Michael C. Nechyba and Yangsheng Xu*

Human-Robot Cooperation in Space: SM^2 for New Space Station Structure

The Self-Mobile Space Manipulator (SM^2) is a seven-DOF inspection robot, capable of walking along the pre-integrated I-beam truss structure of Space Station Freedom. The robot is capable of projecting cameras anywhere interior or exterior of the SSF, and will be an ideal tool for inspecting connectors, structures, and other facilities on the SSF. Using a full-scale model of a small segment of the SSF, experiments have been performed under two gravity compensation systems.

The Self Mobile Space Manipulator (SM^2) being developed at Carnegie Mellon University is a walking robot intended to assist astronauts on the Space Station Freedom in inspection and maintenance tasks. If deployed, it could alleviate the need for dangerous EVA missions by astronauts and may be utilized quickly in emergency situations.

The SM^2 robot was originally designed to operate on a tubular strut-and-node truss structure [5, 6]. At that time, the truss design was rectangular and nodes were spaced at equal lengths from one another. Since then, that truss design has been changed by NASA in favor of the current pre-integrated truss (PIT) design, utilizing I-beam members. The new truss design is hexagonal, rather than rectangular in shape. Therefore, our first goal was to modify the SM^2 configuration to adapt to this new space station truss and demonstrate system feasibility.

The second goal of the project is to specify the SM^2 robot as an inspection robot. There is a vital need for astronauts to inspect facilities on the space station, such as fluid connectors, electric cables, and bolted segments. Able to reach both exterior and interior of the space station, the movable cameras will be essential for this task. SM^2 will be capable of projecting cameras to any position on the space station through its inherent self-mobility.

In this article, we overview the SM^2 configuration and testbed. Then, we discuss the control of autonomous and teleoperated locomotion in detail, including the real-time control architecture, the autonomous stepping motion based on a neural-network vision system, and the teleoperation control.

NEW SM^2 CONFIGURATION AND TESTBED

Robot Kinematic Configuration

In order to enable the robot to step from one face of the hexagonal PIT structure to adjacent faces, and to retain the symmetry of SM^2, the robot requires a total of seven joints, as shown in Figures 1 and 2. The symmetry of the robot mechanism is important for the control of locomotion, so that as the base of the robot is switched, we simply switch the numbering of the joints from the base to the tip. This allows the out-of-plane motion needed to step from one face of the truss to another. Each of the seven joints is identical, self-contained and modular so that a minimum inventory of parts is required for joint repair or replacement. The joints are driven by harmonic motors and are wired in a modular fashion so that only one 16-pin connector is required to deliver all signals and power to each of the joints. Currently, power and communication is delivered to the robot through a

*The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15217

1070-9932/95/$4.00©1995 IEEE

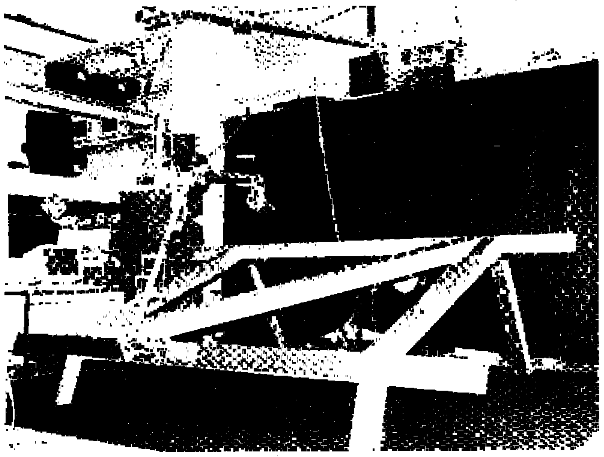


Figure 1. The new SM² testbed with new robot, truss mock-up, gravity compensation systems and control station.

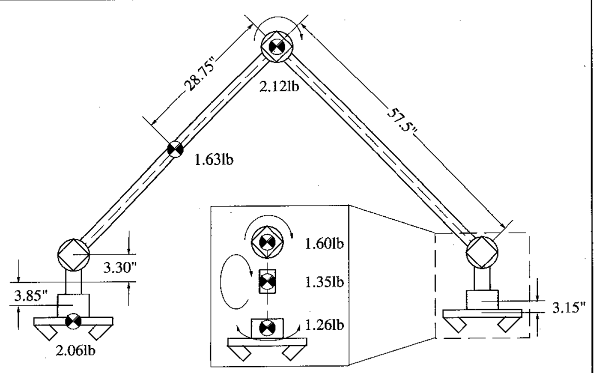
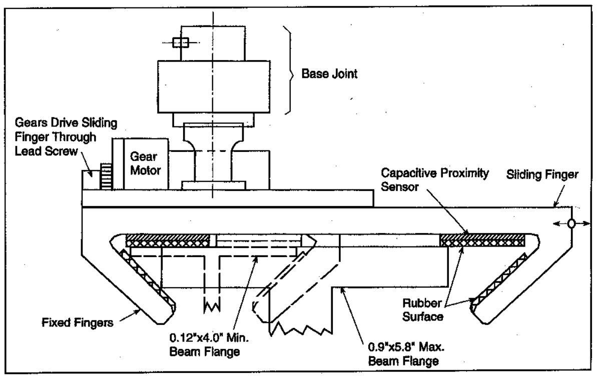


Table 1: Denavit-Hartenberg Parameters for SM²

i	α_{i-1}	α_{i-1}	θi	di	$[\theta_{min}, \theta_{max}]$
1	0	0	θ_1	6.85"	$[-\pi, \pi]$
2	$\pi/2$	0	θ_2	0	$[0, \pi]$
3	$-\pi/2$	3.30"	θ_3	0	$[0, 3\pi/4]$
4	0	57.5"	θ_4	0	$[0, 3\pi/4]$
5	0	57.5"	θ_5	0	$[0, 3\pi/4]$
6	$\pi/2$	3.30"	θ_6	0	$[0, \pi]$
7	$\pi/2$	0	θ_7	0	$[-\pi, \pi]$
8	0	0	0	6.85"	---

Figure 2. Dimensions, configuration, kinematics, and weight of SM² (not drawn to scale).



Base Joint
Gears Drive Sliding Finger Through Lead Screw
Gear Motor
Capacitive Proximity Sensor
Sliding Finger
Rubber Surface
Fixed Fingers
0.12"x4.0" Min. Beam Flange
0.9"x5.8" Max. Beam Flange

Figure 3. Half-scale side view of the newly designed I-beam gripper.

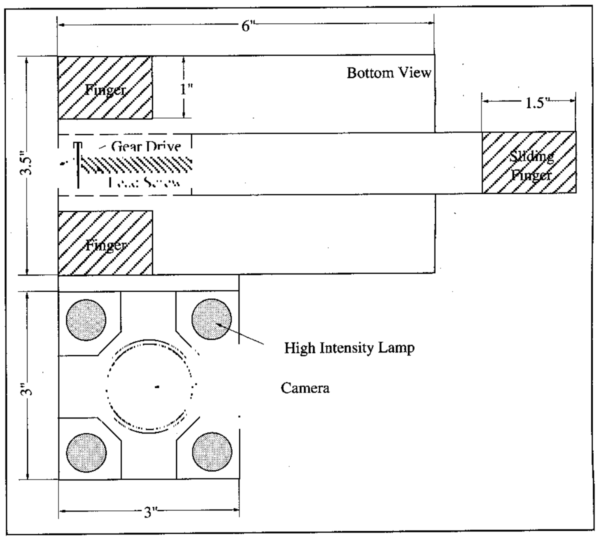


Figure 4. Half-scale bottom view of the newly designed I-beam gripper and end-effector camera position.

number of cables. In the actual deployment, however, the robot must be self-contained and will be required to carry power on-board. For the system described in this paper, this issue has not yet been addressed.

Beam Grippers

The truss structure requires grippers that can attach to the aluminum I-beams of the PIT structure. Each end of SM² is equipped with a three-fingered gripper capable of grasping I-beam flanges of various thickness and width, as shown in Figure 3 and Figure 4. The single finger, driven by a DC motor, slides back and forth to allow opening and closing of the gripper. A linear potentiometer measures the single finger position, while motor current indicates grasp force. Frictional forces are negligible compared to the grasp force.

Each gripper is equipped with sensors necessary for reliably and securely grasping the beam. Using force-sensing resistors, contact switches on each of the three fingers can be checked to verify a good grasp. In addition, capacitive proximity sensors at the base of the fingers sense beam proximity up to about four inches away and are useful in aligning the gripper with the beam.

Cameras

There are three camera modules attached to the robot, one at each tip, and one on the elbow joint. Each camera has separate controllable zoom, focus, and iris with four high-intensity lamps arranged around each camera (Figure 4).

The elbow camera has one motorized DOF as shown in

Figure 5. Since the robot has one redundant DOF, the elbow camera can be thought of having effectively two DOFs in determining it's view. With both ends of the robot attached to the truss, for example, the collection of all possible views sweeps out a half torus about an axis defined by the two base joints at each tip. Thus, the elbow camera can provide valuable visual information about global location on the space station.

The two tip cameras serve twin purposes. The primary purpose is, of course, visual inspection by human operators. The robot tip camera at the free end can provide views of the truss structure that any fixed camera around the space station simply cannot achieve, both inside and outside the truss structure. I-beam connections as well as the inside faces of the I-beams are two locations where a movable camera might provide significantly better views. The cameras are also used for autonomous locomotion on the truss. Neural-network based machine vision with images from the tip camera is used to autonomously mate the gripper to the I-beam flanges.

Gravity Compensation Systems

To simulate the zero gravity environment of space, we use two independent gravity compensation systems developed at Carnegie Mellon University. Each gravity compensation system provides a constant upward vertical force through a counterweight mechanism and a series of cable and pulleys. The support cables are attached to the center of gravity of the two long links on the robot. A 10:1 ratio in the counterweight mechanism keeps the increased inertia in the vertical direction to 10%.

The support cables attached to the robot are tracked overhead by two separate, actively controlled carriage systems. Angle sensors detect x-y movement of the support cables. The first system is a Cartesian gantry system (Figure 6) and allows robot motion in an area that is 17 feet (5.2m.) long and 9 feet (2.8m.) wide. This allows us to test large global stepping motions for the robot. The second system is a smaller cylindrical compensation system supporting a smaller field of motion. This allows a large variety of motions to be tested without the supporting cable of the larger system interfering with the carriage beam of the smaller system [5].

In addition to the mechanical gravity compensation, we provide for active residual gravity compensation in software. This is especially necessary to provide appropriate torques for the three joints at the free end of the robot, whose weight is a significant fraction of the total body weight (Figure 2). The combination of mechanical and active gravity compensation allows for realistic zero gravity experiments and testing.

Truss Mock-up

In our lab, we have built a truss mock-up, which is a full-scale representation of a small portion of the entire truss structure on the space station. The mock-up includes four faces of the hexagonal structure as shown in Figure 1. Each beam is constructed of wood with sheet aluminum laminated to the flange faces to allow for realistic machine vision testing. Varying flange widths and thicknesses also allow for robust testing of the grippers.

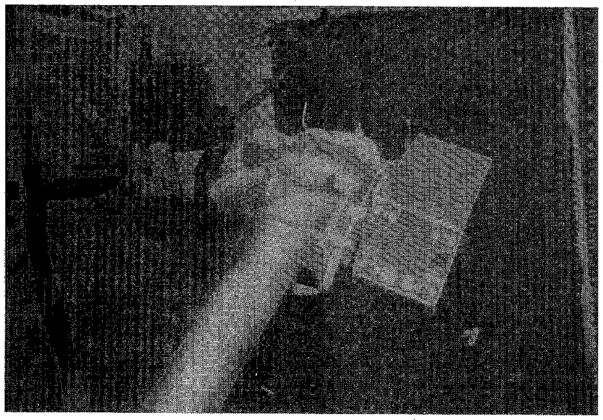


Figure 5. The elbow camera module can be programmed to track the active end of the beam or may be teleoperated.

REAL-TIME SHARED CONTROL ARCHITECTURE

System Overview

At the heart of the SM^2 control software lies a real-time shared control architecture [2] which has been integrated into the real-time operating system CHIMERA being developed at CMU [4]. The control software is modular in design whereby tasks are composed of independent, reusable subtasks. High level tasks for the SM^2 robot range from teleoperation to semi-autonomous (constrained frame) tasks to fully autonomous walking. These tasks often use many of the same subtasks such as trajectory tracking, beam grasping, point convergence, and switching the base of the robot. In essence, subtasks are "shared" among the different tasks. These subtasks are coded as modular library routines which may be dynamically sequenced through a coordination module and state machine.

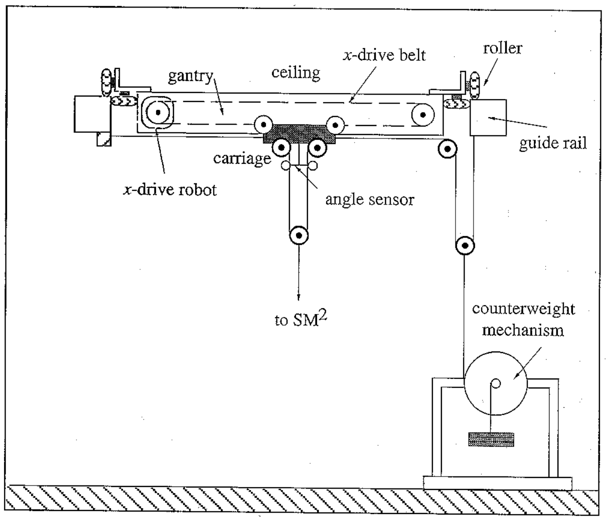


Figure 6. The Cartesian gravity compensation system used to support the weight of SM^2.

Coordination of Tasks

The various task modules need to be coordinated in an intelligent fashion. A state machine, programmable through a simple language and parsed in real-time, does just that as illustrated in Figure 7. The state file describes the following attributes of the state machine:

(1) Defines the number of subtasks and the possible message inputs and outputs for each subtask.
(2) Defines all tasks (states).
(3) Defines all possible transitions and the initial task (state).

A subtask is defined as shown in the following example:

```
SUBTASK grasp
INPUT on off open close stop gripper1
    gripper2
OUTPUT nocontact contact done grabbed
```

The first line merely assigns a label to the subtask. The second line gives a list of valid messages that the subtask "grasp" will accept as input. Each of these inputs is easily understood. For example "open" commands the subtask to open the gripper, while "gripper2" commands the subtask to switch to gripper2. Finally, the last line specifies the outputs of the subtask.These are then used in the sequencing of states.

A typical task specification might appear as follows:

```
TASK        tele_gripper_close
SUBTASKS    grasp tele
START       tele:on tele:grp grasp:close
END         grasp:off
```

Here, again, the first line merely assigns a label to the task. The second line specifies which subtasks are part of the overall task. In this example, both "grasp" and "teleoperation" combine to form the specific task. The next line specifies what messages to send to the various subtasks at the start of the overall task. The first two commands make certain that teleoperation is in the "on" mode and that the control mode is the "gripper mode." The final start message instructs the "grasp" subtask to attempt to close the gripper. In the final line, we specify what messages to send at the end of a task execution. Once the gripper is closed, we instruct the subtask "grasp" to turn off.

Finally, below we show an example of specifying state transitions and an initial state:

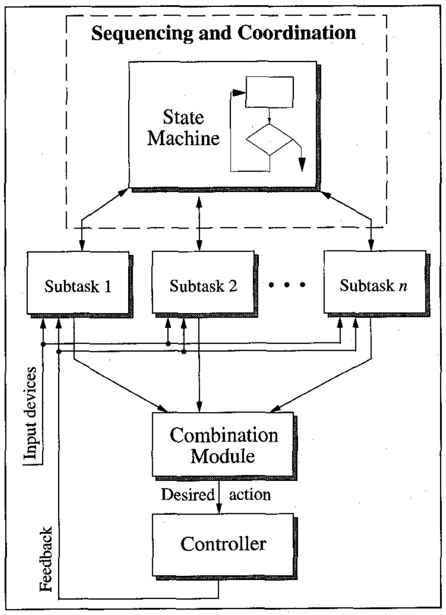


Figure 7. Overall data flow of the real-time shared control architecture.

```
TRANSITION tele:down tele_gripper_idle
    tele_gripper_close
INITIAL_TASK tele_init
```

The transition statement simply states that when the subtask "tele" receives a "down" message—when the appropriate button is pressed on the teleoperation hand controller—the state machine should sequence from the "idle gripper" mode to the "close gripper" mode.

In such a manner, high-level tasks can quickly be programmed from a library of subtasks through the state machine. Note that subtasks are reusable from state to state and can be switched on and off when necessary. For example, the "grasp" subtask is equally necessary in the autonomous locomotion mode as well as the teleoperation mode.

In short, the state machine allows subtasks to be shared by high-level tasks which can be rapidly re-programmed with minimal re-coding and no re-compilation. This allows for elegant and rapid software development.

Task Modules

We have developed several reusable task modules for the SM^2 control software. In each control cycle, the task modules perform four basic functions:

(1) Read messages from the state machine and respond in appropriate fashion.
(2) Read sensor devices, global variables, or receive input from remote tasks.
(3) Generate desirable control motion based on local inputs.
(4) Send appropriate messages to the state machine.

Since each subtask module produces desired control commands based solely on its limited criteria, one module—the "combination" module—is required to intelligently combine these desired control outputs from individual task modules into one coherent control signal. The combination module therefore can ensure reasonable control outputs based on different criteria such as a weighted average or a priority ordering for the control commands of the individual task modules.

Remote task modules do not fundamentally differ from other modules except in one respect. These modules are run on a separate workstation or processing board, usually due to high computational requirements that cannot be met in real time. These modules can interface with the slower real-time boards via UNIX sockets, a VME bus, or serial lines. Menu-driven user interfaces as well as a real-time graphical displays are two examples of such computationally intensive remote tasks. These, along with the other task modules will be discussed in the context of the following two sections which discuss (1) autonomous walking on the truss, (2) and teleoperation.

AUTONOMOUS LOCOMOTION

Model-based Walking

The operating environment for SM^2 is very structured and can easily be modeled with a great degree of accuracy. Hence, it is possible for the robot to execute a pre-planned sequence of walking steps based solely on a model of the space station truss structure. Here, no vision-based sensing is required at the end-effector. We have successfully executed various sequences of four steps on the truss mock-up, including steps of variable length and between different faces of the hexagonal space station truss structure. Each walking step takes between 20 and 30 seconds and can be decomposed into several distinct phases: (1) ungrasping the beam, (2) separating smoothly from the beam, (3) executing a global trajectory, (4) approaching the target beam, (5) executing a straight-line motion towards the beam, (6) closing the gripper, and (7) switching the base for the next step. Figure 8 illustrates one example of how SM^2 moves on the truss structure.

First, the gripper is opened until the sliding potentiometer indicates that the gripper is in the fully opened position. Second, while keeping the orientation of the gripper aligned with the beam, the free end is moved above the beam in a straight-line motion so as to avoid potential collisions with the space station truss. Once the free end is safely above the truss structure, control is switched to the execution of a global trajectory in the state machine.

A global trajectory is defined minimally by the starting point and the target destination. The operator, however, is free to include as many via points as he chooses along the path of the trajectory. These points may be generated alternatively in a pre-programmed file or through the real-time graphical display as discussed in the subsequent section. As the trajectory is being executed, errors are dynamically corrected by continuously calculating a smooth path between the current position and the desired trajectory path. If no, intermediate points are specified along the trajectory, the inverse kinematic algorithm, as explained later on, will generate intermediate points which lead to a smooth trajectory.

The trajectory will finish with the proper gripper orientation about 20 inches above the target beam and location. From there, the state machine enters the next phase of execution; that is, a straight line descent towards the target beam along the surface normal of the beam.

Proximity sensors on the gripper fingers allow the surface normal and the gripper normal to be lined up when the gripper has closed to within four inches of the beam. The three contact switches on the gripper fingers confirm contact once the gripper and the beam are touching. Upon establishing contact, the gripper is closed until a stall current is sensed in the gripper motor. Finally the gripper is opened approximately 0.25 inches and closed once again. This generally guarantees that any slight misalignment of the gripper is passively corrected.

Finally, if another step is to follow, the robot will switch bases. What was the free end before, will now become the fixed base and vice versa. It is important to note that the entire sequence described above is controlled through the state machine. Each phase of the stepping motion will execute only when the appropriate "done" message is sent by the control software to the state machine. The proper "done" message triggers a transition to the next state. The entire walking step is divided into a sufficient number of subtasks, any or all of which can be used during other modes, such as teleoperated or semi-autonomous control.

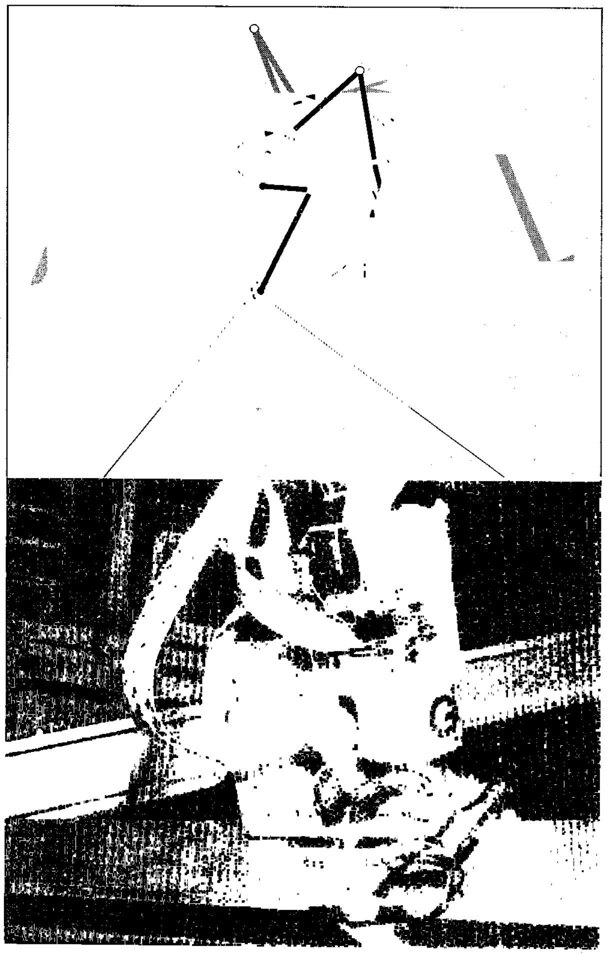


Figure 8. SM^2 can reach any place on the truss through a sequence of autonomous steps. The blow-up illustrates how the gripper mates with the I-beam flanges.

Neural Network Based Visual Servoing

Although we have a good model of the environment, errors can accumulate over consecutive steps if and when the gripper slips while mating with the I-beam. This can potentially lead to failure in properly grasping the next beam. If this should occur, a neural-network based vision system can assume control, correct any such error and properly complete the grasping of the beam. It is preferable to use the vision system only when failing to complete a grasp, however, since the vision system slows the system performance significantly. The main bottleneck is, of course, the acquisition of the images at a high rate.

We have trained a three-layer, feed-forward neural network on 40x40 digitized images of flanges at various translational offsets, heights, and rotations. It has been demonstrated elsewhere [3] that image resolution as low as 30x30 is sufficient for neural-network-based perception. The neural network learns through the standard back-propagation learning algorithm and maps the digitized images directly to Cartesian position and orientation commands at the end-effector. The vision system is functional even when the end-effector is over one meter from the target beam. Once the vision system has placed the gripper in contact with the beam, the state machine returns control to the same states and subtasks used for closing the gripper as mentioned previously. For our walking experiments, the error typically does not accumulate sufficiently over four steps to require the vision system. Thus, in order to verify the vision system, we introduced artificial disturbances to the robot, by giving a slightly incorrect model of the truss mock-up.

Unlike the previous strut-and-node design of the space station truss structure, the current design causes uncertainty in the exact location of the robot on the truss structure, since SM^2 is free to grasp the beam anywhere along the length of the beam. That uncertainty could potentially be removed periodically by using the vision system to locate certain known locations on the space station truss. One such special feature might be where two or more beams join. Further work needs to be done in this direction.

TELEOPERATION

We have developed two different methods for teleoperation. The first method utilizes a six-DOF hand controller to guide the free end of the robot. The second method utilizes the real-time graphics display which provides two views of the space station truss structure. By selecting the target location for the robot arm with a mouse, the robot can be made to execute large global trajectories. In either case, the redundant DOF has to be controlled. Below, we review how we resolve the kinematic redundancy of the SM^2 robot, and then describe the teleoperation modes we have developed.

Redundancy Control

Since SM^2 has seven DOFs, the robot is kinematically redundant and no unique inverse kinematic solution exists. In addition, the computational demands of the overall system in real time restrict the computational complexity of the inverse kinematic algorithm we choose. Finally, for teleoperation, we do not necessarily require exact inverse kinematics. For these reasons, we implemented an efficient, variable-gain, Jacobian-based algorithm to approximate the inverse kinematics. The method is derived from the work in [7].

Here we summarize the algorithm and point to some of its advantages. Denote the 6x7 Jacobian as $J(\theta)$. Then, the differential forward kinematic relationship is given by,

$$dx = J(\theta)d\theta \qquad \text{(Eq. 1)}$$

where dx represents the 6x1 differential Cartesian displacement vector and $d\theta$ represents the 7x1 differential joint displacement vector. Now, define the following terms:

$$r_i = \sum_{i=1}^{7} \left|J_{ij}\right| \qquad \forall j \in \{1,\ldots,6\} \qquad \text{(Eq. 2)}$$

$$c_j = \sum_{i=1}^{6} \left|J_{ij}\right| \qquad \forall j \in \{1,\ldots,7\} \qquad \text{(Eq. 3)}$$

where J_{ij} represents the ixj element of the Jacobian.

Now define the following matrix operator for a positive semidefinite diagonal matrix Σ. Denote Σ by,

$$\Sigma = diag(\sigma_1, \sigma_2 \cdots, \sigma_n) \qquad \text{(Eq. 4)}$$

where $\sigma_i \geq 0$. Define $\Sigma^* = diag(\rho_1, \rho_2, \ldots, \rho_n)$ where,

$$\rho = \begin{cases} \frac{1}{\sigma} & \sigma_1 > \varepsilon \\ \varepsilon & \sigma_1 \leq \varepsilon \end{cases} \qquad \text{(Eq. 5)}$$

Here, ε is some "small" positive value close to zero. It is essentially determined by the machine precision of the digital computer being used.

Finally, define the following two diagonal matrices:

$$\begin{aligned} R &= diag(r_1, r_2, \ldots r_6) \\ C &= diag(c_1, c_2, \ldots c_7) \end{aligned} \qquad \text{(Eq. 6)}$$

Then, the inverse kinematic algorithm may be expressed as,

$$d\theta = C^* J^T R^* dx \qquad \text{(Eq. 7)}$$

Thus, C^* and R^* may be thought of as adaptive, configuration-dependent, positive-definite gain matrices. Global convergence to zero-steady state error has been shown in [7].

The method is very efficient, requiring only 106 additions/subtractions, 42 multiplications and 13 divisions per computational cycle. This makes the method several times more efficient than Jacobian-based pseudo-inverse methods and only slightly less efficient than fixed-gain Jacobian transpose methods. In addition, the method shows much faster error convergence than does the fixed-gain Jacobian transpose method. Typically, our method converges to within 0.1% error within 5 iterations.

Thus, redundancy control is now easily implemented. Either, redundancy resolution is left up to the inverse kinematic algorithm, or the Jacobian is augmented with an additional constraint, such as keeping the second base joint within a certain range.

Hand Controller

We use a commercial, six-DOF, free-flying hand controller [1] as the principal means for teleoperated control. The device operates with a stationary receiver and a moving radio transmitter. Both the position and orientation of the transmitter relative to the receiver is communicated via a serial line to the

The operator can control the robot by moving the free-flying 6-DOF hand controller through space.

controller at a rate of 10Hz. The moving transmitter is attached to a cylindrical stick with an enable switch controlled with the thumb, and another multi-purpose two-way switch controlled with the index finger. Figure 9 illustrates the control station configuration and the use of the hand controller.

The hand controller is used in conjunction with a graphical user interface to determined the mode of operation for the hand controller as well as the function of the two-way switch. The menu-driven user interface allows the operator to select one of three basic modes of operation, as well as which end of the robot is the active one. The three modes are (1) position control, (2) velocity control, and (3) gripper control.

In position and velocity mode, the two-way switch controls the speed corresponding to given displacements of the transmitter. In gripper mode, the two-way switch controls the opening and closing of the gripper. Velocity control is generally used during large global motions of the robot, while position and gripper control are used when grasping a beam and switching the fixed base of the robot.

In each mode, the operator can select whether the motion of the free end of the robot is to be base-relative, tip-relative, or semi-autonomous. Tip-relative motion is generally the most useful when the only visual feedback for the operator is from the elbow and tip camera (i.e., the robot itself is hidden from view). Base-relative motion is useful in conjunction with either fixed camera views or the real-time graphical display which reveal the global position of SM^2 on the space station truss.

In manually mating the free end of the robot to one of the I-beam flanges, the semi-autonomous mode simplifies the process for the operator. The semi-autonomous mode allows the operator to automatically orient the free gripper to the correct orientation for grasping the beam. The control software utilizes knowledge of which beam the fixed end is currently attached to and which beam the operator wishes to grasp in order to select the proper orientation for the gripper. With this semi-autonomous orienting, the process of teleoperated walking on the space station truss is sped up significantly. Requiring only minimal training, we have repeatedly demonstrated teleoperated walking on the truss mock-up, with and without the robot in view of the operator.

The above discussion illustrates several dimensions of the shared control architecture. We achieve a blend of teleoperation and autonomous locomotion without the need for new software code. In the semi-autonomous teleoperated mode, we use the same subtask to achieve the proper orientation of the gripper before grasping as we do in autonomous walking. Furthermore, we are able to use the same grasping subtask for autonomous walking and teleoperation. In fact, the message to the state machine issued during autonomous walking and teleoperated control is exactly the same: "close gripper." Thus, all the safety precautions used for ensuring a secure grasp of the beam during autonomous walking are automatically incorporated when the operator commands the gripper to close on the beam.

In another example, the operator may wish to inspect the length of a beam. Rather than worry about following a precise straight line with the hand controller, the operator may wish to surrender control of one directional degree of freedom (sideways to the beam) so that he can inspect the length of the beam with variable speed, approaching the beam closer if some damage is spotted. This may be achieved by employing the same trajectory subtask as is used for the autonomous walking. Again, the shared control architecture allows an elegant merging of autonomous and teleoperated function. Simply with some minor additions to the state machine, the teleoperation function is seamlessly incorporated into the overall control architecture.

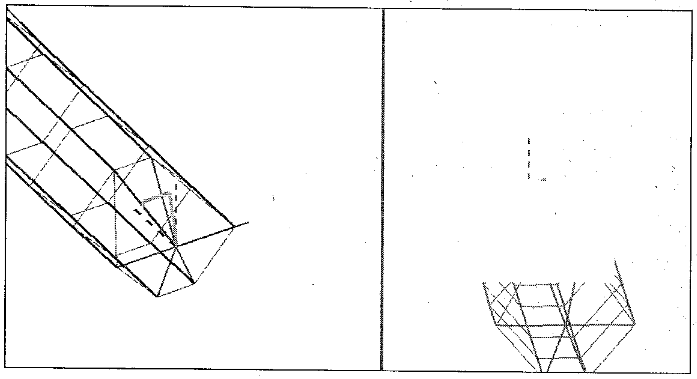


Figure 10. The two separate views in the real-time graphical interface allow the interface to be used as a teleoperated input device.

Real-time Graphical Interface

Rather than explicitly define the trajectory which the robot is to follow, an operator may wish to simply specify a start and stopping point for global stepping motions. To this end, we have developed a real-time graphical interface.

The graphical user interface is a PHIGS and XView-based application which runs as a remote task module. It has been designed to perform the following functions:

- It provides a 3D display of the robot position, configuration, and its location on the space station truss structure. Ambiguities in the 3D display on the 2D screen are resolved by providing two separate, modifiable views as shown in Figure 10.

- It allows for manually controlling task sequencing in the state machine in real time.
- It serves as a teleoperation input device for controlling global robot motions.
- It allows for visually pre-planning and simulating robot stepping motions to avoid obstacles and singular or near singular configurations.
- It serves as visual feedback to an operator by providing a global view of the robot on the space station truss. In addition, it warns of potential collisions by sending appropriate messages to the state machine. The operator can thus modify the robot trajectory accordingly.

In teleoperation mode, the graphical display translates mouse commands into trajectories in real-time. Once again, teleoperation and autonomous function are combined through the shared control structure. After the operator specifies desired steps for the robot, the same subtasks which perform autonomous walking are employed.

CONCLUSION

The SM^2 robot has been redesigned to be compatible with the new space station truss structure. Both the software and hardware of the SM^2 system has been designed to be modular, in order to shorten repair, maintenance, and development time. We have demonstrated both autonomous walking as well as teleoperation functions in a single shared control architecture. Depending on the calibration errors, the model-based locomotion with off-line trajectory planning, and neural-network based vision can be used for reliable walking. The real-time graphics interface provides a valuable tool for specifying control inputs in teleoperation and for displaying the robot configuration under communication delay. The free-flying hand controller provides an easy way to command robot action with two monitor views from the robot cameras.

ACKNOWLEDGMENTS

We would like to thank the following colleagues for their technical contribution and support: Ben Brown, Shigeru Aoki, Alex Douglas, Randy Casciola, Mark Friedman, Tetsuji Yoshida, and Takeo Kanade.

REFERENCES

[1] "The Bird: Position and Orientation Measurement System," Ascension Technology Corporation, 1991.

[2] A. Douglas and Y. Xu, "Real-Time Shared Control System for Space Telerobotics," in *Proceedings: 1993 IROS Conference*, pp. 2117-2122.

[3] D. Pomerleau, "Neural Network Perception for Mobile Robot Guidance," Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, 1992.

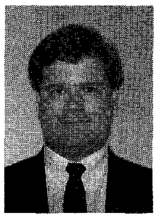
[4] D. Stewart, D. Schmitz, and P. Khosla, "The CHIMERA II Real-Time Operating System for Advanced Sensor-Based Control Applications," in *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, no. 6, pp. 1282-1295, 1992.

[5] Y. Xu, et. al., "Teleoperated Mobile Manipulator for Space Station," in Proceedings: 1993 *JSME International Conference on Advanced Mechatronics*, pp. 830-835, 1993.

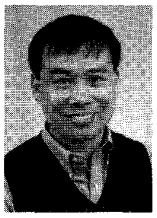
[6] Y. Xu, et. al., "Control System of the Self-Mobile Space Manipulator," in *Proceedings: 1992 IEEE International Conference on Robotics and Automation*, pp. 866-871, 1992.

[7] Y. Xu and M. Nechyba, "Fuzzy Inverse Kinematic Mapping: Rule Generation, Efficiency, and Implementation," in *Proceedings: 1993 IROS Conference*, pp. 911-918, 1993.

[8] M. Nechyba and Y. Xu, "SM2 for New Space Station Structure: Autonomous Locomotion and Teleoperation Control," in *Proceedings: 1994 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1765-1770, 1994.



Michael C. Nechyba was born in Vienna, Austria in 1970. He received a B.S. degree in electrical engineering from the University of Florida in 1992. Since 1992, he has been working on a Ph.D. degree in robotics at Carnegie Mellon University. His research interests include the learning and transfer of human control strategy, dynamically stable systems, neural networks, Hidden Markov Models, and applications in space robotics and teleoperation. Mr. Nechyba is an NSF Graduate Fellow, as well as a DOE Doctoral Fellow.



Yangsheng Xu has been working in the areas of manufacturing and automation, real-time control, and robotics for about 16 years. Early on, he worked in university and research laboratories on dynamics, system identification, and manufacturing process control. In 1988, he developed a robotic compliant wrist system with both passive compliance and sensing mechanism, and implemented the hybrid position/force control schemes in a robot assembly system, at the University of Pennsylvania where he received his Ph.D. degree in 1989. He joined the Robotics Institute of Carnegie Mellon University the same year and has been leading the research effort in the area of space robot control. He developed several space robots and dynamically stable robot systems in the space robotics laboratory that he established in 1990 including the space station robot walker called Self-Mobile Space Manipulator (SM^2). At the space robotics laboratory, he developed two gravity compensation systems to simulate zero gravity environments, and a variety of novel control schemes and real-time control architectures for space telerobotics. He has also contributed to advances in intelligent control in the area of learning human control strategy and automatic transferring of human skill to robots, based on neural networks and hidden Markov model representations. More recently, he has been developing novel dynamically stable robot systems, including underactuated robots with passive joints and one-wheel gyroscopically stable robot.

Dr. Xu has been a principal investigator in several projects funded by industry and government agencies. He has published more than 30 papers in refereed journals, more than 50 papers in conference proceedings and books, and one book on dynamics and control in space robotics. He has been selected as a keynote speaker at several international conferences and a senior technical advisor for the United Nation Development Program. He is a senior member of the American Institute of Aeronautics and Astronautics.