2

Automatic Modeling for Modular Reconfigurable Robotic Systems – Theory and Practice

I-Ming Chen, Guilin Yang and Song Huat Yeo

1. Introduction

A modular reconfigurable robot consists of a collection of individual link and joint components that can be assembled into a number of different robot geometries. Compared to a conventional industrial robot with fixed geometry, such a system can provide flexibility to the user to cope with a wide spectrum of tasks through proper selection and reconfiguration of a large inventory of functional components. Several prototyping systems have been demonstrated in various research institutions (Cohen et al. 1992; Fukuda & Nakagawa 1988; Schmitz, et al. 1988; Wurst 1986). Applications of modular systems have been proposed in rapid deployable robot systems for hazardous material handling (Paredis et al. 1995), in space stationed autonomous systems (Ambrose 1995), and in manufacturing systems (Chen 2000; 2001).

In the control and simulation of a modular reconfigurable robot system, precise kinematic and dynamic models of the robot are necessary. However, classical kinematic and dynamic modelling techniques for robot manipulators are meant for robot with fixed geometry. These models have to be derived manually and individually stored in the robot controller prior to simulating and controlling the robot. Commercial robot simulation software usually provides end users with a library of predefined models of existing robots. The models of any new robot not in the library have to be derived exclusively from the given parameters and commands in the package. For a modular robot system built upon standard modular components, the possible robot geometries and degrees of freedom become huge. As shown by Chen (1994), the number of robot-assembly configurations grows exponentially when the module set becomes large and the module design becomes complicated. To derive all of these models and store them as library functions require not only tremendous effort but also very large amount of disk storage space. In such cases, it is impractical and almost impossible to obtain the kinematic and dynamic models of a robot based on the fixed-geometry approach. Hence, there is a need to develop an automatic model-generation technique for modular robot applications.

Source: Industrial-Robotics-Theory-Modelling-Control, ISBN 3-86611-285-8, pp. 964, ARS/plV, Germany, December 2006, Edited by: Sam Cubero

In this chapter, we introduce a framework to facilitate the model-generation procedure for the control and simulation of modular robots. The framework consists of three parts: a component database; a representation of modular robot geometry; and geometry-independent modelling techniques for kinematics, dynamics, and calibration. The component database maintains the description and specifications of standard robot components, such as actuators, rigid links, sensors, and end effectors. The robot representation indicates the types and orders of the robot components being connected. The geometry-independent modelling algorithms then generate the proper models based on the robot description.

A graph based technique, termed the *kinematic graph*, is introduced to represent the module-assembly sequence and robot geometry. In this graph, a node represents a connected joint module and an edge represents a connected link module. Modules attached to or detached from the robot can be indicated by adding or removing nodes or edges from the graph. The realization of this graph is through an *Assembly Incidence Matrix* (AIM) (Chen 1994; Chen & Burdick 1998). A modular robot can be conceived according to the given AIM without knowing the other parameters, such as joint angles and initial positions. Here, we assume the generic structure of a modular robot is branch-type. The serial type modular robot is a special case of the branch-type structure.

Previous attempt to deal with automatic model generation for modular robots employed Denavit-Hartenburg (D-H) parameterization of the robot (Kelmar & Khosla 1988; Benhabib et al. 1989). However, the D-H method does not provide a clear distinction between the arranging sequence of the modules in the robot chain and their spatial relationship. Also, it depends on the initial position of the robot: the same robot may have different sets of D-H parameters just because of the different initial or zero positions. When evaluating the task performance of a modular robot with respect to its corresponding geometry, complicated equivalence relationships must be defined on the sets of parameters to identify the uniqueness of the robot geometry (Chen & Burdick 1998).

The formulation of the kinematics and dynamics is based on the theory of Lie groups and Lie algebras. The robot kinematics follows a local representation of the product-of-exponential (POE) formula, in which the joints, regardless of the types, are defined as members of *se(3)*, the Lie algebra of the Euclidean group *SE(3)*. The associated Lie algebraic structure can simplify the construction of the differentials of the forward-kinematic function required for numerical inverse solutions. The POE representation can also avoid the singularity conditions that frequently occur in the kinematic calibration formulated by the D-H method (Chen & Yang 1997; Chen et al. 2001). Thus, it provides us with a uniform and well-behaved method for handling the inverse kinematics of both calibrated and uncalibrated robot systems. Since the joint axes are described in the local module (body) coordinate systems, it is convenient for progressive

construction of the kinematic models of a modular robot, as it resembles the assembling action of the physical modular robot components. The formulation of the dynamic model is started with a recursive Newton-Euler algorithm (Hollerbach 1980; Rodriguez et al. 1991). The generalized velocity, acceleration, and forces are expressed in terms of linear operations on se(3) (Murray et al. 1994). Based on the relationship between the recursive formulation and the closed-form Lagrangian formulation for serial-robot dynamics discussed in (Featherstone 1987; Park et al. 1995), we use an *accessibility* matrix (Deo 1974) to assist in the construction of the closed-form equation of motion of a branch-type modular robot, which we assume is the generic topology of a modular robot. Note that all the proposed modelling techniques can contend with redundant and nonredundant modular robot configurations.

This chapter is organized as follows. Section 2 introduces the basic features of the hardware and software of a newly conceived modular robotic workcell. Section 3 briefly reviews the definitions of the AIM presentation and the associated accessibility matrix and path matrix. Section 4 concerns the formulation and implementation of geometry-independent kinematic, dynamic, and calibration models for modular robots. In addition to automated model generation, identification of the optimal modular robot assembly geometry for a specific task from the vast candidate database is also important. The AIM representation facilitates the search/optimization process by using the genetic algorithms approach. Section 5 investigates the task-oriented optimal geometry issues in modular reconfigurable robots and the advantage of using AIM to solve this type of problem. The proposed automatic model-generation method implemented in a Windows Based application for modular robotic automation system, termed *SEMORS* (Simulation Environment for MOdular Robot System) is introduced in Section 6. Prototypes of the modular robotic automation systems configured in both serial and parallel geometries for positioning and machining purposes based on the operation of *SEMORS* are illustrated in Section 7. This chapter is concluded in Section 8.

2. System Architecture

Figure 1 illustrates the system layout of a reconfigurable robotic workcell proposed by Nanyang Technological University and Singapore Institute of Manufacturing Technology (Chen 2001). The objective of this project is to develop a reconfigurable modular robotic workcell which is capable of performing a variety of tasks, such as part assembly, material transfer, and light machining (grinding, polishing and deburring), through rapid change of reusable workcell components. In this system, workcells are made of standard interchangeable modular components, such as actuators, rigid links, end-of-arm tooling, fixtures, and sensors. These components can be rapidly assembled and config-

ured to form robots with various structures and degrees of freedom. The robots, together with other peripheral devices, will form a complete robotic workcell to execute a specific manufacturing task or process. The corresponding intelligent control and simulation software components are then reconfigured according to the change of the workcell configuration. The maintenance and upgrade of the system are simplified by replacing the malfunctioned or outdated components. Converting a manufacturing line from one product to another can be very fast in order to keep up with the rapidly changing marketplace.

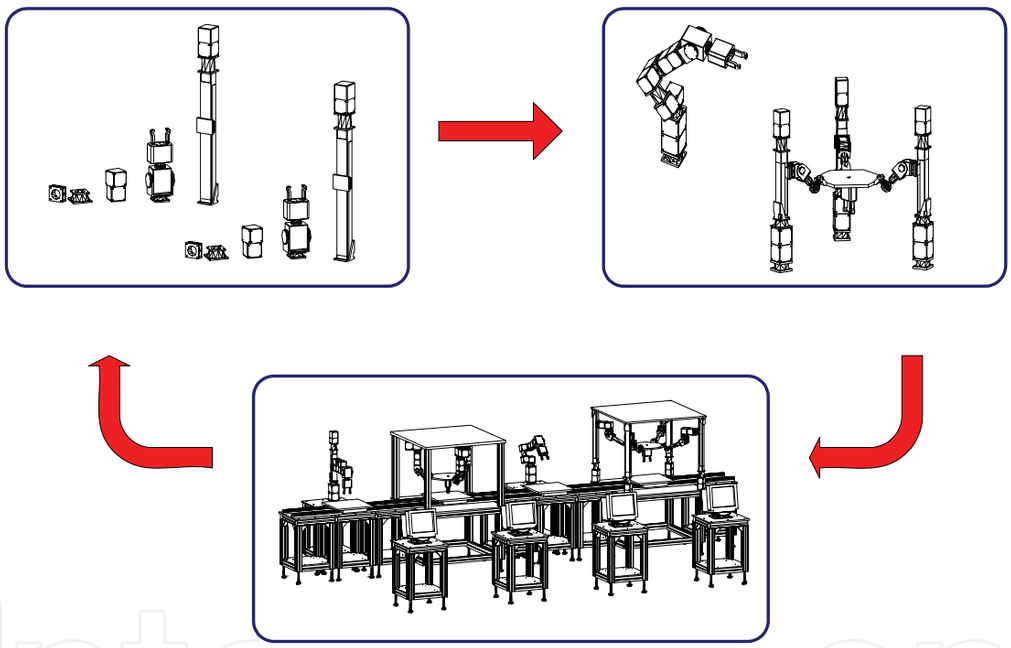


Figure 1. Deployment of a reconfigurable robotic workcell

In this system, the workcell software is designed in reusable- and reconfigurable-object fashion for ease of maintenance and development. Figure 2 illustrates the overall software architecture of the modular workcell. The user environment will provide all the necessary functions to facilitate the end user in controlling, monitoring and simulating the workcell. It consists of the following parts:

Component browser –- for viewing and editing the components available in the component database;

- *Simulator* --- for generating a computer-simulation model of a modular robot and the entire workcell; additionally, the simulator may be employed as the core function for future virtual manufacturing capabilities;
- *Task level planner* --- for determining the optimal geometry of a modular robot for a given task and the overall layout of the workcell for a particular manufacturing process;
- *Programming interface* --- for providing command and control of the system; and
- *Controller* --- for commanding the low-level individual controllers located in the components, and identifying the robot's geometry from the local component controllers.

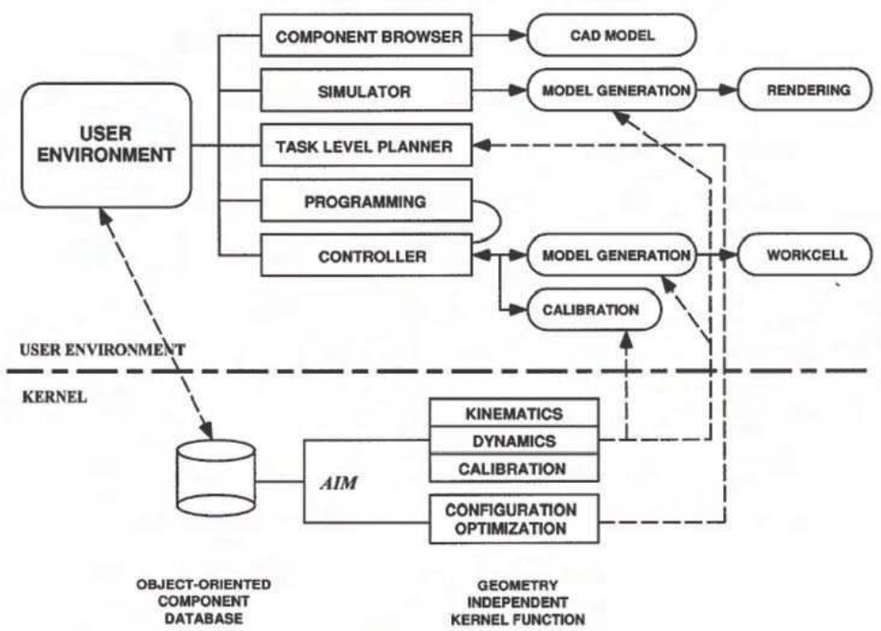


Figure 2. Software architecture for reconfigurable workcell

The system kernel, which is hidden from the user, provides automated model-generation functions and the configuration-optimization function (a component database is also associated with it):

- *Object-oriented component database*----manages the specification of all the components, such as the dimensions and weights of the links, maximum kinematic and dynamic performance of the actuators, etc. It can be accessed by the user for browsing and editing purposes.

- *Geometry-independent kernel functions*---generates kinematic and dynamic models of the robots shared by the simulators and the controller. Using identical models in the simulation and control of the workcell insures the reliability and integration of the system, and enables physically based simulations through the workcell controller. The configuration-optimization function can enumerate all possible robot geometry from an inventory of module components in the database, and select the most suitable one for a prescribed task. This information will pass back to the task-level planner to determine the optimal layout and locations of the robots in the workcell.

The information passing from the component database to the modeling functions is through the assembly incidence matrix. Robot geometries (serial, branch, or hybrid) and detailed connection information, such as the connecting orientation and the types of adjacent modules, are all indicated in the matrix. This matrix is then passed to the geometry-independent functions for model generation.
In such a system, the need to maintain a huge library of robot models is eliminated; instead, we maintain a small selection of the component-database and kernel functions for automated model generation, reducing the overall footprint of the system software.

3. Modular Robot Representation

3.1 Module Representation

To make the automatic model-generation algorithms work on a variety of module components, we introduce a conceptual set of modules whose features are extracted from those of real implementations. The modular systems developed to date have several common mechanical and structural features: (1) only 1-DOF revolute and 1-DOF prismatic joints; (2) symmetric link geometries for interchangeability; and (3) multiple connection ports on a link.

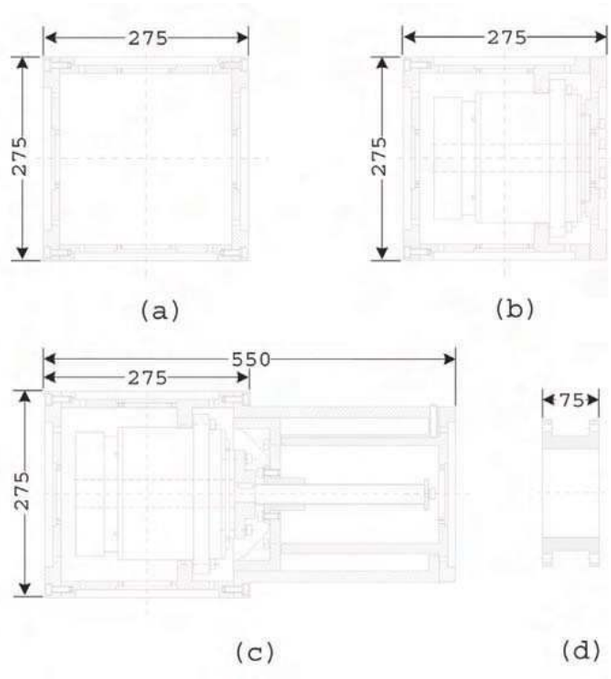


Figure 3. Modular robot components

3.1.1 Joint Modules

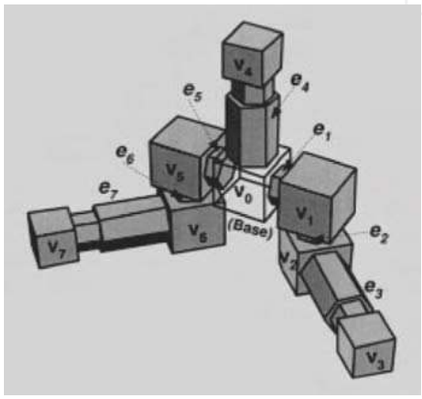
A modular robot joint module is an ``active'' joint, which allows the generation of a prescribed motion between connected links. Two types of joint modules, the revolute joints (rotary motion) and the prismatic joints (linear or translational motion), are considered. Rotary and linear actuators must reside in the modules to produce the required motions and maintain the modularity of the system. Multi-DOF motions can be synthesized with several 1-DOF joints. Joint modules are attached to link modules through standardized connecting interfaces for mechanical, power, and control connections.

3.1.2 Link Modules

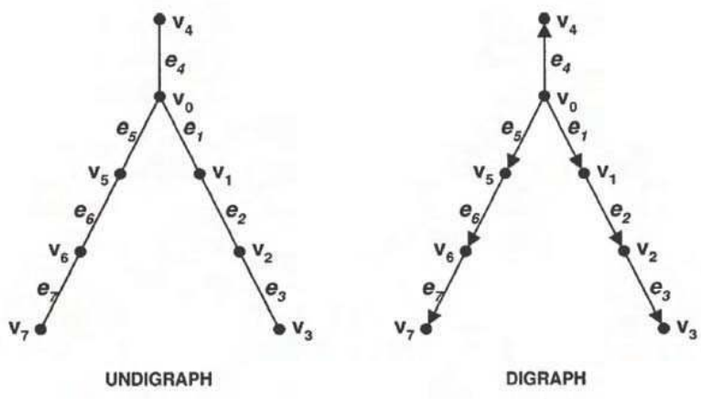
The place on a link module where the joint is connected is called a connecting port. Without loss of generality, we assume that a link module is capable of multiple joint connections, and the link module has symmetrical geometry. Such a design allows modules to be attached in various orientations, and the robot geometry to be altered by simple reassembling. The modular robot components developed in our university are shown in Figure 3. This design follows the building-block principle whereby modules can be stacked together in various orientations through connecting points on all six faces of the cubes.

3.2 Assembly Incidence Matrix

Definition 1. (Graph)
A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *consists of a vertex set,* $\mathcal{V} = \{\mathbf{v}_0, \cdots, \mathbf{v}_n\}$, *and an edge set,* $\mathcal{E} = \{\mathbf{e}_0, \cdots, \mathbf{e}_m\}$, *such that every edge in* $\mathcal{E}$ *is associated with a pair of vertices, i.e.,* $\mathbf{e}_i = (\mathbf{v}_j, \mathbf{v}_k)$.



(a)



(b)

Figure 4. (a) A branching modular robot; (b) kinematic graphs of the robot

In mechanism design theory, a kinematic chain of links and joints is often represented by a graph, termed a *kinematic graph* (Dobrjanskyj & Freudenstein 1967), in which vertices represent the links and edges represent the joints. Using this graph representation, we are able to categorize the underlying structure (or geometry) of a linkage mechanism and apply the result from the graph theory to enumerate and classify linkage mechanisms. A robot

manipulator is also a kinematic chain, thus, admitting a kinematic graph representation. For example, an 8-module 7-DOF branch-type modular robot and its kinematic graphs are shown in Figure 4(a) and 4(b). It is also known that a graph can be represented numerically as a *vertex-edge incidence matrix* in which the entries contain only 0s and 1s (Deo 1974). Entry (i, j) is equal to 1 if edge $\mathbf{e}_j$ is incident on vertex $\mathbf{v}_i$, otherwise, it is equal to zero. This incidence relationship defines the connectivity of the link and joint modules. Because link modules may have multiple connecting points, we can assign labels to the connecting points to identify the module connection. To further identify those connections in the incidence matrix, we can replace those entries of 1 by the labels of the connected ports being identified on the link modules, and keep those entries of 0 unchanged. This modified matrix, termed an *assembly incidence matrix*, provides us the necessary connection information of the modules and also the basic geometry of the modular robot.

Definition 2. (Assembly incidence matrix)
Let $\mathcal{G}$ *be a kinematic graph of a modular robot and* $\mathcal{M}(\mathcal{G})$ *be its incidence matrix. Let* **port** *be the set of labels assigned to the connecting ports on the link modules. The assembly incidence matrix of the robot* $\mathcal{A}(\mathcal{G})$ *is formed by substituting the 1s in* $\mathcal{M}(\mathcal{G})$ *with labels in* **port** *on respective modules. One extra column and row are augmented to* $\mathcal{A}(\mathcal{G})$ *to show the types of link and joint modules.*

Note that the representation and assignment of the labels are nonunique. The labels of the connecting ports may be numerical values (Chen 1994) or may be derived from the module coordinates (Chen & Yang 1996). In this case, the module-component database should use consistent bookkeeping for this information. The AIM of the modular robot (8 link modules and 7 joint modules) shown in Fig. 4 is a 9×8 matrix:

$$\mathcal{A}(\mathcal{G}) = \begin{bmatrix} 1 & 3 & 5 & 0 & 0 & 0 & 0 & B \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & C1 \\ 0 & 1 & 0 & 6 & 0 & 0 & 0 & C1 \\ 0 & 0 & 2 & 0 & 6 & 0 & 0 & C1 \\ 0 & 0 & 0 & 5 & 0 & 2 & 0 & C2 \\ 0 & 0 & 0 & 0 & 5 & 0 & 3 & C2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & C2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & C2 \\ P & R & R & R & R & P & P & 0 \end{bmatrix}. \tag{1}$$

Note that there are three types of link modules in the robot: the base (B), the large cubic module ($C1$), and the small cubic module ($C2$). Cubic modules have six connecting interfaces labeled 1 – 6; i.e., **port** $= \{1,\cdots,6\}$, which follows the labeling scheme on dice. The revolute joints and prismatic joints are denoted by R and P respectively.

3.3 Accessibility Matrix and Path Matrix

Two matrices, namely the accessibility matrix and the path matrix, derived from a given AIM are defined in this section to provide the accessibility information from the base module to every pendant module in a branch-type modular robot. The accessibility information enables us to formulate the kinematics and dynamics of a general branch-type robot in a uniform way.

3.3.1 Module traversing order

The links and joints of a serial-type robot can follow a natural order from the base to the tip. A branch-type robot has more than one tips, and no loops. Therefore, the order of the links of a branch-type robot depends on the graph traversing algorithms (Cormen et al. 1990). Let $\mathcal{G}=(V,\mathcal{E})$ represent the kinematic graph of a branch-type modular robot with $n+1$ link modules, where $V=\{\mathrm{v}_0,\mathrm{v}_1,...\mathrm{v}_n\}$ represents the set of modules. The fixed base module is denoted by $\mathbf{v}_0$ and is always the starting point for the traversing algorithm. The rest modules are labeled by their traversing orders i. The traversing orders of the links in the robot of Figure 4(a) are indicated by the numbers on the vertices of the graph of Figure 4(b). This order is obtained by the depth-first-search algorithm. Note that the farther the module is away from the base, the larger its traversing order.

3.2.2 Directed graphs

A branch-type robot with $n+1$ modules has n joints. Let $\mathcal{E}=\{\mathbf{e}_1,\cdots,\mathbf{e}_n\}$ represents the set of joints, where joint $\mathbf{e}_i$ is designated as the connector preceding link module $\mathbf{v}_i$. With a given traversing order, the robot graph $\mathcal{G}$ can be converted to a directed graph (or digraph) $\vec{\mathcal{G}}$, which is an outward tree for a branch-type manipulator in the following manner. Let $\mathbf{e}_j=(\mathbf{v}_i,\mathbf{v}_j)$ be an edge of the graph $\vec{\mathcal{G}}$ and $i<j$. An arrow is drawn from $\mathbf{v}_i$ to $\mathbf{v}_j$ as edge $\mathbf{e}_j$ leaves vertex $\mathbf{v}_i$ and enters vertex $\mathbf{v}_j$. Suffice to say, link $\mathbf{v}_i$ precedes link $\mathbf{v}_j$. An example of the directed graph is shown in Figure 4(b). From an outward tree with n vertices, an $n\times n$ accessibility matrix can be defined to show the accessibility among the vertices.

Definition 3. (Accessibility matrix) *The accessibility matrix of a directed kinematic graph $\vec{\mathcal{G}}$ of a modular robot with $n+1$ modules (vertices) is an $(n+1)\times(n+1)$ matrix, $\mathcal{R}(\vec{\mathcal{G}})=[r_{ij}]$ $(i,j=0,\cdots,n)$ such that $r_{ij}=1$, if there is a directed path of length one or more from $\mathbf{v}_i$ to $\mathbf{v}_j$; $r_{ij}=0$, otherwise.*

The accessibility matrix can be derived from the AIM once the traversing order on the link modules is determined. For example, the accessibility matrix of $\vec{\mathcal{G}}$ in Figure 4(b) is

$$\mathcal{R}(\vec{\mathcal{G}}) = \begin{array}{c} \\ \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \\ \mathbf{v}_7 \end{array} \begin{array}{c} \begin{array}{cccccccc} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 & \mathbf{v}_6 & \mathbf{v}_7 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}. \tag{2}$$

From $\mathcal{R}(\vec{\mathcal{G}})$, we can obtain the shortest route from the base to the pendant link. This route is called a *path*. The pendant links are the rows of $\mathcal{R}(\vec{\mathcal{G}})$ with all 0s. The number of paths in a branching robot is equal to the number of pendant links. Let link $\mathbf{v}_i$ be a pendant link. All link modules on the path from the base to $\mathbf{v}_i$ are shown in the nonzero entries of column i of $(\mathcal{R}(\vec{\mathcal{G}}) + I_{(n+1)\times(n+1)})^T$. Collecting all the paths, we obtain the path matrix:

Definition 4. (Path matrix)
The path matrix $\mathcal{P}(\tilde{\mathcal{G}})$ of a directed kinematic graph $\tilde{\mathcal{G}}$ of a branch-type robot with $n+1$ link modules (vertices) and m paths is an $m\times(n+1)$ matrix, $\mathcal{P}(\vec{\mathcal{G}}) = [p_{ij}]$, $(i = 1,2,\ldots,m;\ j = 0,1,\ldots,n)$ such that $p_{ij} = 1$, if path i contains vertex j, and $p_{ij} = 0$ otherwise.

For instance, the robot of Figure 4(a) contains three branches (paths). The three paths can be represented as a 3×8 path matrix:

$$\mathcal{P}(\vec{\mathcal{G}}) = \begin{array}{c} \begin{array}{cccccccc} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 & \mathbf{v}_6 & \mathbf{v}_7 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}. \tag{3}$$

Row 1 represents the branch of the robot containing link modules $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$; Row 2 represents the branch of $\mathbf{v}_0$ and $\mathbf{v}_4$; Row 3 represents the branch of $\mathbf{v}_0$, $\mathbf{v}_5$, $\mathbf{v}_6$, and $\mathbf{v}_7$. It can be seen that the rows of $\mathcal{P}(\vec{\mathcal{G}})$ are identical to Columns 3, 4, and 7 of $(\mathcal{R}(\vec{\mathcal{G}}) + I_{(n+1)\times(n+1)})$ respectively.

4. Geometry-Independent Models

4.1 Forward Kinematics

The forward kinematics of a general branch-type modular robot starts with a given AIM and a dyad kinematic model that relates the motion of two connected modules under a joint displacement. A dyad is a pair of connected links in a kinematic chain. Using dyad kinematics recursively with a prescribed graph-traversing order assigned to the robot modules, we may obtain the forward transformation of every branch with respect to the base frame, having a prescribed set of joint displacements. Note that a branch-type robot is one without any closed loop geometry. The kinematics of a closed loop type robot mechanism requires additional constraints, and is not considered here.

4.1.1 Dyad kinematics

Let $\mathbf{v}_i$ and $\mathbf{v}_j$ be two adjacent links connected by a joint $\mathbf{e}_j$, as shown in Figure 5. Denote joint $\mathbf{e}_j$ and link $\mathbf{v}_j$ as link assembly j and the module-coordinate frame on link $\mathbf{v}_i$ as frame i. The relative position (including the orientation) of the dyad, $\mathbf{v}_i$ and $\mathbf{v}_j$, with respect to frame i with a joint angle q_j, can be described by a 4×4 homogeneous matrix,

$$\mathbf{T}_{ij}(q_j) = \mathbf{T}_{ij}(0)e^{\hat{s}_j q_j}, \tag{4}$$

where $\hat{s}_j \in se(3)$ is the twist of joint $\mathbf{e}_j$ expressed in frame j, $\mathbf{T}_{ij}(q_{ij})$ and $\mathbf{T}_{ij}(0) \in SE(3)$. $\mathbf{T}_{ij}(0)$ is the initial pose of frame j relative to frame i. Note that in the following context, the *pose* of a coordinate frame is referred to the 4×4 homogeneous matrix of the orientation and position of a coordinate frame:

$$\mathbf{T}_{ij}(0) = \begin{bmatrix} \mathbf{R}_{ij}(0) & \mathbf{d}_{ij}(0) \\ \mathbf{0} & 1 \end{bmatrix}, \tag{5}$$

where $\mathbf{R}_{ij}(0) \in SO(3)$ and $\mathbf{d}_{ij}(0) \in R^3$ are the initial orientation and position of link frame j relative to frame i respectively. The twist $\hat{s}_j$ of link assembly j is the skew-symmetric matrix representation of the 6-vector line coordinate of the joint axis, $s_j = (\mathbf{q}_j, \mathbf{p}_j)$; $\mathbf{p}_j, \mathbf{q}_j \in R^3$. $\mathbf{p}_j = (p_{jx}, p_{jy}, p_{jz})$ is the unit-directional vector of the joint axis relative to frame j, and $\mathbf{q}_j = (q_{jx}, q_{jy}, q_{jz}) = \mathbf{p}_j \times \mathbf{r}_j$, where $\mathbf{r}_j$ is the position vector of a point along the joint axis relative to frame j. For revolute joints, $\mathbf{s}_j = (0, \mathbf{p}_j)$,and for prismatic joints, $\mathbf{s}_j = (\mathbf{q}_j, 0)$.

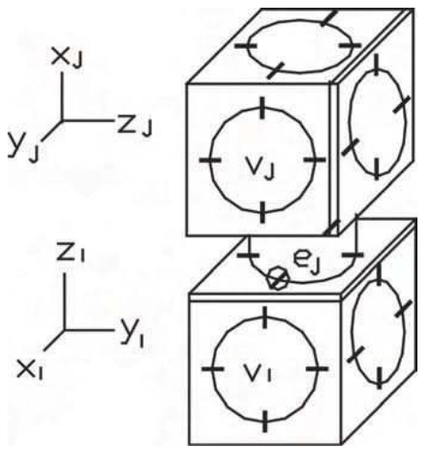


Figure 5. Link-assembly j connected to link i

4.1.2 Recursive forward kinematics

Based on eq. (4), we propose a recursive algorithm for a general branch-type modular robot, termed *TreeRobotKinematics*. This algorithm can derive the forward transformations of the base link to all pendant links based on graph-traversing algorithms. The procedure is illustrated in Figure 6. Implementation details can be found in an earlier work (Chen & Yang 1996). The algorithm takes three inputs: the AIM of the robot $\mathcal{A}(\mathcal{G})$, the base link location T_0, and a set of joint angles $\{q\}$. The forward-kinematics calculation follows the breath-first-search (BFS) traversing algorithm to travel on the connected robot modules.

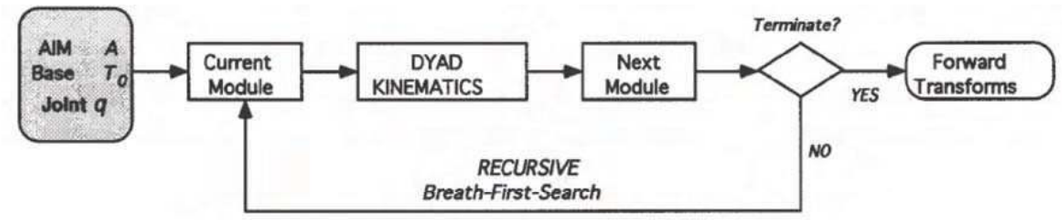


Figure 6. The TreeRobotKinematics algorithm

4.1.3 Path-by-path forward kinematics

A tree-type robot consists of several paths that give the shortest routes from the base to the respective pendant links. Each path can be considered as a serially connected *submanipulator* so that the forward transformation can be derived as conventional industrial manipulator. The sequence of the connected modules in a path is indicated in a row of the path matrix $\mathcal{P}(\vec{\mathcal{G}})$. Let $a=\{a_0,a_1,a_2,\cdots,a_n\}$ represent the links of path k. The base is $a_0\equiv 0$ and the number of links in the path k is defined to be $|a|=n+1$. For instance, path 1 of the robot in Figure 4(a) is $a=\{0,1,2,3\}$. The forward kinematics from the base to the pendant link a_n of path k is given by

$$\begin{aligned} T_{a_0a_n} &= T_{a_0a_1}(q_{a_1})T_{a_1a_2}(q_{a_2})\ldots T_{a_{n-1}a_n}(q_{a_n}) \\ &= \prod_{i=1}^{n}(T_{a_{i-1}a_i}(0)e^{\hat{s}_{a_i}q_{a_i}}) \end{aligned} . \tag{6}$$

For a branch-type modular robot with several paths, the forward kinematics is

$$\mathbf{T}(q_1,q_2,\ldots,q_n)=\begin{bmatrix} T_{a_0a_n} \\ T_{b_0b_m} \\ \vdots \end{bmatrix}=\begin{bmatrix} \prod_{i=1}^{n}(T_{a_{i-1}a_i}(0)e^{\hat{s}_{a_i}q_{a_i}}) \\ \prod_{i=1}^{m}(T_{b_{i-1}b_i}(0)e^{\hat{s}_{b_i}q_{b_i}}) \\ \vdots \end{bmatrix}, \tag{7}$$

where $\mathbf{T}(q_1,q_2,\ldots,q_n)$ represents the vector of 4×4 homogeneous matrices of the poses of all the pendant end-effectors. Since many paths in the branch-type robot share many common modules, there will be repetitive calculations using the model of eq. (7). In actual implementation, we prefer the recursive approach, which introduces no repetitive calculations.

4.2 Inverse Kinematics

The purpose of an inverse kinematics algorithm is to determine the joint angles that cause the end-effector of a manipulator to reach a desired pose. Current robot inverse kinematics algorithms can be categorized into two types: closed-form and numerical. Closed-form-type inverse kinematics requires a complete parameterization of the solution space, usually in terms of simultaneous polynomial equations. Solutions to such a set of simultaneous polynomial solutions exist for a few types of robots with revolute joints or simple geometry. It is very difficult to obtain the inverse kinematics for an arbitrary modular reconfigurable robot in this manner. Here we adopt the numerical approach to solve the inverse kinematics of modular robots. The inverse-kinematics algorithm will construct the differential kinematic model using the local POE formula. The differential kinematic equation of a single branch of a branch-type robot is considered first. Based on the AIM of the robot, one can extend this differential kinematics model to include multiple branch structures. Then the Newton-Raphson iteration method is used to obtain the numerical inverse kinematics solutions. The differential kinematic model can be easily modified to solve the pure position, pure orientation, and hybrid inverse kinematics problems (Chen & Yang 1999).

4.2.1 A Single branch

Let $T_{a_0a_n}$ be the forward transformation of path k as indicated in eq. (6). The differential change in the position of the end-link a_n can be given by

$$\begin{aligned} dT_{a_0a_n} &= \sum_{i=1}^{|a|-1} \frac{\partial T_{a_0a_n}}{\partial q_{a_i}} dq_{a_i} \\ &= \sum_{i=1}^{|a|-1} \left[T_{a_0a_{i-1}} \frac{\partial (T_{a_{i-1}a_i}(0) e^{\hat{s}_{a_i} q_{a_i}})}{\partial q_{a_i}} T_{a_ia_n} \right] dq_{a_i} \\ &= \sum_{i=1}^{|a|-1} \left[T_{a_0a_i} \hat{s}_{a_i} T_{a_ia_n} \right] dq_{a_i} \end{aligned} \tag{8}$$

Left-multiplying $T_{a_0a_n}^{-1}$, eq. (8) becomes,

$$T_{a_0a_n}^{-1} dT_{a_0a_n} = \sum_{i=1}^{|a|-1} T_{a_ia_n}^{-1} \hat{s}_{a_i} T_{a_ia_n} dq_{a_i} . \tag{9}$$

Equation (9) is the differential kinematic equation of a path. Let $T_{a_0a_n}^{d}$ denote the desired position of the end-effector. When it is in the neighborhood of a nominal position of the end-effector $T_{a_0a_n}$, we have

$$dT_{a_0a_n} = T^d_{a_0a_n} - T_{a_0a_n} . \tag{10}$$

Left-multiplying $T^{-1}_{a_0a_n}$ to eq. (9), and using the matrix logarithm,

$$\log(T^{-1}_{a_0a_n} T^d_{a_0a_n}) = (T^{-1}_{a_0a_n} T^d_{a_0a_n} - I) - \frac{(T^{-1}_{a_0a_n} T^d_{a_0a_n} - I)^2}{2} + \frac{(T^{-1}_{a_0a_n} T^d_{a_0a_n} - I)^3}{3} - \cdots \tag{11}$$

We can obtain the following equation by first order approximation:

$$T^{-1}_{a_0a_n} dT_{a_0a_n} = \log(T^{-1}_{a_0a_n} T^d_{a_0a_n}) . \tag{12}$$

Substituting eq. (12) into eq. (9), we obtain

$$\log(T^{-1}_{a_0a_n} T^d_{a_0a_n}) = \sum_{i=1}^{|a|-1} T^{-1}_{a_0a_n} \hat{s}_{a_i} T_{a_0a_n} dq_{a_i} . \tag{13}$$

Explicit formulae for calculating the logarithm of elements of $SO(3)$ and $SE(3)$ were derived by Park and Bobrow (1994). Definitely, $\log(T^{-1}_{a_0a_n} T^d_{a_0a_n})$ is an element of $se(3)$ so that it can be identified by a 6×1 vector denoted by $\log(T^{-1}_{a_0a_n} T^d_{a_0a_n})^\vee$ in which the first and later three elements represent the positional and orientational *differences* between $T_{a_0a_n}$ and $T^d_{a_0a_n}$. Converting eq. (13) into the adjoint representation, we get

$$\log(T^{-1}_{a_0a_n} T^d_{a_0a_n})^\vee = Ad_{T^{-1}_{a_0a_n}} \sum_{i=1}^{|a|-1} Ad_{T_{a_0a_i}} s_{a_i} dq_{a_i} . \tag{14}$$

Conveniently, eq. (14) can also be expressed as the following form:

$$D_{T_k} = J_k dq_k \quad , \tag{15}$$

where

$D_{T_k} = \log(T^{-1}_{a_0a_n} T^d_{a_0a_n})^\vee \in R^{6\times1}$ is referred as the *pose difference vector* for path k;
$J_k = A_k B_k S_k \in R^{6\times(|a|-1)}$, is termed as *body manipulator Jacobian matrix* (Murray et al. 1994);
$A_k = Ad_{T^{-1}_{a_0a_n}} \in R^{6\times6}$;

$B_k = \text{row}[Ad_{T_{a_0a_1}}, Ad_{T_{a_0a_2}}, \ldots, Ad_{T_{a_0a_n}}] \in R^{6\times 6(|a|-1)}$;

$S_k = \text{diag}[s_{a_1}, s_{a_2}, \ldots, s_{a_n}] \in R^{6(|a|-1)\times(|a|-1)}]$; and

$dq_k = \text{column}[dq_{a_1}, dq_{a_2}, \ldots, dq_{a_n}] \in R^{(|a|-1)\times 1}$.

Equation (15) defines the differential kinematics for path k. It can be utilized in the Newton-Raphson iteration to obtain an inverse kinematics solution for a given pose.

4.2.2 Entire manipulator

The paths of a branch-type manipulator may not be independently driven, because of the common sharing modules. This forbids us to treat each path as independent serial-type manipulators. Hence, with a given set of the pendant end-effectors's poses for all branches, the inverse kinematics must be solved simultaneously. With the assistance of the path matrix, we are able to identify the connected and related modules in a path. Then, we can orderly combine the differential kinematic equations (eq. (15)) ofall constituting paths into a single matrix equation of the following form:

$$D_T = Jdq \quad , \tag{16}$$

where

$D_T = \text{column}[D_{T_1}, D_{T_2}, \ldots, D_{T_m}] \in R^{6m\times 1}$,is termedthe *generalized pose difference vector*;

$J = ABS \in R^{6m\times n}$, is termed the *generalized body manipulator Jacobian matrix*;

$A = \text{diag}[A_1, A_2, \ldots, A_m] \in R^{6m\times 6m}$; and

$$B = \begin{bmatrix} p_{11}Ad_{T_{01}} & p_{12}Ad_{T_{02}} & \cdots & p_{1n}Ad_{T_{0n}} \\ p_{21}Ad_{T_{01}} & p_{22}Ad_{T_{02}} & \cdots & p_{2n}Ad_{T_{0n}} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1}Ad_{T_{01}} & p_{m2}Ad_{T_{02}} & \cdots & p_{mn}Ad_{T_{0n}} \end{bmatrix} \in R^{6m\times 6n}$$

The coefficient, $p_{ij}(i = 1,2,\ldots,m; j = 0,1,2,\ldots,n)$ is entry (i, j) of the *path* matrix $\mathcal{P}$, and m is the total number of paths; $S = \text{diag}[s_1, s_2, \ldots, s_n] \in R^{6n\times n}$; $dq = \text{column}[dq_1, dq_2, \ldots, dq_n] \in R^{n\times 1}$.

Rewriting this equation in an iterative form, we get

$$dq^{i+1} = J^* D_T \tag{17}$$

$$q^{i+1} = q^i + dq^{i+1}, \tag{18}$$

where i represents the number of iterations and J^* is the Moore-Penrose pseudoinverse of J. Using the Newton-Raphson method, a close-loop iterative algorithm similar to that of Khosla, Newman and Prinz (1985) is employed (Fig. 7). The iterative algorithm determines the necessary changes in the joint angles to achieve a differential change in the position and orientation of the end-effector. Given a complete robot assembly (or the AIM) and a set of desired poses T^d, this algorithm starts from an initial guess, q^0, somewhere in the neighborhood of the desired solution. It is terminated when a prescribed termination criteria is reached. As one can see, the structure of J depends on the path matrix, which is implied in the kinematic graph of the robot. Therefore, once the assembly configuration of a modular robot is determined and all module parameters are obtained, the differential kinematic model (eq. (16)) can be generated automatically.

Computational examples of the inverse kinematics algorithms for branch-type and serial modular robots are given by Chen & Yang (1999) to illustrate the algorithm's applicability and effectiveness. When compared to the other numerical inverse kinematics algorithm using D-H parameters, our method always use less number of iterations and computing time for the same given pose. This is due to the use of the *pose difference vector* computed from the matrix logarithm in eq. (16), and not the difference of homogeneous transformation matrices. Actual implementation of the algorithm using C++ codes shows that the computation time for each solution can take less than 20 msec on a Pentium II 300MHz PC, which satisfies the basic requirement for real-time control and simulation.

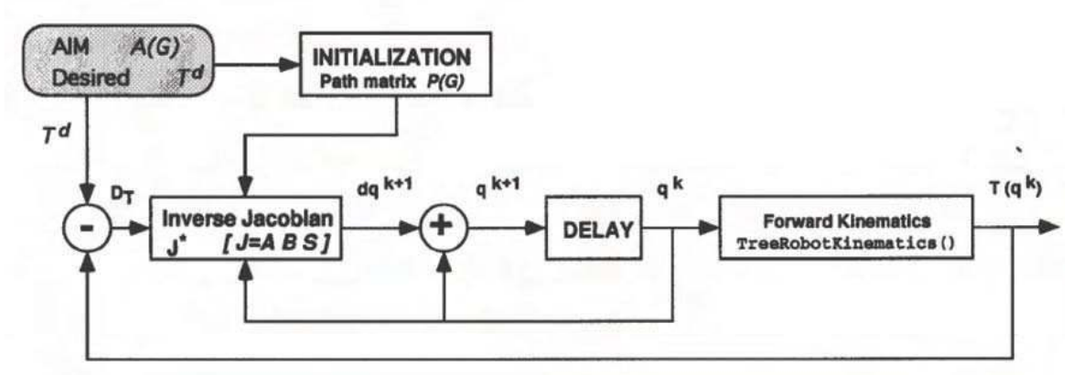


Figure 7. Inverse kinematics algorithm

4.3 Kinematic Calibration

The machining tolerance, compliance, and wear of the connected mechanism and mis-alignment of the connected module components may introduce errors in positioning the end-effector of a modular robot. Hence, calibrating the kinematic parameters of a modular robot to enhance is positioning accuracy is important, especially in high precision application such as hard-disk assembly. Current kinematic calibration algorithms for industrial robots that are designed for certain types of serial manipulators are not suitable for modular robots with arbitrary geometry. Here we propose a general singularity-free calibration-modeling method for modular reconfigurable robots, based on the forward kinematics discussed in previous section. This method follows local POE formulae. The robot errors are assumed to be in the initial positions of the consecutive modules. Based on linear superposition and differential transformation, a six-parameter model is derived. This model can be generated automatically once the AIM of the robot is given. An iterative least-square algorithm is then employed to find the error parameters to be corrected.
The calibration starts with a serial-type manipulator kinematics model:

$$T_{0n}(\mathbf{q}) = T_{01}(q_1)\,T_{12}(q_2)\cdots T_{n-1,n}(q_n) \tag{19}$$

$$= T_{01}(0)e^{\hat{s}_1 q_1}\,T_{12}(0)e^{\hat{s}_2 q_2}\cdots T_{n-1,n}(0)e^{\hat{s}_n q_n} \tag{20}$$

Extension to a general branch-type modular robot is similar to the treatment of the inverse-kinematics model in previous section. Basically, eq. (20) can be treated as a function of the joint angles, $\mathbf{q} = (q_1, \cdots, q_n)$, locations of the joint axes, $\hat{\mathbf{s}} = (\hat{s}_1, \cdots, \hat{s}_n)$, and the relative initial positions of the dyads, $T_0 = (T_{01}(0), \cdots, T_{n-1,n}(0))$:

$$T_{0n} = f(T_0, \hat{\mathbf{s}}, \mathbf{q}). \tag{21}$$

Written in differential form, we have

$$dT_{0n} = \frac{\partial f}{\partial T_0}dT_0 + \frac{\partial f}{\partial \hat{\mathbf{s}}}d\hat{\mathbf{s}} + \frac{\partial f}{\partial \mathbf{q}}d\mathbf{q}\,. \tag{22}$$

The differential dT_{0n} can be interpreted as the difference between the nominal position and the measured position.

4.3.1 Error model of a dyad

Our kinematic calibration is based on the local frame representation of a dyad

described in eq. (4). Two assumptions are made in the dyad of link $\mathbf{v}_{i-1}$ and $\mathbf{v}_i$ of a modular robot chain: first, small geometric errors only exist in the initial position $T_{i-1,i}(0)$; second, the twist and joint angle q_i assume the nominal values through out the calibration analysis. Hence, instead of identifying the module's actual initial positions, joint twists and angle offsets, we look for a new set of local initial positions (local frames, called calibrated initial positions), in the calibration model, so that the twist of the joint remains the nominal value. In other words, the errors in a dyad are lumped with the initial position. Therefore, $d\hat{\mathbf{s}}$ and $d\mathbf{q}$ can be set to 0. Because $SE(3)$ has the dimension of six---three for positions and three for orientations---there can be only six independent quantities in $T_{i-1,i}(0)$, and there will be six independent error parameters in a dyad. Denote the small error in the initial position of dyad $(\mathbf{v}_{i-1}, \mathbf{v}_i)$ as $dT_{i-1,i}(0)$, then

$$\begin{aligned} dT_{i-1,i}(0) &= T_{i-1,i}(0)\,\hat{\Delta}_i \\ \hat{\Delta}_i &= \begin{bmatrix} 0 & -\delta z_i & \delta y_i & dx_i \\ \delta z_i & 0 & -\delta x_i & dy_i \\ -\delta y_i & \delta x_i & 0 & dz_i \\ 0 & 0 & 0 & 0 \end{bmatrix}, \end{aligned} \tag{23}$$

where dx_i, dy_i, and dz_i are infinitesimal displacements along x-, y-, and z-axes of link frame i respectively, and δx_i, δy_i and δz_i are infinitesimal rotations about x-, y-, and z-axes of link frame i respectively.

4.3.2 Gross error model of a robot

Similar to the error model of a dyad, the gross-geometric error, dT_{0n} between the actual end-effector position the nominal position can be described as:

$$dT_{0n} = \hat{\Delta}_{0n} T_{0n} \tag{24}$$

and

$$\hat{\Delta}_{0n} = dT_{0n} T_{0n}^{-1} \tag{25}$$

$$= \begin{bmatrix} 0 & -\delta z_{0n} & \delta y_{0n} & dx_{0n} \\ \delta z_{0n} & 0 & -\delta x_{0n} & dy_{0n} \\ -\delta y_{0n} & \delta x_{0n} & 0 & dz_{0n} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{26}$$

where δx_{0n}, δy_{0n}, δz_{0n} are the rotations about the axes of the base frame, and

dx_{0n}, dy_{0n}, and dz_{0n} are the displacements along the axes of base frame respectively. Note that the gross error, dT_{0n}, is expressed in the base frame. Equation (25) follows the left multiplicative differential transformation of T_{0n}. The calibrated position of the end-effector becomes

$$T_{0n}'(q) = T_{0n} + dT_{0n}. \tag{27}$$

4.3.3 Linear superposition

Based on the assumptions, the errors in the dyads will contribut to the gross error in the end-effector's position dT_{0n}. Since the geometric errors are all very small, the principle of linear superposition can be applied. We assume that the gross errors dT_{0n} are the linear combination of the errors in the dyads $dT_{i-1,i}(0)$, $(i = 1,2,...,n)$; then

$$dT_{0n} = \sum_{i=1}^{n} T_{0,i-1}\, dT_{i-1,i}(0)\, e^{\hat{s}_i q_i}\, T_{i,n}. \tag{28}$$

Equation (28) converts and sums the initial position errors of the dyads in the base-frame coordinates. The forward kinematics of link-frame j relative to link-frame i $(i \le j)$ is represented by T_{ij}. Especially, $T_{ij} = I_{4\times4}$ when $i = j$. Substituting eq. (23) into eq. (28), and right-multiplying T_{0n}^{-1},

$$dT_{0n} T_{0n}^{-1} = \hat{\Delta}_{0n} \tag{29}$$

$$= \sum_{i=1}^{n} T_{0,i-1} T_{i-1,i}(0) \hat{\Delta}_i T_{i-1,i}^{-1} T_{0,i-1}^{-1} \tag{30}$$

From the first order approximation, we have

$$\hat{\Delta}_{0n} = dT_{0n} T_{0n}^{-1} \approx \log(T_{0n}' T_{0n}^{-1}). \tag{31}$$

Converting eq. (31) into the adjoint representation, we have

$$\log^{\vee}(T_{0n}' T_{0n}^{-1}) = \sum_{i=1}^{n} Ad_{T_{0,i-1}}(Ad_{T_{i-1,i}(0)}(\Delta_i)). \tag{32}$$

Equation (32) can also be expressed in the following matrix form

$$y = \mathbf{A}x, \tag{33}$$

where

$y = \log^{\vee}(T_{0n}' T_{0n}^{-1}) \in R^{6\times 1}$;

$x = \text{column}[\Delta_1, \Delta_2, \cdots, \Delta_n] \in R^{6n\times 1}$; and

$\mathbf{A} = \text{row}[Ad_{T_{0,1}(0)}, Ad_{T_{0,1}}(Ad_{T_{1,2}(0)}), \cdots, Ad_{T_{0,n-1}}(Ad_{T_{n-1,n}(0)})] \in R^{6\times 6n}$.

In Equation (33), x represents the error parameters to be identified in a modular robot assembly. The quantities in matrix $\mathbf{A}$ and T_{0n}^{-1} are determined from the nominal model. T_{0n}' comes from the actual measured data. To improve the accuracy of the calibration model, the kinematic calibration procedure usually requires the position of the end-effector to be measured in several different robot postures. For the i^{th} measurement, we obtain y_i and $\mathbf{A}_i$. After taking m measurements,

$$\tilde{\mathbf{Y}} = \tilde{\mathbf{A}} x \,, \tag{34}$$

Where

$\tilde{\mathbf{Y}} = \text{column}[y_1, y_2, \cdots, y_m] \in R^{6m\times 1}$; and

$\tilde{\mathbf{A}} = \text{column}[\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_m] \in R^{6m\times 6n}$.

The least-squares solution for x can be obtained by

$$x = \tilde{\mathbf{A}}^{\dagger} \tilde{\mathbf{Y}} \,, \tag{35}$$

where $\tilde{\mathbf{A}}^{\dagger}$ is the pseudo-inverse of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{A}}^{\dagger} = (\tilde{\mathbf{A}}^T \tilde{\mathbf{A}})^{-1} \tilde{\mathbf{A}}^T$ for $m > n$; $\tilde{\mathbf{A}}^{\dagger} = \tilde{\mathbf{A}}^T (\tilde{\mathbf{A}} \tilde{\mathbf{A}}^T)^{-1}$ for $m < n$; $\tilde{\mathbf{A}}^{\dagger} = \tilde{\mathbf{A}}^{-1}$ for $m = n$.

The calibration procedure is illustrated in the diagram of Figure 8(a). Computer simulation and actual experiment on the modular robot systems described by Chen & Yang (1997) and Chen et al. (2001) have shown that this calibration method can improve the accuracy in end-effector positioning by up to two orders of magnitudes.

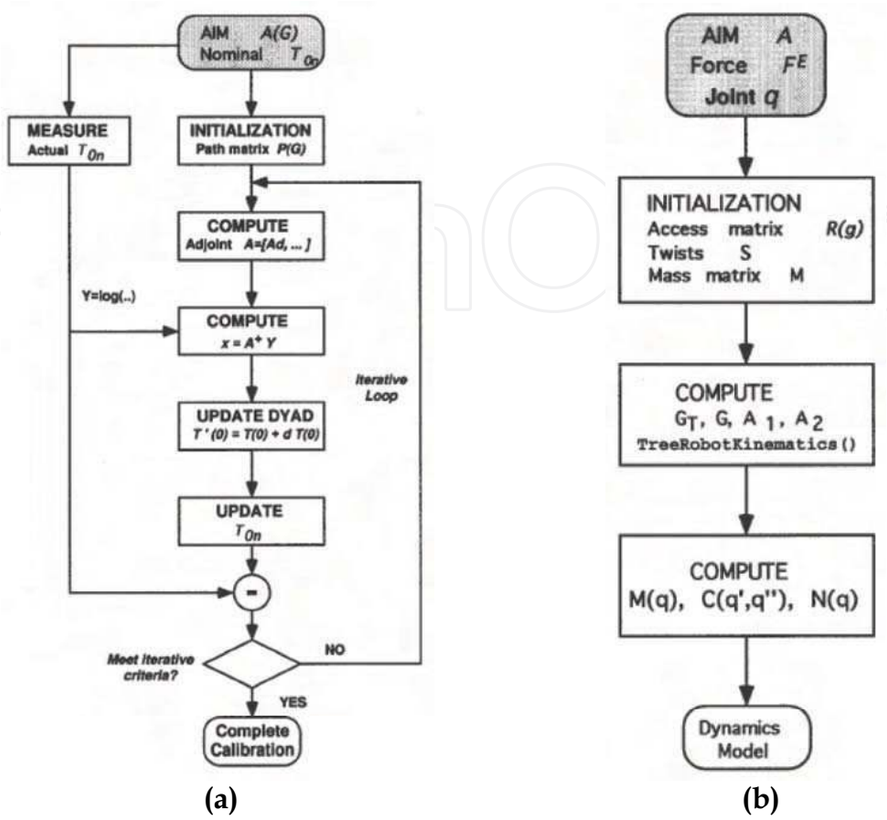


Figure 8. (a) Calibration algorithm for modular robots; (b) Dynamic model generation

4.4 Dynamics

The dynamic model of a robot can be formulated with an iterative method through a recursive Newton-Euler equation. This method can be generally applied to branch-type robots without modification. Here we present a method to generate the closed-form dynamic models of modular robots using the AIM and the recursive algorithm.

4.4.1 Newton-Euler Equation for link assembly

Assume that the mass center of link assembly j is coincident with the origin of the link module frame j. The Newton-Euler equation of this rigid link assembly with respect to frame j is (Murray et al 1994)

$$\mathbf{F}_j = \begin{bmatrix} f_j \\ \tau_j \end{bmatrix} = \begin{bmatrix} m_j I & 0 \\ 0 & J_j \end{bmatrix} \begin{bmatrix} \dot{v}_j \\ \dot{w}_j \end{bmatrix} + \begin{bmatrix} w_j \times m_j v_j \\ w_j \times J_j w_j \end{bmatrix}, \tag{36}$$

where $\mathbf{F}_j \in R^{6\times1}$ is the resultant wrench applied to the center of mass relative to frame j. The total mass of link assembly j is m_j (which is equal to the sum of link $\mathbf{v}_j$ and joint $\mathbf{e}_j$). The inertia tensor of the link assembly about frame j is J_j. Transforming eq. (36) into the adjoint representation,we have

$$\mathbf{F}_j = M_j\dot{V}_j - ad_{V_j}^T(M_jV_j). \tag{37}$$

The following notations are adopted:

- $M_j = \begin{bmatrix} m_j & 0 \\ 0 & J_j \end{bmatrix} \in R^{6\times6}$ is the generalized mass matrix;
- $V_j = \begin{bmatrix} v_j \\ w_j \end{bmatrix} \in R^{6\times1}$ is the generalized body velocity, where v_j and w_j are 3×1 vectors defining body translational velocity, $v_j = (v_x, v_y, v_z)^T$, and the angular velocity, $w_j = (w_x, w_y, w_z)^T$, respectively;
- $ad_{V_j}^T \in R^{6\times6}$ is the transpose of adjoint matrix ad_{V_j} related to V_j

$$ad_{V_j}^T = (ad_{V_j})^T = \begin{bmatrix} \hat{w}_j & \hat{v}_j \\ 0 & \hat{w}_j \end{bmatrix}^T = \begin{bmatrix} -\hat{w}_j & 0 \\ -\hat{v}_j & -\hat{w}_j \end{bmatrix}; \tag{38}$$

- $\hat{v}_j$ and $\hat{w}_j \in R^{3\times3}$ are skew-symmetric matrices related to v_j and w_j respectively; and $\dot{V}_j = \begin{bmatrix} \dot{v}_j \\ \dot{w}_j \end{bmatrix} \in R^{6\times1}$ is the generalized body acceleration.

4.4.2 Recursive Newton-Euler algorithm

The recursive algorithm is a two-step iteration process. For a branch-type robot, the generalized velocity and acceleration of each link are propagated from the base to the tips of all branches. The generalized force of each link is propagated backward from the tips of the branches to the base. At the branching module, generalized forces transmitted back from all branches are summed.

FORWARD ITERATION

The generalized velocity and acceleration of the base link are given initially,

$$V_b = V_0 = (0,0,0,0,0,0)^T \tag{39}$$

$$\dot{V}_b = \dot{V}_0 = (0,0,g,0,0,0)^T \tag{40}$$

where V_b and $\dot{V}_b$ are expressed in the base frame 0. We assume that the base frame coincides with the spatial reference frame. The generalized acceleration (eq. (40)) is initialized with the gravitation acceleration g to compensate for the effect of gravity. Referring to Figure 6, the recursive body velocity and acceleration equations can be written as

$$V_j = Ad_{T_{ij}^{-1}}(V_i) + s_j \dot{q}_j \tag{41}$$

$$\dot{V}_j = Ad_{T_{ij}^{-1}}(\dot{V}_i) + ad_{Ad_{T_{ij}^{-1}}(V_i)}(s_j \dot{q}_j) + s_j \ddot{q}_j \tag{42}$$

where all the quantities, if not specified, are expressed in link frame j.

- V_j and $\dot{V}_j$ are the generalized velocity and acceleration of link-assembly j;
- $\dot{q}_j$ and $\ddot{q}_j$ are the velocity and acceleration of joint $\mathbf{e}_j$ respectively;
- $Ad_{T_{ij}^{-1}}$ is the adjoint representation of $T_{ij}^{-1}(q_j)$, where $T_{ij}(q_j) \in SE(3)$ is the position of frame j relative to frame i with joint angle q_j and $Ad_{T_{ij}^{-1}} = (Ad_{T_{ij}})^{-1}$; and
- $s_j \in R^{6\times 1}$ is the twist coordinates of joint $\mathbf{e}_j$.

BACKWARD ITERATION

The backward iteration of the branch-type robot starts simultaneously from all the pendant link assembly. Let $\mathcal{V}_{PD} \subset \mathcal{V}$ be set of the pendant links of the branch-type robot. For every pendant link assembly d_i ($\mathbf{v}_{d_i} \in \mathcal{V}_{PD}$), the Newton-Euler equation (eq. (37)) can be written as

$$F_{d_i} = -F_{d_i}^e + M_{d_i}\dot{V}_{d_i} - ad_{V_{d_i}}^T(M_{d_i}V_{d_i}), \tag{43}$$

where F_{d_i} is the wrench exerted on link-assembly $\mathbf{v}_{d_i}$ by its parent (preceding) link relative to frame d_i; and $F_{d_i}^e$ is the external wrench exerted on v_{d_i}. Note that the total wrench is $\mathbf{F}_{d_i} = F_{d_i} + F_{d_i}^e$. Now traverse the links in the robot backward from the pendant links. Let $\mathcal{V}_{Hi}$ be the set of successors of link $\mathbf{v}_i$. For every link assembly i, the Newton-Euler equation (eq. (37)) can be written in the following form:

$$F_i = \sum_{j \in V_{Hi}} Ad_{T_{ij}^{-1}}^T(F_j) - F_i^e + M_i\dot{V}_i - ad_{V_i}^T(M_iV_i), \tag{44}$$

where all quantities, if not specified, are expressed in link-frame i; $F_i \in R^{6\times 1}$ is the wrench exerted *to* link-assembly i by its predecessor; $F_j \in R^{6\times 1}$ is the wrench exerted *by* link-assembly i to the successor $\mathbf{v}_j \in \mathcal{V}_{Hi}$ expressed in link-frame j; F_i^e is the external wrench applied to link-assembly i. The total wrench is $\mathbf{F}_i = F_i - \sum_{j\in V_{Hi}} Ad^T_{T_{ij}^{-1}}(F_j) + F_i^e$.

The applied torque/force to link assembly i by the actuator at its input joint $\mathbf{e}_i$, can be calculated by

$$\tau_i = s_i^T F_i . \tag{45}$$

4.4.3 Closed-Form Equations of Motion

By iteratively expanding the recursive Newton-Euler equations (eqs. (39)-(44)) in the body coordinates, we obtain the generalized velocity, generalized acceleration, and generalized force equations in matrix form:

$$\mathbf{V} = \mathbf{GS}\dot{\mathbf{q}} \tag{46}$$

$$\dot{\mathbf{V}} = \mathbf{G}_{\mathbf{T}_0}\dot{\mathbf{V}}_{\mathbf{0}} + \mathbf{GS}\ddot{\mathbf{q}} + \mathbf{GA_1V} \tag{47}$$

$$\mathbf{F} = \mathbf{G}^T\mathbf{F}^{\mathbf{E}} + \mathbf{G}^T\mathbf{M}\dot{\mathbf{V}} + \mathbf{G}^T\mathbf{A_2MV} \tag{48}$$

$$\mathbf{t} = \mathbf{S}^T\mathbf{F} \tag{49}$$

where

- $\mathbf{V} = \text{column}[V_1, V_2, \cdots, V_n] \in R^{6n\times 1}$ is the generalized body-velocity vector;
- $\dot{\mathbf{V}} = \text{column}[\dot{V}_1, \dot{V}_2, \cdots, \dot{V}_n] \in R^{6n\times 1}$ is the generalized body-acceleration vector;
- $\mathbf{F} = \text{column}[F_1, F_2, \cdots, F_n] \in R^{6n\times 1}$ is the body-wrench vector;
- $\mathbf{t} = \text{column}[\tau_1, \tau_2, \cdots, \tau_n] \in R^{n\times 1}$ is the applied joint-torque/force vector;
- $\dot{\mathbf{q}} = \text{column}[\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_n] \in R^{n\times 1}$ is the joint-velocity vector;
- $\ddot{\mathbf{q}} = \text{column}[\ddot{q}_1, \ddot{q}_2, \cdots, \ddot{q}_n] \in R^{n\times 1}$ is the joint-acceleration vector;
- $\dot{\mathbf{V}}_{\mathbf{0}} = (0, 0, g, 0, 0, 0)^T \in R^{6\times 1}$ is the generalized acceleration of the base link;
- $\mathbf{S} = \text{diag}[s_1, s_2, \cdots, s_n] \in R^{6n\times n}$ is the joint-twist matrix in the respective body coordinates;
- $\mathbf{M} = \text{diag}[M_1, M_2, \cdots, M_n] \in R^{6n\times 6n}$ is the total generalized-mass matrix;
- $\mathbf{A_1} = \text{diag}[-ad_{s_1\dot{q}_1}, -ad_{s_2\dot{q}_2}, \cdots, -ad_{s_n\dot{q}_n}] \in R^{6n\times 6n}$;
- $\mathbf{A_2} = \text{diag}[-ad^T_{V_1}, -ad^T_{V_2}, \cdots, -ad^T_{V_n}] \in R^{6n\times 6n}$;
- $\mathbf{F}^{\mathbf{E}} = \text{column}[F_1^e, F_2^e, \cdots, F_n^e] \in R^{6n\times 1}$ is the external wrench vector;

$$\mathbf{G}_{\mathbf{T_0}} = \begin{bmatrix} Ad_{T_{01}^{-1}} \\ Ad_{T_{02}^{-1}} \\ \vdots \\ Ad_{T_{0n}^{-1}} \end{bmatrix} \in R^{6n\times 6} \text{ ;and}$$

$$\mathbf{G} = \begin{bmatrix} I_{6\times6} & 0 & 0 & \cdots & 0 \\ r_{12}Ad_{T_{12}^{-1}} & I_{6\times6} & 0 & \cdots & 0 \\ r_{13}Ad_{T_{13}^{-1}} & r_{23}Ad_{T_{23}^{-1}} & I_{6\times6} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{1n}Ad_{T_{1n}^{-1}} & r_{2n}Ad_{T_{2n}^{-1}} & r_{3n}Ad_{T_{3n}^{-1}} & \cdots & I_{6\times6} \end{bmatrix} \in R^{6n\times6n}.$$

Note that $\mathcal{R}(\vec{\mathcal{G}}) = [r_{ij}] \in R^{(n+1)\times(n+1)}$ is the accessibility matrix. The matrix $\mathbf{G}$ is called a *transmission matrix*. Substituting eqs. (46)-(48) into eq. (49), we obtain the closed-form equation of motion for a branch-type modular robot with $n+1$ modules (including the base module)

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{N}(\mathbf{q}) = \mathbf{t} \tag{50}$$

where

$$\mathbf{M}(\mathbf{q}) = \mathbf{S}^T\mathbf{G}^T\mathbf{M}\mathbf{G}\mathbf{S} \tag{51}$$

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{S}^T\mathbf{G}^T(\mathbf{M}\mathbf{G}\mathbf{A}_1 + \mathbf{A}_2\mathbf{M})\mathbf{G}\mathbf{S} \tag{52}$$

$$\mathbf{N}(\mathbf{q}) = \mathbf{S}^T\mathbf{G}^T\mathbf{M}\mathbf{G}_{\mathbf{T_0}}\dot{\mathbf{V}}_0 + \mathbf{S}^T\mathbf{G}^T\mathbf{F}^{\mathbf{E}} \tag{53}$$

The mass matrix is $\mathbf{M}(\mathbf{q})$; $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ represents the centrifugal and Coriolis accelerations; $\mathbf{N}(\mathbf{q})$ represents the gravitational force and external forces. The procedure for obtaining the closed-form equation (eq. (50)) is summarized in Figure 8(b). It has been successfully implemented in Mathematica code.

5. Configuration Optimization

Introducing modularity in a robotic system implies that the system performance can be optimized through proper selection and reconfiguration of module components. The task planner for the modular robotic workcell will be able to determine the optimal robot configuration and geometry for a given task from an inventory of robot modules. Figure 9 depicts the general approach for determining the optimal assembly configuration. Shaded blocks represent the basic formulation of the optimization problem. With a given set of modules selected from the component database, all possible and unique assembly con-

figurations can be generated and identified through an enumeration algorithm (Chen & Burdick 1998). In the second step, an objective function is formulated to evaluate the performance of every assembly configuration, based on the task specifications. A basic robot task contains task specifications that are provided by the task planner---the goal positions/orientations, force application, accuracy, and dexterity of the end-effectors---and constraints to be overcome---obstacle avoidance, workspace limit, singularity and kinematic redundancy (Chen & Burdick 1995; Yang & Chen 2001). A search/optimization procedure is employed in the last step to find the optimal assembly configuration.

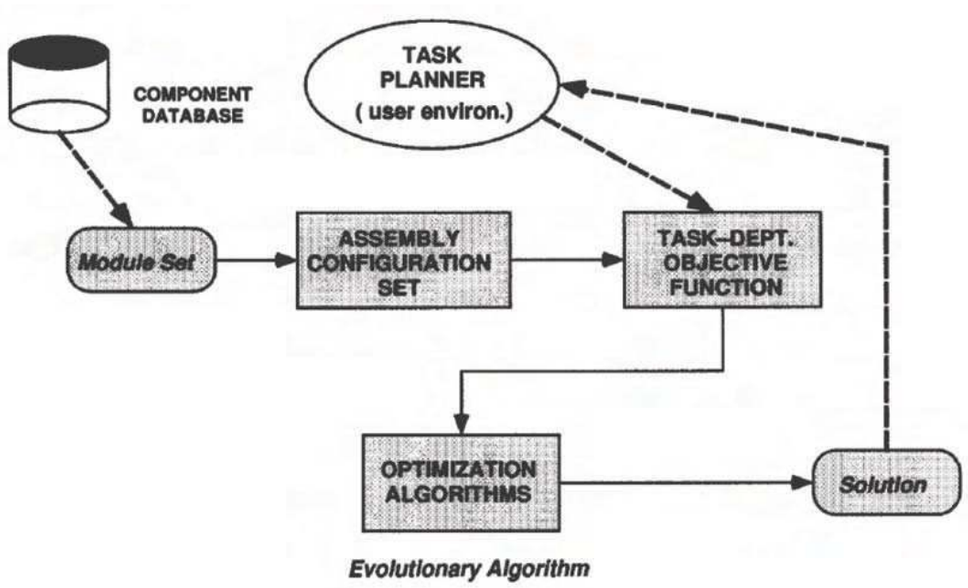


Figure 9. Determination of a task-optimal configuration

Note that all the dimensions of the modules have been previously designed and fixed at the selection stage. With a given set of modules, the possible combination of robot-assembly configurations is always a finite number. Therefore, the parameter space for the optimization is discrete, and combinatorial optimization methods can be applied. Exhaustive search algorithms can be used to find the exact optimal solution, but the exponential growth of the data set impedes the efficient implementation of such an algorithm. Random-search techniques such as genetic algorithms (GA) (Chen 1994) and simulated annealing (SA) (Paredis & Khosla 1995) are more suitable for such problems. Transition rules for data points required in GA and SA can be easily implemented based on a data-representation scheme such as AIM.

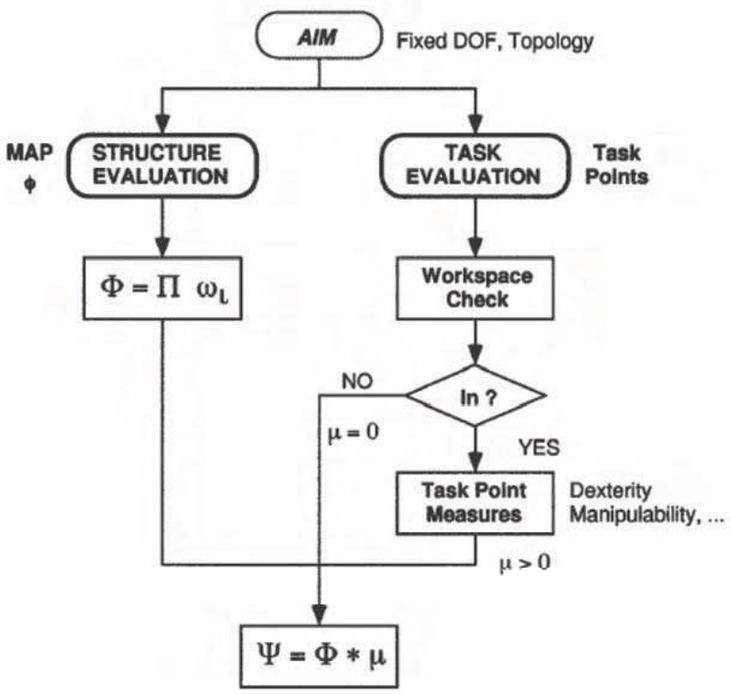


Figure 10. The ACEF for serial modular robots

5.1 Task-Oriented Objective Function

The crucial point in determining the optimal robot configuration is formulating an objective function that will assign a "goodness" value to every assembly configuration accomplishing a specified task. The form of the objective function should be general enough so that it is applicable to a wide variety of task requirements. Two components of a robot task---task specifications and constraints---must be considered in formulating the objective function. We call this function an *assembly configuration evaluation function* (ACEF). The assembly configuration with the greatest ACEF value is deemed optimal. It is also important to note that from a given set of modules it is possible to construct robots with various topologies, such as serial or parallel kinematic structures. Even with a fixed robot-topology class, the number of degrees of freedom (DOF) can alter the kinematic functionality of the system. Here we propose a solution strategy for modular robot with a fixed topology and a fixed number of DOF.

The structure of the ACEF for a serial modular robot is shown in Figure 10. The input is an AIM with a predefined number of DOFs and predefined topology. The output is the ``goodness'' of the AIM in terms of a non-negative real

number. An AIM with a large ACEF value represents a good assembly configuration. The ACEF consists of two parts: task and structure evaluations. Task evaluation is performed according to the given task specifications: the task points (or the positions of the end-effector) and a designated criteria measure, such as the dexterity or the manipulability. A workspace check on the task points is executed before computing the measures for filtering out inaccessible points. Structure evaluation assesses the kinematic constraints (joint singularity and redundancy, link interference) and environmental constraints (workspace obstacles) imposed on the robot in accomplishing the assigned task. The proposed ACEF assumes the modular robot is operated in a structured environment, and that there are no obstacles in the workspace. An auxiliary function, termed the *module-assembly preference* (MAP) is defined on the AIM to exclude undesirable kinematic features. Detailed implementation of the task and structure evaluation can be obtained from Chen (1996).

5.2 Evolutionary Algorithms

An evolutionary algorithm is a probabilistic search/optimization method based on the principle of evolution and hereditary of nature systems (Michalewicz 1994). In this algorithm, a population of individuals for each generation is maintained. The individual is implemented with some data structure and is evaluated by a ``fitness function'' to give a measure of its ``fitness''. A new population is formed by selecting the more suitable individuals. Members in the new population undergo transformations by the ``genetic operators'' to form new solutions. Through structured random information changes, the new generation is more ``fit'' than the previous generation. After a number of iterations, the individuals will converge to an optimal or near-optimal solution. Here we attempt to use the AIMs as the data structure of the solution, and define AIM-related genetic operators (Chen 1996) as solving the task-optimal problem in an evolutionary approach, because AIM is a natural representation of the modular robot and is topologically independent.
Figure 11 depicts the application of the evolutionary algorithm in solving the task-optimal configuration problem. An example of optimizing the configuration of a 4-DOF modular robot is provided in the following example. Suppose we wish to find a 4-DOF fixed-base serial robot with revolute joints that passes through two task points $\mathbf{p}_1$ and $\mathbf{p}_2$. Also suppose that we require there be no redundant joints, and minimum link interference. Let the performance measure of the robot be the manipulability. The initial set of AIMs randomly generated is shown in Figure 12. The population size is 8, and the evolution stopped after 30 generations. The assembly configuration in the target generation that has the highest fitness value is chosen as the optimal one (Fig. 13a). The average and maximum fitness values in every generation are shown in Figure

13(b). As can be seen, the evolutionary algorithm does increase the fitness values generation by generation. Although the best solution may not be guaranteed, a suboptimal solution can always be found, and in return, the efficiency of finding the solution is increased.

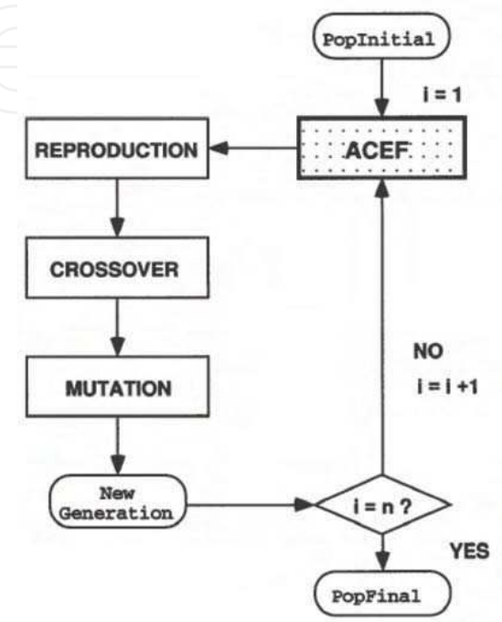
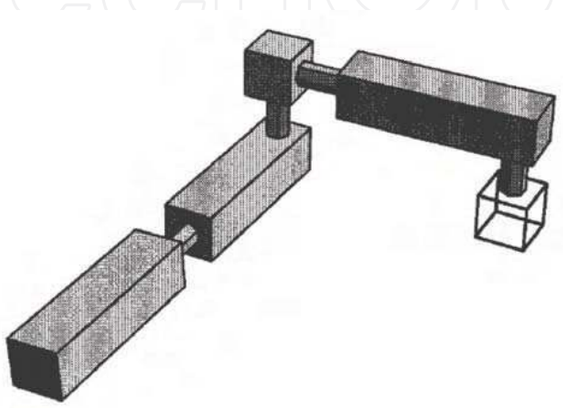
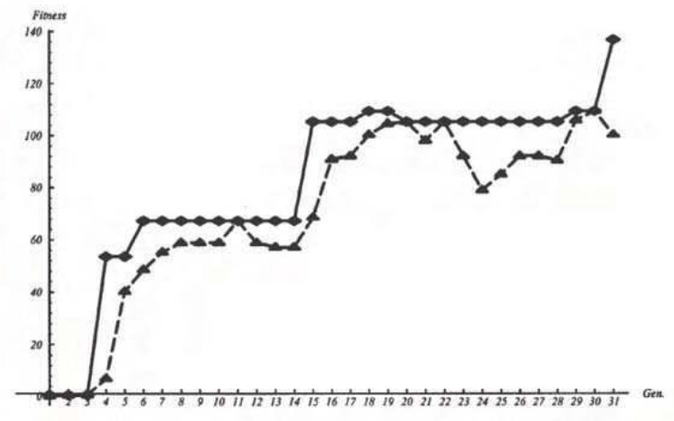


Figure 11. The evolution algorithm

Figure 12. The initial generation



(a)



(b)

Figure 13. (a) Optimal assembly configuration; (b) average and maximum fitness in each generation

6. Simulation Software for Modular Robots

To visualize and simulate the performance of an assembled robot, such as reachability and workspace, a robot simulation software application is necessary. The **S**imulation **E**nvironment for **MO**dular **R**obot **S**ystem (a.k.a. *SEMORS*) is a Windows NT-based object-oriented software application developed for this purpose. Based on the proposed local POE models and AIM data structures, *SEMORS* offers uniform and automatic model construction effort (kinematics, dynamics and calibration) across computer simulation and real-time control of arbitrary robot configurations (Chen et al. 1999). The basic graphical user interface of *SEMORS* is illustrated in Figure 14. *SEMORS* is intended to be a uniform interface for all modular robots and is portable to modular robot systems from different vendors. It will be used both for simulation and for on-line execution of a task, regardless of whether the robot is executing (or is simulated to be executing) the task as a stand-alone application, or as part of a workcell process. Thus, it allows the user to quickly integrate the hardware components into modular robots, and to manage their operations in the reconfigurable workcell. Key features of *SEMORS* include:

- Module and robot builder
- D graphical task simulation
- "Universal" inverse kinematics
- Full dynamics models
- Trajectory and task planning
- Transparent workcell network connectivity

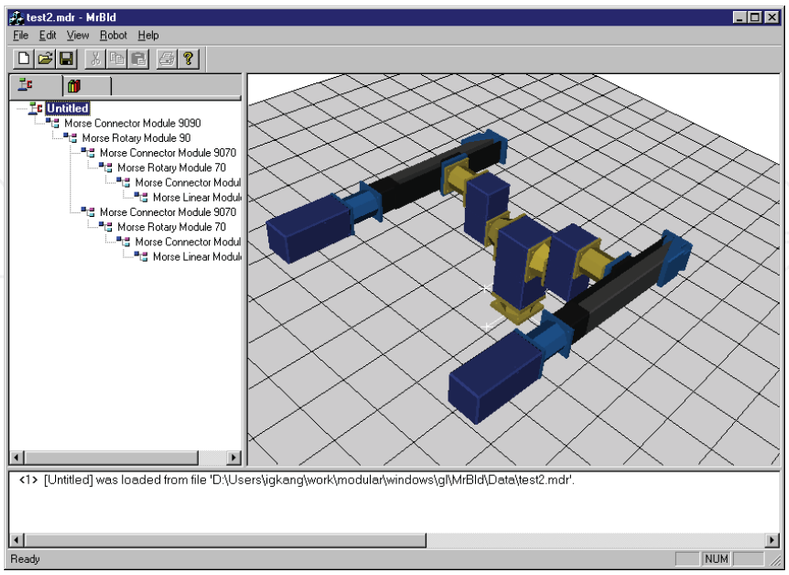


Figure 14. User interface of SEMORS

In addition to the simulation of modular robots, extended features like robot configuration planning/optimization and module database management are implemented as separate application packages to be used along with *SEMORS*. The task-based robot configuration optimization mentioned in Section 5 is a generic and platform-independent methodology. With the capability of task-based robot configuration optimization, designing the modular robot configuration using *SEMORS* becomes no longer an ad hoc approach. The software system will provide end-user an optimized robot configuration according to the input task requirements. The user does not need to start the design work from scratch. Rather, based on the result of optimization, he can fine-tune the suggested robot design or layout. The development effort and time for the workcell can be greatly reduced.

7. Prototype of Reconfigurable Robotic Workcell

To effectively demonstrate the use of a modular reconfigurable robotic system, we have constructed a prototype workcell for light-machining tasks in an industrial exhibition in 1999 (Figure 15). This workcell was built with multiple reconfigurable robots along with other supporting devices under a unified modular approach.

Figure 15. A light machining workcell with modular robot components

- Preliminary design stage

To make use of the advantages of both parallel-typed and serial-typed robots, we intend to make the workcell to perform a complete milling operation of a workpiece, starting from picking up the object, transferring the object to a milling robot, starting the milling process, and returning the workpiece back to a storage rack. Based on this preliminary concept, we decide to use two reconfigurable robots in this workcell: one is a serial-typed robot for the pick-and-place operation, and the other is a parallel-typed robot for the milling operation because of its structural rigidity. The task is to perform milling operation on a dome-shaped top of a cylindrical workpiece with 15cm in diameter. A workpiece transfer system should be used in between the two robots.

- Robot configuration selection and construction

Based on the preliminary task description, the workcell is configured with a 7-DOF redundant serial-type robot, a 6-DOF articulate RRRS parallel robot, and a 1-DOF linear motion stage. From the robot configuration optimization, a 4-DOF SCARA-type robot is sufficient to perform the task. Deploying a redundant robot here is to demonstrate that the proposed model generation algorithms used in *SEMORS* and in robot control are universally applicable for any configuration.

The configuration design of the parallel robot follows a systematic approach (Yang et al. 1999). In principle, a 3-branch parallel structure is used because of the structure stiffness and dexterity. Each branch consists of three rotary joints

(two are active and one is passive) and a passive spherical joint. Once the geometry is determined, the workspace analysis is performed. From the result of this analysis, the lengths of the rigid links and connectors are determined. Because of the modular design, the actuator modules can be freely located at the nine revolute joints. The workspace of the robot changes according to the locations of the actuator modules. A disk-shaped moving platform is attached to the three branches. An end-mill tool actuated by an intelligent motor is mounted at the center of the platform. This motor uses the same control interface as the standard actuator modules. Because of the lack of the force sensor, the task is only carried out in simulated manner, i.e., the end-mill tool only goes through the milling path without touching the surface of the workpiece.
The 1-DOF linear motion stage uses two standard modules: one rotary module to drive the linear slide and one gripper module to hold the workpiece, to ensure uniformity in the workcell control. The specifications of the robots and the motion stage are listed in Table 1.

- Workcell construction and fine-tuning

After the robots and motion stage are constructed, the robot controllers are connected to the robots. Two Pentium II-based industrial PC robot controllers are used to perform high-level trajectory control of the serial robot and the parallel robot respectively. The kinematic models of both serial and parallel robots are generated automatically in ***SEMORS*** and stored in the robot controllers. Kinematic calibration of both robots is performed before the operation. The kinematic calibration is conducted by using articulate-typed coordinate measuring equipment, called "Spin Arm". The obtained calibration data is transferred to the robot controller and then ***SEMORS*** computes and updates the corrected kinematic models of the robots automatically. Because of its simplicity, the control of the motion stage is done by one of the robot controller for this implementation.

- Finalize task sequence and control of the workcell actions

With updated kinematic models, the detailed task sequence of all robots (Table 2) is laid out. The tasks are then programmed into the respective robot controllers. The two robot controllers are connected to a closed-loop workcell LAN running at 10MB/sec. A separate notebook computer is also connected to the workcell network performing supervisory control of the workcell through *SEMORS* running on the individual robot controllers. The task sequence of the workcell is monitored and supervised by the notebook supervisor.
Based on the actual construction, to assembly the described 7-DOF serial-typed robot takes two users about 30 minutes. The time to construct the parallel robot requires two persons about two hours because of the complexity of the structure. Adding the time to install the motion stage, calibrate the robots and fine-tune the workcell hardware, it will take about four hours in total to com-

plete the entire workcell set-up excluding the time spent on the preliminary design stage.

Light-machining Workcell	
7-DOF Redundant Serial Robot	
Work envelope	Approx. sphere, SR = 1200mm
Max speed	750 mm/s
Repeatability	+/- 0.10 mm
Max Payload	5 Kg (excluding end-effector)
Weight	16 Kg (excluding base)
6-DOF RRRS Articulate Parallel Robot	
Work envelope	Approx. hemisphere, SR = 500mm
Max speed	500 mm/s
Repeatability	+/- 0.05mm
Max Payload	25 Kg (excluding end-effector)
Weight	30 Kg (excluding base)
1-DOF Linear Motion Stage	
Effective stroke	L = 1500mm
Max speed	500 mm/s
Repeatability	+/- 0.025mm
Max Payload	45 Kg (excluding fixture)
Weight	35 Kg

Table 1. Specifications of the light-machining workcell

8. Conclusion

We have presented a generic method to automate the model generation for modular reconfigurable robots based on a graph representation of robot geometry, called an assembly incidence matrix, and geometry-independent model building algorithms for the kinematics, dynamics and error models of a robot. The AIMs of the assembly configuration of modular robots facilitate the determination of optimal robot configuration for a specific task using combinatorial optimization techniques. We also presented here an approach to solve the task optimal problem using evolutionary algorithms with customized genetic operators based on the AIM of the robot. The application of this automatic modeling technique is implemented in a modular robot control and simulation software application, *SEMORS* (Simulation Environment for MOdular Robot Systems). In this software system, it is not necessary to maintain a library of robot models, since the possible assembly configurations of a robot is not a fixed number. Instead, only a small set of component database and kernel functions are kept in the robot controller, and required robot models are generated automatically. From the prototype construction, we can con-

firm the advantage of using modular components in constructing the complex robotic workcell with different configurations. The plug-and-play kinematics, dynamics, and calibration robot models are also verified through the actual implementation in the robot controller and the simulation software.

Acknowledgment:

The authors would like to acknowledge work done by other members of the project: Prof. Guang Chen, Dr. Peter Chen, Dr. Weihai Chen, Dr. Wei Lin, Mr. In-Gyu Kang, Mr. Wee Kiat Lim, Mr. Edwin Ho, Mr. S. Ramachandran, Ms. Yan Gao, and Mr. Chee Tat Tan. This project is supported by Singapore Institute of Manufacturing Technology (Upstream Project U97-A006), and Ministry of Education, Singapore (RG64/96 and JT ARC 7/97).

Task Sequence	
1	Robot A picks up workpiece from fixture
2	Robot A places workpiece on the motion stage
3	Motion stage moves workpiece under Robot B
4	Robot B performs milling task
5	Robot A shifts locations of un-processed workpieces
6	Robot B finishes milling task
7	Motion stage moves processed workpiece back
8	Robot A picks up processed workpiece from motion stage
9	Robot A places processed workpiece to the fixture

*Robot A: 7-DOF serial robot, Robot B: 6-DOF parallel robot

Table 2. Task Sequence

9. References

Ambrose, R. O. (1995). Interactive robot joint design, analysis and prototyping, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2119-2124, Washington DC, USA.

Benhabib, B.; Zak, G. & and Lipton, M. G. (1989). A generalized kinematic modeling method for modular robots, *Journal of Robotics Systems*, Vol. 60, No. 5, pp. 545-571.

Chen, I.-M. (1994). Theory and Applications of Modular Reconfigurable Robotic Systems, PhD thesis, California Institute of Technology, Division of Engineering and Applied Science, U. S. A.

Chen, I.-M. (1996). On optimal configuration of modular reconfigurable robots, *Proc. Int. Conf. Control, Automation, Robotics, and Vision*, pp. 1855-1859, Singapore.

Chen, I.-M. (2000). Realization of a rapidly reconfigurable robotic workcell, *Journal of Japan Society of Precision Engineering*, Vol. 66, No. 7, pp. 1024-1030.

Chen, I.-M. (2001). Rapid response manufacturing through reconfigurable robotic workcells, *Journal of Robotics and Computer Integrated Manufacturing*, Vol. 17, No. 3, pp. 199-213.

Chen, I.-M. & Burdick, J. W. (1995). Determining task optimal modular robot assembly configurations. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 132-137, Nagoya, Japan.

Chen, I.-M. & Burdick, J. W. (1998). Enumerating Non-Isomorphic Assembly Configurations of a Modular Robotic System, *International Journal of Robotics Research*, Vol. 17. No. 7, pp. 702-719.

Chen, I.-M. & Yang, G. (1996). Configuration independent kinematics for modular robots. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1845-1849, Minneapolis, MN, USA.

Chen, I.-M. & Yang, G. (1997). Kinematic calibration of modular reconfigurable robots using product-of-exponentials formula, *Journal of Robotic Systems*, Vol. 14, No. 11, pp. 807-821.

Chen, I.-M. & Yang, G. (1999). Numerical inverse kinematics for modular reconfigurable robots, *Journal of Robotics Systems,* Vol. 16, No. 4, pp. 213-225.

Chen, I.-M.; Yang, G.; Yeo, S. H. & Chen, G. (1999). Kernel for modular robot applications - automatic modeling techniques, *International Journal of Robotics Research*, Vol. 18, No. 2, pp. 225-242.

Chen, I.-M.; Tan, C. T.; Yang, G. & Yeo, S. H. (2001). A local POE model for robot kinematic calibration, *Mechanism and Machine Theory,* Vol. 36, No. 11, pp. 1215-1239.

Cohen, R.; Lipton, M. G.; Dai, M. Q. & Benhabib, B. (1992). Conceptual design of a modular robot, *ASME Journal Mechanical Design*, Vol. 114, pp. 117-125.

Cormen, T.; Leiserson, C. & Rivest, R. (1990). *Introduction to Algorithms*, ISBN 0262032937, MIT Press, Cambridge, MA, USA.

Deo, N. (1974). *Graph Theory with Applications to Engineering and Computer Science*, ISBN 0133634736, Prentice-Hall, New York, USA.

Dobrjanskyj, L. & Freudenstein, F. (1967). Some applications of graph theory to the structural analysis of mechanisms, *ASME Journal Engineering for Industry*, Vol. 89, pp. 153-158.

Featherstone, R. (1987). *Robot Dynamics Algorithms*, ISBN 0898382300, Kluwer Academic Publishers, Holland.

Fukuda, T. & Nakagawa, S. (1988). Dynamically reconfigurable robotic system, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1581-1586, Philadelphia, PA, USA.

Hollerbach, J. M. (1980). A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 10, pp. 730-736.

Kelmar, L. & Khosla, P. (1988). Automatic generation of kinematics for a reconfigurable modular manipulator system, *Prof. IEEE Int. Conf. Robotics and Automation*, pp. 663-668, Philadelphia, PA, USA.

Khosla, P. K.; Neuman, C. & Prinz, F. (1985). An algorithm for seam tracking applications, *International Journal of Robotics Research*, Vol. 40, No. 1, pp. 27-41.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, 2ed., ISBN 540580905, Springer-Verlag, Berlin, Germany.

Murray, R.; Li, Z. & Sastry, S. (1994) *A Mathematical Introduction to Robotic Manipulation*, ISBN 0849379814, CRC Press, Boca Raton, FL, USA.

Paredis, C. J. J.; Brown, H. B. & Khosla, P. (1997). A rapidly deployable manipulator system, *Robotics and Autonomous Systems,* Vol. 21, No. 3, pp. 289-304.

Paredis, C. J. J. & Khosla, P. K. (1995). Design of modular fault tolerant manipulators, In: *Algorithmic Foundations of Robotics*, Goldberg, K. (ed.), pp. 371-383, A. K. Peters, ISBN 1568810458, Wellesley, MA, USA.

Park, F. C. & Bobrow, J. E. (1994). A recursive algorithm for robot dynamics using lie groups, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1535-1540, San Diego, CA, USA.

Park, F. C.; Bobrow, J. E. & Ploen, S. R. (1995). A lie group formulation of robot dynamics, *International Journal of Robotics Research*, Vol. 14, No. 6, pp. 609-618.

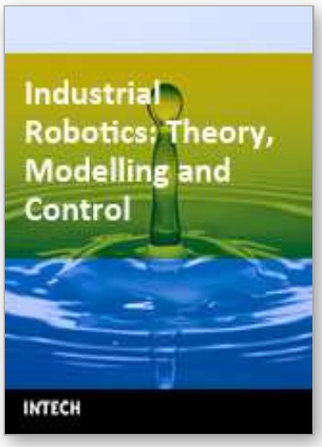
Rodriguze, G.; Jain, A. & Kreutz-Delgado, K. (1991). A spatial operator algebra for manipulator modeling and control, *International Journal of Robotics Research*, Vol. 10, No. 4, pp. 371-381.

Schmitz, D.; Khosla, P. K. & Kanade, T. (1988). *The CMU reconfigurable modular manipulator system*, Technical Report CMU-RI-TR-88-7, Robotics Institute, Carnegie Mellon University.

Wurst, K. H. (1986). The conception and construction of a modular robot system. *Proc. 16th Int. Sym. Industrial Robotics (ISIR)*, pp. 37-44, Brussels, Belgium.

Yang, G. & Chen, I.-M. (2000). Task-based optimization of modular robot configurations – MDOF approach, *Mechanism and Machine Theory*, Vol. 35, No. 4, pp. 517-540.

Yang, G.; Chen, I.-M.; Lim, W. K. & Yeo, S.H. (1999). Design and kinematic analysis of modular reconfigurable parallel robots, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2501-2506, Detroit, MI, USA.



Industrial Robotics: Theory, Modelling and Control

Edited by Sam Cubero

ISBN 3-86611-285-8

Hard cover, 964 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, December, 2006

Published in print edition December, 2006

This book covers a wide range of topics relating to advanced industrial robotics, sensors and automation technologies. Although being highly technical and complex in nature, the papers presented in this book represent some of the latest cutting edge technologies and advancements in industrial robotics technology. This book covers topics such as networking, properties of manipulators, forward and inverse robot arm kinematics, motion path-planning, machine vision and many other practical topics too numerous to list here. The authors and editor of this book wish to inspire people, especially young ones, to get involved with robotic and mechatronic engineering technology and to develop new and exciting practical applications, perhaps using the ideas and concepts presented herein.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen