29

Design of Low-Cost Equipment for Cognitive Robot Research

H. G. Barrow and S. H. Salter
Department of Machine Intelligence and Perception
University of Edinburgh

MARK I DEVICE

A minimal robot, known as Freddy, has been constructed with the aim of connecting a usable device on-line to the Department's ICL 4130, under the Multi-POP time-sharing system, and discovering the snags. (See figure 1).

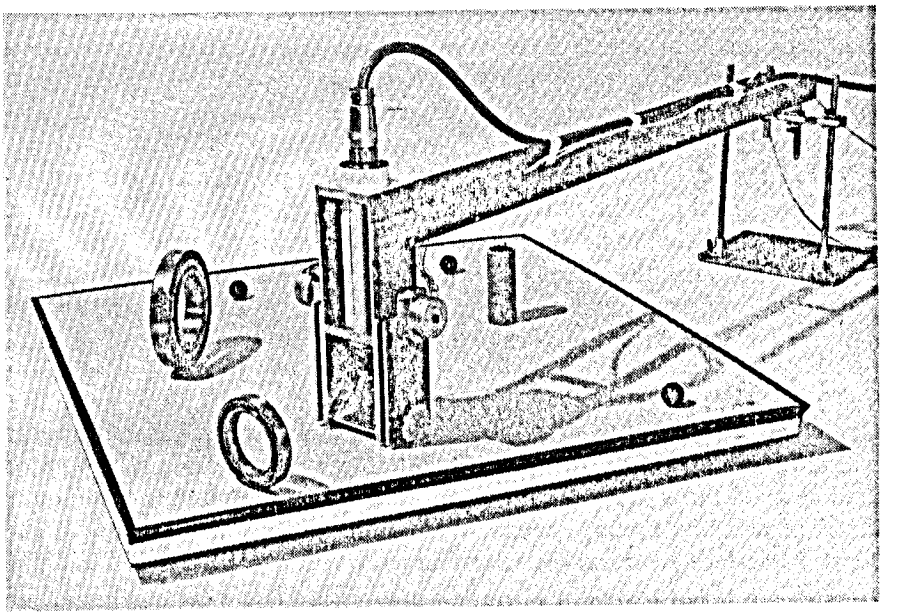


Figure 1. Freddy Mark 1 and his world

Various technical problems arise when such a device runs free. It is much easier to anchor it and allow it to push its world about. Our present world is a three-foot diameter sandwich of hardboard and polystyrene which is light and rigid. It rests on three steel balls and is moved by wheels driven by small

stepping motors, mounted on the robot. Provided that the weights of robot and slab are chosen correctly, a wide range of movements is possible. When stimulated, the motors drive for about half a second, causing the device to move about 5 mm or turn through about 4°.

A pair of bumpers, one in front, one behind, operate two micro-switches to signal contact with obstacles. A television camera mounted vertically sees the world through a 45 degree mirror. A wide angle lens provides extremely large depth of field – about 1½ in. to 3 ft at reasonably high light levels.

Television techniques are not particularly suitable for application to computer eyes, but as we already owned a closed circuit system, the cost of the project so far has been very low.

We have built the circuitry to sample and hold the instantaneous amplitude of the video signal at any point in the picture. The acquisition time is about 50 nanoseconds, which gives a horizontal resolution rather better than the vertical line resolution. Voltage levels from simple D–A converters set the *X* and *Y* coordinates and a simple A–D converter codes the sampled level for transmission to the computer. Bright-up pulses are added to the video signal to produce a cross superimposed on the monitor picture, to show an observer the position in the picture under study.

From the nature of television it is impossible to take fresh measurements 'upstream' without waiting for a new frame. If only a single measurement is made on each frame the data rate is prohibitively slow. It may be possible to achieve random scanning by using the RCA Alphechon tube, in which one gun writes a picture on a charge storage screen, which may be read non-destructively by a second gun.

The camera system which we have now is past its prime, and has a particularly poor signal-to-noise ratio, which can be seen as 'snow' on the monitor. A good system should be able to make measurements to about 1 per cent, whereas in this one the noise is about 5 per cent of maximum brightness.

Control is effected through an 8-bit, loadable command register, and an 8-bit readable conditions register.

General-purpose interface

Communication between Freddy and the ICL4130 is via a general-purpose interface. The aim of the design of the interface was to provide a high speed data channel through a 'bomb-proof' socket so that a number of different devices (including Freddy) could be connected or disconnected at will. The Mark I interface possesses the following features:

High speed. It is capable of operation at over 500,000 8-bit characters per second, but is limited by the ICL4130 transfer rate to about 300,000 characters per second.

Robustness. It will withstand even the application of 240 V mains to its input without permitting the signals to the ICL4130 to exceed their specified limits.

Long-range operation. Devices have operated reliably at over 300 ft from the computer. Maximum length of the connecting cable has not yet been determined.

Compatibility. The Mark I interface uses the same logical signals as a standard ICL4130 interface channel. It could, in principle, be used to operate a standard ICL4100 peripheral remotely.

Expandability. The Mark I interface is logically 'transparent' and of modular construction. Provision has thus been made for the incorporation of a logic module in the Mark II version to detect error states and take appropriate action, and also to perform certain simple operations.

The interface and its protection system have been approved by ICL; devices connected via the interface do not now need approval.

The interface and Freddy control logic have now correctly performed over 65 million operations, during normal public Multi-POP sessions.

Computer communications

Data transferred between Freddy, the device, and the 4130 is in the form of single word transfers of 8 bits (ODUM or IDUM).

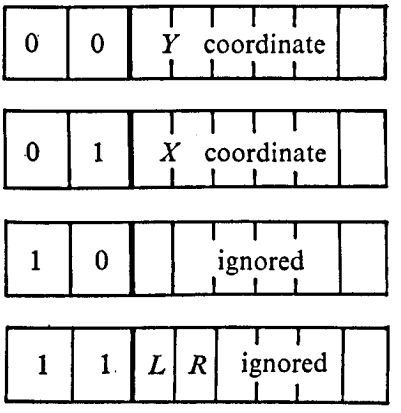
The POPMESS facility in Multi-POP gives the user a function doublet which outputs the least significant 8 bits of an item, or inputs a word from the robot.

```
e.g.   POPMESS([ROBOT])→RFN;
       X→RFN();                    for output,
or     RFN()→X;                    for input.
```

We now come to the decoding of commands and the encoding of sensory input.

Output words

The most significant two bits of the 8 bit word specify the command type:



The commands 00 and 01 cause the least significant 6 bits of the word to be loaded into the *Y* or *X* coordinate register. The 00 command initiates the sampling of the picture.

The third and fourth significant bits of the 11 command specify the directions of drive of the two motors. When the 11 command is given, the motors are started. No further commands may be given until motion has ceased, when the 10 reset command must be given before another move can be made.

Input word

The input word is structured as follows:

Motor flag	Left bump	Right bump	Picture flag	Comparator outputs

The motor flag is cleared to 0 when the drive command is given, and is reset to 1 when motion ceases.

Bumper flags are 1 if in contact with an object.

The picture flag is cleared by the set *Y* command, and is reset to 1 when the sample has been taken.

The four least significant bits give the states of the threshold circuits. Five brightness levels are discriminable:

0000 = 0 decimal = level 0
0001 = 1 „ = „ 1
0011 = 3 „ = „ 2
0111 = 7 „ = „ 3
1111 = 15 „ = „ 4

Level 0 is black, level 4 is white.

Programming practicalities

Motors. The sequence of operations for taking a step or making a turn are as follows:

1. Give the appropriate 11 drive command.
2. Wait several milliseconds for the motor flag to fall.
3. Monitor the flag and wait for it to rise again.
4. Give the 10 reset command.
5. Wait for about ½ second for the drive circuitry to recover.

The waits (steps 2 and 5) can be implemented by the simple function:

```
FUNCTION WAIT N;
L: N-1→N; IF N>0 THEN GOTO L CLOSE;
END;
```

WAIT(100) is sufficient for operation 2. WAIT(700) for 5. To save central processor time during the flag monitoring, the SWAPOFF system function (which swaps the user off the time-sharing system) should be used thus:

```
L: .SWAPOFF;
IF NOT(LOGAND(.RFN, 8:200)) THEN GOTO L CLOSE;
```

Picture sampling. The sequence of operations is as follows:

1. Set X coordinate with 01 command.
2. Set Y and initiate with 00 command.
3. Monitor picture flag until it returns TRUE.
4. Read and decode picture intensity.

In this case it is not necessary to use SWAPOFF in the monitoring loop in 3 above.

Y runs from 0 to 63 from the bottom of the picture upwards.

X runs from 0 to 63 from left to right in the real world (from right to left on the TV monitor picture).

If the picture is sampled randomly, on the average it will be necessary to wait 10 msec for the picture scan to reach the chosen point (since the frame time is 20 msec). The time taken to sample all the available 4096 points is thus 40·96secs.

If the picture is scanned horizontally, row by row, this is almost the worst possible case. The wait for each point is now nearly 20 msec, and the overall time to read all the points is 81·92 sec.

A suitable scanning scheme can markedly reduce these times. By setting X and sampling at several values of Y from top to bottom it is possible to race the TV scan down the picture and take at least 10 samples per frame (20 msec). (The command structure has been chosen for maximum speed; as the most significant two bits of the set Y command are zero, simply outputting the Y coordinate will initiate the sample.) In this way, the time taken to read the whole picture can be less than 8 seconds, the actual time depending upon the amount of computing between samples.

PROPOSALS FOR MARK II DEVICE

Our present Mark I is about the simplest possible machine and, as expected, is becoming obsolete. Its mechanical drawbacks are as follows:

(1) There are several awkward restrictions on possible movements. It cannot, for example, move to the perimeter of the world and then do an about turn.
(2) Rotating movements are slow because of the high inertia of the world.
(3) It is not possible to move directly sideways. It has to: left turn, forward, right turn. This makes building up pictures of an object from all sides very slow.

We have considered many schemes for overcoming these drawbacks. The overall scheme is now complete and many of the details are decided.

Mark II will retain the anchored robot principle for translations. The 'world' will consist of a five foot square of aluminium honeycomb or, perhaps, the present hardboard/polystyrene sandwich mounted on a compound XY slide. The slides will be rails of $2\frac{1}{2} \times \frac{1}{4}$ light alloy with ball bearing constraints. The drive will be through stainless steel multi-strand wires from two static servomotors. The swept area will be 10 foot square. Maximum acceleration will be 0·1 g up to a speed of 10 inches/sec.

Above the world will be a bridge from which may be slung various eyes, or bodies. If rotations are required they will have to be done on this bridge. Several designs are being considered: In the simplest a single small camera mounted on a rotating vertical mast will be limited to the number of rotations allowed by its cable. A second possibility is the use of periscope optics in which the entrance element is rotated to cover a panorama and the resultant rotation of the picture removed with a half speed dove prism. This allows complete freedom of movement and uses little space in the world at the cost of optical complexity. It seems rather difficult but not impossible to extend the principle to binocular vision.

The third possibility is an integrated hand/eye/body system. It seems clear that fairly good range finding is wanted, and that there is a need for higher acuity at points of interest. These needs would be satisfied with some form of triclops arrangement as shown in figure 2.

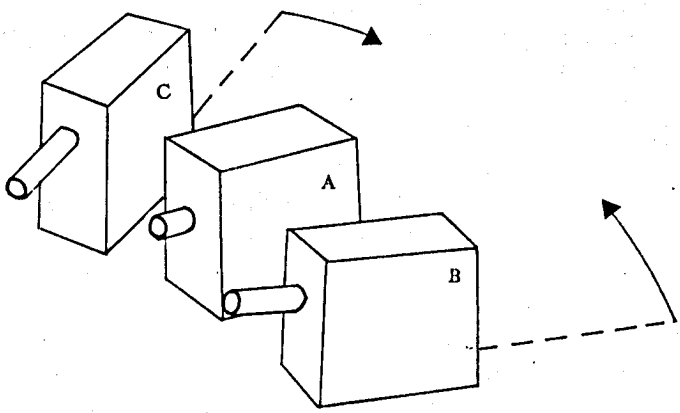


Figure 2. Three cameras for foveae and convergence ranging.

A is a television camera fitted with a wide angle lens, say 10 mm, which provides low acuity information of intensities over a 256 × 256 matrix on a 6 bit grey scale. This camera might well be a Pye Lynx, which could use a slightly improved version of our present Mark I sampling and conversion circuitry. Its function is to provide modest acuity wide angle coverage, similar to peripheral vision in humans.

On either side of this central camera are two others B and C, which are fitted with narrow angle lenses, say 100 mm, to provide the high acuity vision of the foveae. These cameras should really have image dissection tubes, but perhaps we may have to manage with vidicons. An actuator coupled to both outer cameras will cause them to rotate about vertical axes and so converge on objects at various distances. The angle of convergence necessary to fuse the two images will be a measure of the distance of the object. The moment of fusion is detected by a high positive correlation between the two video signals. It seems to be much more efficient to correlate outside the computer, using stochastic multipliers. In order to maintain focus over the required range of distance with such narrow angle lenses, it will be necessary to accommodate the outer pair of eyes. The cams and linkages required will be quite simple. The foveae cover small areas in the centre of the field. In order to examine and range any part, it is necessary to perform some sort of head movement, but before discussing this one should consider hands.

Figure 3 shows two pantograph linkages. The motion of H is a linear combination of the motions of P_1 and P_2, provided that P_1 and P_2, and H are co-linear. If P_1 and P_1' are moved away from each other, then H and H′ move towards each other. If P_2 and P_2' move together in the directions shown, then H and H′ are extended. If the palms of the hands are to be kept parallel then a secondary linkage will be necessary. The advantage of the pantograph linkage is that both actuators are static and well away from the palm position. Now imagine that the eye and hand assemblies are built separately on chassis a little larger than office filing boxes. In figure 4 the two chassis are joined by links at each side. Each chassis can be rotated relative to the link, which will allow the centre of the hand area to be moved vertically as seen by the eyes.

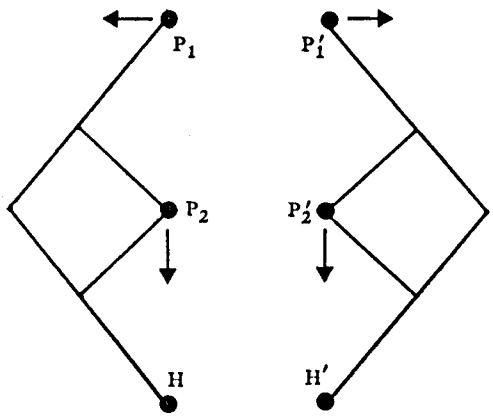


Figure 3. Pantograph arms

The link itself is able to rotate about its mid point on a bearing supported by a vertical fork. The fork hangs from a large horizontal beam and can rotate about a vertical axis. These last two motions are equivalent to body movement with eye and hand relatively locked. If this equipment is slung over the XY table, we will have the minimum number of degrees of movement for free activity.

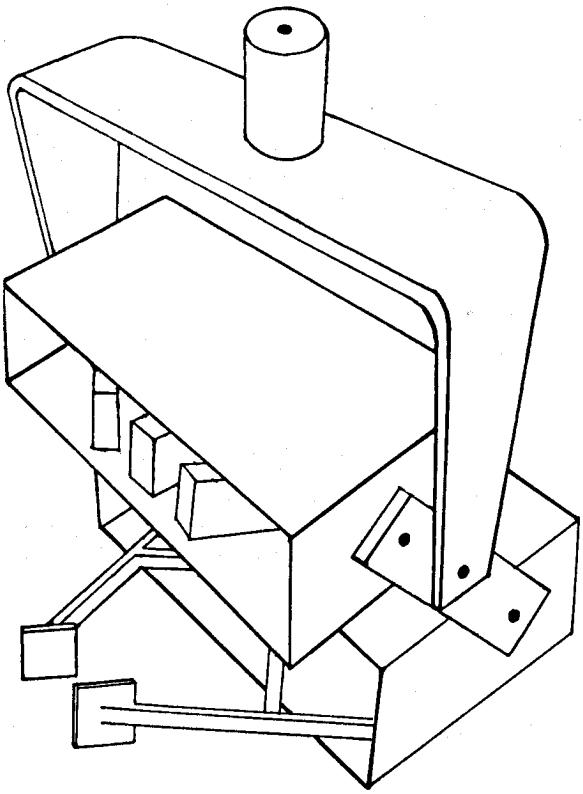


Figure 4. A possible hand/eye arrangement

PRIMITIVE COMMANDS

The following is an excerpt from the documentation provided with our Departmental Multi-POP consoles.

POP-2 PROGRAM LIBRARY
PROGRAM SPECIFICATION

PROGRAM NAME	LIB ROBOT PACK
SOURCE	H.G. BARROW, DMIP.
DATE OF ISSUE	September 1969

Description

This package contains a set of basic functions for on-line operation of the receptors and effectors of the Mark I version of the robot Freddy.

How to use the program

The package assumes that the function variable ROBOTDATA has been assigned the repeater function for the robot, so that this must be done before the package is compiled, namely:

```
VARS FUNCTION ROBOTDATA;
POPMESS([[ROBOT ⟨n⟩]])→ROBOTDATA;
```

where ⟨*n*⟩ is the time in minutes for which the robot is required.

Note that the result of this POPMESS is a doublet for both inputting from, and outputting to the robot.

The package is then compiled by typing:

```
COMPILE(LIBRARY([LIB ROBOT PACK]));
```

The following facilities will then be available:

Constants. A number of useful constants are defined in the package. They are listed later.

Touch. There are three Boolean functions which sample the two bump-detecting switches:

BUMPLEFT()	TRUE if left bumper is operated, else FALSE
BUMPRIGHT()	„ „ right „ „ „ „ „
ABUMP()	„ „ either „ „ „ „ „

Note that each bump detector responds to contact in front or behind. Where the object actually is can be determined by remembering which way Freddy was moving when the bump occurred.

Movement. There are two movement commands, each giving a Boolean result:

```
WALK(⟨distance in millimetres⟩);
TURN(⟨angle in radians⟩);
```

As moves are in fact quantized, the distance Freddy actually walks may differ from the specified distance by ±2·5 millimetres. If the WALK argument is negative, Freddy moves the specified distance in reverse.

Turns are similarly quantized, and actual angle turned may differ from that specified by ±0·035 radians. If the argument is positive, Freddy turns left, if negative he turns right.

The Boolean variable BUMPON determines whether or not Freddy stops on contact with an object.

If BUMPON is FALSE, he ignores bumps, and the results of WALK and TURN are always TRUE. (It is thus possible to push objects.)

If BUMPON is TRUE, then if no obstacles are encountered, he steps or turns the full distance and returns the result TRUE. If an obstacle is touched, he stops immediately and exits from the move function with the result FALSE.

The variable LASTWALK contains the distance actually moved during the last call of the WALK function. LASTTURN contains the angle turned during the last call of TURN. If no obstacles are encountered, the value will be approximately equal to the argument of the function. If a bump causes exit from the move function, the value will be the distance actually travelled (or angle turned) up to the moment of impact. (i.e., WALK(-LASTWALK); will return Freddy to his starting place before the last move).

Dead reckoning. The move routines endeavour to maintain a running estimate of Freddy's position and orientation. Position is measured in Cartesian coordinates, in millimetres. Orientation is measured in radians and is the angle between the x-axis and the direction in which Freddy is pointing. The angle is restricted to the range 0 to 2π.

The estimates are held in the variables:

XNOW, YNOW, ANGNOW

(Typing CTRL and G during motion will only introduce an error of one quantum of angle or distance.)

The function SETPOSITION(X, Y, THETA); sets the estimates to the specified values.

The function PRPOSITION(); prints the current values of the estimates.

Vision. Note first that the TV camera looks into a mirror. What is seen on the TV screen is therefore a laterally inverted version of the real world. The following refers to the *real world* and not the TV monitor.

The picture is sampled at one of 4096 points in a 64×64 array. Bottom left is (0, 0), top left is (0, 63), top right is (63, 63) and bottom right is (63, 0).

The function PICINT(X, Y); samples the picture at point (X, Y) and returns an integer value for light intensity from 0 to NBRIGHTLEVEL. In the Mark I robot, there are five possible values, 0 to 4. Black is signified by 0, white by 4.

The function AVINT(X, Y, N) returns the mean over N samples of light intensity at (X, Y) to reduce effects of noise on the TV signal.

The function DISPLAY(FN, INC, X0, X1, Y0, Y1) prints a picture (via CUCHAROUT) using the characters:

⟨space⟩ , ↑ + * N M to simulate a grey scale.

FN is a function of two arguments, which yields a result (integer or real) in the range 0 to NBRIGHTLEVEL, and INC specifies the increment, and X0, X1, Y0, Y1 specify bounds for X and Y.

The function DISPIC is DISPLAY(%0, 63, 0, 63%) and thus will display the values of FN over the entire field.

e.g., DISPIC(PICINT, 8);

```
%%%%%%%%%%%%%%%%%%%
%                               %
%                       .   .   %
%               M   M   M   .   %
%   M   M   .   .   M   M   M   .   %
%   M   M   *   +   M   M   M   +   %
%   *   M   M   M   M   M   M   M   %
%   M   M   M   M   M   M   N   N   %
%   M   M   M   M   N   M   M   M   %
%%%%%%%%%%%%%%%%%%%
```

The function STATS(FN, INC, X0, X1, Y0, Y1) returns mean and standard deviation of the values of FN(X, Y) over the range X going from X0 to X1 in steps of INC and Y from Y0 to Y1 in steps of INC. In particular, FN may be PICINT or AVINT(%5%) etc.

e.g., STATS(PICINT, 2, 10, 30, 15, 28)→MEAN→STDEV;

puts the average light intensity over the window into MEAN and the standard deviation into STDEV.

The function PRSTATS(FN, INC, X0, X1, Y0, Y1) calls STATS and prints the results.

If an object can be seen to rest on the world by the robot, then the position (Y value) of the foot of the object on his retina can be translated into a distance.

The function GPDIST(Y) yields distances, in millimetres, from the wheelbase to the projection of rows on the retina, for values of Y from 0 to 31, on the ground plane. (The horizon lies between Y=31 and Y=32.)

Global variables

Useful Constants

PI	3·1416
TWOPI	2*PI
WORLDRADIUS	radius of Freddy's world, in mm.
FREDLENGTH	length of Freddy from front to back, in mm.
FREDWIDTH	overall width of Freddy, in mm.
WHEELSEPARATION	separation of wheels, in mm.
EYEHEIGHT	height of eye above ground, in mm.
EYEOFFSET	distance of lens in front of axle (actually negative), in mm.
EYEELEVATION	angle of elevation of view, radians.
PICANGWIDTH	angular width of picture sampling array, in radians.
PICANGHEIGHT	„ height „ „ „ „ „ „
NPICWIDTH	number of sampling positions across picture –1.
NPICHEIGHT	„ „ „ „ down „ –1
NBRIGHTLEVEL	number of distinguishable brightness levels –1.

[*continued*

Useful Constants (contd.)

MMPERWALK	quantum of forward motion, mm.
MMPERBWALK	,, ,, backward ,, ,, (negative)
RADPERLTURN	,, ,, left turn, radians.
RADPERRTURN	,, ,, right ,, ,, (negative)
PICHAR (0 to 7)	array, holding characters printed for grey scale.

Useful Variables

XNOW, YNOW, ANGNOW current estimates of position and orientation.

LASTWALK	distance travelled during last call of WALK, mm.
LASTTURN	angle turned during last call of TURN, radians.
BUMPON	if TRUE, contact causes exit from move functions.

Useful Functions

ABS result is absolute value of argument.

ABUMP, BUMPLEFT, BUMPRIGHT,
SETPOSITION, PRPOSITION,
WALK, TURN,
PICINT, AVINT,
DISPLAY, DISPIC,
STATS, PRSTATS.

Other Globals

Functions WAIT, FLAG, DRIVE, MOVEPAUSE, MOVEDONE, ANGUPDATE, WALKUPDATE, MOVEIT, CHARLOTS, ROBOTDATA.

Store used

The compiled program occupied about 4 blocks of core.
The uncompiled program occupies 17 sectors of disc.

Acknowledgements

The greater part of the hardware work was done in the Department's Bionics Research Laboratory directed by Professor R.L.Gregory and the programming and interfacing work in the Experimental Programming Unit directed by Professor D.Michie. The authors wish to acknowledge financial support from the Nuffield Foundation (S.H.S.) and from the Science Research Council (H.G.B.).