NIST Technical Note 1258

Manipulator Servo Level World Modeling

Laura Kelmar

Robot Systems Division
National Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

December 1989

U.S. Department of Commerce
Robert A. Mosbacher, Secretary

National Institute of Standards and Technology
Raymond G. Kammer, Acting Director

National Institute of Standards
and Technology
Technical Note 1258
Natl. Inst. Stand. Technol.
Tech. Note 1258
31 pages (Dec. 1989)
CODEN: NTNOEF

U.S. Government Printing Office
Washington: 1989

For sale by the Superintendent
of Documents
U.S. Government Printing Office
Washington, DC 20402

Table of Contents

1. Introduction **1**

2. Module Interfaces **5**

2.1. World Modeling to Sensory Processing Interface 5
2.2. World Modeling to Operator Interface 7
2.3. Servo (Level 1) World Modeling to Prim (Level 2) World Modeling Interface 9
2.4. World Modeling to Servo Task Decomposition Interface 9
2.4.1. World Modeling to/from Task Decomposition Job Assignment(JA) 9
2.4.2. World Modeling to/from Task Decomposition Planning(PL) 9
2.4.3. World Modeling to/from Task Decomposition Execution 9

3. World Modeling Module Operation: Level 1 Sensory Processing Support **10**

4. World Modeling Module Operation: Servo Support **11**

4.1. Module Operation: Kinematics 11
4.1.1. Forward Kinematics 13
4.1.2. Manipulator Jacobian and Inverse Jacobian 14
4.1.3. Force and Torque Transformations 17
4.2. Module Operation: Dynamics 17
4.3. Module Operation: Control 19

5. Kinematics in Detail **20**

5.1. Forward Kinematics 21
5.2. Inverse Kinematics for Redundant Manipulators 23

6. Conclusions **24**

7. References **24**

1. Introduction

This document describes the function, interfaces, and structure of a World Modeling module at the lowest level of a hierarchical manipulator control system. The module described in this document is part of a hierarchical control system in which complex tasks are decomposed into simpler and simpler subtasks, or objectives, as described in [ALB87]. The system is divided vertically into levels based on the complexity of the objectives performed within the level. Furthermore, each level is subdivided horizontally into columns: Task Decomposition, World Modeling, and Sensory Processing. The control system is shown in figure 1.

World Modeling maintains the system's internal model of the world by continuously updating the model based upon sensory data. It consists of support processes or functions which simultaneously and asynchronously support Sensory Processing and Task Decomposition. The term *world model* refers to all the support processes, together with the global data system. Figure 2 elucidates the allocation of processes in the control system, through the Task Level, for a telerobot. There are computational hierarchies of Task Decomposition, Sensory Processing and World Modeling modules, for each device and each actuator. The modules can be logically recombined according to their function in the system, as shown in figure 3. The system pictured consists of two main *branches*; the left branch contains the perception processes and the right branch contains the manipulation processes. The two branches decompose tasks independently and communicate via the global data system.

The lowest level of the Task Decomposition hierarchy is called the Servo Level and is responsible for handling small dynamic motions of the manipulator, gripper, and all other devices which require servo control. It computes the motor control signals for the actuators based upon the command attractor set of desired positions, velocities, and accelerations [FIA88]. The lowest level of Sensory Processing, Level 1, gathers and filters raw data from the sensors. The Servo Level Task Decomposition module and the Level 1 Sensory Processing module are supported by and interfaced through the Servo Level World Modeling processes.

A manipulator control system, such as the one described in [ALB87], allows for many controlled devices, as well as many sensors. For example, the Task Decomposition hierarchy may provide servo control for the camera lens, the gripper, and the manipulator joints, while the Sensory Processing hierarchy may process camera images, tactile arrays, and manipulator joint data. In general, each device requires separate World Modeling support at each level. That is, a complete pictorial representation of the control hierarchy would contain a separate World Modeling box for each device and associated sensor. The scope of this document is limited to the discussion of the World Model module at the Servo Level of a manipulator control system, as highlighted in figure 3. We briefly discuss the function of the Level 1 Sensory Processing module in supplying manipulator sensor readings.

Servo Level World Modeling maintains the following: the kinematic and dynamic models, the current joint positions, velocities, forces and torques. All readings and computations are recorded with respect to the specified coordinate system. Also, in support of Sensory Processing, World Modeling maintains filtering algorithm parameters and histories of the sensor readings. The Servo Level World Modeling module interfaces to the

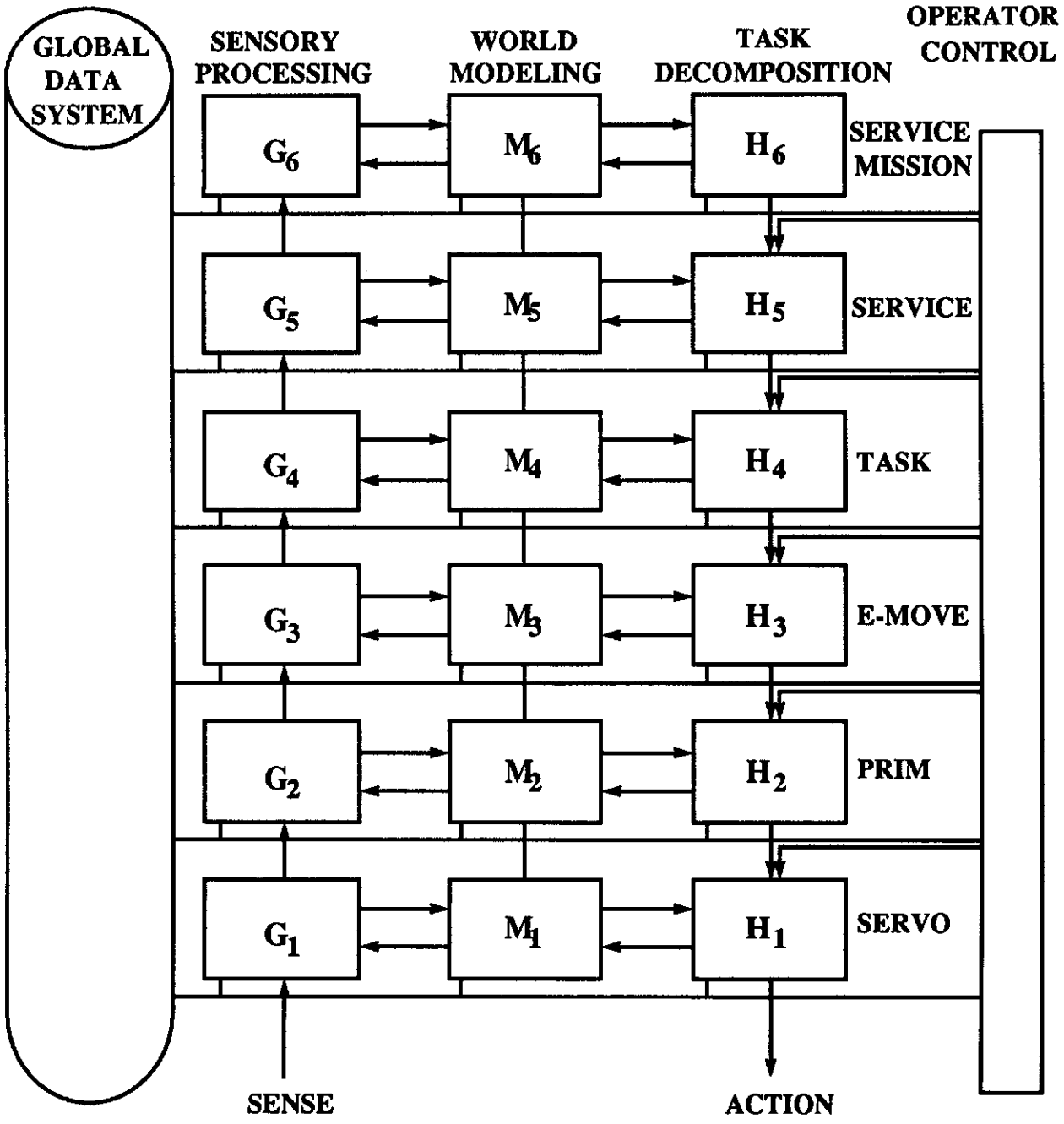

Figure 1. Hierarchical Control System Architecture.

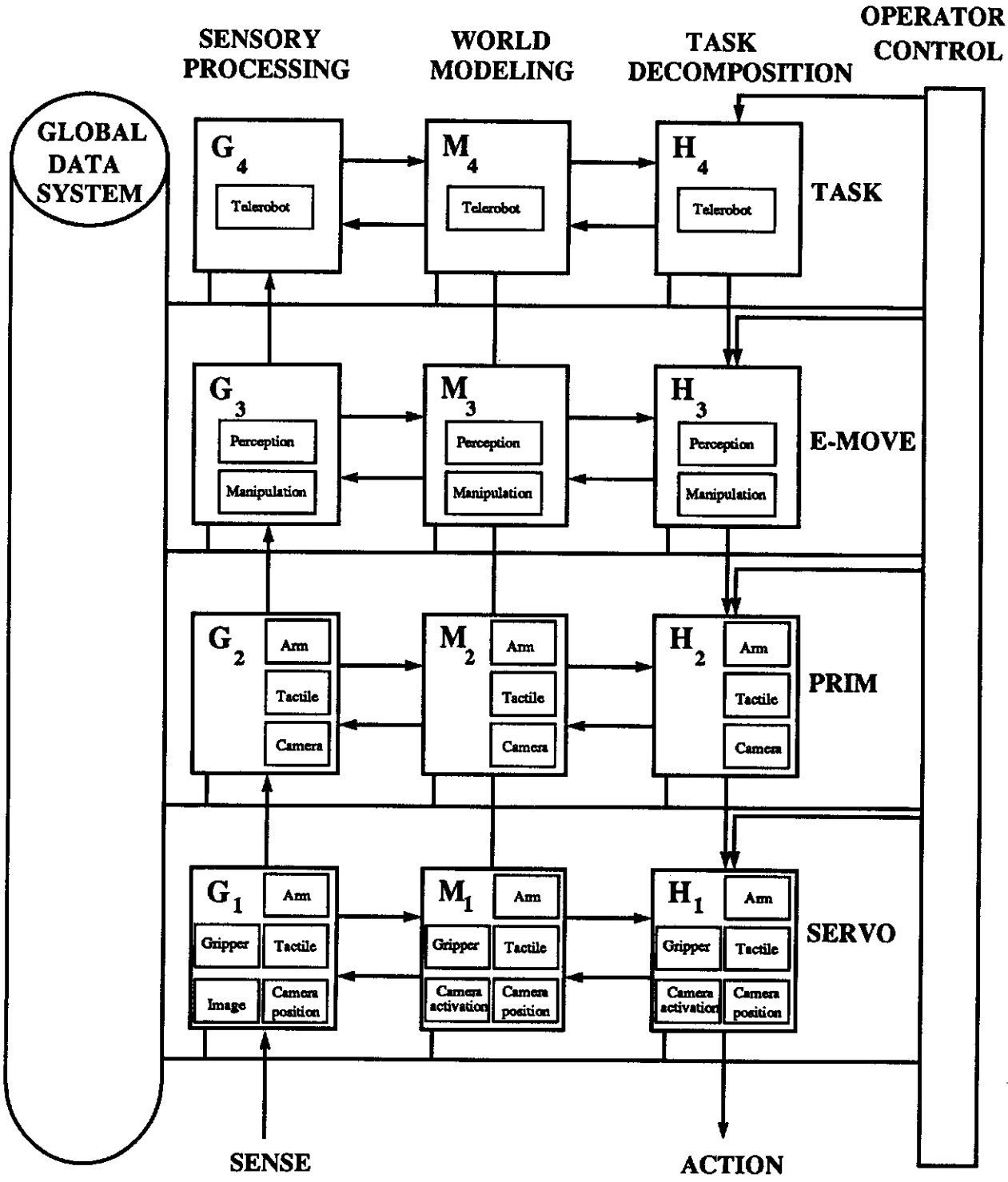

Figure 2. Modules for Each Sensor and Each Actuator.

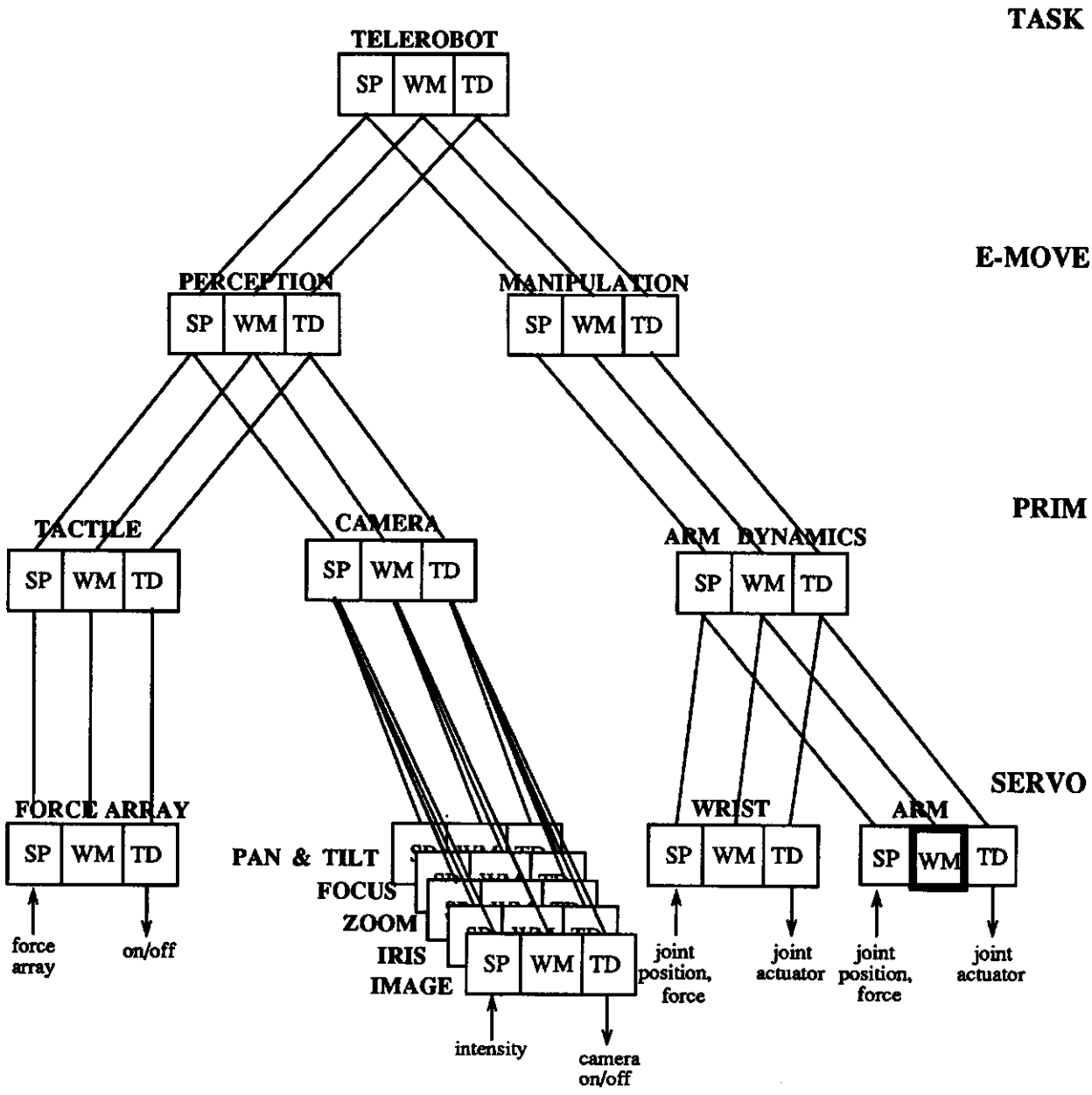

Figure 3. Perception and Manipulation Branches.

following: the Level 1 Sensory Processing module, the Servo Level Task Decomposition module, the system operator, and the Level 2 World Modeling module. The interfaces to/from Task Decomposition are further broken down as interfaces to/from the Job Assignment sub-modules, the Planning sub-modules and the Executor sub-modules as explained in [FIA88]. Section 2 of this document defines the interfaces between World Modeling and the rest of the control hierarchy. Sections 3, 4, and 5 discuss the classes of support processes for both Sensory Processing and Task Decomposition for a manipulator (the most complex device) and its sensors. This document does not provide an exhaustive list of the computations and models required by World Modeling in support of all possible devices and sensors.

2. Module Interfaces

The Servo Level World Modeling module has several interfaces: World Modeling supplies (filtering) algorithm parameters and histories of readings to the Level 1 Sensory Processing which in turn writes sensor readings to the global data system. The operator control transmits data related to the selected coordinate system and the servo algorithm directly to the World Modeling module. Within World Modeling, information is passed between the Level 1 (Servo) module and the Level 2 (Prim) module. Finally, the Servo Level Task Decomposition module passes control algorithm parameters, via global memory, to World Modeling. World Modeling writes the results of its support computations to the global data system for use by Task Decomposition. In addition, each request for information receives a status report in return. In most cases, the status is implied and not explicitly labelled with the interfaces in figure 4. We discuss the Servo Level World Modeling interfaces, as shown in figure 4, in the following sections.

2.1. World Modeling to Sensory Processing Interface

This section describes the interface between the World Modeling and Sensory Processing modules at Level 1. Figure 4 shows the interface, as well as the internal structure of the modules. In the figure, the Sensory Processing module includes four sub-modules. However, sensory processing of manipulator sensor readings (joint angles) probably would not involve *spatial integration* of readings or *detection* of events. It also may not include temporal integration or averaging of readings over time; in these cases the processes are null. The reader is referred to the document on Level 1 Sensory Processing for a camera for details on the module's structure and function [CHA89]. A brief description follows.

The comparators receive data from the sensors and predictions from the world model. Predictions for sensor joint readings might simply be threshold values with which to filter the data. The comparators then perform an algorithm specific computation on the data, such as comparison against a threshold value, and store the results in the world model.

By definition, temporal integrators combine the results of the comparators over a given time window. Temporal integration produces values which represent the average value, over time, of the readings from a particular sensor. Most likely temporal integration would not be applicable for manipulator joint readings, which typically are updated every sampling period. Also, event detection is not applicable for manipulator sensor readings. The readings undergo no further processing and are stored in the world model.

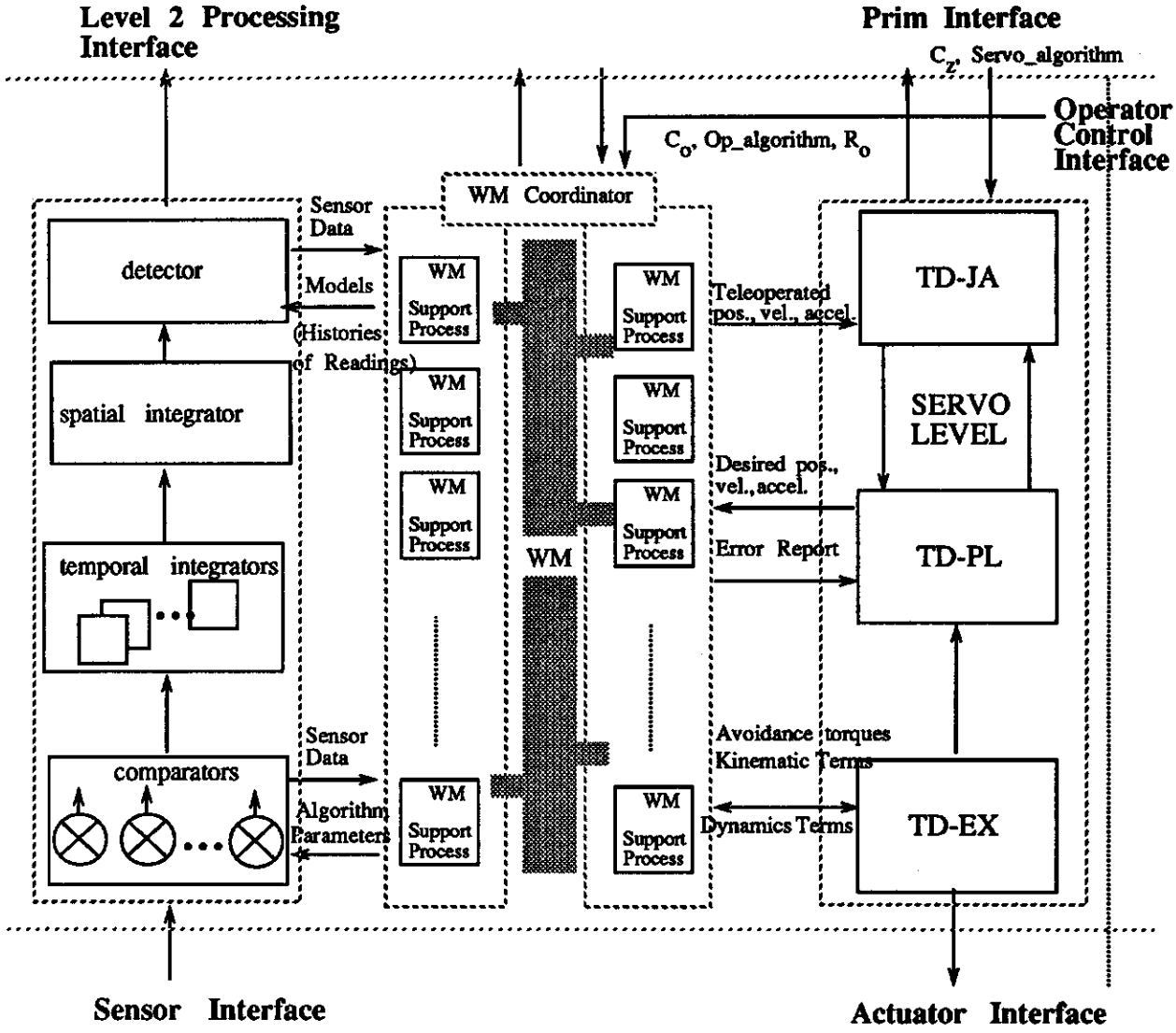

Figure 4. Servo Level World Model Interfaces.

The information which is passed from the World Modeling module to the Sensory Processing module represents requests for data which specify a starting criterion and an ending criterion for sensor readings, as well as sensor update rate. Typically, World Modeling (in support of Servo Task Decomposition) requires continuously updated manipulator joint readings. Therefore, the Level 1 Sensory Processing module for manipulator joint sensors operates in a default mode which continuously processes the joint readings. Also, the specification of the filtering technique and parameters for processing the readings would exist as a default setting. A change of setting would result from feedback from Task Decomposition indicating poor results or a change in time constraints.

Input to World Modeling from Level 1 Sensory Processing consists of filtered, or enhanced sensor readings. Manipulator joint data includes the positions (θ_i), velocities ($\dot{\theta}_i$), and torques (τ_i), where i denotes the joint number. Force sensors mounted on the wrist of the manipulator measure the forces (f_x,f_y,f_z) and torques (m_x,m_y,m_z) acting upon the end-effector.

Associated with each reading is a sensor identification number which includes both the sensor type and the instance of the sensor. For example, the sensor identification number might specify that the reading is coming from a joint sensor, and specifically the sensor at the n^{th} joint of the m^{th} manipulator in the system. In the most general case, each sensor reading has an associated timestamp. (It may not be necessary for all data.) While the timestamp may not be used by Task Decomposition at the Servo Level, it is necessary for temporal processing of the data. Table 1 contains a list of example parameters for the Level 1 Sensory Processing to World Modeling interface for a manipulator.

Table 1. Sensory Processing to World Modeling Interface

READING	DESCRIPTION
$\theta_1,\theta_2,\ldots,\theta_N$	Manipulator Joint Positions
$\dot{\theta}_1,\dot{\theta}_2,\ldots,\dot{\theta}_N$	Manipulator Joint Velocities
$\tau_1,\tau_2,\ldots,\tau_N$	Manipulator Joint Torques
Strain gauge readings	Wrist Forces and Torques

2.2. World Modeling to Operator Interface

The World Modeling module receives information from the operator regarding the

specification of the coordinate system for the position and force commands and information regarding the mode of operation. This information is essential to the World Modeling module for computing the kinematic, dynamic, and control variables. Because the control architecture supports both teleoperation and autonomous operation, the control operator must instruct both the Task Decomposition and the World Modeling modules as to the mode of operation [ALB87].

For teleoperated control the parameter C_o indicates the coordinate system of operation. For autonomous control the parameter C_z, which is passed to World Modeling by Servo Level Task Decomposition, indicates the coordinate system. If the manipulator is to be controlled with shared control (autonomous and teleoperated), the parameter R_o may be needed to resolve redundancy between the systems. Specifically, if the operator coordinates (C_o) are underspecified with respect to the autonomous coordinates (C_z), redundancy resolution must be used to map between the systems. The choices for C_z and C_o are listed in Table 2 shown below. Each choice of coordinate system is considered in section 4.1.2, where we discuss the World Modeling module's process for computing the manipulator's Jacobian.

The transformations T_w and T_{ee} are fixed, rigid-body, homogeneous transformations which locate an object or tool-tip. If the actual manipulator base or end-effector is to be used as the reference point, then they are simply identity transformations.

Table 2. Choices for C_o and C_z

SYSTEM	DESCRIPTION
(joint)	Commands in manipulator's joint space
(end-effector,_, T_{ee})	Commands in Cartesian system fixed at a frame with relation T_{ee} to end-effector
(World,T_w,T_{ee})	Commands in a frame with relation T_w to base for Cartesian system with relation T_{ee} to end-effector

In addition to the coordinate system parameters, the operator passes the parameter Op_algorithm to direct the mode of operation. If Op_algorithm specifies teleoperation, then the job assignment module directs the manipulator control based upon input from the joystick or other input device [FIA88]. For autonomous operation, the job assignment module takes commands from the Primitive Level Task Decomposition module and Servo_algorithm selects the control algorithm.

2.3. Servo (Level 1) World Modeling to Prim (Level 2) World Modeling Interface

The World Modeling modules at Level 1 and Level 2 interface when the support column functions hierarchically to achieve a common end. For example, in order to compute obstacle avoidance torques, the Prim World Modeling support module (Level 2) might provide the Servo support module (Level 1) with the necessary parameters or equations. (The update rate of the Servo Level support may be too great to allow for the extensive computations.)

2.4. World Modeling to Servo Task Decomposition Interface

This section describes the interface between World Modeling and Task Decomposition at the Servo Level. The Task Decomposition module is further decomposed into submodules (Job Assignment, Planning, Execution), each of which interfaces to World Modeling. We describe each of the submodule interfaces separately. Additional explanations of the uses of the parameters and how to compute them can be found in the discussion of World Modeling operation in sections 4 and 5.

2.4.1. World Modeling to/from Task Decomposition Job Assignment(JA)

When the system operates in teleoperation mode, or shared control mode, the vectors of desired position, velocity, acceleration, and force (z_o, $\dot{z}_o$, $\ddot{z}_o$, f_o) for the manipulator are received by the World Modeling module from the input device and then passed to the job assignment module. These vectors are expressed with respect to the coordinate system C_o. For pure teleoperation, they represent the state of the master arm or other input device which serves as the attractor set for the slave (manipulator).

2.4.2. World Modeling to/from Task Decomposition Planning(PL)

The World Modeling module receives the vector of desired position, velocity, acceleration, and jerk (z_d, $\dot{z}_d$, $\ddot{z}_d$, $\dddot{z}_d$) for the manipulator from the Task Decomposition planning module. These vectors are received and stored by World Modeling so that they may be used to compute feedforward compensation terms (for both autonomous and teleoperation modes), as described in section 4.3. The vectors specify goal the state of the manipulator to be used in autonomous control mode, or shared control mode. They may also be used by Task Decomposition for error checking or interpolation.

2.4.3. World Modeling to/from Task Decomposition Execution

The World Modeling module transforms the sensory joint position and velocity readings (z, $\dot{z}$) into the coordinate system specified by Task Decomposition (C_z). The vector of forces and torques at the manipulator's end-effector (f_x,f_y,f_z,m_x,m_y,m_z), is also transformed into the proper coordinate system and made available. The coordinate system specification, C_z, indicates the coordinate system in which the position and force command vectors are expressed, as well as the system in which the servo errors are computed. World Modeling

makes the sensory readings available to the execution module for computation of the servo error terms; it resolves any differences between the coordinate system of the readings (Sensory Processing) and that of the computations (Task Decomposition). For example, if servo executes a Cartesian space control scheme, it is desirable for World Modeling to supply the current manipulator position in Cartesian space. Assuming that Sensory Processing reads joint positions (velocities), World Modeling would calculate the current Cartesian position of the manipulator from the joint space sensory readings.

The World Modeling module supplies the execution module with model-based terms necessary for the dynamics calculation of the control torques. The dynamics terms include the matrix of centrifugal and Coriolis coefficients, a vector of gravity compensation terms, the inertia coefficients, and friction terms.

The manipulator Jacobian and inverse Jacobian may also be needed by the execution module. (If C_z is joint space, then the Jacobians are identity; the desired workspace is equivalent to the workspace of the sensor readings.) If the manipulator is redundant, with respect to its workspace, then computing the inverse Jacobian involves resolving the redundancy. The Moore-Penrose generalized inverse (pseudoinverse) is the most commonly used replacement for the inverse of the Jacobian.

The World Modeling module also supplies any avoidance torques which may be used by the execution module when computing the control torques. The nature and purpose of these torques are explained in [FIA88] and are reiterated in section 4.3 where we discuss how to compute them.

Finally, if an adaptive control scheme is to be implemented, the execution module sends the vector of actuator torques, τ_{act}, to the World Modeling module. The vector represents the torques from the previous sampling period and is used in order to adapt the control algorithm to reduce the error between the desired end-effector position (velocity) and the realized one.

This completes our discussion of the Servo Level World Modeling module's interfaces. We have mentioned some of the operations which must be performed within the World Modeling module. In the next section we elaborate upon the operation of the module.

3. World Modeling Module Operation: Level 1 Sensory Processing Support

The Level 1 Sensory Processing module for manipulator sensors is supported by World Modeling. After the Sensory Processing module filters the data, the World Modeling module accepts the filtered data and transforms it, if necessary, to the coordinate system specified by Task Decomposition (C_z). After the data is transformed, it is stored in the global data system for use by the Task Decomposition module, as well as the World Modeling processes.

The Sensory Processing system often contains force and torque sensors. Force and torque sensing can take place at the joints or the wrist of the manipulator, with wrist sensing being the most widely used. Sensory Processing filters the force and torque data in its "raw form" and then stores the values in the global data system. World Modeling converts the

data from strain gauge readings to forces and torques. The conversion is based upon a model of the position and orientation of the gauges within the force/torque sensor. The gauges measure the forces perpendicular to the plane in which they lie. The measured forces (W) are converted to wrist (joint) forces and torques (F) through a force calibration matrix (R_W) according to:

$$F = R_W W.$$

4. World Modeling Module Operation: Servo Support

Each World Modeling support process can be thought of as a cyclically-executing procedure or function which works upon certain inputs to produce the required values. Task Decomposition directs the operation of the World Modeling support processes with the algorithm parameters contained in the global data system. The processes are coordinated (activated and deactivated) by a World Modeling coordinator, as shown in figure 5. Each support process and each Task Decomposition process, JA, PL, and EX, executes cyclically; all processes communicate through global memory. The ovals in the figure 5 represent the global memory.

Before specifying the World Modeling operation, one must choose a method for representing translations and rotations in the system as a whole. Such representations are used for defining the kinematics and dynamics of the manipulator. They are also necessary for defining positions and orientations in all facets of task planning, including representing objects/frames in the manipulator's workspace.

Many techniques exist for representing translations (positions) and rotations (orientations) [ASA86, DEN55, FUN88, KAN83]. The most common approach to the problem is to use vectors. While vectors lend themselves to expressing translational information, they do not lend themselves to representing 3-D rotations. Thus, spatial transformations typically have taken the form of 4x4 homogeneous transformations, which contain a 3x3 rotational submatrix and a 3x1 positional vector. However, a 3x3 matrix representing orientation in 3-D space necessarily contains redundant information. In order to expedite calculations with rotations in robotics, alternate methods using quaternions and Lie algebra have been proposed to remove the redundancy [FUN88, GU88, PER82].

The discussion of the World Modeling functions that follows is intended to be general. That is, we do not make assumptions about the data structures (representations), the target hardware, or other characteristics of the system or manipulator. Each of the following sections contains an explanation of an operation of the World Modeling, as well as some of the methods for implementing it. We conclude each section with a discussion of the computational considerations for the World Modeling operation.

4.1. Module Operation: Kinematics

Typically, a manipulator task is specified in World (or Cartesian) space. Specifying a task in Cartesian space necessitates a kinematic model to define the relationship between the position and orientation of the end-effector and the joint angles. The complete kinematic model describes all the geometrical and time based motions of the manipulator, without

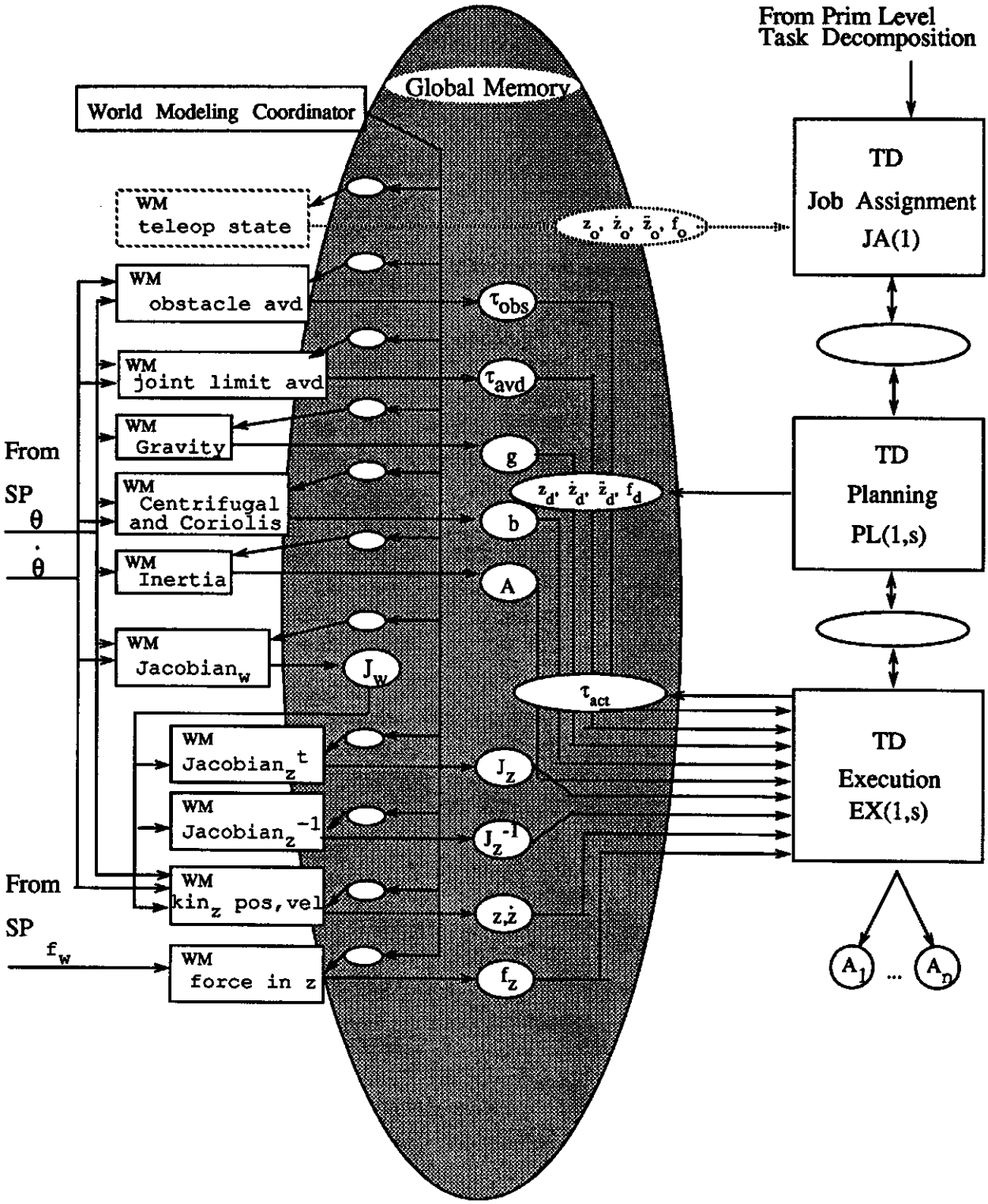

Figure 5. World Modeling Servo Support Processes.

regard for the forces acting upon the manipulator. We begin this section by discussing several methods for creating a forward kinematic model. We then consider the problem of creating the manipulator Jacobian which relates the differential changes in the joint angles to the differential changes in the position and orientation of the end-effector.

4.1.1. Forward Kinematics

For each of the forward kinematic systems we assume that the joint angles (or displacements in the case of prismatic joints) are the variables obtained from the sensors. That is, that the joints are directly actuated. In some cases, however, the joint position must be calculated from the actuator position. For example, a revolute joint may be rotated by a linear actuator through bar linkages [CRA86]. Therefore, an additional transformation may be required from actuator space to joint space, as shown in figure 6.

In general, forward kinematic techniques systematically determine the transformation (position and orientation) between adjacent links in a manipulator. By starting at the base and successively applying (multiplying) the transformations, one can determine the position and orientation of the end-effector with respect to the base as a function of the joint variables.

While many methods exist, the Denavit-Hartenberg system is the most frequently used. (For details regarding the Denavit-Hartenberg system, see sec. 5.) Its popularity is due in part to its clear physical interpretation and to the fact that it uses the minimum number of parameters to specify the kinematics of a manipulator. In addition, it allows for easy derivation of the manipulator Jacobian [PAU81].

Although the kinematic representation created with the Denavit-Hartenberg system uses the minimum number of parameters to specify each transformation, the resulting homogeneous transformation matrices, A_i and T_N, contain redundant information. Maintaining extra terms requires additional memory and additional floating point operations. However, by properly customizing generic matrix operations (e.g., multiplication and inversion), the Denavit-Hartenberg system can be optimized so as to be usable for real time applications [ZHA88].

Another more serious drawback, or deficiency, of the Denavit-Hartenberg system results from the particular set of geometric parameters that are used in the model [EVE87]. When two consecutive joint axes are parallel, the second joint axis is placed at the intersection of its axis and the common normal of the previous axis. Because there are infinitely many common normals, the location of the axis coordinate system is arbitrary. If the consecutive axes are slightly misaligned, they may appear to intersect. The distance between the coordinate frames, however, can become arbitrarily large. That is, for slight misalignments, the corresponding D-H parameter can approach infinity. Several techniques have been developed to adjust for the shortcoming of the D-H system in the case of parallel axes. One such method is call the *S-model* and is discussed in section 5.

Among the other systems for specifying the kinematics of a manipulator are techniques for specifying the position and orientation of a body separately. Specification of the displacement is done simply with a vector of three position terms corresponding to x, y, z. Euler Angles (which include roll-pitch-yaw angles) is a system for specifying the

Figure 6. Kinematic Chain.

orientation. They are a set of consecutive angles of rotation about predefined axes [ASA86,CRA86]. The advantage of such a system is that it requires only three parameters to specify a rotation. Euler Angles (and roll-pitch-yaw angles) are discussed in section 5.

Quaternions are another mathematical object which can be used to represent rotations [FUN88, PER82]. Typically, a quaternion, q, is represented by the following:

$$q = s + ix + jy + kz,$$

where s, x, y, z $\in$ **R**, and i, j, k are orthogonal unit vectors. However, a quaternion can also be written with a trigonometric representation. From the trigonometric form, one can derive a general expression for the rotation of one vector into another using a quaternion as an operator. For a thorough comparison of quaternions and homogeneous transformations, as well as a derivation of quaternions as rotation operators, see [FUN88].

Each of the above mentioned methods creates a forward kinematic model for the manipulator. Each provides a way for representing the position and orientation of the end-effector of the manipulator with respect to its base as a function of the joint variables θ_i, i = 1, ..., N. If in the static description the coordinate frame of reference (as specified in this document by the parameter C_z) is displaced from the base by T_w, then the forward kinematic model must be adjusted accordingly. If T_w and the kinematic system both use homogeneous transformations or quaternions, then the forward kinematic model simply is pre-multiplied by T_w to account for the displacement.

For the purposes of the discussion in this document, we represent the kinematic rotation (orientation) by R_i and the position by P_i. For example, if the Denavit-Hartenberg system is used, R_i is the 3x3 rotation submatrix and P_i is the 3x1 position vector.

4.1.2. Manipulator Jacobian and Inverse Jacobian

In order to control the manipulator in Cartesian space using resolved rate control or force control, the Jacobian is needed to transform the rates or forces to joint coordinates. The Jacobian relates the differential changes in the joint coordinates to differential changes in the Cartesian coordinates. It is a function of the manipulator's position, or joint variables.

Inherent in the derivation of the Jacobian is the choice of frame of reference for the motions. That is, the motions can be specified with respect to the base of the manipulator, with respect to its end-effector, or with respect to another frame with a known transformation from the base or end-effector. In our discussion of the manipulator Jacobian, we consider each case for the choice of frame of reference, as given by the parameter Cz (Co).

The frame of reference for the Jacobian is specified by the parameter C_z. If C_z = (World, T_w, T_{ee}), then the commands and movements are measured in a frame which is related to the base of the manipulator by T_w. The offset from the base can be incorporated into the kinematic model by pre-multiplying the kinematic transformation (A_0 if the Denavit-Hartenberg system is used) by T_w. Similarly, the offset from the end-effector can be incorporated by post-multiplying the last kinematic transformation by T_{ee}. The offset transformations can be appended and treated as constant transformations because they do not vary with the joint variables. Creation of the Jacobian follows, as above, using the appended or modified kinematic transformations.

If C_z = (end-effector, _, T_{ee}), then the commands are sent from the Task Decomposition module with respect to the frame at T_{ee}. The specifications in this case would not contain absolute positions, but rather would contain commands of the form: *move in the direction of the z-axis of the tool tip*. Once again, the transformation T_{ee} can be appended to the last kinematic transformation. The Jacobian for this system would relate changes in the joint angles to changes in the end-effector's motion, as measured with respect to the end-effector.

The inverse of the Jacobian is needed to relate the instantaneous velocities in Cartesian space to those in joint space. For a non-redundant manipulator, the kinematic relationship is as follows:

$$\dot{\theta} = J^{-1}\dot{x},$$

where $\dot{x}$ is the six-dimensional Cartesian velocity, $\dot{\theta}$ is the $n \geq 6$ -dimensional joint velocity, and J^{-1} is the inverse of the Jacobian. This equation determines the necessary changes in the joint variables to achieve a desired change in Cartesian space.

The inverse Jacobian can also be used to relate the dynamic models in joint space to that in operational space [KHA87]. The manipulator equations of motion are given by:

$$\tau = A(\theta)\ddot{\theta} + b(\theta, \dot{\theta}) + g(\theta),$$

where τ is the vector of external or applied torques. A, b, and g represent the inertial coefficient matrix, the Coriolis and centrifugal forces, and the gravity forces, respectively. A similar equation can be written for the torques in terms of Cartesian quantities, which are in turn related to their joint space counterparts through the inverse Jacobian. For example, the joint space (A) and Cartesian space (A') inertial matrices are related by:

$$A'(x) = J^{-T}(\theta)A(\theta)J^{-1}(\theta),$$

where J^{-T} denotes the transpose of the inverse Jacobian. The joint space (g) and Cartesian space (g') gravity potential energy terms are related by:

$$g'(x) = J^{-T}(\theta)\, g(\theta).$$

In the case of a redundant manipulator, Khatib replaces the inverse of the Jacobian

transpose with the generalized inverse $\bar{J}$ given by:

$$\bar{J}(\theta) = A^{-1}(\theta)J^{T}(\theta)\,A'(\theta),$$

which minimizes the manipulator's instantaneous kinetic energy. The transformed dynamics values (A'(x), b'(x), g'(x)) are then used to control the manipulator in operational space. (For additional discussion of manipulator control see sec. 4.3 or [FIA88]; for additional discussion of manipulator redundancy and generalized inverses, see sec. 5.)

Many recursive methods exist for computing the manipulator Jacobian with respect to different frames of reference [DOT87, ORI84, PAU81,VUK79]. Most of them employ iterative techniques to determine the angular and linear velocities. In the following section we introduce several techniques and discuss their computational complexity. The reader is referred to the references for details of the iterative techniques and a comparison of the efficiency of each computational method.

Computational Considerations: The choice of techniques for generating the kinematic variables (forward kinematics and Jacobian) are not entirely independent; certain forward kinematic representations lend themselves to certain techniques for generating the Jacobian. Paul and Zhang developed a computationally efficient kinematics method based upon the Denavit-Hartenberg parameters for manipulators with a spherical wrist [PAU86]. The authors generate the overall forward kinematic model (T_6) in symbolic form. By substituting for common expressions, they are able to reduce the total number of arithmetic operations. Their resulting expressions for the T_6 matrix corresponding to the forward kinematic model of the PUMA 560 manipulator requires 6 sine/cosine pairs, 34 multiplications, and 17 additions [PAU86].

Paul and Zhang then present an efficient method for obtaining the Jacobian from the Denavit-Hartenberg forward kinematic model. They create the Jacobian with respect to the last link frame in symbolic form. Because the manipulator has a spherical wrist, the upper-right 3x3 submatrix of the Jacobian is identically zero. Thus showing that there is no coupling between position and orientation. The resulting evaluation of the Jacobian requires 6 sine/cosine pairs, 46 multiplications, and 19 additions.

Recursive techniques for creating the Jacobian tend to be more general. (They are not limited to 6 degrees of freedom and they do not assume that the manipulator has a spherical wrist.) As a result, they are more computationally intensive. As a rough estimate, computing the Jacobian for a 6 degree of freedom manipulator requires 100-150 multiplications, 60-80 additions, and 10-12 sine/cosine operations [ORI84]. The lower bounds for the number of computations results when one of the middle link's frame is used as the frame of reference. The number of operations and the method of choice depend to a large extent on the desired frames of reference.

Computing the inverse of the Jacobian is computationally intensive. A few measures can be taken, however, to reduce the number of floating point operations. If the manipulator has a spherical wrist, it will have a 3x3 submatrix which is identically zero. If J is written in

terms of submatrices as:

$$J = \begin{vmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{vmatrix}$$

then J^{-1} can be given by:

$$J^{-1} = \begin{vmatrix} J_{11}^{-1} & 0 \\ -J_{22}^{-1}J_{21}J_{11}^{-1} & J_{22}^{-1} \end{vmatrix}$$

Each submatrix inversion can be done symbolically or numerically. Paul and Zhang evaluate the inverse Jacobian with 72 multiplications and 31 additions [PAU86].

4.1.3. Force and Torque Transformations

Force and torque sensing can take place at the joints and/or the wrist of the manipulator, with wrist sensing being the most widely used. Force sensor readings $F = (f_x,f_y,f_z,m_x,m_y,m_z)$ from the manipulator end-effector are transformed to the base or tool tip frame (or any other frame specified by C_z) according to:

$$^{A}F = ({}^{A}_{B}J)^{T}\ {}^{B}F,$$

where A denotes the desired frame of reference and B denotes frame in which the readings were taken [CRA86]. Similarly, the Jacobian transpose maps forces acting at the end-effector into equivalent joint torques by:

$$\tau = J^{T} F.$$

4.2. Module Operation: Dynamics

In order to perform model-based manipulator control, Servo Task Decomposition requires a dynamic model of the manipulator. The dynamics terms, however, generally need not be updated at the servo rate. By updating the dynamic model every 10-16 milliseconds, World Modeling can supply Servo Task Decomposition with sufficiently accurate values.

Recall the symbolic equation for the dynamics of a manipulator:

$$\tau = A(\theta)\ddot{\theta} + b(\theta, \dot{\theta}) + g(\theta).$$

The above equation contains the basic dynamic model which the World Modeling module makes available for the Task Decomposition module. The Task Decomposition module computes the complete dynamic model, taking into account the servo algorithm, as well as any singularity or obstacle avoidance. The dynamic model, as computed by the World Modeling, is either in the form of the forward dynamic terms ($A(\theta)$, $b(\theta, \dot{\theta})$, $g(\theta)$), or it is the overall

torque required for the manipulator to make the specified move, (τ). To a large extent, the form of the dynamic model depends on the type of dynamics formulation used. Additionally, the forward dynamics terms may be transformed to Cartesian space for use in operational space control, as depicted by the dotted line function in figure 7.

While many different methods exist for computing the dynamics of a manipulator [BUR86, FAE86, FEA87, IZA86], the primary concern of them all is the large number of computations involved. Furthermore, because the dynamic equations for a manipulator with 6 (or more) degrees of freedom are highly complex, they are very difficult to compute by hand. The dynamics typically are computed in one of two ways. The first is to generate closed form expressions for the dynamic equations *off-line* using a symbolic code generation package. The model is then evaluated at execution time. The second is to use a recursive (iterative) formulation of either the Newton-Euler or the Lagrangian dynamic equations to compute the model at execution time. Recursive algorithms have been devised to compute the dynamics (forward and backward) in linear time [HOL80, WAL82]. Symbolic generation can achieve similar results through simplifications based upon manipulator geometries and the underlying structure of the dynamic equations [BUR86, IZA86].

Regardless of the method used, certain kinematic and dynamic data is required as input. The World Modeling module computes and maintains the information. For each link, the position and orientation transformations are given by R_i and P_i. In addition to the vector P_i, which locates the joint origins, a vector is needed to locate the center of mass of each link. The vector, r_i, is given below. Finally, the dynamic equations depend upon the inertial matrices, I_i, also shown below.

$$r_i = \begin{vmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{vmatrix} \qquad I_i = \begin{vmatrix} I_{ixx} & I_{ixy} & I_{ixz} \\ I_{iyx} & I_{iyy} & I_{iyz} \\ I_{izx} & I_{izy} & I_{izz} \end{vmatrix}$$

Computational Considerations: Highly efficient recursive algorithms for the inverse dynamics problem result from optimizing and customizing the Newton-Euler algorithm

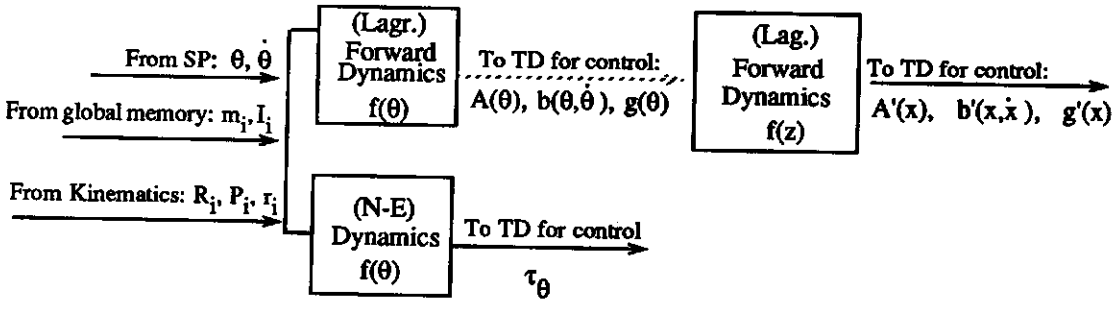

Figure 7. Manipulator Dynamics Computations.

[HOL80, KHO86]. The efficiency of the algorithm is further improved by choosing a middle link's frame as the frame of reference rather than the base frame. The resulting Newton-Euler method requires 852 multiplications and 738 additions to solve the inverse dynamics for a 6 degree of freedom manipulator.

Burdick presented an algorithm for symbolically generating efficient dynamic equations [BUR86]. The equations are based upon the Lagrangian dynamics formulation. The equations are generated and simplified *off-line* by a LISP-based program. The number of computations for a manipulator is greatly reduced by exploiting algebraic identities and the underlying structure (symmetries) of the dynamic equations. Further reductions can be made based upon the "regularity of common manipulator configurations." Specifically, simplifications can be made if consecutive joints in the manipulator have parallel or orthogonal joint axes, as is the case for most commercial manipulators. In addition, the "regular geometry" of the spherical wrist leads to further reductions in the complexity of the dynamic equations. The total computation of the inverse dynamics, for the 6 degree of freedom PUMA 560 manipulator, requires 401 multiplications and 254 additions.

Thus, symbolic equations have the advantage that they can be optimized so as to be more efficient than the recursive methods. Symbolic equations also make it possible to extract the forward dynamics terms ($A(\theta)$, $b(\theta, \dot{\theta})$, $g(\theta)$) from the calculations. Because these terms are explicitly needed to perform operational space control [KHA87], a symbolic preprocessing method may be preferred. However, a recursive method has the advantage that it does not require a symbolic processor (e.g. a LISP program or a symbolic mathematics package such as MACSYMA) and is easily coded.

4.3. Module Operation: Control

The Servo Level of Task Decomposition is responsible for controlling the manipulator so that it tracks a desired path with a desired velocity and acceleration, in a stable manner. Many control schemes are possible, and the reader is referred to [FIA88] for an overview. One such scheme [LUH85, SKO86]:

$$M(\theta)[\, K_p\,(\theta_d - \theta) + K_v\,(\dot{\theta}_d - \dot{\theta}) + \ddot{\theta}_d\,] + \tau_{cent}(\dot{\theta},\theta) + \tau_{gravity}(\theta) = \tau_{act}$$

requires that the World Model compute the dynamic model of the manipulator. Here, the matrix $M(\theta)$ is the inertia coefficient and $\tau_{cent}(\dot{\theta},\theta)$ is the vector of centrifugal and Coriolis compensation torques, both computed from the manipulator model. The manipulator model does not change significantly between servo cycles; World Modeling need not update it at the servo rate. Compensation for other types of disturbances, such as friction, can be included in the control provided a model of the disturbance exists. In the Cartesian domain the computed torque control algorithm requires the terms $A'(x)$, $b'(x,\dot{x})$, $g'(x)$.

In a feedforward control scheme, a model of the manipulator dynamics is used to decouple the actuators. World modeling provides these manipulator model terms. For example, a common feedforward control can be performed by [ASA83, FIA88]:

$$\tau_{model}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) + K_p(\theta_d - \theta) + K_v(\dot{\theta}_d - \dot{\theta}) + \ddot{\theta}_d = \tau_{act}.$$

In this case, all of the dynamic compensating terms are obtained as one torque vector, as shown in figure 8.

World Modeling also might provide avoidance torques to Task Decomposition. The avoidance torques are vectors of joint torques that are added to the control to achieve avoidance of some condition. One example of the use of this type of parameter is for obstacle avoidance [KHA87]. While global obstacle avoidance requires considerable planning at higher levels of the control hierarchy, local avoidance torques can be added to the control torques at the servo level. In Khatib's paper, a hypothetical force pushing away from an obstacle is given by

$$f^* = \begin{cases} \eta\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2}\frac{\partial \rho}{\partial x} & \text{if } \rho \le \rho_0 \\ 0 & \text{if } \rho > \rho_0 \end{cases},$$

where ρ is the distance to the obstacle, η and ρ_0 are constants. World Modeling would compute the hypothetical force (f^*) and convert it to a torque through the manipulator Jacobian by:

$$\tau_{avd} = J^T(\theta)A'(x)f^*.$$

Task Decomposition could then add τ_{avd} to the control torques to provide local obstacle avoidance.

5. Kinematics in Detail

In this section we provide additional details on forward kinematics; we discuss several methods in detail, including the Denavit-Hartenberg parameters and Euler Angles. In the following section we consider inverse kinematics for redundant manipulators. We concentrate our discussion on methods of computing the inverse of the manipulator's Jacobian which exploit the redundancy.

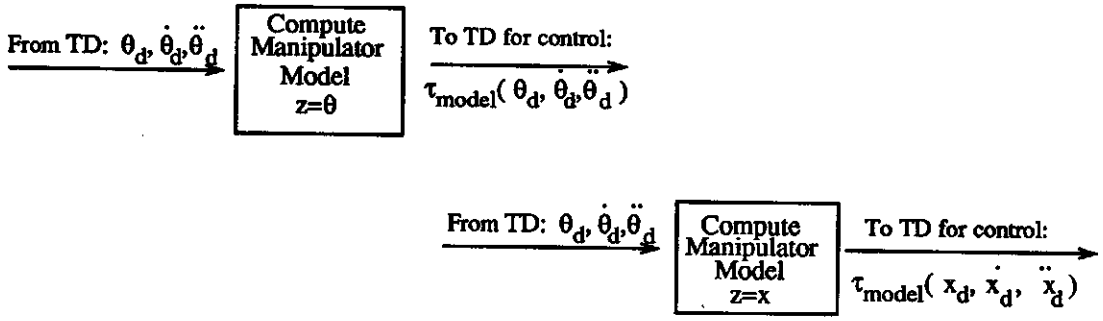

Figure 8. Computing the Manipulator Model for Control.

5.1. Forward Kinematics

The first step in the **Denavit-Hartenberg** systematic process is to assign a coordinate frame number to each link. We assign the number 0 to the base, although we do not consider it as one of the links of the manipulator. The numbers 1, ..., N are assigned sequentially to the rest of the links.

The coordinate frames are aligned so that the z-axis for link *i* is the axis of rotation of the i^{th} Denavit-Hartenberg frame. The following series of transformations:

$$\text{Rotation}(z_{i-1}, \theta_i)\text{Translation}(a_i, 0, d_i)\ \text{Rotation}(x_i, \alpha_i)$$

describes the position and orientation of the i^{th} link in the $i\text{-}1^{th}$ frame. The variables θ_i, a_i, d_i, and α_i are called the Denavit-Hartenberg parameters of the system. The first parameter, θ_i, is the joint variable for the i^{th} revolute joint; it defines the angle of rotation of the joint about the z_{i-1} axis from its home position. Rotating a joint an angle θ_i about its z_{i-1} axis, causes the x_{i-1} axis of its frame to align with the x_i axis of the next link's frame. By translating the $i\text{-}1^{th}$ frame amount a_i along the x_i axis (or equivalently, along the x_{i-1} axis) and an amount d_i along the z_{i-1} axis, its origin rests in the same location as the origin of the i^{th} frame. (In the case of prismatic joints, d_i is the joint variable.) Finally, after rotating the frame an angle α_i about the x_i axis, the z-axes of the two frames coincide and the transformation from one frame to the next is complete.

The homogeneous transformation matrix created from the rotations and translations defined by the Denavit-Hartenberg parameters is called an A matrix of the system. Specifically, A_i denotes the homogeneous transformation from the $i\text{-}1^{th}$ frame the i^{th} frame. It is composed of a 3x3 rotational submatrix and a 3x1 positional vector, as separated by the dotted line in the following expression..

$$A_i = \left[\begin{array}{cccc} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{array}\right]$$

$c\theta_i$ denotes the cosine of θ_i, $s\theta_i$ denotes the sine of θ_i, etc.

By multiplying successive A matrices, $A_1 \cdot A_2 \cdot \ldots \cdot A_N$, where N is the number of degrees of freedom of the manipulator, one obtains the T_N matrix. The T_N matrix represents the position and orientation of the end-effector with respect to the base frame, as a function of

the joint variables.

Another system, similar to the Denavit-Hartenberg system, is the **Whitney-Lozinski model** [STO87]. (The S-model results directly from the Whitney-Lozinski model.) This model uses six parameters, rather than four, to transform between coordinate frames according to:

$$\text{Rotation}(y,\theta_i)\text{Translation}(0,y_i,z_i)\text{Rotation}(z,\phi_i)\text{Rotation}(y,\Omega_i)\text{Rotation}(x,\Psi_i).$$

The homogeneous transformation matrix created from the rotations and translations defined by the Whitney-Lozinski parameters is called an W matrix of the system. By multiplying successive W matrices, $W_1 \cdot W_2 \cdot \ldots \cdot W_N$, where N is the number of degrees of freedom of the manipulator, one obtains the T_N matrix. The resultant T_N matrix has the same properties as the T_N matrix formed with the Denavit-Hartenberg parameters. The Whitney-Lozinski model introduces a greater degree of flexibility in assigning the locations of the link coordinates. However, a drawback of this system is that it requires 6N parameters to completely specify the geometry of the manipulator.

Roll-pitch-yaw angles and **Euler Angles**, use consecutive angles of rotation about predefined axes [ASA86,CRA86].

The series of rotations for roll-pitch-yaw are:

$$\text{Rotation}(z,\alpha)\ \text{Rotation}(y,\beta)\ \text{Rotation}(x,\gamma)$$

where the order of rotations is essential to the technique. Each rotation is performed about the specified axis of the initial frame.

Euler Angles differ from roll-pitch-yaw angles in that each successive rotation is made about the moving frame's axes rather than those of the initial frame. There are two commonly used series of rotations for Euler Angles. The first series, appropriately called Z-Y-X Euler Angles, is:

$$\text{Rotation}(z,\alpha)\ \text{Rotation}(y',\beta)\ \text{Rotation}(x'',\gamma),$$

where the $'$ and $''$ denote that the updated, or rotated, frames are used as the new frames of reference. The second series, called Z-Y-Z Euler Angles, is:

$$\text{Rotation}(z,\alpha)\ \text{Rotation}(y',\beta)\ \text{Rotation}(z'',\gamma),$$

where, once again, the updated frames are used as the frames of reference.

Another possible description of the orientation of a frame, or rigid body, is the **equivalent angle-axis** representation [CRA86]. A general orientation can be written as:

$$\text{Rotation}(k,\theta),$$

where θ represents the angle of rotation, in the right hand sense, about the unit axis k.

A similar representation can be created using **screw coordinates** [ROT84]. Screw coordinates employ six parameters to define the displacement (and orientation) of a rigid body. According to Chasles' Theorem, any displacement can be effected by a single rotation about a unique axis combined with a unique translation parallel to that axis. Screw

coordinates realize Chasles' Theorem which states that four parameters define the screw axis in space, one parameter specifies the rotation about the screw axis and one parameter specifies the translation along the screw axis.

5.2. Inverse Kinematics for Redundant Manipulators

Because of the benefits of kinematic redundancy (e.g. obstacle avoidance, singularity avoidance), many manipulators are configured with more than six degrees of freedom. Introducing redundancy complicates the control algorithm. At what level and how the redundancy should be resolved is still an open problem.

When the manipulator is redundant, the extra degrees of freedom can be used to advantage in defining the endpoint trajectory. For redundant manipulators, the relationship

$$\dot{x} = J\,\dot{\theta}$$

still holds. However, J in this case is non-square and has no inverse; the standard inverse must be replaced by a generalized inverse.

The pseudoinverse, given by $J^{+} = J^{T}(J \cdot J^{T})^{-1}$, is most often used in robotics applications [BAI84, CHAN86, KLE83, NAK86]. It provides the minimum norm solution, $\dot{\theta}_0$, for the kinematic equation $\dot{\theta} = J^{+}\,\dot{x}$. That is, for any other solution, $\dot{\theta}_1$, found using another method, the following is true: $\| \dot{\theta}_1 \| > \| \dot{\theta}_0 \|$, where $\| \ \|$ denotes the Euclidian norm. The pseudoinverse has an additional advantage over J^{-1}. While J^{-1} is not defined at singularities where the Jacobian loses rank, J^{+} provides an approximate solution in the sense of the minimum norm of $\| \dot{\theta} \|$ [KLE83]. These properties make it attractive for robot kinematics.

When computing the pseudoinverse ($J^{T}[JJ^{T}]^{-1}$), care should again be taken to reduce the number of floating point operations. First, because the product $J \cdot J^{T}$ is symmetric, only the elements along the diagonal and above (below) need to be explicitly computed. Second, $J \cdot J^{T}$ is not only symmetric, but also positive definite. Its special form should be exploited when performing the matrix inversion to reduce the ordinarily N^3 operation [JEN75].

While the pseudoinverse provides an adequate means for resolving kinematic redundancy, many researchers have explored methods to improve the pseudoinverse [BAI84, CHAN86, KLE83, YOS84]. In many of these methods, the inverse solution takes the form

$$\dot{\theta} = J^{+}\dot{x} + (I - J^{+}J)\,\phi\,,$$

where J^{+} is the pseudoinverse equal to $J^{t}(J\,J^{t})^{-1}$ and ϕ is an arbitrary joint vector. The operator $(I - J^{+}J)$ maps ϕ into the null space of the Jacobian. The null space vectors

correspond to self-motion of the linkage which does not affect the end-effector motion. In this way it does not change the validity of the solution created by the pseudoinverse. It augments the solution so as to be more robust with respect to the chosen criterion (ϕ). The vector ϕ can be used to minimize an optimization criterion or generate avoidance torques that operate only in the null space of the manipulator end-effector motion. (For a more thorough discussion, see [FIA88].)

The singularity robust inverse [NAK86] presents another alternative for J^+. It replaces the pseudoinverse and exploits the manipulator's kinematic redundancy to achieve singularity avoidance. The singularity robust inverse is based upon the premise that the pseudoinverse solution is problematic in the neighborhood of a singularity. In an effort to converge to an exact solution, the pseudoinverse may generate an infeasible one. That is, it may generate a solution for which one, or more of the joint increments is so large that it cannot be physically realized. To circumvent the problem of excessively large joint angles, Nakamura and Hanafusa propose the singularity robust inverse.

The singularity robust inverse, J^*,is defined as:

$$J^* = J^T(JJ^T + \lambda I)^{-1},$$

where λ is the scale factor between the exactness and the feasibility of the solution. By increasing λ in the neighborhood of a singularity, one creates a feasible solution and avoids excessively large changes in the joint variables [NAK86].

If the kinematic redundancy is to be resolved dynamically, most of the above mentioned kinematic methods are not applicable. However, as mentioned above, the concept of a nullspace vector is applicable for generating avoidance torques [FIA88].

6. Conclusions

This document has given a description of Level 1 World Modeling for a hierarchical manipulator control system. Level 1 includes World Modeling support for Servo Task Decomposition and support for Level 1 Sensory Processing. The function and interfaces of the World Modeling module have been described. While the specific function and computations of the support processes depend on the sensors, devices and control algorithms implemented in the Task Decomposition module, typical computations for a manipulator have been discussed.

7. References

[ALB87] Albus, J.S., McCain, H.G., Lumia, R., *NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM)*, NASA Document SS-GSFC-0027, June 18, 1987.

[ASA83] Asada, H., Kanade, T., Takeyama, I., "Control of a Direct-Drive Arm," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 105, No. 3, 1983.

[ASA86] Asada, H., Slotine, J.J., *Robot Analysis and Control*, John Wiley and Sons, New York, 1986.

[BAI84] Baillieul, J., Hollerbach, J., Brockett, R., "Programming and Control of Kinematically Redundant Manipulators," *Proceedings 23rd Conference on Decision and Control*: 768-774, December, 1984.

[BUR86] Burdick, J.W., "An Algorithm for Generation of Efficient Manipulator Dynamic Equations," *Proceedings 1986 IEEE International Conference on Robotics and Automation*, Vol. 1: 212-218, April, 1986.

[CHA89] Chaconas, K., Nashman, M. "Visual Perception Processing in a Hierarchical Control System: Level 1," NIST Technical Note 1260, NIST, Gaithersburg, MD, June 1989.

[CHAN86] Chang, P.H., "A Closed-form Solution for the Control of Manipulators with Kinematic Redundancy," *IEEE International Conference on Robotics and Automation*, Vol. 1: 9-14, April, 1986.

[CRA86] Craig, J.J., *Introduction to Robotics: Mechanics and Control*, Addison Wesley Publishing, Massachusettes, 1986.

[DEN55] Denavit, J., Hartenberg, R.S., "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *ASME Journal of Applied Mechanics*, Vol. 2, 1955, pp. 215-221.

[DOT87] Doty, K.L., "Tabulation of the Symbolic Midframe Jacobian of a Robot Manipulator," *The International Journal of Robotics Research*, Vol. 6, No. 4:85-97, Winter, 1987.

[EVE87] Everett, L.J., Driels, M., Mooring, B.W., "Kinematic Modelling for Robot Calibration," *Proceedings 1987 IEEE International Conference on Robotics and Automation*, Vol. 1: 183-189, April, 1987.

[FAE86] Faessler, H. "Computer-Assisted Generation of Dynamical Equations for Multibody Systems," *The International Journal of Robotics Research*, Vol. 5, No. 3, Fall, 1986.

[FIA88] Fiala, J., "Manipulator Servo Level Task Decomposition," NIST Technical Note 1255, NIST, Gaithersburg, MD, October 1988.

[FEA87] Featherstone, R., *Robot Dynamics Algorithms*, Kluwer Academic Publishers, Massachusettes, 1987.

[FUN88] Funda, J., Paul, R. P., "A Comparison of Transforms and Quaternions in Robotics," *Proceedings 1988 IEEE International Conference on Robotics and Automation*, Vol. 2: 886-891, April, 1988.

[GU88] Gu, Y.L., "Analysis of Orientation Representations by Lie Algebra in Robotics," *Proceedings 1988 IEEE International Conference on Robotics and Automation*, Vol. 2: 874-879, April, 1988.

[HOL80] Hollerbach, J.M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-10, No.11, November, 1980.

[IZA86] Izaguirre, A., Paul, R., "Automatic Generation of the Dynamic Equations of the Robot Manipulators Using a LISP Program," *Proceedings 1986 IEEE International*

Conference on Robotics and Automation, Vol. 1: 220-226, April, 1986.

[JEN75] Jensen, J.A., *Methods of Computation: The Linear Space Approach to Numerical Analysis*, Scott, Foresman and Company, Illinois, 1975.

[KAN83] Kane, T.R., Likins, P.W., and Levinson, D.A., *Spacecraft Dynamics*, McGraw-Hill Book Company, New York, 1983.

[KHA87] Khatib, O., "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 1:43-53, February, 1987.

[KHO86] Khosla, P.K., *Real-Time Control and Identification of Direct-Drive Manipulators*, Ph.D. Thesis, Carnegie Mellon University, August, 1986.

[KLE83] Klein, C.A., Huang C-H, "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 3:245-250, March/April, 1983.

[LUH85] Luh, J. Y. S., "Design of Control Systems for Industrial Robots," in *Handbook of Industrial Robotics*, Nof, S. Y., ed., John Wiley & Sons, New York, 1985.

[NAK86] Nakamura, Y., Hanafusa, H., "Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control," *Journal of Dynamic Systems, Measurement, and Control*, Vol.108:163-171, September 1986.

[ORI84] Orin, D.E., Schrader, W.W., "Efficient Jacobian Determination for Robot Manipulators," *Robotics Research: The First International Symposium*, MIT Press, 1984.

[PAU81] Paul, R.P., *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, Massachusettes, 1981.

[PAU86] Paul, R.P., Zhang, H., "Computationally Efficient Kinematics for Manipulators with Spherical Wrists Based on the Homogeneous Transformation Representation," *The International Journal of Robotics Research*, Vol. 5, No. 2:32-44, Summer, 1986.

[PER82] Pervin, E., Webb, J., "Quaternions in Computer Vision and Robotics," Carnegie Mellon Department of Computer Science technical report #CMU-CS-82-150, 1982.

[ROT84] Roth, B., "Screws, Motors, and Wrenches that Cannot Be Bought in a Hardware Store," *Robotics Research: The First International Symposium*, MIT Press, 1984.

[SKO86] Skowronski, J. M., *Control Dynamics of Robotic Manipulators*, Academic Press, Orlando, 1986.

[STO87] Stone, H., *Kinematic Modeling, Identification, and Control of Robotic Manipulators*, Kluwer Academic Publishers, Massachusettes, 1987.

[VUK79] Vukobratovic, M., Potkonjak, V., "Contribution of the Forming of Computer Methods for Automatic Modeling of Spatial Mechanisms Motions," *Mechanism and Machine Theory*, Vol. 14:179-200, 1979.

[WAL82] Walker, M.W., Orin, D.E., "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *Journal of Dynamic Systems, Measurements, and Control*, Vol. 104:205-211, September 1982.

[YOS84] Yoshikawa, T., "Analysis and Control of Robot Manipulators with Redundancy," *Robotics Research: The 1st International Symposium*: 735-747, MIT Press, 1984.

[ZHA88] Zhang, Y., Paul, R.P., "Robot Manipulator Control and Cost," Document distributed to NIST from University of Pennsylvania , 1988.

☆U.S.GOVERNMENT PRINTING OFFICE:1989-261-913/00103

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** (See instructions)	1. PUBLICATION OR REPORT NO. NIST/TN-1258	2. Performing Organ. Report No.	3. Publication Date December 1989

4. TITLE AND SUBTITLE

Manipulator Servo Level World Modeling

5. AUTHOR(S)

Laura Kelmar

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
(formerly NATIONAL BUREAU OF STANDARDS)
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered
Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

S/A

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This document describes the interfaces and functions of the World Modeling module at the lowest level of a hierarchical manipulator control system. The World Modeling modules maintain an internal model of the world by continuously updating the model based upon sensory input. At the lowest level of the control system, the Level 1 World Modeling module interfaces to and supports the Data Acquisition Sensory Processing module and the Servo Task Decomposition module. This document contains detailed descriptions of the interfaces between the Level 1 World Modeling module and the rest of the control hierarchy. This document also discusses the function of the support processes within the module. It does not attempt to provide an exhaustive list of the computations and models required by World Modeling in support of all possible devices and sensors in a manipulator control system. Rather, it aims to elucidate the classes of support processes for both Sensory Processing and Task Decomposition, considering control of a manipulator (the most complex device) as an example. The reader should be familiar with ICG Document #001, NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM).

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

control system architecture; hierarchical control; manipulator dynamics; manipulator kinematics; robotics; world model

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES
31

15. Price

USCOMM-DC 6043-P80