UNIVERSITY OF PENNSYLVANIA
THE WüORE SCHOOL OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

PLAYING IT BY EAR: AN INTELLIGENT SONAR GUIDED ROBOT

Russell Lennart Andersson

Ph ilaaetphia, Pennsylvania

Way, 1981

A thesis presented to the Faculty of Engineering and Applied Science of the University of Pennsylvania in partial fulfillment of the requirements for the degree of Master of Science in Engineering for graduate work in Computer and Information Sciences*

Ruzena *• aajcsy

Ar av inG &• Joshi

This thesis describes a Self Contained Independent Mobile Robot (SCIMR) capable of understanding and roving about in a simple but very real world: the Moore School's hallways and intersections. SCIMR learns the topology of his world much as a messenger or taxi driver would, by being told how to get places and piecing the instructions together to form an internal map. SCIMR requires fewer directions as he learns, since new locations may be described relative places SCIMR already knows.

SCIMR is capable of perceiving and remembering his environment, and using his memories to guide his actions. The robot is totally self contained and requires no external processors or guidance equipment. Unlike wire or stripe guided vehicles, SCIMR requires no environmental preprocessing. Efficient control methodology, especially multiprocessing and multitasking techniques, allows small computers to drive the robot in real time without processor induced delays.

TABLE OF CONTENTS PAGE

TABLE OF CONTENTS PAGE

1. Introduction

Men have long dreamed of making intelligent machines. Ultimately, we desire a robot which can enter a complex new (or old) environment, understand it, and then operate within it. The robot which is the subject of this thesis is a humble beginning step towards a self contained independent mobile robot (SCIMR, for short, pronounced skimmer) capable of understanding and roving about in a simple but very real world: the Moore School's hallways and intersections.

A mobile robot like SCIMR might have applications in automated warehouses, mail and parcel sorting, as a messenger, a janitor, or a radiation monitor in nuclear power plants. SCIMR is applicable when there is a need for autonomous but directed transportation of things which may be attached to it in some way.

For example, in a warehouse application, SCIMR might be told to go to a specified location and actuate a picking device, and then return to the load/unload dock. Instructions could be issued automatically via a tie-in to the standard warehouse processing system.

A computer interfaced Geiger counter would turn SCIMR into a mobile radiation monitor. SCIMR would follow a predefined path monitoring radiation level. Any abnormal radiation would be detected and localized, even if localization requires investigation of substantially remote areas not normally monitored. Reproduction of the monitoring effectiveness of a mobile robot by other means

would require a very targe array of fixed Geiger counters* or periodic checks by humans* A robot would increase the reliability of the measurements« since the robot will not (better not!) yet borea$_y$ is very thorough* and works *around* the clock* Use of a robot also prevents the unneccessary exposure of any persons to radiation*

A messenger robot would also *require the* ability to go to new unexplored areas without "site preparation*

SCIWR consists of three interconnected on-board microcomputers, a rotatable sonar system, two motorized wheelst one castoring wheel, and an automobile battery mounted on a foot and a half square platform (see picture)* Part of SCIPR's charm lies in this low parts count and cost*

Parallel I/O ports loosely couple the three computers* A simple operating system supervises up to eight concurrent tasks en each processor* An interprocessor communication facility built into the operating system allows tasks to start and pass data to tasks on the same or a different processor•

Multi-processing and -tasking is an important aspect of SCIMR* Experience gained here is applicable to other real time processing systems*

Processing and I/O interfaces are distributed among the processors* One computer controls the sonar and *stepper* motor turning it* A second computer regulates the motor speeas and performs related calculations* The third computer implements high level intelligence and interacts

*ith a human via a terminal during initiat*iza*tion• Once the machine is ready to go, all connections »ith the outside yorld are severed, and SCIMR is on his own.

2. Past Efforts

I will describe two other efforts at producing a semi-intelligent rover: the Stanford cart, and the French HILARE, and contrast them with SCIMR.

A very significant difference between either robot and SCIMR is the amount of processing power used. The Stanford cart apparently uses all available time on a large DEC KL-10 processor. HILARE uses onboard microcomputers, a local minicomputer, and a remote mainframe. In contrast, SCIMR uses three onboard microcomputers to handle all of its processing. SCIMR is truly self contained.

2.1 The Stanford Cart

The cart uses vision exclusively to navigate, and has no other sensors. According to its descriptors [Reference 3], precise locomotory capabilities have been sacrificed in the interest of simplicity; the vision is required to compensate for the cart's shortcomings in this area.

The vision system performs stereoscopic picture analysis: the camera is mounted on rails and moved back and forth to generate picture disparity. Pictures are broadcast via a local TV link to a remote receiver, where the pictures are digitized and fed into the KL-10. The KL-10 performs crunching to be described, and then outputs a 6 bit word which is transmitted via radio link and a dubious encoding method, to the robot, where the six bit code fairly directly drives the power transistors controlling the robot. Due to the extensive processing, and despite the large processor,

completion cf the feedback path requires about 15 minutes! This necessitates a very smalt motion per iteration: a meter or so (every 15 minutes)* Extensive testing of such a system is obviously prohibitively tine consuming* In contrastf SCIHR moves steadily down hallways at a rate of one foot per second, completing its feedback path in 1*4 seconds •

«hy weutd anyone even bother with the cart*. if its performance is so bad? The cart has a much broader view of the «Grtd than SCIMR* Instead of hallways and intersections! the cart deals with collections of objects spread around blocking its path* The cart must avoid these obstacles* It is interesting to note that the cart researchers apparently did not consider the possibility of the cart actually knowing where it was going$_t$ unlike SCIMR* The cart's world model is a monolithic space populated with circular obstacles* Any object in three dimensional space is mappec into its circular projection onto the floor. The projection is stored in memory as an (x,y) pair and a radius of the obstacle* To simplify later processing, each radius is augmented by the 'safe" radius of the robot*

The visual processing consists of two steps* Firstt areas of interest named features are identified* A feature is defined as *an* area having a local maximum of an interest operator which measures the local gray level gradient* A second operator will locate corresponding features in two images* Given multiple images from the camera taken at

different positions along the track, the distance to each feature may be found* Given two images taken before and after a cart movement, the distance travelled in the motion may be found* The visual processing system's primary prables is in aealing with environments which are featureless according to its criterion* In this situation, crashes and general confusion are likely* An amusing problem with running the cart outdoors is that shadows may move substantial distances between iterations* The shadows make ideal features due to their high contrast ratio* and cause tHe cart to be confused and get lost*

2*2 HILARE

The French HILARE robot attempts to make use of multiple sensory systems for perceiving the world* The primor> senses are vision*. laser rangefinding» and short ultrasonic ranging* I have much less information presently about HILARE than the cart, so I can report on only a subset of its capabilities* I have seen a film of the robot driving around an irregular manufactured wall using its sonar system* HILARE uses a fixed array of ten sonar transducers* In the demonstration shown, progress was slow, at least partially due to the precise orientation constraint given to it* The robot demonstrated accurate low level control (HILARc uses stepping motors] but did not display any "cognitive" capabilities* Information concerning successful use of HILARE^s multiple senses remains to be seen*

Its *crtd model is based on a single level polygonal partitioning* Obstacles are required to be convex* and areas, although not necessarily convex, must not contain any obstacles* At the end of this thesis, a greatly generalized but scaewhat related yorld modelling systeo will be presented*

3. The MUPT Operating System

3.1 Multi-Tasking

3.1.1 Rationale

The MUPT (for Multi Processor, Tasking, pronounced muppet) operating system is designed to support multiple tasks on many processors, and provide communication and synchronization among them. Multitasking is crucial to the efficient operation of a real time control system. Its absence necessitates the construction of a large super-program, typically configured as a polling loop, and a bunch of routines. Each routine is forced to use flags to figure out what to do, or implement its own multitasking. Such systems are inefficient of space and time, and painful to write, debug, and maintain.

MUPT eliminates this problem by providing a coherent mechanism for specifying the occurence of an event, and for waiting for the occurrence of a specific event while preserving machine state. In MUPT, events are named semaphores. Beware: the definition of MUPT semaphores is different than other P and V type semaphores, as will be discussed.

Two primary primitives are supported, TRIGGER and WAIT. TRIGGER accepts two inputs, a value and a semaphore number. It sets the semaphore to the value, and starts any tasks which have been waiting for that semaphore. WAIT has a single parameter which is the number of a semaphore to be waited on. If there is no value in the semaphore which has

not been processed by this task, the task is put to sleep until a value appears. When a value is present, either initially or at some later time, the task is scheduled for execution.

A subsidiary POLLER function tests whether or not an unprocessed value is present in a specified semaphore, and returns either NO or YES,value. POLLER allows a task to wait for one of several events to occur, an impossibility using the standard WAIT, which only waits for a single event. It is also used by interrupt subroutines, which cannot be put to sleep like normal tasks. Both cases are comparitively rare.

3.1.2 Comparison with P and V Operators.

The MUPT semaphores have two major differences from conventional semaphore systems: 1) a distributive nature, and 2) efficient implementation based on bytewide bit operations rather than linked lists. The latter difference was the primary reason for the original design, but in the course of making it more elegant to improve efficiency and effectiveness in several domains, the underlying structure of 1) was discovered, and guided successful implementation.

What does 'distributive nature' mean? First, consider a conventional semaphore system. A P operator requests a resource. If non-zero, the resource counter is decremented and execution continues. Otherwise, execution is suspended until a resource is made available, by the V operator, which unconditionally releases a resource. No actual resource is

involved in the P ana V operations: it is assumed that ever>bod> is talking about the same thing* which is itself elsewhere* Some more advanced systems say actually manipulate pointers to the real resources* Suppose some task is transmitting a stream of values to a semaphore* Under the P anu V system, a given value will be sent to a single receiving task* If a second value arrives before the first is reroved, the second value is stored, so that the next two requests wilt get each value sequentially*

On the other hand, the «UPT system will distribute each value to every task that wants it* If a second value arrives before the first has been processed, it supersedes the previous value*

Semantically, the P and V operators control resource allocation: who gets what when* On the other hand, the WUPT semantics specify data availability: when is data available to solve *my* problem?

In a real time control system, this seems more useful* Typical processing tasks operate in a receive-compute-send loop, with computation beginning when all required data is availaole* If more than one task is processing a set of aata, such as a junction type or sonar reading, both tasks will be processing each individual piece of data* The P and V operators would require the data to be transmitted twice by the sender* which is undesirable since logically the sender does not really care who is using his data; this information (who is receiving a piece of data) should belong

with the actual receivers*

In typicat applications! two tasks communicate bidirectionalty$_f$ using one or more semaphores in each direction* If an extra value should happen to be introduce into one of the semaphores! causality will be lost: the system is desynchronized* In normal operation each task thinks that data it has just sent is causing values to be returned to it* An extra value destroys the causality: instead of receiving a value based on data just sent, the task receives aata based on the previous data sent* This situation sight arise if we wished to ignore a result from some other procedure* The next time we wait on the semaphore$_f$ *e yet the "ignored" value, instead of one based on setae action just performed* From then on, we are out of sync$_t$ *and* everything is downhill* For this reason, multitasking systems must be carefully constructed to absor all results from each function* Alternatively! POLLER may be used to clear a semaphore* The desynchroni2at ion scenario above may occur in either *P and* V or MUPT semaphor systems- A possible fix night be to unconditionally clear semaphore when a wait is started on it* So far, this has seemed unnecessarily extreme, since it would prohibit overlapped operation of communicators*

3.1*3 Task Switching

Unlike multitasking systems found on large computers, tasks have explicit control over when they relinquish the processor tc another task* A task in an infinite loop will

stop all high level tasks in that processor dead. This seems a terrible problem. In a general purpose system, it would be fatal, but SCIMR is far from general purpose programming. Indeed, the failure of a single task means that SCIMR has a heart attack anyway, so that you might as well quit. The only disadvantage is that the system monitor task is also hung: it is not possible to discover which task has died. The system must be restarted. At this point, a clever initialization routine can save the previously running job number, so that the failing task can be uncovered. Note that all processor programs use active initialization of all variables, since programs are restarted without a new copy being downloaded (at 300 baud, horrors!).

Why not switch tasks after each interrupt? Implementation would be straightforward: after each interrupt, jump into the dispatcher, and off we go. The additional program would be on the order of several bytes, or perhaps save a few; hardly consequential.

The primary advantage of not switching tasks on each interrupt is that unitary code sequences may be constructed which are guaranteed to complete before any other high level task on the same processor is allowed to execute.

In conventional systems, such an effect might be obtained by disabling interrupts for the duration of the sequence. This is impossible for real time control systems like SCIMR. In the sonar processor, interrupts occur every

180 microseconds or so while a distance measurement is being taken. The 400 Hz system clock must also be processed in both sonar and motor processors. Disabling interrupts would wreck the I/O system. Furthermore, disabling interrupts would disable the interprocessor communication system interrupts. A task could not specify (and complete) a unitary sequence requiring data input from a remote processor. Note that the processor hardware supports only single level interrupts.

Most importantly, the data transmission XMIT must be unitary and interrupts must not be disabled. If XMIT is not unitary, a task switch could occur in the middle of a transmission, and another transmission requested by the started task. This would result in utter confusion. If two processors should begin XMITs at approximately the same time with interrupts off, a deadlock would ensue because the receive interrupt would be locked out.

The need for unitary high level task operations could be satisfied by a specialized mechanism: a flag to indicate that a task switch should not occur. The discretionary task switch has proven perfectly viable: this last mechanism has not been implemented, although it easily could be.

3.1.4 Usage and Implementation

The MUPT implementation is particularly efficient due to the byte parallel bit operations used. MUPT supports up to eight tasks. Any subset of the tasks can be represented in just a single byte. In particular, MUPT represents the

currently running job, pending tasks, tasks waiting on each semaphore* and tasks for which a semaphore value is available as a (separate) single byte*

Thirty two different semaphores reside in each processor* A semaphore contains a value (a single byte), ana two tit flags for **each** task on the machine* Accordingly*, each semaphore occupies three bytes of memory*

The MUPT operating system effectively supports the SCIHR software* The semaphores and tasks on each processor may be found in Tables 1 and 2* This usage is in accordance with the original expectations that motivated the development of MUPT* There is room for expansion, yet all of the facilities are being used*

3*2 interprccessor Communication

3*2*1 Strategy

Multiple microprocessors must communicate effectively to solve common problems* The backplane pinout limits the number of signals available for intercommunication* The motor ana sonar computers each communicate with the central control computer fn a linear or specialized star configuration* The intercommunication topology corresponds to the heuristic structure of the processing: a sensory, decíbicn, and effectory system*

There are several possible control strategies for intercommunication: hereis$_t$ mutual agreement, and give me* In the hereis strategy*, data is transferred when the sender has new data available, regardless of whether or not a

receiver *desires* the datum. Data is sent via mutual agreement when data is available from a sender, and when a receiver desires the data* Data is sent *ur\6er* the give me strategy when a receiver desires data, regardless of whether or not data is available*

in the give me strategy! an address specifying what is to be given must be transmitted from the receiver to transmitter (of the real datum), and then the line must be reversed so that data may be communicated from sender to requester* A give me strategy corresponds to acquiring the value of a variable on demands although it resides in a different processor*

The mutual agreement format is extremely difficult to implement in the general case of multiple sending and receiving tasks* An unreasonable simplification would be for the sender to broadcast the address of the datum to be sent and maintain it until a receiver, seeing that address, acknowledges its desire to complete the transfer, 'by mutual agreement*" However* this has several serious bugs, basically resolving around the fact that once started by a sender a transfer may be completed only by *an* agreeing receiver* This suggests numerous deadlock possibilities, namely* that nobody wants a value being sent, or that the order of sending and receiving are different! especially in conjunction with multi-tasking*

For these reasons, the sender driven approach was adopted* *The* requirements for the interconnection subsystem

were: that transfer be initiated and controlled by the sender, that it be fault tolerant, and that it integrate and cooperate with the rest of the MUPT operating system.

The hardware available for implementing intercommunication is limited to 8 parallel I/O lines and a single control line. To allow handshaking, the 8 parallel I/O lines are logically partitioned into a single control line from the receiver to sender for handshaking, and 7 address/data bits from sender to receiver. The main control line has directionality from sender to receiver. With this scenario, the direction of data transfer over a given line is constant. Note that each pair of intercommunicating processors has a port in each direction.

Since data transfer is initiated by the sender, the receiver may be executing arbitrary code when a transfer is to begin, accordingly, the sender to receiver control line is set up to cause an interrupt in the receiving processor. Data may be transmitted only seven bits at a time over the interface due to the presence of the handshake reply line, however, the software occludes this limitation, as we shall see.

Data transfers occur only within a single operating system subroutine, by (unenforceable) rule. This subroutine, once entered by a task, retains control of the processor until the transfer is completed. If the processor were relinquished by the transmitter midstream, another task could enter the transmitter and cause complete confusion.

Interrupt handlers may not call the transmit subroutine for several reasons. First, the main line code may already be executing a transfer, which would cause large complications, resolvable only by excessive software gamesmanship. Secondly, since a transfer may take a while, especially if the receiver is currently executing its own interrupt service routine, interrupts may be locked out on the sending processor, to ill effect.

Because of the restrictions on interrupt level transmits, deadlocking, and fixed storage allocation, transmissions from the sonar to motor processors or vice versa cannot be made directly, but must be supervised by a task in the control processor.

An actual transfer takes place as follows: the upper 7 bits of the data item to be transmitted is put on the bus, and an interrupt generated in the receiver with the main control line. The sender waits for an acknowledge on the handshake line, which is a transition from a 0 to a 1. After this occurs, the address and lowest data bit are placed on the data lines, and another pulse generated on the main control line. After the handshake line is restored to zero by the receiver, the sender is allowed to go on its merry way.

3.2.2 Fault Tolerance

The description above is in terms of the point of view of the sender. What must happen in the receiver? A simple reception algorithm might go as follows: wait for an

interrupt* read in a data byte, acknowledge*. wait for a senc pulse while stilt in the interrupt routine, read in an adoress/data byte*. acknowledge*. store the byte in the address*. and return from the interrupt routine*

The original implementation had much this flavor* Unfortunately, it was discovered that fake interrupts could occur cue to the presence of the noisy DC drive motors *and* the low drive capability of the PIA chips implementing the interfaces* If the receiver implementation above encounter! this circumstance, it will hang the receiving machine* Thit was discovered in the obvious way, by experience* A spurious interrupt causes the receiving processor to enter its interrupt routine, and stay there*. since a second pulse may net occur for quite some time, perhaps never* In any case, the sender and receiver are out of sync* In fact, Doth machines »ay eventually hang* This is clearly undCceptable *

The synchronization point in the receiver is the cause of the problem* The current fault tolerant receiver always returns right after Handshaking, and uses the state of its own handshake output to determine its actions upon entry* The *transier* proceeds as follows: interrupt! read, acknowledge*. *r^turn^* interrupt*. *readf* acknowledge! store, and return* The hanoshake output determines the interpretation of the incoming byterO data, 1 address and store #

Removing the loop from the receiver prevents the

receiver of a spurious interrupt from being 'hung,' but fails to cure the synchronization problem. Synchronization failure is detected by the sender at the beginning of a send operation. If the incoming handshake bit is a 1, it indicates that the receiver is expecting an address, which is not what the sender is planning to send. This constitutes detection of failure. The next step is to take corrective action. The only thing the sender may do at this point is send an address, as far as the receiver is concerned. Rather than send the address the sender is planning to send, the sender supplies a fake address, hexadecimal 1F. This address is reserved for error handling. Essentially, it causes the data from the spurious interrupt to be sent to the bit bucket. Once this has been done, the sender may go on with its real business.

All communications failures are counted by the control processor. The two main processor transmit routines directly log failures as they are detected. A special task waits on main processor semaphore 1F, and increments the error counter when something is thrown into the bit bucket.

Experience indicates that at most several failures per hour may be expected under adverse conditions (not fully understood). Although this may seem like relatively few, it must be remembered that without automatic failure detection and correction a machine crash would occur, possibly (and usually, by Murphy's Law) miles from a terminal where SCIMR may be restarted.

Data integrity during transfers is not checked. Accordingly, whether or not data errors actually occur is not known. Operationally, the robot is highly reliable, and does not crash without outside justification. Data transmission is significantly different than control information transmission. A transfer request is being received by an edge triggered device continuously sensitive to the short noise bursts generated by the drive motors. On the other hand, data lines are effectively read during a quite short period of time on the order of 500 nanoseconds, during which a noise burst is statistically unlikely.

Empirically, transfers take around 200 microseconds. During interrupt handling by the receiver, they may take considerably longer, depending on the time required to process an existing interrupt. The required transfer rate for a MUPT program ensemble must be minimized to maintain a high execution rate. This is generally accomplished by efficient task partitioning to achieve data locality.

3.3 Improving Communications

The current intercommunication system seems to operate well, and not cause any performance related problems. If we suddenly needed to transmit much more data, how could we do it?

The first stage would be to widen the data paths between processors to 16 bits plus two handshake lines each way, doubling required hardware. This would allow a one step transmission system, with both address and data being

transmitted *at* the sane time* Address and data parity could also be checked (preferably with hardware support)* Speed woulc ae roughly doufeled* A fault tolerance scheme would have to be worked out, but seems feasible*

At the same ti«e$_f$ ft might also be useful to add a path between the sonar and motor machines*

The next stage would be to have a shared memory which would store the semaphores and operating system control information* Many problems would climb out of the woodworks in the synchronization and deadlock area*

3*4 fchy Three Processors?

Several people have asked why I have three small computers*. rather than a single "big' one* 8y a "big" ntachine$_f$ the apparent conception is that of a single board computer, or maybe several S-10Q caras, with 30 or 40 Kbytes of RANf and a grab bag of peripheral ports of varying types* The processor might be a Z&0. TI 9900$_f$ or even 18086. Why not use such a computer? It would be obscuring the truth to say that availability did not ptay a part in processor selection* Nonetheless, it is my belief that such a machine woulc oe unable to deal with the real time processing tasks by itself* A considerable amount of processing must take place simultaneously with rapid processing of interrupts* Despite the fact that a larger processor might have some fancier instructions! most of the processing actually performed by the algorithms has more of a load, test, and store flavor than that of arithmetic computations* The

speed would still be limited by the memory cycle speed and instruction encoding. Since the direct data reference requirements are small, most references can be made using the 6800 direct addressing mode, yielding two byte instructions. By contrast, more 'advanced' processors typically require 3 or 4 bytes, occupying more space and executing slower.

The three small processors execute interrupt driven real time control tasks more effectively than a single larger machine. The tasks are also partitioned logically, compartmentalizing processing tasks and simplifying the debugging effort.

Of course, embedding some of the algorithms, especially DC motor control, sonar distance counting, and stepper control in hardware might enable a single multitasking processor to be used.

Power consumption must also be considered in the design of any battery powered system.

4. Sonar

The performance of the SCIMR robot and its programming are heavily influenced by the characteristics of the sonar system which provides its input.

The sonar system consists of a Polaroid ultrasonic transducer, driver board, stepping motor to turn the transducer, and an interface from the driver and stepper to the sonar control computer.

4.1 Control Software

The sonar driver accepts as an input a scan type number. The possible scans are shown is Figure 3. The top level driver is a MUPT task which waits for a scan number in a semaphore and then looks it up in a table which defines the scan pattern. Each scan pattern may specify several measurements, each of which consists of an angle bearing and a variable number of local semaphores the distance is to be sent to. Table entries are interpreted sequentially; no operation can commence until the previous operation has completed.

Suppose that a distance measurement is to be taken in a given direction. We will chart the course of events, which are implemented by several interrupt level tasks in the sonar control computer. First, a direction is specified and sent to the handler using MUPT semaphores, which are polled by the handler. The direction is in units of steps in an absolute coordinate system with 0 straight ahead, positive left, and negative right. There are two hundred

(200) steps per revolution of the sonar stepping motor, or 1.8 degrees per step.

The transducer is rotated towards the correct direction at 200 steps per second. When it gets there, a delay of 5 msec allows noise to die down and the stepper to stop mechanically oscillating. At this point, the stepper interrupt task, which drives the actual stepping motor, signals the power interrupt task, and goes to sleep waiting for another directional coordinate.

The Polaroid sonar driver board is intended for operation in cameras, and may seem a bit strange. It will initiate a sonar measurement whenever its power is turned on. Once turned on, the power must remain on for 100 msec, and then go off for at least 40 msec. To achieve optimally fast operation, the rotary positioning task and power control task are overlapped; yielding the multiple tasks mentioned above. The power control task controls, literally, power to the sonar driver, via a high power (2.5A) driver on the interface board.

The power control task waits for a power on signal, then turns on power, and after a delay, the distance measuring interrupt task. The power task then waits 100 msec, turns power off, waits 40 msec, and then begins polling the PULSE flag again. Accordingly, when the stepper task commands a pulse to begin, it may not actually begin until some time later, as determined by the POWER interrupt task.

The STEPPER ana POWER tasks above measure time in 400 *H2* units (£.5 msec), which is wonderful for what they do* However, the actual oistance measurement requires much higher clock rates* The sonar driver board supplies two outputSf XftIT and ECHO* The elapsed time between these signals is linearly proportional to the distance to the (nearest) object* The distance measurement task computes the distance to the object by counting interrupts between X*IT' and ECHO* The time between interrupts is 178*7 usec$_f$ which corresponds to 0*1 feet* The distance measurement task, after initiation by POWER, waits for XNIT and begins counting until ECHO occurs* If ECHO occurs before 0*9 feet it is a false echo* The sonar driver draws 2»5A for a shor period of time during transmission! inducing noise on the ECHO line* The actual transmit burst is a chirp at 6G$_9$ 57, 54, ana 50 Kh2 to achieve maximal target scattering* If th distance measurement task receives no ECHO within 25*5 ft, it returns that distance as the actual* Polaroid specifications indicate that the unit will operate up to 35 ftt but the lost range is not significant in SCIWR's environments and allows him to store distances as a single eight bit (unsigned) integer*

4.2 Operational Characteristics

The beamwidth of the sonar pattern is twenty to thirt deyrees (see Figure 7)* Predicting actual response is difficult, and requires a general numeric simulation of target scattering characteristics as a function of area ana

range. In general, perception of scattering surfaces at an angle may be expected to yield distances longer than the closest distance of approach within the beamwidth.

A very significant operating limitation was discovered regarding smooth hard surfaces such as doors and certain walls: they operate as an ultrasonic mirror. When the sonar attempts a measurement in the direction of a mirror, it instead sees the distance to something else, generally the wall on the other side of the hall. This is precisely akin to making laser rangefinding measurements into optical mirrors. The mirror problem makes the interpretation of sonar rangefinding data much more difficult, especially since mirrors (doors) occur in the area of intrinsically confusing objects, namely, intersections. SCIMR is inoperable in areas with totally mirroring walls. Cinderblock walls are acceptable to SCIMR since they contain irregularities on the order of the wavelength of the ultrasound, about 1/8 of an inch. Corrugated cardboard pasted onto the walls with the outside layer removed should provide enough scattering to be detectable by the sonar, but this has not been attempted.

One way of dealing with this problem is to attempt to make as many measurements as possible at right angles to the walls. With a single sonar unit, it is impossible to make all measurements at right angles; how to do so with more than one sonar and some associated problems will be discussed later.

4.3 Stepper Timing

Figure 3 shows the standard sonar wall following scan pattern. Below is a breakdown of the time spent in each function during a normal wall following scan:

Between	Motion time	Waiting time	Data time
5 & 1	190	0	50
1 & 2	250	0	50
2 & 3	500	0	50
3 & 4	60	30	50
4 & 5	120	0	50
Total	1120	30	250

(all times in milliseconds)

From this analysis, it can be seen that the stepper motor time is the limiting factor in scan pattern time. This data is based on a 5 msec/step rate of motion, currently the highest speed available from the sonar stepper. If the sonar step rate was increased to 1 step/3 msec, the delay time would increase to 72 msec, but the motion time would decrease to 672 msec. The total time would be 1.0 sec.

The traditional method of improving stepper response is to increase the supply voltage and the dropping resistance, thereby reducing the L-R time constant. Switching DC-DC power converters could be used to supply voltages higher than the 12V battery voltage, but this option is not currently available.

A stepping motor with a larger step size would solve the speed problem very nicely. Unfortunately, at this point, no suitable motor is available. The present motor has 200 steps per revolution, and draws 0.8 A per phase at 8

volts. A good candidate stepsize is 15 degrees per step. Motors are apparently available of this type, but I have not seen one with the same low current characteristics. Since only a transducer is being spun, the actual torque requirement is slight. Given an infinitely fast stepper, the sonar would limit the minimum scan time to 0.7 seconds.

4.4 Acoustic Morse Beeper

The sonar processor drives an audio beeper, using Morse code modulation of an arbitrary byte. An interrupt level handler performs the actual serialization.

Whenever the sonar program is running at all, the MORSE task will generate a short beep every 5 seconds or so. When power runs low, the short beep becomes a long beep. Aside from communicating the power status, the periodic beep assures the operator that the sonar processor is booted and running.

Other tasks may send data to a semaphore in the sonar processor, which will be beeped to the operator in Morse (short or long) code. The beep rate is slow enough for human comprehension. The beeper has been invaluable in understanding what SCIMR is thinking as he drives down hallways and intersections.

5. Locomotion

5.1 Motor Control Methods

SCIMR is powered by two DC drive motors. The characteristics of the motors determine the control system used. We will discuss first the characteristics of just a single motor.

Each motor has an integral gear reducer: we will consider performance at the output of the motor-reducer train, and refer to it as that of just the motor.

The motor has a maximum speed of 120 rpm, or 2 rps, coupled with a wheel diameter of 4.5 ft, corresponding to a linear speed of 2.35 fps, or about 1.5 mph. The maximum current drawn, during stall conditions, is about 6 amps.

Speed control of a DC motor is achieved by varying the average current through it. Three possible control methods are: 1) apply a variable voltage, 2) vary the width of a pulse at a fixed frequency, or 3) vary the frequency of a fixed width pulse. A reversing relay selects motor direction.

The first method, applying a variable voltage input, is inherently difficult to implement by a digital system. It requires a digital to analog converter, high current linear amplifier, and as many control bits as the D/A is wide. This is the most expensive, in terms of hardware, of any of the three control methods.

Control methods two and three both use a single output bit from the controlling computer, and a single (Darlington)

transistor switch* This hardware is inherently very simple and suffers from no arift$_f$ temperature coefficients or offsets, unlike analog circuits* It is strictly digital* The modulation method is embedded in the driving software* Timing is generated using a crystal referenced 400 Hz clock* This clock causes an internal interrupt, at which point the modulation method processor is run* Accordingly! no times shorter the 2*5 msec can be measured*

Method 2$_f$ known as Pulse Width modulation (PWM)$_f$ or outy cycle modulation, was implemented first, and its performance evaluated* The modulation frequency was chosen as 25 *H2*$_f$ allowing 17 possible pulse widths* each corresponding to a motor speed* A representative graph of motor speed versus pulse width is shown in Figure 1* These speeds were obtained by driving the robot in circles, and measuring the time required* The motor has very poor response to short width pulses, consequently! there are few pulse widths in the vicinity of 1 fps, where the robot is operated* Post widths correspond to frequencies above 1*5 fps, where the motor is saturated* These speeds *are* too fast$_f$ given the limited sonar input rate* The poor low speed response should be no surprise: the motor windings form an L-tf filter* The Pw* technique was rejected due to the snail number of attainable speeds in the desired range*

The thiro control method$_f$ Pulse frequency Modulation <PF$> gives better characteristics* The low speed characteristics of the motors are improved in PFM since

pulses are all the same width and may be long enough to guarantee a response by the motors. A speed curve using pulse frequency modulation is shown in Figure 2. As can be seen, it has significantly more potential speeds in the area of interest.

Generation of pulse frequency modulation signals is more sophisticated than in PWM. A simple approach would be to divide the incoming 400 Hz clock frequency by an integer, to generate the pulsing frequency. However, this yields a speed inversely proportional to the input value. This is unacceptable: it makes the generation of speed values fairly complicated.

Frequencies may be generated directly proportional to the input with a more complex method. Suppose we have an accumulator A, which operates modulo m, chosen as 256. Let the input frequency specifier be N. Each input clock, we add N to A (modulo M), and generate an output pulse if a carry occurs: A+N > M. The output frequency F is then:

$$F = N * C / M$$

where C is the input clock rate. This is linear with proportionality constant of 1.56 Hz per input value.

This frequency synthesis method is simple to use and generally applicable. It has already been used in a tactile sensing system to drive multiple stepper motors at proportional frequency rates, using variable M values [Reference 5].

5.2 Steering

The mechanical design and its software consequences of the steering system for SCIMR benefited from the excellent examples of what not to do provided by a previous attempt at robot construction at Penn, by S. Rao. SCIMR's predecessor never got beyond a frame and electric motor drivers, and was completely scrapped, except for the actual motors.

The device attempted to steer by turning a front wheel, and propel itself by two powered rear wheels. It was a disaster. The steering wheel was turned by a DC motor of the same type as the drive motors. A gearing system was supposed to reduce the speed of turning to a manageable level, but the wheel still spun much too fast--- when it spun. The mechanical design of the mounting was such that the gears would pull apart and refuse to mesh. The high speed of the motor would then attempt to grind the teeth off both gears, with some success. There was an attempt to provide angular feedback with a potentiometer, but it had the same design problems. To cap it all off, the drive motors had sufficient power, and the steering wheel insufficient traction, to steer: the steering wheel was pushed sideways rather than steering the contraption.

One solution is to make the steering wheel the drive wheel as well. This is the course apparently taken in the R2-D2 'robot' of Star Wars fame. The one-wheel-does-all approach results in a difficult mechanical design which is unappealing. Operationally, the steering wheel approach may

have the limitation of a fixed rate of turning* unless the motor turning the wheel is quite powerful.

SCIftR uses a second solution to the steering problem: differential steering* The front wheel castors while the back wheels are driven at different speeds to achieve tu rning •

Since steering is accomplished by driving the two motors at different speeds, if the motors run at different speeos when it is not intendedt the robot wilt turn* This implies that the speeds of the two motors oust be identical for the same frequency input. Unfortunately$_t$ the two motors have significantly different characteristics: given the same frequency input to each motor, the robot will turn.

when SCIftR is driving down a hallway *and* a turn is commanded, circumstances *ar^* considerably different than curing equilibrium straight driving. The inertia of the rocot causes the effect of a change in speed to be delayed for several seconds* 8y this time, SCIMR will have crashed in to the wa11•

5*3 Speed Regulation

The problems with steering indicate the need for a fast acting motor speed regulation system*, with absolute speeo reference to insure consistency* This need is met by an opticdl tachometer pickup on each wheel* The tachometer consists of a penlight lightbulb with a built-in lens and a photo transistor* The space between the wheel treads to one side has been painted with silver paint*, whereas the wheels

themselves are black rubber. The phototransistor outputs a signal whenever a tread drives past. It is incapable, however, of deciding the direction of tread movement. This does not pose a problem since the motors drive in one direction almost all the time, and reversal is accomplished only from a stop.

A feedback system is required which accepts tachometer pulses and a set speed, and outputs a motor frequency correction. The system predicts the number of tachometer pulses which should occur in a given interval, and compares it to the number which actually occur. At the end of the interval the motor speed is increased or decreased by a constant amount as the predicted or actual motor tachometer is faster. The update interval is 10 msec, which means that only a fraction of the predicted or actual speeds will contain a pulse. During speed changes, the output frequency is changed rapidly to a new value.

5.4 Motor Commands

A motor command processor interprets single byte (plus arguments) commands given to the motor control machine. The commands are: stop, turn left, go straight, turn right, reverse, wall follow left, wall follow right, and go slowly.

The stop command will physically stop the machine as rapidly as possible. Not only does stop set the motor velocity to zero, but the motor output pulse frequency to zero as well.

The turn left, go straight, turn right, and reverse

commands fmplement the four different ninety degree turns. Each acts for a specified time, after which the machine is stepped* and a semaphore set in the control processor* The co»»anas drive one wheel forward and one backwards, pivoting SCI^R about the center of *an imaginary rear* axte.

The wall left and right commands cause the wall follower to be *run on* input data present in the A, B_f and C semaphores, *ana* directly set the motor velocity.

The 90 slow command causes both motors to be set at 0.5 feet per secona# It is usea immediately upon exit from a junction to re-establish wall following.

5.5 Stepping Motors

It may seem that stepping motors would be advantageous for the drive wheels as well as the sonar. The tachometers redigitize the analog characteristics of the DC motors, essentially making them similar to steppers, why not go all the »ay? The motors used in SCIMR have the advantage of being prepackaged with alt gearing. A stepper motor would need semi-custom gearing to supply as much torque as a DC motor. The gearing would reduce the available output shaft speed fceyono the useable point. For example, the sonar stepping motor can revolve at 1 rps. This is already half the fully geared shaft speed of the OC motor. Sufficient gearing to supply the torque needed would probably put SCIMR neck and neck in speed with the Stanford turtle. Perhaps it is possible to build stepping motor systems which will supply enough torque and speed from a standard battery, but

such a system would be time and money consuming to construct.

In fact, even given a working stepper, the inertia of the machine might be great enough to overpower the stepper holding torque, and necessitate the tachometers once again.

A possible redeeming feature to stepping motors is the absence of commutators and the noise they create. Doubtless an appropriate AC motor can be found without commutators, such a motor is speed regulated by reversing voltage polarity, and is at least partially synchronous.

To conclude, stepping motors are not worth the trouble.

6. The Wall Follower

A specialized software system maintains robot orientation and position with respect to a wall while driving down hallways. A precondition of correct operation is that a fairly straight continuous wall exist on a specified side of the robot. Violations of this expectation will be flagged as possible junctions.

6.1 What Will Not Work

It may seem that a single distance measurement perpendicular to the direction of motion of SCIMR will suffice to allow him to remain at an equilibrium distance from the wall. This is not the case.

The problems stem from fact that the robot is not necessarily perpendicular to the wall. When it is not, the distance measurement is incorrect. Furthermore, each possible distance value has two interpretations. Consider two robots, each 1 foot from the wall. The front of one robot points inward towards the wall at 45 degrees, the other points outward from the wall at the same angle. Both robots make a distance measurement at 90 degrees left, where zero is straight ahead, and the wall is to the left. Both will obtain 1.4 feet, yet the first must turn right, and the second left. It is not possible to resolve this conflict in general without additional information. One possibility might be to use several successive readings, however, SCIMR would probably crash first.

SCIMR solves this problem by making three measurements

off to the side with the wall; the wall follower program has the responsibility of interpreting these readings and deciding what to do.

The wall follower follows a single wall because the second wall adds no additional information, and will probably only confuse matters. The noise content of the measurements due to doorways, random widgets on the wall, and so on is effectively doubled when using two walls.

Simple schemes such as turn left if we are too far and facing outward or too close and facing sharply outward, and turn right if too close and facing inward or too far and facing sharply inward, do not work. The original attempt at wall following demonstrated that the update rate from the sonar is too slow to support stepwise feedback systems. Despite serious attempts to make it work, the robot would crash within ten feet of the starting position.

The realization of the unworkability of this approach forced the development of the current wall follower.

6.2 The Algorithm

The wall follower is a linear feedback system. The sensing device is the sonar, and the output device, the motors. The feedback path is closed by the designated wall being followed. The wall follower was developed using computer simulation to verify assumptions and simplifications made in the algorithm, the simulations will be discussed after the algorithm itself.

As the robot drives down a hallway, it must

simultaneously attempt to maintain both its position (distance front* the wail) and its orientation with respect to the «all* The setpoint for distance is an input to the wall follower* At this distance* it should have an angle of 2ero <*ith respect to the wall: it should be parallel* If$_9$ in fact* the robot is not parallel to the wait or at the correct distance from it$_f$ SCIMR must execute a turn of some magnitude to bring both parameters to their desired values* However* it should be observed that the two concerns *are* at odds: if the robot is perfectly parallel to the wall, yet only one foct away, it must turn to get to the correct distancet thus destroying the parallelism* Any wall following algorithm must deal with this tradeoff*

SCI*R~s wall follower posits that for every distance fro* the *a11* there exists a correct angle to the wall, chosen by multiplying the error in distance times a constant* Fro* any initial position, this algorithm yields a curve resembling a hyperbola with an asymptote at the zero error position, goiny parallel to the wall*

The computer calculates the amount of turn to make:

Turn = - KDF * CD • DSET) - AN6

where Turn is the turn in degrees$_v$ KDF is the proportionality constant in degrees/foot, D is the actual distance from the wall in feet, DSET is the desired distance in feet, *anti* AftG is the current angle with respect to the wall in degrees* The angle Turn gives the turn to make if we could make it instantaneously, which we cannot* However,

correcting the turn for the motion of the robot in general and the nonconstant and unknown turning rate is not possible. The SCIM* watt follower simply ignores- this discrepancy altogether. The primary justification for being able to do this is that the wall follower works. A corrector wouto probably result in quicker convergence to the oesired position, but would be expensive. The choice of the KDF parameter is the designer's way of telling SCIMR the relative importance of maintaining angle accuracy versus posit ionat accuracy .

Given the basic feedback equation, two fundamental questions are apparent: 1) how do we mathematically *perform* the calculations, and *I)* how do we obtain the inputs.

6.3 Mathematics

The H68Q8 computers used in the robot do not have a multiply instruction, floating point package, or any other such amenities. Consequently, it is far from having any trigonometric functions* Signed and unsigned S bit fixed point integer data representations are used extensively and exclusively.

A very important consequence of the data representation is that all calculations must be guaranteed to prcauce a correct result: failure due to overflows and precision problems **must** be detected and corrected for. Unlike the course of action of most high level languages, overflows must be detected, rather than "wrapped around? and the *maximum* possible value returned. The design of the

number representations must be such as to minimize the possibilities of overflows occurring while maintaining the maximum precision avaiable.

Meeting these requirements required some explicit assumptions about the conditions of operation of the wall follower; violation of these assumptions constitutes exit from the domain of correct operation of the wall follower: it may, and probably will, fail, justifiably. The primary operating restriction is that the robot be already approximately aligned with the wall, where approximately is defined here as within 45 degrees of parallel. As we will see later, the junction analysis tasks require an even tighter precondition.

Angles are internally represented as a two's complement binary fraction: -1*b7 + (b6b5b4b3b2b1b0)/128, where the maximum possible value, 7F(16), corresponds to 45 degrees (-45/128 degrees), and the minimum value, 80(16), corresponds to -45 degrees. Any angle not within this range is clipped to -45 or +45 degrees(+/- 45/128). When the wall follower's preconditions are violated, by being very far from the setpoint or at a sharp angle, the robot will 'do its best' to get back on the right track.

6.4 Obtaining Inputs

Obtaining inputs to the feedback equation involves some not insubstantial work. The two inputs required are the (perpendicular) distance to the wall, and the angle to the wall. The sonar and stepper measure the distance to the

nearest object in each of five different directions* Three of these *ar^* towards the wall being followed* One measurement is taken perpendicular to the* rooot, and hopefully, the wall* Two other measurements are made at a fixed angle forward ana aft of this measurement (see Figure 3) • The forward, aft_f and central measurments are named A_f B_f anc C* From these we Bust compute the angle and distance to the walI*

6*4*1 Distance to the Wall

Suppose first that we knew the angle to the wall* We couU compute exactly:

DIST * A * COS4PHI 4 TNA)

where PHI 1s the angle of either A or B from C_t and TNA is the <*ngle to the wall* We could write a similar expression for 8 or C as well* As discussed earlier, COS and even * are unoesireab le for implementation. Since operation must be assured only for angles within 4/- 45 degrees of the wait, »e may make the approximation that the distance to the wall is the minimum of A, B, and C* Computer verification shews that this approximation is accurate to better than 1QZ over the input domain* This error is quite acceptable, especially when one considers that the error is less than 1*52 for ançles within 3Q degrees of nominal* For normal "locked on# conditions$_t$ the quantization error of the sonar is approximately 4%• The original equation has been greatly simplified with no loss of real accuracy*

6.4.2 Angle to the Wall

Computation of the angle to the wall is more challenging. The correct equation is:

TNA = arctan(1/tan(PHI)*(A-B)/(A+B))

where everything is as defined previously. Two things are immediately apparent: 1) the arctan must go, and 2) 1/tan(PHI) is a precomputable constant, since PHI is a constant.

The most marvelous thing to do to the arctan is to simply cross it out. In fact, this is what SCIMR does. The Taylor series expansion of arctan x is:

arctan x = x - x**3/3 + x**5/5 - ...

(1 > x > -1) [Reference 4]. Since the constraint corresponds to 57 degrees, and the error term is cubic, we can do very well this way. This approximation is accurate to within 10% at 30 degrees, and 30% at 45 degrees. Although this is considerably worse than the distance approximation, it is acceptable. Within 15 degrees of parallel, an estimate of the operating range, the approximation is accurate within 2.5%. Note that we avoid units adjustments conveniently due to the choice of units for angles.

An experimentally derived heuristic measure:

ATN = (2/(1+TNA*TNA/2))/3 * TNA

yields an accuracy within 1% to 45 degrees, and 2.5% to 70 degrees. However, this measure has not been implemented due to its greater complexity in the face of probably

insignificant improvement in actual performance*

The computation of the angle to the wall using the simplified expression,

TNA = FT * (A-B>/(A+B>

requires careful maintenance of binary point position^ Details of the scaling may be found in Table 3*

6*5 Anyte to Frequency Correspondence

Correspondence between angles to be turned and a change in net motor frequency is determined by spreading thi turn ever the entire length of time until the next sonar scan pattern is complete, yielding an angular velocity. Th< angular velocity is converted to a linear velocity using th< wheelbase of the robot$_t$ and then to wheel revolutions ^er second with the wheel radius. The revolutions per second i« converted tc treads per second (48 treads/revolution), and then to the actual fcOTGR tachometer frequency measurement units•

Luckily, all of the calculations are static and multiplicative, so that the entire process can be reduced t< multiplying by a single constant, named KRL.

6.6 Ignoring Doors

DoorSf windows, water fountains, and other ultrasonic reflectors cause heartourn to the wall follower without compensation* The wall follower attempts to detect when a reflection is occurring and take corrective action to minimize its effect$_t$ much tike the 1nterprocessor

communication system.

A reflection is defined by either the A or B distances being larger than 1.5 times the C distance, which is assumed to be immune from bounce, due to the wall follower preconditions. When a bounce is detected in either the A or B directions, the distance measurement is set to the value of the C measurement. The ensuing angle calculations will be incorrect, but much less than if the correction is not performed. The intent is not to be absolutely correct, but to maintain operation within the wall follower envelope.

The 1.5 factor restrains the viable angle range of the wall follower to about 26 degrees before valid readings will be mistaken for invalid ones. Also, an indentation of more than 1 foot will be detected as a bounce. Certain door-wells in the Moore School exhibit this phenomenon.

In the event that bounce occurs on both radial measurements, the angle will always be zero, and the distance alone determines the wall follower activity. This occurs in hallways with totally reflecting walls. The wall follower makes an attempt to maintain its sanity, but generally will wander slowly into a wall.

6.7 Performance Analysis

Performance of the algorithm was analyzed in three ways: 1) static analyis for accuracy of overall process, 2) computer simulation of SCIMR travelling down a hall, and 3) live 'eyeball' analysis (strictly qualitative).

The static analysis excerpted in Table 4 lists input A

and B numbers, and shows corresponding distance, desired angle, perceived angle, turn angle and motor frequency change numbers, as calculated using absolutely correct arithmetic expressions, the approximations above implemented using real arithmetic, and the approximations computed using the fixed point integer representations. The latter numbers are also displayed in hexadecimal; the live machine program was verified against these numbers. Note that the calculations shown do not use the C value in computing distance, so the nearly parallel results can be expected to be slightly better than shown. Good agreement between the actual and optimal numbers is obtained when clipping does not occur.

The dynamic computer simulation requires the specification of a variety of parameters. The 'auto-mix' constant simulates the effect of the inertia of the robot. The value 3.0 shown means that the turn executed by the robot is 75% new turn, and 25% old turn. Other runs demonstrated convergence despite very poor auto-mix values. The stabilization distance is the distance setpoint. The value shown has been corrected by 1/COS(PHI) from 2.5 feet. If the program had used the C value, this correction would not be needed. Velocity shown is that of the robot. Time per iteration is the time between sonar scans, at which time new data is available to compute course corrections. Sample output is shown in Table 5.

The dynamic simulation shows rapid compensation for

quite poor initial conditions. The robot is shown to lock on within several feet of the initial location.

6.3 Error Sources

SCIMR behaves largely as predicted. It tends to be somewhat underdamped. The largest perturber, aside from intersections, are doors under mirror conditions. Before this situation was understood, doors would cause large scale twisting and turning. However, it would die out within three to four feet of passing the door. Some small oscillations still occur from doors.

Less than optimal wall following performance is due to several causes. Some of them have been discussed previously: the distance approximation, the angle approximation, and the finite number representation, not considering the motion of the robot in determining the turn angle, and systematic underestimation of the angle to the wall because of the spread beamwidth of the sonar. This effect causes the effective PHI to be less than the actual angle of the sonar transducer during the measurement. Precise correction must await the development of an experimental or mathematical sonar detection model.

Another error source is the time interval between the two angle distance measurements. During motion away from a wall, the B radial appears shorter than its actual value, at the time of processing which is the completion of distance reading A. This will cause the angle to the wall to be overstated, and a tendency to steer more back towards

the wall. The time interval is about 0.150 seconds. Even at cruise speed away from a wall, this corresponds only to 0.18 feet. SCIMR cannot do this while moving straight, since the measurement is not parallel to the direction of motion, but at a large angle to it.

During turning the apparent change in distance to a wall can be quite large if either measurement angle is nearly parallel to the wall. Due to the beamwidth of the sonar, and the wall follower preconditions, this error is probably within a unit or two of sonar resolution. Note that if the sonar angle measurements had been placed at a different position within the sonar scan cycle, the results could be quite different. Two terrible things to do would be to have both measurements made at the beginning of the cycle, or one at the beginning and the other at the end. Either would cause failure of the system.

The motor system is far from perfect too. The most significant problem is the acceleration/deceleration rate versus the inertia of the robot. When a motor command is given, the motors must accelerate or decelerate to the desired speed. Consequently, the effect of changes in velocity is diminished. A quantitative analysis of this effect is not available.

7. Intersections

7.1 Current Junction Analysis

Driving down hallways is relatively well defined. Hallways are characterized primarily by a width. By contrast, intersections can be much more complicated. Trying to understand an intersection using a sonar presents severe difficulties. Once again, the sonar characteristics prove a burden.

The intersection detection system utilizes three distance measurements; one forward, and one 90 degrees off to each side. It must return to its superiors a bit array in which each bit is on if there is a hall in that direction, and off if there is not. This appears a superficial transformation, but life is complicated. Obviously, each measurement must be compared with a threshold, and the respective hallway bit turned on if the measurement is longer. The hard part is trying to come up with the appropriate thresholds. We may actually dispense with the forward direction simply, by making its threshold a constant. SCIMR decides to find somewhere else to go when the distance remaining in front of him is less than four feet. This works quite well and is straight-forward, so enough said.

The simple approach of the forward radial will not do for the side radials. SCIMR must operate in hallways of considerably disparate widths, from about four to over twelve feet in width. When SCIMR looks down a narrow

hallway* the distance is less than any reasonable threshold which will avoid 'false detects' in the wider hallway* Consequently, the threshold strategy must be adaptive*

The basic solution is to make the threshold on each side just a bit longer than the width of the hall in that direction. Just a bit longer is 2*5 feet in the current incarnation* Since SCIMR does not drive down the center of hallways, but at a fixed distance from one of the walls* the left and right thresholds are different, and one of the two is fixed at the follow distance plus 2*5 feet* The threshold to the other side* the wall which is not being followed* oust be set dynamically* This is done when SCIKR first enters a hallway* The hallway may narrow or the initial width reading may be taken too soon, yielding the distance SCIMR can see down some other hallway* SCIMR continuously updates his width estimate as he travels down a hallway* Each new reading is averaged with the old width* as long as the new width is not more than 2*5 feet larger than the old width* in which case a hallway would be flagged in that direction and no update performed*

7*2 Problems in Junction Analysis

The junction analysis procedure is simple* and seems to work welt normally* When does it not work? There is one circumstance where this junction analysis fails terribly* when SCIMR crives from a narrow hallway into a much wider one (at least five feet wider) it looks to him as if there are suddenly two hallways off to either side* To a certain

extent this is correct: SCIttR could drive a reasonable distance in either direction before having to stop. However, the wider hallway should really be identified as such, and no turn attempted* An even harsher problem is driving back the other way from wide to narrow, if the narrower hall is not along the wall which SCItfR is following* In this case, when he gets to the transition, the narrow hal Iway will be completely invisible* *Hot* only that, the junction analyzer will be describing a completely dead end, since the junction analyzer will have adapted to the wide hallway* This problem is not the fault of the junction analyzer, but indicates a basic insufficiency of the system in dealing with areas, as contrasted to hallways*

The junction finder has another problem even in normal circumstances* The architects of the Moore School, in their infinite wisdoa* placed fire doors on at least one hallway leading into each intersection* These doors are displaced several inches away from the wall and are highly reflective to sonar beams* When SCIMR drives down a hallway into an intersection, the doorways cause him to turn away from the wall (now acor) he is navigating on, to try to maintain his equilibrium position* This occurs about two to three feet before the intersection, so that SCIWR enters the intersection at an angle* When he makes the distance measurements to determine the junction type, they *are* not alony the axes of the hallways, and SCIWR sees the walls of the halls* rather than infinity. The junction analyzer

offset distance, 2.5 feet, is so short that even under these circumstances, there is a reasonable probability of the junction analyzer obtaining the correct results, which it usually does. This is not the end of the story, because SCIMR must then make a turn, with the desired result to be parallel to the axis of the hallway to be driven down. However, SCIMR always turns 90 degrees, so that the initial error angle is maintained.

SCIMR looks in each direction only every 1.4 seconds. During this time, he is travelling at 1 fps. This causes two complications. First, narrow entrances such as doorways are not always detected, although they may be sometimes. Open doors seem to be detected about one half of the time. Second, SCIMR has travelled varying distances into junctions when they are detected. After a turn, SCIMR may be either close to or far away from the wall which is to be followed. Consequently, large scale perturbations in SCIMR's path immediately after a junction are observed.

7.3 More Advanced Junction Analysis

The purpose of the preceding discussion was to point out the problems associated with trying to decipher junctions. It should be clear that more sophisticated methods are necessary for good results. The basic junction analyzer is satisfactory in deciding when the environment has changed, but not really good at figuring out exactly what has happened. The normal junction analyzer has time constraints: it must not slow down the sonar input rate

adversly affecting the wall follower, and it must scan often enough to detect hallways going by 'on the fly'. For this reason, the junction analyzer was limited to the three 90 degree distance measurements. Once we have decided that a change in junction type has in fact occurred, we no longer have this restriction: we can stop SCIMR and perform as many readings as desired. Optimally, we would like to decipher both the actual junction type and the orientation of SCIMR within that junction so that if a turn must be made, it can be done so that SCIMR winds up headed in the correct direction.

Since the beamwidth of the sonar is about 20 degrees, we could perhaps perform this analysis by making a full sweep of the area, every 20 degrees, from -130 to +130 degrees relative straight ahead. By detecting local maxima, we could determine the actual hallways. The old problem of ultrasonic mirrors returns to haunt us. A bounce off a door would look like a fine hallway. No sonar reading is guaranteed unless it is made precisely perpendicular to the surface to which the measurement is being made. Unfortunately, SCIMR has no way of knowing when this condition has been satisfied! Suppose that SCIMR finds a good long measurement. He needs to know if it is a real or anomalous reading. SCIMR needs to measure the angle between himself and the object being measured. Even if the virtual image happened to be a wall, the door would be too narrow to do the normal angle measurements. A door and a hallway

might be quite indistinguishable* From this I conclude that the sonar is largely insufficient to deal with environments containing ultrasonic mirrors. Later we wilt consider some possible cures to this problem using other devices*

I might point out the human image processing system has much the same problem* Any well mirrored room will illustrate this* Try looking at a mirror through a paper "telescope* such that you cannot see the edges of the mirror. It is quite impossible to decide the distance to the ©trror*

*!•*** Turning: The Low Level Navigator

The navigator is responsible for executing turns *and* driving until *&n* intersection is detected* Its input is a new cirection to drive, which is converted into a relative turn; left* straight, right, or reverse* The initial condition is that SCIftR is in *an* intersection* The navigator turns $CI*R in the given direction and drives resets the junction analyzer to require a very short distance to the wall to be detected before the opening is declared to have "gone away? SCIMR then drives forward until the desired wall to follow is in range, as attested to by the junction analyzer* When in range, the junction analyzer is instructed to reset itself for the width of the hallway, ana then normal wall following operation will commence*

Whenever a deviation from a straight and narrow hallway is detected, the navigator brings SCIftR to a halt ana performs an "official" junction analysis scan* The

original junction type is thrown away, since it may be completely spurious, or it might be incomplete if the robot is driving into a T-junction. The official junction scan ensures that both arms will be detected. If the junction is determined not to exist, the wall following mode is re-entered.

There is also special processing code for detecting when a crash is imminent, in conjunction with the sonar processor, and stopping the robot until the condition is removed.

S* tapping the World

Any creature, biologic or cybernetic, from man to mice, that makes any claim to intelligence must be capable of remembering enough about the world around it to prevent it from being eaten or having its funding suspended*

The Stanford cart has a world model based on circular areas an the floor which cannot be driven into* It is designed to operate in reasonably cluttered environments* It has no facility for representing walls; perhaps they can be dealt with by enough circles* By contrast! SCIMR can operate solely with halls *and* intersections* Obstacles in the way, moving objects or especially waits, curving halls, 'wye' junctions! large open areas, ramps between floors, deadly stairs, sneaky elevator rides, hallways with doors which open *and* shut, are all baffling to SCIMR* The last two ottuscators could be handled by SCIMR with appropriate soft.are, but this has yet to be developed, as it is less th an critical•

8.1 i*ther Possible Naps *

The data format in SCIMR represents the world in terms of hallways and junctions* We might imagine a representation based on a bit map, where each bit corresponds to a location in space* A bit is on if the location is occupied by something, and is off if the location is not occupied by anything* This data representation is hard to generate, memory inefficient, and slow to proces s*

SCIKR does not represent areas and obstactes since his sonar is toe coarse to suppty reliable and useful information in cluttered environments* There might be some hope for dealing with targe empty rooms, but they have not been considered here*

8*2 ipacial Position and Orientation Sense

As an aid to his map making tasks$_f$ SCIMR has a highly developed directional and positional sense, which is intended as partial recompense to not being able to unictuely identify locations by locally observable quantities$_t$ such as signs and landmarks* The location and position are maintained by a task in the MOTOR machine* This task uses events generated every 2*5 feet of travelled distance by the motor feedback system to increment and decrement the appropriate coordinate value* The current XY coordinates of the rctot are continually updated in the main intelligence processor for its use* Additionally, the cumulative direction is maintained as turns are made* SCIMR's location is measured relative his location at birth, that is, when his program is first started up* This locations <Q_fQ) is affectionately named HOME* Directions are designated with respect to North, with East, South, and West assuming their normal positions* Since SCIMR has no compass, SCIMR's initial direction is defined as North by fiat*

The net effect of this system is to allow SCIMR to absolutely identify each intersection and the direction of entrance to it* This ability is crucial to deciding: have I

been here before? The X and Y coordinates are quantized in such large values (2.5 feet) for two reasons: it conserves storage, and it reflects the imprecision in cumulative distance measurements. It is also the largest half integer distance convenient to generate in the assembly language software. Given quantization in 2.5 feet units, two bytes can represent a location anywhere within an area over 2 football fields long on a side. Most reasonable applications will require a much smaller area, but for thos that do not, the actual storage size is not crucial to the intellectual development of the system. While SCIMR drives down hallways, he tends to wiggle a bit, and recognize each junction at a somewhat different location. If we were to b too precise about the required accuracy of the locations of junctions, junction locations would be unrepeatable.

Operationally, the locating system seems quite repeatable, although the values may differ by up to five feet from measured values in some random worst cases. In further experimentation, the distance per unit may be reset to one or 1.5 feet to reduce the maximum quantization error and allow more precise accuracy and repeatability measurements.

8.3 Remembering Maps

'Things' in SCIMR's world may be categorized into two types, hallways and intersections. Hallways and intersections combine themselves in interesting ways which can be most conveniently represented as a graph, where the

hallways are the edges and the intersections are the nodes. Storage is allocated for each intersection; the storage for the hallways is implicit in that of the intersections. At present, not even the width of hallways is stored, since there has been no demand for this piece of information in the high level software. Some information, such as the length of hallways, may be reconstructed from the map; this is not currently performed either. If there was substantial information associated with each hallway, such as a width, color, and door count, for example, the storage system would require the duplication of each piece of hallway data. In this case, a system with separate storage for hallways and intersections, where each points into the other would be more efficient.

Above the detail level of the wall follower, all angles are multiples of 90 degrees. Consequently, there are only four angles at which halls may occur, namely, the N, S, E, and W mentioned earlier. The SCIMR junction descriptor contains four slots, one corresponding to each of the directions. Each slot contains a pointer to the node that SCIMR would arrive at if he were to drive in that direction from the present node. The next node pointers directly describe the interrelationships of the hallways with the junctions. It is assumed that the relationship between two juntions is transitive, that is, if there is a hallway from A to B in the direction E, then there is a hallway from B to A in the direction W. If this is not the case, we get the

classic example of a one way street.

A problem presents itself: how do we know which directions correspond to halls$_f$ and which to walls which will yield instant crashes? If we use the venerable NIL*. it is iffipossiDle to know whether that direction is obstructed by a wall, cr whether it simply has not yet been driven down to fine out what is there* Additionally, we would like to be able to flag certain directions as dangerous: those which lead to stairwells, halls of solid mirror concrete, which cannot be used for navigation by SCIMR, or those which contain violent robophobiacs*. such as exam takers* NIL must yield to three new "special purpose' nodes: DARK$_t$ DANGERf and CSASH*. whose meaning should be self-evident* with this scheme, the encoding of the additional information requires no additional real storage within each node% although it does remove the storage for three whole nodes from use* I am mere concerned with the former than the latter*

Each rode descriptor also contains its location in absolute Xt coordinates*. as obtained by the positional sense described above* A junction also contains a special purpose word, used during path finding to store mark bits* In the future it may oe used for other things as well*

Finally*. the node contains a seven bit node name*. essentially a single character* The node name is used to communicate with the instructor* SCIMR has three methods of identifying a node: the location* the node name, and the node nuaoer* This is a heavy degree of reaundancy* but each

method of naming is used for a specific purpose. When a path is being described to SCIMR, neither the exact location of a junction, or the node number SCIMR will assign to a node, is known to the instructor, at least without a bit of precognition. The node name allows reference to nodes in a convenient and efficient manner. Imagine trying to identify City Hall as 40 deg 20 min 10 sec north, 70 deg 50 min 30 sec west. Where? (Don't ask me!) Admittedly, the node name is so short as to be fairly useless as an English descriptor, yet it is fundamentally required and useful.

SCIMR assumes that the total number of junctions in his world will be less than 32, including the predefined ones. A look at the Moore School map (Figure 4) will show the validity of this, as will most other reasonable environments. A counter argument: maybe the complexity of the world is the reason that someone wants a robot in the first place! As a benefit of this restriction, node numbers can be stored in a single byte. The total storage per node is thus: four bytes of next node pointers, 2 bytes of XY location, 1 byte of marks, and 1 byte of node name, for a grand total of the magical number 8. A next node pointer occupies only 5 bits per byte, what happens to the other three bits? One of them tells SCIMR whether to drive down the hall following on the left or right wall. This can be different or the same from the other end of the hall, depending on the circumstances. The other two bits are spare. All three can be lined up to the right of the node

number$_f$ so that the node number in its left justified position, is multiplied by eight* Since the node size is eight bytes, we achieve a premultiptication which simplifies the array manipulation requirements in the software*

The total SCIttR data base requirement is 256 bytes* The contrast between SCIftR and most typical A«I« crunchers using large virtual memory LISP dynamically allocated storage systems should be crystal clear*

8*4 Generating Paps

Before a *map* can be used, it must be generated* SCIMR generates maps continuously! under control of a specialized task* The trapping, or cartographer, task, as it is called, monitors *data* being transferred between the high level path task, and the navigator*

when a new direction of travel and junction type *are* returned by the navigator, the junction location may be founa in the current X$_f$Y locations maintained in the control processor by the motor control machine* The cartographer looks at the next node in the current direction from the (previous) node, to see if it is DARK*

If the next noae is not DARK, SCIMR has driven in this direction before from the node* The current location should be trie location of the next node* If this is not the case, the cartographer will signal an error* Essentially, the machine is tost, since the world is not conforming to SCIWR's expectations* If the robot is in the correct location, according to its map, the current position and

direction are updated.

If the cartographer discovers that the next link is DARK, and that ft has just been driven down, the DARK must be relieved, and replaced with the destination* A scan is maae to see if SCIMR has ever been at this X_fY location $before_f$ by looking through the locations of all of the nodes* If SCIMR has ever been to the location before, a closed circuit in the map has been detected, and it is closed, by making each node point at the other in the proper directions* If the pointer from the other node is not DARK, the sap is in error, and *an* error is signalled*

If the machine has *neyer* been to this X,Y location before, a new node is created* The node is set up with no name, and X,Y location given by the current machine coordinates* The next node pointers are initialized to either DARK or CRASH using the new junction type* The *reverse* direction is initialized to point back from whence it case* and that pointer set to point at the new nooe*

The map building process is relatively $straightforward_f$ with the exception of new node next node pointer initialization* When an already known junction is driven into, the junction type is not checked against that stored in the map; the conversion of next node pointers in absolute directions to bit strings relative an incoming robot cirection, and vice versa, is painful to implement* Resetting the current robot location to that found when driving into a node at a known location would hold down the

buildup of cumulative position errors. Throughout the process above, the wall to follow is stored in the next node pointers, it was not discussed since it simply confuses the presentation.

8.5 Using Maps

8.5.1 The Path Interpreter

SCIMR must be told what he is to do, by giving him a path to follow through a terminal. During the execution of the path, the map is generated, and may be used in specifying the path as nodes are added. Once an entire map has been created, SCIMR can be sent to any position in it, just by naming the desired destination.

A path is specified by a string of single character node names. Since there is not always a complete map of the destination area in memory, provision must be made for specifying path sections in absolute terms. This is done with the aid of special purpose reserved node names: N, E, S, W, n, e, s, w, D, '=', and '.' The first eight letters specify directions of travel. Lower case letters indicate wall following on the left, and upper case letters indicate wall following on the right (for these predefined directions).

Why not use turn names L, S, R, l, s, r for the same purpose? They would eliminate conversion of absolute directions to relative turns in the navigator. However, when SCIMR finds his own way to a destination, you never known for sure ahead of time which direction he will enter

the node. Accordingly, a specification of a relative turn would be ambiguous. Rather than have both relative and absolute direction specifications, I have implemented just the absolute turns for simplicity, despite the fact that human directions typically use both absolute (North on 95...) and relative (turn left at the light) specifications. The conversion of relative to absolute is straightforward and not of fundamental concern.

The D character is followed by one of N, S, E, or W, and causes that direction from the current node to be marked as DANGERous. The command is in error if the next node in that direction is other than DARK.

The special character '=' tells the path processor to store the following character as the node name of the node at which SCIMR currently resides. An error occurs if the node has a name other than '?' which indicates no name. Duplicate node names are flagged as errors. Special characters may name nodes, but will not be reachable.

SCIMR will stop when the special end of path character '.' is encountered.

Any character aside from the predefined ones will cause a seek to the node with a name equal to that character. If no such name is found, SCIMR signals an error and awaits further directions.

SCIMR processes the path specification slowly, a character at a time, executing all previous steps before any new ones. Paths cannot be resolved in general before

execution, since the path itself may elaborate the map, pointing out short cuts* Also* some node names may be defined curing the course of execution of the path which are later used by it*

3*5*£ Finding minimum Length Paths

The path finaing algorithm is interesting enough *in* its own right to merit a discussion* The problem is to find a series of directions to travel in (n, s, e* w) which will take SCIMR from h*fs current location to a given node, specified by name* The path length is to be minimized to cut ariving tine* To simplify the calculation, SCIMR measures path length in terms of the number of nodes he must drive through, rather than the path length in feet*

Since SCIWR operates in a "taxi-cab" geometry [Reference 23, any path between two points which remains insiae a rectangle whose diagonally opposed corners are the two points, has the sane length, given by the sum of the difference in X and Y coordinates* Most buildings have minimum paths with this property* In such a case, the length of time required for SCINR to get from one point to the other is limited by the number of intersections to be dealt withf since they take the robot nowhere? but reauire several seconds of time* If path length in feet was minimized, most paths would appear equivalent* and the robot might pick a path with more junctions at random* minimization of the number of intersections as a path length also minimizes the prooabitity of an error occurring in the

junction analysis.

The mark byte of the node structure is used to store information for path finding. Each mark byte contains either: end (0), invalid (80), an old direction (010000dd), or a new direction (negative of an old direction).

Initially, the mark of each node is set to invalid. The node name is looked up, and the mark byte of that node set to end. If the node name is not found, execution stops.

The path finder will iterate as long as the mark byte of the current location is invalid. Each iteration, the next node pointer of each node with an invalid mark byte is examined. If the next node is marked with an old direction, the mark byte of the node under test is set to a new direction corresponding to the direction of travel from the current node to the next node. For example, if old(mark(current.N@)) then setnewmark(current,N). Note that end satisfies the old predicate. After each node has been scanned, all new directions are changed to old directions, and the algorithm reiterates.

The algorithm expands a set of known minimum paths outward from the desired location until it reaches the current location. It is based on, but simpler than, the algorithm in Reference 1. Once the algorithm has terminated, the correct minimum path may be unwound by starting at the current location, and driving in the direction specified by the mark byte. The current location is updated, and SCIMR drives in the direction specified by

the ®ark byte of the new current tocation$_f$ and so on until the end mark is reached, which terminates the minimum path finder totally, ana causes resumption of pathway interpretation #

9. Performance Analysis

9.1 An Example Run

For testing purposes, the following two part run was conducted:

N=AN=BW=CS=FEH.

BN=GCDWN=IH.

The command language is not exactly transparent. Reference to Figure 4, the Moore School map, or the path description language writeup, may be helpful. The first part causes SCIMR to circumnavigate the hallways, and return to his starting point, where his internal map was dumped, and the second path piece started. The second path causes SCIMR to revisit some of the nodes and perform additional exploration. A dump of the final map is shown in Table 6.

Four 'crashes' occurred during the total procedure. The first, at node C, resulted from the fire doors at the entrance to node C from B throwing SCIMR off course. SCIMR entered the junction at an angle of perhaps twenty degrees, stopped about a foot into it, turned left, moved forward slightly, and stopped with an imminent crash warning. I twisted SCIMR the twenty degrees so that he could see the hallway in front of him, and he set off down it, completing the rest of the first part of the path successfully.

The second crash was due solely to operator error (me!). I had to relocate SCIMR from the home position to a terminal to see how he had done. Before I moved him, SCIMR was facing south after entering the home node. When I

plunked SCIMR back down, I inadvertently pointed him north out of the cul-de-sac. This was a mistake. SCIMR's first action was to turn 180 degrees around and try to drive into the ladies' room. I turned SCIMR back around in the correct direction before he thought he was done the turn. No harm ensued. Once again, the testee is more reliable than the tester.

The third crash resulted from SCIMR driving too far into a dead end, and the fact that SCIMR pivots around the midpoint between the rear wheels. The front corner of the robot impacted the doors in front of him. This crash was repaired with quick manual intervention.

In the previous three crashes, manual intervention saved the day, without the software really caring. The fourth crash stopped both the machine and the programs. The robot entered node C at an angle again, and had to turn right. However, it was twisted to the left, away from the wall it had to lock onto. Consequently, after the turn, it moved further and further away from the wall it had to lock onto, and lock never occurred. The robot drifted into the north west corner of the intersection. The junction analyzer then decided that the junction type had changed to a straight and right configuration. Since the robot had moved only three or four feet, the cartographer decided that it really had not gone anywhere. It then looked in the map for a link in the north direction from the current location to the current location, which obviously did not exist. The

cartographer decided it was lost, and quit. There is no return from this state aside from complete restart, so this crash ended the run. It is clear that slightly more complicated software could in fact detect this condition and allow a shove to effect recovery.

9.2 Tricks

How might we improve the reliability of the robot? The fire doors cause the most serious problems. The easiest solution would be to simply remove them. It is my belief that this would enable SCIMR to work very well. Late some night, I might experiment with this, but the doors are supposed to stay up.

Another possibility which has been suggested is to slow down the robot to give it enough time to get back on track. I do not like this, since the cart experience is to be avoided, and since the robot does not drive nearly as well at slow speeds. The pulse rate gets so slow that the computer cards are shaken severely, risking damage to them.

The last strategy has been tested and seems effective. The idea is to lengthen the apparent length of the door, and thus enable the robot to get on course by the time he reaches the intersection. An easy way to achieve this effect is to stand flush to the wall SCIMR is following a foot or two in front of the door. This makes the door seem to be about 4 feet long to SCIMR. The same effect may also be achieved by taping strips of corrugated cardboard to the wall at the height of the sonar. The cardboard is bent to

have the same indent as the door. To make the cardboard sonar reflective, one side of the piece is removed* exposing the corrugations, yhich are sufficient to reflect the sonar be am .

10. Different Modes of Operation

The path following mode of operation described earlier is not the only conceivable one. I will describe two other, unimplemented, modes of operation. By mode of operation, I mean that the robot is under control of a substantially different high level strategy program. However, software below the top level will remain unchanged.

10.1 Making Your Own Map

SCIMR is quite capable of generating his own map, with the aid of appropriate control programs. The basic directive is to remove all DARK nodes. When the last is gone, the entire reachable area will have been analyzed. The normal map generating process will have, in the course of the robot's progress, generated a map of this world.

An appropriate algorithm for searching a space is easy to come by. A classic depth first graph search will do the trick. Whenever we reach a dead end, or someplace we have been to before, we can simply drive back along the section of the map already explored to some hallway as of yet not explored.

Two problems may obstruct this task. First, there may be areas hazardous to the robot, such as stairways, or which are too complex and irregular to permit sonar analysis. The path following learning method avoids this problem. In some constrained areas it may be possible to let the robot roam at random, but I tend to doubt this is true in general, as is probably clear from the earlier discussion.

A second problem is that the world may not be closed. If this is the case, the robot could take a long trip. If the vehicle was an extrasolar space craft (shades of Star Trek--- The Movie), such a setup might be desireable.

The point is that sheer learning without direction is useless. Any learning task of this nature must include some direction and bounds to what must be learned, whether physical confines or ethereal directives.

10.2 Getting Unlost

A more interesting problem is trying to rediscover the the robot's current location, given just a map. Such a problem arises in a natural way, if the robot happens to get lost, especially in the case of improper junction analysis. SCIMR can decide when he is lost by the discrepancy between stored junction types and locations with those observed. Once a discrepancy has been detected, a recovery algorithm can conduct a search process to determine the current location, using the stored data base.

The algorithm for 'relocating' attempts to successively refine the location of the robot, starting with the initial assertion that 'we could be anywhere!' An iteration proceeds by driving along until a junction is found. The junction type is then used to cull the set of possible locations of the robot. Each possible turn is then evaluated to determine the net information gain associated with executing that turn, and the turn yielding the most expected information implemented. The information gain from

each turn may be calculated by summing over each possible junction type, the square of the number of potiential robot locations left if this junction type was encountered after taking the turn under consideration. Note that this information content is an approximation; to be precise the operation square(x) should be replaced with x'*log(x'), where x' is the probability of x (obtained with a normalizing factor). The proper turn to select is that with the smallest value of the measure.

In the event of a tie among the turns, the path length can be extended to consider all paths of length two, then three if necessary, and so on. A simpler alternative is to pick a turn at random. The latter choice would probably cause some 'wandering' in the end game, when the number of possibilities is small, and the chances of local indistinguishability larger.

The algorithm terminates when only a single possible location is left. If all of the possibilities disappear in a blaze of glory, the logical thing to do is to go back to the beginning of the entire process. Alternatively, the robot could just give up; it would probably have sufficient justification.

The algorithm must be complicated by the consideration of DARK nodes. It is quite probable that a turn selected for execution will cause a trip into DARK regions. Each possible robot location which yields a trip into netherlands for a given turn is effectively removed from the set of

possibles for that turn* If the turn is selected for execution, that possible tocation is removed from the possible sett and placed on a special stack* As turns are aaaef they are pushed onto the stack so that the return path from the current location to the entrance of each dark node ventured into is available* The return information is valuable when the number of possible locations drops to *zero.* If *the* OARK node return stack is nonempty* the robot say Backtrack up the stack until some location descriptors are found on the stack* These locations *are* then popped and placed into the possible location set, and operation resuaes. If the algorithm has no possible locations at some pointf and an empty DARK stack, then the robot really is off the cieep end*

Cn the other hand* when the algorithm terminates with a nonempty DARK stack the robot may be someplace actually Quite new* The situation becomes very complicated* The (backwards) path on the stack is executable from the newly decided location, and each location on the OARK stack, at least that portion above it* One could therefore backtrack along the entire length of the OARK stack, restoring locations to the possible set as they *are* popped* When the stack is empty, the possible set will contain a number of locationst and the entire culling algorithm repeated* It seems risky to make the statement that this process is guaranteed to stop, so I will not make it, but leave the Question to detailed analysis *and* simulation (not yet

pe ft o[illegible].

As an additional restraint, turns which would cause entrance to a DANGER node from any possible position must not ie considered, on the assumption that any venture into a DANGeS zone might cause device termination* Although it seems extreme, this rule is mandatory to insure robot survival* If there *are* no turns which avoid the possibility of entering a DANGER area, the robot must shut down and await human intervention: it has to ask for directions!

10»3 Deducing Connectivity

The rectangular world SCIKR inhabits suggests certain deductions which might be made on this assumption* A reasonable one is that if two nodes exist in the data base which doth have one coordinate within repeatability tolerance of each other, and which have DARK nodes facing at each other, we might deduce that a hallway joins the two intersections* It would be fallacious to add this deduction to the data base without verification, it can be put in with a flag in one of its lower bits indicating a tentative entr>. Such a system could be integrated into the cartography system in a straightforward manner*

A task to deduce closed loops would probably be fairly exhaustive* ana certainly not very fast* A bad thing to do would be to require its execution immediately before each pathfinding* Instead, it could run continuously and autonomously as a KURT task on the control processor* During wall following, it could run probably at 702

capability, and a run down a hall would be plenty long enough to consider all the possibilities in even a very large map at least once.

Some notes on implementation: if there are several nodes on the same NS or EW line, only adjacent pairs of nodes may be considered for having a path between them. In the event of the failure of a deduced link during minimum path finding, the cartographer will simply remove the link, and create a new node with appropriate DARKs. Since it is highly likely that new deduced links should be set up, the deducer must be run on just the current newly created node. This will not take all that long, and is not hard to implement.

11 • further Efforts

11 •1 Better Sonar

The sonar system suffers from limited bandwidth* How might we increase the sonar scan rate to improve perception of the environment? An obvious approach is to increase the physical number of sonar units taking measurements*

First* we might add a second sonar system also mounted on a stepper* One system would be mounted forward, and the other aft* The forward mounted system looks forward and left or right, depending on which wall is being followed* The *rear* mounted sonar looks alternatively left and right* Perforeance of such a system would be approximately 1 scan per second* Stepper speed still limits maximum input rate*

The stepping motors may be entirely eliminated by the construction of a fixed five element sonar transducer array* One transducer would look forward, *and* two towards each side* Distance measurements would be sequenced to insure non-interference* A full scan could be completed in ZOO msec, which is a reasonable return on hardware investment* By interlacing and not making each junction analyzer measurement each scan, the A and B sonars may be kept fully occupied, with a wall follower scan time of 150 msec, and junction analyzer scan time of 300 msec* The full scan approach is probably more desireable because it does not sacrifice the junction analyzer scan time*

Both speed improvement methods have an additional side-effect: all measurements may be made at right angles to

the halt, consequently, sonar bounce will not occur even on angle measurements« when the robot is parallel to the wall*

The angle to the wall is atan <A-e)/D where D is the linear separation of the sonars* Ignoring the $atan_f$ the calculation requires but a single multiplication (ana a trivial suotraction)•

The difference in the A and B measurements is less sensitive to tne angle to the wait than in the moving transducer rethod which is actually used* The minimum detectable angle is 4*6 degrees* versus 2*3 degrees for the system which is used* The minimum detectable angle is constant under the parallel perpendiculars scheme; with the spread beats system, the minimum detectable angle decreases with increasing distance from the wall*

The development of a synthetic aperture sonar similar to that currently used in underwater antisubmarine warfare anc some radar systems would be nice, but would have to deal with the reflection problem*

11*2 Simple Visual Processing

The ultrasonic sonar is crippled by its inability to make measurements within a narrow region: its beamwidth limits its resolving power to decipher the exact situation* This is most apparent in junction analysis, where the sonar is unable to provide an assessment of its angle and position within the junction* A laser rangefinder can provide the narrow beaawidth, but only at a limited bandwidth* By contrastt a vioeo camera has high bandwidth and a narrow

bearaaicth*

The high bandwidth of TV cameras is both a blessing and a curse* A video image takes a long time to process on present day computers, as the Stanford cart illustrates* for the SCIPR junction finder, the locations of the corners of a junction *ould be most useful, since the angle with respect to the corner say be determined* The junction locations *may* be discovered by analyzing only a strip of the image, to look for vertical gray level transitions* This analysis may be completed many times more rapidly than a full picture analysis of virtually any type* A mechanically scannec linear photooiode *array* camera might be useful in generating the picture in a convenient fashion*

The Stanford cart's problem was the inability to deal with flat unmarked surfaces* SClttR has the means to resolve this profclew: the sonar* Once corner locations are identified, the sonar can take measurements in between, along the axis of the hypothesized hallway, to verify or disprove its existence* The sonar system can provide the necessary focussing information to the camera* In the past, focussing required elaborate crunching which attempts to maximize "noise" content in the image* The sonar can focus the camera directly* The sonar and camera are thus mutually co»p leaentary•

The visual processing requirements may all de met by the bdoitior of a single additional computer of the same basic design as those currently used*

11*3 Advanced «tortd Modelling

It *is* fair to say that SCIMR lives in a small world* How might we increase the *size* of his world? The next step mig.ht ae to deal with a partially clattered world which includes rocms with taoles, desks*, benches, etc* The ultrasonic sonar is incapable of dealing with these problems: the sonar provides insufficient resolution* A table leg is indistinguishable from a large desk, at first glance via sonar* Although they might be resolvea by careful scanning, a robot which plans to make good headway must use vision* The data representation problem becomes much aore acute as more understanding of the world is requi*red* *

A possible world model is based on spaces *and* obstacles* A hallway becomes a long thin space, and a room a thick rectangle* An intersection is an overlap between areas* A table or chair is an obstacle, associated with the area it resides within* Using a rectangular world, *an* obstacle may be denoted by an area, position, length, and width* A small note: doorways*, which are regions of possible trespass between non-overlapping areas (assuming them to be rectangular) may be represented as small rectangles overlapping both areas, such a representation reflects naturally the discovery of a room after passing through a door: there was an old area, then a transition region, the doorway, and then a new room on the other side* Unlike HILAPE, areas have obstacles within them, and

obstacles have areas within them* recursively defining the world* for example, the real world is the outermost area, with buildings, rooms, and desks successively narrowing the region under consideration* Such a model is trivially extended to arbitrary polygons*

The mcdel allows consideration of different levels of detail* Path finding begins by mapping the starting and ending points to the outermost coarsest level* finding a minimus path, locating endpoints, and recursing to a finer level of detail* The path finding task is partitioned into small subtasks*

Multi-floor topologies may be represented by storing a floor number with each node, with elevators tagged as an area per floor linked by a special "vertical area' representing the elevator ride* The path finder will operate precisely as before; the path executer must just *remember* to push the buttons when running the path*

The choice of the data representation simplifies the processing which must fce performed on it* This representation is sufficient to deal with most indoor applications* It is not, however, a general purpose A*I* world model* None of this effort attempts to operate on worlos containing "Rube Goldberg" requirements for locomotion, such as having to push ramps into place*

11*4 Advanced visual Processing

Impterentation of visual processing necessary to construct this world model is difficult* Any attempt must

use incremental picture processing. As the robot drives along, rather than performing a complete analysis of each picture input, the results of past analysis is mapped onto the new picture, and new features located. Once located, these features may be analyzed over the course of several successive images. In this way, picture processing can require a much lower aggregate processing bandwidth than full picture analysis.

The mapping of old objects onto a new picture can be done via prediction, rather than brute force comparison, as in past systems. This requires a knowledge of vehicle motion and past object position. For new objects, the exact three dimensional position is not known. The two dimensional position may be used to guide a localized search for the corresponding image in the new frame. Once found, the location may be deduced for further use. The redundant location information obtained during correlation of existing object positions can yield fine tuning for the robot position and the object locations.

It is my belief that special purpose hardware for implementation of incremental picture processing can be built to process video in real time. The high speed is made possible by the restricted search domains generated by object location prediction.

11.5 Finding Your Own way

Previous discussion has ignored one significant aspect of the human map learning process: the reading of signs.

Their prevalence in the American highway system is indicative of the importance of this path finding mechanism. Much navigation can be performed on the highway system using signs, although they are often suboptimally or incorrectly placed.

Can a visual processing robot read signs? At the current state of the art, the answer is probably not, at least normal signs, like LIBRARY->. One amusing problem is that human signs are at our eye level, which is significantly different from that of SCIMR or other robots with a conservative weight distribution. A robot looking for signs would spend a lot of its time looking up into space, making other picture analysis problems more difficult.

We could kill both birds with one stone, by building robot 'readable' signs which are easier for robot processing. One simple technique would be to place bar code signs similar to the UPC code on groceries down at the robot's camera level. Alternatively, we could use an ultrasonic or radio beacon as a sign, transmitting appropriately encoded data. Of course, directionality would be a problem, but not if the robot could select the correct interpretation based on its own knowledge of its direction, or deduce it from the environment with the aid of beacon data. Most human signs, on the other hand, supply a direction, rather than require it.

What data should a sign contain? Human signs supply,

in various combinations, directions (N, S, E, W), place names, and route names. The route names are mechanisms to reduce the amount of information necessary to specify a path to a place, even though the path so specified may be less than optimal. In a network as complex as the highway system, exact specification of all of the possible destinations of a road is pragmatically impossible.

In a limited applications environment, a covey of robots might as well be programmed with the entire world model, and do away with such niceties as signs. The need for signs would arise as the robots begin to operate in extended environments. The storage of all of a large environment is inefficient, since the robot is interested in only a limited area at a time. A sign, particularly a beacon, can be considered as a download of a portion of the local world model.

11.6 A Bigger Crank

The SCIMR processors were constructed by the author to drive a SIMD parallel processing system, the subject of my senior thesis. The processors are not specialized for operation in SCIMR. How might we optimize them? Contrary to normal expectations, the sonar and motor processors are actually overqualified for their tasks. Currently, each contains 4K bytes of static RAM, and a small (32 byte) download ROM for program loading via the interprocessor communication port, which is a PIA. Additionally, each contains a second PIA which is used to run the I/O devices

in SCIMR.

In reality, 256 bytes of RAM, 2K of EPROM, 1 parallel port (20 lines), and one serial port (at 200K baud or so) would satisfy the hardware requirements. A smaller version of the processors could be built from these specs.

The M68701 single chip computer meets all of the specs except the RAM requirement. In the near future, this should be remedied. Replacement of both processors with a single chip computer would be highly desireable from both space and power standpoints. As an extra bonus, the newer processors contain some instruction set enhancements, such as hardware multiply, that would be nice. Programs would be largely compatible, except for address changes, and slight changes due to using a serial port for data transmission.

The current main control processor is identical to the motor and sonar processors except for the addition of an ACIA and another PROM (64 bytes total) to handle download from it. In a more complex processing environment for dealing with vision, the control processor must be greatly expanded. In particular, consider the hardware for processing the visual world model described earlier based on areas and obstructions.

Necessary processing devices would include a video digitizer and high speed frame processor to perform initial processing on the picture. This processor would be coupled directly into a numeric processor consisting of a 16 bit microprocessor and floating point chip, to handle numeric

correlations and distance trigonometric calculations* The arithmetic processor is coupled to the main control processor through a shared memory/hardwired semaphore interfaces capable of high speed data transfers* The control processor consists of a 16 bit microprocessor with 64K of onboard dynamic RAM chips (64K chips)* The control processor would interface with the sonar and motor processors via some additional hardware which interfaces the low speed sonar and motor transfer interface to the hardwired builtin system on the control processor*

11*7 Acoustic Processing

SCItfR presently has no way of telling what kind of surface he is driving over* We can rectify this by allowing SCIftft to listen to himself roll* As he goes over different surfaces, the emittea sounds are different* An example is the difference between carpet and asphalt* The circuitry required for tftis is a microphone and a spectrum analyzer* The best mounting location for the microphone is not clear* The microphone could be mounted underneath SCIWR oriented to receive sounds conducted through the air* Alternatively! it could be mounted to listen to the sounds conducted to it through the frame*

&uitaing a miniature low cost$_f$ low power spectrum analyzer may seem difficult* However, we may exploit a *prc*perty of the input signal to simplify the device increcibly* Since the robot runs continuously, more or lesst the irput is continuous* Changes in input signal

characteristics occur infrequently. Instead of having to store sections of input signal and perform analysis on them or try to perform analysis in parallel, we can build a swept frequency analyzer, where the frequency sweeps at a fairly slow speed. The device consists of an input filter, a frequency shifter (analog multiplier), a narrow fixed bandpass filter, and a signal power integrator. Mostly analog circuitry is involved. The system operates by digitally synthesizing a frequency which is added or subtracted in the frequency domain to the input, shifting the desired input signal frequency into the passband of the bandpass filter. For example, to find the power at 400 Hz, with a 2000 Hz bandpass filter frequency, we synthesize a 1600 Hz signal. To prevent 3600 Hz from also being passed, the input filter cuts off frequencies less than 2000 Hz. The signal power may be found by letting the positive part of the filter charge a capacitor. The computer discharges the capacitor at the start of a measurement, and then accumulates the length of time for the capacitor voltage to reach a fixed threshold. With an appropriate calibration table, this yields input signal power at the frequency.

An Intel 2920 analog signal processor I.C. could be programmed to perform the spectrum analysis instead. If available, it would be vastly preferable.

In operation, a processor would sweep the analyzer frequency over some band of interest, and record the signal power at intervals. A matching operation to compressed

stored spectrums would identify the material being crossed.

A system of this sort would need facilities to suppress acoustic noise from the sonar/stepper and main drive motors. The latter source is particularly interesting, since the noise frequency and amplitude are related to current motor drive conditions. The frequency range of interest in robot acoustic surface type detection is not known, nor is the frequency of motor noise, except that it can be expected at the frequencies of treads hitting the ground, and of the gears meshing at different speeds in the gear train. Exact determination of the signal characteristics must await at least a microphone on SCIMR, and probably the analyzer itself.

Hoy srrart is SCIMR? I really do not know what scale to use. I think that the range of representable worlds is a good measure of intelligence of robots. I have pointed out many situations SCI PR is incapable of representing. The advanced world model described earlier would resolve many of these problems* if there were a method of generating it. Several remaining problems: ramps, elevators, overhangs, require a knowledge of the three dimensionality of space not present in any of the robots discussed.

SCIMR is like a programming language without type constructors—- he cannot reach beyond his initial understanding. SCI^R can learn, but only within the framework provided. Perhaps this can be allayed in the nestable advanced world model, but the fixity of representation is a strong indictment of SCIMR ana A.I. in genera I.

13. Conclusions

SCIMR is able to drive down hallways, build maps, and use them. The analysis of intersections needs empirical refinement to be reliable using only sonar, or additional sensory capability. By using multiprocessing and multitasking, the robot is able to operate in real time. A memory efficient world model and specialized processing routines allow a small microprocessor to perform artificial intelligence tasks which would traditionally require a large mainframe.

SCIMR paves the way for the development of commercially viable low cost mobile robots.

14. Acknowledgements

The author would like to thank *Or** Ruzena Bajcsy for her ossistarce *and* encouragement over the years, and for allowing *me* to work on such a cra2> project as SCiMR*

1 would also like to thank *Dr.* Samuel Gotdwasser, Davio Brown, Clayton D*ne$_f$ Jeff Wolfe let* and other students for their input to *my* creative process$_f$ and for listening to ana commenting on ay iaeas.

My wife, Amy, was invaluable in her support and proofreading of this document*

Refe r ences

[13 Aho, A.| Hopcroftf J*« Ultman» i# 1974# The Design and Analysis of Computer Algorithms* Acfdf son-Wes ley • pg# 200*

C2 3 bardner, *n** Novemoer 1980* Mathematical Games. Scientific American*

£33 Mcravec* H. 198C* Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover* Doctoral Thesis, Stanford University* Republished by the Robotics Institute of Carnegie-Mellon University*

[43 Standard Mathematical Tables* 1975. CRC Press* pg* 473*

C5 3 bclfeldf J* Private Communication

Table 1: Semaphores

Sonar		From	To
0	Left distance	S1	S2
1	Forward distance	S1	S2
2	Right distance	S1	S2
3	Sonar Angular Position	S1	SI1
4	Sonar Distance	SI3	S1
5	Set Junction Control	C2	S3
6	A distance	S1	S4
7	B distance	S1	S4
8	C distance	S1	S4
10	Scan Pattern select	C2	S1
12	Morse data	many	SI4
1F	Garbage	-	-

Main		From	To
0	A distance	S4	C2
1	B distance	S4	C2
2	Junction type	S2	C2
3	Next direction	C1	C2,C3
4	Turn complete	M1	C2
5	Done	C3	C1
6	SCIMR X location	M2	C3
7	Y location	M2	C3
8	Official junction	C2	C3
F	Minimum distance to wall	S4	C2
1F	Garbage dump	-	-

Motor		From	To
0	A distance	C2	M1
1	B distance	C2	M1
2	Turn timer done	MI1	M1
3	2.5 ft moved	MI1	M1
4	Normal speed	C4	M1
5	Set distance to wall	C4	M1
F	Minimum distance to wall	C2	M1
10	Motor control mode	C2	M1
1F	Garbage	-	-

See Table 2 to identify From and To tasks.

Table 2: Tasks

Sonar		
S1	JUNCON	Interprets scan patterns
S2	JUNTYP	Analyzes junctions
S3	JUNSET	Controls JUNTYP thresholds
S4	RADFIX	Detects and corrects mirroring
Sonar_Interrupt_Tasks		
SI1	STEPPER	Drives stepping motor
SI2	POWER	Controls sonar power cycling
SI3	DIST	Calculates distance to echoes
SI4	MORSE	Beeps the beeper
Control		
C1	PATH	Interprets the path
C2	NAVIG	Performs wall following and turning
C3	CARTOG	Generates maps
C4	MONITOR	Computer to human interaction
Motor		
M1	SPEED	Interprets speed control bytes
M2	LOCATE	Maintains robot location
Motor_Interrupt_Tasks		
MI1		Does all interrupt handling

Table 3: Scaling

Distances	UUUUUUUU.
Angles	S.SSSSSSS
FT	UU.UUUUUU
KDF	0.CCCCUUUUUUUU
KRL	UUUU.UUUU

U indicates unsigned numbers, and S signed numbers.

Table 4: Static Simulation Excerpts

A,B= 15, 15

DIST	DESA	PERA	TURN	FREQ CH.
1.4	25.7	0.0	25.7	5.3
1.5	23.8	0.0	23.8	
15	68	0	68	4
0F	44	00	44	04

A,B= 20, 15

DIST	DESA	PERA	TURN	FREQ CH.
1.5	23.6	19.8	3.8	0.8
1.5	23.8	20.6	3.1	
15	68	59	9	0
0F	44	38	09	00

A,B= 20, 30

DIST	DESA	PERA	TURN	FREQ CH.
2.0	13.8	-26.8	40.5	8.3
2.0	13.7	-28.7	42.3	
20	39	-82	121	8
14	27	AE	79	08

A,B= 20, 40

DIST	DESA	PERA	TURN	FREQ CH.
1.9	15.7	-40.1	55.7	11.4
2.0	13.7	-44.4	44.4	
20	39	-127	127	9
14	27	81	7F	09

A,B= 40, 35

DIST	DESA	PERA	TURN	FREQ CH.
3.4	-14.9	9.6	-24.4	-5.0
3.5	-15.8	9.4	-25.2	
35	-45	27	-72	-5
23	03	18	B8	FB

Table 5: Dynamic Simulation

RUN PARAMETERS: AUTO-MIX CONSTANT 3.0000
STABILIZATION DISTANCE 2.6890
DcGhEES PER FOOT OF ERROR 20.0000 -
VELOCITY ' 1.0000 TIME PER ITERATION 1.0000

REAL X	PcRC. X	REAL AN6	PER. ANG	DES. AN6	NEW Y
1.C	1.1	-30.0	-26.5	31.5	1.0
C.£	0.8	3.5	2.3	37.5	1.9
1.1	1.2	38.8	33.2	30.2	2.7
1.8	1.8	3S.1	40.1	17.3	3.5
2.2	2.2	2C.0	18.6	9.1	4.5
2.5	2.5	3.6	7.8	2.9	5.5
2.6	2.7	1.3	1.1	-0.9	6.5
2.6	2.7	-1.9	-1.7	-0.3	7.5
2.5	2.6	-4.0	-3.6	0.8	8.5
2.A	2.1	-2.9	-2.6	1.9	9.5
2.4	2.6	0.8	0.7	1.8	10.5
2.5	2.6	1.7	1.5	1.6	11.5
2.5	2.6	1.9	1.7	1.0	12.5
2.5	2.7	0.4	0.4	0.1	13.5
2.5	2.7	0.0	0.0	-0.1	14.5
2.5	2.7	-C.1	-0.1	-0.1	15.5
2.5	2.7	-0.1	-0.1	-0.1	16.5
2.5	2.7	-0.1	-0.1	-C.1	17.5
2.5	2.7	-C.1	-0.1	-0.1	18.5

• • •

Figure 1

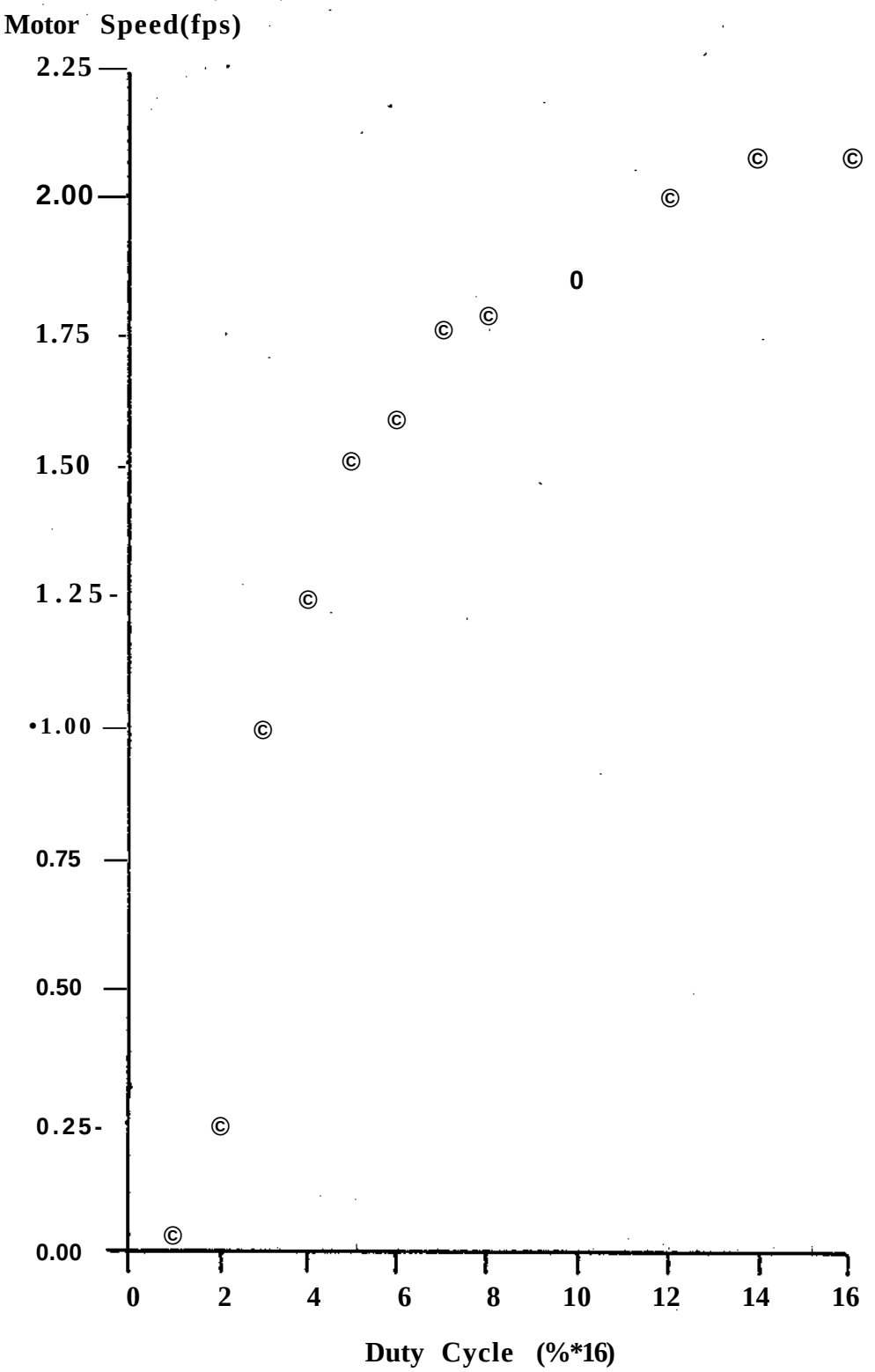

Motor Speed(fps)
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10
12
14
16
Duty Cycle (%*16)

Figure 2

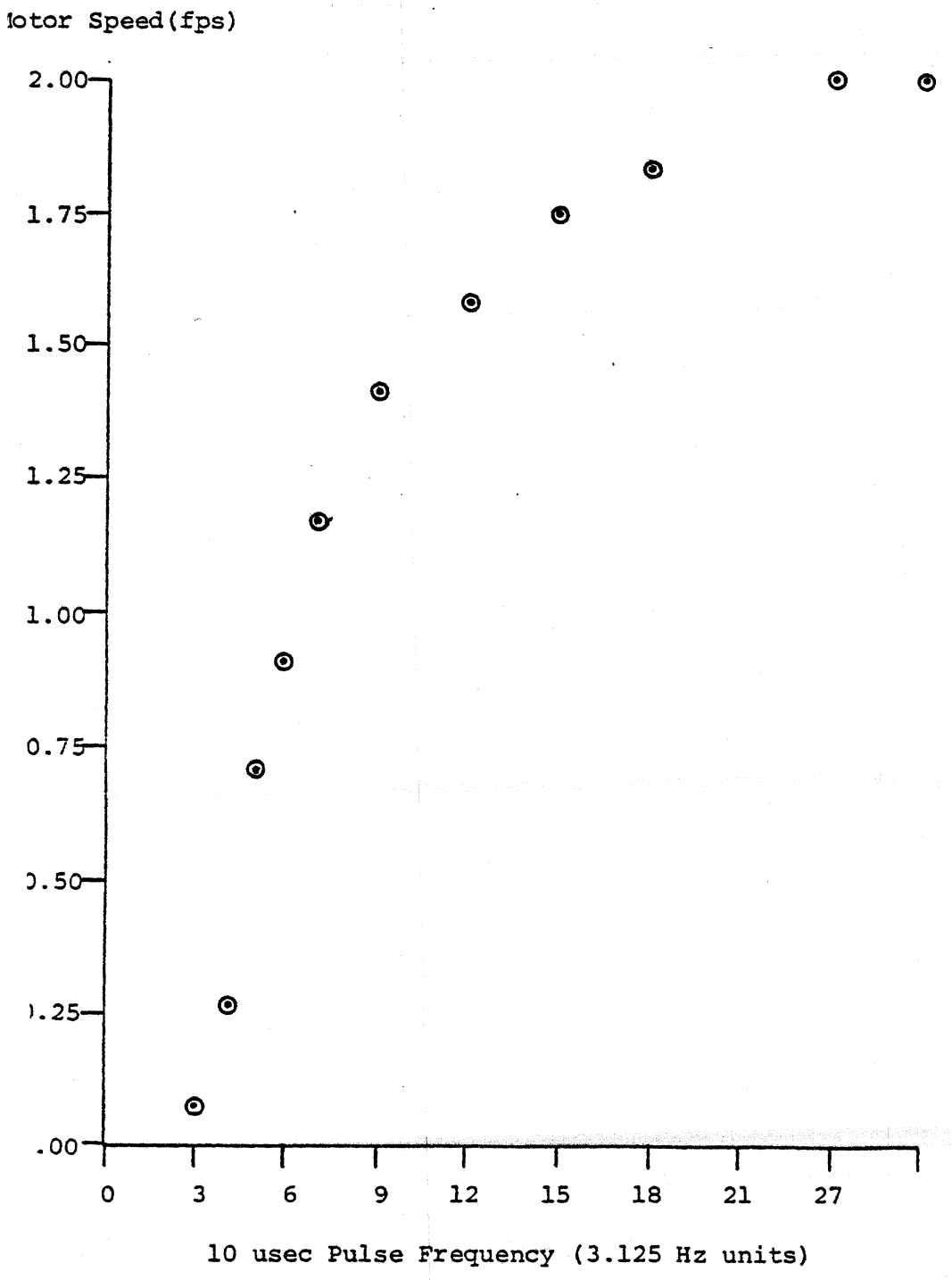
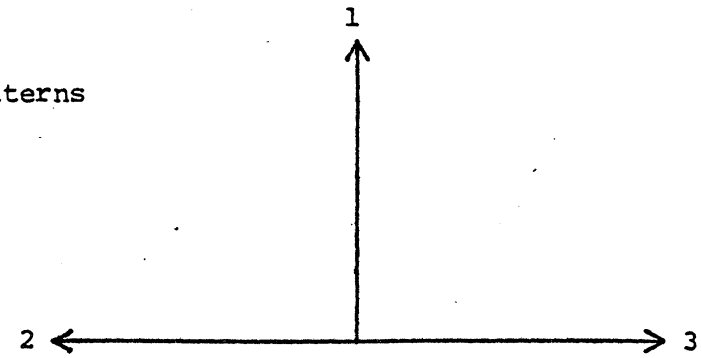
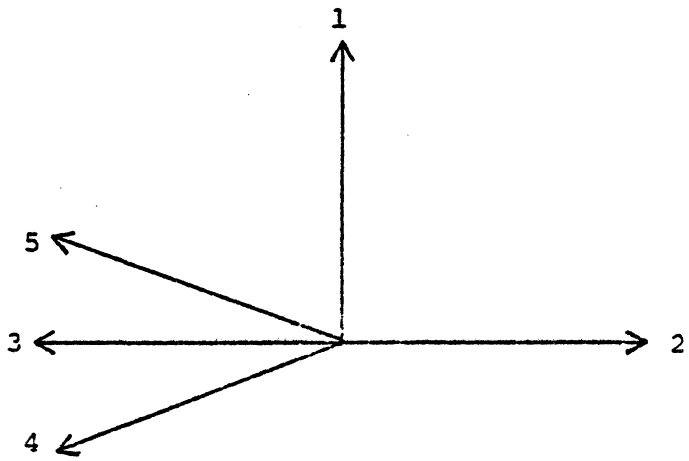
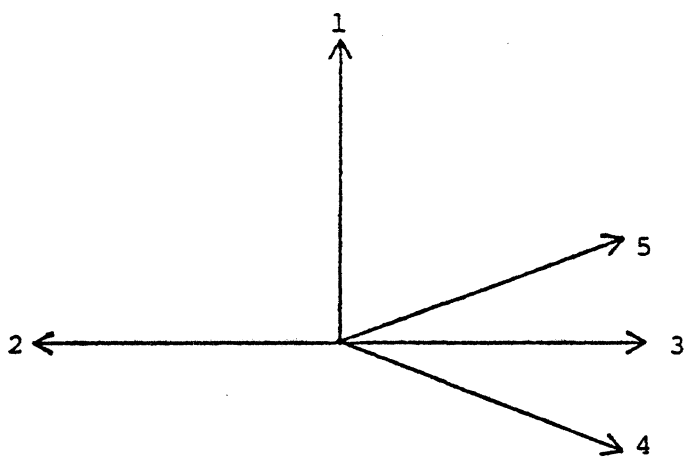

lotor Speed(fps)
2.00
1.75
1.50
1.25
1.00
0.75
).50
).25
.00
0
3
6
9
12
15
18
21
27
10 usec Pulse Frequency (3.125 Hz units)

Figure 3:
Sonar Scan Patterns







Numbers indicate order of measurement

Figure 4:

Moore School, 2nd Floor

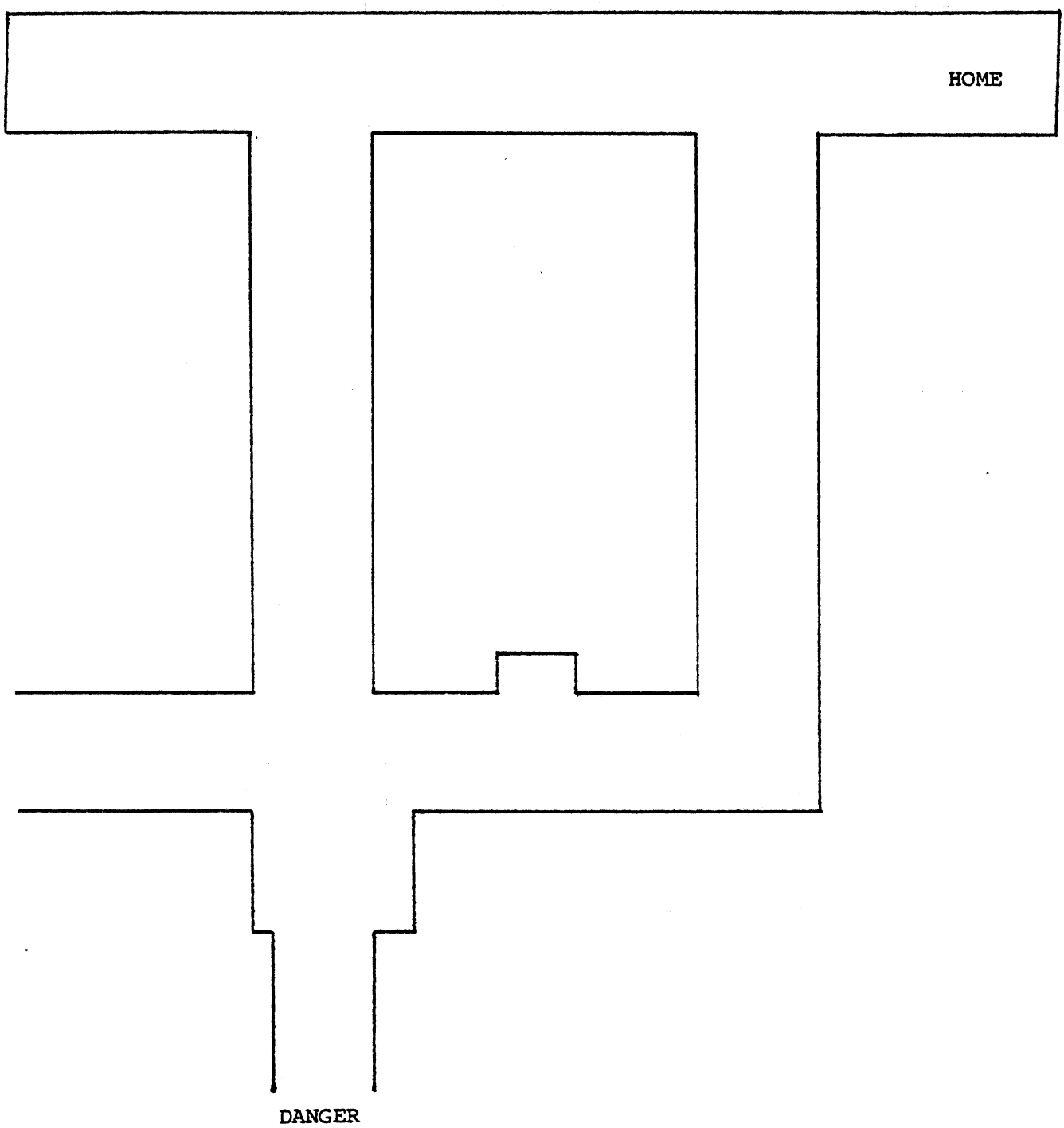


Figure 5

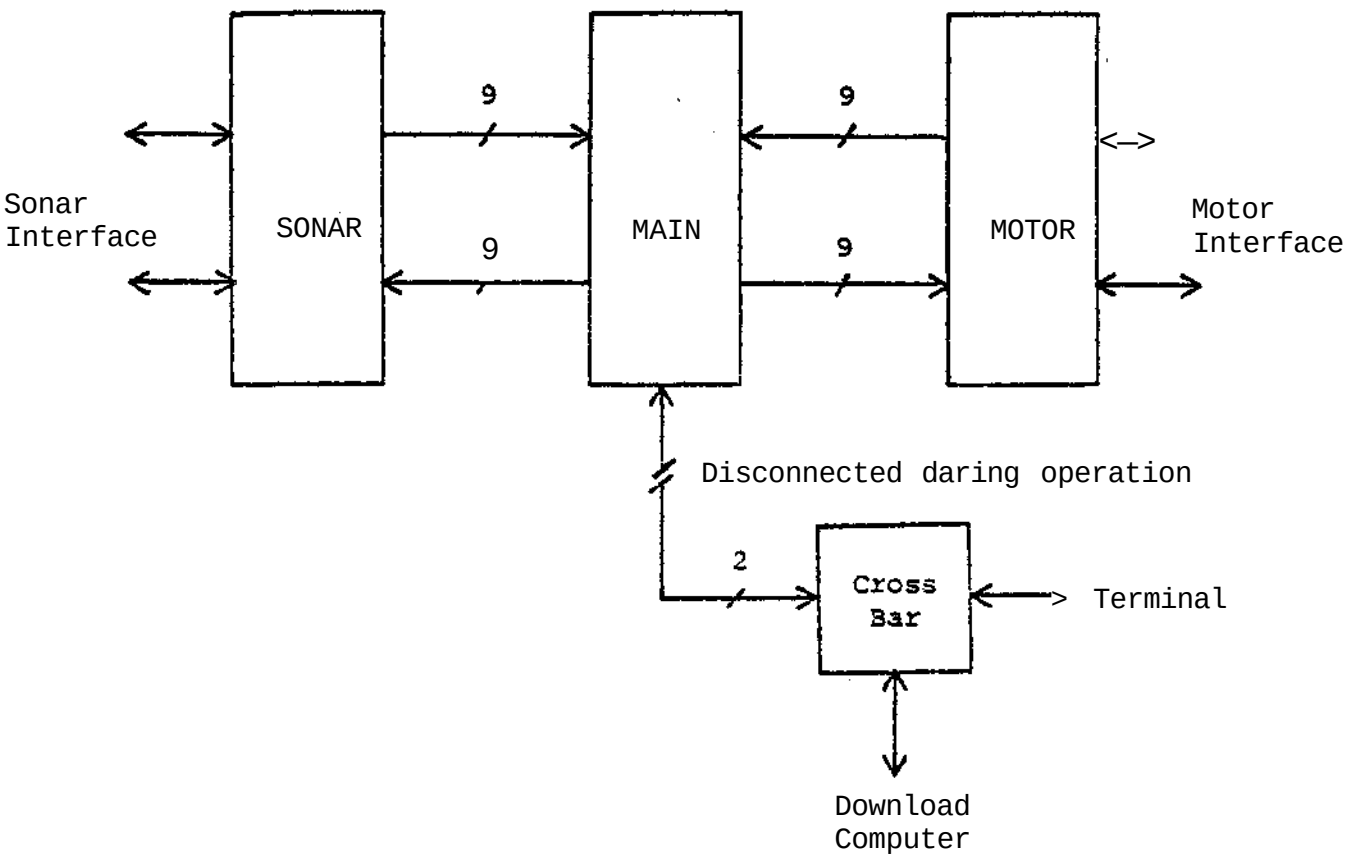

Sonar
Interface
SONAR
9
9
MAIN
9
9
MOTOR
Motor
Interface
Disconnected daring operation
2
Cross
Bar
Terminal
Download
Computer

Figure 6

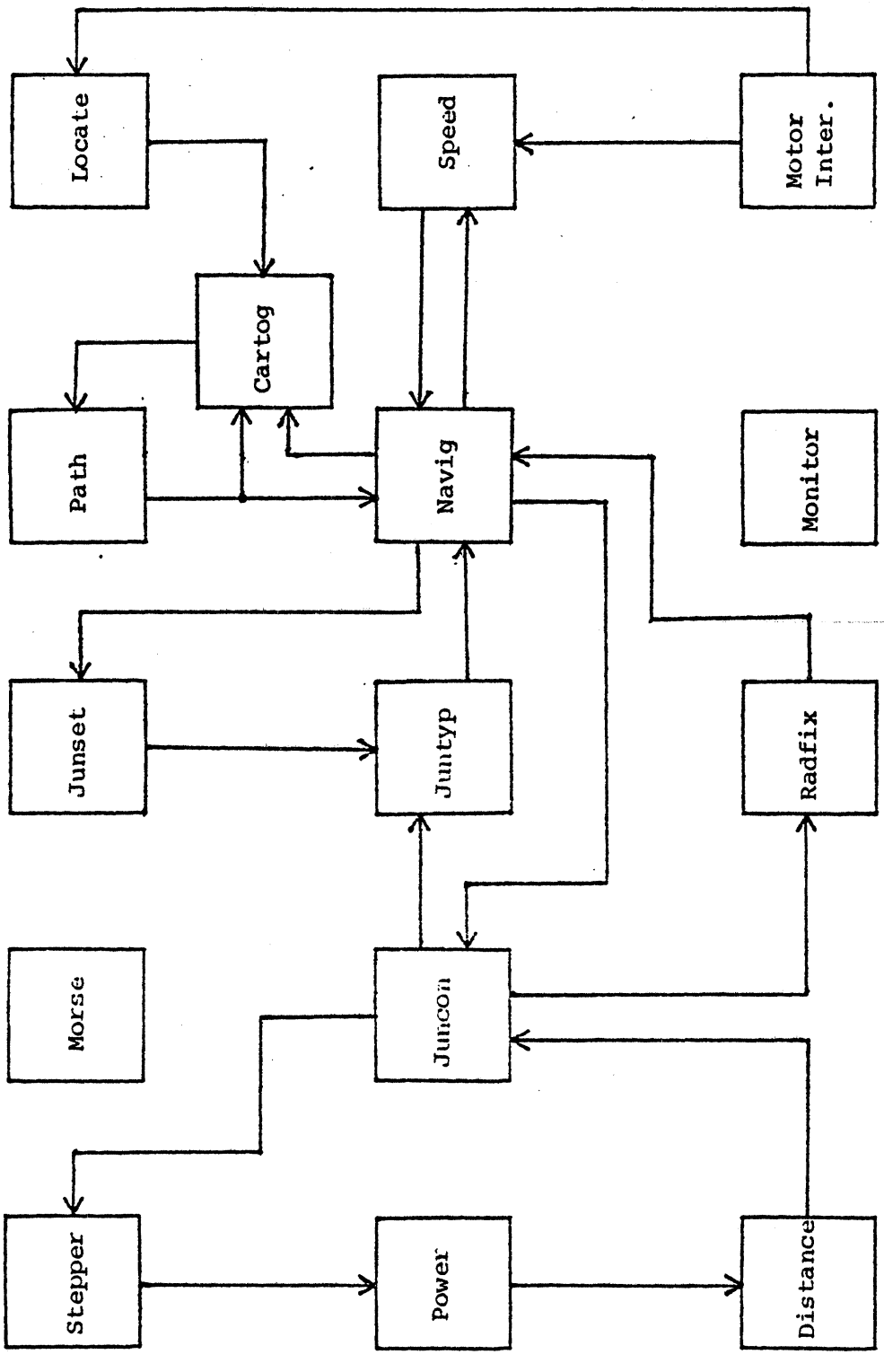
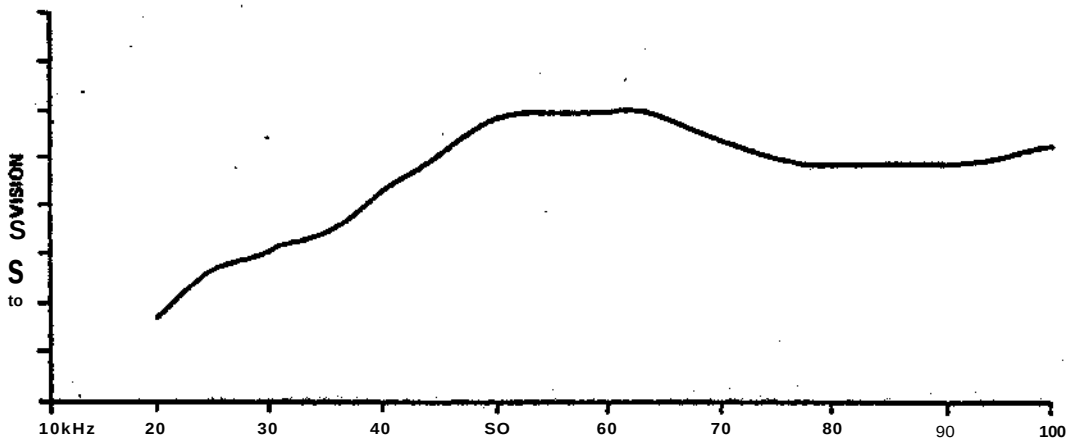
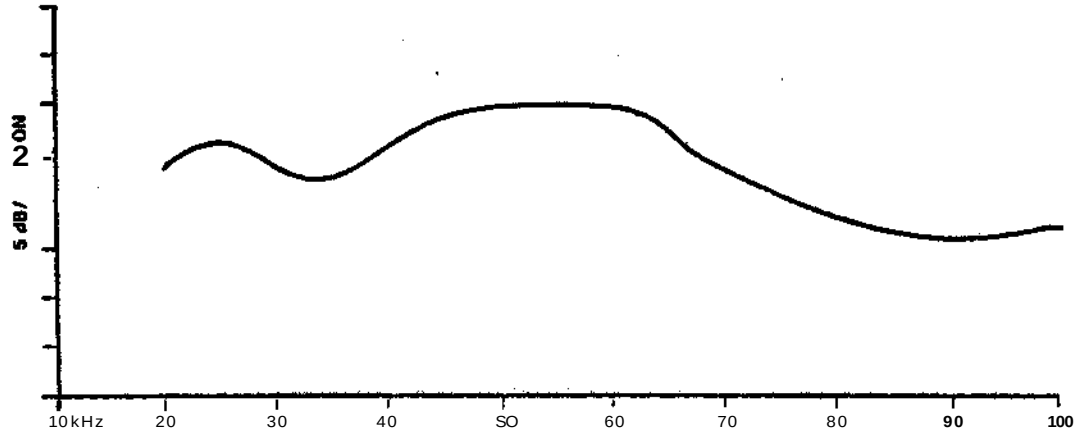

Locate
Speed
Motor Inter.
Cartog
Path
Navig
Monitor
Junset
Juntyp
Radfix
Morse
Stepper
Power
Distance

Figure 7– From the Polaroid Ultrasonic Ranging System Manual

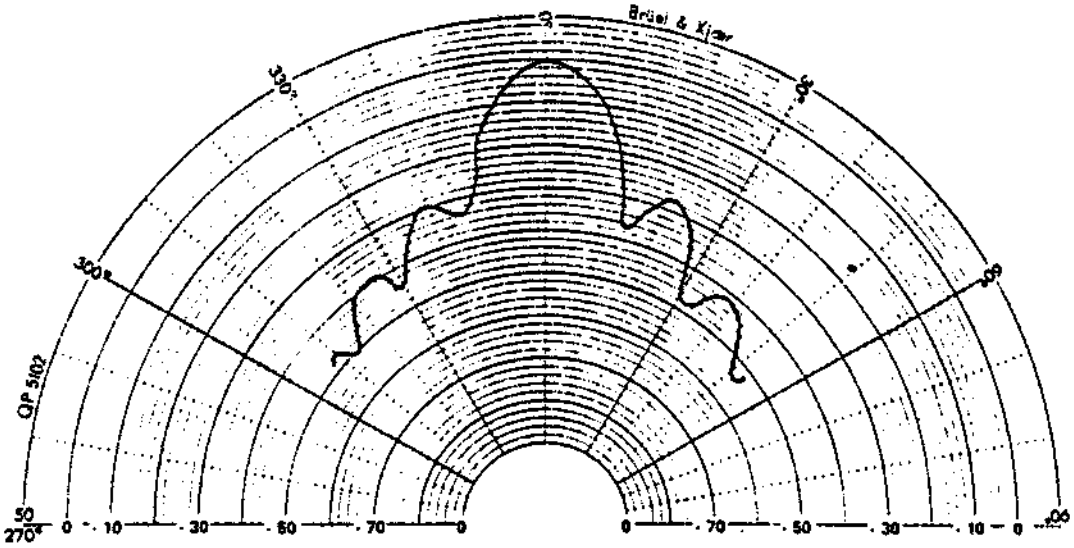
NOTE: These curves are representative only. Individual responses may differ.



TYPICAL TRANSMIT RESPONSE



TYPICAL F^HE FI£! D RECEIVE RESPONSE



TYPICAL SEAM PATTERN
AT 50 kHz

NOTE: db normalized to on-axis respond.

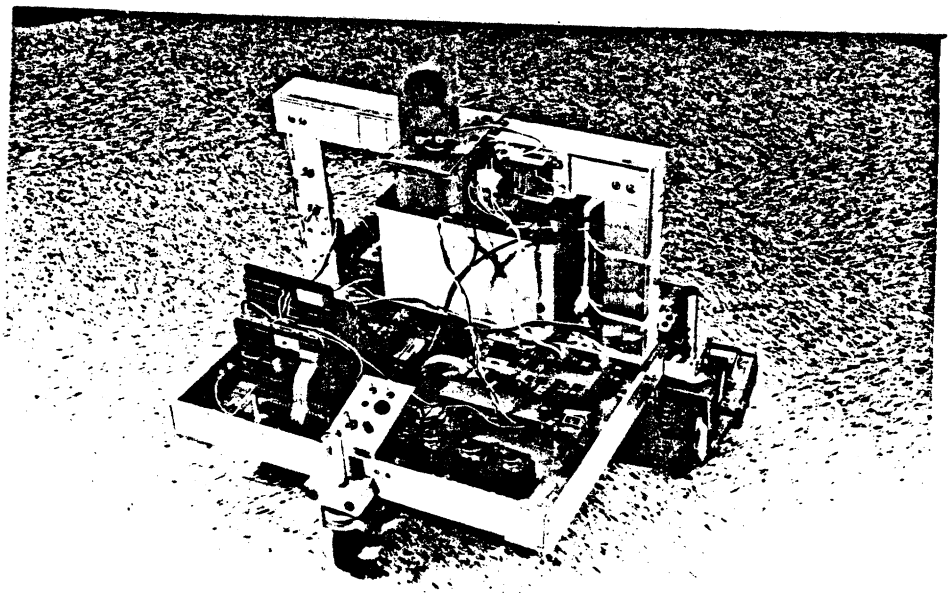


FIGURE 1. Front View of SCIMR Robot

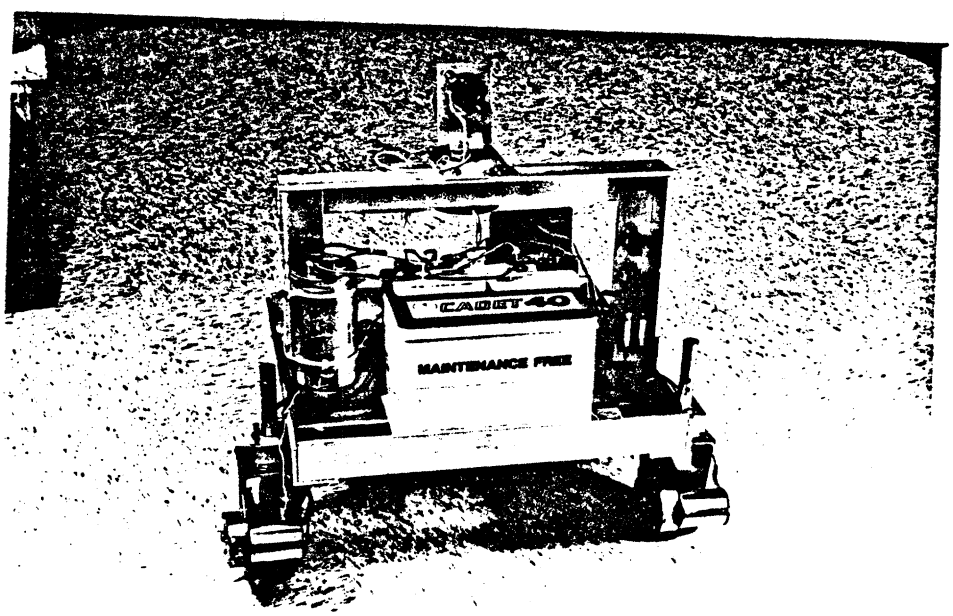


FIGURE 2. Back View of SCIMR Robot

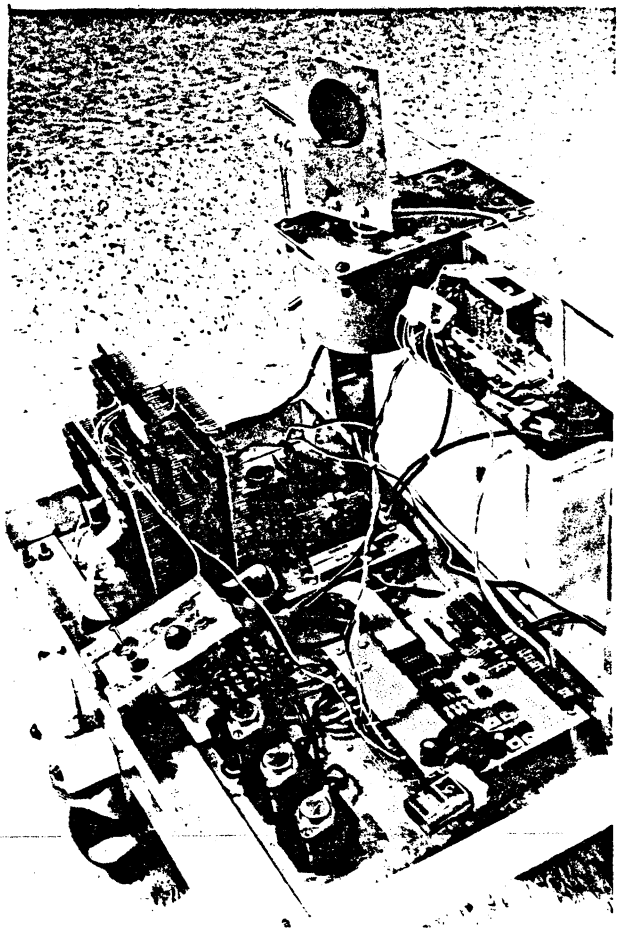


FIGURE 3. Detail of the Electronics of SCIMR Robot

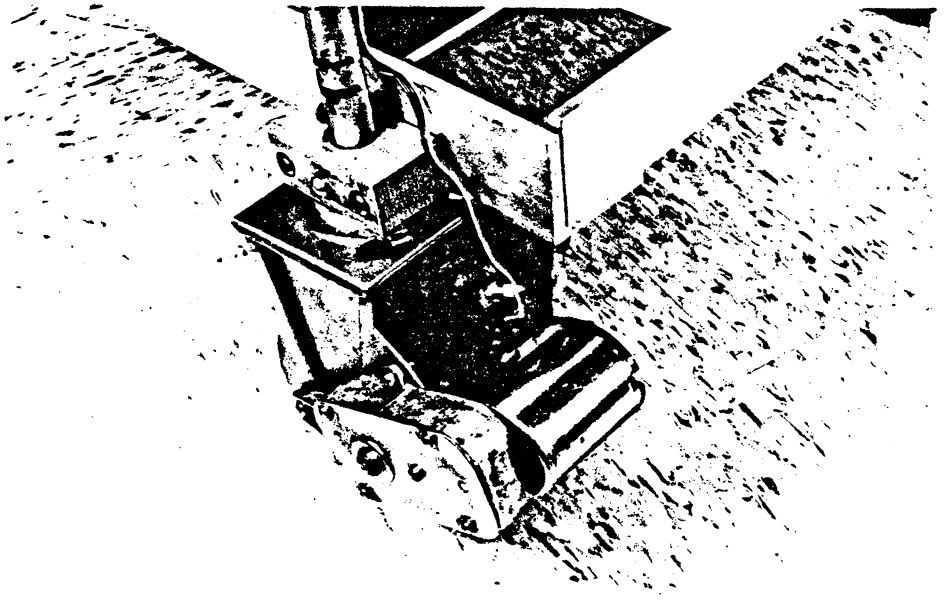


FIGURE 4. Closeup of the Wheel and Tackometer of SCIMR Robot