Table of Contents

1. Introduction

This document describes the interfaces, structure, and function of a World Modeling module at the second 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 modules 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.

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 perception processes provide sensory feedback from devices such as cameras, range sensors and tactile array sensors. The manipulator processes plan and execute manipulator trajectories. Note that while the two branches decompose tasks independently within each branch, communication between processes, both within a branch and across branches, occurs via the global data system.

The second level of the Task Decomposition hierarchy is called the Primitive Level (Prim). It generates dynamic motion and force commands from a static description of the desired behavior of a device. Prim creates the time sequence of *attractor sets* needed to produce a *dynamic trajectory* and sends these as commands to the Servo level of the Task Decomposition hierarchy [WAV88]. The Prim World Modeling support module uses sensory input to provide Prim Task Decomposition with the most current models of the manipulator and its world.

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 control actuators 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, as in figure 3. The scope of this document is limited to the discussion of the World Modeling module at the Prim Level of a manipulator control system, as highlighted in figure 3. Throughout the document, Prim refers to Manipulator Prim Task Decomposition and Prim World Modeling refers to Manipulator Prim World Modeling. The word *module* refers to the processes and associated data contained within a single box in figure 3.

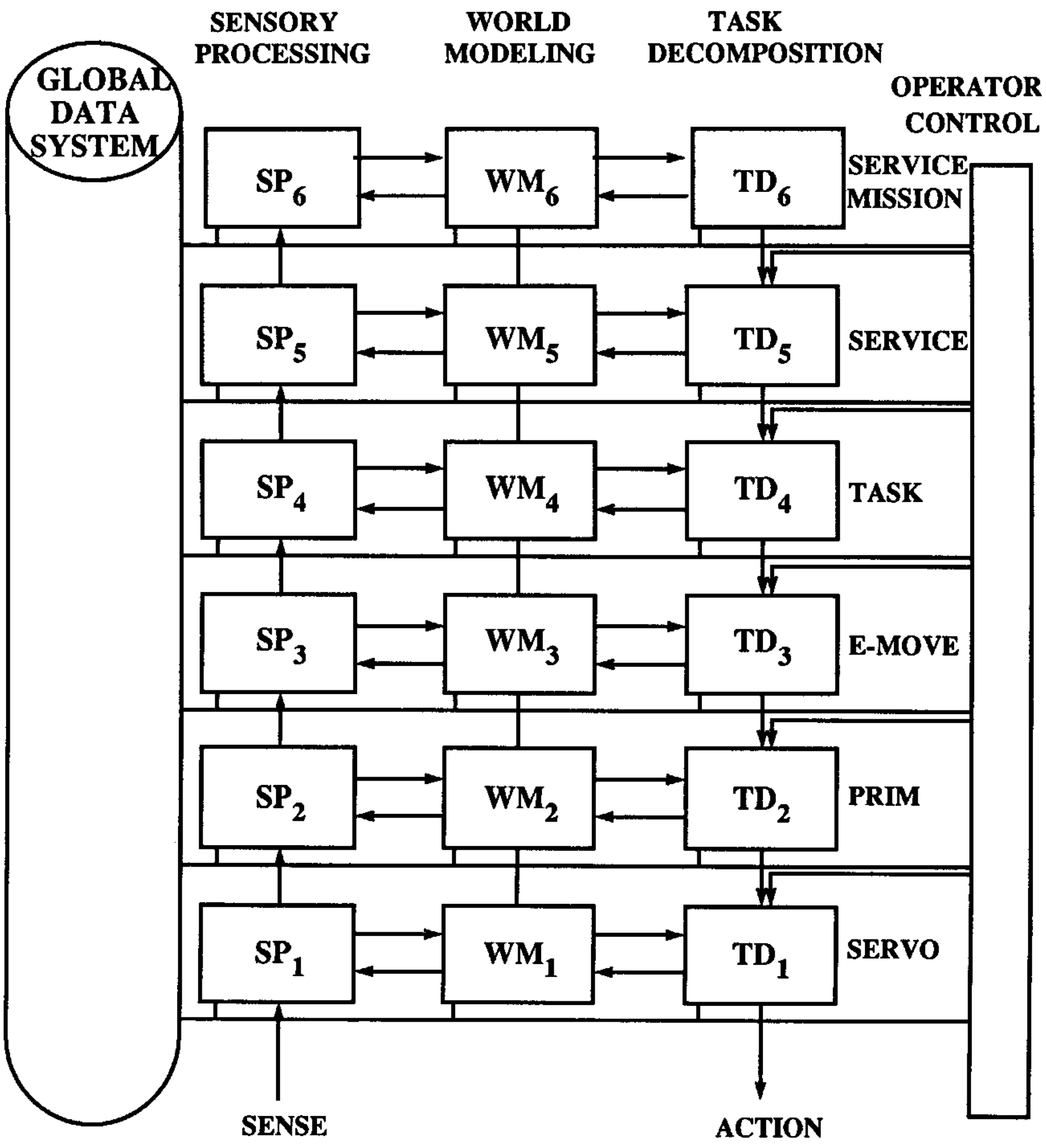

Figure 1. Hierarchical Control System Architecture.

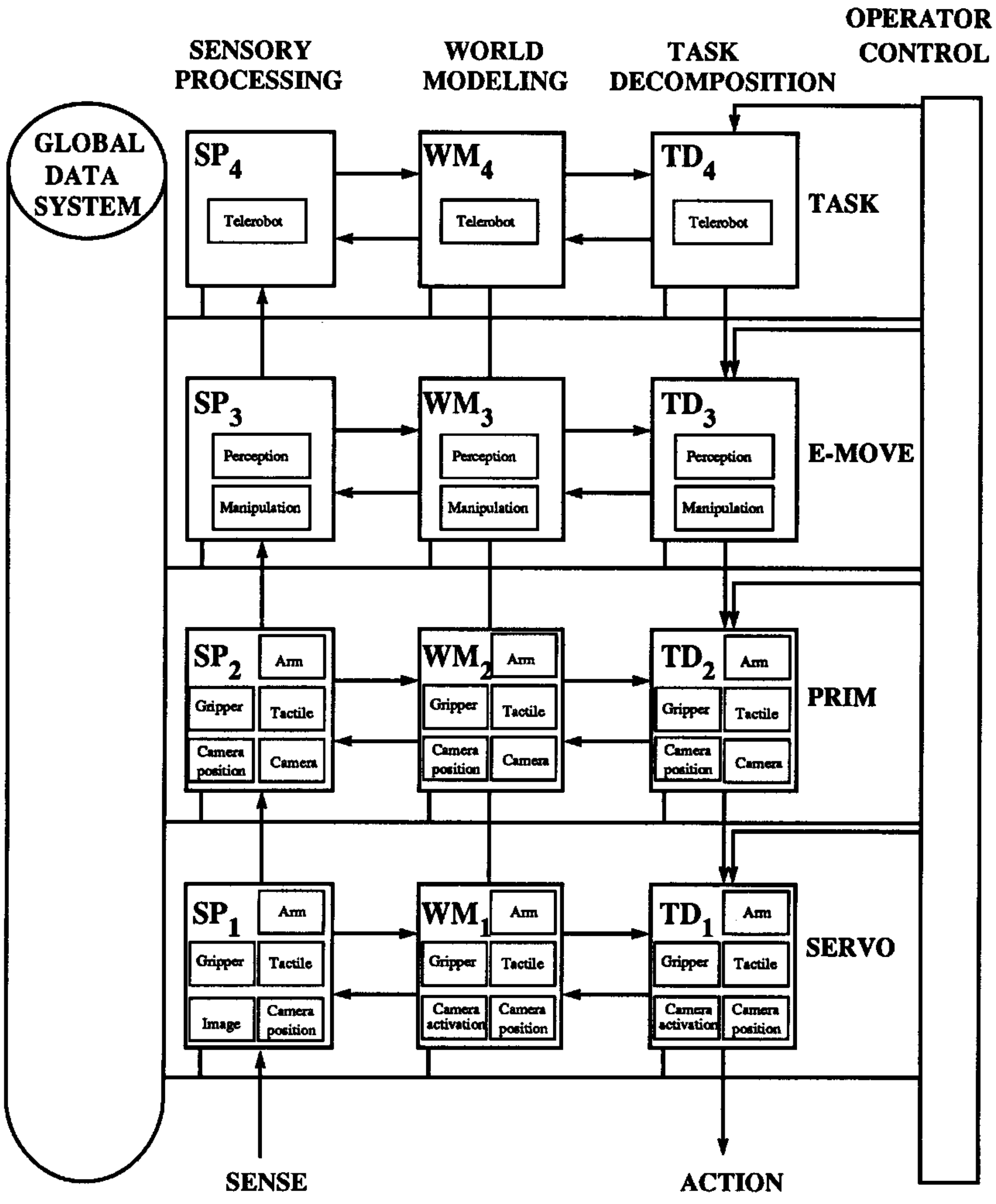

Figure 2. Modules for Each Sensor and Each Actuated Device.

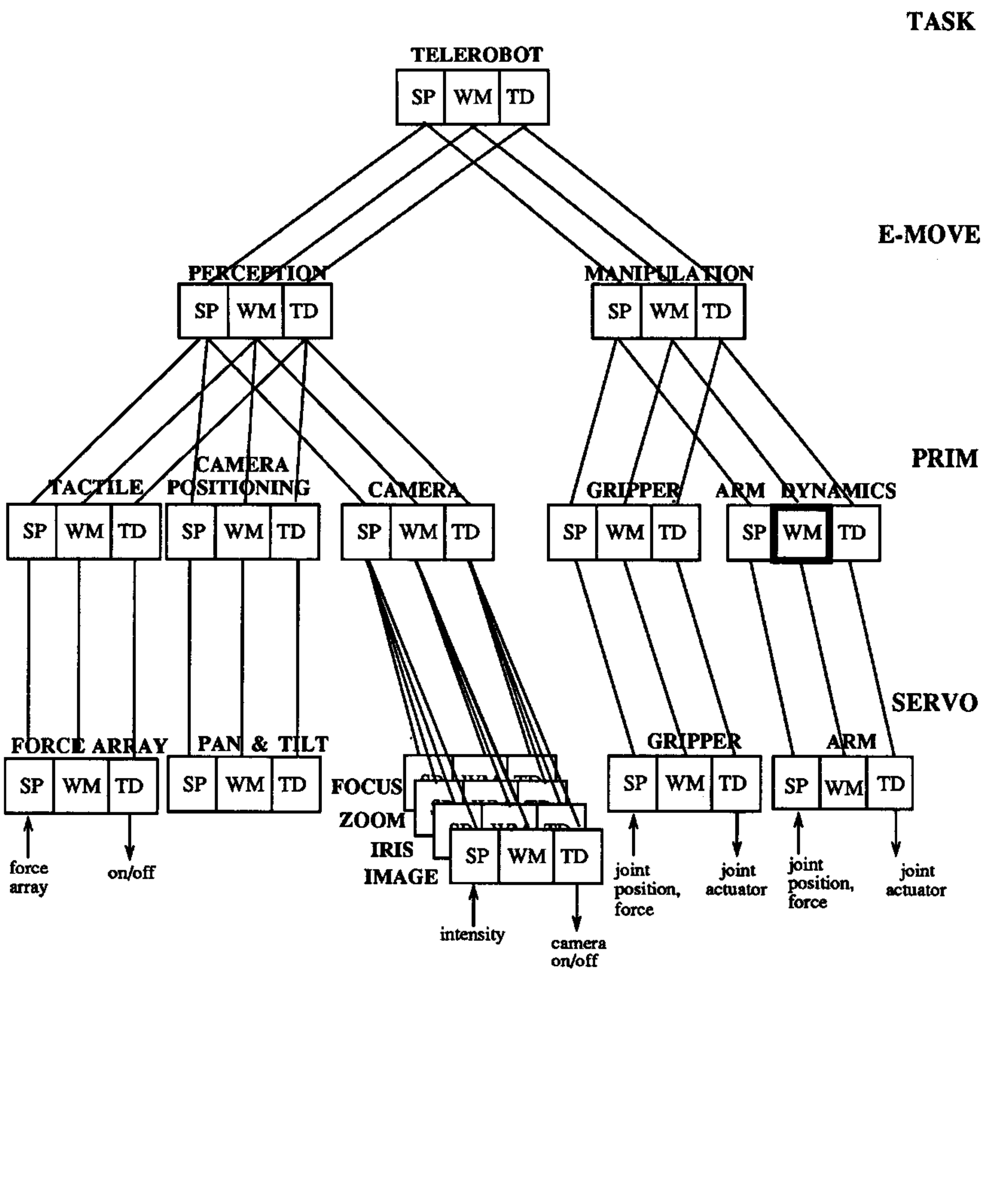

Figure 3. Perception and Manipulation Branches.

The interfaces to the Prim World Modeling support module are many: the Prim Task Decomposition module, the World Modeling support modules above and below it in the hierarchy (E-move and Servo Levels), and the Level 2 Sensory Processing module. The interface to the Prim Task Decomposition module is further broken down into interfaces to the Job Assignment module, the Planning module and the Executor module, as explained in [WAV88].

Section 2 of this document defines the interfaces between World Modeling and the rest of the control hierarchy. Section 3 discusses the operation of the different types of processes. Section 4 discusses the classes of algorithms performed by World Modeling in support of both Sensory Processing and Task Decomposition for a manipulator and its sensors. Many of the algorithms require a variety of object data. This document does not address the general topic of object model representations; it only discusses object information as it pertains to Manipulator Primitive Level algorithms.

2. Module Interfaces

The World Modeling support module interfaces to the submodules of the Prim Task Decomposition module: Job Assignment, Planning, and Execution. In this section we detail the Prim World Modeling interfaces. The information crossing the interfaces at any time will be a subset of the interface information defined below; it will depend on the Prim algorithm being performed at the time. The goal of the following sections is characterize the algorithm information passed to the support module and the corresponding modeling information which is returned. Figure 4 shows the information flow between Prim Task Decomposition and the World Modeling support module.

In order to perform its functions in support of Task Decomposition, the World Modeling support module also interfaces to several other modules in the control hierarchy. The interfaces between the support module and the rest of the control system are discussed in the following sections.

2.1. World Modeling to Prim Task Decomposition Interface

This section describes the interface between the World Modeling module and the Task Decomposition module at the Prim 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 computations in section 4.

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

The Prim Job Assignment module functions without World Modeling support. At this time it appears that all of the information needed by the module to perform its functions are provided by the Operator Control or the next higher Task Decomposition module (E-move Execution) [WAV88].

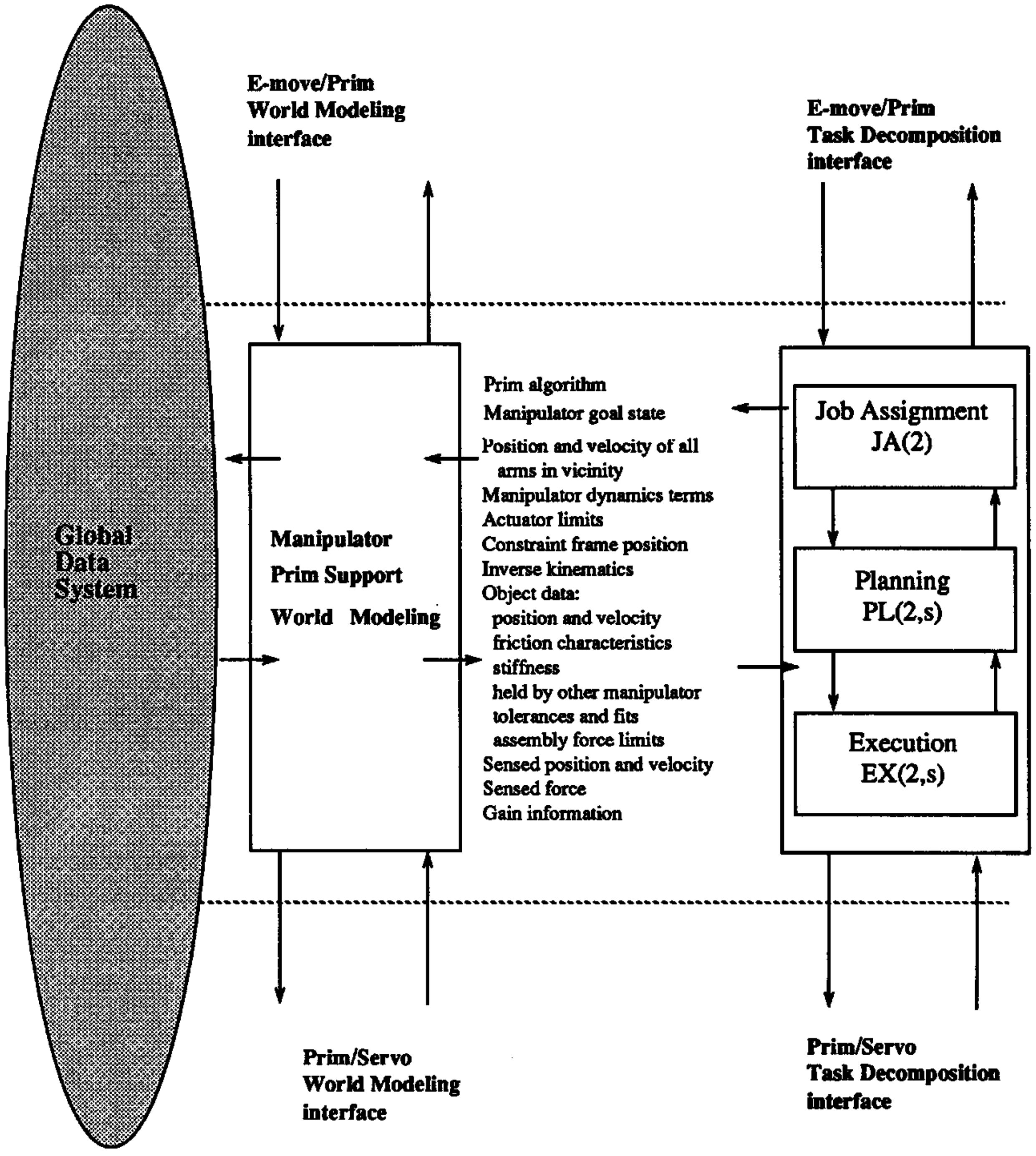

Figure 4. World Modeling to Task Decomposition Interface.

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

The Prim Planning module generates a plan for a dynamic trajectory for a manipulator. The plans may give explicit velocity, and acceleration profiles as a function of time. Alternatively, the plans may specify the shape or characteristics of the trajectory, without explicitly defining the path. Sensory interactive trajectories, such as vision servoed trajectories, are examples of trajectories which cannot be planned explicitly before execution. In order to plan trajectories, the Prim Planner requires considerable support from World Modeling. A list of the data maintained by World Modeling follows. Detailed descriptions of sample algorithms executed by World Modeling are discussed in section 4.

- **Forward and Inverse Kinematics** - When planning joint space trajectories between Cartesian space goal points, the Prim Planner relies on the inverse kinematic model of the manipulator to provide joint angles and velocities. The Cartesian Manipulator Goal State, as specified by Prim, simply may be the desired Cartesian end-effector pose (velocity), or it may include additional manipulator configuration parameters. For example, if the manipulator is redundant, it may be desirable to specify the configuration of the manipulator's elbow, as well as that of the end-effector. The Manipulator Goal State should allow for specification of any or all points of the manipulator.

  Coordinated Manipulators - When two manipulators cooperate to perform a single task, the desired pose and velocity of the manipulators must be coordinated. Computation of the kinematic models for coordinated manipulators may require the specification of additional, or alternate, configuration parameters. For example, consider the case of two manipulators carrying a rigid object such as a block. Coordination of the manipulators could be accomplished by having a planner compute the path of the center of the block. The sequence of desired Cartesian poses of each of the manipulators would be defined by the path of the center of the block and the transform from the center of the block to each manipulator's grasp point.

- **Current (Actual) Arm Position and Velocity** - The Prim Planner bases its future movements on the current position and velocity of the manipulator.

- **Actuator Limits** - Prim requires the torque and acceleration limits, which are a function of the manipulator configuration (joint angles). Also, World Modeling must provide the joint position limits for the manipulator.

- **Consideration of Arm and Payload Dynamics** - The manipulator dynamics must be considered in order for the full capabilities of the manipulator to be available and to prevent exceeding actuator limitations. The Prim Planner also must consider the mass, and distribution of mass, of any object which the manipulator will carry.

- **Gain Information** - The Prim Planner requires gain information for various motions. This information may be constant and stored in the global data system, or may be adjusted based upon experimental or run-time results.

- **Object Data** - The World Modeling support module must be able to provide Prim with a variety of object data. Object data may be available in the database or may be the result of extrapolation or prediction by World Modeling. For example, the position of an object may be constant or may be predictable based upon the velocity profile of the object. The exact nature of the data supplied by the World Modeling support module depends on the algorithms implemented, as discussed in section 4.5. The interface between Prim and its World Modeling support module must be rich enough to support Prim in planning and executing trajectory generation algorithms. It should include the following object data:

- **Position and/or Velocity** - The Prim Planner must know the location of the object(s) of interest. If the object is moving, World Modeling must supply the velocity, as well as the position, of the object. Similarly, if there are additional manipulators in the workspace, World Modeling must maintain maps which demarcate the occupied volumes. If an object is held by another manipulator, World Modeling must update the object's position with movements of the second manipulator.

- **Mass and Distribution of Mass** - The Prim Planner must consider the mass and distribution of mass of objects when planning dynamic trajectories.

- **Friction Characteristics** - The Prim Planner uses information about the friction characteristics between two parts when planning alignment and assembly tasks.

- **Stiffness** - In order to avoid damage and instability, the Prim Planner should consider the stiffness of an object to be manipulated when choosing the values for the manipulator stiffness.

- **Tolerances and Mechanical Constraints** - The Prim Planner uses knowledge of the tolerances and kinematic constraints between two objects when planning and executing assembly tasks. These characteristics should be exploited to reduce the need for extreme positional accuracy.

2.1.3. World Modeling to/from Task Decomposition Execution (EX)

World Modeling supplies the Execution module with various sensed values. The exact sensory information passed from World Modeling to Prim Execution depends on both the particular algorithm being executed and the sensors available in the system. Typically, sensory interactive algorithms require that World Modeling provide continuously updated readings. For example, force and torque readings provide important feedback when the manipulator performs contact tasks, such as inserting a peg in a hole. The Execution module may also require World Modeling to evaluate the legitimacy of a goal joint state vector before passing it to Servo for action.

We discuss some example manipulator sensory interactive control algorithms in section 4.5.3. The data used by Prim Execution in the algorithms enable manipulation when a priori information is not sufficiently recent or precise or to complete the task. Sensory interactive control algorithms are also used when manipulator accuracy must be verified via external sensors during task execution. The interactive sensor data includes:

- **Input Device Information** - For teleoperated control, the manipulator trajectory is guided by an input device, such as a joystick. The values from the input sensor, such as desired manipulator joint velocities, are transformed (if necessary) and made available by World Modeling.

- **Manipulator Position and Velocity** - The Execution module requires the position and velocity of the manipulator as feedback. The values may be in joint or Cartesian space, depending on the value of the coordinate system specifier, C_Z. See [WAV88] for details on coordinate system specification.

- **Object or Feature Position (relative displacement)** - Often the precise location of an object, as measured with respect to the world or to the manipulator's end-effector, cannot be known with sufficient accuracy prior to execution of the task. For such situations, sensory feedback during task execution, via visual or tactile cues, can enable successful completion of the task.

- **Force and Torque Values** - Feedback from manipulator force and torque sensors, can be used to detect a manipulator's contact with the environment. The reading, together with knowledge of how to make corrective motions, also can improve the performance of manipulators in assembly operations.

- **Obstacle Displacements from Proximity Sensor Readings** - Proximity sensors on the manipulator end-effector and/or joints can detect obstacles in the manipulator path. Prim can re-plan corrective motions to avoid damage to the manipulator or the environment.

2.2. World Modeling to Sensory Processing Interface

Each World Modeling module interfaces to Sensory Processing. The Manipulator Prim World Modeling module interfaces to the Manipulator Level 2 Sensory Processing module, as shown in figure 5. Manipulator sensor data includes readings from manipulator sensors, such as joint encoders, force/torque sensors, and proximity sensors. At this time, it appears that only one level of sensory processing is necessary for manipulator sensor data. The readings are filtered by the Level 1 Sensory Processing module and stored in the global data system. Hence, no processing is done by the Manipulator Level 2 Sensory Processing module and no information crosses the interface to the Manipulator Prim World Modeling module. The hierarchical control system model allows for the interface in the event that future algorithms require the connection.

Prim Task Decomposition makes extensive use of the sensor readings. For example, the Prim Planner requires current joint angle readings when planning a path. Prim Task Decomposition also uses data from other non-manipulator sensors, such as cameras. The Prim World Modeling support module interfaces to processed sensory data via the global data system. The coupling of the Task Decomposition and Sensory Processing hierarchies is especially significant when the manipulator performs a sensory interactive algorithm. For example, in *Camera Space Manipulation* [SKA87], controlled motions of the manipulator are tightly coupled with feedback in the form of processed images, from the camera Sensory Processing hierarchy. The coupling is effected by the Prim World Modeling support module. The details of this algorithm are discussed in section 4.5.3.

2.3. World Modeling Hierarchy Interfaces

The manipulator World Modeling support modules interface hierarchically with one another. (Servo Manipulator World Modeling support interfaces to Prim Manipulator World Modeling support, and so on up the hierarchy.) The support modules may function

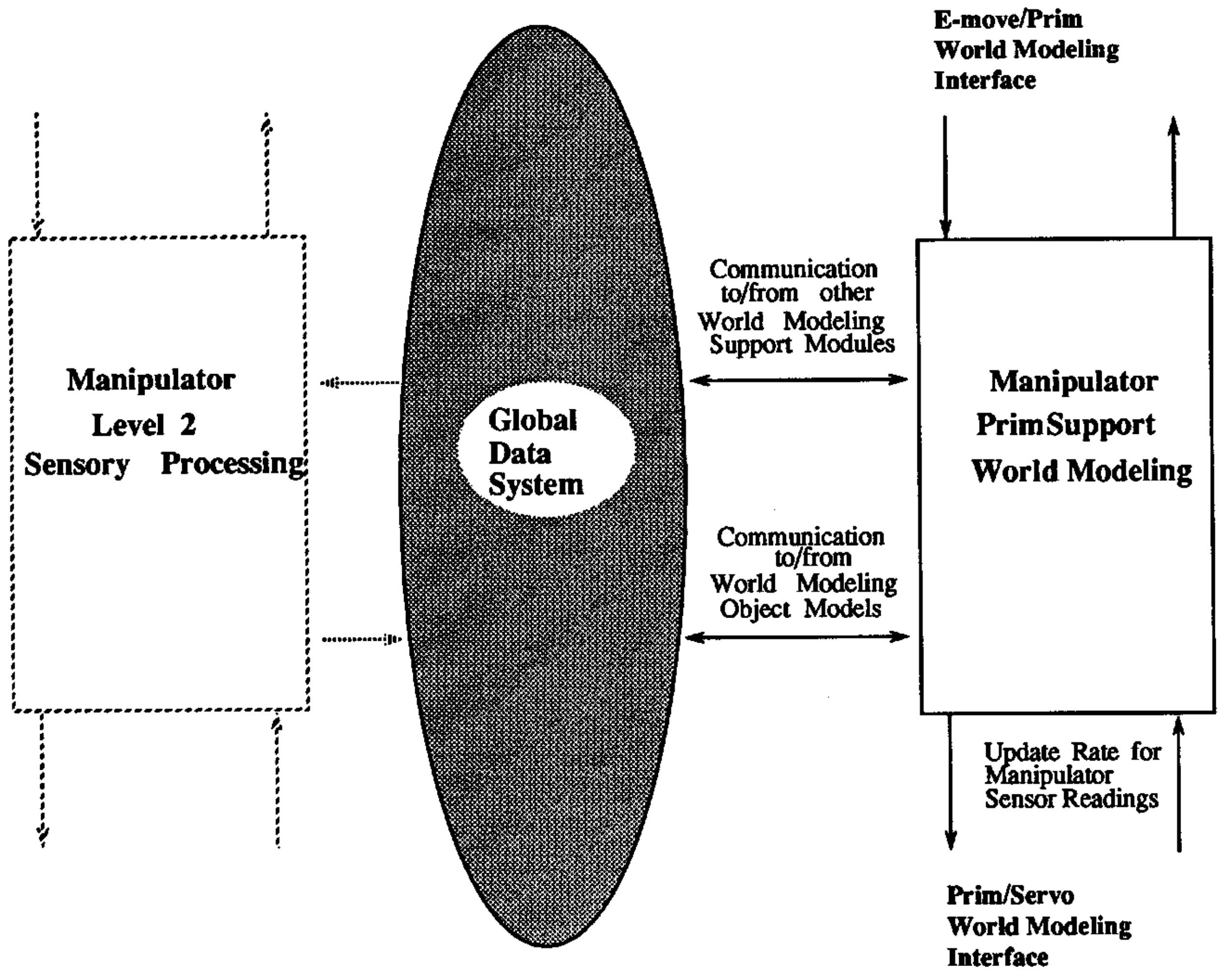
E-move/Prim World Modeling Interface
Manipulator Level 2 Sensory Processing
Global Data System
Communication to/from other World Modeling Support Modules
Manipulator PrimSupport World Modeling
Communication to/from World Modeling Object Models
Update Rate for Manipulator Sensor Readings
Prim/Servo World Modeling Interface

Figure 5. World Modeling to Sensory Processing Interface.

hierarchically to achieve a common end. For example, obstacle avoidance can be considered as a hierarchical problem; higher levels in the control system verify large spaces and lower levels provide local avoidance. In order to compute obstacle avoidance torques at the Servo Level, Prim World Modeling might provide Servo World Modeling with the necessary parameters or equations.

3. Module Operation

In this section we consider the operation and organization of computational units within the world modeling module. The discussion is intended to be general; we do not make assumptions about the programming language or the target hardware (processors) used. We discuss the advantages and disadvantages of different types of computational units, specifically *cyclically executing processes* and *function calls*. (The term *function call* is not meant to specify a particular programming language construct; it is used to denote a computational unit distinct from a *cyclically executing process*, as defined below.)

A cyclically executing process continuously reads inputs, performs computations, and then writes output in a cyclic manner. Such a process always reads and executes on the most current data; it does not wait for new data to arrive. Thus, reliable cyclic execution requires that a module be able to read or write data without delay. Reading and writing uses buffers in global memory which must be properly maintained to avoid conflicts.

Cyclic processes continue to execute even when the input does not change. Thus, for certain operations *continuous* cyclic execution is not appropriate. A coordinator, which can enable and disable cyclically executing processes, should be provided to avoid extraneous computation cycles.

Cyclically executing support processes are appropriate when the results of the computation(s) are needed without delay. Typically, the input to the process would be changing continuously, thus warranting continuous re-evaluation. Often, however, the output does not change significantly between cycles. For example, a cyclically executing process is useful for updating the manipulator dynamic model. It takes the current joint angles of the manipulator as input, and writes the gravity, centrifugal and Coriolis terms as output. For a manipulator following a trajectory, the dynamic terms would be constantly changing, but not dramatically between computations.

Cyclically executing support processes also are appropriate when several computations can, or should, be performed in parallel. For example, Prim often requires constantly updated kinematic and dynamic models. These models are both functions of the manipulator's joint positions, but they are not coupled or dependent on one another. An expeditious organization of World Modeling would include two cyclically executing processes computing these values in parallel, as shown in figure 6.

Cyclically executing support processes may not be appropriate when the Task Decomposition module requires that a World Modeling computation be evaluated for a particular input value. When the value in the world model is maintained by a cyclic process, the Task Decomposition module cannot reliably correlate the value with the input used to generate it. In order to be able to correlate input and output values, Task Decomposition would synchronize (rendezvous) with the cyclically executing process; in effect, it would treat the cyclic process as a function call and wait for the result.

Function calls are made in direct response to a need for information. The calling module encounters a need for information, calls the support function, and waits for a response.

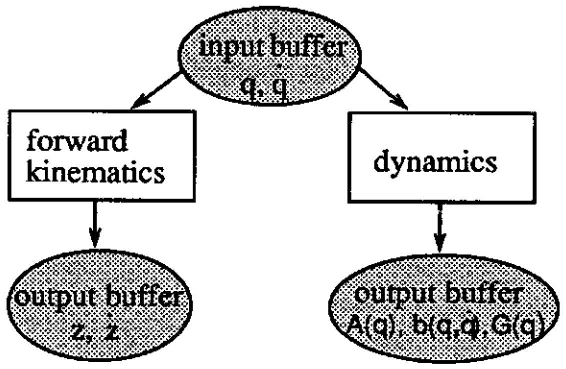

Figure 6. Parallel Cyclically Execution Support Processes.

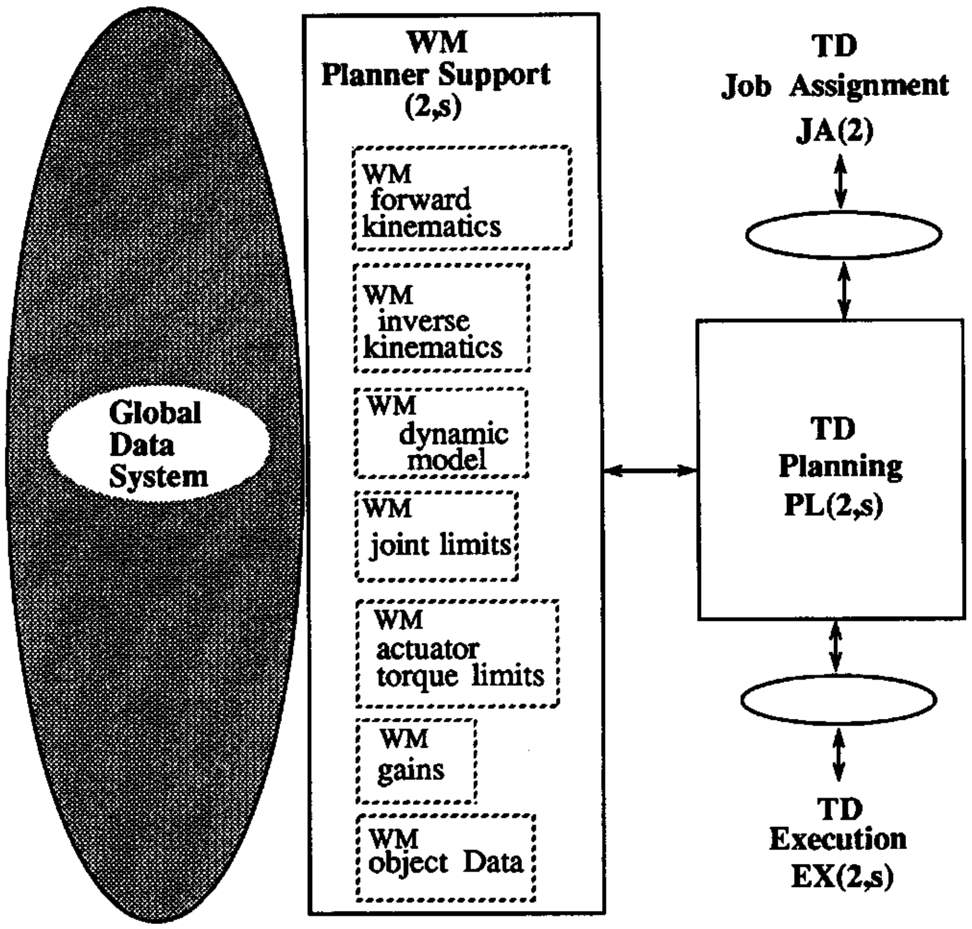

Figure 7. World Modeling Prim Planner Support.

Function calls do not waste processing power executing on unchanging input. However, since the function begins execution in response to a specific need, its output (results) cannot be available immediately; the execution of the calling module is delayed as it waits for the function to execute.

Most World Modeling processing in support of the Prim Planner should be implemented by function calls. The Planner often requires evaluation of hypothetical states. Thus its needs are not continuous and most likely cannot be known by World Modeling in advance of the call. For example, it may require World Modeling to compute the joint angles necessary to reach a projected goal state before actual execution of the move. The World Modeling functions, in support of the Planner, are shown in hashed boxes in figure 7. Some World Modeling data may actually be static and simply reside in the global data system. The information represented by each hashed box may actually be a combination of static data and one or more functions which operate on the data.

It may be useful to provide cyclically executing processes to support the Task Decomposition Execution module. Execution reads its command and input parameters at the beginning of each cycle, evaluates the trajectory, and outputs to the Servo Level [WAV88]. The Execution module basically operates in one of two modes. It either evaluates completely-planned trajectories or it evaluates sensory-interactive trajectories. In both

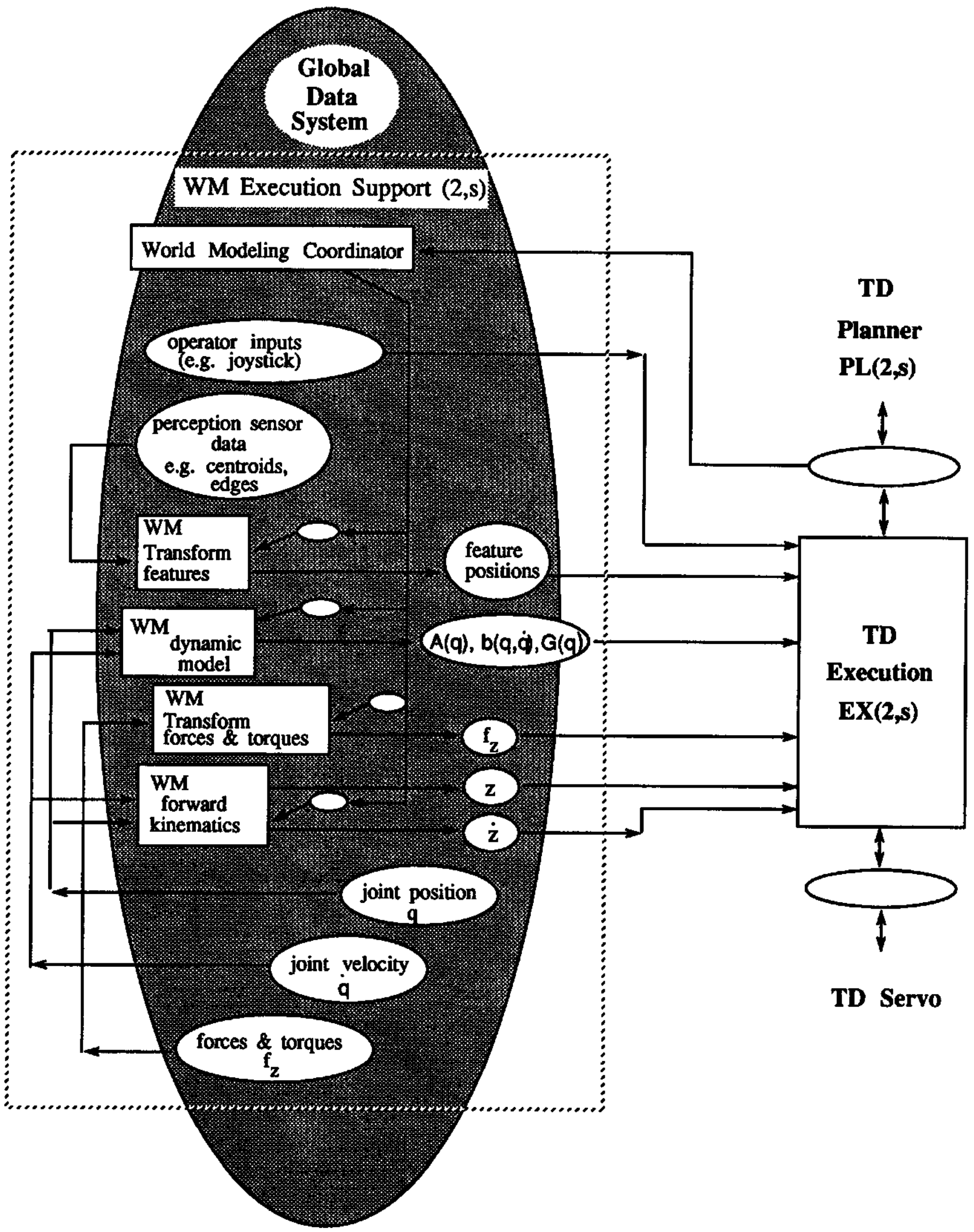

Figure 8. World Modeling Prim Execution Support.

modes, Execution requires quick response from its World Modeling support processes. Communication between Execution and the processes occurs via global memory.

Prim Execution requires that certain information about the state of the control system be readily available. However, when the manipulator is stationary, or other aspects of the environment are not changing, the processes should not continue to execute. The World Modeling coordinator enables and disables the cyclically executing processes, as required. Figure 8, shows the World Modeling Prim Execution support processes. Cyclically executing processes are pictured as rectangles. The coordinator, shown at the top, communicates directly with each process. It obtains information about the manipulator goal state by reading the Prim Execution command buffer.

All communication is done via global memory, as pictured by the ovals in the figure. In section 4, we discuss the specific computations performed by World Modeling in support of Task Decomposition, as labelled in the boxes.

4. Module Computations

The World Model maintains the system's knowledge about the state of the world. Task Decomposition must be able to obtain information about the world with little delay and in a compatible format. Static information, such as constant feedback gains, simply can be stored in the global data system. Dynamic information, such as gains for compliant motion control, must be maintained by World Modeling processes. The information should be updated and transformed into a format which is usable by Task Decomposition in a manner which is transparent the Task Decomposition module.

Each of the following sections contains an explanation of a function of the World Modeling module, as well as some methods for implementing it. Where possible we include an estimation of the computational requirements.

4.1. Kinematics

Typically, a manipulator task is specified in World (Cartesian) space. The kinematic model of a manipulator relates the position and orientation of the end-effector of the manipulator to the joint angles. The complete kinematic model describes all the geometrical and time based motions of the manipulator, without regard for the forces acting upon the manipulator.

Both the forward kinematic model and the manipulator Jacobian are required by Servo Level Task Decomposition and are computed by processes in the Servo Level World Modeling support module. The Prim support module, however, also contains such processes which independently perform the computations. Prim Task Decomposition Planning may consider hypothetical values (trajectory points) and may require World Modeling to compute the kinematics at the values. The reader is referred to [KEL89] for a discussion of forward kinematic representations, as well as methods of generating the manipulator Jacobian. We limit the discussion in this document to inverse kinematics of manipulators.

4.1.1. Inverse Kinematics

In order to do controlled movement of a manipulator, it is necessary to have an inverse kinematic model to determine the joint angles required to achieve a desired position and orientation of the end-effector. Ideally, one derives closed form equations for the inverse kinematics where each joint variable is expressed in terms of other known quantities. However, existence of a closed form inverse kinematics solution depends on the kinematic structure of the manipulator [PIE68, WOL87]. For example, we know that a closed form solution exists for a manipulator which has six degrees of freedom, three of which have intersecting axes, such as in a spherical wrist. This solvability condition is not necessary, only sufficient. The more general, but also more computationally intensive, method of solution involves numerical iteration.

When a closed form solution does not exist, the kinematic model can be solved by standard numerical methods. The equations are solved without regard for their physical significance or the shape of the manipulator. The most commonly used method to solve a system of nonlinear equations, such as those comprising the kinematic model, is Newton-Raphson [KLE83]. While analytic, or closed form, solutions to inverse kinematics often yield multiple solutions for a given position, a solution by Newton-Raphson yields only one. It find the zero for the equations which is closest to the initial guess, or current configuration, by iteratively solving the first order Taylor series expansion [KEL88].

The forward kinematics model can be expressed by the following equation:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}),$$

$$\mathbf{x} - \mathbf{f}(\mathbf{q}) = \mathbf{0},$$

where the Cartesian position (**x**) is a function of the joint angles, **q**. (Note: bold typeface denotes a vector or a matrix.) Expanding the second equation into its first order Taylor series expansion for the independent variable, **q**, we get:

$$\mathbf{q} = \mathbf{q}_0 + \frac{\mathbf{x} - \mathbf{x}_0}{\mathbf{J}(\mathbf{q})},$$

where **J(q)** is the manipulator Jacobian. (The Jacobian is the matrix of partial derivatives of **x** with respect to **q**.) We rewrite this equation as:

$$\mathbf{dq} = \mathbf{J}^{-1}(\mathbf{q}) \cdot \mathbf{dx},$$

and iteratively solve it to determine the necessary changes in the joint variables to achieve the desired change in Cartesian position. (The corresponding instantaneous relation is: $\dot{\mathbf{q}} = \mathbf{J}^{-1}\,\dot{\mathbf{x}}$.) Note, however, that the matrix **J** is only invertible when it is square and of full rank. These conditions are violated when the manipulator is redundant or when the manipulator is in a singular configuration. In the following section we discuss inverse kinematics for redundant manipulators.

Computational Requirements: The majority of the computational burden in computing the inverse kinematics via a numerical method is from computing and inverting the Jacobian.

Paul and Zhang [ZHA88] then present an efficient method for obtaining the Jacobian (of a six degree of freedom manipulator with a spherical wrist) from the Denavit-Hartenberg forward kinematic model. The evaluation of the Jacobian requires 6 sine/cosine pairs, 46 multiplications, and 19 additions. The number of computations needed to find **dq** can be reduced by solving the equation:

$$\mathbf{dx} = \mathbf{J}(\mathbf{q}) \cdot \mathbf{dq}$$

explicitly for **dq** using Gaussian Elimination, or a comparable method. Solving the above equation for a six degree of freedom manipulator requires on the order of 10 divisions, 150 multiplications, and 80 subtractions.

Another concern associated with using numerical methods, such as Newton-Raphson, to solve the kinematics equations, is the convergence or robustness of the method. When the manipulator approaches a singular configuration, the manipulator Jacobian becomes ill-conditioned. The convergence rate of the method decreases, and therefore requires a greater computation time. When the manipulator is in a singular configuration, the Jacobian becomes singular and therefore non-invertible; the equations cannot be solved. The problem of manipulator singularities has received a great deal of attention, specifically in the literature on redundant manipulators. We continue our discussion of inverse kinematics in the next section on redundant manipulators.

4.1.2. Redundant Manipulators

Because of the benefits of kinematic redundancy (e.g. increased dexterity, obstacle avoidance, singularity avoidance), many manipulators are configured with more than six degrees of freedom. Introducing redundancy complicates the control algorithm. Most research on the control of redundant manipulators has involved the instantaneous resolution of the redundancy at the joint velocity level [BAI84, CHA86, KLE83, NAK86]. The inverse Jacobian $\mathbf{J}^{-1}$, which relates the change in joint variables to the change in Cartesian position, is replaced by the pseudoinverse $\mathbf{J}^{+}$.

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

While the pseudoinverse provides an adequate means for resolving kinematic redundancy, many researchers have explored methods to improve the pseudoinverse [BAI84, CHA86, KLE83, YOS84]. The instantaneous changes in the joint angles are found according to:

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{x}} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\nabla H(\mathbf{q}),$$

where $\mathbf{H(q)}$ is an optimization criterion to be maximized/minimized and $\mathbf{I}$ is an identity matrix. The term $(\mathbf{I} - \mathbf{J}^+\mathbf{J})$ causes the gradient of the optimization function $(\nabla\mathbf{H(q)})$ to be a vector in the nullspace of the Jacobian. 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.

An often used criterion is singularity avoidance [BAI84, YOS84]. The function $\mathbf{H(q)}$ can be defined by Yoshikawa's manipulability index [YOS84]:

$$\mathbf{H(q)} = \sqrt{\text{determinant}(\mathbf{J}\cdot\mathbf{J}^T)},$$

which is a measure of the manipulator's distance from a singular point. The value of $\mathbf{H(q)}$ decreases as the manipulator nears a singularity; in a singular state the value of the determinant of $(\mathbf{J}\cdot\mathbf{J}^T)$ is zero. Thus, maximizing $\mathbf{H(q)}$ moves the manipulator away from singular configurations.

The singularity robust inverse [NAK86] presents an alternative for $\mathbf{J}^+$. 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 velocities, Nakamura and Hanafusa propose the singularity robust inverse.

The singularity robust inverse, $\mathbf{J}^*$, is defined as:

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

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, KEL88, MAC88].

The above redundancy resolution methods employ local optimization when resolving end-effector velocities to joint velocities for redundant manipulators. Alternatively, the kinematic equations can be constrained fully by augmenting the six functions for the Cartesian space end-effector coordinates with additional configuration constraint(s). By adding a constraint for each redundant degree of freedom in the manipulator the system becomes specified fully and no longer redundant for the task.

This approach, sometimes called *configuration control* [SER89], utilizes the redundancy to control the manipulator configuration directly in task space. Additional kinematic functions can shape the manipulator without altering the end-effector position or orientation. A constraint can be added to control the Cartesian coordinates of any point of the manipulator. For example, it may be desirable to use the redundant manipulator's self-motion to reconfigure it so that its elbow avoids an obstacle in the workspace. Thus, the entire, or global, motion of the manipulator is controlled.

The above discussion of redundancy resolution has been limited to kinematics. The manipulator redundancy can also be resolved dynamically, as in the method proposed by Hollerbach [HOL87]. The method uses the redundancy to minimize the torque loading by minimizing the kinetic energy. He proposes formulating the generalized inverse in terms of accelerations and incorporating it into the dynamics. In this way the resolution of redundancy is directly related to the joint torques, with the desired effect being that the manipulator avoids exceeding torque limits.

Computational Considerations for Pseudoinverse based methods: The number of computations required to compute the pseudoinverse of the Jacobian depends on the number of degrees of freedom in the workspace, as well as the degrees of freedom of the manipulator. If possible, a closed-form kinematic model should be created for the manipulator Jacobian. 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].

4.2. Manipulator Joint and Actuator Torque Limits

Each actuator of the manipulator has a range of valid positions which it may assume. For example, a joint may range from -180° to +180°. However, truly safe positions depend upon the positions of the other joints in the manipulator. Care must be taken to assure that each joint of the manipulator does not exceed its individual limit and that the manipulator does not collide with the floor (ceiling), or even itself. Safe configurations for the manipulator can be determined by examining the configuration space of the manipulator [NEW89]. Configuration space approaches have been examined in the context of obstacle avoidance. We consider the problem of avoiding other obstacles in the workspace as a distinct problem, which we discuss in section 4.5.6.

Each actuator also has a torque limit. Again, this limit is not independent of the overall manipulator configuration. Each joint's torque limit is a function of its position and velocity, as well as the position of the other joints of the manipulator. World Modeling verifies positions and torques hypothesized by the Prim Planner. It also verifies the final values computed by the Executor before they are passed to Servo for realization.

4.3. Gain Information

The controller gain matrices K_p, K_v, and K_i are chosen to achieve the specified system output response. (The subscripts p, v, and i represent position, velocity, and integral gains, respectively.) The gain matrices may be constant or the control algorithm may vary them during execution.

For example, a fairly simple control law, the individual joint PID control law, controls each joint individually, computing the joint torques according to:

$$\tau = \mathbf{K_v} \dot{E} + \mathbf{K_p} E + \mathbf{K_i} \int E \, dt,$$

where $\mathbf{K_p}$, $\mathbf{K_v}$, and $\mathbf{K_i}$ are diagonal NxN matrices and E and $\dot{E}$ are nx1 vectors of position and velocity errors [CRA86, FIA88]. Typically, the gain matrices are chosen so that the system is critically damped and avoids overshoot of any of the joints. However, no one set of gains will provide a critically damped response of the manipulator in all configurations; the gains should be updated with changes in the manipulator configuration.

The manipulator controller gains also may be varied with variations in the workspace (task) constraints. Often maximal stiffness is desirable for free space motions. However, when performing contact tasks, such as the insertion segment of a peg-in-hole task, it is desirable that the end-effector of the manipulator be compliant. That is, for contact tasks it is desirable to adjust the manipulator gains so as to decrease the stiffness of the system. (See sec. 4.5.5 for a discussion of task compliance.) World Modeling would maintain the matrices $\mathbf{K_p}$, $\mathbf{K_v}$, and $\mathbf{K_i}$ to support the Prim Planner.

4.4. 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. Sensed forces can be used to generate corrective motions of the manipulator. Force sensor readings $\mathbf{F} = (f_x, f_y, f_z, m_x, m_y, m_z)$ from the manipulator end-effector are transformed to the base or tool tip frame (or any other frame specified by C_z) according to:

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

where A denotes the desired frame of reference and B denotes frame in which the readings were taken [CRA86]. World Modeling enables decoupling of the Sensory Processing coordinate system of the readings and the Task Decomposition coordinate system of the control algorithm. The Jacobian transpose also maps forces acting at the end-effector into equivalent joint torques by:

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

World Modeling makes the joint torques available to Prim Task Decomposition.

Computational Requirements: Assuming that the Jacobian exists in the World Model, computing the torques simply involves multiplying the transpose of the 6xN Jacobian (where N is the number of degrees of freedom in the manipulator) by the 6x1 vector of forces.

4.5. Object Data

The Prim Planner requires a variety of data about the objects in the world, and in particular, about the object(s) to be manipulated. In this section we discuss object information as it pertains to Prim Level algorithms, including sensory interactive algorithms performed by Prim Execution. World Modeling provides Prim with the object data necessary to plan and execute trajectories and fine motion strategies.

4.5.1. Mass and Distribution of Mass

Prim must know the mass of an object to be lifted in order to plan an optimal, or even feasible, dynamic trajectory. The manipulator dynamic model must be adjusted to account for the additional mass of the object in the manipulator's gripper. Some components of the manipulator dynamic model, such as the manipulator gravity torques, simply require the mass and a vector to the center of mass of an object in the manipulator's gripper. Other components, such as the inertia torques, including interaction torques, require a representation for the mass distribution. The mass distribution is given by the inertia tensor, which is a 3x3 symmetric matrix, and is computed by integrating the mass density over the volume of the rigid body [ASA86].

4.5.2. Effect of Held Objects on Manipulator Dynamics and Kinematics

The Prim Planning module may require a dynamic model of the manipulator when planning optimal trajectories. Such trajectories minimize the time, actuator torques, or other constraints required to traverse a path to the goal state [WAV88]. One such algorithm is the minimum-cost trajectory planning (MCTP) technique [SHI86], which minimizes the cost function subject to the actuator torque constraints and the manipulator dynamics. World Modeling would supply the dynamic model, as well as the actuator torque constraints. The reader is referred to [KEL89] for a discussion of methods of computing the dynamic model of a manipulator. In this section we consider how to adjust the manipulator dynamic and kinematic models to account for an object in the manipulator's grasp.

The dynamics of a manipulator carrying an object of substantial mass differ from those of the manipulator by itself. The dynamic model of the manipulator carrying an object can be corrected most easily by adjusting the mass and the vector to center of mass of the last link. Specifically, for a 7 degree of freedom manipulator grasping an object, the vector to the center of mass of the combined seventh link and object ($\mathbf{r'}_{\mathbf{com}}$) can be computed from the following:

$$(m_7 + m_{obj})\,\mathbf{r'}_{\mathbf{com}} = m_7\,\mathbf{r}_{\mathbf{com\text{-}7}} + m_{obj}\,\mathbf{r}_{\mathbf{com\text{-}obj}}.$$

In the above expression, m represents mass, and $\mathbf{r}_{\mathbf{com}}$ represents the vector to the center of mass of the object (link). The subscripts "obj" and "7" represent the object and link 7, respectively. The vector to the center of mass of the object ($\mathbf{r}_{\mathbf{com\text{-}obj}}$) must measure the distance from the point of grasp of the object by the manipulator's gripper to the center of mass of the object. The vector can be computed simply given the vectors from a reference on the object to its center of mass and to the point of the grasp.

The kinematic relation between the base of the manipulator and the end point (tool frame) also change when it is carrying an object. For a large (long) object, the change can be significant. For example, when avoiding obstacles, the added length must be incorporated into the calculations. Also, when planning a move to place the tip of the object at a desired location, the inverse kinematics model of the manipulator must account for the length of the object. As for dynamics, the point of grasp of the object is significant in determining the change to the manipulator's kinematics. The displacement kinematic parameter(s) of the last link must be augmented by the distance from the point of grasp to the tip of the object.

In order to determine the necessary vectors for correcting the kinematic and dynamic models of a manipulator when grasping an object, many dimensions of the object must be specified in its model. First, an object reference frame should be chosen. The vector to the center of mass of the object (with respect to the reference frame) should be determined and stored with the model. Depending on the point of grasp, different dimensions are required. If there are a limited number of stable (desirable) grasps, then the vectors from the grasp point to the tip and to the center of mass could be stored. Otherwise, enough dimensions of the object must be stored to allow for computing the necessary vectors.

Computational Requirements: Highly efficient recursive algorithms for the inverse dynamics problem result from optimizing and customizing the Newton-Euler algorithm [HOL80, KHO86]. Symbolic equations have the advantage that they can be optimized so as to be more efficient than the recursive methods. The reader is referred to [KEL88] for additional discussion on the computational requirements of dynamics. Adjusting the dynamic model when a manipulator carries an object adds negligibly more computations to the complex model. The primary additional function for World Modeling is to transform and incorporate sensory readings into the model so that Task Decomposition continues to receive an updated model.

4.5.3. Position and Velocity

Typically, the Planner must know the location of an object of interest. If the object is moving, World Modeling must supply the velocity, as well as the position, of the object. A discussion of rigid body poses and motions follows. In support of the Prim Executor, World Modeling supports sensory-interactive algorithms which often provide *relative* object-manipulator displacements. In this section, we expand upon several visual sensory interactive algorithms and their implementation in a hierarchical control system. Finally, we discuss a reflexive obstacle avoidance algorithm based upon proximity sensor readings of the manipulator's distance to an object.

Position (pose) of an object in space requires both a vector of position (p_x, p_y, p_z) and a representation of orientation. Orientation is typically given by either a 3x3 rotation matrix or a 3x1 vector (axis) and an angle of rotation about the vector. (A quaternion is a specific angle-axis representation.) The pose of a rigid body (object) can be fully specified by the pose of any point on the object. Poses of objects in a manipulator system must be transformed into the working frame of reference. That is, the frame of reference of an object's pose should be consistent with that used for control of the manipulator. (See [KEL89] for additional discussion of pose representations.)

Poses (position and orientation) are often expressed symbolically by T_x, where x denotes the reference frame for the pose. For use in a manipulator control system, two reference frames for the object pose are convenient: the world frame (W) and the manipulator's end-effector (EE). Therefore, the object pose is stored with the object as: T_W or T_{EE}.

When a rigid body moves with linear velocity (translation), each point in the body moves with equal velocity, and the velocity of the body can be given by *v*. However, for general motions, it is convenient to speak of the velocity of the center of mass ($\mathbf{v}_{\mathbf{c.o.m.}}$) of the body and the velocity of all other points with respect to the center of mass [BEE77, LAN76]. That is, the velocity of a point *a* in the rigid body is given by:

$$\mathbf{v}_{\mathbf{a}} = \mathbf{v}_{\mathbf{c.o.m.}} + \mathbf{v}_{\mathbf{a/c.o.m.}},$$

where $\mathbf{v}_{\mathbf{a/c.o.m.}}$ is the velocity of point *a* with respect to the center of mass of the object. The translational velocity of the body is $\mathbf{v}_{\mathbf{c.o.m.}}$. The second velocity term, $\mathbf{v}_{\mathbf{a/c.o.m.}}$, can be expressed as:

$$\mathbf{v}_{\mathbf{a/c.o.m.}} = \mathbf{\Omega} \times \mathbf{r},$$

where $\mathbf{\Omega}$ is the angular velocity of the rotation of the body and $\mathbf{r}$ is the radius vector from the center of mass to an arbitrary point.

Each component of the velocity of a rigid object can be expressed by a parametric equation in *t*. That is:

$$\mathbf{v} = v_x(t)\,\mathbf{i} + v_y(t)\,\mathbf{j} + v_z(t)\mathbf{k}.$$

Alternatively, the motion of an object can be given as the motion of the end-effector (gripper) of the manipulator carrying the object:

$$\mathbf{v}_{\mathbf{obj}} = \mathbf{v}_{\mathbf{ee}},$$

where $\mathbf{v}_{\mathbf{ee}}$ (velocity of the end-effector) includes the translational and rotational velocities. This relationship presumes that the reference frames of the object and the end-effector are coincident and that the grasp is firm so that the object does not move with respect to the grasp point.

4.5.3.1. Position and Velocity in Sensory Interactive Trajectories

Real-time trajectory generation and/or replanning based on sensory feedback, such as vision and proximity, allows performance of tasks based on sensed data rather than a priori knowledge. This provides for robustness of task execution amid incomplete or uncertain knowledge of the environment. Robot positioning systems base their corrective motions on the difference between the desired, or reference, joint positions and the actual ones. Because of possible errors in the manipulator kinematic model, the position of the end-effector corresponding to the desired joint positions, may not actually correspond to the desired Cartesian position. When using joint sensor readings for feedback, there is no way to determine the final Cartesian positioning error since no direct measurement of the final end-effector position is available. With the addition of perceptual sensory feedback, the control system can more reliably relate the pose of the manipulator's end-effector to objects (cues) in the world.

The amount of sensory interaction varies with the control algorithm or technique. For example, in the Camera Space Manipulation technique [SKA87], the joint space trajectory is planned based upon positions of world cues (relative to the manipulator) in camera space (the image). The control algorithm is tightly coupled with the visual sensory data, as well as the joint readings. In this section we consider the role of the Prim World Modeling support module within two different sensory (visually) interactive control techniques: Camera-Space Manipulation (CSM) and Image-based Visual Servo (IBVS).

Camera Space Manipulation

In Camera Space Manipulation (CSM) no effort is made to map camera-space locations into real-space ones. CSM tasks are defined such that accomplishment of a task coincides with achievement of a particular configurational relationship among preplaced visual cues on the objects of interest [SKA87]. In the CSM method, the system iteratively learns the "camera-space kinematics" to plan manipulator trajectories. Corrective feedback is based upon the relative placement of the visual cues in camera-space, *not* real-space. Tedious camera calibration becomes unnecessary.

The CSM relationships are learned in real time, with limited (nominal) a priori positional knowledge regarding the locations of the camera or manipulator in real-space. Consider the position of a cue on an object in the manipulator's grasp. Its camera-space position (X_c, Y_c) is related, with several uncertainty parameters, to its position in real-space (X_r, Y_r, Z_r), as measured with respect the base of the manipulator. The real-space uncertainty parameters derive from uncertainty in the robot kinematic model, as well as uncertainty in the grasp. The camera-space uncertainty parameters result from uncertainty in the "view," or the camera's location and orientation in space.

Both the real-space and camera space uncertainty parameters are estimated and updated numerically (iteratively). Once the uncertainty parameters stabilize to constant values, World Modeling can compute the manipulator joint angles necessary to complete the task. The results, or output, of the method are sequences of joint angles necessary to unite the object cue(s) with the destination cue(s).

The role of Prim World Modeling in CSM is to provide Prim with the sequence of joint angles which realize the goal position for the motion; Prim need not be concerned with the CSM uncertainty parameters and models. World Modeling performs the numerical iterations to update the uncertainty parameters and its model of the "camera-space world". Sensor input consists of the centroids of the cues in the image, as well as manipulator joint readings. The camera modules in the perception branch of the hierarchy (fig. 3) write the image data to the global data system; the manipulator Level 1 Sensory Processing module writes the joint angle readings to the global data system. Prim World Modeling retrieves the sensor values and provides updated goal states (joint angles) for Task Decomposition.

Image-based Visual Servo

Visual servo control structures take many forms. Many are static "look and move" strategies where manipulator pauses as the vision system estimates the object pose. The controller then computes the error between the manipulator position and the reference position. Such strategies are used for joint space control algorithms, as well as Cartesian space or "position based visual servoing". In both, the vision system provides direct world space information (position). However, in the latter, the controller computes both the error and correction in Cartesian space, thereby eliminating the need for time consuming inverse

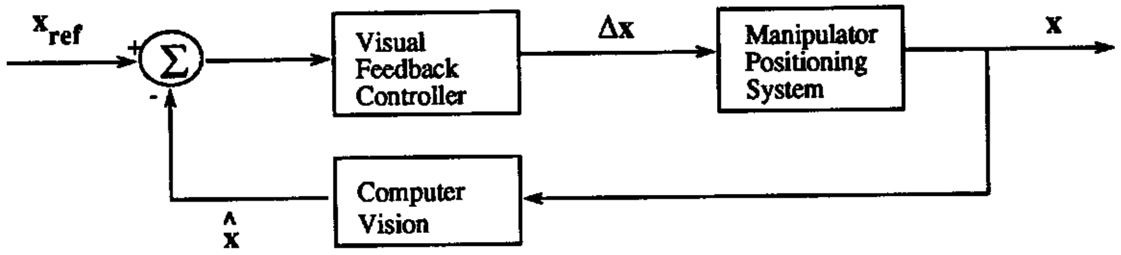

Figure 9. Dynamic Visual Servo Control [WEI87].

kinematic computations.

Weiss, et al. [WEI87], have presented a method for image-based visual servo (IBVS) control. The method employs image features to determine spatial position. The features are used as the basis for control, rather than position estimates of the manipulator joint angles. In an IBVS control system, the feedback signals are defined in terms of image features which correspond to current robot positions. Figure 9 shows the closed-loop schematic for the method. The reference (x_{ref}) and feedback vectors ($\hat{x}$) are based upon image space features. IBVS is a dynamic control system; the three steps in figure 9 are executed in parallel. The position estimate ($\hat{x}$) and position error (Δx) are updated as fast as they can be measured, while the robot is moving with a camera mounted on its wrist. The position error (Δx) is fed into the closed-loop manipulator positioning system which includes dynamic joint servo controllers and kinematic routines necessary to generate movements in Cartesian space.

The box labelled "Computer Vision" (fig. 9) is performed by the camera Sensory Processing modules. Features, such as edges or centroids, are extracted from the filtered or enhanced images. It is important to choose features which are not highly sensitive to changes in camera position. Sensory Processing is guided by World Modeling towards this end. The box labelled "Visual Feedback Controller" (fig. 9) represents the processing performed by World Modeling. It includes differencing the position of the reference or desired feature(s) and the actual feature(s). The error in image space must then be transformed into corrective motions in real space. Prim Task Decomposition computes the sequence of attractor sets to achieve the real space correction.

Reflexive Obstacle Avoidance

The primary mode of obstacle avoidance at the Prim level is reflexive, in response to unexpected or unmodeled obstacles. Such avoidance must occur rapidly based upon new sensory input. One method, proposed by Espiau and Boulic, uses proximity sensors to detect obstacles [ESP86]. The method exploits the extra degree of freedom of a redundant manipulator. The inverse kinematics model for the manipulator incorporates the vector of readings from a proximity sensor at each joint to propel the joint from the obstacle. The algorithm generates motions in the nullspace of the manipulator Jacobian to avoid collisions.

Another reflexive obstacle avoidance method employs avoidance torques which are added to the servo control torque [KHA87]. In Khatib's method a hypothetical force pushing away from an obstacle is computed, converted to a torque through the manipulator Jacobian, and added to the control torques to provide local obstacle avoidance. Because Khatib's method generates joint torques, it would be implemented at the Manipulator Servo Level and is discussed in [FIA88, KEL88].

4.5.4. Stiffness and Friction Characteristics

The Prim Planner uses a measure of the stiffness of objects in its environment when planning contact tasks. When the end-effector of a manipulator contacts an object, a reaction force is applied to the manipulator. The amount of deflection experienced by the manipulator depends on the stiffness of the manipulator, as well as the stiffness of the object. The Prim Planner can anticipate the reaction forces by considering the stiffness characteristics of the object.

Prim also can use knowledge of the friction characteristics of object surfaces in developing its plans. Most algorithms for grasping and part alignment via sliding exploit contact friction. Objects stick to surfaces as a result of tangential reaction forces caused by friction. The coefficient of friction of each surface of an object should be stored in the global data system for use by the Prim Planner.

Much research has been done on the area of *sensorless manipulation* [PES85, PES88, ERD86]. Sensorless manipulation aims to reduce the positional and orientational uncertainty of assembly parts. As the name suggests, the uncertainty of a part on a plane is reduced without using sensing. The surface friction between a part and the surface upon which its slides is exploited to position and orient the part. By sliding the part into a sequence of carefully aligned straight fences, the pre-specified position and orientation of the part can be achieved. To date, only a limited number of parts have been successfully positioned using this highly specialized method.

4.5.5. Mechanical Constraints and Task Compliance

In order for a robot to perform tasks involving contact of rigid bodies, the compliance of the robot or the objects (world) must be considered. Under position control without such compliance, small errors in the positions of the objects can lead to failure of the task or damage to the manipulator or object. Compliant motion methods have been proposed which control force and position in complementary directions (axes) at each instant during execution of a task [HOG85, MAP86]. Directions which are position controlled should be stiff to avoid positioning errors; directions which are force controlled should have little or no stiffness.

Compliant motion control is concerned with the control of a manipulator in contact with its environment. It is dependent on the geometric and mechanical contact characteristics of the task configuration. For this reason, compliance is often addressed with respect to an object and its role in a specific task [MAS81]. The term *specific task* does not preclude classes of tasks and generalizations.

The task of inserting a peg in a hole is an example of an end-effector constrained problem because of the contact at the end-effector with the environment. It can be broken into three

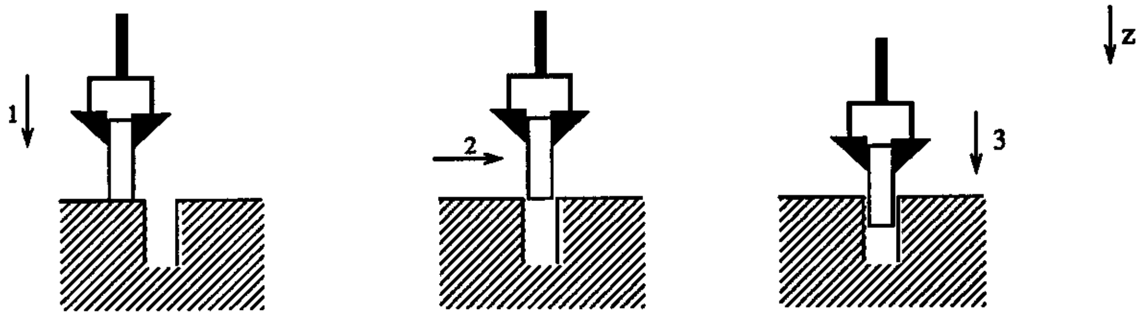

Figure 10. Peg-in-Hole Task.

phases in which the number of degrees of freedom of the peg decreases: guarded move until the opposing force in the z-direction becomes greater than 0 ($f_z > 0$), sliding move until f_z = 0 and the peg experiences a downward velocity ($v_z > 0$), snug fit insertion. (See fig. 10.) The constraints of the task should be exploited during the final stage where the peg is constrained to move only in the z-direction.

While the compliance between two objects is task dependent, the model should be general enough to handle variations. Consider, again, the "peg-in-hole" task. Many mating operations can be considered as variations on the task and can be controlled as such. Also, many variations for the "holes" themselves may exist; several are depicted in figure 11. Sliding towards the hole may require more "thought" than would be anticipated by just examining the example in figure 10.

Compliance can be achieved by equipping the manipulator's end-effector with a *passive* mechanical device composed of springs and dampers. One such device, the Remote Center Compliance (RCC), allows one to place the *compliance center* at the point of contact of the task [WHI82]. At the compliance center, applied forces cause pure translation of the contact point and applied torques cause pure rotation about the point [ASA86]. Control can then be achieved by specifying the motion of the compliance center in the task space.

A passive mechanical device provides compliance capabilities for a specific task. However, a change in task parameters, such as the size of the peg, may render the device unusable. An *active* compliance scheme, including a software model of the task, is more flexible and more easily modified with variations in the task parameters.

World Modeling provides Task Decomposition with enough information about the geometry of the task to enable it to execute successfully. In hybrid position/force control

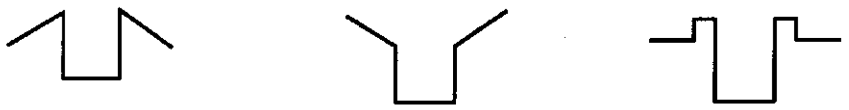

Figure 11. Variations on Peg-in-Hole Task.

[MAS81, RAI81, WES85, WHI87], the end-effector space is divided into two orthogonal domains, a position domain and a force domain. Raibert and Craig design a hybrid controller using a selection matrix, S, to *select* between the position and force controlled directions. The selection of which end-effector directions should be controlled in which mode does not remain constant; it changes with the position of the phase of the task and the position of the manipulator. World Modeling either could maintain the S matrix, or it could provide Task Decomposition with a geometric representation of the task which enables it to switch control of the end-effector between position and force modes.

Several methods have been developed which rely on a priori knowledge of the contact surfaces encountered during a mating task [ELM89, LAU89, SAW89, XIA89]. In [LAU89] assembly strategies are automatically generated and stored as state graphs. The method consists of two phases: the *analysis phase* which constructs a state graph by reasoning about ways of dismantling the assembly and the *search phase* which finds an optimal path (in terms of efficiency and/or reliability) to complete the assembly. In a real-time control system, the first phase, generation of the state graph, would be performed prior to task execution by World Modeling; the results would be stored in the global data system. Prim would access the state graph during execution of fine motions.

Programmed compliance is a logical extension of compliant motion. The objective of *programmed compliance* is "to program the compliance of the manipulator, in advance, in such a way that during assembly the force vector characterizing any possible positional error is mapped into a corrective displacement of the manipulator." [PES88] In such a system, if during execution of a peg-in-hole task the manipulator encounters a force along the axis of the peg (the z-axis), it knows to correct for the positional displacement from the hole by sliding along the surface to the hole.

4.5.6. Spatial Occupancy

In order to enable robots to operate in environments which are not fully constrained, robot control systems must be able to react to obstacles (both expected and unexpected) in the robot's environment. Certainly, in a hierarchical manipulator control system, planning for obstacle avoidance would occur high up (at the E-move level) in the hierarchy. However, obstacle avoidance techniques should be employed at all levels. Lower levels in the hierarchy must provide *reflexive* obstacle avoidance to enable the manipulator to avoid previously unaccounted for obstacles. In this section we present several obstacle avoidance techniques and discuss the role of World Modeling in their execution.

Inherent in the problem of obstacle avoidance is the problem of representing the obstacles, including both stationary and moving objects and manipulators. World Modeling maintains a model of the manipulator's workspace. The model includes models of objects, as well as models of other manipulators in the environment. The object models may be generated by various CAD modeling systems. They be stored as boundary representations, generalized cones (swept volumes), octrees, or other alternative representations. Our intention in this document is not to discuss the various representations except as they apply to the problem of obstacle avoidance at the Prim Level.

An octree is a recursive decomposition of a cubic space into subcubes [JAC80]. A spatial occupancy map, or octree, may be created and updated based upon approximate volumes occupied by objects. World Modeling requires only a rough representation of the volume of an object in order to incorporate it into the octree. For example, an object may be represented by the rectangular parallelepiped which bounds it. Octrees provide a spatially-indexed representation of the world which allows for quick determination of occupancy (or vacancy) of a particular region of the world. The hierarchical, multiresolution nature of octrees may be utilized to improve the speed of search algorithms [HER86]. Paths hypothesized by Task Decomposition can be checked easily for collisions with obstacles by World Modeling. The octree would exist in the global data system; it could be accessed and updated by World Modeling at several levels. E-move World Modeling support could check for large volumes (rough approximations) of unoccupied space. It could communicate the volumes of interest to Prim World Modeling support which would perform higher resolution decomposition of the volumes of interest.

An alternate representation of free space can be generated using configuration space approaches [CHE88, NEW89]. Such approaches would most likely be instigated at the E-move Level of the system. These approaches require a search space representation of obstacles which is distinct from the world space representation. Obstacles (objects) in the manipulator environment must be mapped into configuration space obstacles. Creation and maintenance of such a representation for a dynamic environment is computationally intensive and not feasible at the Prim Level. Furthermore, Prim's function in a hierarchical control system is to plan trajectories for closely-spaced manipulator goal states [WAV88].

Newman describes a *reflex control* scheme. It is a configuration space approach for avoiding obstacles which exploits the continuous nature of path planning to reduce the number of computations. The reflex controller does not re-inspect regions which it has inspected on previous cycles [NEW89]. In order to compute collision free paths in real time, Newman chooses a configuration space approach. Setpoints found to be safe during one cycle, are assumed safe on the next. The controller only performs incremental inspections, in configuration space, between the established setpoints and the goal setpoint. Such a model could be maintained by the E-move World Modeling module. The model could be used at the Prim Level to verify the vacancy of locations at an incremental displacement.

5. Conclusions

This document has given a description of Prim World Modeling for a hierarchical manipulator control system. The function and interfaces of the World Modeling module have been described. The specific functions and computations of the support processes depend on what algorithms are implemented in the Task Decomposition module. Also, the particular transformations required by the World Modeling support module depend on how the model of the world is stored in the global data system. However, algorithm independent interfaces, which allow for both pre-planned and sensory interactive trajectories, have been defined. Examples of several general classes of computations have been discussed. We included a discussion of cyclically executing processes and function calls in an attempt to raise some of the issues and trade-offs involved in organizing the module.

6. 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.

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

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

[BEE77] Beer, F.P., Johnston, E.R., *Vector Mechanics for Engineers: Statics and Dynamics*, 3rd Edition, McGraw-Hill Inc., New York, 1977.

[CHA86] 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.

[CHE88] Chen, Y., Vidyasagar, M., "Optimal Trajectory Planning for Planar n-Link Revolute Manipulators in the Presence of Obstacles," *Proceedings 1988 IEEE International Conference on Robotics and Automation*, Vol.1, April, 1988.

[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, pp. 215-221, 1955.

[ELM89] Elmaraghy, H.A., Payandeh, S., "Contact Prediction and Reasoning for Compliant Robot Motions," *IEEE Transactions on Robotics and Automation*, pp. 533-538, August, 1989.

[ERD86] Erdmann, M., "Using Backprojections for Fine Motion Planning with Uncertainty," *The International Journal of Robotics Research,* Vol.5, No.1, Spring, 1986.

[ESP86] Espiau, B., Boulic, R., "Collision Avoidance for Redundant Robots with Proximity Sensors," *Robotics Research: The Third International Symposium,* MIT Press, 1986.

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

[HER86] Herman, M. "Fast, Three-Dimensional, Collision-Free Motion Planning," *Proceedings 1986 IEEE International Conference on Robotics and Automation*, Vol. 2: 1056-1063, April, 1986.

[HOG85] Hogan, N., "Impedance Control: An Approach to Manipulation," *ASME Journal of Dynamic Systems, Measurement, and Control,* pp. 1-24, March, 1985.

[HOL87] Hollerbach, J.M., "Redundancy Resolution of Manipulators through Torque Optimization," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 4, August, 1987.

[JAC80] Jackins, C.L., Tanimoto, S.L., "Oct-trees and Their Use in Representing Three Dimensional Objects," *CGIP*, 14:3, pp.249-270, November, 1980.

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

[KEL88] Kelmar, L., Khosla, P.K., "Automatic Generation of Kinematics for a Reconfigurable Modular Manipulator System," *Proceedings 1988 IEEE International Conference on Robotics and Automation*, Vol. 2: 663-668, April, 1988.

[KEL89] Kelmar, L. "Manipulator Servo Level World Modeling," NIST Technical Note 1258, NIST, Gaithersburg, MD, March, 1989.

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

[LAN76] Landau, L.D., Lifshitz, E.M., *Mechanics*, Pergamon Press, Oxford, 1976.

[LAU89] Laugier, C., "Planning fine motion strategies by reasoning in the contact space," *Proceedings 1989 IEEE International Conference on Robotics and Automation*, Vol. 2: 653-659, May, 1989.

[MAC88] Maciejewski, A.A., Klein, C.A., "Numerical Filtering for the Operation of Robotic Manipulators through Kinematically Singular Configurations," *Journal of Robotic Systems*, 5(6), 527-552, December, 1988.

[MAP86] Maples, J.A., Becker, J.J., "Experiments in Force Control of Robotic Manipulators," *Proceedings 1986 IEEE International Conference on Robotics and Automation*, Vol.2: 695-702, April, 1986.

[MAS81] Mason, M.T., "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 6, June, 1981.

[MER87] Merlet, J-P., "C-surface Applied to the Design of an Hybrid Force-position Robot Controller," *Proceedings 1987 IEEE International Conference on Robotics and Automation*, Vol. 2, 1055-1059, March 1987.

[MOR85] Mortenson, M.E., *Geometric Modeling*, John Wiley & Sons, Inc., 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.

[NEW89] Newman, W.S., "Automatic Obstacle Avoidance at High Speeds via Reflex Control," *Proceedings 1989 IEEE International Conference on Robotics and Automation*, Vol. 2: 1104-1109, May, 1989.

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

[PES85] Peshkin, M.A., Sanderson, A.C., "The Motion of a Pushed, Sliding Object, Part 1: Sliding Friction," Technical Report CMU-RI-TR-85-18, Carnegie Mellon University, 1985.

[PES88] Peshkin, M.A., "Programmed Compliance for Error Correction in Robotic Assembly," *Proceedings of the 3rd IEEE International Symposium on Intelligent Contro*l, August, 1988.

[PIE68] Pieper, D., *The Kinematics of Manipulators Under Computer Control*, Ph.D. Thesis, Stanford University, 1968.

[RAI81] Raibert, M.H., Craig, J.J., "Hybrid Position/Force Control of Manipulators," *Journal of Dynamic Systems, Measurement and Control*, Vol. 102: 126-133, June, 1981.

[SAW89] Sawada, C. et al., "Specification and Generation of a Motion Path for Compliant Motion," *Proceedings 1989 IEEE International Conference on Robotics and Automation*, Vol. 2: 808-815, May, 1989.

[SER89] Seraji, H., "Configuration Control of Redundant Manipulators: Theory and Implementation," *IEEE Transactions on Robotics and Automation*, pp. 472-490, August, 1989.

[SHI86] Shin, K.G., McKay, N.D., "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators," *IEEE Transactions on Automatic Contro*l, Vol. AC-30, No. 6, June, 1986.

[SKA87] Skaar, S.B., Brockman, W.H., Hanson, R., "Camera Space Manipulation," *The International Journal of Robotics Research*, Vol. 6, No. 4, Winter, 1987.

[TRI87] Trinkle, J.C., Abel, J.M., Paul, R.P., "Enveloping, Frictionless, Planar Grasping," *Proceedings 1987 IEEE International Conference on Robotics and Automation*, April, 1987.

[WAV88] Wavering, A. "Manipulator Primitive Level Task Decomposition," NIST Technical Note 1256, NIST, Gaithersburg, MD, October 1988.

[WEI87] Weiss, L.E., Sanderson, A.C., Neuman, C.P., "Dynamic Sensor-Based Control of Robots with Visual Feedback," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 5, October, 1987.

[WES85] West, H., Asada, H., "A Method for the Design of Hybrid Position/Force Controllers for Manipulators Constrained by Contact with the Environment," in 1985 *IEEE International Conference on Robotics and Automation*, pp. 251-259, March, 1985.

[WHI82] Whitney, D.E., "Quasi-Static Assembly of Compliantly Supported Rigid Parts," *Journal of Dynamic Systems, Measurement, and Control*, Vo1.104: 65-77, March, 1982.

[WHI87] Whitney, D.E., "Historical Perspective and State of the Art in Robot Force Control," *The International Journal of Robotics Research*, Vol. 6, No. 1, Spring, 1987.

[WOL87] Wolovich, W.A., *Robotics: Basic Analysis and Design*, Holt, Rinehart and Winston, New York, 1987.

[XIA89] Xiao, J., Volz, R.A., "On Replanning for Assembly Tasks Using Robots in the Presence of Uncertainties," *Proceedings 1989 IEEE International Conference on Robotics and Automation*, Vol. 2: 638-645, May, 1989.

[YOS84] Yoshikawa, T., "Analysis and Control of Robot Manipulators with Redundancy," *Robotics Research: The 1st International Symposium*, pp. 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.

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NIST/TN-1273	2. Performing Organ. Report No.	3. Publication Date December 1989

4. TITLE AND SUBTITLE

Manipulator Primitive 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 a World Modeling module at the second 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 second level of the control system, the Primitive Level World Modeling module supports the Level 2 Sensory Processing module and the Primitive Task Decomposition module. This document contains detailed descriptions of the interfaces between the module and the rest of the control hierarchy. It also discusses the function of the support processes within the module. 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	14. NO. OF PRINTED PAGES
[✓ XX] Unlimited	36
☐ For Official Distribution. Do Not Release to NTIS	15. Price
[✓ XX] Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.	
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161	