Journal of Intelligent and Robotic Systems **5**: 39–47, 1992.

Sensory Interactive Robot Trajectory Control Using a Real-Time World Model

L. KELMAR
United Parcel Service, Danbury, Connecticut, U.S.A.

and

R. LUMIA
Robotics Systems Division, National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899, U.S.A.

(Received: 10 September 1990; revised: 12 February 1991)

Abstract. A major consideration in the design of sensory interactive trajectory generation software for a Flight Telerobotic Servicer (FTS) is the availability and maintenance of a current model of the manipulator's world. The NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), which has been adopted by NASA for control of the FTS, provides a logical computing architecture for telerobotics. It defines a hierarchical control system in which complex tasks are decomposed into progressively simpler subtasks, or objectives. It contains a hierarchy of world modeling modules which maintain the system's internal model of the manipulator and its world by continuously updating the model based upon sensory data. Each world modeling module contains support processes or functions which simultaneously and asynchronously support sensory processing and task decomposition. This paper discusses the role of world modeling in support of trajectory generation and execution. For sensory interactive robot motion control, the world modeling modules must operate rapidly so that the model of the world remains in registration with the real world.

This paper discusses the world modeling modules of a hierarchical control system which facilitate sensory interactive trajectories by decoupling and supporting the sensory and manipulator planning processes. We define the types of information which should be included in the interfaces to the modules, as well as the modules' structure and function. Finally, we discuss the real-time implementation and experimental results of a particular sensory interactive algorithm performed in our laboratory.

Key words. Robotics, image processing, real-time control, space applications, control systems, hierarchical control, telerobotics.

1. Introduction

This paper describes the interfaces, structure, and function of the world modeling modules which support trajectory generation for a hierarchical control system for telerobots. As shown in Figure 1, the NASREM telerobot control architecture [1] contains separate hierarchies of sensory processing, world modeling, and task decomposition modules. Sensory processing acquires and processes information about the state of the world, which may include data from position, force, tactile, vision, range, and other sensors. The task decomposition hierarchy decomposes high-level goals into simpler and simpler subgoals, and accomplishes them by planning and executing appropriate actions. The world model consists of modeling processes and a global

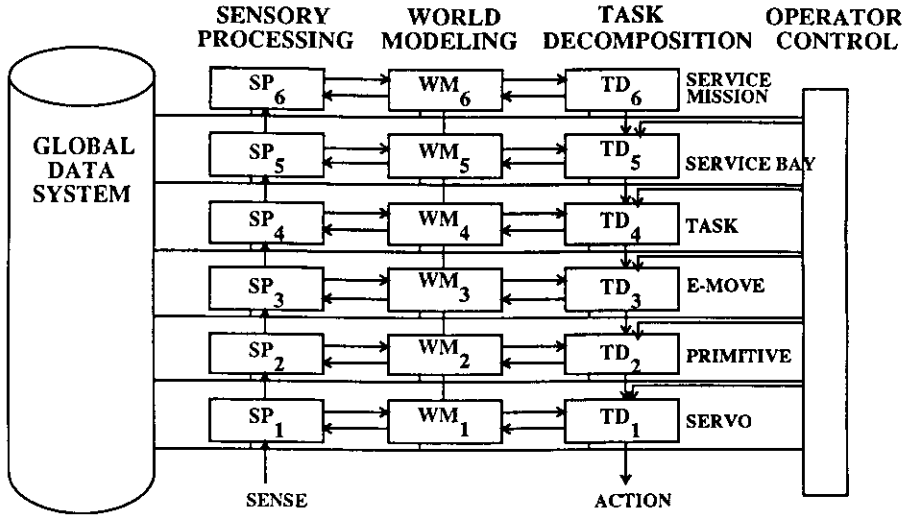

Fig. 1. A hierarchical telerobot control system architecture.

data system; it serves as an intermediary between the sensory processing and task decomposition hierarchies. The world modeling processes perform calculations based on manipulator, object, and environment models to provide estimates of current and future world states. The operator control provides a means by which a human operator can access status information, supervise, and directly control the telerobot at any level of the system.

We are implementing the NASREM control architecture in our laboratory. Our telerobot system consists of a manipulator and a camera. The processing modules, for this subset of the NASREM architecture, can be recombined according to their function in the system, as shown in Figure 2. The system 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 the camera; the manipulation 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 perception and manipulation branches cooperate during trajectory generation and execution.

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 [6].

Real-time trajectory generation 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

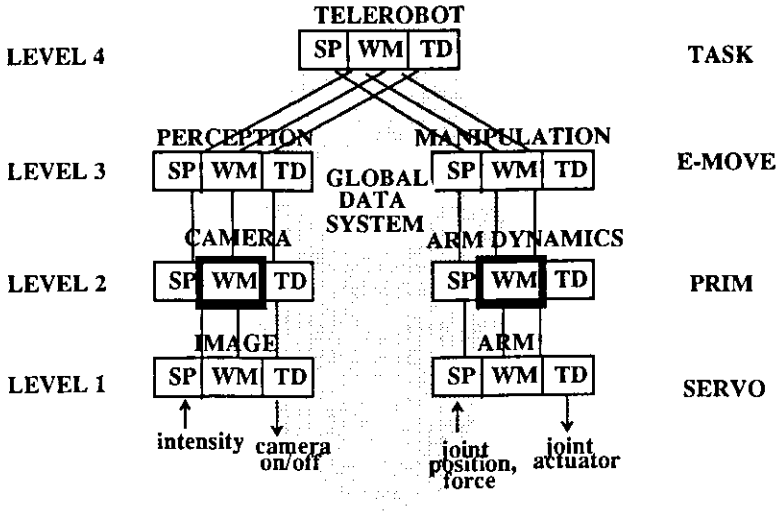

Fig. 2. Perception and manipulation branches for a NASREM sub-system.

motions on the difference between the desired, or reference, joint positions and the actual ones. Because of possible errors in the manipulator kinematic model and/or the locations of object in the workspace, the relative position of the end-effector and the object may be in error. With the addition of perceptual sensory feedback, the control system can more reliably relate the relative pose of the manipulator's end-effector to objects in the world.

The discussion in this paper focuses on the world modeling modules highlighted in Figure 2: one which supports the manipulator trajectory generation module (Prim) and one which supports visual perception. In the following sections, we characterize the algorithm information passed to the support modules and the corresponding modeling information which is returned. Algorithm independent interfaces, which allow for both pre-planned and sensory interactive trajectories, will be defined. We also introduce the role of the operator interface for initializing perception processing. Section 5 discusses a particular sensory interactive algorithm executing on our system and shows how the modules defined accommodate the algorithm.

2. World Modeling to Manipulator Prim Task Decomposition Interface

To be able to perform tasks, FTS manipulators must be controlled by a motion generation system which enables the specification, planning, and execution of a wide range of dynamic trajectories. The task decomposition servo and trajectory generation software modules most affect these behaviors. Together, the modules define what types of trajectories, or large dynamic motions, the manipulators can perform (trajectory generation), and how well the manipulator will be able to perform these motions (servo) [5].

The Prim task decomposition module generates and executes plans for dynamic trajectories for a manipulator [6]. 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. For example, when using vision data in real time to perform a trajectory toward a moving object, the path that the manipulator follows in space is determined as the trajectory is being performed. In such cases, it may be more appropriate to simply command as goal state which defines what is to be achieved, along with an algorithm specification that defines how to achieve it. These types of algorithms perform what is referred to here as sensory-interactive trajectory generation. The world modeling modules provides the necessary connection between the manipulator and the perception sensory processing.

The goal of this section is to characterize the algorithm information computed and maintained by the Prim world modeling module. The information listed below is not specific to any one algorithm; it defines the module which supports a wide range of trajectory generation algorithms [4]. We begin with support for the Prim in planning trajectories.

- *Forward and Inverse Kinematics* – The manipulator configuration specification may be the desired Cartesian end-effector pose or 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.
- *Current (Actual) Arm Position and Velocity* – Prim 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.
- *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 it from exceeding actuator limitations. When the manipulator carries an object, the dynamic model should reflect the additional mass.
- *Gain Information* – Prim requires gain information for various motions. The information may be constant and stored in the global data system, or may be adjusted based upon experimental or run-time results. Often, the choice of gains is strictly task dependent and is maintained Prim.
- *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 suplied by the world modeling support module depends on the algorithms implemented. 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 position and velocity of the object(s) of interest, as well as any tolerances and kinematic constraints for mating objects.

During execution of a trajectory, world modeling supplies Prim with various sensed values. The exact sensory information passed from world modeling to Prim 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 data used by Prim enable manipulation when *a-priori* information is not sufficiently recent or precise to complete the task. Sensor data may include:

- *Input Device Information* – For teleoperated control, the manipulator trajectory is guided by an input device, such as a master or 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.
- *Object or Feature Position (relative displacement)* – Often the precise location of an object cannot be known with sufficient accuracy prior to execution of the task. For such situations, sensory feedback during task execution 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.

Each World Modeling module also interfaces to Sensory Processing. The Manipulator Prim World Modeling module interfaces to the Manipulator Level 2 Sensory Processing module, as shown in Figure 2. 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; it is filtered and stored at Level 1.

3. World Modeling to Level 2 Perception Sensory Processing Interface

The Level 2 sensory processing module for image data extracts features from iconic information received from Level 1 and converts the features to symbolic representations. The types of features include edges, corner points, centroids, and moments. The features are entered into the global data system to be used by a manipulator control module, by higher level sensory processing modules for scene interpretation,

or by both. World modeling maintains a history of recent readings which it uses for filtering out spurious readings. The world modeling support module also provides predictions of expected features. The predictions may be based upon the histories of readings, the knowledge of the camera's location with respect to the object of interest, as well as knowledge of how the object and/or camera is moving.

For example, consider the task of tracking a ball moving on a planar surface. In one approach, Level 1 image sensory processing acquires two sequential images and subtracts them. The resulting iconic representation, of the difference in intensities between the images, represents the area of motion. Level 2 sensory processing computes the centroid of the difference image. If the ball is the only object which changes position between successive images, then the algorithm successfully locates the ball. However, if the integrity of the scene cannot be guaranteed, a model should be employed to isolate the motion of the ball from possible motion of other objects. World modeling supports sensory processing by providing a window of interest around the expected location of the ball in each image. The predictions are based upon previous reading(s) and knowledge about the movement of the objects between sampling instants.

4. World Modeling to Operator Interface

The operator interface provides a means by which human operators can observe, supervise, and directly control the system [1]. The interface includes such input devices as a joystick, a mouse, and a keyboard. Operator interaction may be used in the absence of a complete world model. For example, the operator can indicate safe pathways or the loation of an object to be manipulated. The transformation from the manipulator or world reference frame to the object of interest would be computed and stored in the world model. Similarly, the operator can assist the perception processing modules, especially during initialization of a task. Consider the task of tracking an object. The object, or features, of interest can be indicated via the operator interface. Thereafter, tracking the object can be accomplished autonomously by sensory processing (with windowing and predictive support from world modeling).

5. Current System Setup and Experimental Results

In this section we discuss an initial task performed in our laboratory involving the lowest two levels of the NASREM control hierarchy. The demonstration successfully integrates the sensory processing, world modeling and task decomposition modules, thus closing the sensory feedback loop. In the experiment, a small ball is released at the top of an inclined board which contains randomly placed pegs, as shown in Figure 3. As the ball rolls down the board it is tracked by the single camera vision system and then caught by the manipulator at the bottom of the board. Our manipulator is a seven degree of freedom arm. The vision processing is performed by a real-time pipelined image processing machine, PIPE (Pipelined Image Processing

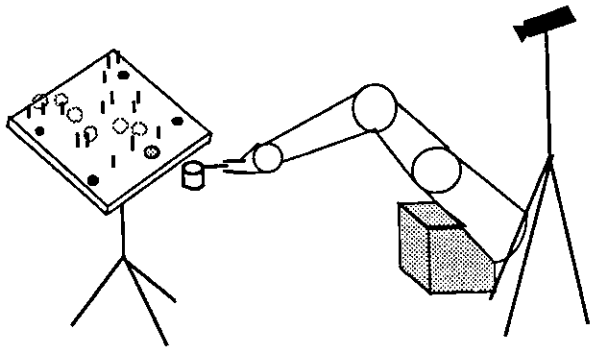

Fig. 3. Ball catching task setup.

Engine), which is commercially available through Aspex, Incorporated.* The software is written in Ada and runs on 68020 processors. This section details the role of the world modeling modules in closing the feedback loop between sensory processing and manipulation; Figure 4 shows the world modeling computations for the task.

The operator interface facilitates initialization of the system. The board is placed within the workspace of the manipulator. Its location is then determined by "teaching" the position of three corners of the board with the manipulator. From the three points, the position and orientation of the board, with respect to the manipulator, are computed. The camera is placed at the start of the task; the only constraint on its placement is that the entire face of the board must be in the field of view. The camera's location is determined using a four coplanar point algorithm [7]. The transformations from camera to board and from manipulator to board are stored in the world model. After initialization, all computations are performed by cyclically executing processes which communicate via global read-write interfaces [2].

The vision algorithm used for the experiment determines the location of the ball using the difference of successive images. The PIPE machine acquires successive images every 1/30 second. After smoothing the images, PIPE differences the entire images in one cycle. The resulting differenced image is thresholded and then passed to Level 2 for additional processing. At Level 2, the centroid of the area in the differenced image is computed for use by Prim world modeling. Currently, world modeling does not supply a window of interest around the expected location of the ball.

The centroid value, as supplied by sensory processing, is a two-dimensional centroid in image space; Prim requires a three-dimensional position with respect to the manipulator. World Modeling performs several computations to transform the two-dimensional centroid data to a value usable by Prim. First the location of the ball's centroid in the image is represented as a three-dimensional point on the camera's image plane. The ball's position on the inclined board is computed by projecting a ray from the camera's optical center through the point on the image plane. The

* Products named in this paper are listed for purposes of information only. There is no implied endorsement of any products or implication that they are the best available for the purpose.

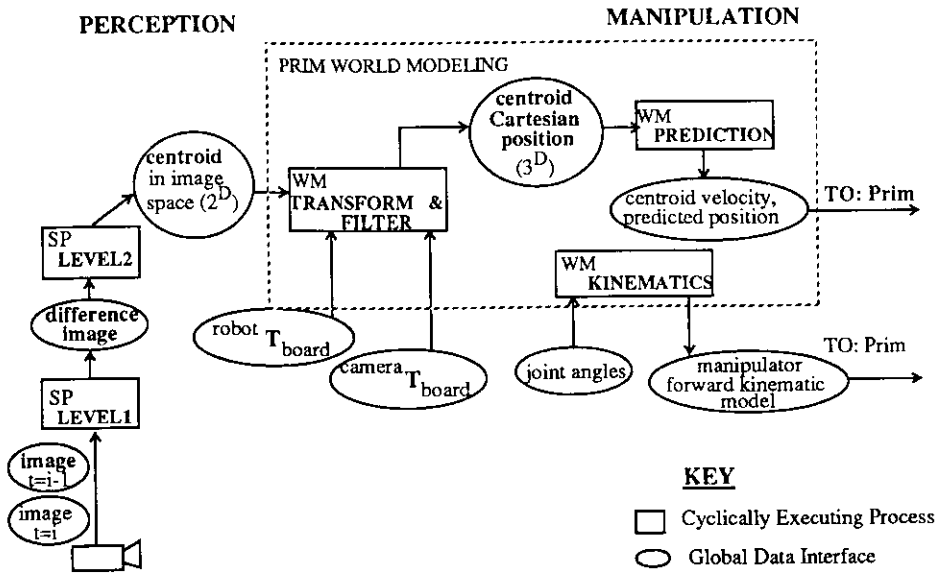

Fig. 4. Level 2 world modeling support for ball catching task.

intersection of the ray with the inclined board represents the ball's location. World modeling then filters out spurious or invalid data points corresponding to locations beyond the boundaries of the inclined board. Finally, world modeling transforms the point to a convenient coordinate system for Prim, using the pre-stored transform.

The actual position of the ball, as computed from the image, is of little use to Prim. In order to catch the ball, the manipulator must move toward predicted locations of the ball. The algorithm world modeling uses to predict the location of the ball extrapolates linearly from the current and the last readings. No attempt is made to model the ball's collisions with the pegs on the board. Thus, the first-order predicted ball position ($X_{t=i+1}$) sent to Prim is computed from the two most recent ball locations (X-ball) by

$$X\text{-ball}_{t=i+1} = X\text{-ball}_{t=i} + (X\text{-ball}_{t=i} - X\text{-ball}_{t=i-1}),$$

where t represents the sampling instant. The Prim algorithm computes the projection of the (predicted) ball position on the base of the board. As the ball nears the bottom of the board, Prim also must determine how far out from the bottom of the board to catch the ball. The faster the ball is rolling, the further out from the board it must be caught. Prim sends the desired location (X_d), as well as the position and velocity control gains (K_p, K_v) for the move to Servo. The gains are adjusted to maintain the smoothness of the servo algorithm, as well as to increase the response of the manipulator as the ball nears the board's bottom.

The manipulator Servo level computes the control torques based upon errors in Cartesian space according to [3]

$$\tau = J^T[K_p(X_d - X) - K_v(\dot{X})] + \tau_{\text{gravity}} + \tau_{\text{friction}},$$

where $\tau_{gravity}$ and $\tau_{friction}$ are computed from models of the gravity and friction forces, respectively, acting upon the manipulator in any given configuration. Upon sending the control torques to the manipulator, Servo completes the sensory feedback loop. Thus we demonstrate the integration of the sensory processing and task decomposition hierarchies via world modeling.

6. Conclusions

This paper 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. Algorithm independent interfaces, which allow for both pre-planned and sensory interactive trajectories, have been defined. Finally, we demonstrated how a sensory interactive trajectory generation algorithm is accommodated by the modules' structure and interfaces.

References

1. Albus, J.S., McCain, H.G., and Lumia, R., 1987, NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM), NASA Document SS-GSFC-0027.
2. Fiala, J., 1989, Note on NASREM implementation, NIST, Gaithersburg, MD.
3. Fiala, J. and Wavering, A., 1990, Implementation of a Jacobian-transpose algorithm, ICG Note # 23, NIST, Gaithersburg, MD.
4. Kelmar, L., 1989, Manipulator primitive level world modeling, NIST Technical Note 1273, NIST, Gaithersburg, MD.
5. Lumia, R. and Wavering, A., 1989, Trajectory generation for space telerobots, *Conference on Space Telerobotics, Pasadena, California*, Jan 30.
6. Wavering, A., 1988, Manipulator primitive level task decomposition, NIST Technical Note 1256, NIST, Gaithersburg, MD.
7. Yeh, P.S., Barash, S., and Wysocki, E., 1988, A vision system for safe robot operation, *Proc. 1988 IEEE Internat. Conference on Robotics and Automation*, April 24–29, pp. 1461–1465.