PROCEEDINGS
Of SPIE-The International Society for Optical Engineering

Volume 851

Space Station Automation III

Wun C. Chiou, Sr.
Chair/Editor

Sponsored by
SPIE—The International Society for Optical Engineering

Cooperating Organizations
American Association for Artificial Intelligence
IEEE Robotics and Automation Council
National Aeronautics and Space Administration

2-4 November 1987
Cambridge, Massachusetts

Servo Level Algorithms for the NASREM Telerobot Control System Architecture

John C. Fiala, Ronald Lumia, James S. Albus

Robot Systems Division
National Bureau of Standards
Gaithersburg, MD. 20899

ABSTRACT

The NASA/NBS Standard Reference Model (NASREM) Telerobot Control System Architecture defines a logical computing architecture for robotics. The architecture provides a framework for integrating a variety of control techniques, and for combining teleoperation and autonomy in one system. This paper demonstrates these aspects of NASREM for the lowest level of the architecture, the Servo Level. The Servo Level supports algorithms for robot manipulator control found in the literature.

1. INTRODUCTION

The NASA/NBS Standard Reference Model (NASREM) Telerobot Control System Architecture, as presented in Albus, et. al.[1], defines the basic architecture for a robot control system capable of teleoperation and autonomy in one system. Recently, efforts have been directed at specifying in detail the architecture requirements for robotic manipulation. An important criterion for the design is that it support the algorithms for manipulator control found in the literature. This assures that the control system can serve as a vehicle for evaluating algorithms and comparing approaches.

Section 2 of this paper briefly presents the concepts of the NASREM Architecture for robotic manipulation. This is followed by a detailed discussion of the components of the architecture at the Servo Level in Section 3. The Servo Level interfaces support many algorithms found in the literature. Section 4 gives examples of Servo Level algorithms for autonomous manipulator control. Section 5 discusses teleoperator control algorithms.

2. NASREM ARCHITECTURE

The fundamental structure of the architecture is depicted in Figure 1. The system consists primarily of a three-legged hierarchy of computing modules. These computational elements are supported by a common memory for shared data and an operator interface for interacting with a supervisory human operator. The computational modules carry out sensory processing, world modeling and task decomposition activities for the control of a robot.

The sensory processing (G) modules obtain sensory information from sensors. The modules filter and integrate the information for use in the world model. The world modeling (M) modules update internal information on the state of the robot and its environment, providing the task decomposition hierarchy with the current "best estimate" of model related quantities. The task decomposition hierarchy is the part of the system most directly involved in control. The H modules decompose the high-level goals into low-level actions. Within these modules, there are submodules concerned with planning and executing.

The Task Level is the highest NASREM control level concerned solely with a single autonomous robot. The Task Level receives commands from the Service Bay Level. Each command is decomposed into a sequence of "elemental manipulations" understandable to E-move. To achieve this decomposition, the Task Level planning module may use artificial intelligence techniques to plan autonomously, or use prespecified plans entered by an operator.

The E-move Level receives commands from Task that represent "elemental manipulations," such as MOVE <object> TO <destination>. E-move plans all aspects of the manipulation. This includes finding a collision-free path and determining any fine-motion strategies. As at the Task Level, part of E-move's decomposition could be obtained from prespecified plans. E-move outputs path segments executable by the Primitive Level.

The Primitive Level is responsible for generating the time sequence of desired manipulator state vectors needed to produce a dynamic trajectory from the E-move command. Primitive determines the behavior of the manipulator on a time scale between that of E-move and Servo. It transforms path segments into dynamic sequences of path points directly executable by Servo.

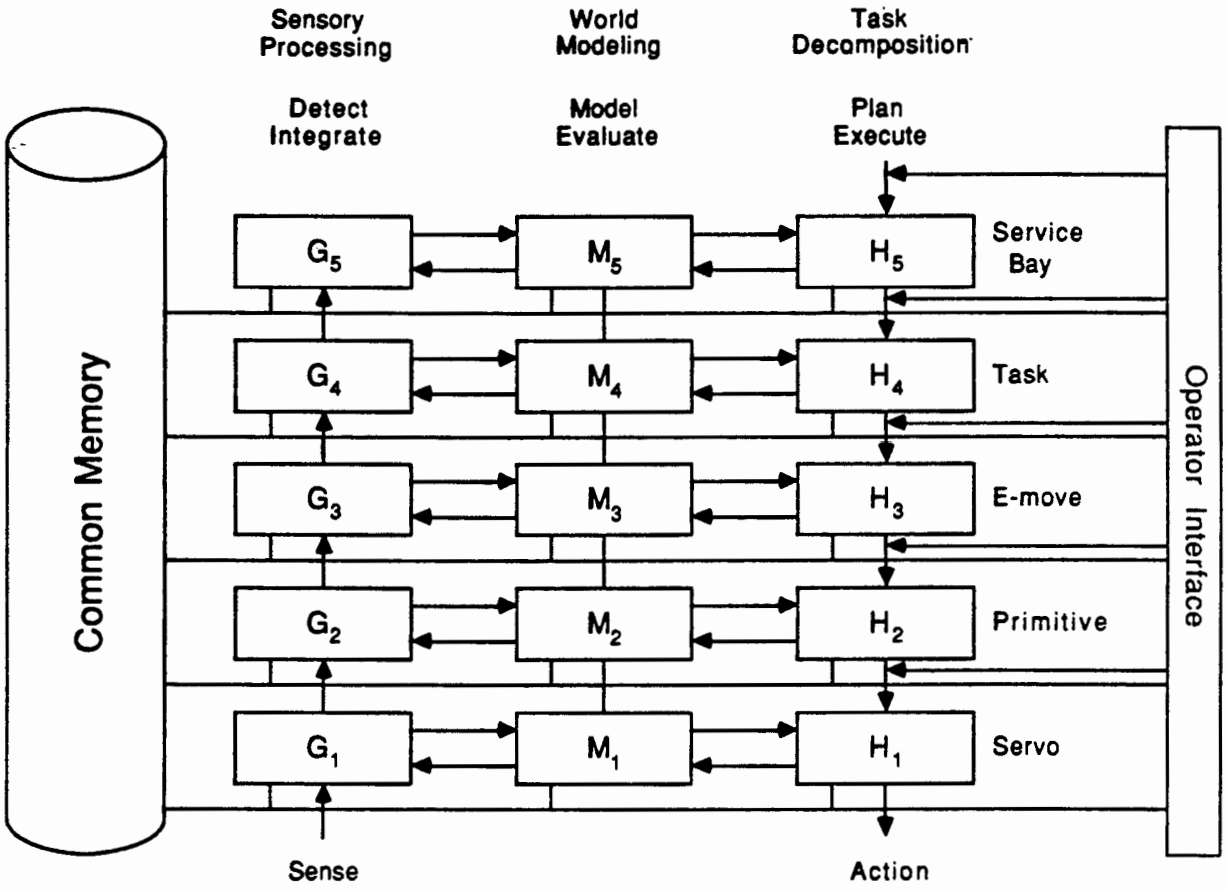

Figure 1: NASA/NBS Control System Architecture.

The Servo Level, the lowest level of the task decomposition hierarchy, computes the motor control signals for the actuators. The job of the Servo Level is to handle motion small in a dynamic sense. The level executes a specified algorithm for approaching the desired manipulator state vectors (or attractor set.) The attractor set is typically a point in a complex trajectory computed by Primitive. Primitive updates the attractor set, and possibly the algorithm, periodically.

Although between levels in the hierarchy there are well-defined interfaces, these interfaces are general enough to support a number of different algorithms at a level. This is discussed in Sections 4 and 5 for the Servo Level.

3. SERVO LEVEL COMPONENTS

The task decomposition (control) module at the Servo Level receives input from Primitive in the form of the command specification parameters shown in Figure 2. The command parameters include a coordinate system specification C_z which indicates the coordinate system in which the current command is to be executed. C_z can specify joint, end-effector, or Cartesian (world) coordinates. Given with respect to this coordinate system are desired position, velocity, and acceleration vectors ($\mathcal{Z}_d, \dot{\mathcal{Z}}_d, \ddot{\mathcal{Z}}_d$) for the manipulator, and the desired force and rate of change of force vectors ($F_d, \dot{F}_d$). These command vectors form the attractor set for the manipulator. The parameter R of the command specification indicates the method of manipulator joint redundancy resolution. The K's are the gain coefficient matrices for error terms in the control equations. The selection matrices (S,S') apply to certain hybrid position/force control algorithms. Finally, the "Algorithm" specifier selects the control algorithm to be executed by the Servo Level. The function of the command parameters is clarified by examples in Sections 4 and 5.

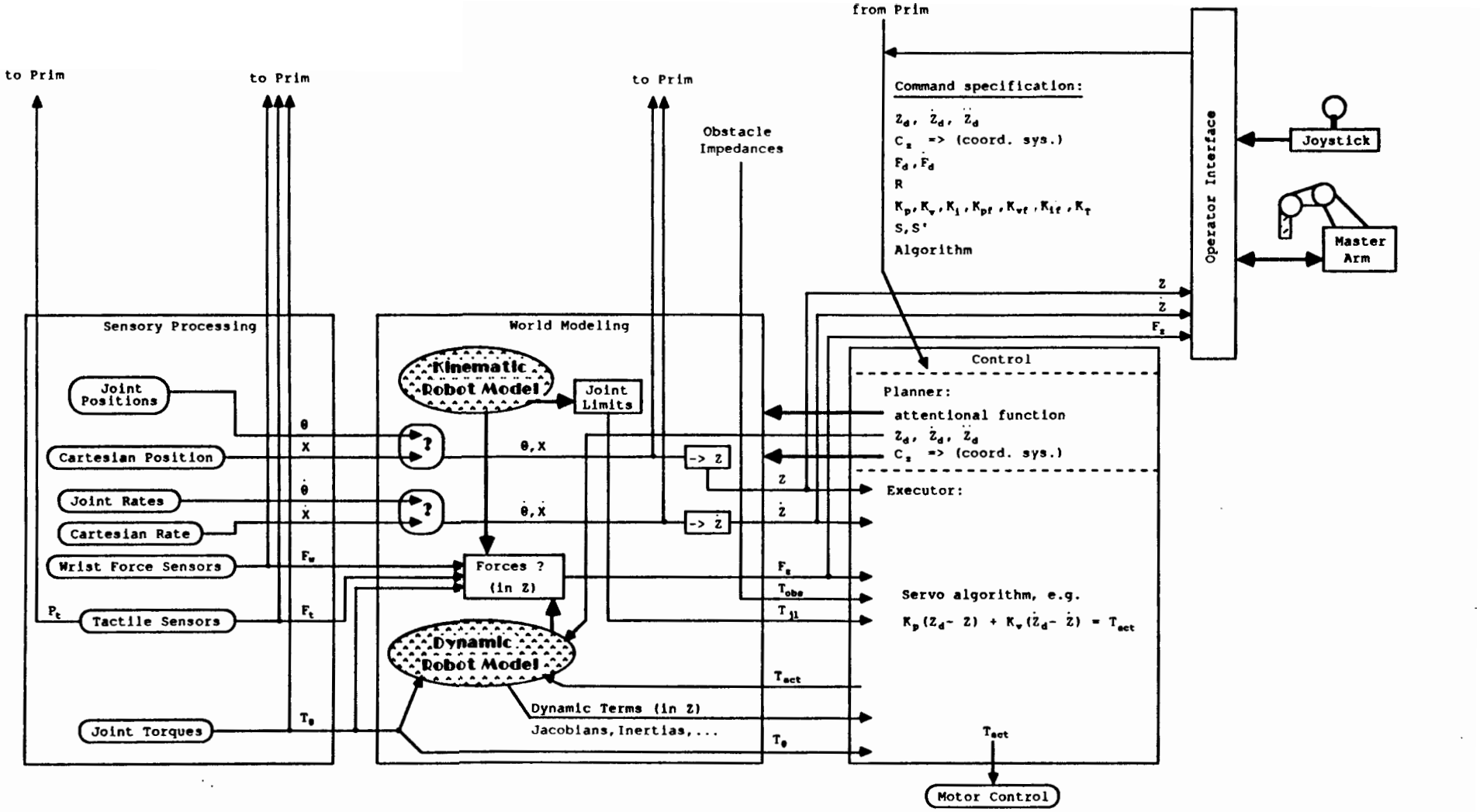

Figure 2: Servo Level Architecture.

When the Servo Level planner receives a new command specification, the planner transmits certain information to world modeling. This information includes an attentional function which tells world modeling where to concentrate its efforts, i.e. what information to compute for the executor. The executor simply executes the algorithm indicated in the command specification, using data supplied by world modeling as needed.

The world modeling module at the Servo Level computes model-based quantities for the executor, such as Jacobians, inertia matrices, gravity compensations, Coriolis and centifugal force compensations, and potential field (obstacle) compensations. In addition, world modeling provides its best guess of the state of the manipulator in terms of positions, velocities, end-effector forces, and joint torques. To do this, the module may have to resolve conflicts between sensor data, such as between joint position and Cartesian position sensors.

Sensory processing, as shown in the figure, reads sensors relevant to Servo and provides the filtered sensor readings to world modeling. In addition, certain information is transmitted up to the Primitive Level of the sensory processing hierarchy. Primitive uses this information, as well as information from Servo Level world modeling, to monitor execution of its trajectory. Based on this data, Primitive adjusts the stiffness (gains) of the control, or switches control algorithms altogether. For example, when Primitive detects a contact with a surface, it may switch Servo to a control algorithm that accommodates contact forces.

4. AUTONOMOUS CONTROL

The Servo Level supports many algorithms for autonomous manipulator control. Some examples of algorithms for manipulator control from the literature supported by the NASREM Servo Level are given below.

To support computed torque control[2], a coordinate system specification of C_z = (joint) can be chosen. Thus, the position vectors Z_d and Z represent desired joint positions and sensed joint positions, respectively. With the appropriate dynamic parameters obtained from world modeling, the control algorithm in the Servo executor is

$$M(\theta)[K_p(Z_d-Z) + K_v(\dot{Z}_d-\dot{Z}) + \ddot{Z}_d] + T_{cent.}(\dot{\theta},\theta) + T_{gravity}(\theta) = T_{act}.$$

For hybrid position/force control[5], C_z = (Cartesian) is chosen, along with a control

$$K_pJ^{-1}(\theta)S'(Z_d-Z) + K_vJ^{-1}(\theta)S'(\dot{Z}_d-\dot{Z}) + K_i\int J^{-1}(\theta)S'(Z_d-Z) + J^t(\theta)SF_d + K_{pf}J^t(\theta)S(F_d-F_z) + K_{if}\int LMT[J^t(\theta)S(F_d-F_z)] = T_{act}.$$

For this control scheme the S and S' selection matrices are used to choose between force or position control on each axis. The function LMT() is a threshold function for limiting the integral feedback term.

Other types of control available with the Servo Level include stiffness control[8], given by

$$C_z = \text{(Cartesian)}$$
$$K_pJ^{-1}(\theta)(Z_d - Z - K_{pf}F_z) - K_vJ^{-1}(\theta)\dot{Z} = T_{act},$$

and operational space control[4], given by

$$C_z = \text{(Cartesian)}$$
$$J^t(\theta)\{ S[F_d + K_{pf}(F_d-F_z)] + \Lambda(\theta)SK_{vf}\dot{Z} + \Lambda(\theta)S'[K_p(Z_d-Z) + K_v(\dot{Z}_d-\dot{Z}) + \ddot{Z}_d] + \mu(\dot{\theta},\theta) \} + g(\theta) = T_{act}.$$

In the operational space control algorithm the S and S' matrices play the role of Khatib's task specification matrices[4], instead of binary selection matrices[5].

5. TELEOPERATOR CONTROL

Teleoperation of a manipulator can be achieved by using the task decomposition interface of the autonomous system as the interface to the operator control system. As shown in Figure 2, the operator interface sends commands to Servo using the same command specification used by Primitive. The operator interface must receive feedback information from the Servo Level in order to provide force reflection to the operator and to determine when the Servo manipulator comes in contact with the environment. Thus, force F_d, and position and velocity $(Z,\dot{Z})$ are obtained from the world modeling/control interface of the Servo Level. This feedback

information is sufficient for most algorithms.

The operator can control the Servo manipulator with any of several devices. These include joystick type devices, such as DFVLR's sensor ball[3]. The operator may also use a "master arm" to control the manipulator, in which case the Servo arm could be referred to as a "slave arm"[6,7]. When a joystick input device is used, the operator is unable to receive force feedback through the device. However, the operator should be able to control the amount of force exerted by the manipulator. This can be done by allowing the joystick input to be interpreted as force commands when the robot is contacting the environment, and as motion commands otherwise[3]. Thus, the operator is controlling the robot with a combined position/force scheme[5]. This control scheme is most useful when the commanded motions and forces are specified in a frame related to the task.

For joystick control using the combined position/force scheme in a constraint frame fixed to the end-effector, a coordinate system selection of C_z = (end-effector, T_e) would be made. In this case, the T_e would give the transformation to a frame at the tip of the workpiece, as shown in Figure 3. The control algorithm for the Servo executor would look something like

$$K_v {}^{e}J^{-1}(\theta)S'(\dot{Z}_d-\dot{Z}) + {}^{e}J^{t}(\theta)SF_d + K_{pf}{}^{e}J^{t}(\theta)S(F_d-F_z) = T_{act}.$$

For "master-slave" teleoperatiom, several forms of control are possible. The simplest possibility is to have the slave arm track the master arm positions.

$$K_p(Z_d-Z) + K_v(\dot{Z}_d-\dot{Z}) = T_{slave}$$

would be the control with C_z = (joint).

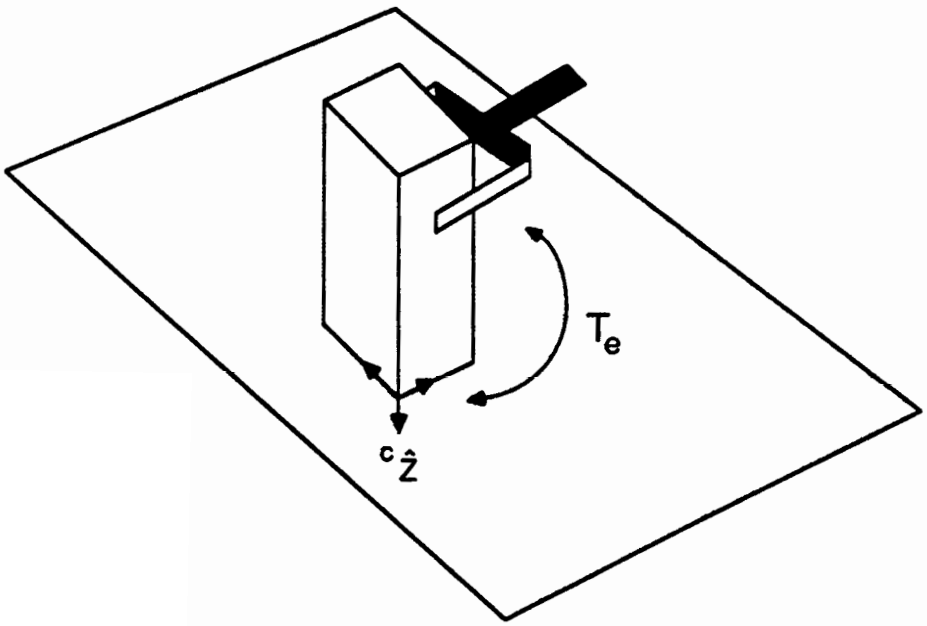

Figure 3: Constraint Frame Coordinates.

This can be done with position feedback to the master arm to provide the operator with a better "feel" for the work being done[6,7]. Thus, the above control is used along with a control

$$K_{p_m}(Z-Z_d) + K_{v_m}(\dot{Z}-\dot{Z}_d) = T_{master} ,$$

on the master arm. (The master arm control is handled by the operator interface.) Here the assumption is made that both arms have the same kinematics.

Another type of bilateral control is to let the torques on each arm produce corresponding torques on the other[6], i.e.

$$C_z = (\text{joint})$$
$$K_{pf_s}(F_d-F_z) = T_{slave} ,$$
$$K_{pf_m}(F_z-F_d) = T_{master} .$$

A third type of control is to have the position error drive the slave arm and the force difference drive the master arm[6]. This yields

$$K_{p_s}(Z_d-Z) = T_{slave} ,$$
$$K_{pf_m}(F_z-F_d) = T_{master} ,$$

where F_d, although not used in the command to Servo, represents the master arm torques.

6. CONCLUSION

The NASREM Architecture appears to provide an excellent framework for manipulator control. The interfaces between the subsystems can support a wide variety of control algorithms, as demonstrated with the Servo Level. In fact, the controller supports most of the low-level control techniques from the literature. One of the principle benefits of the architecture is that it provides a nice environment in which to implement and compare algorithms.

REFERENCES

1. Albus, J. S., McCain, H. G., Lumia, R., NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), NASA Document SS-GSFC-0027, Dec. 4, 1986.
2. Craig, J. J., Introduction to Robotics: Mechanics and Control, Addison-Wesley, Reading, Mass., 1986.
3. Hirzinger, G., "The Space and Telerobotic Concepts of DFVLR ROTEX," IEEE Conf. Robotics and Automation, Raleigh, N.C., March, 1987.
4. Khatib, O., "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," IEEE Jour. Robotics and Automation, Vol. RA-3, No. 1, Feb., 1987.
5. Raibert, M. H., Craig, J. J., "Hybrid Position/Force Control of Manipulators," Transactions of the ASME, Vol. 102, June, 1981.
6. Thring, M. W., Robots and Telechirs, Ellis Horwood Ltd., Chichester, England, 1983.
7. Vertut, J., Coiffet, P., Teleoperations and Robotics: Evolution and Development, Prentice-Hall, Englewood Cliffs, N.J., 1986.
8. Whitney, D. E., "Historical Perspective and State of the Art in Robot Force Control," IEEE Conf. Robotics and Automation, St. Louis, March, 1985.