Proceedings of NASA Conference on Space Telerobotics
Pasadena, CA, Jan. 31-Feb. 2, 1989.

Trajectory Generation for Space Telerobots

R. Lumia, A. J. Wavering

National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

The purpose of this paper is to review a variety of trajectory generation techniques which may be applied to space telerobots and to identify problems which need to be addressed in future telerobot motion control systems. As a starting point for the development of motion generation systems for space telerobots, the operation and limitations of traditional path-oriented trajectory generation approaches are discussed. This discussion leads to a description of more advanced techniques which have been demonstrated in research laboratories, and their potential applicability to space telerobots. Examples of this work include systems that incorporate sensory-interactive motion capability and optimal motion planning. Additional considerations which need to be addressed for motion control of a space telerobot are described, such as redundancy resolution and the description and generation of constrained and multi-armed cooperative motions. A task decomposition module for a hierarchical telerobot control system which will serve as a testbed for trajectory generation approaches which address these issues is also discussed briefly.

1. Introduction

The Flight Telerobotic Servicer (FTS) has been conceived as a device which will be able to take the place of an extravehicular crew member to perform such tasks as truss assembly, changeout of orbital replacement units (ORUs), electrical and fluid connector coupling and uncoupling, and solar cell array cleaning [27]. To be able to perform these tasks, FTS manipulators must be controlled by a motion generation system which enables the specification, planning, and execution of a wide range of dynamic behaviors. The task decomposition elements of a hierarchical control system which most affect the types of behaviors possible are the servo and trajectory generation software modules. Together, these 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). This paper is concerned primarily with identifying the considerations involved in generating the larger dynamic motions for space telerobots. The preliminary design of a trajectory generation software module for a hierarchical control system is also presented.

2. Trajectory Generation Considerations

A natural place to start in the development of trajectory generation software for space telerobots is to examine how the problem is approached for terrestrial manipulators. In this section, important aspects of the description and planning of different types of motions are described. Most of these considerations have emerged from work on earth-based systems, but special implications of operation in the space environment are presented where appropriate.

Path Constraints

The most conceptually straightforward, though not the only way to describe a desired manipulator motion is to explicitly define the path which the manipulator should follow through space. Virtually all commercial manipulators use positional paths to specify motions. Paths may usually be specified in either joint space or Cartesian space, and may be represented as equations which are functions of a dimensionless path parameter s. The path parameter s indicates the fraction of the path traversed. The motion to perform a given path is created by planning a smooth trajectory function for the path. The trajectory function determines the manipulator position, velocity, and acceleration at any and all times for the duration of the trajectory. Functions for smooth transitions between trajectory segments may also be planned. The trajectory function is evaluated periodically during execution, which results in a sequence of closely-spaced goal points. These goal points are typically sent to individual joint position servo processes, whose task it is to ensure that the

manipulator follow the commanded trajectory as closely as possible. Current trajectory generation techniques which operate in this manner are discussed in detail in [4], [5], and [14]. This type of trajectory planning is sometimes referred to as "constraint satisfaction" [4]. It will be seen that there are other constraints in addition to a position path constraint, which may need to be observed in generating manipulator trajectories.

The path constraint itself is not really an absolute one. Cartesian motion, for example, cannot be performed exactly (for articulated manipulators), since Cartesian goal points must be transformed pointwise into joint coordinates to send to the individual joint servos. If a sufficient number of closely-spaced points are used, a reasonably close approximation to the desired Cartesian path will result. Often, however, it is good enough if a trajectory can be generated which will lie within some position tolerance of the desired path. The tolerance may, in fact, be quite large; particularly for intermediate points in a multi-segment trajectory. If it is specified explicitly, this path tolerance may be used to advantage to simplify the planning of Cartesian straight-line trajectories and to generate more efficient movements. Taylor recognized this, and devised the bounded deviation strategy as an approach to performing Cartesian trajectories [21]. Given a Cartesian straight-line path and an allowable deviation, the bounded deviation strategy will determine the number of points to be traversed in joint space which will keep the trajectory within the stated bounds. Path tolerance can also be seen as an additional freedom in trajectory planning which allows the nominal path to be modified slightly to better achieve the desired objective. The use of path tolerance in creating minimum-time trajectories, for example, is discussed by Suh and Bishop [20].

In planning the trajectory function, conservative limits on manipulator velocity and acceleration are commonly used to prevent planning a motion which cannot be performed due to actuator torque or force limitations. Except in some experimental laboratory systems, the manipulator and payload dynamics are not taken into account during trajectory planning. If the system dynamics are not considered, it is not possible to achieve maximum manipulator performance. This aspect of trajectory planning is discussed in the following section.

Manipulator and Payload Dynamics

The dynamics of the manipulator and payload, along with joint saturation characteristics and environmental interaction forces, determine the achievable motions of the manipulator. For free space motions, dynamics considerations are most crucial when it is desired to extract the maximum performance from a manipulator, as in planning time-optimal motions. In this case the usual conservative limits on path acceleration and velocity are too restrictive. There has been considerable interest recently in developing algorithms that, given a parameterized path specification, will determine the time sequence of torques required to travel the path in mimimum time or some other optimal fashion ([18], for example). These algorithms use the manipulator dynamics and actuator torque constraints in computing the optimal trajectory. The dynamics are usually reformulated in terms of the path parameter variable s and its derivative, rather than joint variables. An optimization technique (dynamic programming, for example) is then used to minimize or maximize the desired performance index subject to the constraints imposed by actuator limitations and manipulator dynamics. An interesting research issue in itself is to identify appropriate objective functions to use for generating trajectories for motions required to perform FTS tasks.

The ping-pong playing robot of Andersson takes manipulator dynamics into account in a somewhat different fashion; the dynamics are used to determine the achievability of postulated motions [2]. That is, a motion is planned to move to a particular state in a certain amount of time, which may or may not be possible. If the planned trajectory is determined to be infeasible, a new motion is postulated, based on some indication of why the former plan could not be performed. This approach has the advantage of reduced computational requirements, since it is only necessary to determine feasibility, not optimality.

The zero-gravity characteristic of the space environment has significant impact on manipulator dynamics, with resulting implications for trajectory planning. Most obvious is that the lack of gravity means that a manipulator does not have to exert actuator torque to support the weight of the manipulator and payload. This implies that there is additional actuator torque available for performing motions, and that there is no explicit payload limitation as with terrestrial manipulators. If the manipulator base is securely attached to a vehicle of sufficient inertia, the limitation is on how quickly objects may be moved around, rather than on the size and mass of the objects. Indeed, several proposed FTS tasks, such as truss assembly and certain ORU changeouts (see [27]), involve handling objects with large masses and/or rotational inertias. This implies that for certain tasks, at least, the payload inertia must be included in the computation of the manipulator dynamics used in trajectory planning and verification.

The lack of gravity also results in an environment in which all objects, including the vehicle or platform

to which the FTS is attached, are free-floating. Since manipulator motions result in forces and torques being transmitted through the base to the supporting object, the supporting object will move in response to manipulator motions unless maneuvering jets or reaction wheels are used to counteract the base forces. This issue is addressed by Vafa and Dubowsky [22]. The implications for manipulator trajectory planning are: 1) the manipulator workspace with respect to the supporting vehicle may be reduced, 2) manipulator motions can cause an undesired change in vehicle attitude, and 3) manipulator motions can produce accelerations which may disrupt micro-gravity experiments on the supporting vehicle. Vafa and Dubowsky approach analysis of the effects of free-floating manipulator configurations through application of a "virtual manipulator," which is a massless arrangement of links which originate from the inertial reference frame located at the system center of mass. The link lengths are determined by the original configuration of the actual manipulator. Once the virtual manipulator has been defined, motions of the actual manipulator are planned by determining the motions of the virtual manipulator which would be required to perform the motion.

Real-time Sensory Information

Although they may be useful for some free space motions, positional path specifications are not appropriate for all types of manipulator actions. There are many situations where it may not be possible or appropriate to define the desired path of the manipulator a priori. 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 a 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*. Sensory interactive capabilities will be required for the FTS to perform such tasks as docking with spinning satellities and manipulating objects with a large amount of locational uncertainty (which may be due to shifting during launch or the flexible nature of the object, for example).

There are several possibilities for incorporating vision information into trajectory formation. One may use carefully placed and calibrated cameras to try to locate the target object with respect to the world frame, for example. The calibration requirement of this approach is a major disadvantage, however, since any disturbance of the camera setup will require recalibration. Alternatively, the possibility of using *relative* position differences in camera space to guide motions has been investigated recently [19]. In this approach, the joint position of the manipulator which will achieve the desired camera space relationship of end effector and target object features is repeatedly estimated as the movement is performed. This approach eliminates the reliance on camera calibration and is more robust to camera disturbances.

Another question regarding sensory-interactive motions is how they should be planned and executed. Sensory interaction may be accomplished either by frequent replanning of the trajectory as it is executed or by planning a trajectory function which represents a general profile of the desired motion, with the trajectory details produced dynamically as the trajectory is executed. The first approach has been demonstrated by Andersson for real-time trajectory generation for the ping-pong playing robot [2]. A potential problem with the replanning approach is that there may be a discrepancy between the estimate of what the state of the system will be at the end of planning (which is used as the initial state for the plan) and the actual state at that time. This may result in non-smooth transitions between plans, as the servo module attempts to correct the error. In the approach of determining a general profile, planning may still occur, but it is limited to determining general characteristics of the trajectory implicitly, rather than specifying all aspects explicitly. A set of trajectory parameters is planned which is used to compute the portion of the distance to the goal which should be moved at each cycle. An example of this approach is presented by Myers et al. [12].

Smoothness

It was mentioned previously that the trajectory functions computed for a path should be "smooth." That is, the resulting manipulator motion should not be jerky. This is particularly true for manipulators in space, since excessive jerkiness can excite strucural resonances of both the manipulator arm and the supporting structure, resulting in decreased accuracy and possible instability. Jerky motions also cause accelerated wear of mechanical components. One measure of smoothness is the mean squared magnitude of the rate of change of acceleration (jerk) [9]. Trajectory functions can be planned which give very smooth motion when executed in conjunction with a position servo controller. A position function which is a quintic polynomial of time, for example, results in minimum-jerk motion [9]. However, as stated above, if there is a difference between the estimated initial conditions used to plan the motion and the actual conditions at the beginning of the motion, the corrective servo motions which result will be jerky. An interesting alternative is to use a constant position goal and modulate the gains of the servo module instead of updating the position goals and using

constant gains [7]. If proper gain functions are chosen, the resulting trajectory will always be smooth, and there is no need to preplan position and velocity functions explicitly. Although further analysis and experiments need to be done to assess the usefulness and stability of this technique in actual manipulator applications, it is also interesting in terms of its apparent similarity to some human movements.

Closed Kinematic Chains

Many FTS tasks require contact with another manipulator or other object in the environment, resulting in the formation of a closed kinematic chain which is capable of sustaining internal forces. Some examples of such tasks are truss assembly, tasks where two arms are used to manipulate a single object, and coordinated motions of fingers of a dextrous hand when an object is grasped. The FTS trajectory generation module must be able to plan and execute trajectories for such constrained motions.

One approach for performing constrained actions is to use position-controlled motion along with carefully-engineered passive compliance to guide the motion and prevent excessive interaction forces from being generated [25]. Trajectory planning for constrained motions which use a passive compliance device is based on following a positional path, as described above. This approach has been applied successfully in industrial applications and results in high-speed assembly capabilities, but suffers the drawback of requiring a device which is rather task-specific.

Work has been done to develop appropriate servo control techniques which are more general than a physical device for performing constrained motions. Hybrid position/force control [15], for example, is a control approach which allows the specification of desired position along unconstrained degrees of freedom, and desired forces along those directions which are subject to a position constraint. Given that this type of servo control is available, one has to be able to specify the desired position and force paths and generate the corresponding trajectories which will accomplish a particular task. This has been done primarily for relatively simple tasks and constraint situations, and additional work needs to be done if this approach is to be applicable to more complex tasks. An additional problem is robust on-line determination of the actual constraints the manipulator is subject to during trajectory execution. Some progress has also been made in this area (see [11], for example).

An alternative to the hybrid position/force control approach is to modulate the relationship between manipulator position and force, or the apparent impedance of the manipulator [8]. The first attempts at control of this nature were subsets of generalized impedance control, and include active stiffness control [16] and generalized damper control [26]. An attractive aspect of the impedance control approach is that it represents a step in the direction of approaching contact with the environment as an acceptable, commonplace, and necessary occurrence, rather than as an exceptional circumstance. Trajectory generation for impedance-controlled motions usually consists of planning and executing a nominal position trajectory and controlling the manipulator gains to modulate the impedance, allowing the controlled stiffness, damping, and inertial characteristics to prevent excessive interaction forces during contact. In the case of generalized damper control, a nominal motion direction, rather than path, is used.

Studies of dual-arm cooperative motion controls have also started with identifying useful servo control techniques [17]. Again, standard trajectory generation approaches are typically used with these techniques. The additional problem of synchronization appears when two arms are to perform a coordinated motion. There are two possibilities for two-arm trajectory generation. The first is that each arm has a separate trajectory generation module. The trajectories for each arm are planned independently (although in the same manner), and must be synchronized via some external variable during execution. The other possibility is to have a single trajectory generation module which plans trajectories for both arms at the same time and executes them simultaneously as well. While this approach is straightforward in terms of coordination, there is no spatial decomposition of the task—all of the planning must be performed by a single module, even when the arms are operating independently.

Several important questions regarding constrained motion trajectories remain. What type of static representation for the desired task is most appropriate? How much knowledge of the details of object kinematics and physical properties such mass, stiffness, and friction coefficient, is needed to plan and execute a trajectory that will accomplish the task? Is there a representation for the task and an approach for performing such actions that simplifies the planning required?

Kinematic Redundancy

Another important aspect of trajectory planning is determining how to most effectively use extra kinematic degrees of freedom which result when the kinematic freedoms of the manipulator exceed those

required by the given task. Extra degrees of freedom may be used to avoid singular manipulator configurations, avoid obstacles, distribute torque requirements more evenly among the joints, and achieve similar motion subgoals, while still allowing the end effector to follow a prescribed path. For space applications, an additional useful possibility might be to use redundant degrees of freedom, or self motion, to control base reaction forces and disturbances to the supporting vehicle. The manipulators for the FTS will have redundant degree(s) of freedom with respect to a six degree-of-freedom positioning task, and kinematically-redundant commercial manipulators have recently been introduced. Redundancy also occurs when a six degree-of-freedom arm is equipped with a dextrous multi-fingered hand. Some means of distributing the desired motion between the hand and the arm is required.

One possibility for resolving redundancy is to use global optimization techniques to determine the joint positions or torques for an entire trajectory as in [13]. Nakamura's approach makes use of Pontryagin's Maximum Principle to determine how the extra degree(s) of freedom may best be used over an entire trajectory to globally optimize a specified performance index. Although this approach results in global optimization, it is extremely computationally intensive and therefore useful primarily for off-line computation.

An alternative to resolving redundancy globally over an entire trajectory is to perform local kinematic inversion for points along a Cartesian trajectory as the motion is being executed. In this case, redundancy resolution takes place during the execution, rather than the planning, of a trajectory. Although they do not result in global optimization of any objective function, there are local redundancy resolution techniques which may be performed sufficiently fast as to be used on-line during trajectory execution. Reference [3] presents a survey of a number of these techniques.

Operator Interaction

Not only must the control system be able to accomplish tasks autonomously, but it must also be able to accept operator input at all levels to allow teleoperation and execution of operator commands. Most of what is usually thought of as teleoperation—that is, direct manipulator control by an operator—is performed by the servo module, rather the trajectory generation module. For teleoperation, the operator performs trajectory generation directly by manipulating a master arm or other control device. Interaction with the trajectory generation module consists primarily of entering static motion commands of the same type which are sent to the trajectory generation module when the system is operating autonomously. This gives the operator the capability of indicating a desired path or goal state, and allowing the trajectory generation module to plan and execute the desired motion. In addition to this type of interaction, it is also desirable to give the operator an overriding control of the manipulator velocity, so that the operator can slow down or stop the manipulator at any time during trajectory execution.

3. Trajectory Generation Software Module Design

The aspects of trajectory generation outlined above have been considered in the design of a trajectory generation module for a hierarchical manipulator control system. The overall framework of such a system for autonomous and teleoperated telerobot control is described in [1]. The trajectory generation module is part of the *task decomposition* hierarchy, which subdivides high-level tasks into simpler and simpler subtasks. There are also separate hierarchies which perform *sensory processing* and *world modeling* functions. World and manipulator state information and communication interfaces are contained in a *global data system*, available to all processes. The reader is referred to [1] for additional details on these parts of the system. The decomposition performed by the trajectory generation module is to generate a time sequence of closely-spaced manipulator goal states from a static description of the desired motion. As such, it generates primitive trajectories, and is called the Primitive (Prim) task decomposition module.

During autonomous operation, Prim receives commands from the Elemental Move (E-move) level of the task decomposition hierarchy, which performs such functions as grasp planning and obstacle-free path planning. The Prim module can also accept commands from the Operator Control. The output commands of the E-move level are time-independent descriptions of motions, for example static position or position and force paths. In addition to these types of commands, E-move can also simply specify a set of termination conditions, or goal states, along with an algorithm specification which determines the strategy to be used achieve them. This type of specification is useful with sensory interactive trajectory algorithms, where the exact path is determined *as* the motion is performed, based on sensed data. Commands entered by the operator through the Operator Control contain the same information and have the same format as those which come from E-move.

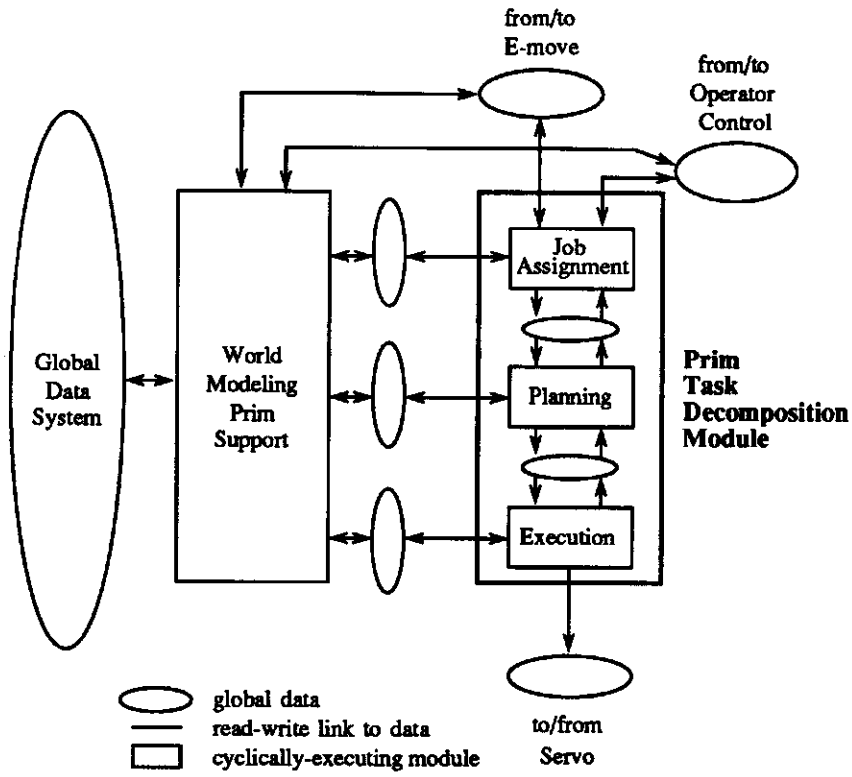

Figure 1. Prim Task Decomposition Structure.

Prim generates the time sequence of attractor sets needed to produce a dynamic trajectory from the E-move or Operator Control command, and sends these as commands to the Servo level of the task decomposition hierarchy. The Servo level, the lowest level in the task decomposition hierarchy, controls the behavior of the manipulator in performing small motions between closely-spaced goals. The function and interfaces of a Servo level for manipulators which accommodates a broad spectrum of published control algorithms is described in [6]. In addition to determining position, velocity, acceleration, and/or force trajectories to be commanded to Servo, Prim also has the task of determining appropriate manipulator impedance, stiffness, damping, and inertial characteristics which are controlled by adjusting the servo loop gains commanded to Servo.

Structure

As illustrated in Figure 1, the Primitive task decomposition module is composed of three concurrent, cyclically-executing processes; the Job Assignment module, the Planning module, and the Execution module. These processes execute independently of one another and have different cycle times, with the Execution module typically operating at a much higher rate than the Job Assignment and Planning modules. Each of these submodules continually repeats a cycle which consists of reading inputs, performing computations, and writing outputs. This type of operation prevents the entire system from locking up if a single process hangs during an attempt to compute or communicate. Such freedom from system lock-up is an essential feature for safe and reliable manipulator control. The functions of the three submodules are discussed below.

The Job Assignment module coordinates transitions between autonomous and operator control through management of the input command queue. The input commands to Prim are queued so that future commands will be available to the Planning module in order to plan transitions between motion segments. When the operator wishes to take control at the Prim level, he or she may do so by editing the queue to insert and delete commands.

The Prim Planning module handles the generation of a plan for a dynamic trajectory (for example, time

functions of manipulator position, velocity, acceleration, and/or force). It is important to realize, however, that the terms "trajectory planning" and "trajectory generation algorithm" as used here do not always refer to methods of preplanning the exact position, velocity, and acceleration of the manipulator at every instant during the motion. Instead, the Planning module may only determine functions or parameters which determine what the general profile should look like for the motion, and the exact path the robot takes during execution is determined by sensory or other external inputs. For planning all types of motions, the Prim Planning module looks ahead by an amount of time which depends on several factors, including the length of the motion itself, the trajectory generation algorithm used, and the nature of the objective function to be optimized during the motion. The Prim planning horizon will typically be on the order of 100 ms, although it may be as much as one to a several seconds.

In addition to planning a suitable trajectory, the Planning module must also select an appropriate servo algorithm and servo loop gains. The position, force, and torque servo loop gains determine the behavior of the manipulator in response to new position and force goals, and to external disturbances. The Prim Planning module may determine the apparent manipulator impedance by adjusting these gains. A constant set of gains that works reasonably well for a variety of manipulator configurations and payloads might be used for free space moves. Such gains result in compromised performance, however, since the manipulator dynamics are configuration-dependent. Alternatively, gains may be varied as a function of the manipulator state by the Execution module during trajectory execution.

Another task performed by the Planning module is redundancy resolution when a global optimization technique is used to transform a six degree-of-freedom path specification into a seven or more degree-of-freedom joint trajectory. The Planning module also determines the intervals of time for which the position and force trajectory functions should be evaluated.

The Prim Execution module has two primary functions. First, it must evaluate the position, velocity, acceleration, jerk, force, and time derivative of force functions of time for the intervals specified by the Planning module. This results in the point attractor vectors which are sent as commands to the Servo level. In addition, the Execution module is responsible for monitoring position, velocity, force, and other sensor states or world model conditions for achievement of the termination conditions. The Execution module must also monitor the commands it sends to Servo to make sure they are achievable. This includes making sure that excessive joint velocities are commanded, even if the manipulator is near a singular configuration. Also, the resolution of redundancy, if it is performed by kinematic criteria, is performed by the Execution module. Furthermore, the Execution module incorporates operator velocity control and single-step interactions. The Execution module also should calculate the estimated termination time for motions when this is possible.

Information about the state of the manipulator and the world is needed to perform planning and execution tasks. This information is provided to the task decomposition module via the world modeling support module shown in Figure 2. This world modeling support module contains processes which access data stored in the global data system by the sensory processing side of the control hierarchy, and perform model-based computations (such as manipulator kinematics and dynamics). The Prim world modeling support module may access information which has been processed by any level of the sensory processing hierarchy. Note that this implies there may not be a direct correspondence between sensory processing levels and task decomposition levels.

Interfaces

An attempt has been made to identify the types of information which should be included in the Prim command and status interfaces to allow the implementation of trajectory generation algorithms which take into account the considerations presented in Section 2. The Prim input command and output status interface information is given in Table 1. The command specification consists of an algorithm and a set of parameters. The parameters which have been included represent commonly-used means of describing manipulator motions in a time-independent manner, and expressing what factors are important in transforming the command into a dynamic movement. The interface parameters are discussed in detail in [23,24].

Implementation

An implementation of the trajectory generation module described above is currently being developed in Ada. Clearly, the development of such a module with all of the desired capabilities is quite a formidable task. The approach which has been taken is to implement a skeletal module which contains the desired concurrent functional submodules (Job Assignment, Planning, and Execution), and interface variables. A small number of simple trajectory generation algorithms are being implemented initially, along with basic

Table 1. E-move to Primitive Interface Elements.

Primitive Input Command Elements	Primitive Ouput Status Elements
Command number	Job Assignment status
Prim algorithm	Planning command number
Coordinate system	Planning status
Position command description	Execution command number
Force command description	Execution status basis
Held object	Estimated termination time
Destination object	
Termination condition(s)	
Redundancy resolution specification	
Priority	
Objective function	

world modeling functions. Building upon this foundation, additional algorithms and capabilities will be added. The mapping of the system architecture into computing hardware for this implementation is discussed in [10].

4. Conclusions

The capabilities desired of the FTS place rather severe demands on the control system trajectory generation module. Although significant progress has been made in developing advanced techniques for trajectory generation, continued work is needed in the areas of task representation, constrained motion generation, redundant manipulator control, coordination of multiple arms with dextrous end effectors, and vision servoing. Further investigations addressing the use of manipulator dynamics in trajectory generation are also needed. Although not discussed in this paper, advances must also occur in the techniques used in the sensory processing and world modeling hierarchies to provide the information needed for advanced trajectory generation.

This paper has not addressed the hardware and software requirements imposed if all of the considerations discussed are to be included. The requirements are quite formidable, however, and a logical approach is to develop first a basic structure which allows for the implementation of many different algorithms of varying complexities. One may then start with straightforward, proven algorithms and proceed to add capabilities as improvements in algorithms and computing hardware come about. This is the approach which has been taken at NIST for motion control system development.

5. 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, December 4, 1986.

[2] Andersson, R. L., "Agressive Trajectory Generator for a Robot Ping-Pong Player," Proc. IEEE Conf. Robotics and Automation, Philadelphia, PA, April 1988.

[3] Baillieul, J., et al., "Kinematically Redundant Manipulators," Proc. NASA/JPL Workshop on Space Telerobotics, Pasedena, CA, January 1987.

[4] Brady, M., et al., eds., Robot Motion: Planning and Control, MIT Press, Cambridge, Mass., 1982.

[5] Craig, J. J., Introduction to Robotics: Mechanics and Control, Addison-Wesley, Reading, Mass., 1986.

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

[7] Fiala, J. C., "Generation of Smooth Trajectories without Planning," submitted to 1989 IEEE Int. Conf. on Robotics and Automation.

[8] Hogan, N., "Impedance Control: An Approach to Manipulation," Journal of Dyn, Sys., Meas., and Control, March 1985.

[9] Hogan, N., "An Organizing Principle for a Class of Voluntary Movements," Journal of Neuroscience, Vol. 4, No. 11, November 1984.

[10] Lumia, R., Fiala, J. C., "The FTS: From Functional Architecture to Computer Architecture," submitted to 1989 NASA/JPL Workshop on Space Telerobotics.

[11] Merlet, J-P., "C-surface applied to the design of an Hybrid Force-Position Robot Controller," Proc. IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, March 1987.

[12] Myers, D. R., Leake, S. A., Juberts, M., "Control System Architecture for Telemanipulator Operation," in Recent Trends in Robotics: Modeling, Control, and Education, M. Jamshidi, J. Y. S. Luh, M. Shahinpoor, eds., North-Holland, New York, NY, 1986.

[13] Nakamura, Y., Hanafusa, H., Yoshikawa, T., "Task Priority Based Redundancy Control of Robot Manipulators," Inter. Journal of Robotics Research, Vol. 6, No. 2, Summer 1987.

[14] Paul, R. P., Robot Manipulators: Mathematics, Programming, and Control, MIT Press, Cambridge, MA, 1981.

[15] Raibert, M. H., Craig, J. J., "Hybrid Position/Force Control of Manipulators," Transactions of the ASME, Vol. 102, June 1981.

[16] Salisbury, J. K., "Active Stiffness Control of a Manipulator in Cartesian Coordinates," Proc. 19th IEEE Conf. Decision and Control, Albuquerque, NM, December 1980.

[17] Seraji, H., "Adaptive Hybrid Control of Manipulators," Proc. NASA/JPL Workshop on Space Telerobotics, Pasedena, CA, January 1987.

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

[19] Skaar, S. B., Brockman, W. H., Hanson, R., "Camera Space Manipulation," Inter. Journal of Robotics Research, Vol. 6, No. 4, 1987.

[20] Suh, S. H., Bishop, A. B., "The Tube Concept and its Application to the Obstacle-Avoiding Minimum-Time Trajectory Planning Problem," Proc. IEEE Int. Conf. on Sys., Man, and Cybernetics, Alexandria, VA, October 1987.

[21] Taylor, R. H., "Planning and Execution of Straight Line Manipulator Trajectories," IBM J. Res. Develop., Vol. 23, No. 4, 1979.

[22] Vafa, Z., Dubowsky, S., "On the Dynamics of Manipulators in Space Using the Virtual Manipulator Approach," Proc. IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, March 1987.

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

[24] Wavering, A. J., Lumia, R., "Task Decomposition Module for Telerobot Trajectory Generation," Proc. SPIE Space Station Automation IV Conference, Cambridge, MA, November 1988.

[25] Whitney, D. E., Nevins, J. L., "What is the RCC and What Can It Do?," Proc. 9th Int. Symp. on Industrial Robots, March 1979.

[26] Whitney, D. E., "Force Feedback of Manipulator Fine Motions," Journal of Dynamic Sys., Meas., and Control, December 1982.

[27] Zimmerman, W., Myers, J., Ruth, D., "Task Ranking for the Telerobot Demonstration Final Report," JPL Contract 957908, July 1988.