{i,j}$ for the block off-diagonal matrices:

$$\mathbf{\Omega}_{i,j} = {}^{h}_{i}\mathbf{L}^{*T} \, \mathbf{\Omega}_{h,h} \, {}^{h}_{j}\mathbf{L}^{*} \tag{28}$$

where h is the nearest common ancestor of leaf links i and j. Note that the recursive version of Equation (28) is presented in Table 1 (step 4).

As for the block diagonal matrices, the block off-diagonal matrices, $\mathbf{\Lambda}^{-1}_{e_i,e_j}$ $(i \neq j)$, can be computed by transforming $\mathbf{\Omega}_{i,j}$ of leaf links i and j to the corresponding tips using Equation (21).

Finally, we can compute the extended operational space inertia matrix, $\mathbf{\Lambda}_e$, by inverting $\mathbf{\Lambda}^{-1}_e$ in Equation (16). Table 1 summarizes the recursive algorithm developed in this section.

Note that although most processing occurs along the path from the root link to the leaf links with end-effectors, the effects of the other links enter through the articulated-body inertias (9) of the links in the path.

Figure 4 illustrates the recursion processes of the proposed algorithm for the branching robot shown in Figure 1. Arrows indicate the direction of the recursion. Also, there is no computation required among the nodes connected by dotted lines.

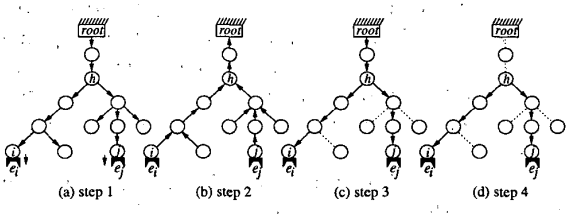

Figure 4: Recursion processes of steps in Table 1: (a) outward recursion for ${}^{h_i}_{i}\mathbf{X}$, (b) inward recursion for ${}^{h_i}_{i}\mathbf{L}$, (c) outward recursion for $\mathbf{\Omega}_{i,i}$, and (d) outward recursion for $\mathbf{\Omega}_{i,j}$ $(i \neq j)$.

Table 1: Recursive algorithm for $\mathbf{\Lambda}_e$.

1. **Outward Recursion:** Compute the spatial transformation matrices:

$$ {}^{h_i}_{i}\mathbf{X} = \begin{bmatrix} {}^{h_i}_{i}R & \mathbf{0} \\ \widehat{{}^{h_i}_{i}r_i} \, {}^{h_i}_{i}R & {}^{h_i}_{i}R \end{bmatrix} \tag{2}$$

2. **Inward Recursion:** Compute the force propagators:

$$ {}^{h_i}_{i}\mathbf{L} = {}^{h_i}_{i}\mathbf{X} \left[\mathbf{1}_6 - \mathbf{S}_i \, \overline{\mathbf{S}}_i \right]^T \tag{7}$$

3. **Outward Recursion:** Compute the block diagonal matrices starting with $\mathbf{\Omega}_{root,root} = 0$:

$$\mathbf{\Omega}_{i,i} = \mathbf{S}_i \, \mathbf{K}_i^{-1} \, \mathbf{S}_i^T + {}^{h_i}_{i}\mathbf{L}^T \, \mathbf{\Omega}_{h_i,h_i} \, {}^{h_i}_{i}\mathbf{L} \tag{23}$$

4. **Outward Recursion:** Compute the block off-diagonal matrices with nearest common ancestor h of links i and j:

$$\mathbf{\Omega}_{i,j} = \begin{cases} \text{return} & \text{if } i = j = h \\ {}^{h_i}_{i}\mathbf{L}^T \, \mathbf{\Omega}^T_{j,h_i} & \text{else if } j = h \\ \mathbf{\Omega}_{i,h_j} \, {}^{h_j}_{j}\mathbf{L} & \text{otherwise} \end{cases} \tag{28}$$

5. **Spatial Transformation:** Compute $\mathbf{\Lambda}^{-1}_{e_i,e_j}$ from $\mathbf{\Omega}_{i,j}$ of leaf links with end-effectors:

$$\mathbf{\Lambda}^{-1}_{e_i,e_j} = {}^{i}_{e_i}\mathbf{X}^T \, \mathbf{\Omega}_{i,j} \, {}^{j}_{e_j}\mathbf{X} \tag{21}$$

6. **Matrix Inversion:** Compute the extended operational space inertia matrix, $\mathbf{\Lambda}_e$, by inverting $\mathbf{\Lambda}^{-1}_e$ (16).

3.3 Computational Complexity

This section presents the computational complexity of the proposed algorithm for an n-link branching mechanism with m operational points. Note that m can be considered as a small constant for any realistic robotic mechanism; $m = O(1)$.

From Table 1, steps 1 and 2 can be computed in $O(n)$ since there are n links in the system. Also, since step 3 requires one outward recursion involving at most n links, all $\mathbf{\Omega}_{i,i}$ can be computed in $O(n)$. In step 4, since there are at most n links to propagate for each of $\frac{m(m-1)}{2}$ $\mathbf{\Omega}_{i,j}$ $(i \neq j)$, all $\mathbf{\Omega}_{i,j}$ can be computed in $O(m^2 n)$. Spatial transformations of $\frac{m(m+1)}{2}$ block matrices cost $O(m^2)$ in step 5. In step 6, an inversion of $\mathbf{\Lambda}^{-1}_e$ $(6m \times 6m)$ requires $O(m^3)$. Therefore, with $m = O(1)$, the overall running time of the proposed algorithm is $O(n)$.

Thus, as the number of links in a mechanism increases, the proposed algorithm performs significantly better than the existing symbolic method [9] which still requires $O(n^3)$ inversion operations for $\mathbf{A}^{-1}$ (13) .

4 Experimental Results

Using the proposed algorithm, we were able to perform the computation of the extended operational space inertia matrix for a complex branching robotic mechanism (Figure 2) in less than 1 msec using a PC with a 266 MHz Pentium II running under the QNX real-time operating system. This branching robot has an operational point at each of its 2 end-effectors and 24 links connected by 1 degree-of-freedom joints.

This result implies that the proposed algorithm enables highly redundant robotic mechanisms such as a human-like robotic mechanism with multiple operational points to be efficiently controlled at a high servo rate in a low-cost hardware environment.

In order to show the effectiveness of the proposed algorithm, we also have controlled this robot under the extended operational space formulation [9] using the proposed algorithm. In the real-time dynamic simulation environment developed in our laboratory, we have achieved a servo rate of 300 Hz with the setup above.

Figure 5 shows the motion sequence when this robot was commanded to put the box on the floor while being advised to keep its self-posture the same as the initial configuration. Notice that the robot had to adjust its advised self-posture in the null space without producing any coupling acceleration at both end-effectors in order not to violate the primary task. This was done automatically without any additional commands.

5 Conclusion

We have proposed an efficient recursive algorithm for the extended operational space inertia matrix of an n-link branching (tree-like) redundant robotic mechanism with m operational points. Since m can be considered as a small constant in practice, we obtain the linear running time of $O(n)$ for this algorithm.

Therefore, as the complexity of a robotic mechanism increases, the proposed algorithm performs significantly better than the traditional $O(n^3)$ symbolic method. The real-time simulation results with a complex redundant robotic mechanism ($n = 24$, $m = 2$) illustrate the efficiency of this algorithm.

Acknowledgments

Many thanks to Oliver Brock, Bob Holmberg, Oscar Madrigal, Diego Ruspini, and Machiel Van der Loos for their comments and support during the preparation of this manuscript.

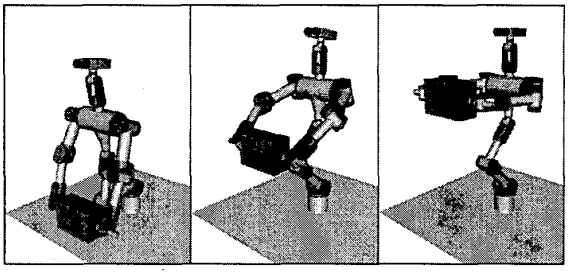

Figure 5: Sequence of putting a box on the floor.

References

[1] R. Featherstone. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, 1987.

[2] O. Khatib. A unified approach to motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, RA-3(1):43–53, February 1987.

[3] O. Khatib. The impact of redundancy on the dynamic performance of robots. *Laboratory Robotics and Automation*, 8:37–48, 1996.

[4] K. Kreutz-Delgado, A. Jain, and G. Rodriguez. Recursive formulation of operational space control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1750–1753, April 1991.

[5] K. W. Lilly. *Efficient Dynamic Simulation of Robotic Mechanisms.* Kluwer Academic Publishers, 1992.

[6] K. W. Lilly and D. E. Orin. Efficient O(n) recursive computation of the operational space inertia matrix. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(5):1384–1391, September/October 1993.

[7] B. V. Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems.* PhD thesis, University of California at Berkeley, Berkeley, California, U.S.A., Fall 1996.

[8] G. Rodriguez, K. Kreutz, and A. Jain. A spatial operator algebra for manipulator modeling and control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1374–1379, May 1989.

[9] J. Russakow, O. Khatib, and S. M. Rock. Extended operational space formulation for serial-to-parallel chain (branching) manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1056–1061, Nagoya, Japan, May 1995.