24th International Symposium on Automation & Robotics in Construction (ISARC 2007)
Construction Automation Group, I.I.T. Madras

A GENERIC FRAMEWORK FOR CONTROLLING BI-HANDED ROBOTIC MANIPULATORS

Jayakrishnan T
Country Manager
Energid Technologies
12/193 B1, Moolepadam Road
Vazhakkala, Kakkanad West P.O
Kochi-682030, Kerala, India
jk@energid.com

James D English
Chief Technical Officer
Energid Technologies
124 Mount Auburn Street
Cambridge, MA 02138
U.S.A
jde@energid.com

Neil Tardella
Chief Operating Officer
Energid Technologies
124 Mount Auburn Street
Cambridge, MA 02138
U.S.A
nmt@energid.com

ABSTRACT

From design to implementation, bi-handed humanoid manipulators are more complex than common industrial robots. Humanoid manipulators have the properties of redundancy and bifurcation, which enable placement of multiple hands or tools at desired positions and orientations within the robot's workspace in an unlimited number of ways. To address and benefit from these complexities, Energid Technologies has developed a generic XML-based control method supporting virtually any number of degrees of freedom, any type of end-effector constraint, any type of joint, and any optimization criterion for fixed and mobile robots. This paper will address how this technology works, the problems it solves, its ongoing application as a software toolkit accessible to the robotics community.

KEYWORDS

Branching Robot, Kinematics, Control, Simulation

1. INTRODUCTION

Robots have had a presence in constructional engineering for decades. Today, robots are contributing to the production of goods through applications such as semi automated earthmoving, material handling, and assembly [1]. Robotic systems not only help in improving productivity, but also make it possible to work in hazardous environments and extreme situations (such as during natural-calamities).

Recently, humanoid robots have risen in prominence, and these have potential to stand in for human workers in a variety of construction-related activities [1]. The control of humanoid robots is challenging, however. As the application of robotics in construction increases, it becomes important to look into a common control method to suit changing robotic shapes and configurations.

2. COMMON SOFTWARE FRAMEWORK

The robot-control framework we describe here incorporates the kinematic, dynamic, and physical-extent properties of the manipulator components, and it allows flexible specification of velocity control algorithms in a language based on the Extensible Markup Language (XML). This framework can control and simulate any number of cooperating manipulators. The primary requirement for controlling a manipulator is knowledge of its kinematic and dynamic properties.

3. THE LINK-BASED MODEL

A bi-handed manipulator is represented in software through a tree/branching structure topology as shown in Figure 1 below.

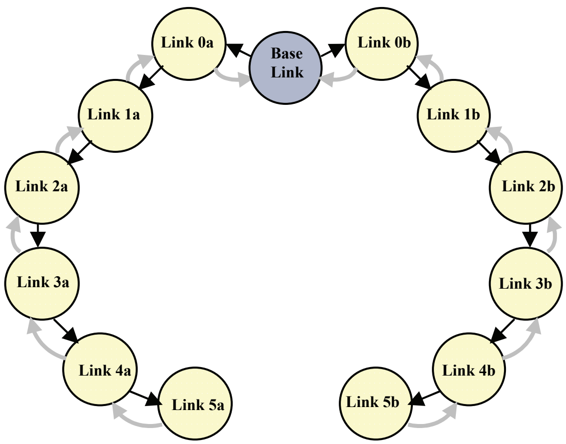

Figure 1 Manipulator Link Model

The manipulator is composed of links in a branching structure, where each link moves relative to its parent in the tree. Programmatically, each link has access to all of its children and to its parent.

With this established topology, the burden of defining the manipulator is equivalent to defining each of the individual links.

4. LINK KINEMATICS AND DYNAMICS

The link model uses three reference frames. The proximal D-H frame, as we will call it, is rigidly attached to the parent. The distal D-H frame aligns with the proximal D-H frame of the children. The rigid body describing the link moves with respect to the distal D-H frame of its parent when the joint value of the link changes. Each link also has a special frame, the primary frame. The primary frame is rigidly attached to the distal D-H frame, but with a fixed offset. It is used to define link properties such as mass and physical extent. It is also used as a reference when defining end effectors. These frames are illustrated in Figure 2 below.

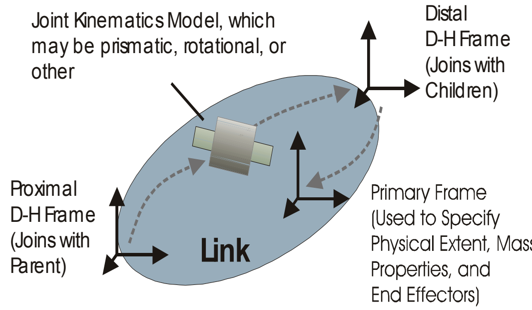

Figure 2 The Link Reference Frames

The proximal and distal D-H frames are so named because they correspond to the Denavit-Hartenberg (D-H) frames when this formalism is used for describing the joints

Energid's toolkit supports the two most common D-H approaches, as found in robotics textbooks. The first is Paul's Denavit-Hartenberg notation [2], which uses the kinematic sequence {z-rotation, z-translation, x-rotation, x-translation}. The second is Craig's notation [3], using the kinematic sequence {x-rotation, x-translation, z-rotation, z-translation}.

In addition to the Denavit-Hartenberg representation, support is included for general axes of rotation and translation. These options allow utmost flexibility in describing links with rotational or prismatic joints. Listing 1 below shows part of the XML description for the Cyton A7D-1G (Energid's humanoid manipulator) configuration.

This representation of robot link information is conceptually different from the traditional representation. Traditionally, robotic links are connected with joints—that is, the links lie between the joints. In this formulation, the link conceptually contains the joint. The distal frame of one link is rigidly attached to the proximal frame of a child. This provides a more flexible way to represent kinematic properties. It allows multiple formalisms (such as Paul or Craig's Denavit-Hartenberg notation) to be used internally to the link. It also supports the representation of new types of joints.

Listing 1 XML from the Cyton A7D-1G Arm Description

```
  <mn:linkKinematicsContainer>
- <mn:denavitHartenberg>
  <mn:a>0</mn:a>
  <mn:alpha>0</mn:alpha>
  <mn:d>0</mn:d>
  <mn:dhType>generalPaul</mn:dhType>
  <mn:jointType>rotational</mn:jointType>
- <mn:precursor>
  <mn:orient q0="1" q1="0" q2="0" q3="0"
      />
  <mn:translation x="0" y="0" z="0" />
     </mn:precursor>
- <mn:primaryFrame>
  <mn:orient q0="1" q1="0" q2="0" q3="0"
      />
  <mn:translation x="0" y="0" z="0" />
     </mn:primaryFrame>
  <mn:theta>0</mn:theta>
     </mn:denavitHartenberg>
   <mn:firstMoment x="0" y="0" z="0" />
  <mn:mass>1.180</mn:mass>
- <mn:secondMoment>
  <mn:jXX>0.02</mn:jXX>
  <mn:jXY>0</mn:jXY>
  <mn:jXZ>0</mn:jXZ>
  <mn:jYY>0.01</mn:jYY>
  <mn:jYZ>0</mn:jYZ>
  <mn:jZZ>0.01</mn:jZZ>
     </mn:secondMoment>
     </mn:massProperties>
- <mn:shapeContainer>
- <mn:trianglePhysicalExtent>
 <mn:bypassDynamics>0</mn:bypassDynami
      cs>
  <mn:canDeform>0</mn:canDeform>
  <mn:deformable>0</mn:deformable>
  <mn:detailLevel>3</mn:detailLevel>
```

To describe link kinematics (position, velocity, and acceleration), the algorithm supports general joint types with one degree of freedom. Thus, not only are rotational and prismatic joints supported, but also screw-type joints, motion along a rail, and any other type of motion that can be parameterized by a scalar.

5. CORE ALGORITHMIC FRAMEWORK

Once a robotic mechanism is defined as described in the previous section, the generic software framework is capable of implementing and exchanging a variety of velocity control algorithms for all types of kinematically redundant, bifurcating manipulators.

The core velocity framework is based on the manipulator Jacobian equation:

$$\mathbf{V} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \tag{1}$$

Here, $\mathbf{V}$ is an *m*-length vector representation of the motion of one or more hands (usually some combination of linear and angular velocity referenced to points rigidly attached to parts of the manipulator); $\mathbf{q}$ is the *n*-length vector of joint positions (with $\dot{\mathbf{q}}$ being its time derivative); and $\mathbf{J}$ is the $m \times n$ manipulator Jacobian, a function of $\mathbf{q}$. For traditional industrial robots, **V** is often formed by combining the linear and angular velocity of a single tool or hand. A key point for bi-handed humanoid manipulators is that **V** takes on a larger meaning—it includes the concatenation of point, frame, or other motion of multiple end-effectors.) This is illustrated in Figure 3 below.

Figure 3 shows Energid's Cyton A14D-2G, 14DOF humanoid Bi-handed manipulator, and the illustration of the parameters for velocity control. The column vector V represents the combined motion of both hands (for positioning and orienting both hands, it would be 12×1), and $\mathbf{q}$ represents the concatenated joint values (14×1). The Jacobian $\mathbf{J}(\mathbf{q})$ is the matrix that makes (1) true for all possible values of $\dot{\mathbf{q}}$. Note in the toolkit $\mathbf{V}$ can represent a concatenation of other types of end-effectors as well.

For any physical manipulator that is not self-connecting, a manipulator Jacobian can be defined to make equation (1) true. When the manipulator is kinematically redundant, the dimension of $\mathbf{V}$ is less than the dimension of q ($m<n$), and (1) is underconstrained when $\mathbf{V}$ is specified. By using V to represent relative motion, (1) can also support self-connecting mechanisms by setting the relative motion to zero.

The velocity control question, then, is the following: given a desired hand motion V, what are the joint rates $\dot{\mathbf{q}}$ that best achieve this motion? To answer this, the framework is built on a method which uses a scalar α, a general column vector $\mathbf{F}(\mathbf{q})$, and a matrix function to solve for $\dot{\mathbf{q}}$ given $\mathbf{V}$ through the following formula:

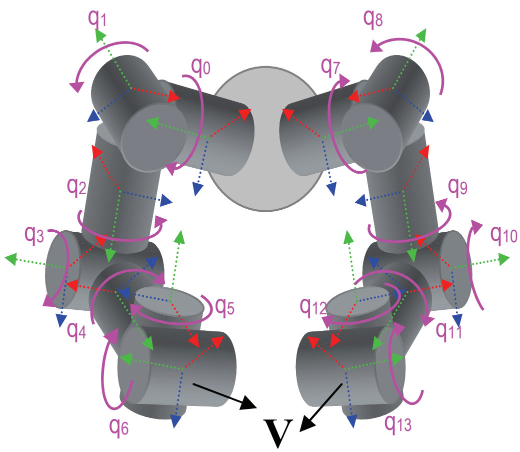

Figure 3 Configuration of Cyton A14D-2G

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J} \\ \mathbf{N}_\mathbf{J}^T \mathbf{W} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{V} \\ -\alpha \mathbf{N}_\mathbf{J}^T \mathbf{F} \end{bmatrix}. \quad (2)$$

where $\mathbf{N_J}$ is an $n \times (n-m)$ set of vectors that spans the null space of $\mathbf{J}$. That is, $\mathbf{JN_J} = 0$, and $\mathbf{N_J}$ has rank $(n-m)$. $\mathbf{N_J}$ is generally a function of $\mathbf{q}$. By changing the values of α, $\mathbf{W}$, and f, many new and most established velocity-control techniques can be implemented [4].

Equation (2) is the core velocity-control algorithm used by the frame work. Mathematically, it achieves the desired $\mathbf{V}$ while minimizing $\frac{1}{2}\dot{\mathbf{q}}^T \mathbf{W} \dot{\mathbf{q}} + \alpha \mathbf{F}^T \dot{\mathbf{q}}$. The parameters α, $\mathbf{W}$ and $\mathbf{F}$ can be defined using XML to give many different types of velocity control, including pseudoinverse control [5], weighted pseudoinverse control [6], augmented Jacobian techniques [7], [8], extended Jacobian techniques [9], [10], and projection methods [11], [12].

6. LOCAL POSITION CONTROL

The local position control uses the following procedure: The current pose of an end-effector (for example location or location and orientation for point and frame end effectors, respectively) and the desired pose are used to construct an end-effector velocity that, if followed, will give alignment.

For point end effectors, the desired velocity of the point is simply a scalar gain times the difference in position. That is, if $\vec{p}_a$ is the actual position and $\vec{p}_d$ is the desired position (both with respect to the system frame), then the desired velocity, $\vec{v}_d$, is given by the following, where k_ℓ is a user-specified positive gain.

$$\vec{v}_d = k_\ell \cdot (\vec{p}_d - \vec{p}_a). \quad (3)$$

The similar calculation for three-dimensional frames is more complicated. For this, the velocity is calculated as specified in equation (3) and in addition an angular velocity is calculated as a vector proportional to the minimum rotational angle needed to align the actual orientation with the desired orientation. Every orientation can be expressed as an axis of rotation and an angle of rotation about that axis. For the rotation between the actual and the desired frame, let this axis be $\hat{u}$ (having unit norm) and let the angle of rotation be θ. Then the desired angular velocity is given by the following, where k_a is a user-specified positive gain:

$$\vec{\omega}_d = k_a \cdot \theta \cdot \hat{u}. \quad (4)$$

In the framework, the angular and linear velocities calculated through (3) and (4) are also capped at a user-defined threshold. The points used in calculating (3) are specified through a user-defined offset. In simulation or for path finding, the actual hand position and orientation values are calculated using ground truth, but in a feedback-based fielded application, they are calculated from sensed joint positions or found directly using visual or other sensors.

7. EXAMPLE

As one example of the type of manipulator that can be controlled using Energid's framework, Figure 4 shows NASA's Robonaut manipulator. This is a robot that can be used for construction, albeit in space. The real robot is shown at the top and a simulated version being controlled and simulated is shown below. This model has 57 degrees of freedom in motion, including two 7-dof arms, two 19-dof hands, a 3-dof waist, and a 2-dof neck and head. The method described in this paper can be used to control this in real time on a laptop PC.

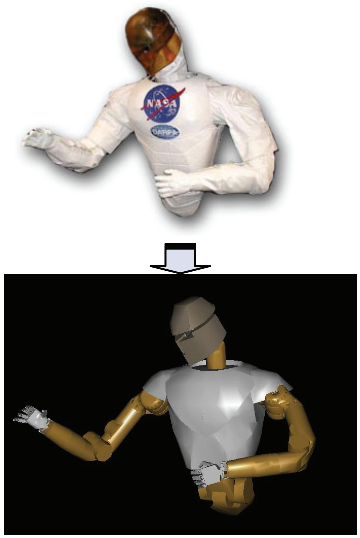

Figure 4 Simulation of Robonaut

Figure 4 shows the result of simulation done for NASA's Robonaut, a bi-handed space-based construction and repair robot with 57 degrees of freedom in motion.

8. CONCLUSIONS

A comprehensive and generic method to control virtually any type of bi-handed robotic mechanisms has been presented. This technology has direct application to robots used to support construction, with its rich diversity of tasks. The proposed method supports single and cooperating robots and the selection of end effectors and tools according to the type of material handled or job performed.

The proposed approach supports the optimization of secondary criteria, such as collision avoidance that enables robotic operation in dynamic work environments. It also supports remote control for hazardous or remote missions. The framework is ideal for automating construction and support operations.

9. REFERENCES

[1] Innovation in Construction by Automation and Robotics, T. Bock, IAARC Newsletter, Issue 5, Second Period, April 2006.

[2] R.P. Paul (1981), Robotic Manipulators: Mathematics, Programming, and Control, Cambridge, MA: MIT Press, 1981

[3] J.J. Craig (1989), Introduction to Robotics, Reading, MA: Addison-Wesley, 1989.

[4] J.D. English and A.A. Maciejewski (2000) On the Implementation of Velocity Control for Kinematically Redundant Manipulators, IEEE Trans. on Sys., Man, and Cybernetics—Part A: Systems and Humans, vol. 30, no. 3, May 2000, pp. 233-237.

[5] C.A. Klein and C.H. Huang (1983), Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators, IEEE Trans. on Sys., Man, and Cybernetics, vol. SMC-13, pp. 245-250, Mar./Apr. 1983.

[6] K.L. Doty, C. Melchiorri, E.M. Schwartz, and C. Bonivento, (1995), Robot Manipulability, IEEE J. Robot. Automat., vol. 11, pp. 462-468, June 1995.

[7] Egeland,(1987), Task Space Tracking with Redundant Manipulators, IEEE J. Robot. Automat., vol. RA-3, pp. 471-475, Oct. 1987.

[8] H. Seraji(1989), Configuration Control of Redundant Manipulators: Theory and Implementation, IEEE Trans. Robot. Automat., vol. 5, pp. 472-490, Aug. 1989.

[9] J. Baillieul 1985), Kinematic Programming Alternatives for Redundant Manipulators, Proc. 1985 IEEE Int. Conf. Robot. Automat., St. Louis, MO, Mar. 25-28, 1985, pp. 722-728.

[10] H. Seraji and R. Colbaugh (1990), Improved Configuration Control for Redundant Robots, J. Robot. Syst., vol. 7, no. 6, pp. 897-928, 1990.

[11] M.Z. Huang and H. Varma (1991), Optimal Rate Allocation in Kinematically Redundant Manipulators—The Dual Projection Method, Proc. 1991 IEEE Int. Conf. Robot. and Automat., Philadelphia, PA, Apr 24-29, 1988, pp28-36.

[12] H. Zghal, R.V. Dubey, and J.A. Euler (1990), An Efficient Gradient Projection Optimization for Manipulators with Multiple Degrees of Redundancy, Proc. IEEE 1990 Int. Conf. Robot. and Automat., Cincinnati, OH, May 13-18, 1990, pp. 1006-1011.