# Precise manipulation with endpoint sensing

by Russell H. Taylor
Ralph L. Hollis
Mark A. Lavin

**This paper describes recent work on manipulation strategies that rely on "coarse-fine" robot hardware and direct sensing of part-workpiece relationships. The experiments reported use an extremely precise, high-bandwidth planar "wrist" and an industrial vision system to perform accurate alignment of small parts. The system architecture, experimental hardware, and programming methods employed are all discussed.**

## Introduction

Industrial robots have traditionally been used as general-purpose positioning devices. In a typical application, the robot is moved through a sequence of positions so that a tool or part held by the robot achieves a desired relationship to a workpiece whose position is fixed or known in relation to that of the robot. Although textual programming methods and languages for robots have been around for some time [1], the overwhelming number of robots are still programmed by "teaching" a sequence of points under teleoperator control. When a robot is programmed in this manner, the most important requirement for successful accomplishment of its task is the *repeatability* with which it can return to the taught positions and with which successive workpieces can be presented to it. This method of using a robot is easily understood, and many people have developed considerable expertise in designing the necessary end-effectors, fixtures, and setup procedures to go with it.

Unfortunately, reliance on simple repetition of taught points is inadequate for several increasingly important classes of robot applications, especially automatic assembly and "data-driven" manufacturing, in which position goals must be computed directly from design data. Assembly tolerances are often at the extreme limit of a robot's repeatability, and small variations in parts can have a big effect on successful task completion. Part presentation equipment and workpiece fixtures often represent a significant fraction of the total system cost for an assembly robot. Even in cases for which teaching "works," the necessity of reteaching at least some points any time the robot or a fixture is changed (or even subjected to routine maintenance) can create significant operational difficulties.

Data-driven automation is especially important in the aerospace industry and in electronics manufacturing applications such as printed circuit card assembly and testing. These applications often require both the ability to align a part or tool very precisely relative to a workpiece and the ability to move through a large work envelope. The requirements are likely to become more and more stringent in the future.

A major goal of robot research and development over the past fifteen years has been finding techniques for improving the *effective accuracy* of the manipulator, i.e., the precision with which it can place a part or tool at a computed position relative to a workpiece. Several major themes have emerged from this work.

*Calibration methods* use sensing to measure and correct for the inaccuracies of the robot. A typical approach, e.g., [2], commands the robot to move nominal positions relative to an accurately constructed calibration fixture. Sensors are used to measure the corresponding alignment errors, and the data are then used to update a mathematical model of the robot. Although these techniques are frequently effective in improving manipulator accuracy, their usefulness in any particular application depends somewhat on the number of calibration points required, the dimensional stability of the robot, and the difficulty of installing the calibration fixture. A related technique is to use the workpiece itself as the calibration fixture. For example, the robots used in testing backpanel wiring used touch sensing to locate the corners of each printed circuit board. Commanded positions for features on a board were then computed by interpolation. This method has the advantage of automatically accounting for small variations in workpiece dimensions or orientation, but has the drawback of requiring that the robot spend time calibrating itself before beginning each job cycle. This is tolerable where the calibration time is short compared to the rest of the job (as here) or where sensing incidental to normal execution can be used to update a calibration model [3].

*Compliance methods* are widely used in automatic assembly. The robot simply moves through a nominal path, and the manipulator structure, the workpiece, or a cleverly designed mechanism, e.g., [4], provides the necessary "give" to make up for any positioning errors that result. Limiting factors for this approach include the design time and cost associated with special fixtures and the fact that different steps in a task may require different compliances. To get around these limits, there has been considerable attention to controlling the *force* exerted by the robot, rather than its position, and then relying on software to synthesize whatever effective compliance is required, e.g., [5–9]. Friction and inertia create practical difficulties for implementing these methods on existing robots. There has been considerable recent activity in building "direct drive" manipulators that substantially eliminate friction [10, 11], and in designing "wrists" with redundant actuators to avoid the inertias associated with "big" joints [12, 13]. These research activities may be expected to have a significant effect on future industrial robots, but they are not a panacea for applications which are not well adapted to force compliance. These difficulties are particularly relevant in electronics manufacturing, since electronic parts are often small and delicate and since it may not be possible to design parts with chamfers or other features to facilitate force-compliant assembly.

*Endpoint sensing methods* rely on sensors to measure part-workpiece or tool-workpiece misalignments directly and then move the robot accordingly. If force sensing is used, this is reduced to a compliance method. However, many other forms of endpoint sensing have been used, including vision [14–16], touch [14, 17], proximity [18, 19], and so forth. The principal factors limiting the alignment precision that can be achieved with endpoint sensing methods are the resolution of the sensing system and the motion resolution of the robot. The motion resolution of robots used in present-day electronics manufacturing (i.e., the size of the smallest incremental motions that they can reliably make) is typically a bit better than 0.1 mm. However, many future applications are likely to require precision on the order of 0.01 mm [16]. The difficulty of achieving these precisions with existing robot designs is exacerbated by the need to retain a large working volume and high motion speed.

The manipulation approach reported in this paper uses endpoint sensing to measure part misalignments and a fine-positioning wrist to get around the resolution limitations of the robot. This approach may be summarized as follows:

1. Use the coarse joints of the robot to bring the tool or part into approximately the desired position and orientation relative to the workpiece.
2. Use the fine-motion joints to null out the sensed misalignment.

Subsequent sections of this paper describe the architecture of our experimental system, the fine-positioning wrist, and two application experiments illustrating the approach.

## System architecture

### *Objectives*
Experience with an earlier system [20] and with a number of applications within IBM convinced us that the key problem with industrial robots was not so much manipulation as the integration of a broad spectrum of capabilities, including manipulation, sensing, computation, and connections to factory control systems and data bases [3]. Addressing this problem required a control computer and powerful software, providing three main classes of function: configurability, flexible user interfaces, and reliability.

### *Components*
The principal components of the system are a system controller, operator interface hardware, robot and sensor hardware, and system software. Each component is described briefly below.

#### *System controller*
The system controller is responsible for coordination of all activity at the robot workstation. It consists of an IBM Series/1 minicomputer, together with data processing peripherals and workstation interface electronics. Data

processing peripherals vary somewhat according to the particular application requirements but normally include such items as keyboard, display terminal, printer, diskette drives, hard disk, and teleprocessing attachments to other computer systems.

The workstation interface electronics performs a number of functions, including positional control of robot joints, safety interlocks, and robot power controls, input of sensor values, and output of control signals to miscellaneous devices at the workstation. The original Research and IBM 7565 implementations of these functions used nonprogrammable custom-designed electronics. The robot had analog position feedback, and joint control was accomplished by analog "PD" loops with some compensation for the nonlinearities of the hydraulic actuators.

In order to facilitate experimentation with more advanced control methods and to simplify problems associated with interfacing many different robots and sensors to the system, we developed a family of programmable attachments called "RRA cards." Each attachment has a standard base, consisting of a Series/1 channel interface, a 4K-byte shared memory, a Motorola M68000 processor, timers, miscellaneous support chips, and a custom sensor interface area. These attachments have been used for a number of robot and device control applications.

*Operator interface*
Operator interface hardware includes an operator's console on the robot and a hand-held pendant containing a small display and a number of switches and lights. Except for safety-related functions, the interpretation of all operator input/output is determined by application software.

*Robot and sensor hardware*
The key design objectives for both the robot and the sensors were modularity and configurability. The system architecture makes no assumptions about the kinematic structure of the manipulator, which it views simply as a collection of position-controlled "joints" together with associated power and safety interlocks. The system provides coordinated straight-line motion in configuration space, and kinematic transformations are handled by built-in subroutines.

This approach has been fairly successful in allowing us to use a number of different robot configurations with the system. The most common configuration has been a cartesian structure similar to the IBM 7565. Other configurations have included an IBM 7535 SCARA-type robot and a number of specialized structures put together for particular applications.

The present robot in our laboratory is a two-armed cartesian electric-drive manipulator developed for use in IBM clean room manufacturing. Each arm consists of three linear actuators providing $X$, $Y$, and $Z$ motions, three rotary actuators providing *Roll, Pitch,* and *Yaw* motions, and a gripper with linearly actuated fingers. For the application experiments described in this paper, the *Pitch, Roll,* and *Gripper* actuators of one arm have been removed and replaced with an extremely accurate planar wrist providing fine motions in the $X$ and $Y$ directions.

Sensors typically include force transducers and a light beam presence sensor mounted in the fingers, several solid state television cameras, and miscellaneous application-specific sensors, such as empty-feeder indicators.

*Software*
One component of the system controller software provides an interactive programming environment for a high-level programming language, AML [21, 22], which is used for all application programming. A second software component performs trajectory planning, motion coordination, sensor monitoring, and other real-time activity in response to commands from the AML interpreter. A third component supplies standard supervisor services, such as file and terminal I/O.

## • *AML language*

*Objectives and overview*
The principal programming interface to the robot workstation is the AML language [3, 21, 22]. AML was designed to be a well-structured, semantically powerful interactive language that would be good for robot programming. The central idea was to provide an expressive base language with simple subsets which would be usable by programmers with a wide variety of experience. Although the language primitives have been chosen to provide a natural way of describing manufacturing applications, we make a clear distinction between the language constructs and the semantics of the application environment. No special syntax is supplied for robotic concepts. Instead, all access to system functions is accomplished through calls to system-defined subroutines that are called exactly like those written by a user. This transparency provides a natural mechanism for system extensibility through the use of subroutine libraries for customization of the runtime environment for particular application domains.

*Language summary*
AML supports a number of "scalar" data types, including *INT*eger, *REAL*, and *STRING*, and provides the usual unary and binary operations on them. A number of auxiliary types useful in the construction of application subroutine libraries and debugging packages are also supported. The language supports ordered lists, called "aggregates," of scalar and/or aggregate AML objects. The language includes constructs, somewhat reminiscent of APL's [23], for generalized indexing of aggregates and for uniform mapping of scalar

operations over aggregate objects. Variables are typed and are declared by binding an identifier to the value produced by evaluating an expression. For example,

*var: NEW 2 + ⟨⟨1,2⟩,3.5⟩;*

would bind the identifier *var* to an aggregate data object whose initial value is ⟨⟨3,4⟩,5.5⟩.

The language is expression-oriented, in the sense that every legal AML construct produces a value which may be used as a part of some other expression. Expressions are evaluated left-to-right, and the grouping of expressions is determined by operator precedence and parentheses as in most common programming languages. The normal constructs of structured programming,

```
IF e1 THEN e2 ELSE e3
WHILE e1 DO e2
REPEAT e1 UNTIL e2
BEGIN e1; . . .; en END
```

are supported by the language and also produce values.

AML subroutine definitions have the general form

```
subr_name: SUBR (formal_1, . . ., formal_n);
  statement_1;
  statement_2;      -- Comments are preceded
        :           -- by a "--", as this
  statement_k;      -- example indicates.
END;
```

and are called by expressions with the general form

*subr_name (expression_1, . . ., expression_n).*

The language supports both "by value" and "by reference" passing of parameters and has a number of special constructs provided for building subroutine packages.

- *Vision system*

Our system supports attachment of up to four 128 × 128-pixel solid state binary cameras to the Series/1 controller. The programming interface, AML/V [24], is an extension of AML. It was designed to provide the power and ease of use of AML for machine vision application development, and to allow close coupling of robot control and vision sensing. The latter capability is very important if endpoint sensing is being used to improve robot accuracy, since communication delays can have a significant effect on system throughput and since application data may be used to provide input to the vision system.

AML/V images are stored as AML strings and may be packed binary, run-length-coded binary, or gray-scale representations of the TV input. System-defined subroutines provide a number of image processing functions. *Image I/O functions* allow the user to define "logical vision input devices," control their operating parameters, and acquire images. Additional I/O functions support display of images on raster output devices (such as TV monitors) and storage/retrieval operations on disk files. *Image-to-image* functions perform boolean operations on packed binary and run-coded binary images, image windowing and shifting on run-coded images, arithmetic operations on gray-scale images, and conversion between the various representations. *Image analysis functions* provide histogramming and binary region analysis functions.

## Planar fine-positioner

We decided to concentrate initially on precise motion in the *XY* plane, since many of the principles of coarse-fine motion could still be studied, and since a planar fine-positioner could be applied to several practical problems in electronic testing and assembly. We wanted a device that was fast, strong, and as free as possible from backlash, friction, or hysteresis.

We used a single direct-drive two-dimensional actuator rather than coupling a pair of linear actuators together to achieve two-dimensional motion. This parallel arrangement maintains symmetry between the *X* and *Y* axes, offering the advantage of nearly identical inertia for each axis, while eliminating the problems of serial kinematics stack buildup and resulting error accumulation.

**Figure 1** shows an overall view of the fine-positioner. The design combines four major elements: electromagnetic drive, flexure spring suspension, lateral effect device position sensor, and digital control system. Subsequent sections provide a brief description of each design element and summarize its operational modes. A fuller discussion of the design may be found in [25].

- *Design*

*Two-dimensional motor*

Many actuator technologies can be considered, including shape memory alloys, magnetostriction, piezoelectrics, pneumatics, hydraulics [12, 26], electrodynamics [13], and electromagnetics. The need for high actuation speed, contamination-free operation, large motion range, good stiffness range, and controllability narrowed the choice to either an electrodynamic or an electromagnetic drive. A further need to generate fairly high forces in a small package led to the selection of an electromagnetic drive based on the Sawyer motor principle [27]. The Sawyer motor is really a linear stepping motor which operates by permanent magnet flux-steering. In the fine-positioner device, it is used as a direct-drive analog positioner with a total range of one step.

*Flexure spring assembly*

A flexure spring assembly supports the moving armature without friction or backlash. This assembly allows motion in the *X* and *Y* directions, but is rigid in the *Z* direction and in torsion about *Z*. As the springs bend in a slight arc, some *Z* motion (worst case 11 $\mu$m) occurs which can be compensated for in most robotic applications. We are considering alternative flexure arrangements that would practically eliminate this deflection but that may be less compact.

*Position sensor*

A commercially available semiconductor lateral-effect cell [28] provides noncontact position sensing in two dimensions. A light-emitting diode (LED) attached to the moving armature produces a spot of light whose position is measured by the fixed lateral-effect cell. The position of the light spot on the cell surface is determined by measuring the generated photocurrents in four electrodes arranged on the periphery of the square active surface of the cell.

In addition to the built-in position sensor, the fine-positioner is normally operated in a manner which requires a separate external sensing means. External sensors can take many forms, including fiber optics, image sensors, or other means appropriate to the task, and may sometimes be used in place of or together with the internal sensor in servocontrol of the fine-positioner.

*Controller*

The fine-positioner open-loop response is very underdamped due to the frictionless suspension and can be closely modeled as a second-order complex pole with natural frequency $\omega_n = 17.5$ Hz and damping ratio $\zeta = 0.03$. After amplification, voltages proportional to the lateral-effect currents are sampled by a multiplexed 12-bit analog-to-digital converter (ADC) and an M68000-based microcomputer on the RRA card. The control algorithm running in the microcomputer computes the required control efforts from commanded and measured positions, establishing proper values of currents in the drive coils through a pair of 12-bit digital-to-analog converters (DAC) on the RRA card. We have experimented with $z$-transform direct design algorithms as well as analog and digital proportional integral derivative (PID) controllers. **Figure 2** shows the response obtained for large and small step position commands with a digital PID controller. Since peak dynamic performance depends greatly on the detailed characteristics of the servo controller, we are continuing to refine the algorithms.

- *Fine-positioner operation*

A brief listing of the fine-positioner specifications appears in **Table 1**. Our primary emphasis has been on using the device as a high-precision positioner attached to a general-purpose robot. Other modes of operation include use as a variable compliance device, variable forcing device, or measuring device.

We have already mentioned that the device can be used to execute rapid submicron motions in $X$ and $Y$ over a total

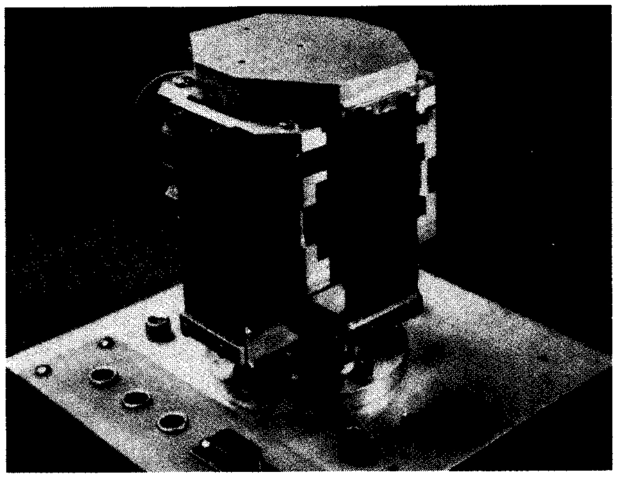


**Figure 1**

Two-dimensional planar fine-positioner.

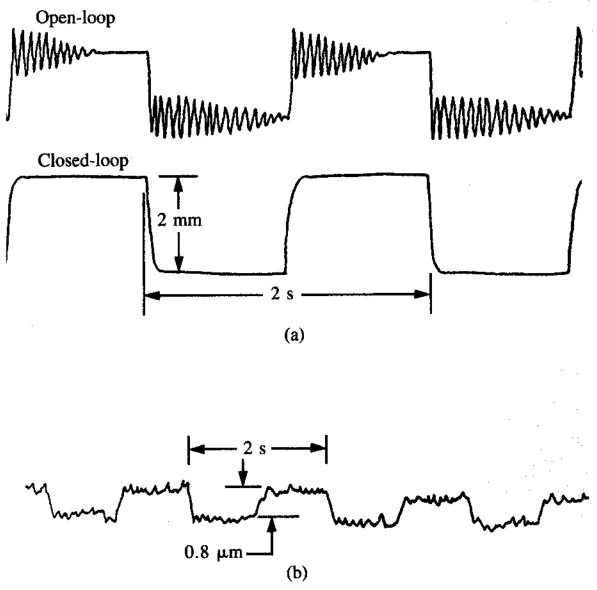


**Figure 2**

Position vs time trace: (a) large step response; (b) small step response.

**Table 1** Planar fine-positioner characteristics.

| | |
|---|---|
| Approximate physical dimensions | 76 mm on a side |
| Approximate mass | 1 kg |
| Force | 13 N |
| Motion range | ±0.9 mm |
| Positional resolution with internal sensor | 0.5 $\mu$m |
| Closed-loop bandwidth with PID controller | 52 Hz |
| Rise time (10% to 90%) for 1-mm move | 8 ms |

range of approximately 1.8 mm, which is large enough to accommodate most errors in robot positioning. The ability to execute fine motions has many applications in science and engineering as well as in robotics.

Alternatively, by varying the closed-loop gain parameter (in our implementation this is done simply by changing coefficients in the computer control program), the compliance can be varied over a range from much greater than to much less than the natural spring compliance.

In some applications, it is desirable to exert known forces on a workpiece. If negligible motion occurs, the force exerted on the armature is proportional to the coil current. For direct force measurement, an external transducer can also be mounted on the armature.

By using the built-in sensor, the device can be used as a passive measuring device, in a mode where the coil drive currents are disabled. Applications such as parts profiling can be accomplished by sensing the relationship between a mechanical probe or stylus attached to the movable armature and the fixed part of the device.

Since the device incorporates digital control, operation can be switched between the various modes described above as necessary to perform a given task.

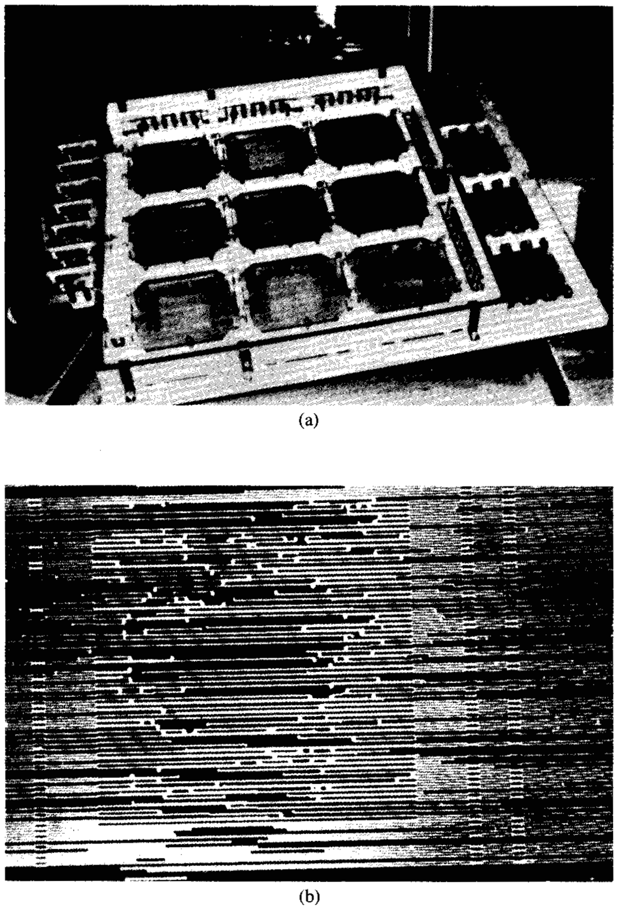
## Application experiments

### • *Trace probing*

*Application overview*

The electronic components of IBM mainframe computers are packaged in high-density thermal conduction modules (TCMs), each of which can accommodate up to 118 chips. The TCMs have 1800 I/O pins and plug into special high-density printed circuit boards ("TCM boards") with zero-insertion-force connectors. The TCM boards measure 600 mm × 700 mm and contain twelve power planes and six signal planes, each of which can contain several thousand signal lines [29].

In order to improve system reliability and to reduce the cost of reworking, it is very desirable to perform exhaustive


(a)

(b)

Figure 3

TCM board: (a) assembled board; (b) signal plane traces.

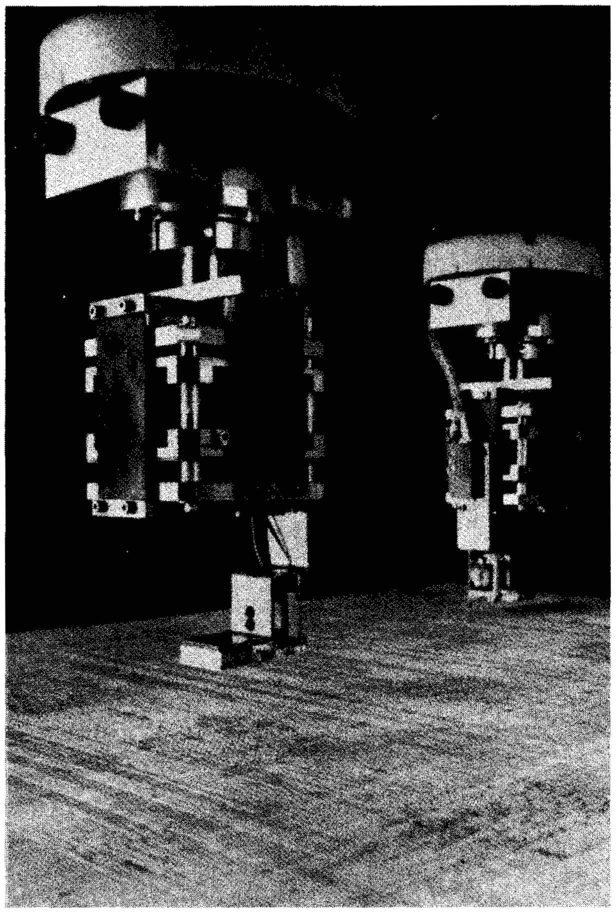


Figure 4

Robotic system for TCM-board probing.

electronic testing of the signal planes before they are assembled into the TCM board. Testing requires that electronic probes be placed at the ends of each signal line. If a defect is found, it must be localized by probing points along the line. The lines are only 80 $\mu$m wide and can be spaced as close together as 150 $\mu$m, as illustrated in **Figure 3**.

The probes must be placed within $\pm 20$ $\mu$m of the center of a line in order to avoid damage to the product. Maintaining this accuracy over a large area is quite challenging with a conventional robot. The problem is exacerbated by the manufacturing tolerances in producing the sublaminates [30]. Consequently, a coarse-fine manipulation strategy that relies on a robot to get the probes "close" to their targets and direct sensing of the lines coupled with fine manipulation to "home in" seems natural.

*System overview*

A robotic system for electronic testing of TCM boards is illustrated in **Figure 4**. The robot has two arms, each of which has three coarse actuators, providing $X$, $Y$, and $Z$ motions, and a fine-positioning mechanism providing fine $XY$ motions. Each fine-positioner carries an electronic test probe and a fiber-optic sensor for detecting circuit wires, as shown in Fig. 4.

The sensor is quite simple. Light from an LED source is passed through the optical fiber and shines on the TCM board. A photodetector then measures the amount of light reflected back up through the fiber. **Figure 5** shows the sensor output when the fine-positioner is used to scan it over two closely spaced TCM-board wires.

*Job cycle*

The application consists of two phases: a setup phase, in which a sublaminate is placed under the robot and an initial calibration is done to determine the transformation relating sublaminate coordinates to robot coordinates, and a test phase in which each wire is tested as follows:

1. Read design data for next wire.
2. Place the probes at the endpoints of the wire and conduct the test (see below). If the test is successful, go on to the next wire. Otherwise, conduct a binary search along the wire to determine the location of the defect to within a few millimeters. Note that the possibility of multiple defects in a single wire cannot be ignored.

The method for conducting each electronic test may be summarized as follows:

1. Using the board-to-robot transformation determined in the calibration phase, compute the robot coordinates corresponding to the circuit features to be probed.
2. Move both arms to these coordinates.

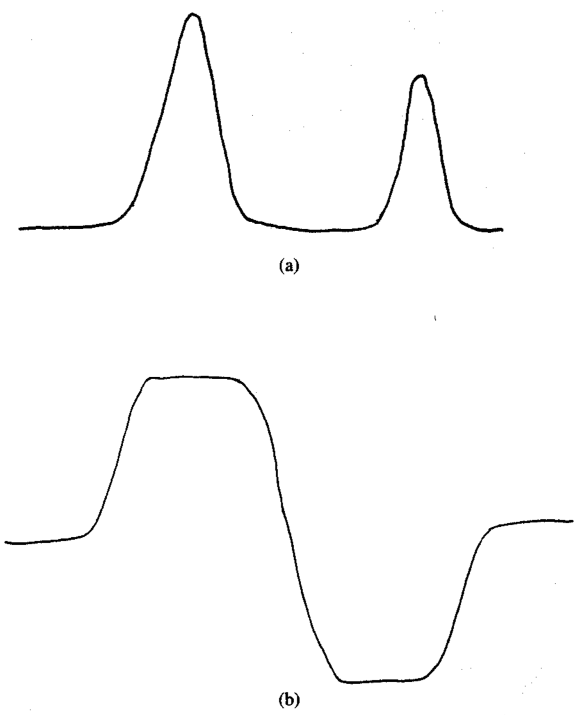


Figure 5

Fiber-optic sensor output: (a) shows the output of the fiber-optic sensor when it is swept past a pair of TCM-board wires spaced 0.15 mm apart. (b) shows the output of the differential sensor as it is swept past a TCM-board wire. The sensors and these figures were supplied by Mark Johnson, IBM Research.

3. Use the fine-positioners to scan the optic probes rapidly across the circuit features. Use the resulting data to determine the center of each feature.
4. Move each fine-positioner so that the electronic probes are centered over the features.
5. Conduct the electronic test.

*Probe placement sensitivity*

One difficulty often encountered with endpoint-sensing methods is that of finding an *independent* means of verifying that the method is achieving its desired result. We used the apparatus illustrated in **Figure 6** to investigate the ability of the fine-positioner and fiber-optic sensor combination to compensate for small variations in wire placement and robot inaccuracies. A lateral-effect cell was attached and a small piece of TCM-sublaminate was affixed to a micrometer stage. A probe attached to the fine-positioner armature carried both the fiber-optic sensor and an LED so arranged that the light spot from the LED fell on the lateral-effect cell when the fiber-optic sensor was centered over a sublaminate wire.

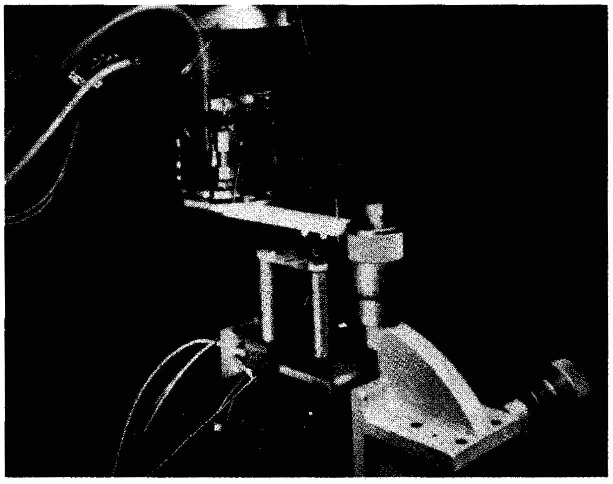


**Figure 6**

Probe placement sensitivity experiment aparatus: The micrometer stage is used to displace the small section of TCM board by known amounts, and the lateral-effect cell is used to measure the probe displacement after the wire is located.

**Table 2** Probe placement sensitivity experiment results: Five trials were made for each displacement value. Mean and standard deviation lateral cell readings are given.

| *Wire displacement* ($\mu$m) | *Lateral cell reading* ($\mu$m) |
|---|---|
| 0 | 730.1 ± 1.6 |
| 5 | 723.7 ± 2.1 |
| 10 | 730.9 ± 1.1 |
| 15 | 731.2 ± 0.9 |
| 20 | 725.1 ± 2.6 |
| 25 | 728.2 ± 1.4 |
| 30 | 731.9 ± 1.4 |
| 35 | 732.0 ± 0.9 |
| 40 | 732.9 ± 0.7 |
| 45 | 732.9 ± 0.7 |
| 50 | 732.5 ± 0.8 |
| 75 | 732.8 ± 0.4 |

The robot was then repeatedly moved to a nominal wire position, the optic probe was scanned over the wire, and the resulting data were used to center the optic probe over a line, as described in the previous section. The lateral-effect cell was then read to determine the relative placement of the probe to the sublaminate board. After five trials, the micrometer stage was used to displace the sublaminate by a known amount, and the process was repeated. **Table 2** shows the results obtained. The probe placement relative to the line varied by at most about 10 $\mu$m when the wire was displaced over 75 $\mu$m, well within the ±20-$\mu$m precision required by the application. The principal limiting factor was a 25-Hz room vibration coupled to the robot frame.

In a subsequent experiment, a differential sensor consisting of one illumination fiber and two detection fibers was used to generate a direct measurement of the probe-to-wire alignment error. This error signal was fed back to the microprocessor controlling the fine-positioner, which used the information to "lock" the probe onto the line. When this was done, the probe stayed centered on the wire within 1–2 $\mu$m, even in the presence of the room vibration and static frame deflections of 25 $\mu$m or so.

## *Disk slider assembly*

### *Requirements*

The high storage density of IBM 3370 and 3380 disk products requires that the distance between the recording head and the spinning magnetic disk be extremely small and almost constant. To accomplish this, the head is supported by a small "slider" which rides on an air cushion created between the slider and the disk [31, 32]. The slider has two rails and is supported by a leaf-spring suspension, as shown in **Figure 7**. It is approximately 3 mm wide and 4 mm long. Since the alignment of the slider with respect to the suspension can have a significant effect on its flying characteristics, both its position and orientation must be tightly controlled during assembly. In this application experiment, we wish to place the slider on the suspension so that its center is displaced at most ±10 $\mu$m relative to its nominal location in the $XY$ plane, and its orientation about the $Z$ axis is controlled to within ±0.15°.

### *Hardware*

The experimental setup is shown in **Figure 8**. A standard carrier was used to dispense sliders, and a specially constructed carrier was used to hold both suspensions and completed slider-suspension assemblies. For this experiment, we simply bolted the carriers to the table. An actual production application would include provisions for replacing them at appropriate times, and would probably include some sensing scheme to locate them relative to the robot.

A standard vacuum pen was mounted to the fine-positioner and was used to transport both sliders and suspensions. This required us to make some undesirable compromises in the vacuum head design. The small size required for picking up the sliders made it difficult to hold the suspension firmly enough to prevent it from rotating about the pen axis while being moved to the assembly fixture. In a production application, two pens would probably be used. This would permit better design and would reduce the cycle time, since both parts (the slider and the suspension) could be brought to the assembly fixture at the same time.

The assembly fixture was a translucent block, backlit to provide a high-contrast image for the binary vision system.

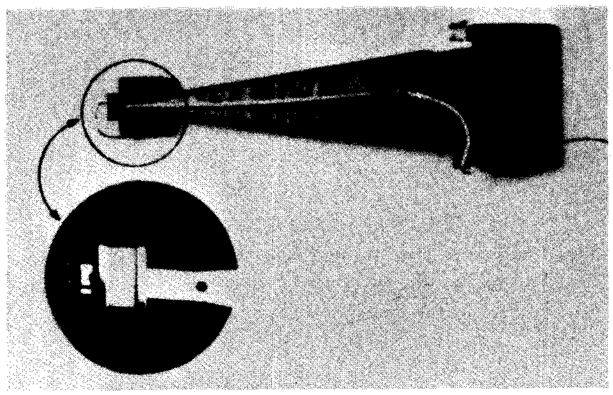


Figure 7

IBM slider-suspension assembly.

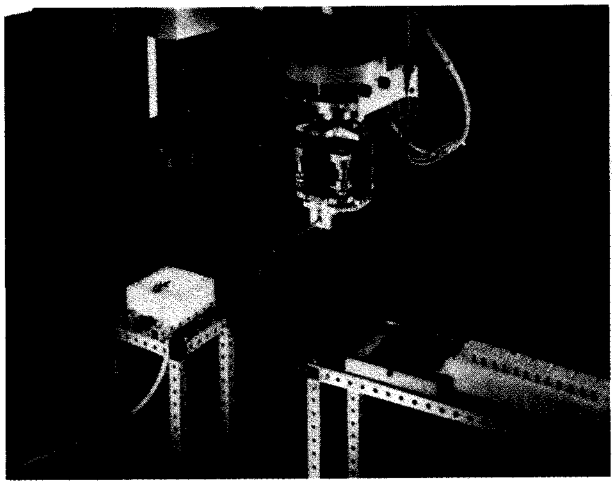


Figure 8

Slider-suspension assembly experiment.

The field of view of the 128 × 128-pixel TV camera was approximately 7 mm × 7 mm, so that each pixel corresponded to a square approximately 55 $\mu$m on a side. The camera was rotated relative to the fixture so that the slider and suspension were tilted in the image. This tilting was necessary to achieve subpixel resolution for the slider-suspension alignment (see below). The block was machined to provide a small "platform" for the suspension, and a suction port was used to hold the suspension in place during assembly. In order to get around the difficulties encountered in transporting the suspension, several alignment pins were added to the fixture, so that the suspension was approximately oriented. These could probably be eliminated in a production application.

*Job cycle*

An AML program for performing slider-suspension assembly is shown in **Figure 9**. The execution steps are as follows:

*Step 1* Get a suspension.
Using the vacuum pen, pick up the next suspension from the suspension carrier and place it in the assembly fixture. Turn on the fixture suction to hold the suspension in place.

*Step 2* Locate the suspension.
Use the vision system to locate the suspension.

*Step 3* Apply a drop of glue to the suspension.
For this experiment, we used a heat-setting epoxy. Since our primary interest was in demonstrating visual alignment of the head to the suspension, we chose to do this step manually with a hypodermic needle. In an actual production application, any one of a number of techniques could be used to automate it.

```
job_cycle: SUBR;
   step_1:   next_suspension;
             get_suspension;
             susp_to_fixture;
   step_2:   susp_xyw = locate_suspension;
   step_3:   apply_glue;
   step_4:   next_slider;
             get_slider;
             safe_z_move(ARM,tv_station);
   step_5:   adjust_slider;
   step_6:   cure_glue;
             pen_vacuum(OFF);
   step_7:   replace_suspension;
             BRANCH(step_1);
    END;
```

Figure 9

AML program for slider-suspension assembly: Significant subroutines are shown in subsequent figures. An actual production program would include a number of other elements, including operator interfaces, links to the production control system, and error recovery procedures.

*Step 4* Get a slider.
Using the vacuum pen, pick up the next slider from the slider carrier. Using the coarse joints of the robot, move the slider to its approximate final position relative to the suspension.

*Step 5* Use visual feedback to adjust the slider relative to the suspension.

*Step 6* Cure the glue.

*Step 7* Return the completed suspension.
Using the vacuum pen, pick up the completed

suspension assembly and return it to the suspension assembly.

*Getting a suspension*

The AML code for picking up a suspension and transporting it to the assembly fixture is shown in **Figure 10**. A common subroutine, *pen_pickup*, is used for picking up both suspensions and sliders. This subroutine moves the arm to the pickup point, turns on the pen vacuum, waits for a specified time, then lifts the pen by a specified amount. The code for placing a suspension into the assembly fixture is also straightforward. The arm is moved through an appropriate approach trajectory to the setdown point, the fixture vacuum is turned on, and the pen vacuum is turned off. The fine-positioner then makes several quick back-and-forth motions to break the pen suction, and the arm moves up to a clear plane.

*Locating the suspension*

The 2D position and orientation of the suspension are determined as follows:

*Step 1* Acquire a run-coded binary image *rc* of the suspension; a typical image is shown in **Figure 11**.

*Step 2* Extract a "centerline" image of the suspension (shown superimposed on the original suspension image in **Figure 12**) by evaluating the AML/V expression:

```
get_suspension: SUBR;
    pen_pickup(suspension_pallet(susp_no),0.05,
                    susp_dwell,0.3,<0.02,0.1,0.1>);
    MOVE(ZJT,safe_z,,<SPEED,0.5,0.5>);
    END;

pen_pickup:SUBR(loc,
                    approach_dz DEFAULT 0.020,
                    dwell_time DEFAULT .25,
                    depart_dz DEFAULT safe__z-loc(3),
                    depart_ctl DEFAULT DEFAULT);
    safe_z__move(ARM,loc+<0,0,approach__dz,0,0,0>);
    MOVE(ZJT,loc(3));
    pen_vacuum(ON);DELAY(dwell__time);
    DMOVE(ZJT,depart_dz,,depart__ctl);
    END;

susp_to_fixture: SUBR;
    safe_z_move(arm,fixture+fixture_app1);
    MOVE(ARM,fixture+fixture_app2);
    MOVE(ARM,fixture);
    fixture_vacuum(ON);
    pen_vacuum(OFF); delay(.5);
    scrub_fp(4,.2);
    MOVE(ZJT,safe_z);
    END;

scrub_fp: SUBR(n,df);
  i: NEW 0;
  f0: NEW QGOAL(fp);                        -- remember where we start
    WHILE n GE i=i+1 DO
        BEGIN
        MOVE(fp,f0+df,,<1.,1.,1.,0>);       -- full speed, accel, no settling
        MOVE(fp,f0-df,,<1.,1.,1.,0>);       -- full speed, accel, no settling
        END;
    MOVE(fp,f0,,<1.,1.,1.,1>);              -- full speed, accel, settling
    END;
```

**Figure 10**

AML subroutines for getting a suspension.

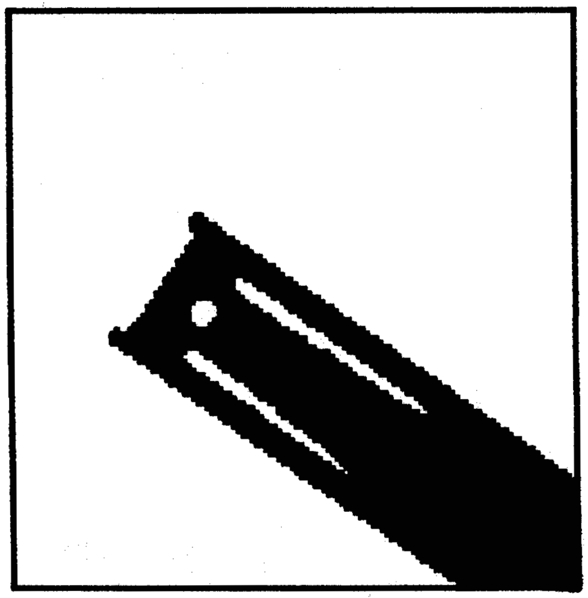


**Figure 11**

Binary image of suspension.

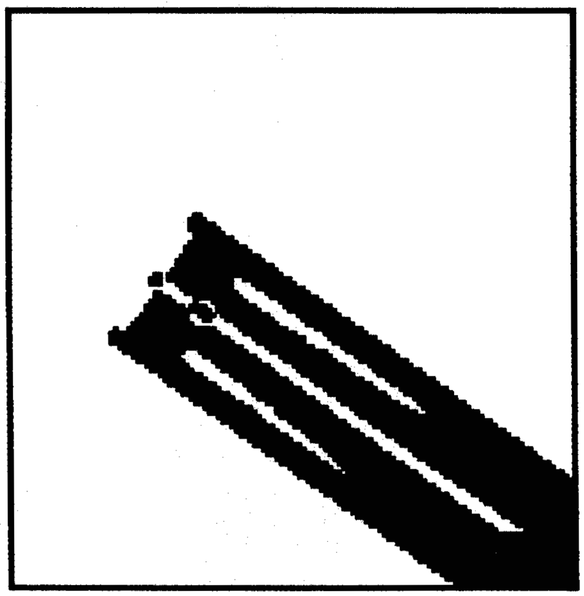


**Figure 12**

Suspension centerline, superimposed on suspension image.

```
rc_suspension_center =
  RCOR(RCSHIFT(rc,suspension_half_
  width,WHITE),
    RCSHIFT(rc,-suspension_half_
    width,WHITE))
```

where *suspension_half_width* is a vector that would span slightly less than half the small dimension of the rectangular suspension. This vector is obtained during calibration. The technique is insensitive to up to ±5° variations in its orientation.

*Step 3* Perform binary region analysis on *rc_suspension_center*; from second-moment features of the white centerline region, determine the suspension orientation.

*Step 4* Perform binary region analysis on *rc*; use the centroid of the white region lying most nearly on the centerline (the locating hole in Fig. 11) as the *XY* coordinates for the suspension.

Subpixel resolution is obtained for both position and orientation parameters, since region features are derived from sums of values over all pixels in the region (cf. [33]).

*Getting the slider*

The AML code for picking up a slider is shown in **Figure 13**. The principal complication arose because the clearance between the rails of the slider and the tip of the vacuum pen was only 0.2 mm. This is comparable to the potential slider-vacuum tip misalignment at the pickup point. In order to ensure that the vacuum tip is not cocked on the rails, the fine-positioner makes a small "scrubbing" motion before the robot lifts the slider out of its nest. This motion takes almost no time and has the added benefit of squaring up the slider somewhat, thus significantly reducing rotational misalignments at the assembly fixture.

```
get_slider: SUBR(loc DEFAULT slider_tray(slider_no));
    pen_pickup(loc,,slider_dwell,.02);
    scrub_fp(3,<0.,0.05>);
    MOVE(ZJT,safe_z);
    END:
```

Figure 13

AML subroutine for getting a slider.

*Locating the slider*

The 2D position and orientation of the slider are determined as follows:

*Step 1* Acquire a run-coded binary image *rc* of the slider; **Figure 14** shows a typical image.

*Step 2* Extract a binary image of the slider's "minor axis" (shown in **Figure 15**, superimposed on the slider

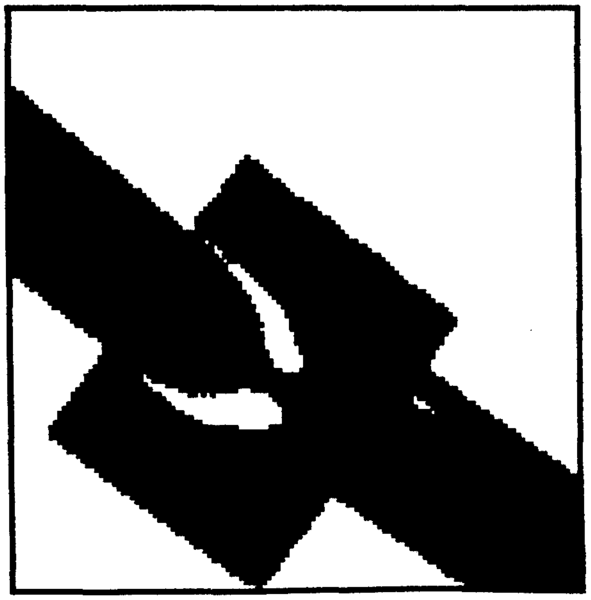


Figure 14

Binary image of slider.

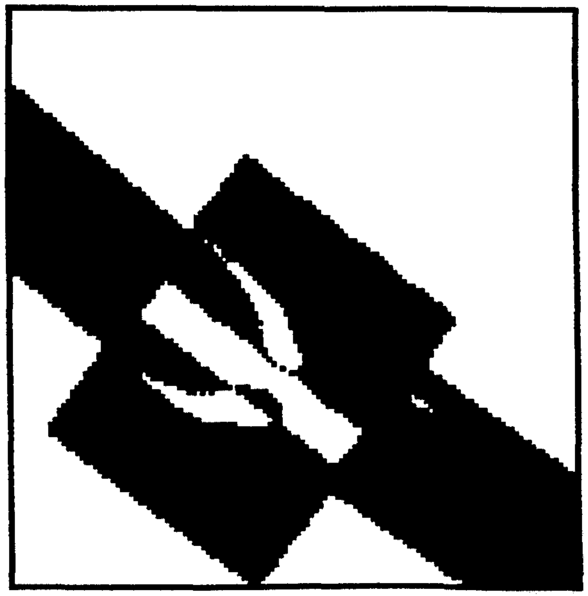


Figure 15

Slider minor axis, superimposed on slider image.

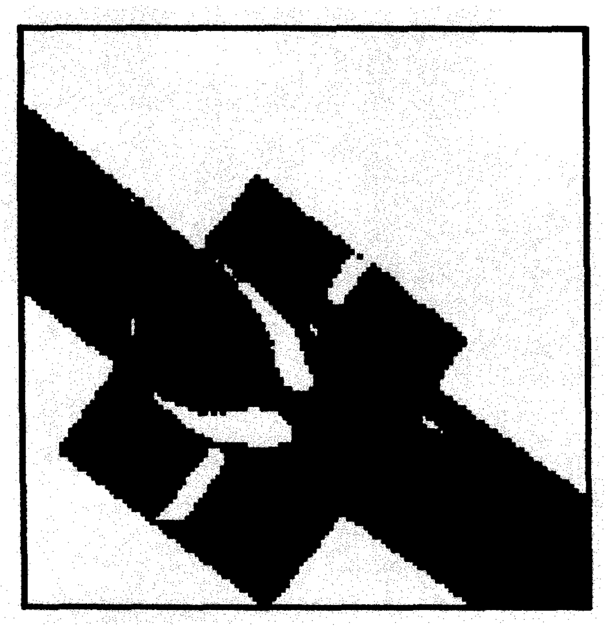


Figure 16

Slider major axis, superimposed on slider image.

```
adjust_slider: SUBR;
   slider_target: NEW susp_xyw + slider_offset(susp_xyw(3));
   e_xyw: NEW slider_xyw;
   tvw: NEW tool_vect(qgoal(6));
   loop:
    slider_xyw = find_slider;
    e_xyw = slider_xyw - slider_target;
    IF ABS(e_xyw(2)) GT r_tol THEN
            BEGIN
            DMOVE(xyw,(-tvw+tvw=tool_vect(qgoal(6)-e_xyw(2)))#<e_xyw(2)>);
            NULL__OUT(ARM(6));
            BRANCH(loop);
            END;
    IF DOT(e_xyw(1),e_xyw(1)) GT xy_tol*xy_tol THEN
            BEGIN
            dmove_in_cam(-e_xyw(1));
            BRANCH(loop);
            END;
    END;
```

Figure 17

AML subroutine for adjusting slider.

image) by evaluating the AML/V expression

*rc_slider_minor_axis =*
  *RCOR(RCSHIFT(rc,slider_half_length,WHITE),*
    *RCSHIFT(rc,-slider_half_length,WHITE))*

where *slider_half_length* is a vector that would span slightly less than half the long dimension of the rectangular slider.

*Step 3* Compute $X$, $Y$, and $\theta$ parameters for the minor axis using first- and second-moment features derived from binary region analysis of *rc_slider_minor_axis.*

*Step 4* Extract a binary image of the slider's "major axis" (shown in **Figure 16** superimposed on the slider image) by evaluating the AML/V expression

*rc_slider_major_axis =*
  *RCAND(RCNOT(RCSHIFT(rc,slider_half_width,BLACK)),*
    *RCNOT(RCSHIFT(rc,-slider_half_width,BLACK)),*
    *RCSHIFT(rc,1.1*slider_half_width,BLACK),*
    *RCSHIFT(rc,-1.1*slider_half_width,BLACK))*

where *slider_half_width* is a vector that would span slightly less than half the short dimension of the rectangular slider.

*Step 5* Compute $X$, $Y$, and $\theta$ parameters for the minor axis using first- and second-moment features derived from binary region analysis of *rc_slider_minor_axis.*

*Step 6* Compute slider $X$, $Y$ from the intersection of the minor and major axes, and $\theta$ from the orientation of the minor axis.

As with the suspension, subpixel resolution is obtained for both position and orientation parameters, since region features are derived from sums of values over all pixels in the region. By experimentally moving the slider by known amounts, we were able to verify that resolutions on the order of 0.2 pixels, corresponding to displacements of about 10 $\mu$m, were obtained.

*Adjusting the slider*

The AML code for aligning the slider with respect to the suspension is shown in **Figure 17**. An iterative approach is used:

*Step 1* Determine the position and orientation of the slider, using the method described above. Compute the displacement and orientational misalignment relative to the suspension.

*Step 2* If the slider is correctly oriented relative to the suspension, go to Step 3. Otherwise move the coarse $X$, $Y$, and $\theta$ joints to correct for the misalignment while keeping the slider in the camera's field of view, and go back to Step 1.

*Step 3* If the slider's displacement relative to the suspension is small enough, stop. Otherwise, move the fine-positioner $X$ and $Y$ actuators by an appropriate amount to correct for the misalignment and go back to Step 1.

The principal complications with this approach arise from using the coarse joints to correct for angular misorientations. The distance from the axis of the $\theta$ motor to the tip of the vacuum pen is about 14 cm. Thus each 1° of rotation correction introduces an additional 0.24-mm lateral displacement of the slider. In principle, up to 4° orientation could be compensated by using the fine-positioner axes. However, we found that the $\theta$ joint was slow enough that it made sense to move the coarse $X$ and $Y$ motors as well and to reserve the full range of the fine-positioner for Step 3.

The final motions are made using the fine-positioner actuators. For various reasons, we found it most convenient to compute the desired displacement in camera coordinates. The necessary camera-to-fine-positioner transformation is determined by using the fine-positioner to displace a slider by known amounts and then calling the *find_slider* subroutine. The transformation is then computed by ordinary least squares.

## Conclusion

This paper has described an endpoint-sensing method for achieving very precise alignment of a part or tool with respect to a workpiece. Our approach relies on the coarse joints of a robot to bring a tool or part within the "capture range" of a fine-positioning system carried by the robot, which is then used to null out sensed misalignments. To investigate this approach, we have developed a compact "wrist" mechanism capable of making fast and extremely precise motions in two directions. We have used it in several application experiments which demonstrate the feasibility of achieving roughly an order-of-magnitude improvement in the effective precision of a robot (from about 0.1 mm to 0.01 mm) while retaining a large working volume and high speed for long motions. The principal requirement is that it be possible to sense the misalignment between the tool or part and the workpiece.

Many applications require the ability to correct small rotational misalignments as well as displacement errors. In this paper, we used a "coarse" wrist motor to supply these corrections. Work is proceeding on an $XY\theta_z$ model of the fine-positioner that will allow fast rotational motions through small fractions of 1°. Work is also proceeding on additional endpoint-sensing experiments, and on investigation of other operating modes such as force compliance.

## Acknowledgments

Many people contributed to the success of this work. We are grateful to all of them. We would like to express our particular thanks to Bela Musits, who assisted in the detailed design of our first batch of fine-positioners, to Frank Morgan, who constructed them, to Richard Perrone, who was responsible for constructing their control electronics, to Patrick Muir, who built electronic interfaces for the first fine-positioner prototype and who implemented an initial prototype of the TCM-board testing application, to Mark Johnson, who developed the fiber-optic sensors for locating the wires, and to Friedrich Fischer, who built the fixtures for the disk-slider mounting experiment.

## References and note

1. T. Lozano-Perez, "Robot Programming," *Proc. IEEE* **71**, 821–841 (July 1983).
2. R. Podoloff, W. Seering, and B. Hunter, "An Accuracy Test Procedure for Robotic Manipulators Utilizing a Vision Based, 3-D Position Sensing System," *Proceedings, American Control* Conference, San Diego, June 6–8, 1984.
3. R. H. Taylor and D. D. Grossman, "An Integrated Robot System Architecture," *Proc. IEEE* **71**, 842–855 (July 1983).
4. S. Drake, "Using Compliance in Lieu of Sensory Feedback for Automatic Assembly," D.Sc. Thesis, Massachusetts Institute of Technology, Cambridge, 1977.
5. C. C. Geschke, "A System for Programming and Controlling Sensor-Based Robot Manipulators," *IEEE Trans. Pattern Anal. & Machine Intelligence* **PAMI-5**, 1–7 (January 1983).
6. M. T. Mason, "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Trans. Syst. Man Cybern.* **SMC-11**, 418–432 (1981).
7. M. Raibert and J. Craig, "Hybrid Position/Force Control of Manipulators," *J. Dyn. Syst. Measure. Contr.* **102**, 126–133 (1981).
8. N. Hogan, "Mechanical Impedance Control in Assistive Devices and Manipulators," *Proceedings, Joint Automatic Control Conference*, San Francisco, 1980.
9. T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Manipulators," *Int. J. Robot. Res.* **3**, 3–24 (1984).
10. H. Asada and T. Kanade, "Design of Direct-Drive Mechanical Arms," *Robotics Institute Report CMU-RI81-1*, Carnegie-Mellon University, Pittsburgh, PA, 1981.
11. H. Asada and K. Youcef-Toumi, "Development of a Direct-Drive Arm Using High-Torque Brushless Motors," Laboratory for Manufacturing and Productivity, Massachusetts Institute of Technology, Cambridge, 1983.
12. André Sharon and David Hardt, "Enhancement of Robot Accuracy Using Endpoint Feedback and a Macro-Micro Manipulator System," *Proceedings, American Control Conference*, San Diego, June 6–8, 1984, pp. 1836–1842.
13. K. Asakawa, F. Akiya, and F. Tabata, "A Variable Compliance Device and Its Application for Automatic Assembly," *Proceedings, Autofact 5 Conference*, Detroit, November 14–17, 1983.
14. R. Bolles and R. Paul, "The Use of Sensory Feedback in a Programmable Assembly System," *Computer Science Report STAN-CS-396*, Stanford University, CA, October 1973.
15. Y. Shirai and H. Ionue, "Guiding a Robot by Visual Feedback in Assembling Tasks," *Pattern Recog.* **5**, 99–108 (1973).
16. G. Beni, S. Hackwood, and W. S. Trimmer, "High-Precision Robot System for Inspection and Testing of Electronic Devices," *Proceedings, IEEE International Conference on Robotics*, Atlanta, March 1984.
17. P. Dario, D. DeRossi, C. Domenici, and R. Francesconi, "Ferroelectric Polymer Tactile Sensors with Anthropomorphic Features," *Proceedings, IEEE International Conference on Robotics*, Atlanta, March 1984.
18. G. L. Miller, R. A. Boie, and M. J. Sibilia, "Active Damping of Ultrasonic Transducers for Robotic Applications," *Proceedings,*

*IEEE International Conference on Robotics*, Atlanta, March 1984.
19. T. Kanade and T. Somer, "An Optical Proximity Sensor for Measuring Surface Position and Orientation for Robot Manipulation," *Proceedings, First International Symposium on Robotics Research*, Breton Woods, August 1983; MIT Press, Cambridge, MA, pp. 547–564.
20. P. Will and D. Grossman, "An Experimental System for Computer Controlled Mechanical Assembly," *IEEE Trans. Computers* **C-24**, 879 (1975).
21. R. Taylor, P. Summers, and J. Meyer, "AML: A Manufacturing Language," *Int. J. Robot. Res.* **1**, 19–41 (1982).
22. *IBM Manufacturing System: A Manufacturing Language Reference Manual,* Order No. 8509015, 1983; available through IBM branch offices.
23. *APL Language*, Order No. GC26-3847-4, 1978; available through IBM branch offices.
24. M. Lavin and L. Lieberman, "AML/V: An Industrial Machine Vision Programming System," *Int. J. Robot. Res.* **1**, 42–56 (1982).
25. R. L. Hollis, "A Fine Positioning Device for Enhancing Robot Precision," *Proceedings, Robots 9 Conference*, Detroit, June 2–6, 1985.
26. M. R. Cutkosky and P. K. Wright, "Position Sensing Wrists for Industrial Manipulators," *12th International Symposium on Industrial Robots*, Paris, 1982, pp. 427–438.
27. B. A. Sawyer, "Magnetic Positioning Device," U.S. Patent No. 3,457,482, issued July 22, 1969.
28. For specifications of the lateral-effect position sensor, see data sheets from Silicon Detector Corporation, Newbury Park, CA.
29. Roy F. Bonner, John A. Asselta, and Frank W. Haining, "Advanced Printed-Circuit Board Design for High-Performance Computer Applications," *IBM J. Res. Develop.* **26**, No. 3, 297–305 (May 1982).
30. J. R. Bupp, L. N. Chellis, R. E. Ruane, and J. P. Wiley, "High-Density Board Fabrication Tolerances," *IBM J. Res. Develop.*, **26**, No. 3, 306–317 (May 1982).
31. Sanford A. Bolasna, Kenneth L. Deckert, Michael F. Garnier, and Robert B. Watrous, "Air-Bearing Support of a Magnetic Recording Head," *IBM Disk Storage Technology*, pp. 12–15, Order No. GA26-1665-0, 1980; available through IBM branch offices.
32. R. B. Watrous, "Transducer Suspension Mount Apparatus," U.S. Patent 3,931,641, 1976.
33. J. Hill, "Dimensional Measurements from Quantized Images," *Machine Intelligence Research Applied to Industrial Automation (Tenth Report)*, SRI International, New York, November 1980, pp. 75–106.

*Received December 17, 1984; revised February 27, 1985*

**Ralph L. Hollis** *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Hollis received a B.S. and an M.S. in physics, both from Kansas State University, Manhattan, in 1964 and 1965, respectively. From 1965 to 1970 he was employed by the Autonetics Division of North American Aviation, where he was engaged in computer simulation of space flight vehicles. Beginning in 1970, he attended the University of Colorado, receiving the Ph.D. degree in solid state physics in 1975. After a brief postdoctoral appointment at the University of Colorado, he was a National Science Foundation/Centre Nationale de Recherche Scientifique Exchange Scientist at the Université de Pierre et Marie Curie, Paris, for part of 1976–77. Since joining IBM in 1978, he has worked at the Thomas J. Watson Research Center in the fields of magnetism, acoustics, and robotics. Dr. Hollis has received two IBM Invention Achievement Awards.

**Mark A. Lavin** *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Lavin is manager of the industrial machine vision project in the Manufacturing Research Department at the IBM Thomas J. Watson Research Center. He joined the automation research group in 1979, having worked previously with that group as a summer student in 1975. His interests are in applications of computers to problems involving geometry, including machine vision, robotics, and computer graphics. He received his B.S. and M.S. degrees in electrical engineering in 1973 and his Ph.D. in artificial intelligence in 1977, all from the Massachusetts Institute of Technology, Cambridge. From 1977 to 1979, Dr. Lavin worked at Bolt, Beranek and Newman, Inc. in Cambridge, Massachusetts, where he participated in the development of a molecular modeling system. Dr. Lavin is a member of Eta Kappa Nu, Sigma Xi, and Tau Beta Pi.

**Russell H. Taylor** *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Taylor received a B.E.S. degree from Johns Hopkins University, Baltimore, Maryland, in 1970 and in 1976 a Ph.D. in computer science from Stanford University, California. In 1976 he joined the automation research project at the Thomas J. Watson Research Center, where he worked on robot systems and languages. He spent 1978 and 1979 doing systems architecture work with IBM advanced manufacturing systems in Boca Raton, and in 1980 rejoined IBM Research, where he continued to pursue research in robotics. Since returning from a six-month sabbatical leave at the Massachusetts Institute of Technology Artificial Intelligence Laboratory in the summer of 1982, he has been the manager of the robot systems technology project at IBM Research. His research interests include robot systems, programming languages, and model-based programming. Dr. Taylor is a member of the Institute of Electrical and Electronics Engineers, Phi Beta Kappa, Sigma Xi, and Tau Beta Phi. He has received several IBM awards, including a Research Division Outstanding Contribution Award for his work on the AML language.