The invention is a method of operating a robot in successive sampling intervals to perform a task, the robot having joints and joint actuators with actuator control loops, by decomposing the task into behavior forces, accelerations, velocities and positions of the joint actuators for the current sampling interval from both behavior forces, accelerations velocities and positions of the joint actuators for the current sampling interval, computing actuator accelerations of the joint actuators for the current sampling interval from the actuator velocities and positions of the previous sampling interval, and, finally, controlling the actuators in accordance with the actuator accelerations, velocities and positions of the current sampling interval and actuator velocities and positions of the previous sampling interval, computing actuator accelerations of the joint actuators for the current sampling interval from the actuator velocities and positions of the previous sampling interval, and, finally, controlling the actuators in accordance with the actuator accelerations, velocities and positions of the current sampling interval and actuator velocities and positions of the previous sampling interval.

31 Claims, 10 Drawing Sheets
OTHER PUBLICATIONS


**FIG. 2**

- **Basic Commands**
- **Behavior Kinematic Quantity Processor**
- **Motion Acceleration Processor**
- **Velocity Processor**
- **Joint Servo Loops**
- **Robot Sensors**
FIG. 3

START

40

RECEIVE AN APPLICATION COMMAND

MAP APPLICATION COMMAND TO SIMULTANEOUS BEHAVIOR FORCES, ACCELERATIONS, VELOCITIES AND POSITIONS

44

TRANSFORM TO MOTION SPACE

COMPUTE COMMANDED MOTION SPACE ACCELERATIONS OF THE NEXT SAMPLE INTERVAL FROM:
(A) SIMULTANEOUS BEHAVIOR FORCES, ACCELERATIONS, VELOCITIES AND POSITIONS, AND
(B) COMMANDED MOTION SPACE VELOCITIES AND POSITIONS OF THE CURRENT SAMPLE INTERVAL

48

SENSE JOINT POSITIONS

46

TRANSFORM TO ACTUATOR SPACE

50

COMMAND ACTUATORS WITH ANY OR ALL OF THE COMMANDED ACCELERATIONS, VELOCITIES AND POSITIONS COMPUTED FOR THE CURRENT SAMPLE INTERVAL

GO TO NEXT SAMPLE INTERVAL
FIG. 6

Torque Magnitude (N-m)

Force Magnitude (N)
FIG. 9

Angle (Degrees)

Time (s)
EXTENDED TASK SPACE CONTROL FOR ROBOTIC MANIPULATORS

BACKGROUND OF THE INVENTION

1. Origin of the Invention

The invention described herein was made in the performance of work under a NASA contract, and is subject to the provisions of Public Law 96-517 (35 USC 202) in which the contractor has elected not to retain title.

2. Technical Field

The invention relates to a compliant motion control system for controlling a robot to perform specific tasks defined in terms of many simultaneous behaviors to be exhibited by the robot in different spaces in response to multiple inputs.

3. Background Art

The invention and its background will be described with reference to the following publications:

References


SUMMARY OF THE DISCLOSURE

The invention is a compliant motion control system having a control architecture which allows execution of a task to be considered as the resultant behavior of execution of multiple concurrent behaviors. The dimensionality of the execution space of each behavior and the resultant behavior can be extended to the dimensionality of the controlled mechanism. Task description consists of decomposing the desired execution into the multiple simultaneous behaviors. Each behavior generates motion commands which are merged in a common motion space to compute a resultant command to the manipulator. The task space of each behavior can have the dimensionality of the mechanism being controlled. Control of a seven degree of freedom manipulator is described here so the available task space for each behavior has dimensionality seven.

While the invention is described below as applied to control of a seven DOF manipulator, it is also applicable to other redundant and non-redundant manipulators with various numbers of DOFs.

The invention controls the robot in successive sampling intervals to perform a task, the robot having joints and joint actuators with actuator control loops. The invention decomposes the task into behavior kinematic quantities of plural behaviors to be exhibited simultaneously in respective behavior spaces of the robot, transforms the behavior kinematic quantities from the respective behavior spaces to a common space and computes, in the common space, from the behavior kinematic quantities command kinematic quantities and controls the actuators in accordance with the command kinematic quantities.

The computing of the command kinematic quantities is accomplished by computing a first command kinematic quantity for the next sampling interval from (a) the simultaneous behavior kinematic quantities and (b) other command kinematic quantities of the current sampling interval and computing the other command kinematic quantities for the next sampling interval from the first command kinematic quantity of the next sampling interval and from the other command kinematic quantities of the current sampling interval.

In a first embodiment of the invention, the first command kinematic quantity includes commanded acceleration and the other command kinematic quantities include commanded velocity and commanded position, while the behavior kinematic quantities include behavior forces, behavior accelerations, behavior velocities and behavior positions.

In a second embodiment of the invention, the first command kinematic quantity includes commanded velocity and the other kinematic quantities include commanded acceleration and commanded position.

The common space is a motion space common to which all of the behavior kinematic quantities are transformable. This requires transforming at least some of the command kinematic quantities from the motion space to an actuator space of the actuators before applying them to control the actuators.

The robot further includes joint position sensors for detecting positions of the joints and the transform to the actuator space is based upon the current angles of the joints detected by the joint position sensors.

The decomposing of the task defines the behavior kinematic quantities of each of the behaviors in respective behavior spaces of each of the behaviors, requiring transforming the behavior kinematic quantities of each respective behavior from the corresponding behavior space to the motion space. Each respective behavior space has a dimensionality corresponding to respective degrees of freedom of the robot not exceeding the maximum number of degrees of freedom of the arm. The behaviors can include trajectory control, force control singularity avoidance, joint limit avoidance, obstacle avoidance, teleoperation, dither motion and manipulability.

Where one of the behavior is avoidance of one of a joint limits, joint singularity or obstacle, the corresponding behavior kinematic quantity is a function of the proximity of the joint limit, joint singularity or obstacle, respectively. Each of the behaviors is governed in accordance with a corresponding input parameter of the type including force set points, joint singularity angle, joint limit angle, obstacle location and the like, and the input parameters are reprogrammable each sample interval, whereby to provide dynamical control the plural behaviors.

The command kinematic quantities include acceleration, velocity and position, the behavior kinematic quantities include force, acceleration, velocity and position, and the computing of the first command kinematic quantity is performed by equating a sum of proportionate differences between the command and behavior accelerations, velocities and positions, respectively, with a sum of measured forces of the robot and of behavior forces. The differences are proportionate in accordance with inertia, damping and stiffness characteristics, respectively, of the robot.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram illustrating the task decomposition into simultaneous behaviors and the computation of control commands in motion space.

FIG. 2 is a block diagram of a system architecture embodying the invention and corresponding to FIG. 1.

FIG. 3 is a flow diagram illustrating the operation of the system of FIG. 2.

FIG. 4 is a photograph of a door opening task performed by the invention.

FIG. 5 is a graph of angle as a function of time in a door opening task performed by the invention using shared control in which the solid line is the door rotation about the MOTION frame X axis (hinge axis) and the dashed line is the integrated teleoperation input angle.

FIG. 6 is a graph of force and torque magnitudes over time in the door opening task using shared control in which the solid line is the force vector magnitude in the MOTION frame and the dashed line is the torque vector magnitude in the MOTION frame.

FIG. 7 is a graph of a door opening task performed by the invention using force control, in which the solid line is the door rotation about the MOTION frame X axis (hinge axis) and the dashed line is the torque about the MOTION frame X axis.

FIG. 8 is a graph of the door opening task using shared control, in which the solid line is the force vector magnitude in the MOTION frame and the dashed line is the torque vector magnitude in the MOTION frame.
FIG. 9 is a graph of door rotation angle over time in the MOTION frame of a door opening task performed by the invention using spring control. FIG. 10 is a graph of force and torque over time of the door opening task using spring control, in which the solid line is the force vector magnitude in the MOTION frame and the dashed line is the torque vector magnitude in the MOTION frame.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Control architecture for multiple simultaneous behaviors

The control architecture for simultaneous execution of multiple control behaviors is shown in FIG. 1. The Application Space includes all potential application tasks which the robot control system must be able to accomplish. These application tasks could be sequenced together to accomplish a larger task. Execution of a given application task can be decomposed into concurrently executing behaviors. For example, a door opening task could utilize a trajectory generator to generate the nominal trajectory while force control adds small perturbations to adjust for errors between the planned trajectory and the physical system motion. The Command To Behavior Map performs the mapping between the task and the required concurrent behaviors. This could be done automatically or through interaction with an operator.

The Behavior Space includes all of the independent control behaviors. Trajectory Tracking is a control behavior which provides a trajectory generator to generate real-time trajectories. The Teleoperation behavior takes real-time operator inputs and generates control inputs. Dither generator to provide visual servoing. Other position based behaviors generate velocities which are added to the trajectory generator velocity term. Therefore, $\Sigma F_i = F_o + F_r + F_p$ (2)

where $F_o$ is the actual measured force, $F_r$ is the reference (desired) force, $F_p$ is the virtual force from the teleoperation inputs, $F_r$ is the virtual force for joint limit avoidance, and $F_p$ is the virtual force for joint singularity avoidance (or $F_{pol}$ below).

Position based behaviors generate $X_p$, $X_r$, and $X_c$. The trajectory generator generates the reference trajectory, $X_p$, $X_r$, and $X_c$. Various trajectory generators can be used. The one used for this implementation can be found in [7]. An alternative trajectory generation scheme is described below. Other position based behaviors generate velocities which are added to the trajectory generator velocity term. Therefore, $X_v = X_p + \Sigma \dot{X}_i$ (3)

For example,$\Sigma \dot{X}_i = X_{d} \dot{X}_i$ (6)

where $X_d$ is the velocity due to a dither signal and $X_v$ is the velocity generated for proximity control. An important feature of position based behaviors is that they can be generated in a Cartesian frame different than the MOTION frame. The velocities in the separate behavior frames for the position based behaviors can be transformed to equivalent velocities of the MOTION frame using rigid body kinematics.

Equation 1 is implemented with:

$$X^{\cdot\cdot\cdot} = X^{\cdot\cdot\cdot} + M^{-1}(\Sigma F - B \cdot (X^{\cdot\cdot\cdot} - X^{\cdot\cdot\cdot} - K (X^{\cdot\cdot\cdot} - X^{\cdot\cdot\cdot})))$$ (7)

where $\Delta t$ is the sample interval, and the post superscript
indicates the sample interval, i.e., n is the present sample interval and n+1 is the next sample interval. This gives the desired acceleration of the mechanism in the Motion Space. The velocity motion commands are mapped into the actuator space of the mechanism as described below.

The Actuator Space is defined as the space of active actuation of the mechanism. Mechanisms which have more than six actuator DOFs fall into two general categories, kinematically redundant and actuationally redundant. Typically, kinematically redundant mechanisms have additional behaviors associated with position and actuationally redundant manipulators have additional behaviors associated with joint force [4]. For most applications the motion space should completely span the actuator space of the manipulator to provide the widest array of behaviors for task execution. The mapping is then one-to-one and common Jacobian transpose and Jacobian inverse techniques apply. If there are more DOFs in the actuator space than in the motion space, the mapping is underconstrained and a variety of techniques can be used including pseudo-inverse or minimum kinetic energy [4]. Conversely, if there are fewer DOFs in the actuator space than the motion space, the problem is overconstrained and dumped least-squares [8, 9] and other techniques are available. Care must be taken to assure that a one-to-one mapping between motion space and actuator space does not degenerate at or near a singularity.

Although Jacobian inverse routines could be used, a damped-least squares inverse is used here to allow further task prioritization and singularity robustness [8]. The motion space velocity vector of the manipulator has three translational coordinates, three orientation coordinates and the arm angle. A composite Jacobian is formed from the individual Jacobians that relate the rate of change of the joint angles to the rate of change of the motion parameters. Here the composite Jacobian, \( J^c \), is given by [6]:

\[
J^c = \begin{bmatrix}
J^a \\
J^l
\end{bmatrix}
\]

(10)

where \( J^a \) is the angular velocity Jacobian, \( J^l \) is the linear velocity Jacobian, and \( J^o \) is the arm angle Jacobian. \( J^a \) and \( J^o \) are readily available using:

\[
\begin{bmatrix}
J^a \\
J^o
\end{bmatrix} = \begin{bmatrix}
\begin{bmatrix}
\dot{\theta}_1 \\
\vdots \\
\dot{\theta}_i \\
\vdots \\
\dot{\theta}_n
\end{bmatrix} - \begin{bmatrix}
\begin{bmatrix}
\dot{\theta}_1 \\
\vdots \\
\dot{\theta}_i \\
\vdots \\
\dot{\theta}_n
\end{bmatrix}
\begin{bmatrix}
X_{i1} \\
\vdots \\
X_{in}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\]

(11)

where \( r \) is the velocity reference point, \( \hat{z}_i \) is the \( z \) axis of joint \( i \), and \( P_{ij} \) is the position vector from the \( i \) th link frame to the velocity reference point \( r \). The arm angle Jacobian is available from [6]:

\[
J^o = \frac{(\hat{w} \times p)^T}{|p|^2} E +
\]

(12)

where \( \hat{w} = P_{0w} \), \( P_{0w} = -\hat{\psi}(\hat{w} \times c) \), \( V_p \) is the vector specifying the reference plane, \( h = (w \times \dot{V}) \times w \), \( E \) is the elbow linear velocity Jacobian, and \( W \) is the wrist linear velocity Jacobian. Notice that most of the required data for \( J^o \) is available from a by product of a forward kinematic iteration [6].

With the motion space command vector, \( \delta_{\nu} \), and the motion to actuator space map, \( J^c \) the joint servo velocity commands can be computed using damped-least squares with

\[
0_{i2} \begin{bmatrix} J^f, W_{\nu}, & 0 \end{bmatrix}^{-1} J^f, W_{\nu}, \delta_{\nu}
\]

(13)

where \( W_{\nu} \) is a diagonal task task weighting matrix that relates the relative priorities of the tasks. \( W_{\nu} \) is a diagonal velocity weighting matrix which weights the norm of joint velocities. It is important to note that while a non-zero \( W_{\nu} \) matrix will provide robustness to singularities by limiting excess joint velocities, it will also induce tracking error over the entire workspace. By setting \( W_{\nu} \) to identity and \( W_{\nu} \) to zero a standard inverse Jacobian result is provided with the same algorithm. The present implementation utilizes joint position servos so the joint velocity commands are integrated to generate the joint position commands.

Previous approaches to redundancy resolution include pseudoinverse and augmented (or complete) Jacobian methods. A wide variety of pseudoinverse approaches have utilized projection operators on the null space of the end-effector Jacobian to resolve the redundancy. These methods cannot in general guarantee cyclic or conservative motion and more importantly often cannot guarantee direct control of the entire mechanism. This often results in some unspecified internal motion of the mechanism. These methods also rely heavily on optimization functions to resolve the redundancy with only local results and often with objective functions of questionable utility. Augmented Jacobian methods focus on defining kinematic functions that provide forward and differential kinematic relationships to attempt to fully specify the motion of all degrees of freedom of the mechanism. These approaches have been fairly successful [5, 6, 11] but still have some difficulties. Often the forward kinematic relationships are difficult to compute and may not be defined over the entire workspace. Additionally the majority of these functions do not have closed form Jacobian relationships and require numerical techniques for computation. Many of the problems with artificial singularities of these non-cyclic functions and discontinuities in switching from one function or subtask have been addressed using damped least squares for position controlled applications [12, 5, 6]. However, damped least squares with joint velocity weighting introduces tracking error throughout the workspace and requires the selection of weighting matrices which is often non-intuitive. Many of these issues have not been addressed for force or impedance control applications.

The approach to control of kinematically redundant manipulators described here uses a task space of the same dimension as the robot. This is desirable from the standpoint of directly controlling all degrees of freedom and eliminating uncontrolled internal motions which can pose safety, collision, and cyclicity problems. Additionally it is desirable to make use of all available degrees of freedom to complete kinematically challenging tasks in unstructured environments that may not, be part of t, he planned suite of capabilities.

The task space parameterization does not change over time. Instead an impedance model is applied to the full dimension of the task space and shared control techniques are used to combine real and virtual sensory inputs to the impedance model. In this way, degree(s) of freedom normally referred to as redundant are unified with the task space and reference trajectories as well as sensor data cause motion of the entire mechanism. Additionally since the task space parameterization is fixed, issues of switching discontinuity do not arise and singular regions associated with the task space parameterization are fixed. The entire task space of the manipulator can be used to accomplish tasks as motion is mapped to the appropriate degrees of freedom. Thus on-line numerical techniques are not required to com-
compute differential (Jacobian) relationships. When operating in these regions potential fields may be applied to the impedance model to inhibit movement in the singular or increasingly singular directions. Alternatively, if position based impedance control is being used, the damped least squares methods can be used with the tracking error producing velocity weighting only active in the singular regions. The parameterization of redundancy discussed here generally have some physical significance, however, there also exist a wide variety of kinematic arrangements that may not have a “natural” redundancy parameter that has physical significance or spans a significant portion of the workspace. In such cases one can rightly inquire into the motivation behind these regions potential fields may be applied to the imped-


tance or spans a significant portion of the workspace. In hand side of Quation 1. This is shown in FIG. 1, 15 where m is the mass term of M, b is the damping term of B, and is the stiffness in the force controlled DOF. Either approach to control of forces is available if the difference between the reference and actual forces is added to the right hand side of Equation 1. This is shown in FIG. 1 with the difference $F_s - F_r$.

### Trajectory Generator

The trajectory generator behavior computes the reference trajectory acceleration $X_r$, velocity $X_v$, and position $X_p$. The arm angle is also generated as part of the trajectory. An alternative trajectory generation scheme has also been implemented. Here $X_p$ is the reference position, $X_r$, is set to be the desired final position. The stiffness term of the impedance equation then causes the arm to move to the destination. A slight drawback with this “spring” trajectory is that the motion accelerates quickly initially and then approaches the destination slowly. This is fixed by putting a nonlinear filter on the stiffness error, $(X_r - X_p)$. Maximum “spring lengths” corresponding to the X, Y, and Z translational components of the difference, the equivalent angle of rotation of the orientational component, and the arm angle component are given as parameters in the task command. The corresponding components of $(X_r - X_p)$ are limited to these input parameters during control.

### Joint Limit Avoidance

Joint travel limiting provides an artificial potential field at the end of travel limits on each joint. This field is then mapped to the motion space to resist operator commands that exceed joint limits. While the local site path planner can predict and avoid joint limits in its commands, often the operator using teleoperation cannot. The joint travel limiting sensor resists this motion so the operator does not induce a fault condition which would interrupt the current task.

As a joint limit is approached, a joint velocity is computed, $\theta_v$, based upon a repelling force.

$$\theta_v = -\frac{k_{bu}}{\theta_{act} - \theta_{lim}}$$

where $k_{bu}$ is the gain for joint limit $i$, $\theta_{act}$ is the actual joint angle, and $\theta_{lim}$ is the limit that the joint is approaching. The velocities corresponding to all the joints are placed in the vector $\theta$. The task space virtual forces for joint limit and joint singularity avoidance are then computed with

$$M_{kin}^{TN} F_{J} = B_j^{TN} J_{TN}^{J} \dot{\theta}$$

There are two types of Jacobians used in Equation (19), a Jacobian relating Cartesian velocities at two different frames attached to the same rigid body,

$$M^{TN} J^{TN} F^{TN}$$

and a Jacobian which relates joint space velocities to Cartesian velocities, $J^{J} F^{J}$. The $M^{TN}$ frame is the frame where Equation 1 is evaluated. $B_j$ is a damping matrix converting Cartesian velocities to forces.
joint limit avoidance behavior as a velocity input (a term of $\sum \dot{X}$ in Equation 6), the $B$ term would be removed leaving a task space velocity. A force implementation may also be used where the repelling perturbations are computed as joint space forces rather than the velocities of Equation (18).

Singularity avoidance and robustness

Similar to the joint travel limiting behavior is the behavior which limits motion in the manipulator workspace singular regions. Information in joint space or motion space about the singular regions is required. Some singular regions are qualitatively located at joint limits; these are taken care of by the above behavior. Others are located throughout the workspace; for example, the robot of the present implementation has singularities at configurations when the seventh joint frame is within 0.2 meters or beyond 1.1 meters of the first joint frame. Thus if $\|P_7\|<0.2$ meters then

$$ F_{\text{singularity}} = -K_{\text{singularity}} \cdot (\|P_7\| - 0.2) \cdot \dot{P}_7 $$

or if $\|P_7\|>1.1$ meters then

$$ F_{\text{singularity}} = -K_{\text{singularity}} \cdot (\|P_7\| - 1.1) \cdot \dot{P}_7 $$

where $K_{\text{singularity}}$ is the gain for the singular region avoidance.

is the rigid body transformation between the joint 7 frame and the motion space coordinate system. $P_{\text{actual}}$ is the actual current position of the joint seven frame, $P_{\text{actual}}$ is the unit vector in the direction of $P_{\text{actual}}$, and $F_{\text{singularity}}$ is the singular region avoidance behavior command in the motion space. Note that if the manipulator is not near a singular region there is no commanded motion from this behavior.

In space or other unstructured environments unexpected events may cause the need to operate in singular regions. This can be accomplished by setting a non-zero $W_6$ in Equation 13. However this induces undesired tracking error in non-singular regions. To eliminate this tracking error in non-singular regions, yet allow motion in singular regions, $W_6$ is set with:

if $\|P_7\|>1.1$ meters then

$$ W_6 = K_{\text{robustness}} (\|P_7\| - 1.1) $$

or if $\|P_7\|<0.2$ meters then

$$ W_6 = K_{\text{robustness}} (0.2 - \|P_7\|) $$

where $K_{\text{robustness}}$ is the gain for singularity robustness.

Again, if the manipulator is not near a singular region, $W_6$ is zero and there is no tracking error.

Dither

Dither inputs are incorporated to reduce the effects of friction in tight tolerance tasks. A triangular dither wave function is provided with input parameters specifying its amplitude and period as well as the Cartesian frame in which to compute the waveform. Other waveforms could be used.

The dither input, $X_{d,\phi}$ provides a velocity input to Equation 6 or could generate a force input by multiplying by a damping matrix.
being divided among a number of processors, including a behavior kinematic processor 10, a velocity processor 15, an acceleration processor 20, joint servo loops 25 and robot position and force sensors 30. In operation, the behavior kinematic processor 10 decomposes a received task command into plural behaviors consisting of various behavior kinematic quantities including forces, accelerations, velocities and positions in respective behavior coordinate systems or spaces (block 40 of FIG. 3). The behavior kinematic processor 10 transforms these quantities to a common motion space to which all of the behaviors are transformable (block 42 of FIG. 3). The acceleration processor 20 computes command accelerations for the next sample interval from all of the behavior kinematic quantities as well as from command velocities and positions of the current sampling interval (block 44 of FIG. 3). The velocity processor 15 computes the command velocities and positions of the next sample interval from the command accelerations of the next sample interval and from the command velocities and positions of the current sample interval (block 46 of FIG. 3). At least one of the commanded accelerations, velocities or positions of the current sample interval is transformed to an actuator space of the robot (block 48 of FIG. 3), and applied to the joint servo loops 25 (block 50 of FIG. 3). (In the main embodiment described herein it is the commanded acceleration, while in an alternative formulation described below it is the commanded velocity which is applied to the joint servo loops.) Thereafter, the next sampling interval commences.

Laboratory implementation

The data driven merging of concurrent behaviors for a redundant manipulator has been developed for control of Space Station manipulators. The development and implementation has been clone in the JPL Supervisory Telerobotics (STELER) laboratory. The STELER lab telerobot system is composed of a local site where task commands are specified by an operator with a graphical interface [13] and a remote site where the commands are executed [14]. The remote site was developed to be able to execute multiple concurrent behaviors as described by local site command parameterization, and has been implemented in Ada to be consistent with language constraints for Space Station systems. The system currently uses a seven DOF Robotics Research Corporation K-1207 dextrous manipulator with a six DOF LORD force-torque sensor at the wrist and a served gripper. Autonomous commands are generated with the local site command and sent for execution at the remote site. For teleoperation and shared control tasks, the operator uses a six DOF hand controller. The system is implemented in a six CPU 68020/68881 environment and generates joint position commands each 20 ms which are sent to the manufacturer controller which supplies the joint servo control.

Results

A door opening task, as shown in FIG. 4, is used to demonstrate the use of different control inputs to execute a task. FIGS. 5-10 show experimental results opening the door using shared control, force control, and the spring trajectory. In all three cases the motion frame, where the impedance equation is evaluated, was placed such that its X axis was aligned with the hinge axis. The diagonal M matrix had translational masses of 10 kg and rotational inertias of 2 kg-m^2. The diagonal B matrix had translational gains 350 kg/s and rotational gains of 80 kg-m/s. The diagonal K matrix had translational gains 100 N/m and rotational gains of 10 N/rad.

For shared control door opening, the M and B parameters of the impedance equation were specified and the K matrix was set to zero. The reference force setpoints were all set to zero so that force control would provide compliance to accommodate for inaccuracies in the teleoperated motion of the robot. Tool mode teleoperation of the hand controller was used with the mapping set such that a one DOF rotation of the hand controller wrist was mapped to a rotation about the MOTION frame X axis. The diagonal B matrix was set to zero except for the X rotation component so the operator could only specify rotation about the hinge axis. The results of the door opening task using shared control are shown in FIGS. 5 and 6.

For force control door opening, the M and B parameters of the impedance equation were specified and the K matrix was set to zero. The reference force setpoints were all set to zero except for the torque about the X axis which was set to 15 N-m. This torque setpoint caused the door to open. The results of the door opening task using force control are shown in FIGS. 7 and 8.

For spring trajectory door opening, the M, B, and K parameters of the impedance equation were specified. The reference force setpoints were all set to zero so that force control would provide compliance. The reference position setpoint Xp was set to the destination position representing a rotation of 35 degrees. The reference velocity and acceleration were set to zero. The results of the door opening task using the spring trajectory are shown in FIGS. 9 and 10. The sensed force and torque magnitudes could probably have been reduced by setting the K matrix gains to zero except for about the X axis. Inaccuracy in the specified goal position versus the physical position after opening the door could have caused the sensed forces and torques to increase as the door opened. The rate of opening reduces as the door opens since the difference between the commanded and reference positions reduces.

Alternative Formulation

An alternative implementation approach for Equation 1 is

\[ X_p = X_{ct} - X_r + X_m + X_f + X_{fr} - F_r - F_{fr} + F_{t} - F_{tr} + F_{t} - F_{tr} = X_{ct} - X_r + X_m + X_f + X_{fr} + X_{tf} \]

where

\[ X_{ct} = X_{ct} \]

The terms \( X_m, X_f, X_{fr} \) and \( X_{tf} \) are described below. The other terms, \( X_r, X_{ct}, \) and \( X_{fr} \) are the same as the velocity inputs for the corresponding position based behaviors described above for teleoperation, joint limits, joint singularities, dither, and proximity. Since these terms are velocity inputs, the inputs can be computed in unique frames for each behavior before transforming the resulting velocities to the MERGE frame for summation. The summed velocity perturbation is integrated to produce the sensor based motion which is merged with the reference trajectory motion. A useful result of the approach is that the damping observed by the different sensors can be specified independently.

Inertia control is achieved by differentiating the computed perturbation velocity to provide the perturbation acceleration and then multiplying by the inertia and damping matrices, i.e.,
The resulting perturbation velocities due to position tracking error is, from Equation (27),

\[ \dot{X}_p = \dot{B}^{-1} \cdot X \]

(30)

The perturbation velocities due to force tracking errors in the FORCE frame are given by

\[ \dot{X}_{F} = \dot{B}^{-1} \cdot (P_F - P_p) \]

(31)

Force tracking (force control) can be achieved by setting the stiffnesses, \( K \), reference position trajectory velocity and virtual force inputs to zero in the desired force controlled degrees of freedom. The vector of actual forces \( F_a \) then tracks the reference forces, \( F_r \).

In conclusion, a control architecture for data driven merging of concurrent control behaviors has been developed and implemented on a redundant manipulator. The data driven fixed software approach provides a means for controlling remote space manipulators. The abstraction between task behaviors and manipulator specific mappings allows the approach to be applied to a wide variety of mechanisms. This would be most easily accomplished at the local site.

While the invention has been described in detail by specific reference to preferred embodiments, it is understood that variations and modifications thereof may be made without departing from the true spirit and scope of the invention.

APPENDIX: Teleoperation

Teleoperation inputs originate with an operator moving a 6 DOF hand controller. Two general cases are then possible: world mode teleoperation where the Cartesian velocity of the hand controller grip is computed relative to the fixed base of the hand controller, and tool mode teleoperation where the Cartesian velocity of the hand controller is computed relative to a frame fixed in the hand controller grip. Camera mode, where the manipulator moves relative to the viewing angle provided to the operator via cameras on a separate camera manipulator, is a special case of world mode. For systems with a local-remote architecture, computations for the teleoperation inputs may occur at both the local operator site and at the remote robot control site. In the implementation described in this specification, the local site computations are

\[ \dot{X}_{F} = \dot{B}^{R} \cdot B \cdot (P_F - P_p) \]

(32)

for tool mode and

\[ \dot{X}_{F} = \dot{B}^{R} \cdot B \cdot (P_F - P_p) \]

(33)

for world and camera modes. The joint velocities of the hand controller, \( \theta_a \), are measured in this implementation by differencing successive measured joint positions. The Jacobian \( \dot{H}^{HH} \) transforms the hand controller joint velocities to equivalent Cartesian velocities of the hand controller handle with respect to a frame, \( HH \), attached to the handle. The Jacobian \( \dot{H}^{HH} \) transforms the hand controller joint velocities to equivalent Cartesian velocities of the hand controller handle with respect to a frame, \( HB \), attached to the fixed based of the hand controller.

Computations at the remote site generate the velocities for the TELEOP (T) frame. For tool mode, the velocity of the TELEOP frame is given by

\[ \dot{X}_{T} = \dot{W}^{R} \cdot \dot{W}^{R} \cdot \dot{X}_{F} \]

(34)

where \( \dot{W}^{R} \cdot \dot{X}_{F} \) is computed at the local site with Equation (32) and

\[ \dot{X}_{F} = \dot{B}^{R} \cdot B \cdot (P_F - P_p) \]

rotates the translational and rotational velocity vectors of \( \dot{W}^{R} \cdot \dot{X}_{F} \) from a frame aligned with HH to a frame aligned with TELEOP. (The rotation matrices used here rotate both the translational and rotational components of a velocity vector from one frame, to another ignoring the displacement between the frames.)

\[ \dot{X}_{T} = \dot{W}^{R} \cdot \dot{W}^{R} \cdot \dot{X}_{F} \]

(35)

is constant and is often an identity rotation. Therefore in tool mode, the velocities input along and about the axes of the TELEOP frame are the same as the velocities along and about axes of the hand controller handle frame, \( HH \).

For world mode, the velocity of the TELEOP frame is given by

\[ \dot{X}_{T} = \dot{W}^{R} \cdot \dot{W}^{R} \cdot \dot{X}_{F} \]

(36)

where \( \dot{W}^{R} \cdot \dot{X}_{F} \) is computed at the local site with Equation (33),

\[ \dot{X}_{F} = \dot{B}^{R} \cdot B \cdot (P_F - P_p) \]

rotates the velocities from the remote WORLD frame to the TELEOP frame at the remote site.

\[ \dot{X}_{T} = \dot{W}^{R} \cdot \dot{W}^{R} \cdot \dot{X}_{F} \]

(37)

is dependent on both the position of the arm base relative to the WORLD frame and the current arm joint angles. Both the local and remote sites have a WORLD frame, \( WL \), and \( WR \), respectively. These are the same frames and are what link absolute motion at the local and remote sites.

For camera mode, the velocity of the TELEOP frame is given by

\[ \dot{X}_{T} = \dot{W}^{R} \cdot \dot{W}^{R} \cdot \dot{X}_{F} \]

(38)

where \( \dot{W}^{R} \cdot \dot{X}_{F} \) is computed at the local site with Equation (33),

\[ \dot{X}_{F} = \dot{B}^{R} \cdot B \cdot (P_F - P_p) \]

rotates the velocities from local WORLD to the ST frame which is fixed in the stereo display screen which the operator
uses to view the remote site screen. The ST frame at the local site corresponds to the EYE frame at the remote site. The EYE frame is fixed between the stereo cameras at the remote site which are generating images for the stereo view at the local site.

rotates the velocities from EYE to WORLD at the remote site. In the present implementation, the stereo cameras are mounted on a separate camera robot so

is dependent on the current joint angles of the camera arm.

rotates the velocities from remote WORLD to TELEOP.

Force reflection back to the hand controller can be integrated with the control algorithm. Force reflection is independent of the remote site control, but is included here for completeness of the capability. Any force at the remote site could be reflected back to the local site, depending on the application. The classic single arm case is to reflect back the actual force measured with the wrist force-torque sensor, \( F_w \). For the dual arm case, the move space forces are reflected back. Alternatively, the force tracking error, \( F_r - F_w \), may be better to reflect to the operator. The forces selected to be reflected back to the local site will be called \( F_{wr} \), indicating the force reflection forces in the TASK (TK) frame. The TK frame will be different depending on the application or which forces are being reflected back to the local site.

Computations for tool, world and camera modes are different, but the results for all cases are the forces to apply at the actuators of the hand controller. For tool mode the hand controller actuator forces are computed with

\[
\tau_k = I_{H}^{-1} J_{H}^R \cdot \{ F_{wr} \} \cdot TK \cdot F_{p}
\]  

For world mode the hand controller actuator forces are computed with

\[
\tau_k = I_{H}^{-1} J_{H}^B \cdot \{ F_{wr} \} \cdot TK \cdot F_{p}
\]  

For camera mode the hand controller actuator forces are computed with

\[
\tau_k = I_{H}^{-1} J_{H}^B \cdot \{ F_{wr} \} \cdot TK \cdot F_{p}
\]  

The computations for force reflection may be separated into local and remote computations. This may be required if the video representation of the scene is fed back with a different time delay than the force data. Then for camera mode force reflection it may be desirable to have the remote EYE frame time synchronized with the video images rather than with the reflected forces.

What is claimed is:

1. A method of operating a robot to perform a task in an environment of objects, said robot having joints and joint actuators, said method comprising the steps of:
   decomposing said task into behavior kinematic quantities of plural behaviors to be exhibited simultaneously in respective behavior spaces by said robot, said plural behaviors including exerting force on at least one of said objects;
   first transforming said behavior kinematic quantities from said respective behavior spaces to a common space;
   computing, in said common space, from said behavior kinematic quantities command kinematic quantities including a force kinematic quantity corresponding to the force to be exerted on said one object and controlling said joint actuators in accordance with said command kinematic quantities;
   wherein said computing comprises computing a first command kinematic quantity for a next sampling interval from:
   (a) said behavior kinematic quantities, and
   (b) other command kinematic quantities of a current sampling interval; and
   computing said other command kinematic quantities for said next sampling interval from said first command kinematic quantity of said next sampling interval and from said other command kinematic quantities of said current sampling interval.

2. The method of claim 1 wherein said computing a first command kinematic quantity comprises the computing of a commanded acceleration from a commanded velocity and a commanded position, and from behavior forces, behavior accelerations, behavior velocities and behavior positions.

3. The method of claim 1 wherein said computing a first command kinematic quantity comprises the computing of a commanded velocity from a commanded acceleration and a commanded position.

4. The method of claim 1 wherein:
   said common space is a motion space common to which all of said behavior kinematic quantities are transformable; and
   said controlling said joint actuators is preceded by second transforming at least some of said command kinematic quantities from said motion space to an actuator space of said joint actuators.

5. The method of claim 4 wherein:
   said robot further comprises joint position sensors for detecting positions of said joints; and
   said second transforming comprises computing a transform to said actuator space based upon current angles of said joints detected by said joint position sensors.

6. The method of claim 5 wherein:
   said decomposing defines the behavior kinematic quantities of each of said behaviors in respective behavior spaces of each of said behaviors;
   said first transforming comprises transforming the behavior kinematic quantities of each respective behavior from the corresponding behavior space to said motion space.

7. The method of claim 6 wherein said decomposing step defines the behavior kinematic quantities of each of said behaviors in respective behavior spaces each having a dimensionality corresponding to respective degrees of freedom of the robot.

8. The method of claim 7 wherein said decomposing step defines the behavior kinematic quantities of each of said behaviors in respective behavior spaces each having a dimensionality that does not exceed a maximum number of degrees of freedom of said robot.

9. The method of claim 1 wherein said decomposing step comprises the decomposing of said task into behavior kinematic quantities of said behaviors wherein each said behavior is governed in accordance with a corresponding input parameter of the type including force set points, joint singularity angle, joint limit angle, and obstacle location,
said method further comprising reprogramming selected ones of said input parameters each sample interval, whereby to dynamically control said plural behaviors.

10. The method of claim 1 wherein said command kinematic quantities comprise acceleration, velocity and position, said behavior kinematic quantities comprise force, acceleration, velocity and position, and wherein the computing of said command kinematic quantity comprises equating a sum of proportionate differences between command and behavior accelerations, velocities and positions, respectively, with a sum of measured forces of said robot and of behavior forces.

11. The method of claim 10 wherein said computing of said command kinematic quantity further comprises making said proportionate differences proportionate in accordance with inertia damping and stiffness characteristics respectively, of said robot.

12. A method of operating a robot in successive sampling intervals to perform a task, said robot having joints and joint actuators with actuator control loops, said method comprising:

- decomposing said task into behavior kinematic quantities comprising behavior forces, accelerations, velocities and positions of plural behaviors to be exhibited in respective plural behavior spaces by said robot simultaneously and first transforming said behavior forces, accelerations, velocities and positions to a common space;
- computing in said common space command accelerations for a next sampling interval from:
  (a) said behavior forces, accelerations, velocities and positions; and
  (b) command velocities and positions of a current sampling interval;
- computing command velocities and positions for said next sampling interval from said command accelerations of said next sampling interval and from said command velocities and positions of said current sampling interval; and
- controlling said joint actuators in accordance with at least one of said command accelerations, velocities and positions of said current sampling interval.

13. The method of claim 12 wherein:

- said common space is a motion space common to which all of said behavior forces, accelerations, velocities and positions are transformable; and
- said controlling said joint actuators is preceded by second transforming said command accelerations, velocities and positions from said motion space to an actuator space of said joint actuators.

14. The method of claim 13 wherein:

- said robot further comprises joint position sensors for detecting positions of said joints; and
- said second transforming comprises computing a transform to said actuator space based upon current angles of said joints detected by said joint position sensors.

15. The method of claim 14 wherein:

- said decomposing defines the behavior forces, accelerations, velocities and positions of each of said behaviors in respective behavior spaces of each of said behaviors; and
- said first transforming comprises transforming the behavior forces, accelerations, velocities and positions of each respective behavior from a corresponding one of said respective behavior space to said motion space.

16. The method of claim 15 wherein the decomposing step defines the behavior forces, accelerations, velocities and positions of each of said behaviors in respective behavior spaces each having a dimensionality corresponding to respective degrees of freedom of the robot.

17. The method of claim 16 wherein the decomposing step defines the behavior forces, accelerations, velocities and positions of each of said behaviors in respective behavior spaces each having a dimensionality that does not exceed the maximum number of degrees of freedom of an arm.

18. The method of claim 17 wherein the decomposing step defines the behavior forces, accelerations, velocities and positions of an avoidance of one of a joint limit, joint singularity or obstacle, for which the corresponding behavior kinematic quantity is a function of the proximity of the joint limit, joint singularity or obstacle, respectively.

19. The method of claim 12 wherein said computing said command accelerations comprises equating a sum of proportionate differences between command and behavior accelerations, velocities and positions, respectively, with a sum of measured forces of said robot and of said behavior forces.

20. The method of claim 20 wherein said computing said command accelerations further comprises making said proportionate differences proportionate in accordance with inertia damping and stiffness characteristics, respectively, of said robot.

21. The method of claim 20 wherein said computing said command accelerations further comprises:

- means for decomposing said task into behavior kinematic quantities of plural behaviors to be exhibited simultaneously in respective behavior spaces by said robot, said plural behaviors including exerting force on at least one of said objects;
- means for transforming said behavior kinematic quantities from said respective behavior spaces to a common space; and
- means for computing, in said common space, from said behavior kinematic quantities command kinematic quantities including a force kinematic quantity corresponding to the force to be exerted on said one object and controlling said joint actuators in accordance with said command kinematic quantities;

wherein said means for computing comprises a second means for computing a first command kinematic quantity for a next sampling interval from:

- (a) behavior kinematic quantities, and
- (b) other command kinematic quantities of a current sampling interval; and

third means for computing said other command kinematic quantities for said next sampling interval from said first command kinematic quantity of said next sampling interval and from said other command kinematic quantities of said current sampling interval.

22. A robot control system for operating a robot in successive sampling intervals to perform a task in an environment of objects, said robot having plural joints and joint actuators, said system comprising:

- a method of computing comprises a second means for computing a first command kinematic quantity for a next sampling interval from:
  (a) behavior kinematic quantities, and
  (b) other command kinematic quantities of a current sampling interval; and

- third means for computing said other command kinematic quantities for said next sampling interval from said first command kinematic quantity of said next sampling interval and from said other command kinematic quantities of said current sampling interval.

23. The system of claim 22 wherein said second means for computing comprises means for computing a commanded
acceleration from commanded velocity and commanded position, and from behavior forces, behavior accelerations, behavior velocities and behavior positions.

24. The system of claim 22 wherein said second means for computing comprises means for computing a commanded velocity from commanded acceleration and commanded position.

25. The system of claim 22 wherein:
   said common space is a motion space common to which all of said behavior kinematic quantities are transformable; and
   said first means for computing and controlling said joint actuators comprises second means for transforming at least some of said command kinematic quantities from said motion space to an actuator space of said joint actuators.

26. The system of claim 25 wherein:
   said robot further comprises joint position sensors for detecting positions of said joints; and
   said second means for transforming comprises means for computing a transform to said actuator space based upon current angles of said joints detected by said joint position sensors.

27. The system of claim 26 wherein:
   said means for decomposing defines the behavior kinematic quantities of each of said behaviors in respective behavior spaces of each of said behaviors; said first means for transforming comprises means for transforming the behavior kinematic quantities of each respective behavior from the corresponding behavior space to said motion space.

28. The system of claim 27 wherein the means for decomposing defines the behavior kinematic quantities of each of said behaviors in respective behavior spaces each having a dimensionality corresponding to respective degrees of freedom of the robot.

29. The system of claim 28 wherein said means for decomposing defines the behavior kinematic quantities of each of said behaviors in respective behavior spaces each having a dimensionality that does not exceed the maximum number of degrees of freedom of said arm.

30. The system of claim 22 wherein said command kinematic quantities comprise acceleration, velocity and position, said behavior kinematic quantities comprise force, acceleration, velocity and position, and wherein said first means for computing comprises means for equating a sum of proportionate differences between command and behavior accelerations, velocities and positions, respectively, with a sum of measured forces of said robot and of behavior forces.

31. The system of claim 30 wherein the means for computing comprises means for equating the sum of said differences wherein said differences are proportionate in accordance with inertia, damping and stiffness characteristics, respectively, of said robot.