Model-based Telerobotics with Vision

John E. Lloyd, Jeffrey S. Beis, Dinesh K. Pai, David G. Lowe
Dept. of Computer Science, University of British Columbia
Vancouver, B.C., Canada
{lloyd,beis,pai,lowe}@cs.ubc.ca

Abstract

We describe an implemented model-based telerobotic system designed to investigate assembly and other tasks involving contact and manipulation of known objects. Key features of our system include ease of maintaining a world model at the operator site and a task-centric operator interface. Our system incorporates gray-scale model-based vision to assist in building and maintaining the local model. The local model is used to provide a task-centric operator interface, emphasizing the natural and direct manipulation of objects, with the robot's presence indicated in a more abstract fashion. The operator interface is designed to work with widely available and inexpensive desktop computers with low DOF input devices (such as a mouse). We also describe experimental results to date which include performing assembly-like tasks over the internet.

1 Introduction

Model-based telerobotics, sometimes also referred to as teleprogramming [7], has recently been proposed as a means of overcoming the problems of time-delay and/or limited bandwidth between a teleoperated manipulator and an operator control station. Under this framework, an operator interacts with a model of the remote site, rather than with the remote site directly. This interaction is in turn used by the system to generate motion and task commands which are transmitted to the remote site. Besides overcoming time-delay, model-based teleprogramming systems permit other advantages, such as operator control of the viewpoint, the ability to test and preview actions, and the introduction of artificial graphical and kinesthetic aids for task specification. More generally, they provide the opportunity to raise the semantic level of the interaction between the operator and the manipulator system.

However, there are several problems that must be resolved before telerobotic systems can become more widely used. This paper describes a model-based telerobotic system developed at the University of British Columbia to address some of these problems. We focus on two of the most critical: ease of maintaining a useful world model, and development of natural, task-oriented interfaces on widely available platforms.

Model acquisition and update. A central problem in model-based telerobotic technology is obtaining and maintaining accurate models of the remote site. Our system facilitates this using a fast gray-scale vision system at the remote site which can recognize objects of known type and return their spatial positions to the operator.

Task-oriented interfaces. Existing telerobotic systems have a tendency to be somewhat "robot-centric", in requiring the operator to specify tasks by controlling a virtual robot within the local model. We believe that the existence of a local model provides the opportunity to specify tasks more directly and in ways that may be more intuitive for the operator. Correspondingly, the "presence" of the remote manipulator in the task specification process can be reduced, as long as the constraints it places on task execution are still clearly conveyed.

A deliberate decision was made to base the operator interface on inputs from a simple 2D mouse, because such devices are both inexpensive and ubiquitous. Relatively expensive, specialized equipment will always have to be located at the remote site, but avoiding the need for specialized equipment at the operator site reduces costs and greatly increases operator site "portability". Similar considerations are expressed in [20].

The system described below has been used successfully in a series of demonstrations in which the operator site was located in Montreal, Canada, the remote site was located in Vancouver, and communication between the two was effected through the Internet. The operator interface was a low-end Indy workstation with no special hardware. Nevertheless, users were able to successfully perform manipulation tasks involving physical contact at the remote site with only a couple of minutes of training.

The remainder of this paper is organized as follows: Section 2 gives an overview of previous work. A general description of the system and its hardware is given in Section 3. Section 4 presents the vision system and its model-based recognition algorithm. The task specification interface and related topics are covered in Section 5. Finally, Section 6 discusses the system's performance in the above-mentioned Montreal-Vancouver demonstration.

2 Related Work

Teleoperation has a long history that predates robotics itself; a good overview is given in [21]. Inserting environmental models into the control loop to insulate against time delay is a more recent concept. Predictive graphical displays have been used, particularly in space-based teler-

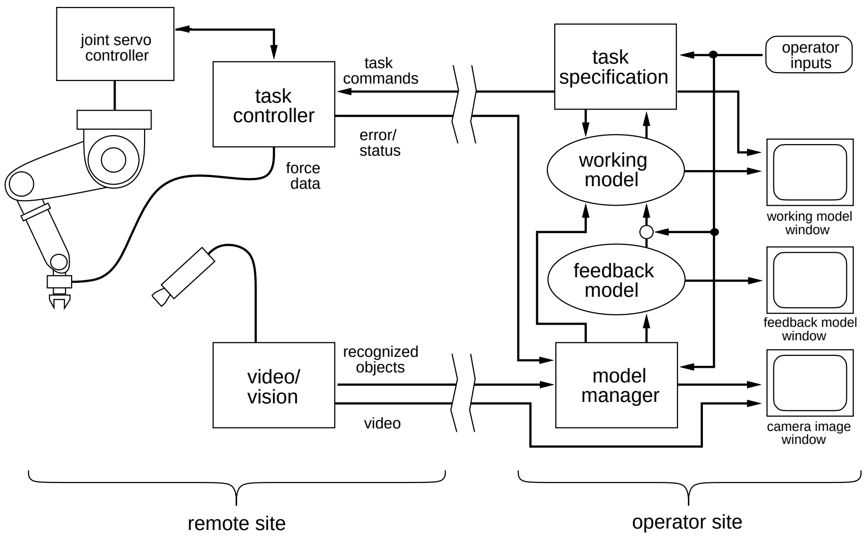

Figure 1: System architecture.

obotic applications [11, 4]. Extending the use of local models to include kinesthetic as well as visual feedback is described in [7]. A behavior-based teleprogramming system is described in [22]. Various researchers have also considered introducing synthetic aids for task specification into the operator's display [19, 15].

Commands sent to a remote site in teleprogramming systems tend to be at the level of "guarded moves". The ability to send higher level commands asking the remote manipulator to achieve a particular contact state (recovering if necessary from any intervening contact states which can be anticipated) is investigated in [5], using a Petri-Net-based contact state model.

Less attention seems to have been paid to the problem of initializing and maintaining the local model. A traditional technique, also included in our system as a backup, is to have the operator manually indicate known object features in a video image of the remote site, and use the 2D image coordinates of these features to solve for the 3D positions of the associated objects [16, 12].

Finally, in the last couple of years there have been a number of projects making teleoperated robotic systems of various kinds available to casual users on the World Wide Web [17, 8, 23].

3 System Overview

The system (Figure 1) consists of an *operator site* and a *remote site*, connected by communication links implemented as TCP/IP sockets. This allows communication to be routed through the Internet to create various time delays and bandwidth limitations.

3.1 Remote site architecture

At the remote site there is a *video/vision* module which continuously collects camera images and processes them using a model-based vision algorithm to locate objects in the scene. A list of the objects found, along with their spatial positions, is then transmitted back to the operator site. The camera image itself is also compressed and transmitted back to the operator site, where it is displayed in a separate window. If bandwidth is limited, the full image will likely arrive more slowly than the recognition data, allowing the latter to be thought of as a sophisticated form of compression. The video/vision module runs asynchronously with respect to the rest of the system.

A *task controller* takes motion and manipulation commands from the operator site, executes them, and returns status and error information. It is implemented using the Robot Control C Library (RCCL) [13], and is divided into an asynchronous *supervisor program* and a 100 Hz *trajectory generator*. The former resolves operator site commands into individual motions and analyses their performance, while the latter performs the inverse kinematics, motion interpolation, and smoothing required for each motion, implements a position-based impedance control similar to that described in [18], and monitors motion termination conditions. Output from the trajectory generator consists of a stream of joint position setpoints which are tracked by a 1 Khz joint servo controller.

3.2 Operator site architecture

At the operator site, there is a *working model* of the remote site environment, with which the operator interacts

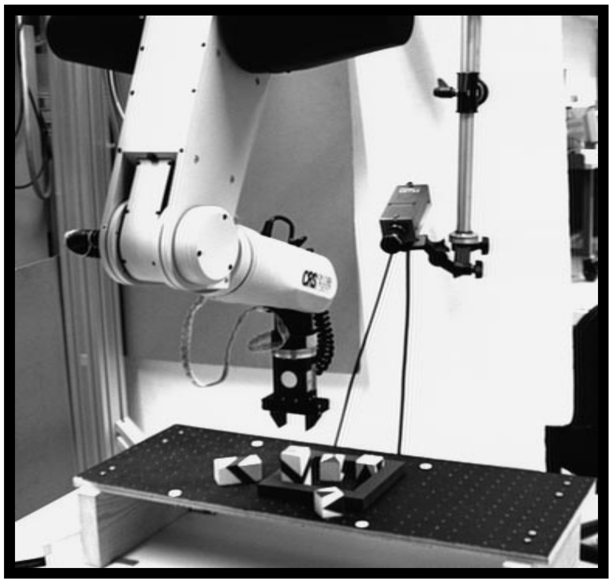
Figure 2: Remote site, showing the robot, camera, and work area.

(via a *task specification* module) in order to specify remote tasks. The model consists of a polyhedral representation of the work space objects, plus kinematic and geometric information about the remote manipulator (dynamical information is not necessary for the low-speed contact operation presently being investigated). Other information about the remote objects, such as friction and stiffness models, may be added later if required. Internally, the objects are stored using structures provided by the SGI 3D modeling package Open Inventor [25]. Geometrical relationships between objects are described using an Inventor scene graph. Inventor was chosen because of its extensive capabilities for graphical rendering and viewing, its widespread availability, and the possibilities for migration to VRML.

At present, object dimensions are assumed to be fixed, and so the model information that needs to be determined on-line is limited to object existence and spatial positions. This can be obtained in three ways:

A. *automatic reporting* by the vision system;

B. *manual reporting*, using operator indicated object features in the video image to solve for object position [12];

C. *contact inference*, using manipulator position and contact state to constrain object positions.

A *model manager* administers these information sources and reconciles differences between them. It also uses known constraints (such as the fact that objects close to the work space table surface must be resting on the surface) to reduce positioning errors.

Information from source A is first deposited into a separate *feedback model*, which has its own Inventor-based graphic window and can be viewed at any angle by the operator. Wireframe renderings of the objects can also be overlaid on the camera image returned from the remote site. At the request of the operator, the contents of the feedback model are added to the working model. In the event of spurious results by the vision system, the operator can graphically edit the objects (by “clicking on them”) and select particular ones to be added. Correspondence between the two models is not presently maintained. Instead, the operator can also graphically edit the contents of the working model, deleting some or all objects to make room for the updates.

Information from source B, provided as a backup to source A, is under operator control and so is deposited directly into the working model. Source B is also used (for now) to initially determine the position of the robot and the remote site table top.

Source C has only been implemented in a very rudimentary fashion. For instance, the z coordinate of the remote site table top can be determined by a manipulator guarded move.

3.3 Hardware details

The video/vision module executes on an SGI Indy rated at about 20 Mflops. The task controller runs on a SUN SparcStation V under Solaris 2.X with 48 Mbytes of memory (Solaris has sufficient real-time capabilities to run the RCCL trajectory generator at 100 Hz.). We use the 1 Khz PID joint servo controller supplied by the robot manufacturer, connected to the Sparc V via a direct link.

The robot itself is a CRS A460, which has a PUMA-like geometry without the shoulder offset. Force information is obtained (at 100 Hz) using a wrist-mounted Zebra force torque sensor, with a connection into the Sparc V system. Part grasping is accomplished using a parallel jaw electrically actuated gripper. Figure 2 shows a photo of the remote site work area.

Another SGI Indy serves as the operator station. Four windows are used: two for displaying the working model and feedback model, one for showing the camera image, and another for textual interaction (Figure 3). As mentioned in the introduction, operator inputs for manipulating objects are obtained using a standard 2D mouse.

4 Vision System

The vision system, which is itself model based, must be able to provide accurate identification and location of objects in the work space. Model-based vision has only recently developed to the point where these capabilities can

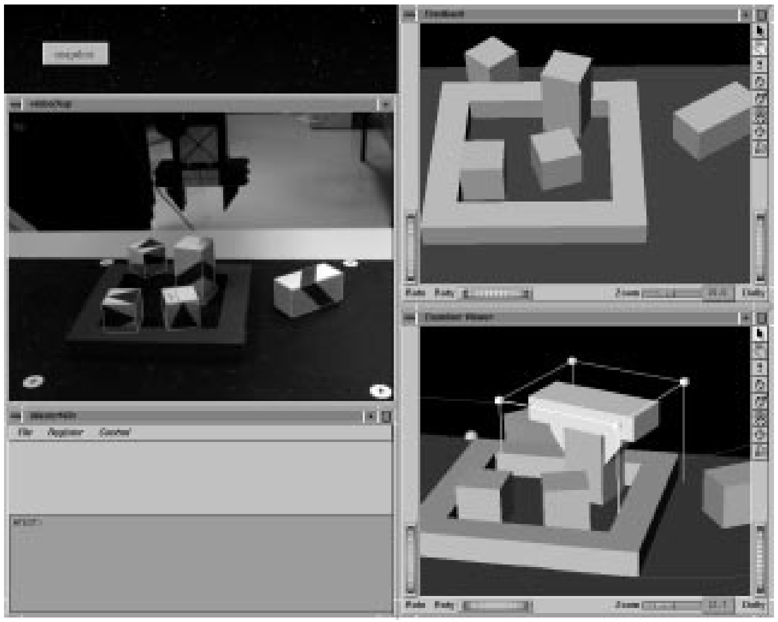

Figure 3: Screen image on the operator site workstation. Clockwise from bottom-right: working model window (with tall block being manipulated by a dragger box), feedback model window, camera image window, and textual interface window. Note the partial occlusion of several of the recognized blocks.

be achieved with reasonable speed and reliability. This project employs the model-based recognition system of Beis and Lowe [3], which uses a novel form of rapid indexing to recognize 3-D objects from any point of view in single 2-D images.

The vision system is currently restricted to using straight edges of the objects in the recognition process. A model must be provided for each object type which specifies surface visibility and the 3-D location of all prominent lines and edges. For the demonstrations described in this paper, these models were generated by hand. However, a separate tool has also been developed that allows models to be automatically generated from a number of images of an object, with human input limited to pointing out corresponding edges in the different images [2].

4.1 Recognition and matching

The recognition process begins by finding all linear edges in an image and identifying groups of edges that are connected to one another or are nearby and parallel. Groups of 4 or more line segments (see Figure 4) are used to generate a vector of measurements giving the relative lengths or angles between the lines. This “index” vector is invariant to 2-D translation, rotation and scaling, but will vary with the projection of different 3-D rotations of the object. In contrast, most other current methods require fully invariant feature groupings, which severely limits the type of object that can be used. A precomputed index covering a sample of all 3-D object rotations is used to estimate the probability that a particular vector was produced by a particular object. Full details of this indexing approach are given in the paper by Beis and Lowe [3].

Once a tentative interpretation has been made for some image features, it is possible to estimate the object location and orientation in 3-D [14]. This is used to predict the locations of other object edges in the image and obtain further correspondences. At each stage, the solution for object location and orientation in 3-D is performed with a least-squares fit minimizing residuals in predicted versus actual image locations. So, while the current system uses only straight edges, models might easily contain other feature types with location information, to aid in verification and pose determination. The solution for object pose is substantially over-determined, which means there is little likelihood that an incorrect correspondence will be found to fit more than a few image features. If a good fit is not found for a number of image edges, then the match is rejected and a new indexing hypothesis is used. Therefore, the final recognition has good robustness and accuracy, even though the initial indexing is probabilistic.

4.2 Speed, accuracy, and calibration

The full recognition process currently requires about 5 seconds running on the SGI Indy described in Section 3.3. This will be much improved in the future once a number of optimizations have been made for speed. Much of the time

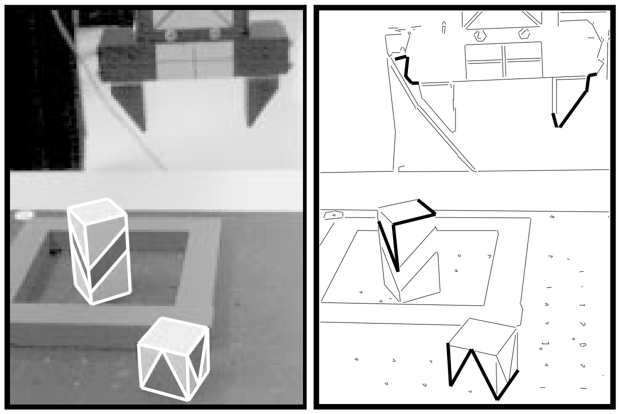

Figure 4: Left frame shows cropped image of work space overlaid with wire-frame models at positions determined by the recognition algorithm. Right frame shows edge-detected image with examples of correct and incorrect feature groupings used in the indexing process.

is currently spent on the low-level edge detection process, which could greatly accelerated by using some image processing hardware.

Image lines are determined through a least-squares fit to each pixel along an edge, and model location is based on a least-squares fit to these lines. Therefore, accuracy is usually precise down to the pixel level of the image, although its mapping to the 3-D world depends on the camera location and optics. With a single camera, the location of the object parallel to the camera image plane is more precise than location towards and away from the camera. If this is a problem, then it would be possible to use a similar approach to recognition with 2 or more cameras to achieve full accuracy in all dimensions. For this project, accuracy was improved using other constraints, such as the fact that objects close to the worktable surface must in fact be resting on the surface.

A standard pin-hole camera model is assumed. Intrinsic parameters (focal-length and radial distortion) were calibrated off-line using the algorithm in [24]. The camera position relative to the work space was calibrated manually by having the operator identify, in the camera image, workspace features of known position.

5 Task Specification

Our aim has been to create a task specification interface that is both intuitive and "task-centric." To the extent possible, we would like to support the operator's mental model of directly manipulating objects relevant to the task, with minimal emphasis on the remote manipulator.

The principal tasks to be performed are part placement and part mating. These may be specified by having the operator graphically "select" and move a desired part within the display of the working model. Based on the operator's actions, appropriate grasping and repositioning commands are generated and sent to the remote site. Generally this occurs at the request of the operator. For example, after a part has been satisfactorily repositioned within the working model, the operator can confirm (with a single mouse click) that the system should generate the appropriate remote commands for grasping and repositioning.

Direct manipulation of objects in the model does not preclude the need to restrict the operator to tasks which the remote manipulator system can actually perform. In particular, selected parts must be graspable, and goal positions (plus the intervening path) should not penetrate obstacles in the workspace and must be physically realizable by the manipulator.

Graspability is ensured when the operator selects a part: the face on which the operator clicks in order to do the selection is used to indicate the grasp; if this leads to an infeasible grasp or a collision between the robot and the environment, the selection is disallowed.

After the part is selected, a rendering of the gripper assembly appears around the object, along with a *dragger box* (see Figure 5). As the part/gripper combination is dragged around the working model, any motions which result in penetrating collisions with the environment are disallowed. Keeping the robot away from joint limits and singularities (or collisions involving more proximal parts of the arm) could be achieved similarly, though at present these restrictions are handled by confining actions to a well-behaved region of the manipulator's workspace.

Figure 5 shows a typical example of this interaction.

5.1 Dynamics for User Interaction

Our means of task specification entails having the operator directly manipulate the positions of solid models. In general, this is not an easy problem, particularly when contact is involved. The problem is further complicated by our deliberate decision to use an ordinary 2D mouse as an input device.

Fortunately, object positioning can be made much easier be endowing any part being manipulated with simple dynamics that allow it to "slide-around" obstacles in the workspace. For instance, to place a part exactly in a corner by positioning alone is tedious (particularly when trying to get the orientation correct). On the other hand, with our dynamics model, one can simply slide the part into the corner and it will automatically align and seat itself. Being able to exploit the natural constraints of the workspace in this way is important. Some authors have implemented "virtual fixtures" within teleprogramming systems for achieving similar ends [19]. Here, our aim is to use the environment itself as a fixture, which has a natural advantage in terms of being intuitive for the operator to use.

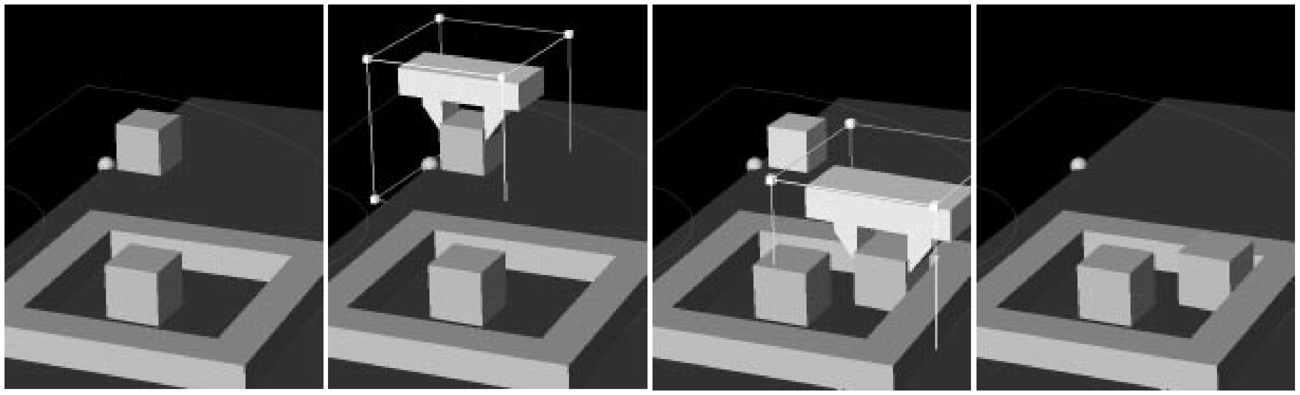

Figure 5: Example of user interaction. A part can be selected by "clicking" on it with a mouse. A rendering of the gripper assembly is then placed around it, along with a graphical Inventor object known as a "box dragger" (second frame from left). The operator can then "click" on different faces or edges of the dragger and "drag" the object to other parts of the scene, using 2D mouse movements which are mapped into suitable planar or rotational motions within the scene depending on which face or edge of the dragger box was picked. A "ghost image" marks the original object position (third frame from left), until the operator confirms the repositioning action, at which point the ghost, gripper, and dragger are removed.

We found that a first order dynamics model (as opposed to a second order dynamics involving inertia), with frictionless rigid body contacts, results in motions that are stable and easily understood by the operator. They are also easier to compute; for instance, there is no need to emulate Coulomb friction to bring objects suddenly to rest. We emphasize that the purpose of this dynamic model is *only* to assist in the interaction and not to simulate the dynamics at the remote site. Recall that in the teleprogramming framework, the actions performed by the operator are not sent directly to the remote site, but are translated into remote site commands which achieve the same goal state.

Computing the required interaction dynamics require that we be able to detect penetrating collisions between objects, as well as determine the distances (and closest features) between objects which are not colliding. This is achieved by modeling all objects as the union of a set of convex objects and then using the package I-COLLIDE [6].

The interaction dynamics are determined as follows: motion is initiated by clicking on a particular point of the dragger box containing the gripper and object. Let the three-dimensional point so selected be called the *grab* point $\mathbf{x}_g$. Subsequent mouse positions are mapped into another three-dimensional point $\mathbf{x}_p$, called the *pull* point. Points $\mathbf{x}_p$ and $\mathbf{x}_g$ are then assumed to be connected by a virtual spring, creating an applied force

$$\mathbf{f}_a = k(\mathbf{x}_p - \mathbf{x}_g). \tag{1}$$

This force acts on the body at $\mathbf{x}_g$. As for obstacles, if the gripper/object is located sufficiently close to another object, it is assumed to be in contact with that object. Information returned by I-COLLIDE can be used to determine a suitable finite set of contact points $\mathbf{p}_i$ and normals $\mathbf{n}_i$ modeling all the contacts[1]. Forces $\mathbf{f}_i$ acting along the contact normals in reaction to the applied force $\mathbf{f}_a$ are determined using Baraff's algorithm [1]. The net force and moment acting at $\mathbf{x}_g$ is given by

$$\mathbf{f} = \sum_i \mathbf{f}_i + \mathbf{f}_a, \quad \mathbf{m} = \sum_i (\mathbf{p}_i - \mathbf{x}_g) \times \mathbf{f}_i.$$

Object motion is then induced using a first-order dynamic model, so that if $(\mathbf{v}^T \boldsymbol{\omega}^T)^T$ is the spatial velocity at $\mathbf{x}_g$, then

$$\mathbf{v} = d_t \mathbf{f} \quad \text{and} \quad \boldsymbol{\omega} = d_r \mathbf{m} \tag{2}$$

where d_t and d_r are suitable constants. All these calculations can be comfortably performed in real time.

Within the graphical display system, motions are calculated incrementally using a fixed time step (generally 50 msec). Equation (2) is used to compute a desired linear displacement for the object/gripper at the beginning of each time step. If this displacement is found to result in a penetrating collision with an object, then the displacement is truncated at a point where collision first occurs (such a point can be found, to within sufficient precision, using a binary search along the proposed path). At the end of each time step, the set of contact points is updated.

5.2 Robot Programs from Operator Actions

The system is primarily designed for assembly-like tasks in which objects are repositioned and mated with other objects in the workspace. Certainly the positioning aspect of the task is easy to specify: the part should be placed where

[1]Face-face or edge-face contacts can be reasonably simulated using a finite set of point contacts; see [9].

the operator drags it. The problem, of course, is that a simple repositioning command may fail at the remote site due to modeling uncertainties. Instead, the remote site needs to execute a short robot program: a guarded grasp of the object; motion to a "free zone" away from obstacles; approach to a position near the goal; and object placement with a guarded move. We note that in general this is a complex motion planning problem, but is considerably simpler if the specific task domain allows simplifications (see, for example, [10]).

In addition to a standard set of free-space motion and gripper control commands, two basic repositioning program primitives have been implemented:

- ♢ *guarded grasp*, in which the geometry of the object is used to prepare a schedule of guarded moves that allow an object to be approached and grasped. This includes a simple search strategy to be triggered in the event of premature contact.

- ♢ *guarded place*, in which guarded moves and impedance control are used to place an object in a specified location.

At present, the guarded move schedules produced for these primitives do not allow for recovery from intervening contacts. In other words, the motions should succeed provided that the anticipated final contact state is the first one arrived at. We are currently experimenting with more elaborate planning techniques to handle intervening contacts; this problem is also considered in [5].

Upon completion of each remote command, an acknowledgement is returned to the operator site containing the command completion status and the current robot state. An error is assumed to have occurred when a command does not meet the anticipated termination predicates. Further operations are then discarded until a reset command is sent by the operator. When the operator receives error notification, she may use video imagery and remote object recognition data to determine the nature of the error and edit (if necessary) the working model. A more elaborate error recovery procedure, involving the automatic unwinding of actions within the working model to the point where the error occurred, is described in [20]. We have not (yet) found this to be necessary because the operator can easily use the feedback model to correct the working model.

6 Operational Results and Discussion

This system was successfully demonstrated at the 1996 PRECARN meeting in Montreal, with the operator site located there, and the remote site in Vancouver. Communication was effected through the Internet.

Meeting attendees were invited to use the system to reposition a set of blocks located at the remote site. Typically about four or five blocks of the type shown in Figure 4 were present at any given session. Between operator sessions, the blocks were rearranged randomly by an attendant at the remote site. Rectangular blocks were used to simplify grasping and contact operations, not to provide simplicity for the vision system. In fact, the vision system tends to perform better on more complex objects which have larger numbers of straight-line features.

Novice operators found the task interface quite easy and intuitive to use. There were occasional questions concerning the contents of the feedback model window compared with the camera image window, largely because the information in these two windows arrives at different rates. Operators also had an interesting habit of moving blocks to a part of the workspace out of view of the camera and then relying solely on the model to retrieve them.

Some mention should be made of calibration. A significant amount of the time spent developing the system was spent addressing calibration issues, including the determination of camera intrinsic parameters and the development of procedures for calibrating the remote site camera and robot positions (procedures which are manual but could be automated using the vision system itself). The robot kinematics also had to be calibrated, since the nominal kinematic model resulted in positioning errors of up to 1.5 cm over a workspace about 60 cm in length (not unusual for serial manipulators). However, this error could be reduced to about 1 mm using a local linear correction.

Some comments should also be made about the performance of the vision system. In general, it exhibited a significant degree of robustness, as there were no special lighting arrangements and the recognition algorithm tolerated a moderate degree of occlusion. Indexing did occasionally fail due to occlusion and/or noise, if no appropriate feature grouping were detected for a particular object. Full robustness in this area can be achieved by indexing with several types of grouping in parallel. Modules using smaller, more local groupings can then succeed in the difficult cases of major occlusions, at a penalty of slightly increased processing time, with the larger groupings providing better efficiency in the typical cases. During an early instantiation of the demonstration, it was noted that sometimes objects were falsely detected, due to the simplicity of the verification criterion (50% of visible feature lengths matched). More sophisticated criteria should indeed be used, although this is rarely a problem when the objects have greater complexity. For the PRECARN demonstration, false positives were eliminated using constraints on the range of the robotic arm, by pruning objects far from the manipulator's workspace.

7 Conclusion

We have successfully demonstrated the practical ability of using gray-scale vision techniques to provide world model construction and maintenance for model-based telerobotic applications. While improvements in the generality, speed, and accuracy of the vision system would certainly be useful, the system has already shown its utility in the moderately simple applications described above. It provides an enormous saving in time over the alternative method of model updating, which is to have the operator manually indicate features in the camera image. Robustness of the method can be guaranteed by allowing the operator control over the model update process (by editing the contents of the feedback model into the working model), and by providing a manual system as a backup.

We have also successfully demonstrated the ability to easily specify part positioning tasks, in cluttered environments, using operator inputs from low degree of freedom devices such as a 2D mouse. This was done using various graphic aids (e.g., dragger boxes) combined with appropriate 2D to 3D mappings, acting in combination with a first-order dynamic behavior which allowed the part being manipulated to slide along and be re-oriented by contacts made with other workspace objects.

Our intention in designing this system is to experiment with more "task-centric" ways of specifying telerobotic tasks. We believe that the environmental model inherent in the teleprogramming paradigm offers significant opportunity to simplify the programming of robotic tasks in general, and that the utility of model-based telerobotics extends well beyond teleoperation situations involving time delay or limited bandwidth.

Acknowledgement

The authors would like to thank Jane Mulligan, Rod Barman, Stewart Kingdon, and Brian Gilgan. This work was supported in part by the Institute for Robotics and Intelligent Systems (IRIS projects IS-6, HMI-6, and IC-1), the BC Advanced Systems Institute, and the National Sciences and Engineering Research Council (NSERC).

References

[1] D. Baraff, "Fast Contact Force Computation for Nonpentrating Rigid Bodies", *SIGGRAPH 94 Conference Proceedings*, July 1994, pp. 23–34.

[2] J. S. Beis, "Building models with planar faces using a structure-from-motion algorithm plus a small amount of post-processing". Technical Report number TR-96-14, Computer Science Department, University of British Columbia, June 1996.

[3] J. S. Beis and D. G. Lowe, "Learning indexing functions for 3-D model-based object recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle (June 1994), pp. 275-280.

[4] A. K. Bejczy, W. S. Kim, and S. C. Venema, "The Phantom Robot: Predictive Displays for Teleoperation with Time Delay". *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, May 1990, pp. 546–551.

[5] Y. J. Cho, T. Kotoku, and K. Tanie, "Discrete-Event-Based Planning and Control of Telerobotic Part-Mating Process with Communication Delay and Geometric Uncertainty". *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Pittsburgh, Pennsylvania, August 1995, pp. 1–6 (Vol. 2).

[6] J. Cohen, M. Lin, D. Manocha and K. Ponamgi, "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scaled Environments". *Proceedings of ACM Int. 3D Graphics Conference*, 1995, pp. 189–196.

[7] J. Funda, T. S. Lindsay, and R. P. Paul, "Teleprogramming: Toward delay-invariant remote manipulation". *Presence*, Winter 1992, pp. 29–44 (Vol. 1, No. 1).

[8] `http://cwis.usc.edu/dept/garden/`.

[9] S. Goyal, E. N. Pinson, and F. W. Sinden, "Simulation of Dynamics of Interacting Rigid Bodies Including Friction I: General Problems and Contact Model". *Engineering with Computers*, Springer-Verlag, London, 1994, pp. 162–174 (Vol. 10).

[10] T. Lozano-Pérez, J. Jones, E. Mazer, P. O'Donnell, W. E. L. Grimson, and A. Lanusse, "Handey: A Robot System That Recognizes, Plans and Manipulates". *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, April 1987, pp. 1713–1717.

[11] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-Based Space Robotics – ROTEX and Its Telerobotic Features". *IEEE Transactions on Robotics and Automation*, October 1993, pp. 649–663 (Vol. RA-9, No. 5).

[12] W. S. Kim and L. W. Stark, "Cooperative Control of Visual Displays for Telemanipulation". *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, May 14-19, 1989, pp. 1327 – 1332.

[13] J. E. Lloyd and V. Hayward, *Multi-RCCL User's Guide*. Technical Report, Center for Intelligent Machines, McGill University, April 1992.

[14] D. G. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 5 (May 1991), pp. 441-450.

[15] National Research Council (U.S.A.), *Virtual Reality. Scientific and Technological Challenges*. National Academy Press, Washington, D.C. 1995.

[16] E. Oyama, N. Tsunemoto, S. Tachi, and Y. Inoue, "Experimental Study on Remote Manipulation Using Virtual Reality". *Presence*, Spring 1993, pp. 112–124 (Vol. 2, No. 2).

[17] E. Paulos and J. Canny, "Delivering Real Reality to the World Wide Web via Telerobotics". *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, April 1996, pp. 1694–1699.

[18] M. Pelletier and M. Doyon, "On the Implementation and Performance of Impedance Control on Position Controlled Robots". *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, California May 8-13, 1994, pp. 1228–1233 (Vol. 2).

[19] C. R. Sayers and R. P. Paul, "An Operator Interface for Teleprogramming Employing Synthetic Fixtures". *Presence*, Fall 1994, pp. 309–320, (Vol. 3, No. 4).

[20] C. R. Sayers, "Operator Control of Telerobotic Systems for Real World Intervention". Ph. D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA, 1995.

[21] T. Sheridan, "Telerobotics, Automation, and Human Supervisory Control". MIT Press, Cambridge, Massachusetts, 1992.

[22] M. R. Stein and R. P. Paul, "Operator Interaction, for Time-Delayed Teleoperation, with a Behaviour-Based Controller". *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, California, May 1994, pp. 231–236.

[23] `http://telerobot.mech.uwa.edu.au/`.

[24] R. Tsai, “A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses”. *IEEE Transactions on Robotics and Automation*, August 1987, pp. 323–344 (Vol. RA-3, No. 4).

[25] J. Wernecke, *The Inventor Mentor*. Addison-Wesley, Reading, Massachusetts, 1994.