The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Working Paper 265 February 1985

A Mobile Robot Project

Rodney A. Brooks

Abstract. We are building a mobile robot which will roam around the AI lab observing and later perhaps doing. Our approach to building the robot and its controlling software differs from that used in many other projects in a number of ways. (1) We model the world as three dimensional rather than two. (2) We build no special environment for our robot and insist that it must operate in the same real world that we inhabit. (3) In order to adequately deal with uncertainty of perception and control we build relational maps rather than maps embedded in a coordinate system, and we maintain explicit models of all uncertainties. (4) We explicitly monitor the computational performance of the components of the control system, in order to refine the design of a real time control system for mobile robots based on a special purpose distributed computation engine. (5) We use vision as our primary sense and relegate acoustic sensors to local obstacle detection. (6) We use a new architecture for an intelligent system designed to provide integration of many early vision processes, and robust real-time performance even in cases of sensory overload, failure of certain early vision processes to deliver much information in particular situations, and computation module failure.

1. Introduction

Suppose you could walk into Sears and buy a USMM brand home robot. You put it in your trunk, drive home and place the box on your living room floor. You open it up and grab your USMM robot by the head, pull it out, and place it on the floor. You don't need to switch it on–just say "Hello". It springs to life, looks around the room for a bit, thinks, then heads off towards the kitchen. You follow it as it explores the house. Occasionally it turns to ask you a question, like "Would you like this floor waxed or just mopped?". Sometimes you take the initiative and say something like "Never come in this room before 10am, and always knock in any case.". After a couple of circuits of the house your USMM robot is happy, and you never have to do a scrap of housework again.

Ignore for the moment the problems of speech recognition and natural language understanding. What would it take to build a software system which could control a robot such as described above? An attempt to answer that question is the topic of this working paper.

There are a number of assumptions and foundation points which make our approach different to the approaches used by other groups building and controlling autonomous mobile robots. We discuss these points below. As we go we will compare our approach to those of [**Moravec 1983**]'s rover project and [**Giralt et al 1983**]'s Hilaire. These are the most advanced examples of mobile robot projects.

1.1 *Dogma*

▸ **1.** We are interested in building robust mobile robot control systems useful for cheap robots (hence imprecise in both sensing and control) working in unstructured domains such as the home, material handling in factories, street cleaning, office and hotel cleaning, mining and agriculture. The same capabilities can be useful for robots, which do not have to be so cheap to be economically feasible, and which do tasks like: planetary exploration, space station maintenance and construction, asteroid mining, nuclear reactor operations, military reconaissance and general military operations.

For all these applications a robot must wander around an unstructured environment. If it is able to build and maintain a map by itself its functionality within the environment will be much improved.

▸ **2.** We do not believe that a robot can perform many useful tasks if it models its world as a projection in a two dimensional plane. Nor does it suffice to decorate those projections with "height-of-feature" information. The world a robot must operate in is inherently three dimensional even if it can locally move in only a two dimensional plane. Many objects in the world are small enough that the robot can see over the top of them, but not so small that it can roll or step over them. On the other hand some objects hang over space that is navigable by the robot. Such objects can not always be ignored. They may on occasion block the robot's view of landmarks for which it is searching. They may have surfaces that the robot wishes to interact with; e.g., a table top that it can reach up to and dust, or a deck that the robot can get to via some stairs or a ramp.

▸ **3.** Almost all mobile robot projects have had as one of their underlying assumptions that it is desirable to produce a world model in an absolute coordinate system. However all sensors and control systems have both *systematic* and *random* errors. The former can be dealt with by calibration techniques (although these are often time consuming and are confounded on mobile robots by the fact that the robot itself is not fixed to any coordinate system). The latter are always present. It is usual to model some worst case bounds on such errors but this will not always suffice (e.g. mismatches in stereo vision can produce depth measurements with error magnitude the full range of depths which can be measured). In any case the bounded errors at least must be dealt with in building models of the world and using them. A number of approaches have been taken to this problem:

- a. Ignore it. This has only been successful in the most toy-like of worlds.

- *b.* Use fixed reference beacons. This implies that the environment is either structured for the robot's benefit in the case that beacons are explicitly installed, or that the environment has been pre-surveyed for the robot's benefit in the case that known positions of existing beacons (e.g. power outlets) are used.

- *c.* Some sort of inertial navigation device is used to determine accurately where the robot is in the global coordinate system. These sensors have a large number of disadvantages, including high cost, drift requiring recalibration to some global reference every few hours, long startup time for gyroscope based sensors, errors too large for use on a small scale such as within a factory or house, and uncertain behavior when subject to high jerk components (such as a tank crossing ditches, or your child playing tag with the vacuum cleaner robot).

- *d.* Almost all wheeled mobile robots come equipped with shaft encoders on their steering and drive mechanisms. In a perfect world with no slip or slide between wheels and ground surface readings from these sensors would provide an extremely accurate estimate of robot position. Unfortunately wheels slip and slide with magnitudes that are functions of, at least, wheel velocity and acceleration, exact ground surface composition and shape, wheel load, and tire wear. These aspects can all be modeled but the surface the robot runs on must be restricted or there will still be errors with essentially the same magnitude. Of course such techniques will be less accurate on legged vehicles.

- *e.* Landmarks are chosen by the robot and used as reference points when next seen, to update the robot position estimate. The landmarks must be chosen for recognizability within some expected uncertainty area.

Approaches a, b, and c above are ruled out for us because of costs or our unstructured domain. We are left with inaccurate motion estimates and the need for visual landmark recognition. Compare this to the *Hilaire* project at Toulouse [**Giralt et al 1983**].

The Hilaire project is probably the most complete and advanced of any mobile robot projects. They have produced by far the best non-vision based results. It uses methods b (infrared beacons), d (shaft encoders), and e (a directable laser range finder) from above to try to produce as accurate as possible a model of the environment in an absolute coordinate system. Even with an elaborate error tracking system and the use of fixed beacons errors still accumulate to the point where they are forced to break up observed objects into pieces and let the pieces change relative coordinates (i.e. the observed objects get deformed in the model relative to what was observed), to maintain a consistent position model [**Chatila and Laumond 1985**]. The underlying problem is that worst case error needs to be assumed in placing things in an absolute coordinate system, and cumulative worst cases soon lead to useless models globally.

We will use no absolute coordinate system. We will not ignore errors nor use beacons or inertial navigation systems. Instead we will use only local coordinate systems with relative transforms and error estimates. We will use shaft encoder readings and visibility analysis and landmark recognition to build a consistent and accurate world model. In the future we might use a non-magnetic compass as these are cheap and from theoretical considerations it appears that a compass may provide a large amount of useful information.

▶ 4. Some mobile robot projects make no real attempt to model the environment on the basis of their perceptions. Such robots will necessarily have limited capabilities. Other projects, especially those based on indoor robots, which have made serious attempts to model the environment from perceptions have often assumed that the world is made up of polyhedra. This assumption manifests itself in two ways:

- a. The perception system produces models of the world whose primitives are polyhedra implicitly assumed to correspond very directly with the surfaces in the environment.

- *b*. The environment in which the robot is allowed to roam is constructed out of large planar surfaces with typically less than 100 planes in the complete environment of the robot.

In fact since most projects have used 2 dimensional representations of the world they often actually model the world as polygons, and the artificial worlds are constructed from vertical planes reaching from the ground to above the range of the sensors.

When algorithms developed under such assumptions and tested in such domains are applied to more realistic worlds, they break down in two ways:

- a. Time performance degrades drastically if there are any algorithms with even moderate time complexity as a function of the number of perceived surfaces. A quadratic algorithm can be fatal and even a linear algorithm can have unpleasant consequences.

• *b.* More seriously, the mapping between the world and its perceived representation usually becomes very unstable over slight changes in position, orientation or, in the case of vision, illumination.

This is not a criticism of using polyhedra as the primitives for modeling. Rather it is a criticism of trying to model the world exactly rather than using multiple scales of resolution.

The second of the above points can be disastrous for two reasons:

• *i.* It becomes much more difficult to match a new perception with an existing partial model of the world. Straightforward matching of edges to edges and vertices to vertices no longer suffices.

• *ii.* If the representation of free space is based on object faces (e.g., [**Chatila 1982**] has the most complete example of such a scheme) providing defining edges of convex polygons, then such representations get almost hopelessly fragmented into many, many convex polygons. The polygons are often thin slivers which extend distances many times the size of the object providing an observed face of the polygon. Thus the polygons have no semantic relevance, besides being completely unstable.

For the reasons we will build no artificial environments. The robot must exist in a "real" environment that humans inhabit and have constructed for their own use. We need to tackle the representation problems with multiple scales of representation to filter out the small high frequency perturbations of the environment which have no real effect on tasks to be performed within that environment.

▸ **5.** Many groups building mobile robots are relying on sonar sensors (usually based on the Polaroid sensor used for auto-focusing) as their primary source of three dimensional information about the world. The reasons for choosing such sensors seem to be:

• *a.* they give direct digital readings of depth, or distance to first echo, and as such

• *b.* they seem to require very little processing to produce a two dimensional description of the world, which means

• *c.* they produce almost real-time data with very cheap processors, and

• *d.* the sensors themselves are cheap so they can be hung all over the robot giving 360° sensing without the need for mechanical scanning.

Unfortunately sonar sensors also have many drawbacks which force many such groups to spend a great deal of effort overcoming them. The major drawbacks are:

> • *a.* the beam is either wide giving ambiguous and sometimes weak returns or when the beams are more focused it is necessary to have either very many sensors or to mechanically scan with a smaller number of them.
>
> • *b.* sonar returns come from specular reflections (analagous to mirror-like reflections in the visible spectrum) and thus the emitted sound pulse often skims off a surface leading to either no return, or a secondary return giving the depth of an object seen (heard) in a mirror, rather than in the line of sight (hearing), and
>
> • *c.* because of these problems large amounts of processing turn out to be necessary to get even moderately low noise levels in the data, defeating one of the original reasons for choosing sonar sensing, and lastly
>
> • *d.* sonar is not usable over large distances without much higher energy outputs than that of the standard cheap sensors making it much more hardware expensive and also subject to errors due to atmospheric effects.

A final problem with sonar is that it is not usable on the surface of the Moon or Mars nor in space outside of a pressurized vehicle.

For these reasons we prefer to use the natural alternative of passively sensing electromagnetic radiation in the near visible range. It suffers from none of the drawbacks we have listed above, although it does have its own; such as the need for large amounts of computation for even the simplest of data extractions.

We will however consider using sonar for tasks for which it is suited. For example it is an excellent sensor for monitoring local obstacles, missed by the visual sensor, as the robot moves along. Thus it acts as a backup safety sensor only. The Hilaire project [**Giralt et al 1983**] is another project which has come to similar conclusions concerning the proper role of sonar.

▸ **6.** We are interested in building *artificial beings* in the sense described by [**Nilsson 1983**]. A robot purchased from Sears must work reliably for at least many months, and hopefully many years. It must work under the "control" of a completely unskilled, unsophisticated and untrained operator; i.e., the purchaser (you!). It must operate sensibly and safely under a large class of unforeseen circumstances (some quite bizarre). To achieve such performance we may make only a minimal set of assumptions about the environment. The control software must be **reliable** in a sense orders of magnitude more complex and sophisticated than that which is normally considered under the banner of "software reliability". We suspect new hardware architectures will be necessary, besides software developments to meet these challenges.

1.2 *The task*

Our mobile robot is a circular platform with three wheels which always steer together and can point in any direction. It has sonar sensors and stereo TV cameras on board. Its environment is restricted to indoors but we plan to let it wander anywhere within our Lab. We want to use vision to build reliable maps of the world which are suitable for navigating about the world. We plan on using a system of [Khatib 1983] to provide a robust lower level control system avoiding local obstacles through acoustic sensing.

We would like the robot to operate at a speed which makes its actions observable and so that it can interact with a world inhabited by humans in a reasonable manner (e.g. avoiding running into them).

The first, existing, robot will not be able to do anything other than go places and look around. A later robot may have an arm. Such a robot could then do "useful" tasks. Example tasks which might be within the reach of our control technology over the next few years include:

- a. Retrieve soda cans from offices.

- *b.* Pick up the golf balls.

- c. Provide a local package delivery service within the Lab.

- *d.* Operate as a demonstration machine loader in the robotics laboratory.

- e. Vacuum the floors.

- *f.* Retrieve lunch trays and return them to the ground floor cafeteria. (Harder.)

The algorithms described in this paper have not been implemented and therefore have not been tested on real data. In some cases only a computational problem has been formulated and no algorithm is given here.

2. Subsumption as an architectural methodology

Most Artificial Intelligence systems built in laboratories are extremely fragile. They usually take as input a few hundred, or at most a few thousand, ASCII characters and produce some output, again a stream of ASCII characters. Someone who has not worked on developing the program can usually cause it to fail in an embarrassing manner simply by typing at it for a few minutes, unaware of the subtle limitations of the program. Commercial exploitation of expert system technology can be expected to force companies to address these problems and build more robustness into their systems. It is reasonable to expect that the cost will be roughly an order of magnitude increase in complexity (or number or rules) (for instance, the R1 system [**Bachant and McDermott 1984**] has gone from 777 rules to 3303).

Mobile robots have had to be somewhat more robust if for no other reason than the volume of noisy unreproducible data that they must handle even in a single sojourn into their world. However they have usually lived in a restricted world with a well understood class of possible scenes to be encountered. The most robust of the mobile robots [**Moravec 1983**] is not of this type. It moves in a cluttered people inhabited environment (a laboratory). However the task it performs is a fairly low-level one. It avoids obstacles in reaching a goal point, and builds an extremely simple map of the world, useful for only that task, as it goes.

2.1 *Animals and robots*

A USMM home robot purchased from Sears must be robust in a number of senses:

- *a.* It must be able to operate in an unknown environment whose characteristics could not be predicted by its designers

- *b.* It must operate in a changing environment.

- *c.* It must operate safely endangering neither life nor limb of neither itself, humans, or the family dog.

- *d.* It must maintain itself, making sure that it recharges itself regularly no matter what constraints are put on it by its owner.

Clearly humans are able to achieve these goals. In fact in a strong sense all animals are able to achieve these goals. They very nearly define what it means to live and survive. Consider then the difference between the manner of the design and construction of the control structures of most mobile robots, and that of animals. *

▶ **1. Mobile robots.** Typically there are a number of modules or interacting computer programs, each of which has a specific set of tasks or data transforms it must carry out. For instance there might be one which processes sonar data to produce a 360 degree depth map of the current surroundings. Another module controls robot motion and is given heading and distance commands which it turns into motor commands. An intermediate module takes the depth map and plots a local course of headings and distances. If any module fails, or makes an error, or is overloaded by time constraints forcing an early decision, the whole system fails. Robot control systems are like diamonds; a single flaw and the whole thing is ruined.

▶ **2. Animals.** Animal brains, on the other hand, are like balls of mud. They started out as little balls of mud, and over time, through evolution, extra lumps of mud got added to

* These analogies are borrowed (stolen) from Joel Moses who used them to compare LISP and some algorithmic language.

them. If one of those little clumps of mud fails the rest of an animal brain is still a perfectly good ball of mud, just not as big, not quite as capable a ball of mud. As clumps of mud got added over time, some took over the functions of old clumps of mud in cases where they usually performed better than the old. In some cases the old clumps disappeared. In other cases the old clumps of mud remained, continuing to compute almost as before, providing backup in the cases that the new clumps could not handle. Actually in many cases the new clumps did not spring to life fully functional. Instead the new clumps piggy-backed off the old, using results of their partial computations, and providing replacement partial results to other parts of the old clump of mud.*

Evolved systems are terrible messes that happen to work. At first sight there may seem to be nothing about them worth emulating. However a system built by evolution can be extremely robust. This robustness can be manifested in a number of ways:

- *a.* **Adaptability of perception.** Because of the large number of semi-redundant and overlapping modules, the system is adaptable, perceptually at least, to a wide range of environments. Humans and other animals have easily adapted perceptually to a wide range of environments not experienced by any of their ancestors; e.g., space, underwater, modern cities and interiors decorated by modernists and neo-modernists.

- *b.* **Reliability of total system.** An evolutionarily produced system is often extremely reliable, continuing to function well in the face of severe impairment of either sensors, motor control systems, or processing modules. The redundancy of partial functions of processing modules is the main reason for this. Redundant sensors and actuators are also contributing factors.

- *c.* **Buildability.** A correctly "evolved" robot control system with some given level of performance might be more buildable than one built with a more standard approach to operate at the same level of performance. The reason for this is that the system can be tested along the way as just a few modules are integrated, and then incrementally as each new module is added. Each module operates at specification in these incremental tests. The system is tested in a real environment, on real data. As such plenty of test data is available. Conversely in the conventional approach to system building it will more likely be necessary that testing proceeds as follows. Individual modules are first tested on a small number of painfully built data sets, with exhaustive analysis of their results (skipping these steps is dangerous because of the next step). Then a large number of modules are connected together. Usually it is necessary to run them at "half-speed" (i.e. on simplified and contrived

* These meanderings are based on my understanding of evolution pciked up from magazines and popular books over the years. Some people might argue with me.

data sets), in order to debug the interfaces, or because not all modules are yet ready for testing.

2.2 *Subsumption*

In light of the above considerations we will endeavor to build the control structure for our mobile robot, and all the algorithms embedded as modules, using a layered approach we call *subsumption*.

The layers, in a sense, recapitulate evolution. However we wish to avoid the penalties often suffered by systems which have evolved. These include evolutionary dead-ends, sometimes useless redundancies and occasionally grossly suboptimal algorithms.

A subsumption control structure is not heterarchical, nor is it really hierarchical. Instead when a new layer is added to an existing and complete control system a new complete control system results with some specified level of competence, i.e. class of situations under which it can meet some performance index. Over the life of the project many new layers will be added to an initial control system. After any particular layer is added the robot will possess a complete, at some level of competence, and working control system which can be extensively tested in the real world.

In general the performance index specified for successive layers will be harder and the class of situations over which they can meet that index will be neither a subset nor superset of the previous class of situations. A new layer is permitted to make use of modules, or algorithms, which are parts of earlier layers. An early layer may not be modified to make use of modules introduced as part of later layers. When a particular control layer notices that it cannot meet its performance index for the current situation it abdicates control and lets the next lower control level attempt to meet its own performance index.

2.3 *Levels of competence*

Some examples of successively more complex performance indices follow. We will also refer to these as levels of competence, for they define the class of tasks, with corresponding dynamism of environment, which a robot can routinely perform successfully.

- *i.* Wander aimlessly around without hitting things.

- *ii.* "Explore" the world by seeing places in the distance which look reachable and heading for them.

- *iii.* Build a map of the environment and plan routes from one place to another.

- *iv.* Notice changes in the "static" environment.

- *v.* Reason about the world in terms of identifiable objects and perform tasks related to certain objects.

- *vi.* Formulate and execute plans which involve changing the state of the world in some desirable way.

- *vii.* Reason about the behavior of objects in the world and modify plans accordingly.

A housecat might perform at about level *iv* on this scale.

Note that not every one of the above levels of competence relies on the one preceding it and there is more than one ordering of the layers consistent with the highest level of competence.

Is it foolish to attempt to emulate evolution in building a mobile robot control system? It would be if that were our goal. Instead our goal is to build as robust a robot control system as is possible. We must be careful not to fall into traps of cuteness for cuteness sake, but instead to keep in mind the ultimate goal and range of applications intended for the control system.

In section 6 we give a few examples of how the subsumption methodology can lead to robust performance of a mobile robot. In section 7 we give a few ideas on the sort of hardware bases which might adequately support a subsumption architecture, and explain how we will implement a subsumption architecture on standard machines.

2.4 *The problems*

We believe there are four key problems which must be worked on in order to build the type of robot control system we have been describing.

- *1.* Dealing with uncertain sensors and control.

- *2.* Finding an adequate multi-scale representation of space and objects perceived which is stable and provides the types of information needed by the control system over increasing levels of competence.

- *3.* Building a robust vision system which can operate over are large class of scenes.

- *4.* Implementing a subsumption architecture in such a way that synchronization does not become a dominant problem causing a loss in robustnesss.

These topics are covered in detail over the next few sections. Section 3 on uncertainty is more detailed than the other sections because that is the area we have so far spent most time on. The other topics are each probably harder than the uncertainty issue.

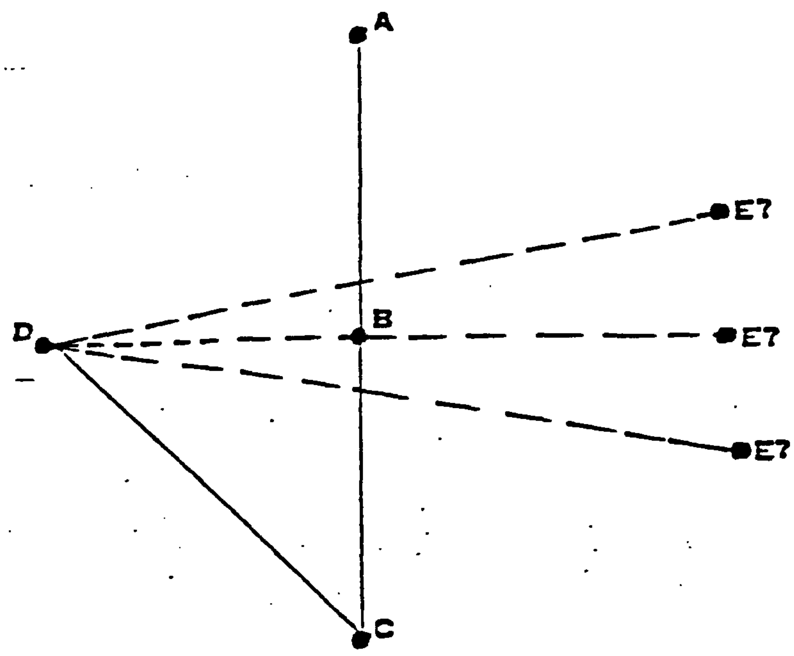

Figure 1. Metric information leads to incorrect representations when the information contains uncertainty.

3. Uncertainty

A visually guided mobile robot inhabits a mental world somewhat different from the real world. Its observations do not exactly match the real world. Its physical actions do not occur exactly as intended. The task of navigating around the world can be eased by having the world map based on primitives suitable for navigation. The map should also be constructible from visual observations. We believe depth estimates are essential to the task of navigation. They can be extracted from stereo pairs of images or from motion stereo. We will concentrate on the former for the purposes of this paper.

Observations of the world are uncertain in two senses, providing two sources of uncertainty. Action in the world is also uncertain leading to a third source of uncertainty.

▸ **1.** There is inherent error due to measuring physical quantities. In particular, a pixel raster as used in computer vision enforces a minimal spatial resolution at the sensor beyond which there is no direct information. Physical constraints, such as continuity of surfaces in the world or averaging an identified event over a number of pixels, can be used to obtain sub-pixel accuracy of certain measurements, but the finite amount of information present ultimately can not be escaped. For the purposes of this paper we will only consider single pixel accuracy at best. For stereo vision this discretizes the possible depth measurements into a small number of possibilities, based on the maximum disparity considered by the algorithm. The error in depth measurements, increases with the distance from the cameras and the possible nominal values become more sparse.

▸ **2.** There may be errors in stereo matching. Thus a depth point may be completely wrong.

▸ **3.** When a robot is commanded to turn or go forward it does not carry out the action completely accurately. If drive shaft encoders are used then the best possible accuracy is to

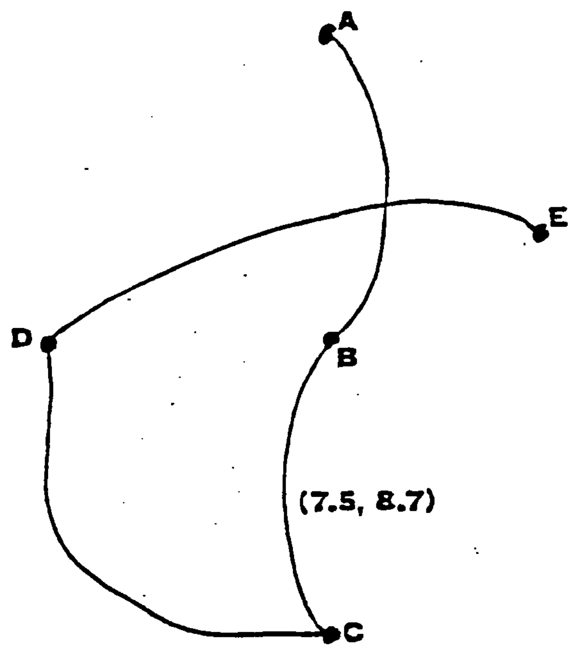

Figure 2. A better representation. The graph is not embedded in the 2-d plane, except for the purpose of drawing this diagram.

within an encoder count. If this were the only source of error it could almost be ignored as shaft encoders can be built with many thousands of marks resulting in very small physical errors. The real source of error is wheel slippage on the ground surface. This can be reduced somewhat with very accurate dynamic models, and accurate knowledge of the ground surface and wheel characteristics. However, large uncertainties usually remain.

Our approach is to take explicit account of the first and third sources of observational uncertainty in the design of all algorithms and to use the principle of least commitment to handle it in a precise and correct manner. The second source is handled more heuristically, by making the algorithms somewhat fault tolerant.

3.1 *A rubbery map*

These ideas can be built into a map representation by avoiding the use of a 2-d coordinate system. Instead, only relationships between parts of the map are stored, in a graph representation. The relationships include estimates on their associated uncertainties.

As an example of how this helps consider figure 1 where we do use a 2-d coordinate system. The figure is one aspect of a map built by a mobile robot as it has moved from some place A, to place B, and so on to place E. It tried to travel in straight lines between places. Straight lines in the map correspond to straight line paths in the world. It did not know the places beforehand but has been labeling them as it goes. Suppose it has been using nominal distances traveled and nominal angles turned, to give nominal 2-d coordinates for A, B, etc. Given that these values include errors there may be three physical positions for E, that give rise to the same measurements mad by the robot in moving from D. Any coordinates chosen for E implicitly add unsupported conclusions to the map. It can not be known whether the path from D to E crossed the path from A to B, went via place B, or crossed the path from B to C.

A better approach then is to use an abstract graph as in figure 2 where the arcs, which represent straight line motions in the real world, are not represented as straight lines in the model, but are simply arcs with labels. For instance, arc BC is labeled with the robot's estimate of the distance it traveled. Intersections of arcs in this representation are purely an artifact of our attempt to draw the graph on paper. It will be possible however, to determine from the arc labels that the path represented by arc DE somewhere crossed the path followed by the robot from A to C.

3.2 *Dealing with Uncertainty*

If a mobile robot is moving in a flat two dimensional world, and if it has a labeled direction as forward then its space of possible locations and orientations is a three dimensional *configuration space* [Lozano-Pérez 1983]. We can label its axes x, y, and θ. When the robot is at two dimensional coordinates (x_0, y_0) with orientation θ_0, its configuration corresponds to point (x_0, y_0, θ_0) in configuration space. For now lets refer to such a configuration as P_0.

Suppose the robot has configuration P_0, and it re-orients by angle η then travels distance d. Its new configuration would be:

$$P_1 = (x_0 + d\cos(\theta_0 + \eta), y_0 + d\sin(\theta_0 + \eta), \theta_0 + \eta).$$

However there is always error associated with the robot's motion. Typical errors might be $\pm 5°$ angular error and $\pm(5 + 0.05d)$ centimeters, in distance error, as a function of the distance traveled.

3.3 *Uncertainty manifolds*

We have shown that P_1 can not be uniquely identified. Instead P_1 can range over an *uncertainty manifold* (see figure 3) in configuration space. If the range of possible values for d is $[d_l, d_h]$ and for η is $[\eta_n - \alpha, \eta_n + \alpha]$ then the uncertainty manifold is:

$$M_1(x_0, y_0, \theta_0) = \tag{1}$$

$$\{\,(x_0 + d\cos(\theta_0 + \eta), y_0 + d\sin(\theta_0 + \eta), \theta_0 + \eta) \;|\; d \in [d_l, d_h], \eta \in [\eta_n - \alpha, \eta_n + \alpha]\,\}.$$

Notice that this is a two dimensional manifold in three dimensional space.

A simple question, but nevertheless a useful one to ask while exploring the world, is "Am I back some place I've already been?", which, without loss of generality, can be simplified to "Am I back where I started?". Given that each individual motion is uncertain, it will be necessary to take into account the cumulative uncertainty, and in fact, without further sensing, the toughest question that can be answered is "Is it plausible that I am back where I started?". We will call this the *am-I-there-yet* question.

The uncertainty manifold resulting from two motions can be written as

$$C_{12}(x_0, y_0, \theta_0) = \bigcup_{(x,y,\theta) \in M_1(x_0, y_0, \theta_0)} M_2(x, y, \theta) \tag{2}$$

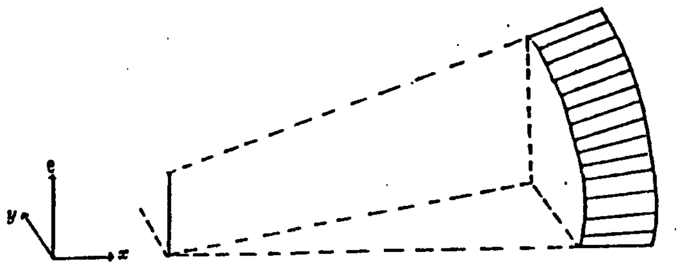

Figure 3. An uncertainty manifold arising from an uncertain motion in an uncertain direction.

and from two motions:

$$C_{123}(x_0, y_0, \theta_0) = \bigcup_{(x,y,\theta)\in C_{12}(x_0,y_0,\theta_0)} M_3(x, y, \theta)$$

(Note that this is different from an incorrect formulation given in [Brooks 1984].) We call the manifolds C_{12}, C_{123}, etc., "cascaded manifolds". These manifolds are three dimensional and "solid"; i.e., they have a non-empty interior. The surfaces of these manifolds become progressively harder to express as the number of motions increase.

One approach to answering the am-I-there-yet question is to introduce three more variables for each motion (besides nominal angles and distances and bounds on errors in each), and write explicit conjunctions of symbolic inequalities of the form

$$P_1 \in M_1(x_0, y_0, \theta_0),$$

$$P_2 \in C_{12}(x_0, y_0, \theta_0),$$

etc. Note that each P_i introduces three more variables. Then we can add the constraints

$$(x_0, y_0, \theta_0) \in C_{12\ldots n}(x_0, y_0, \theta_0), \tag{3}$$

and, using the methods of [Brooks 1983] ask whether all the constraints are together satisfiable. This is *forward reasoning* with constraints. Additionally one would like to be able to use auxiliary information, such as from landmarks, to be able to assert inequalities such as (3). Then one would use the symbolic bounding algorithms from the above paper to determine the implications in terms of constraints on the actual physical values of angles of re-orientation and distances traveled. This is *backward reasoning* with constraints. Unfortunately this approach doesn't work well because of the presence of so many trigonometric terms.

3.4 *Approximating uncertainty manifolds*

A second approach, used more successfully in reasoning about uncertainties in assembly processes, is to use bounds on trigonometric functions, such as

$$\sin\alpha \leq \sin(\eta - \eta_n) \leq \eta - \eta_n$$

for $\eta \geq \eta_n$ and

$$1 - \frac{1}{2}(\eta - \eta_n)^2 \leq \cos(\eta - \eta_n) \leq 1$$

This has the effect of making an individual 2-d manifold M_i a little fuzzy, giving it some three dimensional volume. This makes the resulting manifolds (e.g. equation (2)) a little nicer in form. Unfortunately, again the constraint propagation methods fail because of the large number of cascading variables.

Broader bounding volumes for the uncertainty manifolds, with fewer parameters, and with simpler interactions, are needed if we are to make use of them in either forward or backward reasoning.

3.5 *Cylinders in configuration space*

The projection of the uncertainty manifold (1) into the x–y plane is can be bounded in the x–y plane by a circle with radius

$$\frac{(d_l + d_h)^2}{4\cos^2\alpha} - d_l d_h$$

centered a distance

$$\frac{d_l + d_h}{2\cos\alpha}$$

from the start of the motion. We can then bound the complete manifold in configuration space by a cylinder sitting above the bounding circle with have constructed, and ranging from $\eta_n - \alpha$ to $\eta_n + \alpha$ in height.

A bounding cylinder B has four parameters and is a function of three variables (just as a manifold M_i itself is a function of position and orientation of its root point in configuration space). The four parameters are distance d, radius r, central orientation η, and orientation radius α.

Suppose that $B_{d_1,r_1,\eta_1,\alpha_1}(x,y,\theta)$ bounds $M_1(x,y,\theta)$, and that $B_{d_2,r_2,\eta_2,\alpha_2}(x,y,\theta)$ bounds $M_2(x,y,\theta)$. Then it turns out we can bound the cascaded manifold

$$C_{12}(x_0, y_0, \theta_0)$$

by the bounding cylinder:

$$B_{d_{12},r_{12},\eta_{12},\alpha_{12}}(x_0, y_0, \alpha_0)$$

where

$$d_{12} = \sqrt{{d_1}^2 + {d_2}^2\cos^2\alpha_2 + 2d_1d_2\cos\eta_2\cos\alpha_2}$$

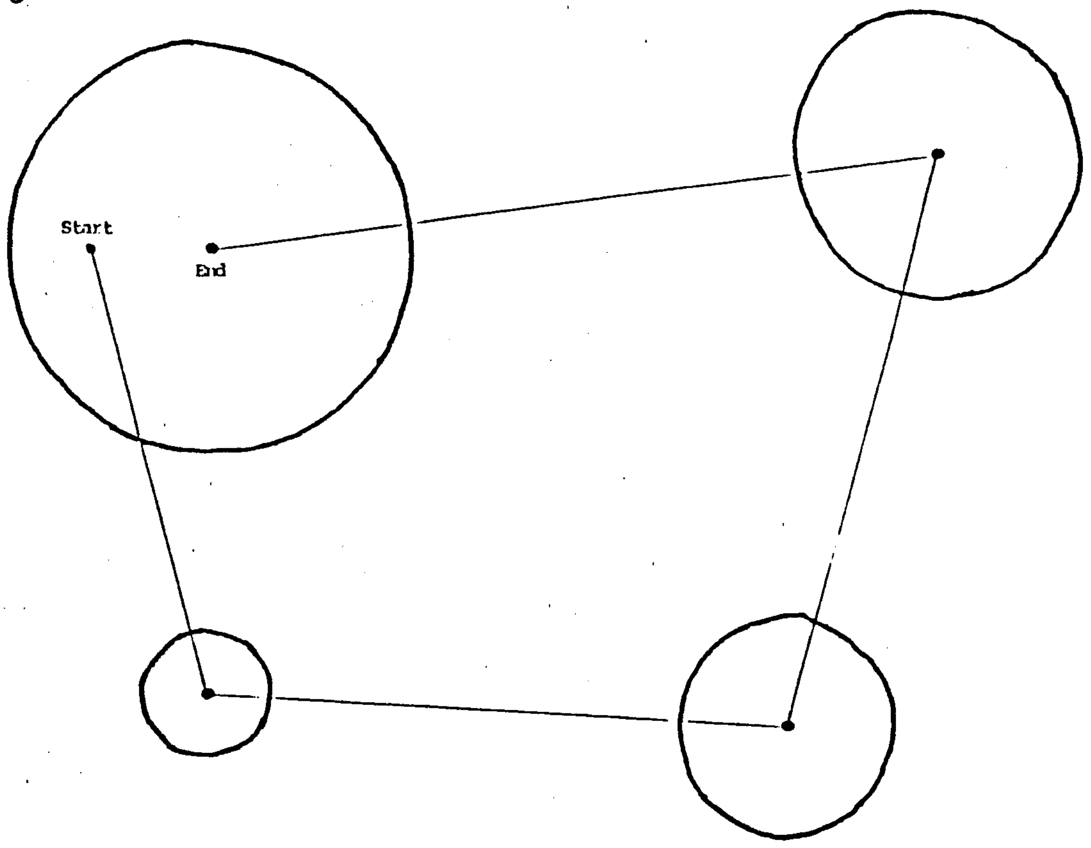

Figure 4. An example of forward reasoning, deciding that it is plausible that the robot is back where it started.

$$r_{12} = d_2 \sin \alpha_2 + r_1 + r_2$$

$$\eta_{12} = \eta_1 + \eta_2$$

$$\alpha_{12} = \alpha_1 + \alpha_2$$

Now we are able to cascade many motions together without increasing the complexity of our representation for the amount of uncertainty which results. The penalty we pay is that our bounds are sometimes rather generous.*

Figure 4 shows the projection of bounds on uncertainty manifolds produced by four successive motions into the x–y plane. After the last, the am-I-there-yet question has an affirmative answer, i.e. "maybe", relative to the point of origin.

In figure 5 we see an example of backward reasoning. Suppose we have used forward reasoning to determine that it is plausible that the robot is in the same convex region from which the journey commenced. It might be plausible that the robot is in other regions too.

* [Taylor 1985] has suggested that we should produce an analytic expression for C_{12} and then build a tighter bounding cylinder for it directly. This is an idea worth pursuing. By the time one tries the same trick for C_{123} however it is clear that the expressions will be too complex to have much hope of succeeding. For more than two motions we then need to use the bounding approximations we have presented here.

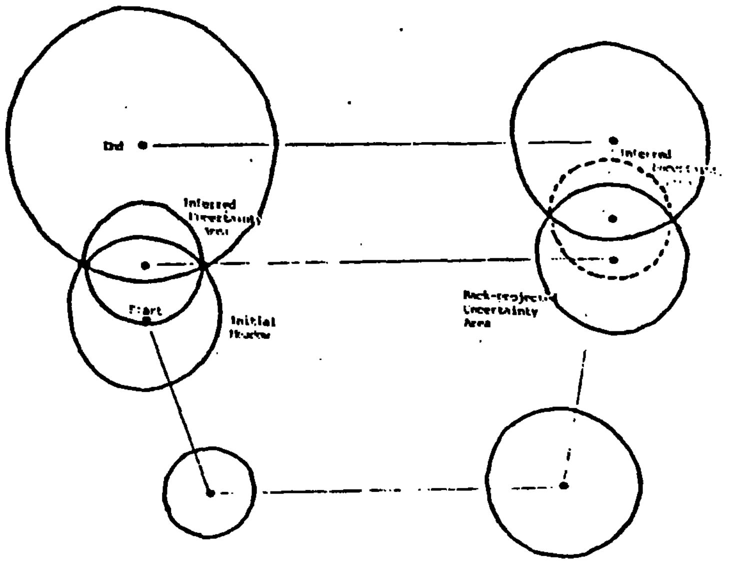

Figure 5. Backward reasoning. Additional information indicated the robot was in the region. The dashed circle shows bounds where the robot really must have been before.

In any case the plausibility condition cuts down the number of possible regionss the robot might be in. If there are some visual landmarks associated with the region then perhaps one of the hypotheses can be verified (if a similar landmark is visible in another region then the forward reasoning will sometimes provide the disambiguation). So suppose such reasoning does indeed determine that the robot is in the original region. The intersection of the uncertainty cylinder cross section and the region determines a smaller region where the robot really might be. We bound that with a circle to produce a new uncertainty cylinder. If orientation information was gleaned from the landmark observation then that can be used to cut down the cylinder's range in the θ direction. This projection of this cylinder is illustrated in figure 5.

Now we can use backward reasoning, and ask "What are all the points I could have been at, before the last motion and gotten into this new uncertainty cylinder?". This does not involve shrinking the cylinder by subtracting the uncertainty cylinder of the last motion, rather it involves cascading its inversion through the origin of configuration space. The resulting uncertainty cylinder, is shown in figure 5. Since the robot got to the third observation point by traveling forward around the path, it must have been in the intersection of the two cylinders before the fourth motion. Now a new uncertainty cylinder can be constructed to bound that intersection. It specifies an uncertainty relationship between the start of the journey, and any observations which were made after the third stop. Thus it ties down some of the rubbery map just a little bit better, and the effects of this additional constraint will be able to be used in later references to the map. Notice that the backward reasoning can continue from the last uncertainty cylinder we constructed. Soon however the uncertainties will become so large that no new information will be gained.

This technique was independently developed under another name by [**Chatila and Laumond 1985**]. They call it *fading*.

4. Representing space.

The key maxim to keep in mind when designing a representation of space and objects (or a representation of any "knowledge" structure) is:

A representation is not a model which is analogous to what is being represented. Rather it is a data structure containing as many facts as possible that are reasonable to infer from the perceptual observations, and nothing else.

There are a number of things we require of our representation(s) of space:

- a. Incorporation of uncertainty information.

- *b.* Stability of representation given small differences in what is perceived.

- c. Multiple scales of representation. E.g., the robot should be able to plan to go from A, to room B, to room C, without having to worry about the detailed plans for crossing through room C.

- *d.* Multiple aspects of shape descriptions. Sometimes a corridor is a space to roll down in order to get somewhere. Other times it is a space where objects are stacked against the walls, or where the robot simply wants to cross to get from one room to another (e.g., in an office building).

- e. It must be able to support operation in a true three dimensional world, where objects have heights and where spaces can be bounded from above.

We take these requirements to suggest the following properties of a spatial representation:

- a. There can be no global coordinate system. Since all observations are subject to error, trying to keep things in a global coordinate system will result in either enormously large, and hence useless, error bound on the position of features within the representations or eventual inconsistencies within the representation.

• *b.* There can not be a unique representation for each point in space. Uncertainties in observations dictate that it will not always be possible to identify a point observed in space as one which has been seen before. As a corollary, in no way can the representation of the world be "exact".

• *c.* There must be a hierarchy within the representation so that, for instance, rooms can be represented and also the free space within them.

• *d.* The perception modules must build up descriptions of what is seen by having multiple parallel descriptions capturing the different aspects which are required. In addition there will be different local clusterings of primitives to handle different aspects of some portion of space. This means that, unlike other systems, it is not necessary in this representation to derive all plausible semantic attachment from the most primitive, and usually lone, observational primitive.

• *e.* The representation of space and objects must include, at least, bounds on their extent in the vertical direction.

From the above properties of the representation we choose the following corresponding options in designing a representation to meet our requirements:

• a. Everything in a our maps will have position and orientation only relative to other items in the maps.

• *b.* The spatial primitives in our map will be allowed to overlap. In fact they will be encouraged to overlap. The primitives will have broad properties only (say two or three parameters at most). More detailed representation will require moving down the representation hierarchy to a larger collection of similarly vague primitives.

• c. There will be markers for containment and overlap within the map representation. Collections of primitives within the map can have associated with them a finer grain alternate representation, which inherits the appropriate containment and overlap properties.

• *d.* Modules built very early in the project will describe free space in terms of *visibility channels* which capture regions of navigability. Later modules will capture other aspects, such as convex open spaces, height of barriers, horizontal surfaces and their heights, "clutter"-type objects, etc.

• *e.* For modules built early in the project it may suffice to represent objects and spaces as prisms in the vertical direction. Locally the world can look like a stack of two dimensional projects. Visibility channels are a superset of prisms.

[**Brooks 1983a**] introduced a new representation for free space, useful for solving the find-path problem for a convex polygon. (See [**Brooks 1983b**] for an extension to the case of a manipulator with resolute joints.) The key idea is to represent free space as *freeways*, elongated regions of free space which naturally describe a large class of collision-free straight line motions of the object to be moved. Additionally there is a simple computation for determining the legal orientations (i.e. those where the moving object stays completely within the freeway) of the object while it is moved along a freeway.

5. Vision.

In deciding on techniques to use for vision on the mobile robot we need to keep in mind the following constraints and influences:

- a. We would like the robot to operate in "real time". If every image is processed by a time consuming algorithm this will not be possible. So perhaps some images will should only be processed by fast algorithms, in order to close a control loop including vision, while others are processed by more computation intensive algorithms in a more background mode.

- *b.* We need to examine the use we intend to make of the output of vision algorithms in order to select the correct algorithm. I.e., the vision of a mobile robot should be goal directed. Initially at least we wish our robot to explore its environment and to build maps of it.

- *c.* A large number of early vision techniques have been developed over the last few years (see [**Brady 1982**] for a survey of some). We should make use of what is available rather than spending time inventing new early vision techniques. We have plenty of other things to keep us busy.

- *d.* No early vision technique is either completely adequate in a single situation, nor moderately adequate over all situations.

Thus we should be content to have many early vision modules running, to use the results of some directly, and to build some intermediate modules which combine evidence from multiple early vision modules.

It is by no means clear to this author that all the early vision modules need ever have their representations combined into some glorious unified model of the world. Instead the approach we seem to be falling into, as a result of the constraints on the problem we are trying to solve, is that various early vision processes inject their results into various control loops at different places.

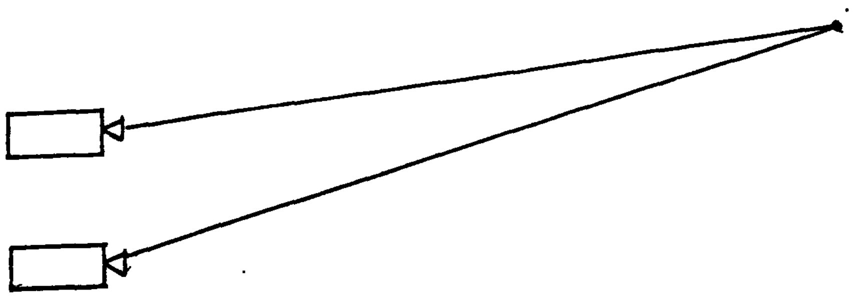

Figure 6. Observation of a point implies that on lines between it and each camera there are no obstacles.

5.1 *Empty vision.*

Suppose a point in the distance is visible in both images of a stereo pair. Then we can draw lines from the center of focus of each camera to that point in three space and known that there are no obstacles intersecting either of the lines (see figure 6 which shows a projection of two such line into the ground plane). We call this the *visibility constraint*–there is nothing blocking an undisturbed line of sight (water, of course, violates this constraint but we will worry about such things later or when the robot is observed going for its first swim).

Now suppose there are some other visible points nearby. We can draw more lines to them. Soon we get a mass of lines which are not blocked by obstacles. See figure 7. Analytically we can compute an upper bound on the possible size of an obstacle in the three dimensional convex hull of all these lines. We call that convex hull a *visibility channel.*

We propose using an interest operator and the multiscale stereo correlation algorithm of [Moravec 1979] to find points in the world. When a clusters if found (the details of how many points and what density is needed remain to be worked out) we can construct a visibility channel. Such a channel provides a potential path for the robot, with bounds on the size of unseen obstacles. The robot can move down such a path checking acoustically for obstacles. Like humans it might be fooled by piano wire strung up in its way, but such obstacles have low probability of existing. Thus the robot is able to find freeways of navigability with a very cheap image analysis algorithm. (We expect to take on the order of one or two seconds on the CADR, with hardware convolver, to select a new visibility channel to explore.)

5.2 *Better depth maps.*

To build reliable maps of the world we will need to do more vision than simply find a few visible points stereoscopically. We need to be able to find surfaces and edges at the very least. We expect to adapt the work of [Grimson 1981].

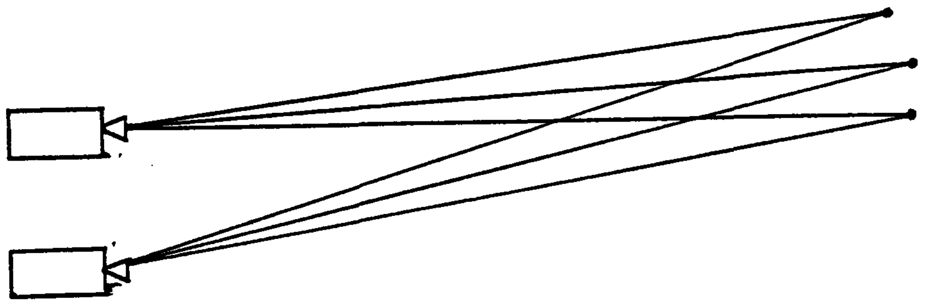

Figure 7. When we see multiple points nearby we can lower an upper bound estimate on the size of an obstacle between the robot and the observed points.

5.3 *Landmark recognition.*

As we move to doing landmark selection and identification, in order to reduce accumulated uncertainties, we will need to consider shape descriptions.

6. How subsumption fits in.

Our subsumption architecture has two major classes of components: *modules* and *nodes.*

- *Modules.* These have inputs and outputs, and are the core processing elements. Typical modules carry out tasks like, plan a local move to avoid known local obstacles, transform point depths into visibility channels, etc.

- *Nodes.* These control the combination of two or more streams of data into a single stream.

In particular we will use *suppressor nodes* and *union nodes.*

A suppressor node takes two or more input streams. The input streams are labeled with a priority. The larger the number, the higher the priority. A suppressor node acts as a switch. If more than one input stream provides data it only lets through the one on the higher priority input. If only one input stream provides data it gets straight through. See figure 8.

A union node assumes there will be sets coming in on its input lines, and if more than one ever arrives it takes the union of the sets and sends them along the output line.

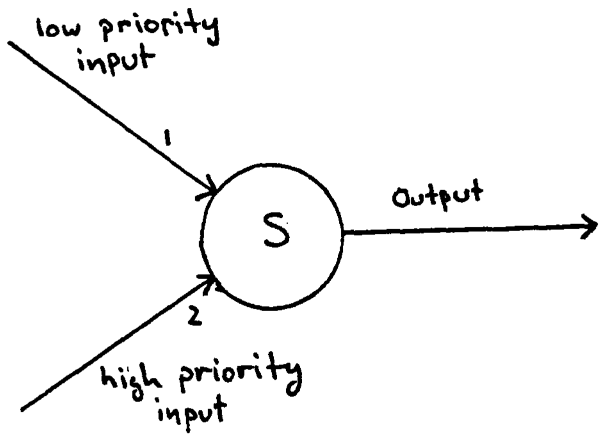

Figure 8. A suppressor node takes one or more input lines and always lets the highest priority line send on its contents to the output line.

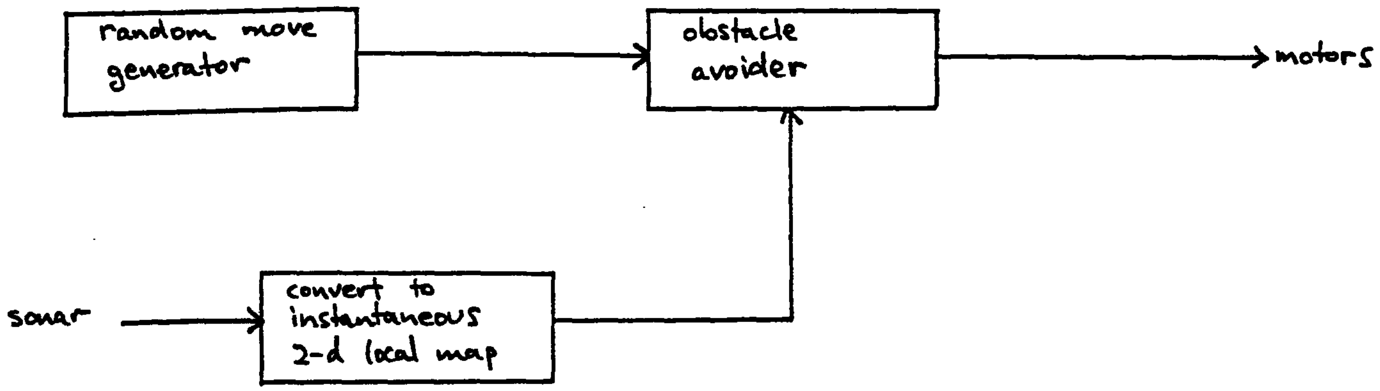

Figure 9. A "wiring diagram" for achieving level *i* competence.

6.1 *An example.*

A control system to achieve level *i* competence is shown in figure 9. Suppose now we want to add a vision module so that we can get to level two competence.

We first build a [Moravec 1979] stereo system. We notice that it provides obstacle information so we union that with the acoustic obstacles to get more robust behavior. We also route the output of the stereo module to a channel selector module. It provides a new source of motion, which are used to suppress those provided by the random move generator: see figure 10. Thus the vision system finds a place in the distance to head towards it will, otherwise the random behavior of level *i* competence will take over and the robot will wander around until it chances to a vantage point where it sees something interesting.

6.2 *Obstacle avoidance.*

In this section we give an example of a detailed analysis of the best way to add in a new module to an existing control system.

Consider the example above of the the motion control system which avoids local obstacles detected acoustically. Such a module would be a key part of achieving level *i* competence defined above. Suppose we have such a module which uses potential fields to model the environment [Khatib 1983]. It models obstacles detected with its sonar as repulsive forces and the goal as an attractive force. It then commands the robot to move in the direction defined by the sum of the forces acting upon it. The repulsive forces increase dramatically very close to the surface of an obstacle so that no matter how attractive is the goal it can not force the robot to crash into an obstacle trying to get to it. We build such a module and debug it thoroughly over many hours of running the mobile robot in a real environment. * Call this module, module A.

Now we add a module, call it module B, to the system that plans docking manouevers of the robot with some workstation (or recharging station). If module B were simply to plan a series of motions, with the final goal point such that the robot would be docked, and feed them to module A then the docking would never take place! As the robot approached the docking station its sonar sensors would detect it and it would become a repulsive obstacle. There are a number of ways we might approach this problem:

- a. We could modify module A so that module B could feed it the coordinates and shape of the workstation, as well as the sonar data. When assigning repulsive forces to obstacles it could explicitly check whether each piece of data (or each piece of some indexed subset perhaps) corresponds to a workstation and if so suppress the creation of a repulsive force for it.

- *b.* Module B could be inserted on the input side of module A intercepting the sonar data before it arrived at module A. It would check for sonar returns corresponding to the expected position of the workstation and suppress them from reaching module A.

- *c.* Module A has to add the contributions of all observed obstacles in order to compute the resultant force. Instead of initializing the summation variable to zero we could modify module A so that it accepted an initial value from module B. Then module B can compute the

* Actually we notice a bug. The robot can get trapped in local energy minima. This bug needs to be fixed by a more competent global planner. The details of interfacing the higher level module are similar to the modification we are illustrating above so are omitted here.

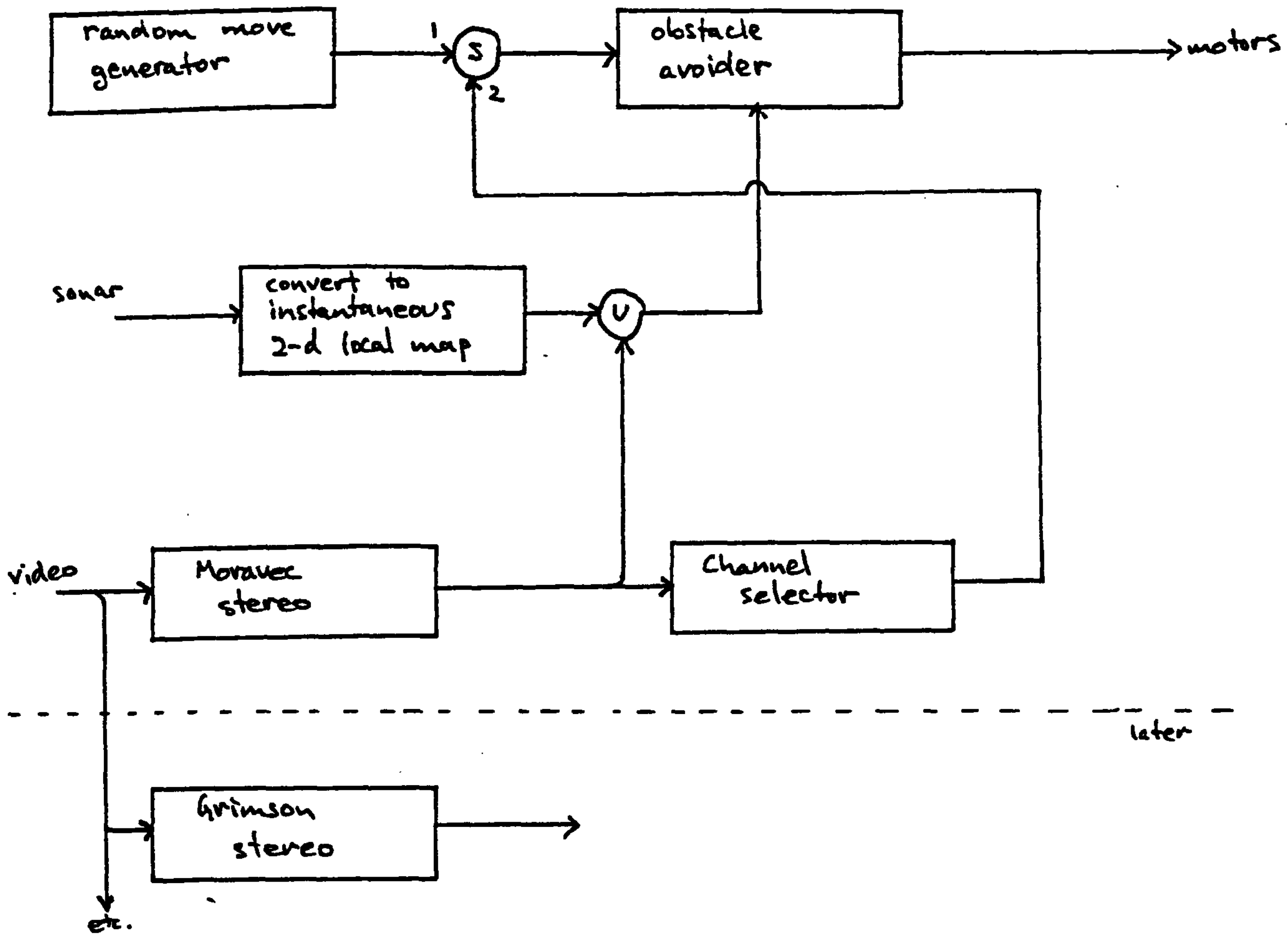

Figure 10. A "wiring diagram" for achieving level *ii* competence.

negative of the contribution expected to be made by the workstation and send that to module A.

• *d.* A combination of two simple strategies could be used. Firstly, to get the robot almost docked we could modify module A so that it can receive from module B a magnifier for the attractive force of the goal point. Module B then feeds module A a series of goal points with ever increasing magnifiers for their attractive forces. Module A will send the robot towards these goal points while, in light of the higher attractive forces, decreasing the safety margin it leaves around obstacles, in particular the workstation. The strategy will eventually fail, without extremely high attractive forces. So at some point module B should suppress the output of module A and replace it with its own motor commands. Thus module B eventually takes over the role of module A; this is an instance of what we call *subsumption.* In this case module B does no checking for collisions. It only takes over A's role when the robot is far enough into the workstation docking adaptor that the rest of the motion can be achieved with no fear of external obstacles.

The first two of these approaches do not fall within the subsumption methodology at all. The third is probably acceptable, but by far the better method is the fourth. Consider the following analyses:

- a. There are two main drawbacks to this method. First, we must modify module A giving it a new input. Second, we change the internal behavior of module A making it test extra conditions as it computes, and giving it an extra internal control path. Thus we are essentially rebuilding this module, making it more complex, and forcing a whole new testing program. The increased complexity might eventually impact the capability of the module for maintaining required real-time data rates. In the conventional approach to system building this might be the best option, however. It keeps the number of interactions between modules small.

- *b*. In inserting part of module B into the input line to module A we have decreased the robustness of the system. Now module A always sees it data through module B and is subject to any slowdowns that B might incur under cluttered environments. Furthermore even level i competence is now endangered by failure of module B even though that module does not contribute to achieving that level of competence.

- *c*. Here we need make only a very simple change to module A where module B can provide a different initial value for one variable if it wishes. This does not add extra levels of processing, nor does it cause A to fail when B becomes overloaded or fails. It does have two slightly undesirable features however. First, the force summation computation is now duplicated in the new module as it must compute an expected resultant repulsive force generated by the workstation. Second, to do this it needs a fairly accurate geometric model of the workstation and an understanding of how its shape will translate into sonar returns. This, of course, was also a problem with each of methods a and b above.

- *d*. This method has the advantages of method c, but not the disadvantages. It provides a very simple extra input channel to module A which may turn out to be very useful for other modules to use (e.g. "cut corners when fleeing from an aggressor"). It neither slows down the usual mode of operation of module A, nor does it make it dependent on the functioning of module B. Furthermore module B does not need to do any computations involving the image of the workstation under sonar sensing.

7. Hardware support and software simulation.

Suppose a single processor is used to control a mobile robot. As the level of competence of the robot is increased it must do more processing per unit time, if it is to maintain a constant rate of interaction with the environment. This suggests that the single processor must eventually be made faster when it becomes saturated.

But consider our block diagrams from figures 9 and 10. The modules are relatively loosely coupled. If each was a single processor they would not need high bandwidth communications. Furthermore as the level of competence of the robot was increased more processors would be added, maintaining the response times established at the lower levels of competence.

7.1 *A hardware implementation.*

There are two obvious strategies in implementing the subsumption control system in hardware.

▸ **1.** We could obtain an existing multiprocessor. For example a BBN butterfly, or IBM's RP3. Both provided shared memory over all processors using a large switch to hide the details from the user. The former uses Motorola 68000s as its processing nodes, while the processors for the latter have been selected but the information has not been made public. The BBN multiprocessor already works and there are a number of them, with varying numbers of processor nodes, installed in universities around the country. The IBM machine is still in the design stage. There are a number of other such projects at various companies mostly based on 68000s or National *N*32032s.

The idea would be to dedicate individual processors to one or more software modules (some modules are quite small and could happily co-exist on processors with the power of a 68000 each running at "realtime" speed. Suppressor nodes and communication links would all be implemented in software. It is unlikely that any more than a small portion of the switch capacity would be needed for communication between processes. [**Kanayma 1983**] has come to similar conclusions. That means we would be paying for a large piece of hardware that would be underutilized. The flexibility it gives however makes the approach an attractive option.

▸ **2.** We could choose some type of microprocessor, preferably a single board machine, and put mount many of them in a rack. (Perhaps more than one processor type would be used: lower cost processors for the simpler modules.) We could build suppressor and union node boards; probably based on a very simple microprocessor. Then we could use serial links of moderate throughput to connect the processors and suppressor nodes together.*

* One can even imagine having a big patch panel where you "wire" up a "brain". Maybe Frank Herbert was right...

7.2 *A software simulation.*

Rather than go out and buy or build the necessary hardware to implement the robot control system described above, we intend to simulate it in software on a uniprocessor for a while at least. The main drawback to this scheme is the slowdown we will achieve as we add more modules to our system.

We will use a dedicated CADR ([**Moon, Stallman and Weinreb 1985**]). It includes a frame buffer and a convolver suitable for convolutions of separable masks. We have developed a scheduling system which has the following capabilities:

• a. Background processes can be scheduled to run at any given frequency subject to availability of the processor. These processes are not modules, but the glue that holds input and output together in the simulation of the multiprocessor system. For instance one background process will request sonar readings from the robot at regular intervals (probably around once per second). Another updates the performance display described below every 5 seconds. Another will be added when we have the TV cameras and transmitter working to request a stereo pair of images. Background processes put pieces of data in input buffers of modules.

• *b.* Modules can be declared. They are defined by a name and an entry point (a function of no arguments), and a set of input and output buffers. The buffers correspond to the "wires" in the diagrams of figures 9 an 10. Each module is allocated storage in a private region of memory, for performance evaluation purposes.

• *c.* A scheduler tries to select and run a module whenever the machine is not busy with background processes. Each module has a dynamic priority. The module with the highest priority is always run. Priorities get increased whenever an one of a modules input buffers has something placed in it.

• *d.* There are a variety of buffer types. Some queue inputs and others only maintain the most recently received data.

• *e.* There are software suppressor and union node simulators, implemented as procedural attachments to buffers, which manage the flow of data between buffers.

• *f.* There is a graphics performance monitor. It instruments the modules and display their storage consumption, the amount of runtime they are using and how often they are getting run. Other performance characteristics can be easily added to the display if we decide they are desirable. There are two reasons for having the performance monitor. First it will help us debug and modify an essentially real time control system. Secondly it sizes the relative computational demands of the various modules providing design data for a later hardware implementation.

8. Conclusion

We have described a mobile robot project which is grounded on a number of assumptions and methodologies quite different to those adopted by other mobile robot projects. There are a large number of details remaining to be worked out. Many of these should wait until we have some practical experience with our first mobile robot.

References

[**Bachant and McDermott 1984**] Judith Bachant and John McDermott, *R1 Revisited: Four Years in the Trenches* in **The AI Magazine**, Vol 5, No. 3, 21-32.

[**Brady 1982**] J. Michael Brady, *Computational Models of Image Understanding* in **Computing Surveys.**

[**Brooks 1982**] Rodney A. Brooks, *Symbolic Error Analysis and Robot Planning* in **International Journal of Robotics Research**, vol 1, no. 4, 29-68.

[**Brooks 1983a**] Rodney A. Brooks, *Solving the Find-Path Problem by Good Representation of Free Space* in **IEEE Systems, Man and Cybernetics**, SMC-13, 190-197.

[**Brooks 1983b**] Rodney A. Brooks, *Planning Collision Free Motions for Pick and Place Operations* in **International Journal of Robotics Research**, vol 2, no. 4, 19-44.

[**Brooks 1984**] Rodney A. Brooks, *Aspects of Mobile Robot Visual Map Making* in **Preprints of the Second International Symposium of Robotics Research**, Kyoto, 287-293.

[**Chatila and Laumond 1985**] Raja Chatila and Jean-Paul Laumond, *Position References and Consistent World Modeling for Mobile Robots* in these proceedings.

[**Giralt et al 1983**] Georges Giralt, Raja Chatila and Marc Vaisset, *An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots* in **Proceedings of the First International Symposium on Robotics Research**, Bretton Woods, New Hampshire, published by MIT press 191-224.

[**Grimson 1981**] W. Eric L. Grimson, **From Images to Surfaces: A Computational Study of the Human Early Visual System**, MIT Press.

[**Kanayama 1983**] Yukata Kanayama, *Concurrent Programming of Intelligent Robots* in **Proceedings IJCAI-83**, Karlsruhe, West Germany, 834–837.

[**Khatib 1983**] Oussama Khatib, *Dynamic Control of Manipulators in Operational Space* in **Sixth IFTOMM Congress on the Theory of Machines and Mechanisms**, New Delhi.

§

[**Laumond 1983**] Jean-Paul Laumond, *Model Structuring and Concept Recognition: Two Aspects of Learning for a Mobile Robot* in **Proceedings IJCAI-83**, Karlsruhe, West Germany, 839–841.

[**Lozano-Pérez 1983**] Tomás Lozano-Pérez, *Spatial Planning: A Configuration Space Approach* in **IEEE Transactions on Computers**, (C-32):108–120.

[**Moon, Stallman and Weinreb 1984**] David Moon, Richard M. Stallman and Daniel Weinreb, **Lisp Machine Manual**, MIT AI Lab.

[**Moravec 1979**] Hans P. Moravec, *Visual Mapping by a Robot Rover* in **Proceedings IJCAI-79**, Tokyo, Japan, 598–600.

[**Moravec 1983**] Hans P. Moravec, *The Stanford Cart and the CMU Rover* in **Proceedings of the IEEE**, (71)872–884.

[**Nilsson 1983**] Nils J. Nilsson, *Artificial Intelligence Prepares for 2001* in **The AI Magazine**, Vol 4. No. 4, 7-14.

[**Taylor 1985**] Russell H. Taylor, *personal communication,* January 28.