MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A. I. Memo 1016 **January, 1988**

HERBERT: A SECOND GENERATION MOBILE ROBOT

Rodney A. Brooks, Jonathan H. Connell and Peter Ning

Abstract. In mobile robot research we believe the structure of the platform, its capabilities, the choice of sensors, their capabilities, and the choice of processors, both onboard and offboard, greatly constrains the direction of research activity centered on the platform. We examine the design and tradeoffs in a low cost mobile platform we have built while paying careful attention to issues of sensing, manipulation, onboard processing and debuggability of the total system. The robot, named Herbert, is a completely autonomous mobile robot with an onboard parallel processor and special hardware support for the subsumption architecture [Brooks (1986)], an onboard manipulator and a laser range scanner. All processors are simple low speed 8-bit micro-processors. The robot is capable of real time three dimensional vision, while simultaneously carrying out manipulator and navigation tasks.

Acknowledgments. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the research is provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, in part by a grant from the Systems Development Foundation, and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

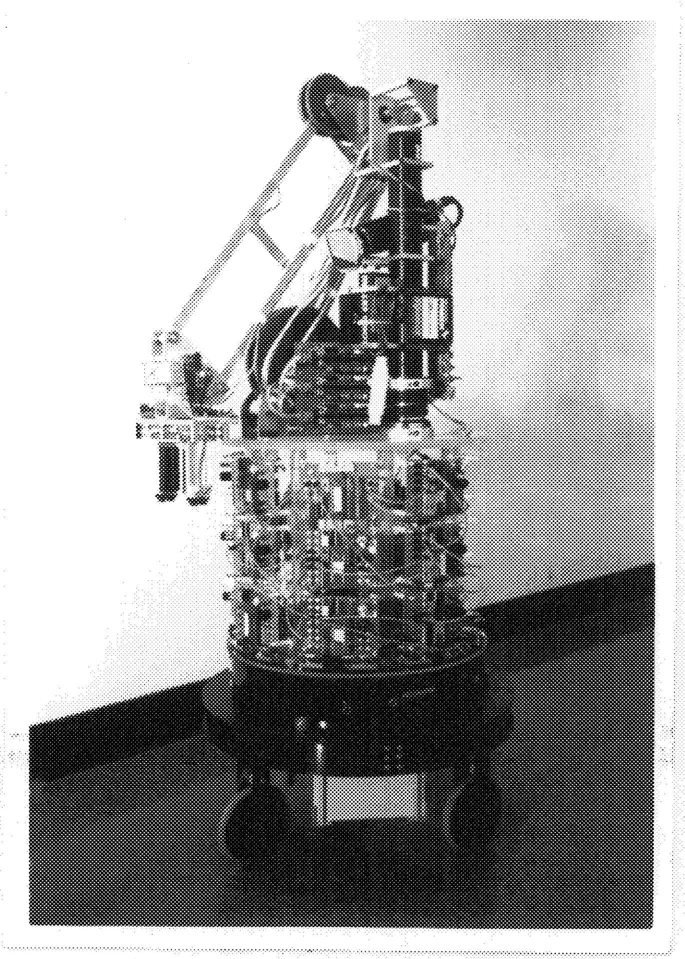

Figure 1. The robot Herbert, showing three wheeled base, 24 processors, laser light striper and arm.

1. Introduction

We have built a completely autonomous mobile robot, named Herbert, for indoor use, based on the design presented in [Brooks, Connell and Flynn (1986)]. The robot is pictured in figure 1. All its subsystems are now operational and it has successfully carried out navigation, recognition and manipulation tasks.

The major innovations in its design have been an onboard loosely coupled parallel processor, a lightweight manipulator and a simple but robust laser depth scanner. This paper gives an overview of some design optimizations useful for building a small indoor mobile robot for experimental use. The theme of the design has always been simplicity and reliability at the local level, with high global performance being produced by careful integration of such components.

2. Hardware Implementation of the Subsumption Architecture

The subsumption architecture [Brooks (1986)] is based on loosely coupled networks of finite state machines. Each individual machine can have some timers (clocks that signal an event after some prespecified amount of time) and can access a limited computation engine to do

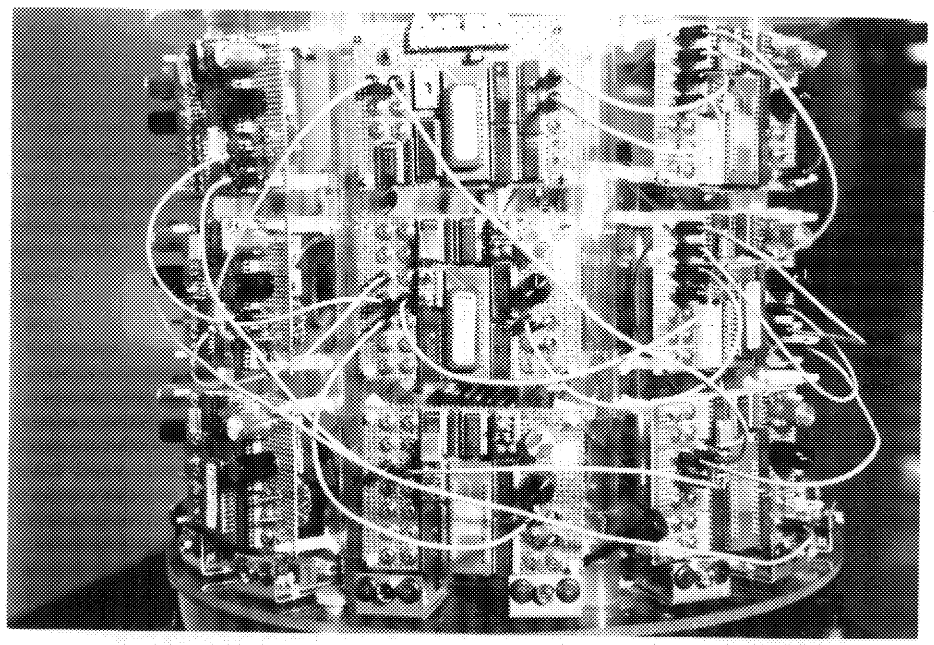

Figure 2. Processors are Hitachi CMOS 6800s and are linked via two conductor cables, transmitting 24 bit messages, and a distributed patch panel.

simple arithmetic and geometric computations. These finite state machines are connected by low bandwidth wires, over which messages can be sent.

In our first implementation of the subsumption architecture we used a conventional uniprocessor to simulate a parallel machine where each simulated processor was precisely one finite state machine. This simulation was sufficient to successfully demonstrate the fundamental utility of the subsumption architecture, but it suffered from a number of drawbacks.

- The implementation did not demonstrate the lack of central control and data that is inherent in the subsumption architecture. With the existence of the central processor and shared memory it required faith on the part of the observer to believe that there really was no need for sharing or synchronization. (In fact it turns out that in the experiments reported in [Brooks (1986)] there was a subtle reliance on the simulation and its mechanism for time sharing. In later implementions the particular networks we used there had to be modified slightly.)
- The computer used for simulation was too large to mount onboard the robot so we needed to run it at all times with either a cable or a radio link. The former was not very satisfactory operationally and the latter was not very reliable. Both introduced considerable latency into the system.
- The implementation was not indefinitely extensible; we soon ran into performance constraints when we added the simulation of more and more finite state machines on a single processor. As we increased the capabilities of the robot it started to miss its real-time performance goals.

Our new implementation successfully overcomes all these drawbacks.

Figure 2 shows a close up of part of the implementation. We implement the subsumption architecture on a collection (of unlimited size) of small 8-bit microprocessors (Hitachi 6301s, which are a CMOS implementation of the 6800 architecture). The only shared resource for

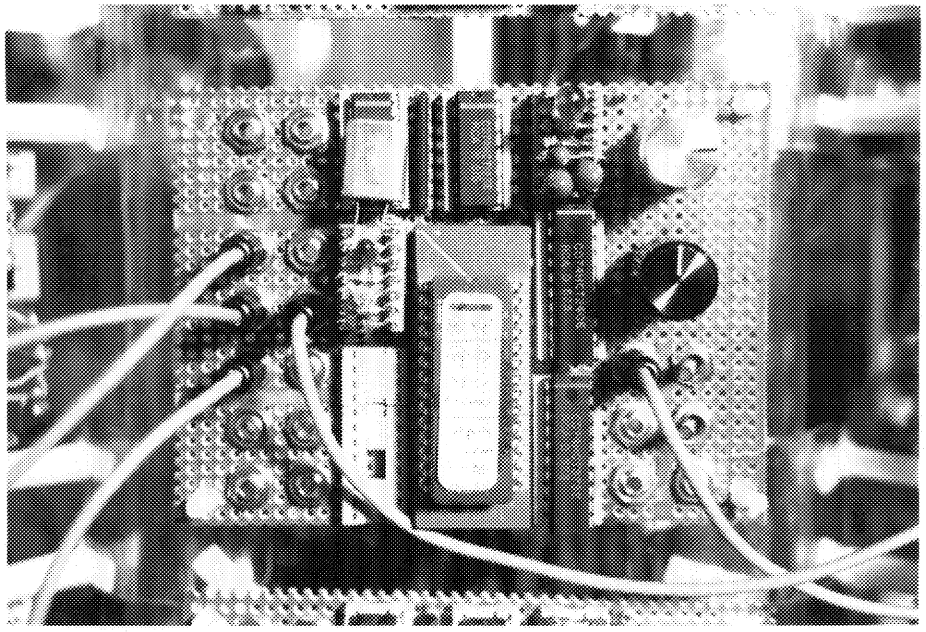

Figure 3. Each standard processor board has a 6800 with its 128 bytes of onboard RAM, an unused slot for 2K bytes expansion, and a suppression node.

the processors is power. There is no global clock, no shared memory, no shared backplane, and no global communication network.

Each standard board (as pictured in figure 3) includes a processor and a suppression node. The processor has 128 bytes of onboard RAM, and accepts an 8K byte piggy-back EPROM. To date, the 128 bytes of RAM has been sufficient for our applications, but as a safety measure there is a socket for an extra 2K bytes of RAM. The processor and support chips are all CMOS—low power consumption is critical for an autonomous mobile robot.

The processor chip has a large number of reconfigurable parallel port bits. We use a number of these together with software running on the processor to implement serial interfaces to the processor card. Each processor has three input serial lines, and three output serial lines. Each serial line has two signals; a control line and a data line. A falling control signal specifies that a 24 bit data message is about to be delivered on the data line. As there is no global clock, the data messages are self clocking. 24 pulses are sent, and length of each of them specifies whether each of the 24 bits is a 0 or a 1. The clocks of the sending and receiving cards need only be within 20% of a common frequency for this scheme to work. The software on the processors polls the parallel port bits every half millisecond to handle the serialization. Worst case transmission speed is roughly 280 baud (about 12 packets per second).

There are two additional special input lines on each card. One is a reset line: any message arriving on this line will reset the processor to the power-up state. The other is an inhibit line: a message arriving on this line inhibits output messages on the first of the card's three output lines for a pre-specified time period. The time period is controlled by a potentiometer on the card and can range from fractions of a second to hundreds of seconds. For both these special inputs the data line is actually ignored; only the control line is important. The remaining three input lines on each card, as described above, can all deliver complete messages to the processor.

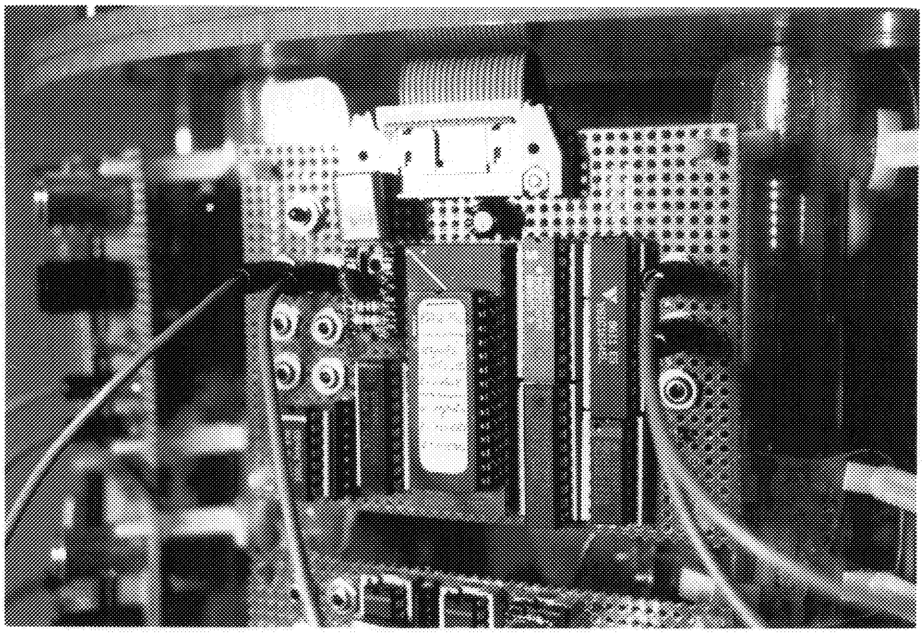

Figure 4. Some processor boards replace the suppression node with an 8-bit parallel port for communicationg with I/O devices.

Originally we planned to have each individual processor simulate precisely one finite state machine. We later realized that was rather wasteful and now have each of them simulate some bounded larger number. When we want to add new finite state machines, and hence augment the robot's capabilities, we take another processor board from stock, mount it on the robot frame, connect it to power and wire it into the network at the appropriate place. In this way we never stretch the real-time capabilities of existing finite state machines or processors.

Each processor board has some additional space. On standard boards this space is occupied by a *suppressor* node [Brooks (1986)]. This device allows one module to override the output of another module for a pre-specified amount of time. The time constant can be adjusted by a potentiometer on the board.

On a small number of cards the suppressor node is not present. Rather, as shown in figure 4 there is an 8-bit parallel port. These ports are needed to communicate with input/output devices, such as the locomotion servo computer, the manipulator analog servo card, the infrared proximity sensor driver card, and the laser scanner processing cards. Additionally, we have one processor card which uses this board space to house a UART and an RS-232 serial line so that a diagnostic terminal can be plugged into the robot.

Lastly there is a special class of processor cards, called *Line Oriented Vision Processors* or *LOVPs*, pictured in figure 5 that are used to support the processing of depth images from the laser scanner. We describe these cards in more detail in section 4.

3. Effectors

Herbert has two effectors: a drive mechanism and a manipulator.

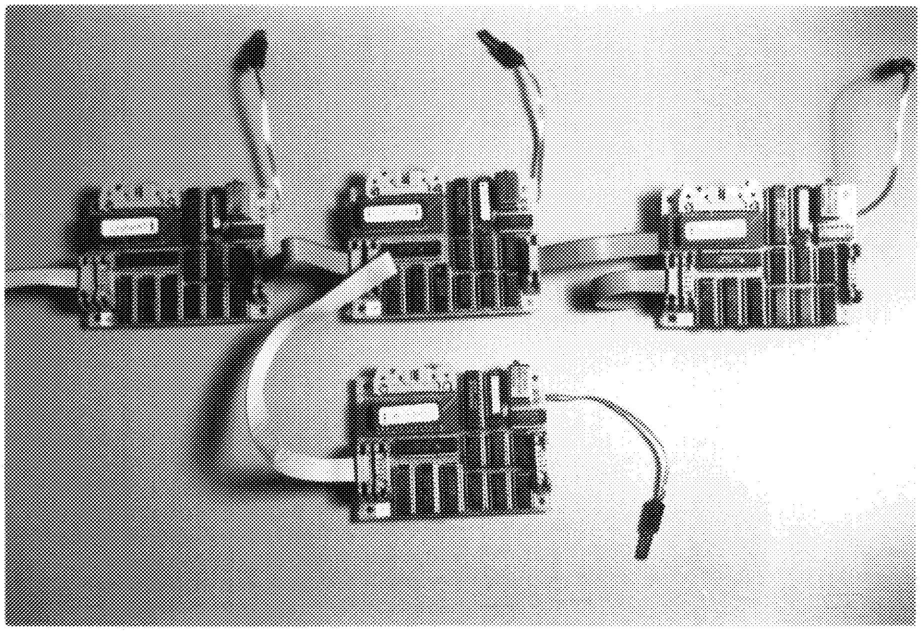

Figure 5. A third type of processor board, the Line Oriented Vision Processor, has the same microprocesser, but includes a high speed 2K byte RAM. The boards can be linked in a tree to build a serpentine image memory with pipelined processing. Four LOVPs are shown in this picture.

Drive mechanism

The drive mechanism of the robot was purchased from Real World Interface and is identical to the drive mechanism of Allen, our earlier robot [Brooks (1986)]. The base comes with a servo computer and its own set of lead-acid gel cells for power. Physically, the base is 18 inches in diameter and stands about 12 inches tall. There are three wheels which always point in the same direction, as does the top plate of the robot base. The orientation of these is controlled by a chain drive mechanism. The three wheels are all powered by a single drive motor, again through a chain drive mechanism. The robot is thus able to turn in place and, as it does, the torso we have built on top of the base top plate also turns. The laser scanner and manipulator always face in the forward direction of motion of the robot.

The upper part of the robot, which we built, is powered by 16 silver-zinc cells. These have an extremely high power density and power the parallel processor, the laser scanner, the infra-red proximity sensors and the manipulator. The total power consumption of the robot is about 100 watts. These batteries let the robot operate for approximately one hour.

Manipulator

To allow Herbert to do more than wander around passively we decided to include an arm onboard. There were two choices: place a commercial arm onboard or build one ourselves. We decided that a commercial arm was not a viable option because all such arms were either too heavy or not did not have a workspace that extended much over the side of the robot. We wanted a lightweight arm with a large workspace. We decided that the arm should be capable of pick and place operations both at ground level and at table-top level.

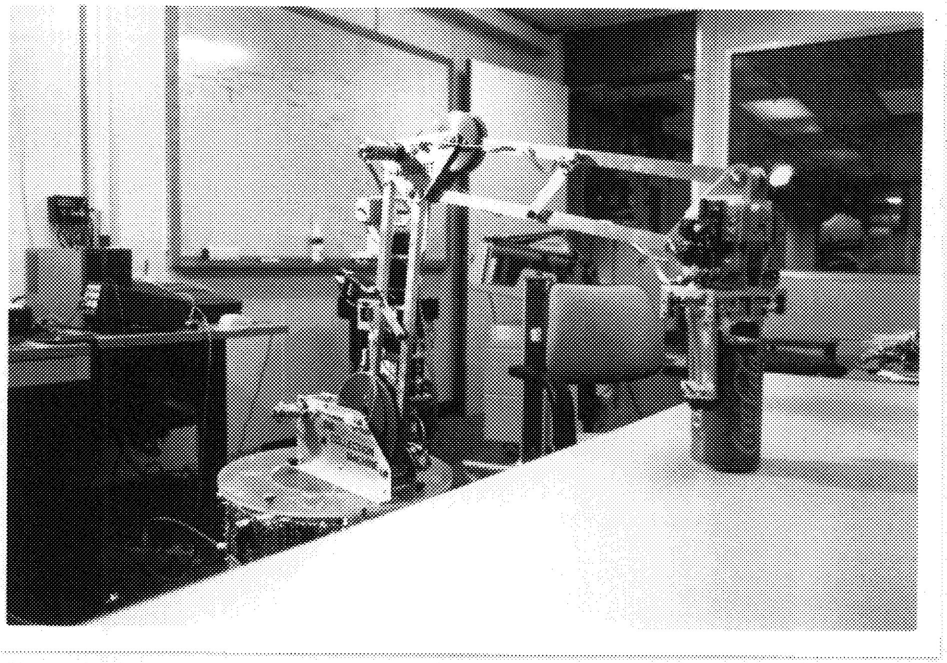

Figure 6. The onboard manipulator has a long reach, and can pick and place objects at both ground level and table top level. The hand is laden with simple sensors which allow it home in on and grasp objets in uncertain locations.

The arm, pictured in figure 6, has only two degrees of freedom plus a parallel jaw gripper which is always oriented with the jaws vertical. The gripper can move in a vertical plane which runs through the center of rotation of the robot base. Thus by rotating the base we can provide a side to side motion for the gripper. The gripper can be moved to all points in a workspace which is 40 inches high by 18 inches deep.

Each joint, plus the gripper, is driven by a lightweight DC gearhead motor. These motors generate 80 oz-in of torque which is further increased by a chain drive. Although this allows the arm to carry a payload of up to 2 pounds, it means the arm moves fairly slowly: full down to full up takes about 6 seconds at top speed.

Bolted to the side of the arm are three identical analog position servos; one for each of the two arm joints and one for the gripper. These are interfaced to a subsumption architecture processor through an 8-bit parallel port. The processor can read the joint angles and the gripper separation throught this port. It also has access to the servo loop error voltages. To control the arm, the processor sends a velocity command to the arm servo which integrates this command over time to generate a desired position. This setup allows us to control the position of the arm to a resolution better than the joint angle encoders and also gives us good control of the instantaneous velocity of the gripper.

4. Sensors

Allen, our earlier robot [Brooks (1986)], relied on sonar as its primary sensor. For Herbert we primarily use two types of sensors: infrared proximity sensors for local obstacle detection and avoidance, and a laser triangulation scanner for longer range object recognition. There is also a cluster of specialized sensors on the hand itself.

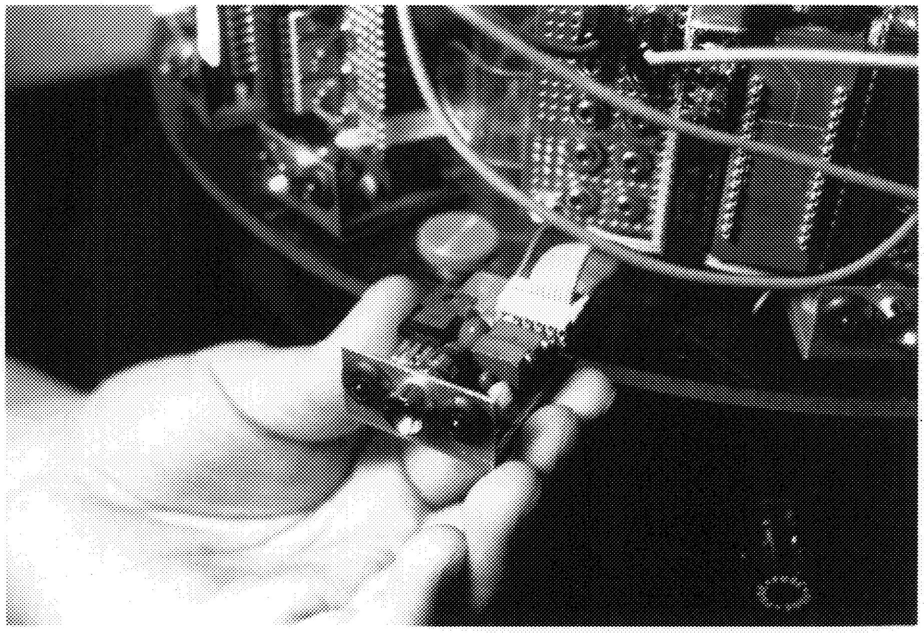

Figure 7. The infra-red proximity sensors provide a very coarse depth estimate. In practice we simply use them to determine the presence of nearby objects.

Proximity Sensors

The infrared proximity sensors pictured in figure 7 return intensity based depth estimates. We extract only two bits of information from each sensor. We have established empirically that this is sufficient accuracy for simple obstacle avoidance (both moving and stationary obstacles).

The major advantage of infrared proximity sensors over sonar is the fast response time. Sonars are limited by the speed of sound, but of course our proximity sensors are only limited by the speed of light. This lets us complete a full 360 degree scan of the environment quickly without worrying about adjacent sensors interfering with each other.

The major disadvantage of our infrared proximity sensors are that they are intensity based. This means, that they are albedo sensistive - dark objects do not appear as close as lighter colored objects. Furthermore, the visual angle subtended by an obstacle also affects the return intensity. The sensor gets the same reading for a small nearby object as for a larger, further away object.

Laser Range Scanner

The laser range scanner is the primary sensor used as the basis for intelligent action. It is pictured in figure 8.

A 7 mW Helium-Neon laser is mounted vertically. A cylindrical lens spreads the beam into a flat plane which is reflected off a moveable mirror to generate a horizontal stripe. The mirror scans this stripe downwards in a range from 10° above the horizontal to 48° below. A CCD camera (510×492) pixels with an optical broad-band interference filter is mounted on the side of the laser pointing about 20° downwards. The camera is turned on its side so that the normal "horizontal" scan lines run vertically.

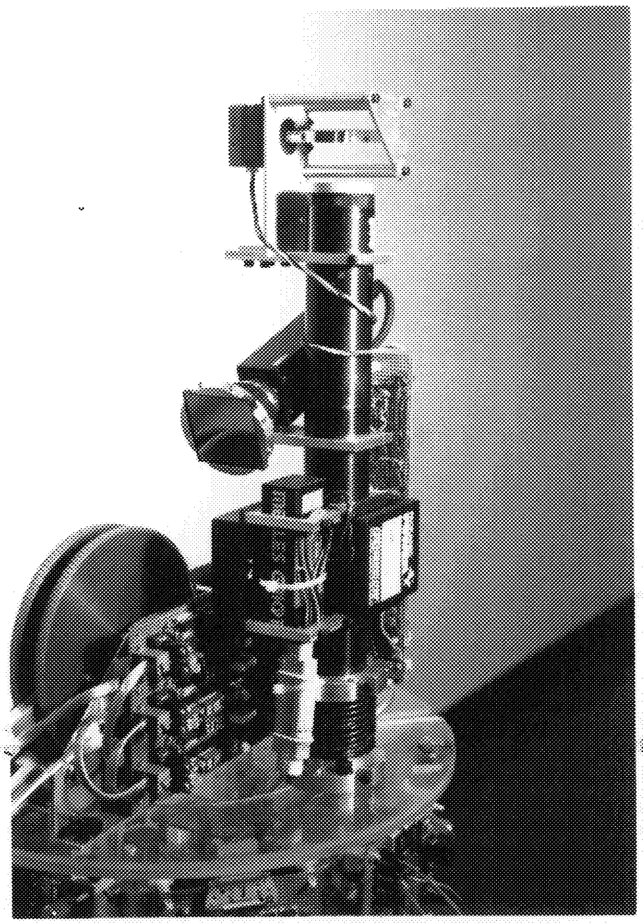

Figure 8. The laser light striper mechanically scans a plane of laser light over the scene every second. A disparity map 256 pixels wide, by 32 pixels deep, by 8 bits per pixel is delivered to the first LOVP.

The camera runs at 60 Hz providing 30 frames of interlaced odd/even fields every second. We ignore the data in odd fields. However, we take this field's vertical synchronization pulse from the camera and use it to drive a stepper motor which moves the scanning mirror 0.9° per step. The laser beam is thus deflected 1.8° per step. The stepper motor was chosen to be as fast as possible, and the mass of the mirror was kept as low as possible so that the motion settles in about 2 milliseconds, minimizing vibration during the CCD's integration time.

During the second sixtieth of a second of a frame we use the even field of the camera to provide depth data. At the start of each camera scan line we reset a counter which is then incremented during the scanning of the line using the horizontal pixel clock from the camera. We stop the counter when an analog electronics filter and threshold device detects the laser beam in the camera's horizontal scan line (which, remember, is physically in the vertical direction). The further away an object is, the closer the detected laser line is to the start of the scan line. These counter values are buffered for each line to create a complete image. Every thirtieth of a second therefore we get a 256 wide set of 8 bit disparity readings. These are fed through to the first in a tree of Line Oriented Vision Processors (LOVPs). A complete depth image consists of 32 such disparity arrays, one for each position of the sheet of laser light.

Four LOVPs are shown in figure 5. A LOVP has the same size and same processor as all our other processor cards. A LOVP has a 2K byte buffer and two high speed tranfer ports. The idea is that each LOVP should maintain two scan lines of data from the laser scanner, and every thirtieth of a second shunt the older of the two onto the next processor in a chain, while receiving a new scan line from its own predecessor. We allocate 4 bytes to each pixel; one for raw data and three for temporary results. Thus 4×256 bytes, or 1K bytes, are transferred in from the previous processor and 1k bytes are transferred out to the succeeding processor. This whole process takes only 1 millisecond and is done once every 33 milliseconds. In the remaining 32 milliseconds, the 8-bit processor can access the data to carry out various image processing functions. The LOVPs can fan out in a tree, enabling us to carry out many high level vision functions in parallel.

Besides the pipeline memory and associated input/output ports, each LOVP has an 8-bit parallel port compatible with the other 8-bit parallel ports used by Herbert. This is how the results of the vision processing, such as the location of objects, are reported to the normal processors.

Hand Sensors

The hand itself is equipped with a number of sensors. There are mechanical contact switches on the tips of the two fingers and a conventional "break beam" type sensor between the fingers.

At the front of the hand are two infra-red proximity sensors similar to those installed on the base of the robot. The beams from the two sensors are crossed at 45 degrees and angled about 10 degrees downward. The crossed sensors are operated in both an intensity based mode, like the body IRs, and a geometric ranging mode. The idea is to use the geometry of the two sensors to tell when an object is in the intersection of their beams. We do this by checking whether the left sensor can see light emitted by the right sensor and vice versa.

5. Debuggability

Our previous experience with research robots had shown us that most of the time a given mobile robot is stripped down for modifications, and even when operational there is a very short mean time between necessary repairs and minor adjustments.

Ease of access and subsystem removal is then of primary importance. We therefore made such capabilities a primary goal in building Herbert. All circuit boards snap onto plastic supports and are accessible with no disassembly of the robot. All signals to them are via simple connectors. All chips and other components are mounted in chip sockets and are individually removable. All mechanical subsystems, such as the manipulator, laser scanner and infrared proximity sensors, can be removed in less than 30 seconds without using tools.

Through careful choice of connectors and mechanical interfaces we have built a robot where trivial adjustments really are trivial to make.

6. How the Hardware Combines to Support the Mission

One of the goals we have for Herbert is for it to wander around collecting interesting objects with his manipulator and then bringing them back to a central location. We plan on doing this in a number of steps. Each of these steps has already been demonstrated either on Herbert, on our earlier robot Allen, or on susbsytems of Herbert before they were integrated.

- First, Herbert wanders around using his body IRs and light striper to follow walls, traverse corridors, and go through doors. While he is doing this a rough record is kept of the distance he drives and angles he turns using the encoders on the base.
- Meanwhile, the light striper is looking for collections of objects at some height off the floor (it can't reliably find small objects from far away). When it finds a likely area, the robot drives toward it.
- On the way to its goal, the body IRs and, to some extent, the light striper, keep the robot from hitting any intervening obstacles.
- When it gets close enough, the light striper can detect objects suitable for grasping. It commands the arm to move in the right general direction.
- Once the hand gets in the vicinity of the object to be grasped, the specialized local sensors take over to control the fine positioning of the hand and the actual acquisition of the object.
- The arm is then retracted and the robot uses the path memory it created while wandering to get it back to its original location. At home it deposits the object and goes out in search of others.

The interesting aspect of this set of behaviors is that no goals or intents are communicated internally between them. Rather the observable state of the world and the robot is used to trigger what to do next.

References

[**Brooks (1986)**] "A Robust Layered Control System for a Mobile Robot", Rodney A. Brooks, *IEEE Journal of Robotics and Automation*, RA-2, April, 14–23.

[**Brooks, Connell and Flynn (1986)**] "A Mobile Robot with Onboard Parallel Processor and Large Workspace Arm", Rodney A. Brooks, Jonathan H. Connell and Anita M. Flynn, *Proceedings AAAI Conference, Philadelphia, PA*, 1096–1100.