SPACE MAPPING AND NAVIGATION FOR A BEHAVIOUR-BASED ROBOT

(APPLICATION TO MOBILE ROBOT)

THESIS

Presented at the faculty of science, to obtain the grade of doctor in science, by

Yoel Gat

Jury

Jean-Pierre Müller

Daniel Nicoud

Marie-Claude Thomas

Pierre-Jean Erard

UNIVERSITÉ de NEUCHATEL

Faculté des Sciences

Institut d'Informatique et Intelligence Artificielle

rue Emile-Argand 11
CH 2007 Neuchâtel

Abstract

This work is dedicated to mobile robot navigation. It presents a solution to the navigation problem using a simple world model based on a new approach to behaviours' mapping. This new theory of behaviours' mapping proposes a general solution not only to mobile robots' navigation but to the control of a wide range of autonomous (intelligent) behaviour based systems.

Based on the phenomena of bifurcation and non-continuity of a behaviour or a group of behaviours that consist an autonomous behaviour based system, the new behaviours' space mapping theory enables direct mapping of the system's existing space to a graph. The latter mapping creates a world model that is made exclusively in terms of behaviours.

We believe that an intelligent system require a world model to plan and to predict the result of its actions. We also believe that behaviours' based systems cope more successfully with the dynamics and changes of the real world. Combining the two poses a real problem in the creation of autonomous intelligent systems. The world modelling in behaviours' terms permits the cohabitation of the classic AI planning and the behaviours' based execution and control.

A new approach to navigation that demonstrates the feasibility and efficiency of the latter theory was developed. This navigation system takes advantage of the well known and developed Voronoi Diagram and combines it with a simple and robust behaviour that follows the centre of the free space to realise an efficient and robust navigation system for a mobile robot.

A new technique of optimal planning within the graph frame exploits the classic AI side of the behaviours' space mapping theory. The near optimal path finder finds a path close to the optimal theoretic path without exact geometric calculations and optimisation using information that attribute to the graph nodes and arcs.

To extract the skeleton of the free space which is used to create the Voronoi diagram, an improvement to Rosenfeld Kak (RK) and Wang Zhang (WZ) thinning algorithms was developed.

To conclude, this works covers the problem of autonomous robot navigation from the presentation of a new behaviours space mapping theory and a new approach toward navigation, to the development of optimisation technique, skeltonising algorithm and basic behaviour. This work proves that the two rival approaches, the classic and the behaviour based AI can cohabit and even contribute each to its rival in order to create more successful and intelligent systems.

Thanks

I am using this opportunity to thank here some of the people that make all this possible.

First I want to thank professor Jean-Pierre Müller my research and thesis director, for the opportunity to make this research, for the guide, help and long and fruitful discussions that were so essential for my work. Especially I want to thank him for the patience and acceptance of my some times quite irregular and odd method of work. Without him all this would not be possible.

I want to thank professor Hans-Heinrich Nagli the Dean of the Faculty of Sciences and professor Pierre-Jean Erard the head of the institute.

Special thanks to Miguel Rodriguez with whom I shared for the last four years not only the same room at the University but many thoughts and ideas as well. His help in the difficult moments, the long discussions we had and encouragement he gave contributed a lot to the success of my work.

Thanks to all the stuff of I^3A

Thanks to my family, especially my children who paid a lot for this adventure. For them I do hope that it was worth it.

Neuchâtel, July 21 1994

Contents

Figures

1. Introduction

Everything should be made as simple as possible,
but not simpler

- Albert Einstein

1. Introduction

This work is dedicated to mobile robot navigation. Trying to comply with the citation above it presents a solution to the navigation problem using a simple world model based on a new approach to behaviours' mapping.

One can look at the navigation problem as a representative problem for autonomous (intelligent) systems. In navigation an autonomous mobile robot tackles a wide range of problems arising from the operation in a real or almost real world.

The classic approach defines the term *navigation* by three questions that should be answered: [Lev-Law 90]

1. Where am I ?
2. Where are other places relative to me ?
3. How do I get there (to the other places) ?

The answer to the first question can be a geometric position, a sensorial state or it can be defined in symbolic terms. In some systems and approaches this question is omitted on the irrelevance basis, as is the case in pure reactive systems. The answer to the second question is given in geometric terms or topological terms (graph) in the classical systems. In the reactive and behaviour based systems the answer is combined with the answer to the third question and is given in terms of actions. The answer to the third question is planning.

For many years the approach and the methods that control the Artificial Intelligent community were the general problem solver or alike. That approach gained control from the Dartmouth conference in summer 1956 [McCor 79]. Among the participants we find some of the greatest names in the field like John McCarthy, Marvin Minsky, Clod Shannon, Newel, Simon and others. They laid the base to the new science for which McCarthy gave the name Artificial Intelligence. They have seen the problems to be tackled as formal and logical. In fact all the AI community followed them by developing systems that play chess, prove mathematical laws based on axioms and rules or play in the well-defined world of blocks.

Finally as Brooks claimed at IJCAI in 1992 the Classical AI became a problem of search and not of intelligence [Broo 91]. When it comes to Autonomous Systems that operate in the 'real' world many others share the same opinion, the classical approach is not the answer to the complicity the real world presents [Chap 91] [Mill 91] [Such 87]. They and others that tried to embody intelligent systems have found out that the classic approach of modelling the world, planning and generating a priori plans for systems that operate in the real world is quite impossible.

The reaction to this failure of the classical planning was something like back to sources. Based on the ideas of the Cybernetics, a science founded by N. Wiener [Mas 90] [Wiener 48] and faded out at the mid 70's, the reactive machines became 'in' in the second half of the 80's. Brooks, one of the leaders of that movement went to the extreme while he coined the slogan *"The world is its best model"* [Broo 90].

McDermott [McDer 92] argues that the creatures of brooks use a world model, an implicit model that lies in the behaviours and in the mind of their creator, [Gat-Rod 92] share this point of view. Their success in dealing with the immediate problems and uncertainties of the real world led many of their supporters to believe that the behaviour based purely reactive or situated action systems are the only solution to *autonomous* systems. They threw away all the knowledge, experience and successes of the planning approach but could not deliver something better to solve complicated problems.

The answer to the question whether a model is essential to intelligence or not depends on how one defines intelligence. Since there is not a widely accepted definition of intelligence [Lec 94] the best thing that we can do is to examine already existing intelligent entities. We can take insects as our objects of reference and produce systems that will demonstrate behaviours similar to the insects' behaviours. This approach has two severe drawbacks: (1) we know some creatures that are widely accepted as more intelligent than insects and (2) we do not know whether insects have and use world's models.

In contrary the human race is generally accepted to be the most intelligent entity that we know. As well, a lot of work has been done on world modelling in humans. Jean Piaget [Piaget 67] [Bring 77] had shown that children develop a world model based on their experience. On the basis of this model they reason about the results of actions as well as about which action to choose to achieve a certain goal.

Let's go back to navigation which is one of the most critical problems an autonomous system faces. Navigation is the problem of arriving from the current position to a target that can be specified in terms of perceptive situation [Rod 94] [Mat 90] or in geometric terms [Lat 90].

A purely reactive approach system can assure target achieving if it has already made the way and learned it or if it was designed to follow a certain path. On the other hand the classic plan based navigation is assured to find the path if it has all the necessary information but not to realise it. The planning approach has the advantage of foreseeing, something that the reactive approach lacks completely.

A lot of effort was invested to bridge that gap. Behaviours' based systems with planning capability are the goal. They combine the efficiency of planning with the flexibility and adaptive ability of the reactive systems. Some possess learning capability and can re-execute a trajectory they have already followed. These systems store the trajectories that connect landmarks they can recognise as actions [Rod 94] [Mat 90 91] or sequence of actions [Hayes-Roth 93]. Planning is a search of a graph or other type of data base that hold the information already gathered by the system. There is no reasoning in these systems. A priori information can be introduced to these systems only in terms of behaviours, actions and perceptive situations that match the systems' capabilities.

The introducing of the a priori information requires an external system that can translate the raw data of the world, a work that till now was done manually. AuRa of Ron Arkin [Ark 90] is an example of a navigation system that use such an a priori knowledge that was introduced manually and uses behaviours to execute a mission.

The problem is to find a description of the world in terms of behaviours that enable reasoning about the results of a behaviour before actually exercising it. To find a description that the system itself would be able to create from raw data like a map or occupancy grid. This description should be as simple as possible to enable rapid reasoning and to match the system capabilities. It should be as general as possible to deal with a variety of behaviours and it should support communication between man and machine as well as between different machines.

In this work we propose a new approach, a normalisation of behaviours mapping. This mapping is based on the continuity equivalence and bifurcation of behaviours. It enables the mapping of reactive behaviours, reasoning about their results and understanding the relations among them. It also results in what we present as meta-behaviour that reduces to minimum the basic presentation of the world. Under this meta-behaviour the world can be represented using a two terms' vocabulary.

This work was done within the frame of the MARS project in the I^3A (Institut d'Informatique et d'Intelligence Artificielle) at Neuchâtel University. Project MARS is an effort to develop behaviour based autonomous intelligent system embedded in mobile robot. A description of the project is given in §2.6.

A survey of the state of the art in navigation is presented in chapter 2. The first part describes the classic planning approaches. The second part presents some of the most known reactive and behaviour based robot systems

The behaviours' mapping approach is presented in chapter 3. A definition of the behaviours' space mapping is given as well as some behaviours that explain the theory and demonstrate its applicability.

In chapter 4 a navigation theory based on this approach is presented.

A planning method that enables combining the classic planning approach with behaviour based execution is presented in chapter 5.

Chapters 6 presents the realisation of this approach in mobile robot navigation and some experimental results

Chapter 7 concludes this work with survey of its contribution and propositions for future work.

2. Survey of Mobile Robot Navigation

2. Survey of Mobile Robot Navigation

2.1 Introduction

The classic approach defines the term *navigation* by three questions that should be answered: [Lev-Law 90]

1. Where am I ?
2. Where are other places relative to me ?
3. How do I get there (to the other places) ?

For most of the navigation systems the answer to (1) is *localisation*, the answer to (2) is a *map* or a *model*, and to (3) are *path planning* and *path following*. For some reactive system only questions (2) and (3) are of interest. For these systems (2) is answered by attractors that the system senses and (3) is just a movement toward the attractors controlled by certain reactions and behaviours.

There are two different answers to the path following problem that inflict the planning as well. The older and so called "classic" is the *metric* approach in which the trajectory is given in geometric terms and the path following is realised by execution of a sequence of motor pre defined commands. A modern approach is the *reactive* navigation in which the robot reacts to the immediate perception demonstrating a certain behaviour. In this approach the trajectory is described as a sequence of behaviours.

Consequently, we can speak of two kinds of planning approaches that result in two different kinds of plans [Agre-Chap 90] [Hayes-Roth 93]:

1. Plan is a program.
2. Plan is not a program.

Plans as programs are the results of the classic planning approaches. The plan is like a program or computer code to be executed by the robot. The trajectory is given in a sequence of actions that should be executed and completed. The metric navigation is the best example of such a plan. Some of these planning techniques are described in 2.3.

Plans that are not programs cover a wide span of planing approaches and navigation systems. From navigation without any explicit program [Brait 84] [Broo 90] [MILL 90a], through a program as an advise or a guide [Agre-Chap 90], to a program as a target's (or targets') definition where the system chooses at each stage the best behaviour from several possibilities [Ark 90] [Hayes-Roth 93] [Rod 94]. These systems are described in 2.5.

The first part of this chapter deals with the classic approach. §2.2 describes the maps and world representation approaches which are basis of the classical methods. In §2.3 we survey the most important planning methods. §2.4 concludes the first part, describing some methods of localisation which is a fundamental problem in plans' execution.

The second part of the chapter describes reactive and behaviours based systems. §2.5 surveys the history and the state of the art of these systems and §2.6 describes the project MARS (Mobile Autonomous Robot System) within its frame this research was done.

2.2 Maps and World Representation

Maps are the mathematical or geometric representations of the relevant knowledge that we have about the world. *Models* are a higher level of treatment and

representation in which we give meaning to the knowledge, meaning that enables planning and localisation.

Most of the graph based planning methods and all the potential field planning methods use maps of the world as an intermediate level from which an appropriate world model is generated. However some graph based systems generate the graph directly from their perception of the world. Table 2.1 summarises the different maps and the world models that are generated from them.

Table 2.1 Maps and Models

	Map	Model	Navigation System
1.	Occupancy grid	Direct Cells graph Voronoi Diagram Potential field	Moravec Elfes Mars Y. Gat Barraquand Latombe
2.	Polygons	Line of Sight Freeways Voronoi Diagram Potential Field	 Brooks Schwartz Sharir Gambardella
3.	Traversability Zones	Weighted Region Hierarchical Cell Connection Graph	Mitchel Meystel
4.	Objects Geometric	Object topology	AuRa R. Arkin Toto M. Mataric
5.	Direct	Object Topology	Mars M. Rodriguez

Figure 2.1 describes the possible relations among maps, models, planning and execution. The double lines represent the classical approach, the double dotted lines the purely reactive approach and the bold lines represent our approach in project MARS.

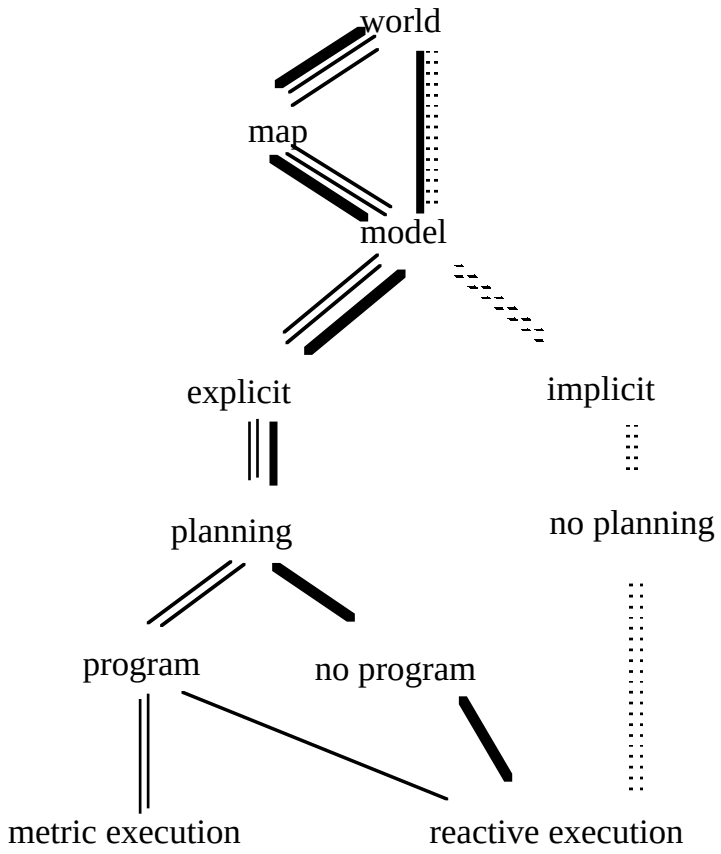


Figure 2.1 The relations among maps, models, plans and execution

The first step toward path planning is the generation of a world representation that the planning system can work with. There are four general approaches to represent the world:

1. Occupancy grid:- A 2D or 3D grid is used as the base of the representation. Each cell of the grid is given a value that indicates the occupancy state of the cell.
2. Geometric description:- A description of the obstacles or of the free space is given in geometrical terms mostly polygons
3. Objects:- Here the description of the world is given directly in objects and their geometric or topologic relations. The objects can also be symbols or sensorial states.
4. Configuration space:- The world is described in terms of an n-degrees-of-freedom system's configurations

2.2.1 The Configuration Space

The configuration space is a technique of creating a map that facilitates modelling and planning. This representation is applicable to occupancy grids as well as geometric maps.

The idea of shrinking a dimensional robot to a point, parameterising its dimensions and kinematics, and mapping the obstacle or the free space into a multidimensional space was first introduced by Udupa [Udu 77]. It was further investigated by Lozano-Perez [Loz 79] who also gave the name *configuration space*. In this space the robot is represented by a point and each dimension is one of the robot state variables. We map into the configuration space all the "legal", non collide, configurations. The union of all the latter defines the free space. This mapping transforms the problem of motion planning of dimensional robot into a point motion planning problem and represents the constraints on the motion in an explicit form.

An example of a configuration space for a round mobile robot in 2D working space is given in figure 2.2 below. We see there that the borders of the obstacles were extended by the robot radius Hence the free space (*Cfree*) describes all the permitted positions of the robot centre.

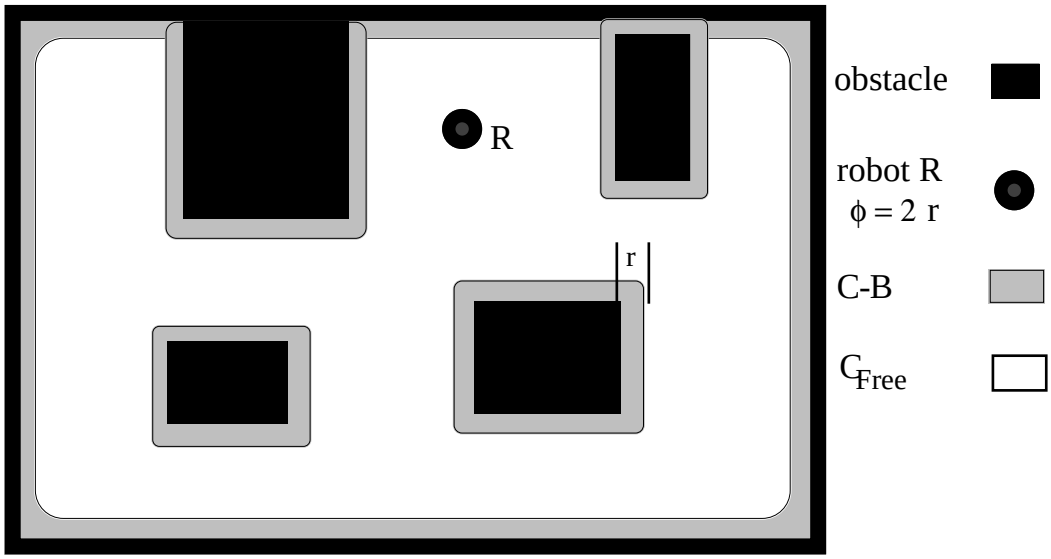


Figure 2.2 Configuration space for round robot

The formal representation of the configuration space that is described here is given by Latombe [Lat 91 p7-11] and the same notions he uses will be used throughout this work.

Let W be an Euclidean space R^N with N =2 or 3, called the workspace
Let A be the robot that can be described as a compact (i.e. closed and bounded) subset of W.
Let F_A and F_W be Cartesian frames linked to A and W respectively.
Let B be a closed subset of W representing the obstacles.

By definition, since A is rigid, every point a of A has a fix position with respect to F_A, but its position in W depends on the position and orientation of F_A relative to F_W. A configuration of an object is a specification of the position of every point of this object relative to a fix reference frame. Therefore, a configuration q of A

is specified by the position P and the orientation Θ of F_A with respect to F_W. The configuration space of A is the space C of all the configurations of A.

Now we can define the free space in C, which we denote C_{free} as the union of all the configuration of A for which $A \cap \{B_1,...,B_m\} = \{\phi\}$. C-B is the representation of $\{B_1,...,B_m\}$ in C.

2.2.2 Occupancy Grid

The occupancy grid is a direct digitised modelling approach. A grid is imposed on the working space. Each of the grid cells can have one of the following values: occupied, free [Bar-Lat 89] [Mitch 88], and in some systems also unknown [Elf 85] [Elf 89] [Weis 87]. This kind of model can be updated easily by updating the value of each grid cell without having any influence on the values of other cells. Mitchel [Mitch 88] and Elfes [Elf 89] plan their path directly on the occupancy grid by searching a path that passes only through empty cells. Mitchel also introduces a refinement to the grid by using a quadtree. Barraquand and Latombe use the grid as a base on which they impose a potential field. [Bar-Lat 89] [Lat 91]. The calculation of the potential field gradient and the search of the path are executed directly within the grid cells. One of the main advantages of this representation is that it enables both the integration of the free space information and direct plans of a path within the same single "chip". [Crow 87]

2.2.3 Moravec - Elfes probabilistic grid

Moravec and Elfes have developed a navigation system that is based on a probabilistic occupancy grid and sonar. Their system includes planning, mapping and localisation all within the occupancy grid [Mor-Elf 85] [Elf 85, 89].

The occupancy grid representation employs a 2D tessellation of the space into cells, where each cell stores the probabilistic information of its occupancy state. Formally an occupancy field is a discrete-state stochastic process defined over a set of continuous spatial co-ordinates, while the occupancy grid is a lattice process, defined over a discrete spatial lattice [Elf 89]. With each cell C of the occupancy grid a state variable $s(C)$ is associated. $s(C)$ is defined as a discrete random variable with 2 states occupied (OCC) and empty (EMP). The cell state is exclusive and exhaustive hence $P[s(C) = OCC] + P[s(C) = EMP] = 1$

The information of the occupancy state of each grid cell can be obtained from several sources and from several readings each of which with a certain accuracy probability. For an a priori information obtained from a map the reliability depends on the time that passed from the mapping and the nature of objects. The reliability and hence the probability decreases with the age of the information. The rate of decrease depends on the nature of the object. A mountain is more likely to stay in the same place and shape than a house, which in turn is more stable than a temporary barrier.

For sensing devices, like sonar, IR and other distance sensors, the accuracy depends mostly on the sensor itself, the distance, and the shape, the texture and the material of the object to which the distance is measured. To each sensor we denote a probability density function of the form $P(r)$ where r is the measured distance. A probability function is ascribed to other sources of information. The estimation of the occupancy probability is done by fusing the information regarding each cell using a Bayesian estimation procedure.

The occupancy grid here is modelled as a Markov random field of order 0 so each cell value is estimated independently of the others. To allow an incremental composition of information a sequential updating formulation of Bayes' theory is

applied. Given a current estimate state s of cell C_i at a time t, $p[t|s(Ci) = OCC]$ and the updating information $p[t+1|s(Ci) = OCC]$ the cell state at $t+1$ is evaluated as follow:

$$p[t+1|s(Ci) = occ] = \frac{p[t+1|s(Ci) = occ]p[t|s(Ci) = occ]}{\sum_{s(Ci)} p[t+1|s(Ci)]p[t|s(Ci)]}$$

The above formula is the multiplication of the current probability that the cell is occupied and the updating probability that the cell is occupied normalised by the sum of the multiplication of the old and updating probabilities to keep the sum of the new probabilities equals to 1.

An ideal sensor would have an occupancy probability profile that can be characterised by:

$$p(r|z) = \begin{cases} \delta(r-z) & ; r \le z \\ 0,5 & ; r > z \end{cases}$$ z is the real distance.

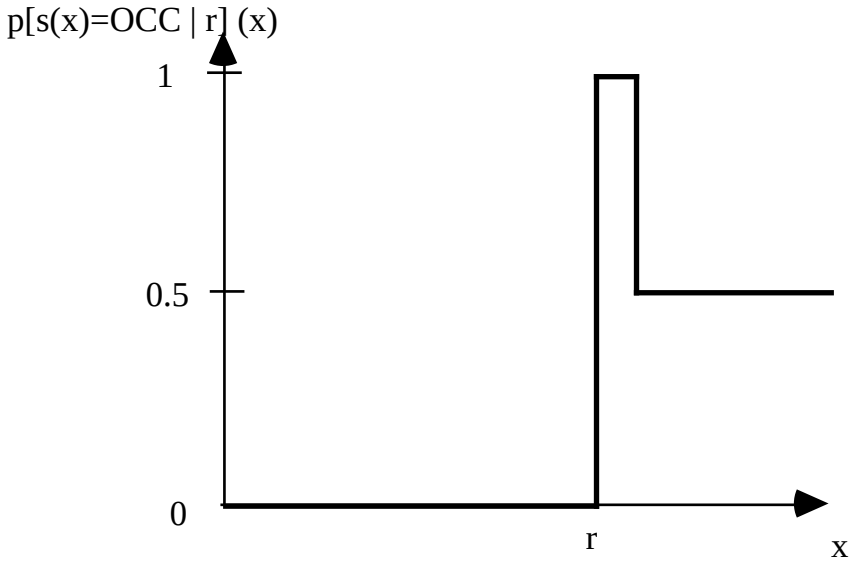


Figure 2.3 Occupancy probability profile for an ideal sensor

When applied to a grid it is translated to a rectangle of one unite height over one cell (figure 2.3).

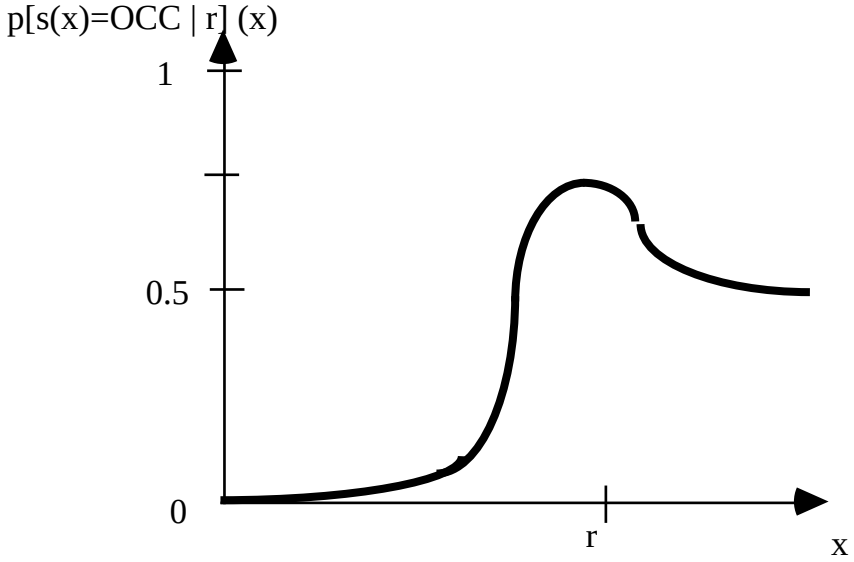


Figure 2.4 occupancy probability profile for real sensor

For a real sensor we have to calculate or more often to measure the probability profile. An example is given in figure 2.4.

2.3 World modelling and Planning Approaches

The ultimate goal of path planning is to find a way for the robot to move from one location or configuration, its current location, to another location (configuration) without colliding with any obstacles. A good planning method is evaluated by its completeness (its ability to find such a path if one exists), its applicability and its complexity. In many cases we also require that the path will be optimal or at least near optimal.

A good world model is essential to path planning. A navigation system, and actually any moving intelligent creature, should transform the physical properties of the world into some level of symbolic representation from which it would be able to extract the information relevant to navigation. The symbolic levels of the representation are varying from almost direct representation of the environment as is the case in occupancy grids (see for example [Elfes 89] or [Bar-Lat 89]) through intermediate levels like geometrical description, mathematical description and sensory states' representation [MAT 90a], to highly sophisticated representation. In the latter the system is given objects of the world like walls, corridors, doors and roads, their topologic relations, their geometric location and attached relevant properties like passability [Lev-Law 90] [Hors 88] [Mitch 88].

There is a strong coupling between the model and the planning process. A large number of world modelling methods were developed to solve the path planning problem. These methods can be divided into two groups according to the planning method they serve: *graph based* methods and *potential field* methods

The potential field methods impose an artificial potential on the robot working space. The potential field consists of an attractive field centred at the target configuration and repulsive potentials on the borders of the obstacles. The robot itself is represented by a point particle in a configuration space or a collection of connected points in the working space moving under the force generated by the combined potential fields.

2.3.1 Potential Field Methods

This approach is inspired by fluid mechanics and electromagnetic. The basic idea is to impose an artificial potential field on the map (a configuration space, occupancy grid or polygons map) and to consider the robot as a particle (or connected particles) moving under its influence. The potential field, if adequately chosen, is supposed to reflect the structure of space in such a way that will cause a particle moving under its influence to choose a collision free (and normally optimal) path from the current configuration to the goal configuration.

A "good" potential field is usually a superposition of an attractive force that pulls toward the goal configuration and several repulsive fields that push the robot away from obstacles and prevent collisions.

When a potential field, denoted by the potential function U, is imposed on the configuration space the movement of the robot and hence the path are calculated consecutively by moving along the line of force that are induced by the potential field and defined as $\overline{F}(q) = -\nabla \overline{U}(q)$ at any point q of the configuration space.

Originally the potential field method was developed by Khatib [Kha 86] as a real time collision avoidance system. The treatment of the environment is local and the system acts like fast descending optimisation system. The latter is the cause for local minima which are the most problematic drawback when applying this approach as a general path planner. There are two different approaches to deal with the local minima, the first is the attempt to design an "ideal" potential function that has only one minimum, at the goal configuration, regretfully this is almost impossible. The second is to furnish the system with efficient techniques to escape from the undesired minima. It is impossible to ensure a minimum free potential function in high dimensional (3D and more) configuration space, therefore most of the existing systems are using a combination of these two approaches to deal with the local minima's problem.

2.3.1.1 Barraquand Latombe Distributed Representation

J. Barraquand and J-C Latombe [Bar-Lat 89] propose a numerical potential navigation function based on a grid representation of the configuration space. They compute the "Manhattan distance" in GC_{free} (the collection of all grid's free cells) from the goal to each of GC_{free} cells. The distance is calculated by a "wave front expansion" algorithm. The goal cell is denoted the value 0, each of its free neighbours is given the value 1, each of theirs free non valued neighbours is given the value 2 and so on. The process terminates when the entire subset of GC_{free} accessible from goal cell is explored. A best first search algorithm is used to find the shortest (in "Manhattan distance") path from the initial cell to the goal cell. A drawback of this method is that it usually induces paths that rub the $C_{obstacle}$.

An improved method that induces paths remote from the $C_{obstacle}$ is based on extracting the skeleton of GC_{free} in the first stage and computing a potential field guided by the skeleton in the second and third stages. In the first stage the distance from the borders of GC_{free} to each cell $c \in GC_{free}$ is computed using a wave front expansion algorithm. Whenever two wave fronts meet, the cell is registered in the skeleton S.

Next the goal and the shortest path from the goal to S are added to S and the distance of each cell $s \in S$ from the goal is computed and S cells are given their distance as potential value. At the third stage the distance of each cell c ($c \in GC_{free} \forall c \notin S$) from S is computed in the same wave front expansion method starting now from S. c is given a value based on its distance from S.:

$d_{cS} = dist(c, s | s \in S)$ where s is the nearest skeleton point to c.

$V(c) = V(s) + d_{cS}$ (V stands for the cell's potential value).

A best first search procedure is applied to find the path, which is not necessarily the shortest.

2.3.1.2 Discussion

Usually these planning methods are incomplete in the sense that they are not guarantied to find a solution even when one exists. Their main advantage as motion planners is their rapidity [Lat 91].

2.3.2 Road Maps (Graph Based Methods)

The *road map* approach is a graph representation of the workspace or the configuration space. The basic general idea is to capture the region connectivity and represent it in a graph. The search of a path is performed on this graph of "normalised" paths, hence the problem of path planning is simplified and reduced to graph search. Several different approaches are used to create the graph, two of them are surveyed here (i) *"visibility graph"* and (ii) *"free ways net"*. A third approach, the *retraction* methods, which is the base of the representation used in our theory of navigation, is surveyed here briefly and discussed in details in chapter 4.

Schwartz & Sharir [Sch-Shar 88] give a definition of retraction that is applicable to all the road maps' methods (with the necessary adoptions):

> *Any pair of positions lies in the same connected component of the c-space (workspace) if and only if their normalised graph representation lie in the same connected component of the road map.*

2.3.2.1 Visibility Graph

This method is among the first used in motion planning. It was firstly introduced by Nilsson [Nils 69]. When the c-space is given in a polygonal description it can be shown that if a free path exists between q_{init} and q_{goal} then it exists a semi free path (a path that also touches the borders of the free space) that pass through the vertices of the obstacles' borders. Moreover, the shortest path will pass through one or more vertices, unless there is a straight line that connects q_{init} to q_{goal} and lies entirely within C_{free} [Lat 91] [Mitch 86] [Mitch 88].

> The visibility graph G is then defined as follows:
> *G's nodes are q_{iniit}, q_{goal} and the vertices of C-B, (at finite distance). Any pair of nodes is connected if and only if the straight line that connects the vertices they represent lies entirely in C_{free} except possibly its two end points, or it is an edge of C-B [Lat 91 p156].*

After constructing G, it should be searched for connected path between q_{init} and q_{goal}. The search can be worked out using various searching techniques. An improvement to the basic visibility graph is G' in which all the nodes that represent concave vertices where deleted. The naive approach to construct the visibility graph is to check the line segments connecting every pair of convex vertices. If a line segment is an edge or lies entirely in the free space than it is added to the graph. This method is very expensive and requires a computation time of $O(n^3)$ where n is the number of convex vertices.

Some more efficient methods where developed which cost $0(k + nlogn)$ where $k = n^2$ in the worst case and $k = n$ in the best case. The biggest problem of this method is the size of the searching space. An example is given in figure 2.5. The lines of sight from concave vertices are not considered for the search in order to save time and computing efforts.

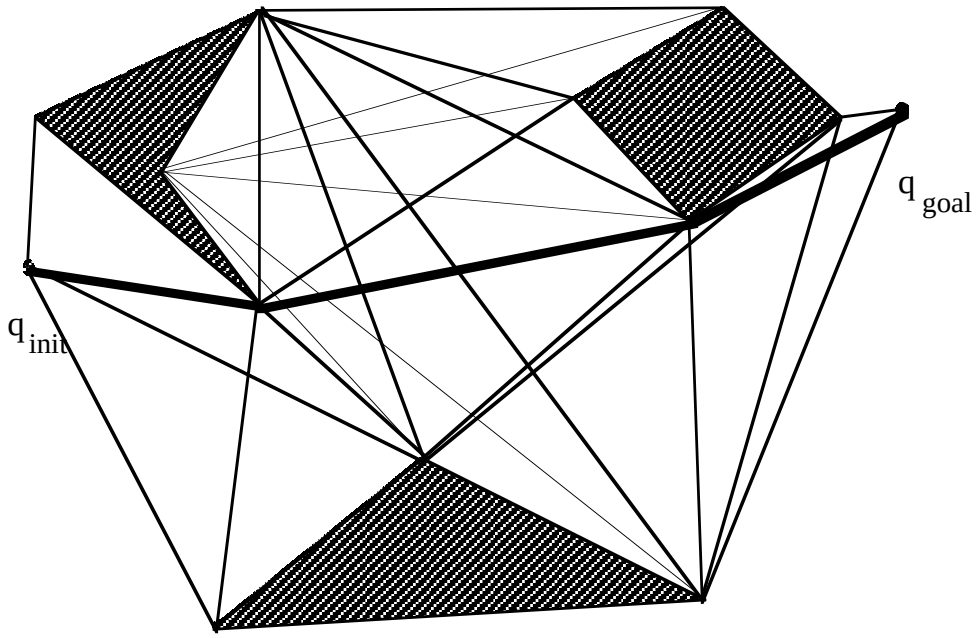


Figure 2.5 Visibility graph. The lines from the concave vertex are broken. The calculated path in bold lines

2.3.2.2 Freeway Method

The freeway method, which was presented by Brooks [Brooks 83] is adequate to solve the problem of translation and rotation of moving polygonal object, in 2D and 3D polygonal space. It is based on extracting straight linear general cylinders from the free space. These cylinders are called Freeways and connecting them into a graph creates the freeway representation.

In the 2D case each pair of edges that belong to the free space borders is checked whether they face each other (meaning that a straight line connecting them passes entirely in the free space [Lat 91]). Whenever they are, the bisector of the lines that contain the edges is taken as the axis of the cylinder (figure 2.6). A perpendicular line is moved along the axis, its intersections with the edges (E_1 and E_2 in figure 2.6) define the cylinder section, when it passes the edge end points (convex vertices) it keeps its length and the cylinder continues till the line intersects another edge. A transfer from one freeway to another is possible whenever the two intersect and the intersection is non empty.

A freeways' net is the representation of the possible motion along and between the cylinder axis. Each freeway is checked for the possible range of free orientation of the moving object and each intersection is checked for the possibility to transfer from one freeway to the other. The results are mapped into a connectivity graph that can be searched for a path from one point to another in the workspace.

The freeways' representation has the advantage of simplicity when solving the problem for moving polygonal object because it keeps the configuration space dimension as low as the real world dimension. However this representation is in general incomplete.

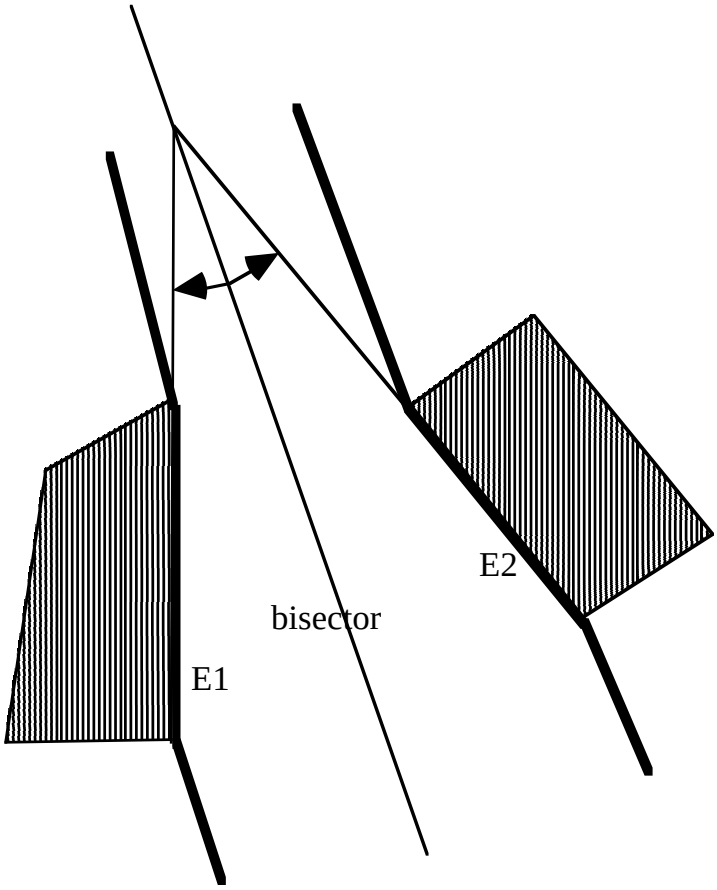


Figure 2.6 Extraction of a freeway

2.3.2.3 Retraction

The retraction methods proceed by retracting the working space or the configuration space onto a lower dimensional (usually 1D) subspace, while conserving the connectivity of its components. This reduces the dimension of the problem and emphasis the essential property of connectivity. When the subspace is one dimensional the problem of path planning becomes one of graph search.

The most well known and widely used of those methods is the Voronoi diagram and its variants. Shortly the Voronoi diagram is the union of all point in a given space that are equidistant from at least two point of a given subset of the space. In navigation the latter subspace is usually the borders of the obstacles or the borders of the free space. Being the base tool used by the navigation presented in chapter 4, a formal treatment of the Voronoi diagram is discussed there.

The very earliest motivation for the study of the Voronoi diagram stemmed from the theory of quadratic forms. Gauss (1840) observed that quadratic forms have an interpretation in the form of the diagram. The idea was exploited by Dirichlet (1850) to establish a simple proof of the unique reducibility of quadratic forms. A generalisation to higher dimensions was provided by Voronoi, after whom the diagram is named. A detailed and good review of the Voronoi diagram and its application in science is given by F. Aurenhammer [Aur 91].

In robotics, the use of the Voronoi diagram is very popular [Sch-Shar 88]. When constructing the diagram, on top of conserving the connectivity of the space

components, the lines that represent the free space are the farthest as possible from the obstacles. The Voronoi diagram is also used as a base for potential field planning [Bar-Lat 89] who developed a way to create the diagram from an occupancy grid (see 2.3.1.1.).

2.3.2.4 Constant Traversability Zones (CTZ)

One of the most important parameters for navigation is the traversability of the areas the robot moves through. A model of the world based on the traversability of each area is used by several navigation systems [Meys 91] [Mitch 88]. A traversability factor is computed for each area of the working space. The traversability factor can be defined in many forms, for example as the ratio between the mean velocity in the area and the maximum velocity or the inverse of the required propulsion power.

The world is divided into homogenous zones (CTZ Constant Traversability Zone), each zone is composed of areas with equal traversability factor (within certain distance from the mean value). The representation can be in a graph where each node represents the centre of a CTZ and the branches connects neighbouring zones. Another possible representation is a quadtree [Mitch 88]. This approach enables a world modelling at any scaling (resolution) hence it supports a hierarchical planning. Another advantage is the optimisation oriented traversability factor that represents directly the phenomena relevant to the optimisation.

2.3.2.4.1 Hierarchical Representation and Planning

A three levels nested hierarchical system has been proposed by Meystel [Meys 91]. The system consists of a planner at the top level, a navigator at the intermediate level and a pilot at the low level. Each of the levels uses a specific world representation in adequate scale. The planner uses a high scaling low resolution long range CTZ model from which it derives a path to the goal. The path is define as a sequence of strip and transmitted to the navigator.

The navigator uses a lower scaling higher resolution representation. It explores the planner path piecemeal, trying to find a path within the strip's borders. If succeeded it transmits the path to the pilot that deals with the immediate move control, if failed it returns the control to the planner that searches another solution. The pilot uses a detailed representation of the immediate environment to find an obstacle free path, when failed it returns the task back to the navigator.

2.3.3 Advantages and Disadvantages.

The main advantage of the graph based method is the rapidity of the search and the compactness of their presentation. From all these methods only the visibility graph suffers from the same disadvantage usually related to the potential field methods. Any time the position of the robot or the target is changed the model should be updated. The other graphs are perhaps a little bit more difficult to construct but once it has been done they rest unchanged as long as the world does not change.

The main drawback of all the metric planning methods is the rigidity of the plans. The robot is obliged to follow the planed path very accurately. Whenever the environment changes or the robot drift from the planed path we are in troubles. The problem of drift is dealt with by introducing self localisation abilities, that enable the robot to correct its position based on its sensors (see 2.4)

The problem of a changing world is dealt with partially by obstacle avoidance systems. The robot makes a detour around unexpected obstacles and then returns to its original trajectory, but when the changes are more severe than an obstacle on the way, the map and the model should be updated and re-planning is essential.

The metric planning which is followed by metric execution is limited to static or quasi static world where everything is known and expected. Brooks argues [Broo 90, 91] that planning as such is not realistic and in his words "cannot be grounded" and

disconnected from the real world. Chapman in [Chap 91] discusses the combinatorial explosion of exact planning and argues for plans as communication, in which the plan is regarded as an advise to the robot who has the ability to take the right actions based on the perceived situations.

To overcome this drawback most of the current robotics navigation tries to introduce more flexibility and reactivity in planning and execution. The rest of the chapter is dedicated to such navigation systems.

2.4 Localisation in Metric Navigation

The answer to the question "where am I" has essential role in navigation, in path planning as well as in path following. A path can not be planned unless the planner can situate the robot current position and the goal on the map or in the configuration space. When executing a pre-planned path the robot position relative to the path should be known in order to control the robot move. When a path is computed in geometrical terms the answer to the question should be in the same terms. The robot drift from the path, hence its ability to arrive to the goal, depends heavily on its localisation accuracy.

2.4.1 Dead Reckoning

Dead reckoning covers all the localisation approaches which rely solely on internal computation. These navigation systems wander around the world like blinds and do not use any external feedback to verify and correct their position.

Terrestrial dead reckoning is based almost solely on odometery, which in some systems is aided by compass or gyro, especially for outdoor navigation. The position computation from the odometery output is very simple. The reading of the distance is integrated using the following formulas:

given:

x, y are the Cartesian co-ordinates

θ is the direction of move in the same co-ordinates system

s is the distance travelled by the robot.

we have:

(1) $$x(s) = \int_s Cos\theta(s)ds$$

(2) $$y(s) = \int_s Sin\theta(s)ds$$

The direction of move, θ can be obtained either directly from an attitude sensor, e.g. compass or gyro, from the direction of the wheels or from computation based on the distance difference between two wheels as in (3).

When s_r, s_l are the distances travelled by the right and left wheels respectively and D is the lateral distance between the wheels then:

(3) $$\theta(s) = (s_r - s_l)/D$$

2.4.2 Sensorial Based Localisation

The main problem of the odometery based positioning is the error accumulation. Several methods were suggested to compensate the positioning error using sensors input that varies from sonar at the lower end to video and image processing at the higher end.

A. Elfes presented a localisation correcting system that matches a local occupancy grid with a global one [Elf 89]. The system creates a local occupancy grid

from current sonar readings. It tries to find a position around the position known from the odometery that will give the best matching between the local and the global occupancy grids.

J. Crowly suggested line segments matching to correct the positioning errors [Crow 88a]. Line segments are extracted from sonar readings and matched with straight lines that represent the borders of the free space in the global map. The method is based on the assumption that the free space is characterised by straight lines and distinguishable corners hence it is well adapted to indoor navigation. A probability process corrects the position of the robot and the results are quite good. A work based on Crowly's ideas was made by Wang [Wang 93] in our institute.

The triangulation approach is well known and widely used in navigation. From the known position of three identified points and the angles between the directions toward these point the current position is calculated. Only a simple trigonometric calculation is needed. A reduced method is based on a known global direction, from a compass for example, and the azimuth to only two known identified points.

Proximity beacons are use as well for localisation. There are active beacons that transmit a radio or light signal with identification code as well as passive beacons using for example bar-code identification. While the robot pass closed to a beacon it corrects its positioning.

2.5 Reactive Navigation

The classical approach to navigation that is based entirely on planning and on an explicit symbolic model of the world has exhausted the computation resources all along the way [Broo 91]. Even more, it does not seem to operate successfully in a dynamic changing world. It has difficulties in dealing with sensors' errors as well. The models it uses are not realistic, it appears that the world is to complicated to be presented completely. Whenever an attempt to create a complete model that includes all the essential knowledge needed to deal with the uncertainties and surprises of the real world, the model became enormously big and the planning too expensive in time and computer resources as J.P.L found out while developing their Martian robot [Mill 89 91].

The reactive approach to movement control and navigation takes the opposite extreme. One of the leader of this approach, Rodney A. Brooks stated it clearly and loudly in a very short sentence *"The world is its best model"* [Broo 90 91]. In these articles he also claims that, all in all, the symbolic system, which is a fundamental property of the *"classical AI"* , is in bankruptcy.

To overcome the severe limitation of classical AI the reactive or situated activity proposes an alternative dogma based on what Brooks calls the physical grounding hypothesis. The robot is connected to the world through a set of sensors and actuators, all the knowledge in the system is extract from physical sensors and all the goals and desires are expressed in terms of physical actions. The basic module of such a system is the behaviour that is achieved by direct or almost direct connection of the sensors to the actuators. The following sections survey some of the reactive robots that demonstrate behaviours and navigation abilities.

2.5.1 Negative Feedback (Cybernetics)

Situated reactive robots were developed and tested at the late 40s and the beginning of the 50s. Based on the idea of negative feedback that was developed by Norbert Wiener and Julian Biglow, Wiener and J. Wiesner built a simple bug robot that could behave like a moth (e.g. attracted by light) or like a bedbug (e.g. repelled by light). Two photocells were mounted on a tricycle, one at each side. The output from the cells, after amplification, reaches the tiller controlling the steering wheel. Depending on the direction of the output voltage, the cart will go toward or away from light. [Mas 90].

The same idea was presented by A. Braitenberg [Brait 84] where the light detectors are connected directly to two motors, each drives a wheel on opposite side of the robot. The motors' speed is directly proportional to the intensity of the light. When the left detector is connected to the left motor and the right to the right motor the robot tends to turn away from a light source. For an external observer the robot seems to b timid. A cross connection will create an aggressive behaviour, the robot will follow and chase the light source.

At the beginning of the 50s Dr. W. Grey Walter of the Burden Neurological Institute in Bristol, England built a robot resembling the bug of Wiener but with touch sensor in addition. This robot named the Tortoise is attracted to moderate light repelled by intensive light and avoids obstacle on its path [Mas 90].

These systems demonstrate how even with the simplest architecture one can imagine that a somewhat "intelligent" behaviour can emerge (from the observer point of view).

2.5.2 The Creatures of Brooks

Brooks and his team in the Mobot Lab at MIT have constructed and developed several mobile robots based on the direct coupling of perception and action. Their work

is a bottom up development that implies the implementation of basic low level behaviours from which emerge more sophisticated and intelligent behaviours.

In order to construct physically grounded systems Brooks has developed the subsumption architecture. The subsumption program is built on computational substrate that is organised into a series of layers, each connecting perception to action. In his robots the substrate is a network of finite state machines augmented with timing elements (AFSM). [Broo 86] [Broo 90].

Each AFSM has a set of registers and a set of timers, or alarm clocks, connected to a finite state machine that control a combination network by the registers. Messages from other machine are written into the registers, replacing any existing information, through input wires that attached to them. The arrival of a message or the expiration of a timer can trigger a change of state. The states can either wait, dispatch to one or two other states, or compute a combination function from the registers' values. The results are directed either back to one of the registers or to an output of the AFSM. Sensors deposit their values in certain registers and certain outputs direct commands to actuators.

New machine can be connected to an existing network in a number of ways. New inputs can be connected to existing registers. New machine can inhibit existing output or suppress existing input, by being attached as side-taps to existing wires. Inhibition and suppression are the mechanism by which conflict between actuator commands is resolved.

The behaviour language groups multiple processes (each of which usually turns to be implemented as a single AFSM) into behaviours, which are more manageable units, being selectively activated or de-activated. Messages passing, suppression and inhibition can exist within a behaviour as well as between behaviours. Behaviours act as abstract barriers, one cannot reach inside another.

2.5.2.1 Allen

Allen is equipped with sonar range sensors and odometery onboard and Lisp machine to simulate the subsumption architecture off board. The control consists of three layers. The first layer is an obstacle avoidance mechanism based directly and only on the sonar return. The control command is the vector sum of the negative inverse square of the sonar returns. As a result, each obstacle induces a repulsive force and the robot tends to turn away from obstacles. An additional reflex halts the robot when an obstacle on front moves in its direction.

The second layer, the wander around, is a random direction generator that produces a new direction about every 10 seconds. This direction is coupled with the obstacle avoidance in vector addition to generate the control command.

The third layer makes the robot look with his sonar to distant places and tries to go toward them. It monitors the relative position by the odometery and generate a direction that suppresses the second layer desire and added to the first layer in vector addition.

The robot seems to exercise two intelligent behaviours, when only the first and second layers are activated it wander around while avoiding obstacles. When the third layer is activated as well, the robot goes toward a target searching its way among obstacles and avoids collisions. [Broo 86] [Broo 90]

2.5.3 JPL/CAL TECH Reactive Mobile Robots

David Miller from JPL (Jet Propulsion Laboratory at California Institute of Technology) has developed 2 totally reactive robots for indoor and outdoor use. His

work was motivated by the failure to develop such autonomous robots based on the classical approach [Mill 89] [Mill 91].

2.5.3.1 Indoor Total reactivity

This robot was built on a remote controlled toy car. It uses only 2 bytes of memory to model the world and is programmed to patrol an area, locate pickable objects and dump them in central location. The sensory consists of contact and proximity sensors and light detectors which are wired to the robot's effectors. The proximity and contact sensors induce repulsive force so when a sensor fired an appropriate behaviour is executed (i.e., turn right if the left proximity sensor is fired). The light detectors induce attraction.

Such a system tends to be locked in extensive loops, to avoid these loops three methods where applied. An asymmetry of opposite behaviours causes the robot to crab aside while exercising two opposite behaviours successively. A counter notes behaviours' transition, if the number of direction changes exceeds the limit a random steering is executed. To get out of blind alleys the robot will backup a certain distance while its way is blocked. If afterwards it won't be able to go forward more than it has backed next it will backup a longer distance. This, combined with the random steering would be sufficient.

When the robot had acquired an object it would continue until it could see a homing beacon. It would then steer toward the beacon but in lower priority than the avoidance behaviours. When the robot reached the beacon it would dump the object and continue exploring.

This robot has no vision and the only world model it maintains is the states of its behaviours, the switch of direction counter and the backup counter. The robot accomplished a quite complicated task of moving through the world while doing its job. The shortcoming of the system is the inefficiency of the path it follows.

2.5.3.2 Outdoor Reactive

In addition to the indoor robot the outdoor robot is equipped with a very narrow depth field camera. A set of images at different focus settings is taken. The areas of the image that are in focus are analysed and compared to an idealised plane. Significant deviations are considered as obstacles that should be avoided by the robot. The behaviours are the same as for the indoor robot and the only difference is the distance of sensing.

2.5.4 Why Pure Reactivity is not the Solution

Pure reactivity provides the basic behaviours and reflexes that enable the immediate low level of existence. The robot can move without bumping into walls and obstacles. It can even go toward or follows certain pre defined stimulus and performs some simple task on its way like collecting empty beer's cans. However its abilities are quite rigid, whenever we want the robot to demonstrate a certain behaviour in given circumstances, a long and accurate tuning is needed. For example the robot Herbert [Broo 91] was programmed and conditioned to turn left after passing through a door in order to return to the place where he dumped the soda cans. Changing the dumping site will require new programming.

Moreover the reactive robots are quite sensitive to local conditions. The robot behaviour in the presence of several stimuli of the same kind depends on the local conditions unless behaviours that deal with such cases were provided. For example a robot that follows corridors when confronted with a Y junction will follows one of the possible corridors. Which one will be chosen depends, if there is no preliminary fixed preference to one side, on the exact position and attitude of the robot, conditions that can be changed from one passage to another. Hence a consistent behaviour of the robot cannot be guarantied.

If we want to assure the safe home returning of a robot like Herbert we should provide sets of rules and behaviours that covers all the possibilities in its working space. In a complex environment that will require enormous programming effort as well as huge memory. That limitation leads to the development of robots that use world models and presentation to control their reactive basic behaviours.

2.5.5 "Toto" - Behaviours based Navigation

Toto is a robot that belongs to the ensemble of Brooks creatures, its description is given in [Broo 90] [Mat 90a]. It is a result of the limits of the pure reactive "navigation". This 3 wheeler is capable of following a continuous trajectory in discontinuous velocity. It is equipped with a ring of 12 sonar range sensors in equal angular partition and a flux gate.

The robot was programmed in the behaviour language within the layered paradigm of the subsumption architecture. The system consists of 3 main layers: (i) collision-free move, (ii) landmarks detection, and (iii) map learning and path planning.

An object boundary tracing emerges from the combination of 4 simple navigation rules [Mat 89]:

Stroll: uses stop go and backup commands based on the distance from the danger zone and allows the robot to move safely forward.

Avoid: changes the robot direction to turn away from any obstacle within the safe area. Combined with stroll it furnishes the robot the collision free wander around behaviour.

Align: keeps the robot in the edging zone by keeping it from turning away from the followed object. The combination with stroll and avoid produces a convex boundary following behaviour.

Correct: by monitoring the side sonars it allows the robot to follow concave boundaries.

Danger zone, minimum safe distance and edging distance are three thresholds distances.

2.5.5.1 Landmarks

Being frequent and static Mataric chose walls and corridors to be the landmarks for Toto world representation. The low level navigation behaviour produces a path at more or less constant distance from the objects' boundaries. When the side sonar readings are short and the average compass bearing is constant for a certain time without too large deviation from that average, the landmark detection mechanism concludes the existence of a wall on that side. The presence of walls in both sides indicates a corridor. The actual definitions of landmarks in the system are: (i) left wall, (ii) right wall and (iii) corridor to each attached its compass bearing.

2.5.5.2 Spatial Learning

A distributed graph based on the subsumption architecture is used to represent the world, i.e. the landmarks and their relations. Each node of the graph is a behaviour consisting of a collection of AFSMs [Mat 90b] and is equivalent to any other robot's behaviour. Each node is an independent agent that corresponds to certain inputs, landmarks, and generates messages that pass to neighbouring nodes or in certain cases as directives to the motors.

The graph interconnection is pre-compiled and implemented in hardware which result in static topology. The chosen topology is a linear list and the robot is initially given an empty nodes graph which it fills during its exploring of the environment. By connecting all the behaviours (nodes) to a switchboard that routes appropriately jumper connections the problem of loops and junctions that implies connection between non neighbours is solved and full flexibility of dynamic connection is achieved. To keep the complexity linear with the number of nodes, the number of connection from each node is limited to 4, a number that was proved experimentally sufficient.

Each node of the graph contains the type of the landmark, the compass bearing, rough position estimation and rough length (in time) estimation. Whenever a landmark is detected its type and compass bearing are broadcast to the entire graph. Initially the graph nodes are empty and the landmark information is stored automatically in the first node that corresponds now to the robot current position and becomes activated.

When a node receives a broadcast landmark it compares the type, compass bearing and estimate position to its own, taking into account spatial duality. A matching causes activation of the node and deactivation of the precedent. When none of the nodes match the landmark it is stored in as new in the node adjacent to the active node, or when it is not empty to an empty node which is defined adjacent by a jumper. Whenever a node is active it spreads expectations to its neighbour(s) in the direction of move. A match is considered true only if expected, when not expected it is either false or indicates a loop.

Based on the graph a path to a given destination can be planned and executed. The goal node continuously sends a call to its neighbours which in turn propagate this call to their neighbours. The call is eventually reach the currently active node. The robot by following the direction from which the call is arrived will reach the goal node on the shortest topological path. To find the shortest physical path the length estimation is used, when a node receives more than one call it will choose the direction with the attached shortest accumulate length.

2.5.5.3 Discussion

M. Mataric and the robot Toto have demonstrated that map can be built based on simple landmarks perceived by poor sensory and coarse position estimation. It was also shown that a map can be built in a distributed manner. The system lacks the option of introducing a priori knowledge as well as the ability to communicate useful information to other systems. The information is totally "personal" and has meaning only to the robot itself. Planning is limited to paths the robot has already followed and any "creativity" if at all, is based on random wander around behaviour used to explore the world.

2.5.6 AuRA - The Supermarket

AuRa - Autonomous Robot Architecture is an ambitious experiment to integrate all that exists in the domain of autonomous mobile robotics. Ron Arkin [Ark 90] has developed a system that consists of reactive behaviours (which he calls schema), classic control, high level planning, explicit knowledge representation, distributed knowledge representation, long term memory for a priory knowledge and short term memory for local perception knowledge representation.

Like in a supermarket, a large variety of basic behaviours is available, from which the planning subsystem chooses the adequate behaviour. The choice of the behaviour is based mainly on the plan, but is effected by the current perception as well. The approach used in AuRA for developing navigational techniques is as follows. First the motor behaviours required for a specific navigational task and domain are developed and tested. Next the adequate perception strategies and treatment are developed in order to supply the information required in mission. Finally, the planning and knowledge representation are updated to include the new possibilities.

2.5.6.1 AuRA System description

AuRA consists of five subsystems that were designed to provide navigation capabilities over a wide range of problem domains.

The perception is the gateway for all the sensory data into the system, shunted to the motor schema manager where the perception schema processes the information to be used by the reactive motor behaviours and to the cartographer for the construction of

local world models. Specific information pre-processing and flow control occur in the perception subsystem.

The cartographer is responsible for the acquisition and maintenance of both the a priori knowledge which is stored in the long-term memory and the local temporary perception which is stored in the short-term memory.

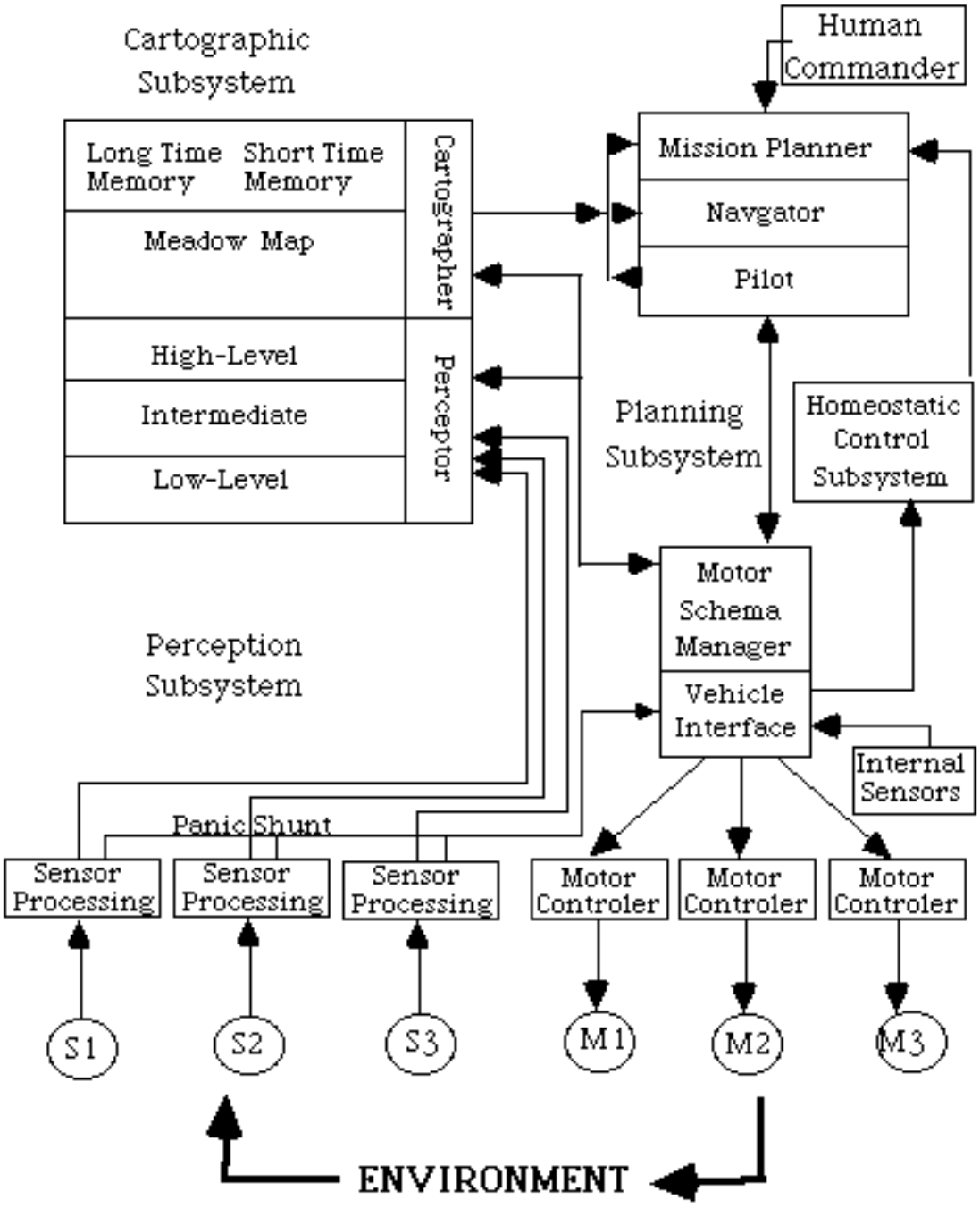


Figure 2.7 The AuRA architecture

The planning subsystem consists of hierarchical planner that generate the plan and a distributed reactive plan execution system. The hierarchical planner generates a plan which comprises linear legs and intermediate goals based on the global world model which is contained in the long-term memory. Each segment is translated into a sequence of motor action and perception schema. The planning subsystem includes the motor schema, which is the collection of all the possible behaviours of the robot, as well.

The motor subsystem translates the velocity vector to physical motor commands.

The homeostatic control monitors the internal robot conditions. It provides the relative information to the planner and protects the inner system from damage.

2.5.6.2 Discussion

The system is an attempt to integrate the contradicting approaches, classical hierarchical planning and reactive execution. It suffers from two severe disadvantages. The global world representation is human made which cannot be produced by the system itself hence is rigid and non dynamic. The behaviours and perception schema are domain dependent, new behaviours should be developed to match any unexpected change in the world.

The ensemble of behaviours is limited by the perception schema abilities that require an exact identification of landmarks to enable behaviour's execution. The plans are rigid and the selection of action is pre-dictated hence reactivity and flexibility are limited to local problem solving. This approach has two remarkable advantages:
(1) The planning is done within high level presentation and results in choosing the proper schema and the conditions for their execution.
(2) The execution is independent of the planning, no exact planning (in metric sense) is needed.

2.6 Project MARS (Mobile Autonomous Robot System)

This work was done within the frame of the MARS project in the I^3A (Institut d'Informatique et d'Intelligence Artificielle) at Neuchâtel University. Project MARS is an effort to develop behaviour based autonomous intelligent system embedded in mobile robot. We define autonomy as the capability of a system to use at any time the actual circumstances to serve its purpose (survival for biological systems, any functionality or role for artificial systems). The definition of autonomy requires a compromise between: (1) behaving in a situated way, that is in the context of the particular, concrete circumstances, otherwise it would not be able to deal with the dynamics of the world and (2) behaving in order to ensure its survival or its role otherwise it would just be driven by the environment and, therefore, would not be autonomous.

A three level architecture was proposed and developed to realise that compromise, figure 2.8 describes it. The three levels communicate via a common blackboard on which information is written and collected by the robot, the behaviours and the cognitive level. This system was implemented on SUN work stations network and the Nomad 200 mobile robot.

Level 0, the physical level, consists of the Nomad 200 mobile robot system. A description of this system is given in chapter 6. The robot writes the sensors' output on the blackboard and reads from it the next command to be executed.

In level 1 we find a collection of behaviours that were developed in I^3A and IMT (Institut de Microtechnique). Each behaviour is a stand alone asynchronous close loop process that accesses the blackboard directly and in the case of the vision based behaviours also the vision system. Each behaviour can be stimulated by the sensors'

input. The behaviours communicate to the cognitive level their state of stimulation via the blackboard. When stimulated each behaviour generates a command that is written on the blackboard.

The cognitive level is based on a sensory-motor graph. It judges its situation based on the ensemble of the present stimuli and activates the appropriate behaviour-robot connection that will lead the robot toward achievement of its goal.

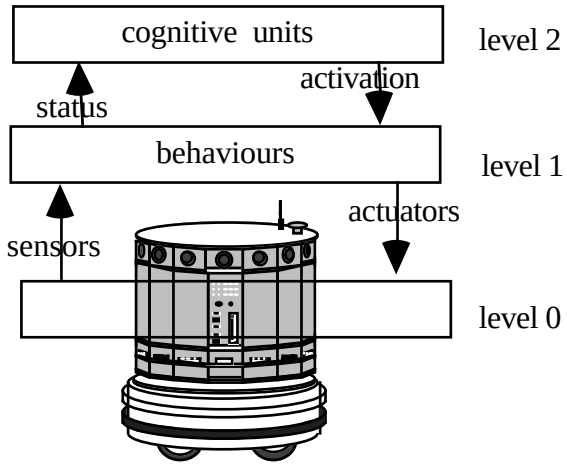


Figure 2.8 The three levels' architecture

2.6.1 Sensors and Vision System

2.6.1.1 Sonar and Infra Red

Sonar for long range measuring and IR distance detectors for short range are the Nomad 200 built in sensors for information about its environment. The sensors are arranged symmetrically around the robot body, 16 of each type. Based on these sensors two behaviours were created: wander around and follow the passage.

2.6.1.2 Vision Devices

Two vision devices were developed by the IMT: a Vision by structured light device using laser and a Landmark vision system. Based on these vision systems several behaviours were created. Those are described in §2.6.2 below.

2.6.1.2.1 Laser range by structured light

The device uses the principle of structured light to derive the geometry of a profile in front of the robot. A laser device projects a light plan oriented ahead of the robot. The developed system uses the profile of laser light to measure the environment 1 m ahead of the robot. The output is a laser range profile given by a list of 3D segments belonging to the ground.

2.6.1.2.2 Landmark vision system

This active vision system uses a light source coupled to a camera to enhance the detection of reflecting landmarks distributed in the environment. The bright landmarks

are detected, labelled and tracked in a dedicated Transputer system that produces the time sequence of labelled landmarks at an approximate rate of 15 Hz.

2.6.2 The Behaviours

2.6.2.1 Wander Around.

The wander around behaviour was developed in the IMT. The robot moves straight ahead and changes the direction only when an obstacle is detected in front of it. The new direction is such that the new move is away from the obstacle. Here, an obstacle is a configuration of radial range field detected by the infrared sensors.

2.6.2.2 Follow a Passage.

The behaviour we are using consists in going in the direction of the general orientation of the free space computed from local sonar and IR readings. The readings are integrated in a local map of the configuration space and a skeleton is extracted based on Voronoi graphs (detailed description is given in chapters 4 and 6). Four independent behaviours based on the four principal directions of the space where developed, namely: follow the link to the north, to the south to the east and to the west. Whenever skeleton lines can be extracted, the behaviours coincide with their direction are stimulated.

2.6.2.3 Vision Based Behaviours

Go Toward

This behaviour was developed at IMT. It moves the robot towards a landmark. When several landmarks are visible, the move is towards the landmark just ahead of the robot. The behaviour is no longer stimulated when the landmark is near to the robot. In this behaviour, landmarks are visible spots, detected, labelled and tracked by the landmark vision system.

Go Along

Developed at the IMT. It moves the robot along extended obstacles like walls, keeping a constant distance to them. This behaviour comes in several flavours depending on the sensing device used for its implementation and the preferred wall following direction. Regarding the implementation, a first one is based on the radial range profile from infrared, the other comes from an interpretation of the laser range profile.

Go Along Left, Go Along Right

These behaviours are specialised forms of Go Along. Whereas any direction of following is possible with Go Along, these two new forms have forced following directions.

Push Box

This behaviour was developed at IMT. It is stimulated by an object near to the robot. It moves the robot towards this object and upon collision, continues its move by pushing the object straight ahead. The robot's moves are controlled to keep the object on the straight line. Here, object detection for moving toward it and for controlling the pushing is based on radial range field detected by the infrared sensors.

Homing

Developed in IMT. On activation, it moves the robot to a fixed location and orientation with respect to two landmarks. It is stimulated on detection of two appropriate landmarks detected by the landmark vision device.

2.6.3 The Cognitive Level

This level was developed by Miguel Rodrigez of the I^{3}A [Rod 94 a & b]. Its heart is the sensorimotor graph. This graph represents the relations between sensory states and action. Figure 2.9 shows a working space and its corresponding sensorimotor graph for a robot that possesses the four behaviours of following passage. A sensory state is defined as a set of stimuli and in the case of MARS it is equal to the set of stimulated behaviours. Hence a sensory state defines unambiguously the set of behaviours that can be selected. The sequence of behaviour selections and the resulting sensory state constitute the unique source of knowledge for creating the graph and controlling the system. A node in the graph is defined by the current sensory state and its history (the sequence of behaviour-sensory state that led to the current state). This approach, proposed by Gat and Rodrigez [Gat-Rod 92] enables the creation of an unambiguous graph based only on the stimuli's states. Three modules of control were defined and developed: the Learner, the Localiser and the Planner.

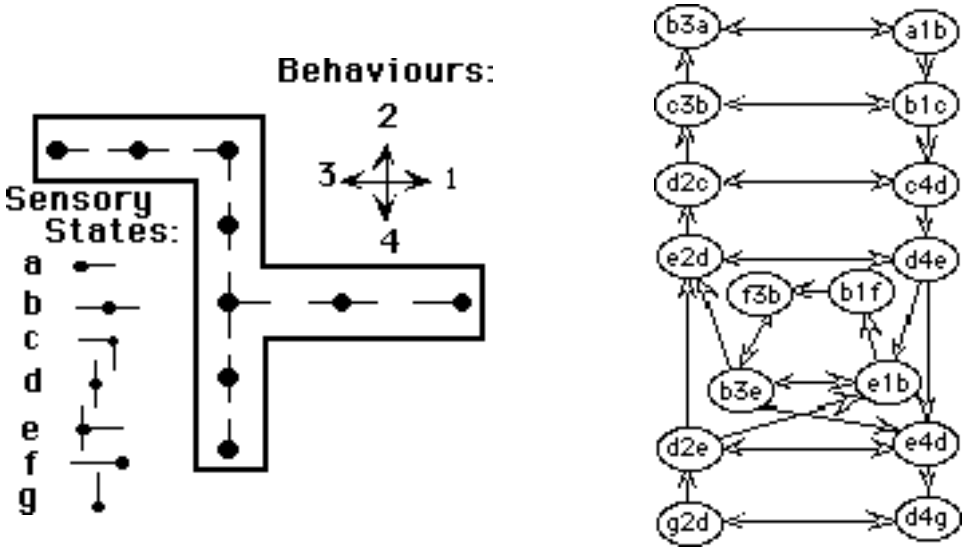


Figure 2.9 A working space and its sensorimotor graph

2.6.3.1 The Learner

The learner finds causal regularities (recurrent sequences in the history) and organises the information into a topological graph, exploiting sensorimotor neighbourhoods as exposed in [Gat-Rod 92]. So, the nodes are the perceptive states composed of sub sequences of sensorimotor states and the edges the behaviours that conduct the robot from one state to another. It selects the behaviours in order to explore the unknown trails.

2.6.3.2 The Localiser

The localiser finds in which node of the topological graph the system currently is. When lost, it selects the behaviours in order to experience a known sensorimotor sub sequence (path) corresponding to a node of the topological graph.

2.6.3.3 The Planner

A goal is defined for the robot as a desired sensory state or a desired node in the graph. The goal node is activate and spreads the activation to it neighbours which

spread the activation to their neighbours and so on in process known as activation propagation. In each stage the activation is reduced so the farther the node is from the goal the less it is activated. The controller selects at each state the behaviour that conducts to the most active neighbour.

2.6.4 Discussion

This approach like the one presented by M. Mataric gives a solution to the problem of mapping and planning based on behaviours and stimuli. It creates a world model by finding and mapping regularities and their connections in terms of behaviours. It suffers from the same drawback, it can't use a priori data and based only on learning and experimenting.

2.7 What Next

The current navigation systems suffer from several inherent drawbacks that prevent good exploitation of the advantages residing in the two contradicting approaches. The exact planning and metric navigation assure complete control of the robot trajectory. A priory knowledge is easy to introduce, and mapping abilities enable model's updating by the system itself. On the other hand the reactive behavioural based approach provides the ability to deal with the unexpected and the surprises of the real world. It although eliminates or reduces to minimum the dependence of the system on a world model.

While the classic navigation requires a complete, totally accurate and detailed model, the reactive approach, when using a planning, can use a vague and incomplete model leaving the exact local "planning" and execution to the active behaviour itself. The description of a trajectory is a sequence of behaviour. It is shorter and leaves more freedom to the behaviour to react to the actual conditions. The combination of the two approaches will benefit of the advantages of both worlds. The problem of the actual system is the lack of ability to pass from a physical description of a world (like map) to model consisting of behaviours and reaction. To overcome this problem there are systems that can create a model only by following and memorising the actual robot behaviours [Mat 89 90] [Rod 93].

Another solution is to create directly a stimuli based model of the world [Ark 90] [Lev-Law 90], where the stimuli origins are mapped manually in geometric or topological model. The next chapters are dedicated to a theory and navigation system's description that combines the two approaches and provides, based on a single meta behaviour a complete description of the world as well as the means for reactive following of a planned trajectory.

3. Behaviours Mapping

3. Behaviours Mapping

3.1 Introduction

For any system we can define a space that includes all the system's states. We call this space the *state space*. Each point of the space represents a different state of the system. The configuration space is an example of such a space that covers all the geometric position of an n -degrees-of-freedom system in an Euclidean space [Lat 91]. Another example is the phase space in which each point represents the kinematics state of the system. Maturana and Varela [Mat-Var 80] [Bourg-Mat 91] use a topological space to describe the possibilities of a living system. A subspace of the states' space is used to describe the "permitted" domain of the system, the free space in the case of the configuration space or the viability subspace in the case of Maturana-Varela-Bourgine. We name this subspace the *subspace of existence* or the Viability *subspace* for a system.

One can regard the system behaviour as the results of a function (or functions) that the system uses to change its state and the state of the world. By analysing the behaviour of the latter in the existence subspace one can obtain an interesting result. The existence subspace can be divided into zones in which the function(s) is continuous and to zones in which the function is not continuous. Those zones of non continuity can be regarded as zones of decision where the system or its operator should select one of the possibilities, if there are any, or activate another function. When these zones are retracted to points and the continuous zones in between are retracted to lines connecting these points, a graph representation of the existence subspace is obtained. The advantage of such a representation lies in the control of the system, in continuous zone it is not necessary to consider or calculate any option, only following that zone by the right behaviour and a local optimisation are required. In the next paragraph the mapping of single behaviour is discussed. This theory of behaviour mapping can be generalised to a multi behaviour system. In this case we define the term behaviours' equivalence. Behaviours are defined equivalent if at a given state their effects are topologicaly equal, that is if they change the system situation to an equal topological state.

3.2 Single Behaviour Mapping

Let consider a system that exercises only one behaviour that conforms with the existence constraints. Hence from any place in the existence subspace the behaviour will conduct the system to a place inside that subspace. We define as trivial the case when that behaviour is continuous all over the subspace of existence. A more interesting case is when there are points or zones of non continuity and bifurcation of the behaviour in the existence subspace. We can use these zones and points of non continuity as landmarks and describe the existence subspace of a system as graph in which these zones are the nodes. Whenever the behaviour connects one zone to another their representations in the graph are connected. A formal definition is given below.

Let :

W be the topological space that represents the world.

V be a connected subspace of W that represents the subspace of existence for the system.

$f: V \to V$ an application defined over V that represents the system behaviour.

f is continuous at $v \in V$ iff:

(1) $f(v) = u, u \in V$

(2) for any neighbourhood of u, N_u, we can find a neighbourhood of v, N_v, such as $Nu \supseteq f(Nv)$

Let d be an open set of non continuous points such as

(1) $v \in d$ if $f(v)$ is not continuous

(2) d is a connected set

The closure of d cl(d) is composed of continuous points such as that at any neighbourhood of $v \in$ cl(d) there is at least one point of d.

Let $D = \{d_1 \ldots d_n\}$ be the collection of the non continuous sets of V under f.

Let $f^n(v) = f^\circ f^\circ f^\circ \ldots^\circ f(v)$ describes a sequence of n iteration of the behaviour f.

The set d_j is said to be directly connected to d_i by f^n iff

1. there exists $x \in cl(d_i)$ such as $f^n \in cl(d_j)$.
2. $f^1(x) \ldots f^{n-1}(x)$ are continuous under f

We define a path from d_i to d_j as $l(x | x \in d_i, f^n \in d_j) = \{f(x) \ldots f^n\}$

Let $p_{ij} = l(x_1) \cup \ldots \cup l(x_k)$ where $l(x_1) \ldots l(x_k)$ are all the paths from d_i to d_j. We call p a passage from d_i to d_j.

Any d from which there are two or more passages is a bifurcation zone.

V now can be described as a directed graph. The non continuous sets $\{d\}$ are the nodes of the graph which are connected according to the passages defined above.

Now we can plan the future of the system based on the graph. The nodes play quite a significant role, they are the decision points for the system. On a passage the behaviour will lead the system to the same neighbourhood independent of perturbations or deviations on the way. In the non continuous zones the situation is quite different, any deviation or little perturbation can lead the system to a totally different situation. We call these zones decision zones because a decision can be taken here, a decision that will have quite an influence on the future of the system. A mobile robot navigation system that was developed based on this theory and benefits the advantage of navigation and planning using only one behaviour is presented in the next chapter.

3.2.1 An example of uni-behaviour

We bring here as an example go-toward behaviour. The behaviour was developed by a team of the Institute for Microtechnology of the University of Neuchâtel (IMT). A vision system tracks a reflecting target and the robot moves toward that target. When the target is too close the robot stops. The angle of vision is 90 deg.

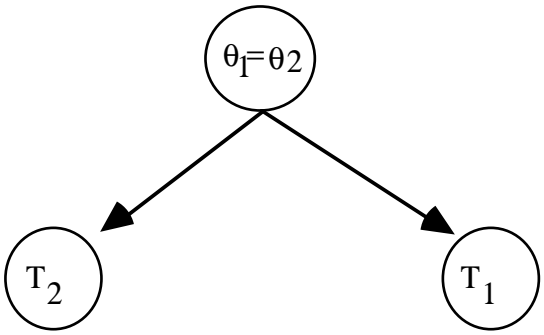


Figure 3.1 The graph of go-toward behaviour with two targets

When there is only one target the non continuity zones are:

1. when the distance from the robot to the target is less than the minimum and the robot stops
2. When the angle between the centre line of the camera and the target is out of the ± 45 deg. range the robot looses the target and stops

For a distance bigger than the minimum and within the ± 45 deg. the behaviour is continuous.

When a second target is introduced into the scene the situation changes. The robot is programmed to follow the target which is closer to the centre line. The line defined by $\theta_1 = \theta_2$, where θ_1 and θ_2 are the angles to the first and the second target respectively, becomes a bifurcation zone. The graph describing the subspace of existence for the robot under the go-toward behaviour is given in figure 3.1

3.3 Multi Behaviours Mappings

The analysis of the subspace V can be generalised to a multi behaviours' mapping. Here we deal not only with bifurcation of a single behaviour. To define bifurcation points for several behaviours first we have to define the term *b-equivalence* that stands for behaviours' equivalence. We use here the term b-equivalence when different behaviours give the same results when exercised at the same situation. (Note that this use of equivalence is different from the normal use of the term equivalence in mathematics). Two behaviours are considered to be b-equivalent at a certain part of the existence subspace if when exercise at any point of this part they bring the system to the same topological state. Hence any behaviour of the system that follows gives the same results. The points of bifurcation are the points where the behaviours' results separate. These points are used to create a graph description in which the nodes represent the bifurcation points and the branches are defined according to their connection by the behaviours.

3.3.1 B-Equivalence (Behaviours' Equivalence)

This section represents formally the multi behaviours mapping as discussed above.

Let :

- W be the topological space that represents the world.
- V be a non empty connected subspace of W that represents the subspace of existence for the system.
- $f, g: V \to V$ be two mappings representing behaviours defined in V.
- V' be the collection of the point of V where both f and g are defined.
- F be the collection of all the applications that describe the system's behaviours

f and g are b-equivalent at $x \in V'$ iff:

1. $f(x)$ and $g(x)$ are continuous
2. There exist n and m such as for any $i \le m$ and any $j \le n$ and for any $h \in F$
   - 2.1. $g^j(x)$ and $f^i(x)$ are continuous
   - 2.2. $h^\circ g^j(x) = h^\circ f^i(x)$.

Let $\{d_{fg}\}_e$ be the open sets in V' in which f and g are not b-equivalent

Let $\{d_{fg}\}_c$ be the open sets in V in which f and/or g are not continuous

We can now define the bifurcation zones as:

$$\{d_{fg}\} = \{d_{fg}\}_e \cup \{d_{fg}\}_c$$

in which either at least one of the behaviours is not continuous or where the behaviours are not b-equivalent.

The set d_j is said to be directly connected to d_i by f and g iff

1. there exists n, m and $x \in cl(d_i)$ such as $f^n(x) \in cl(d_j)$ and $.g^m(x) \in cl(d_j)$
2. g and f are continuous and b-equivalent along the path

We can now describe the existence space of the multi behaviours system by a graph. The nodes of the graph represent the bifurcation zones. The branches are according to the direct connectivity defined above.

An example for such a system is a corridors' navigator mobile robot that possesses the three following behaviours:

1 follow-the-centre of the corridor
2. follow-the-right-wall
3. follow-the-left-wall.

As long as the robot is in a corridor the three behaviours are b-equivalent. When the robot arrives to a corridors' intersection the follow-the-centre behaviour becomes non continuous, the robot can follow anyone of the corridors that exit the intersection. The other two behaviours are not equivalent. The follow-the-left-wall will follow the most left corridor while the follow-the-right-wall will follow the most right corridor. Any selection of a behaviour will give different results, as a consequence the corridors' intersection becomes a decision zone (see the example in 3.3.2 below).

3.3.2 Example: Corridors' Navigation

We take for example a simple system, a mobile robot equipped with two sonar cells. The robot moves in constant speed and its control parameter is the speed of rotation dq/dt. The schematic description of the robot is given in fig. 3.2 below

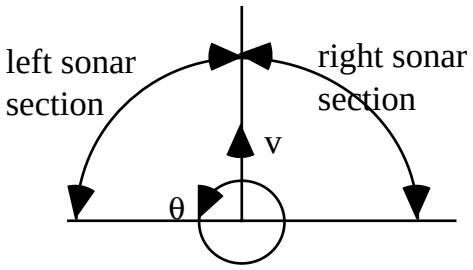


Figure 3.2. Robot's schematic description

Let consider the corridor navigator from §2.2.1.

We can describe the three behaviours as the following functions:

1. f_1 $dq/dt = r_o - r_R(t)$ will result in follow-the-wall on the right
2. f_2 $dq/dt = r_L(t) - r_o$ will result in follow-the-wall on the left
3. f_3 $dq/dt = r_L(t) - r_R(t)$ will result in follow-the-centre

Where $r_L(t)$ and $r_R(t)$ are the left and the right sonar readings at time t and r_o is a control parameter that specifies the desire distance from the wall.

In figure 3.3 we see part of a working space. In the passages p_1, p_2 and p_3 the three behaviours are equivalent. In the decision zone d f_1 and f_2 are not equivalent but for not too big r_o they are continuous. f_3 is not continuous because when arriving from p_1 it can continue either to p_2 or to p_3 depending on the exact orientation of the robot.

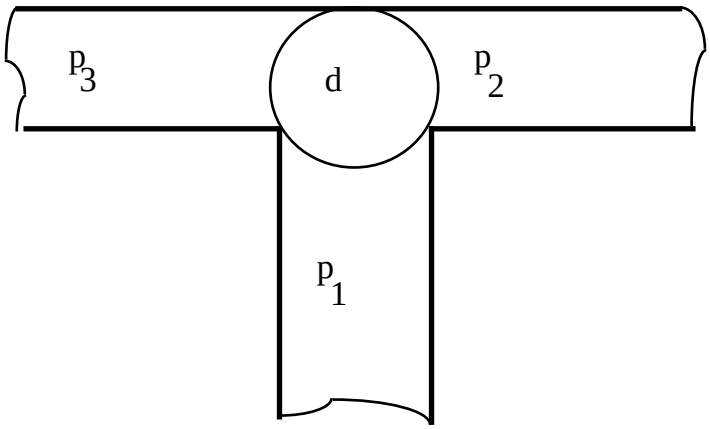


Figure 3.3 corridors and their decomposition to passages and decision zone

In figure 3.4 we see the graph representation of the working space. The junction is created by the non-b-equivalence of behaviour 1 and 2 and the bifurcation of behaviour 3.

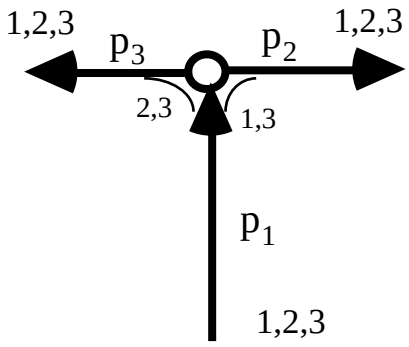


Figure 3.4 The graph representation of the working space

3.4 Meta Behaviour Mapping

In this section we generalise the notion of a behaviour by defining the *Meta Behaviour*, which is the result of looking at the system's actions as being one behaviour even if it results from the composition of several behaviours. In the following paragraph we lay the mathematical foundation for such a definition.

Let $F: V \rightarrow V$ F : V $\rightarrow$ V be an ensemble of behaviours $\{f\}$.

Let $D = \{d\}$ be the finite ensemble of the sets d in V such as:

$x \in d$ iff

(1) there exists at least one $f \in F$ that is not continuous at x.

or

(2) there exist at least f_i and f_j, i ≠ j which are not b-equivalent at x.

Let $P = \{p\}$ be the ensemble of passages defined by one or more of the behaviours $\{f\}$.

Let $G_F(V)$ be a graph in which the nodes represent the closed sets $\{d\}$ and the branches represent the passages $\{p\}$ that connect them. Such a graph is a unique representation of V under F.

We call F a *meta-behaviour*, a behaviour that consists of all the behaviours $\{f\}$. F is not continuous in D but continuous elsewhere in V.

3.4.1. Meta-Behaviour and Control

We can look now at a system that possesses an ensemble of behaviours B that it executes in a working space U. Let assume that U is decomposable to decision zones D and passages P under the behaviours B. Looking at the behaviour of the system within the frame of the graph description of the working space an interesting phenomenon emerges. The system demonstrates a behaviour of following the passages from one decision point to another. This behaviour which we denote meta-behaviour is independent of the behaviour actually executed by the system as long as it follows the passages.

A definition of a trajectory can be given as a sequence of passages to be followed. Following a given path is reduced to selecting at each decision point the behaviour that will follow the next passage. Each of the equivalent behaviours will be adequate.

To navigate, the system should be able to recognise the decision points and to select the appropriate behaviour while in the decision zone. The selection of the behaviour can be made based on local parameters or using a plan. In the latter case the plan is not the classic plan as a program but more like a general frame that gives the system the criteria by which it makes the selection.

3.4.2 Project MARS in the Eyes of Meta-Behaviour

When we inspect the cognitive (control) level of project mars (see §2.6) we see that each node is defined by the appearance of a new set of stimuli. A new set of stimuli means a new set of behaviours which in the terms of meta-behaviour is a decision point. The nodes of the sensorimotor graph are connected by behaviours, each connection is defined by one or several behaviours. Hence the sensorimotor graph can be created and analysed by means of the meta-behaviours.

3.5. Discussion

The theory enables to bridge the gap between the classic planning approach and the reactive execution. A reactive behaviours based system can be controlled and can realise a planned trajectory. It furnishes a description of the world based on and in terms of reactive behaviour.

The description of a trajectory is very simple. It uses only the notions of passages and decision points without even specifying the exact behaviour(s) that should be selected to realised the trajectory. The immediate execution control will choose the right behaviour to follow the desire passage. Hence it leaves a lot of freedom and flexibility to the executing system. We give here an answer to the problem posed by Agre and Chapman [Agre-Chap 90] e.g. a way to create a plan as an advise.

When on a passage the system is free of the decision making problem and can use the liberated resources to fulfil other tasks like planning or learning. It is only in decision points that the system will use its capacity to make a decision.

To create the graph and find the point of decision where the meta behaviour bifurcates we can analyse the space as it is presented in the next chapter or the system can identify them on the spot. The latter can be done by giving the system the ability to define these points directly from its input, for example using a neural network. Another

option is to test the results of all the behaviour that can be activated at any point as it is done in the navigation system of Miguel Rodriguez [Rod 94].

4. Reactive Navigation Based on A Simple World Modelling and One Behaviour

4. Reactive Navigation Based on A Simple World Modelling and One Behaviour

4.1 Introduction

In this chapter we introduce a navigation system for a mobile robot. This system is based on the theory that has been developed chapter 3. This navigation system uses only one behaviour that we will show to be sufficient to answer the three basic questions of navigation.

The navigation system is divided into two subsystems (see figure 4.1 below):

1. The *Global* subsystem creates a graph from a map of the working space and generates plans
2. The *Local* subsystem based on sensors input takes care of self localisation and reactive execution of the plans

We base the global navigation subsystem on the a Voronoi diagram representation of the working space. Each closed connected occupied set is a Voronoi cell. A medial axis transform of the free space generates a skeleton which lines are the borders of the Voronoi cells. We call the skeleton lines' intersections *Junctions* and the skeleton lines segments that link them we call *Links.* A graph based on the J-L (Junctions & Links) model gives a symbolic description of the working space.

The local subsystem perceives the world in terms of a local J-L model. It exercises one behaviour which is following the centre of the free space ahead. This behaviour results in a *Follow-a-Link* behaviour.

The Follow-a-Link behaviour bifurcates and discontinues exactly where the skeleton lines bifurcate and discontinue. Hence the graph based on the decision points of the Follow-a-Link and the graph generated by the J-L model are identical.

The answer to the first question "Where am I" is given by a robot that can identify locally the J-L model and correspond it to the graph.

"Where are other places" is answered by their corresponding representation in the graph.

The third question "How do I get there" is answered in two stages: (1) using the graph for planning and (2) following the appropriates links in the working space.

In §4.2 the Voronoi diagram and the graph representation of the working space are defined. §4.3 gives a brief description of the planning as a graph search. A planning approach is developed and discussed in chapter 5. The execution subsystem is described in §4.4.

4.2 Definition of the Generalised Voronoi Graph

In this part we present and discuss the formal definition of our world modelling. The mapping which is discussed in this section maps the free space of a multi-dimensional space into a graph. Connectivity and relative order are invariant substances under the mapping. It is a two steps mapping, the first step is a medial axis transform which maps the free space into a multi-dimensional skeleton, the second step maps the skeleton into a graph keeping a certain order of the branches at each node.

Even though navigation in particular is taking place in 2D or at the most 3D Euclidean space, we impose no such limitations on the dimensions and the metric in our mathematical treatment. The basic approach is adequate to deal not only with navigation problems but also with problems whose solution can be represented as a free space in a configuration or phase space.

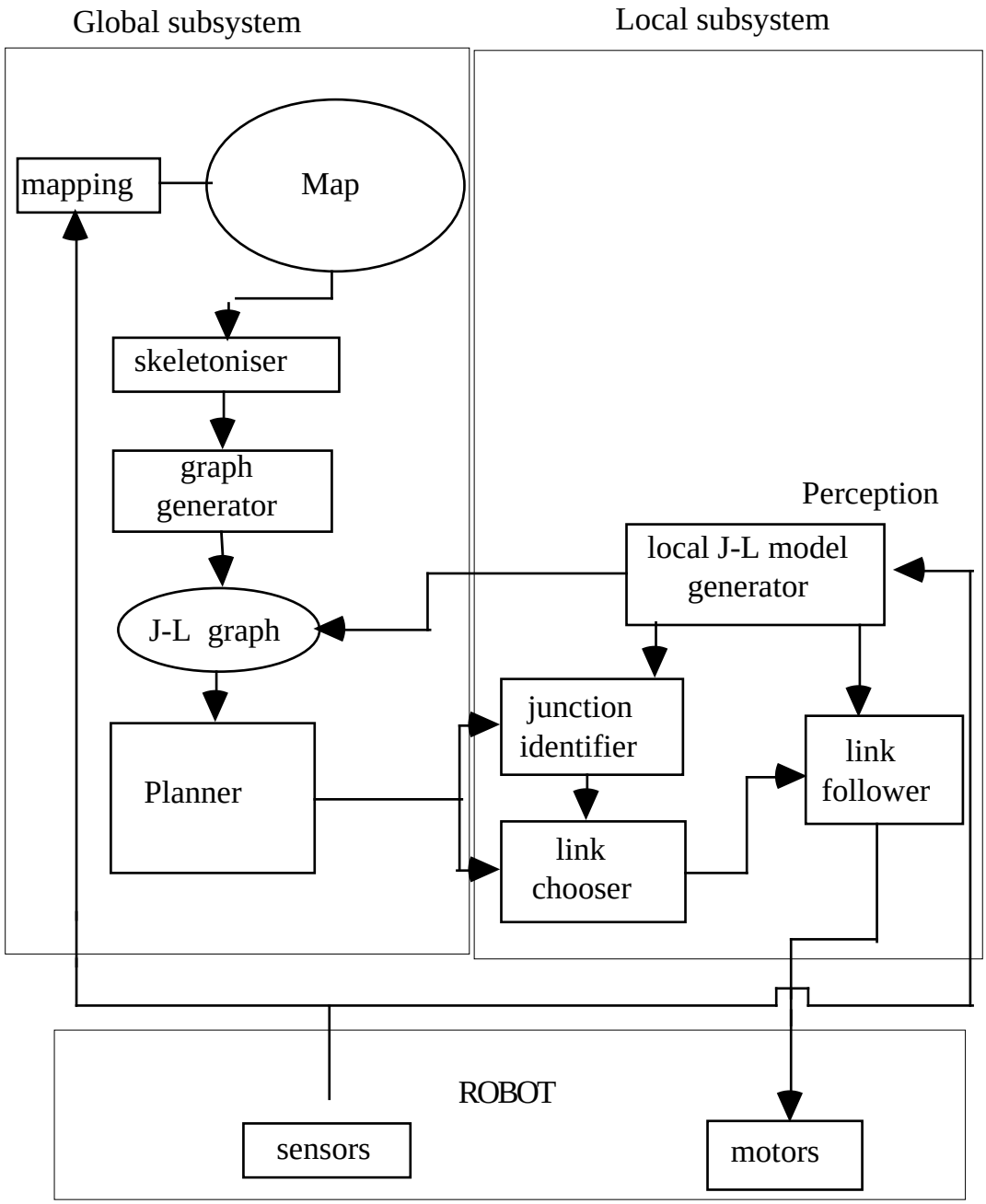


Figure 4.1 Schema of The Navigation System

4.2.1 The Skeleton

The medial axis transform is well known and frequently used in computer vision [Ball-Bro] [Pav 77] and in robot navigation system mostly under the name Voronoi graph [Bar-Lat 89] [Sch-Shar 88] [Brait 84]. The definition used hereafter is a somewhat different from the one used by Pavlidis [Pav 77] or others [Bar-Lat 89][Sch-Shar 88] imposing no restriction on dimensions or metrics.

The medial axis transform creates a skeleton representation. Starting from n dimensional space we reduce the free space's degree step by step. At each step we create n dimensions spheres that contain only free space points and touch the borders of the free space at two different points. By connecting the centres of all the spheres that met these conditions we create a sub space of one degree less. Continuing until arriving to a one dimension subspace creates the skeleton.

In n dimensional space the extraction of the skeleton can be defined by the following iterative process:

1. Let M be an n dimensional metric space and M_F the collection of all the free points within M.
2. Let $M_B{}^i$ be the set of $M_F{}^i$ boundary points, at each step i ($i = n \ldots 2$)
3. Let S_i be a set of i dimensional sphere in M.
4. A sphere s belongs to S_i if and only if :

4.1. No point inside s belongs to $M_B{}^i$.

4.2. At least 2 points on s surface belong to $M_B{}^i$.

5. The centres of all the spheres which belong to S_i define a sub-space of degree $i--1$ in M which we denote as $M_F{}^{i-1}$.
6. Let $M_B{}^{i-1}$ be the set of $M_F{}^{i-1}$ boundary points.
7. The 1 degree sub-space which is reached at the end of an iterative process of steps 2 - 6 is the skeleton of the free space $M_F{}^i$.

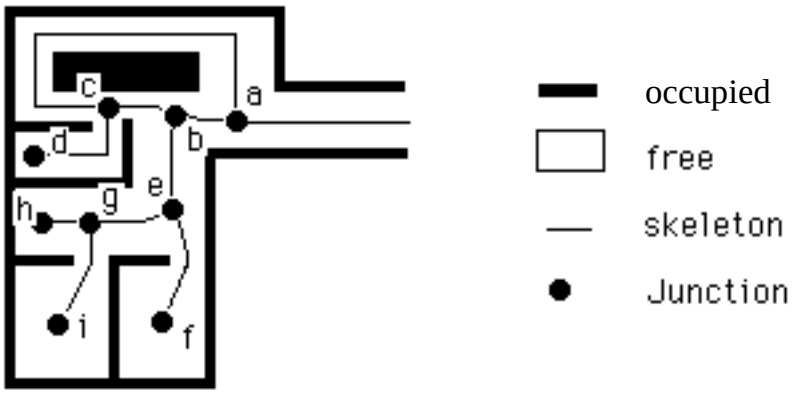


Figure 4.2 2D map and its skeleton. The junctions have been already identified and marked.

4.2.2 Junctions Links and Graph Representation

In the skeleton we define a junction as a point of intersection of two skeleton's lines and a link as a line segment of the skeleton. A formal definition is given below:

1. Let X_S be the set of all the points belonging to the skeleton.
2. A point x is a junction J if and only if x belongs to X_S and in any neighbourhood of x there are at least 3 points which belong to X_S .
3. A link L is a line segment of the skeleton that:-
   3.1 All its points belong to X_S.
   3.2 At least one end point is a Junction.
4. A Link with only 1 Junction is defined as a dead-end link
5. A Link with junctions at both ends connects them.
6. Being a point of non continuity a deed end is treated as a junction for mapping and planning.

Under the behaviour of following the skeleton line the junctions becomes bifurcation points where the behaviour is not continuous and the links are passages to different junctions.

It should be noted that any link can bee looked at as a junction where an execution of the inverse behaviour, meaning following the skeleton line in the opposite direction, will give different results. Since it changes basically nothing in the graph representation of *M* we overlooked for mapping purposes and introduce the notion of *inverse* behaviour.

On the basis of the junctions and links we can define a graph in which the junction and the dead-end points are represented by nodes. The connections between the nodes are according to the links and represented by branches (see figure 4.3). The transformation from the multi-dimensional skeleton to the graph is unique but not a one-to-one mapping. Only one graph exists for each multi-dimensional skeleton but the opposite is not necessarily true. It is obvious that the graph keeps the connectivity of the skeleton and therefore the connectivity of the original free-space.

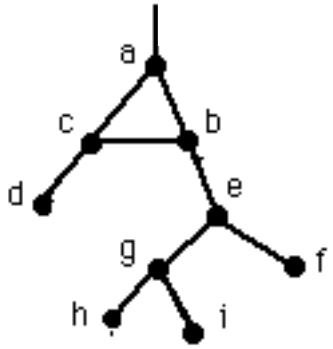


Figure 4.3 The graph representation of fig 4.2

We order the links which are connected to a junction in a certain cyclic order which can be reconstructed independently. For example: an order which is based on the skeleton projection on the plan generated by e_1 e_2. In case of ambiguity we will be helped by the projection on e_2 e_3 plan and so on (e_1...e_n the unit vectors which create the space M).

4.3 Path Planning

The path planning answers the first part of the third question a navigation system should answer. In this section we discuss the generally the path planning within the frame of the J-L model. A new method for planning within the Voronoi diagram graph representation is presented and described in details in the next chapter.

4.3.1 Path planning as a graph search

The path planning is processed within the graph frame. Planning a path becomes a “simple” task of searching a graph to find the path that connects the starting node to the target node. A variety of techniques for graph searching are described in AI literature ranging from simple Depth First Search to some sophisticated back jumping that also includes learning of constraints.

Most of the methods are described in any AI text book (for example see [Wins 84]). More sophisticated methods are described in the literature (see for example [Lig90] [Dech 90]). Any method can be used and the choice depends on the information we relate to the branches and nodes, the kind of required optimisation and the kind of heuristics used to control the search. In the next chapter we present a planning system that finds a near optimal path within the frame of the Voronoi diagram.

4.3.2 Two words representation of a path

The path is represented in the terms of chapter 3. Each link is a passage and each junction is a decision point. On a link the follow a link behaviour is continuous and will follow the link to its end. In a junction the behaviour is no longer continuous, it can follow any of the link. Hence a unique description of a path should give the criteria to choose the right link to follow.

Starting at known node and branch it is possible to represent any path through the graph using only two symbols, J & L. In the graph frame J represents a node and L a branch, in the real world J represents a skeleton junction and L a skeleton link. A sentence describing a path will be looked like the following sentence "JLLLJLJ..." meaning: from the starting junction you take the 3rd link to the next junction than take the first link to the next and so on. For example the sentence "LJLLJLLJLJLLJ" represents the path (a b e g i) in fig 4.2.

If we communicate such a sentence to any system that can identify junction and links in the same way, and is able to reconstruct the same links' ordering at any junction, this system is able to follow the path described by the sentence in the relevant space, requiring no other means to control its movement.

4.4 Path Following

To perform a reactive path-following the robot should be equipped with adequate perception and behaviours. A perception that enables reactive path following based on the above world representation is described in sub-section 4.4.1. In sub-section 4.4.2 we described the basic behaviour in which the robot should react to the information that it perceives. The process of reactive path-following is described in sub-section 4.4.3.

4.4.1 Perception

A general, abstract and immediate robot perception can be developed, based on the above world modelling. The robot transforms its immediate local view of the world directly to Junctions and Links. The robot operates and controls its move within this abstract world. It only has to know the location of junctions and location and direction of links relative to itself, in its own co-ordinates' frame.

This perception can compete successfully in any structured or quasi structured area of the world, and most parts of our world match this requirement. Any area that can be divided into passable and non-passable, free and occupied, or any other phenomena division that the robot sensors can detect and identify is adequate for the implementation of this perception.

a b

Figure 4.4 Perception of the immediate surrounding.

Equipped with this ability the robot perceives its environment directly in the most important phenomena to the control of its movement and navigates autonomously in the world. As we will show later this perception and representation of the world is,

alone, sufficient for navigation and movement control. We have developed an algorithm that perceives the sonar and IR reading of fig 4.4a in terms of junctions and links, fig 4.4b.

4.4.2 Basic behaviour

The robot has only one behaviour which is defined bellow:
Follow-a-Link:- while perceiving a link the robot follows it by moving along the link line in the real world
When perceiving being in a junction the robot chooses the next link to follow according to the sentence that describes the path by counting the links in the agreed order.

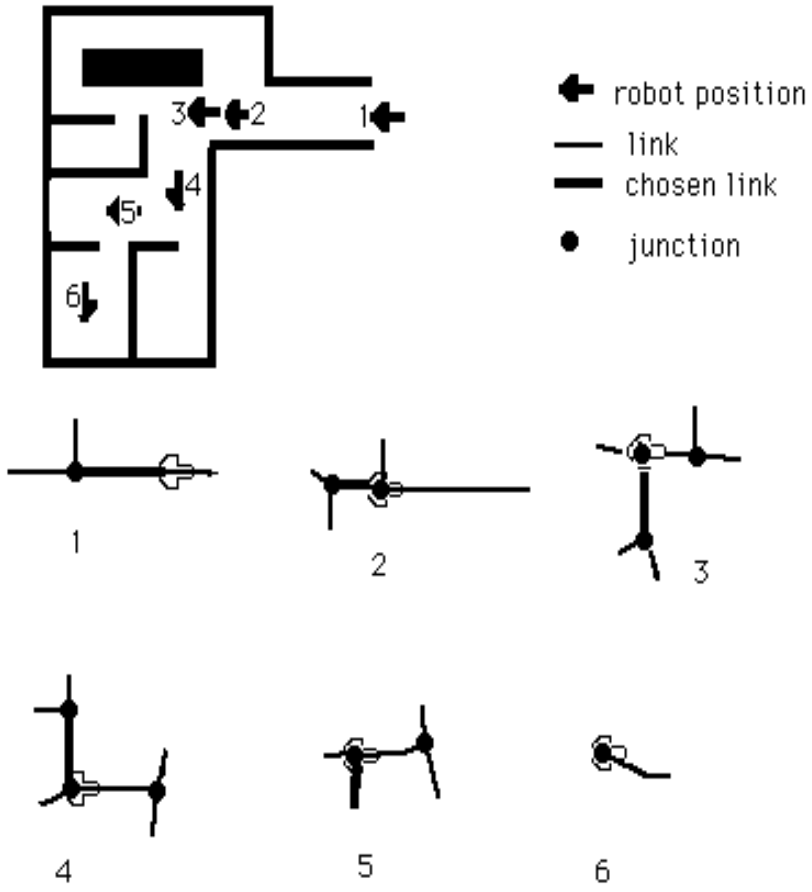


Figure 4.5 The path as a perceived sequence of images

4.4.3 Execution of path following

Starting from a known junction and link the robot should follow the link to the next junction. Perceiving the link the robot reacts in the Follow-a-Link behaviour till the next junction. Using its ability to identify links and junctions, the system identifies the junction and all the links that leave that junction in the ordering agreement and chooses the next link it should follow.

The process of link following, junction identification links identification, and link choosing will continue until the system will arrive to its target. A path for the robot is a sequence of perception images to which it reacts. The sentence mentioned above is

a symbolic description of this images sequence. Following the images in the order they are given the robot follows the required path in the world. Figure 4.5 describes the sequence of perception matching the required path. The local views are shown for each labelled point

4.5. Discussion

The navigation approach presented in this chapter shows the feasibility of *reactive* navigation based on the *behaviours mapping* even with very basic and low performances sensory. It combines two different approaches to navigation, the metric and reactive, and exploit their advantages. The planning is performed in a symbolic geometric based framework an approach well explored and quite successful. The execution is purely reactive, giving robustness and enable dealing with changes and uncertainty.

The junctions and links model can be applied almost to any kind of environment when the robot is equip with adequate sensory. For example navigation of a car on roads will require only a sensory that can detect roads. Another example is a topographic navigation that can be based on physical crest lines as the phenomena to extract the skeleton.

The graph representation enables us to incorporate many kinds of domain specific knowledge that can be related to the branches and the nodes in order to facilitate path planning in one hand and links and junctions identification in the other hand.

To deal with the real world problems (such as noise and inaccuracy) the reactive navigation approach required some more complex and sophisticated treatment which includes help from dead-reckoning and filtering (see chapter 6 for description). But not like the localisation system presented by Crowly [Crow 88a] it does not require exact geometric definition of the world. The positioning based primarily on the topology of the world and only aided by the geometry.

5. Near Optimal Path Planning for Voronoi Diagram

5. Near Optimal Path Planning for Voronoi Diagram

5.1 Introduction

The path planning system is based on the Voronoi diagram that was described in chapter 4. It has the advantage of operating directly on the graph form and results in a near (geometric) optimal path.

Geometric optimisation within the graph framework is limited to the cases where the movement is restricted to the medial lines. The Barraquand-Latombe system is a good example. For the shortest path in general this is not the case. An optimal path search for a robot can be guided by the Voronoi diagram while the actual move is performed along the medial line but not necessarily on the line. In our case the follow-a-link behaviour optimises locally the move by going directly toward the furthest point of the path that it sees at each moment.

The near optimal path finder is based on the generalised representation of the passable areas in a specific form of Voronoi graph, attaching some additional information to the nodes and the branches of the graph. A multistage recursive search of the graph is performed, eliminating at each stage most of the non shortest paths. A heuristic based on upper and lower bounds of path length is introduced to enable and facilitate a search based on A* without requiring an exact optimisation computation.

5.2 Additional information attached to the nodes and branches

Additional information is attached to the graph. This information will be used to determine the bounds of links and paths length. The way this information is used is described in the following section, here we only define this information.

To each node in the graph we attached the following information which is related to the junction the node represents:

1. Junction diameter - the diameter of the spheres whose centres creates the junction point in the skeleton.
2. Junction co-ordinates.

To each branch in the graph we attached the following information which is related to the link the branch represents:

1. Geometric description of the link.
2. Nominal length - the length of the skeleton line segment the link represents.

5.3 Upper and Lower Bounds of Link Length

The movement in the space is not restricted to a movement on the links. The optimal path that connects 2 adjacent junctions can pass anywhere along the link meaning anywhere in the segment of the free space represented by the link. The computation of the optimal path is relatively complex and requires exact geometric methods. In this section we present a method to bound the length of the optimal path along a link.

5.3.1 Link Length Upper Bound

The link upper bound (LUB) for the minimal travel distance between 2 junctions is the length of the skeleton segment which connects these 2 junctions. By definition, the movement on this line is possible therefore the length of the shortest passable path that connects these junctions will be equal to or less than the link length. Consequently, the link nominal length produces an upper bound for the shortest possible path between 2 adjoined junctions.

5.3.2 Link Length Lower Bound

We calculate the *LLB* (Link Length Lower Bound) by approximation using partitioning of the link to several sections. The order of the approximation is the number of sections used for the calculation of the *LLB* Before we introduce the method to compute the *LLB* we require the following definitions:

1. Section -
   - 1.1 let L be the link that connects the 2 junctions J_i and J_j.
   - 1.2 let P denote the n-1 degree subspace perpendiculars to the skeleton line represented by L and containing the point p belonging to L.
   - 1.3 let X_L be the collection of all the points of free space segment which is represents by L.
   - 1.4 let $X_S = P \cap X_L$.
   - 1.5 the set X_S is a section of L at the point p.
2. Order - the number of section used to compute the *LLB.*
3. *dist* (i,j) - distance between sections:
   - 3.1. let X_{Si} and X_{Sj} be 2 sections of *L.*
   - 3.2. $dist(i,j) = \min\left[dist(x_i, x_j) | x_i \in X_i, x_j \in X_j\right]$.

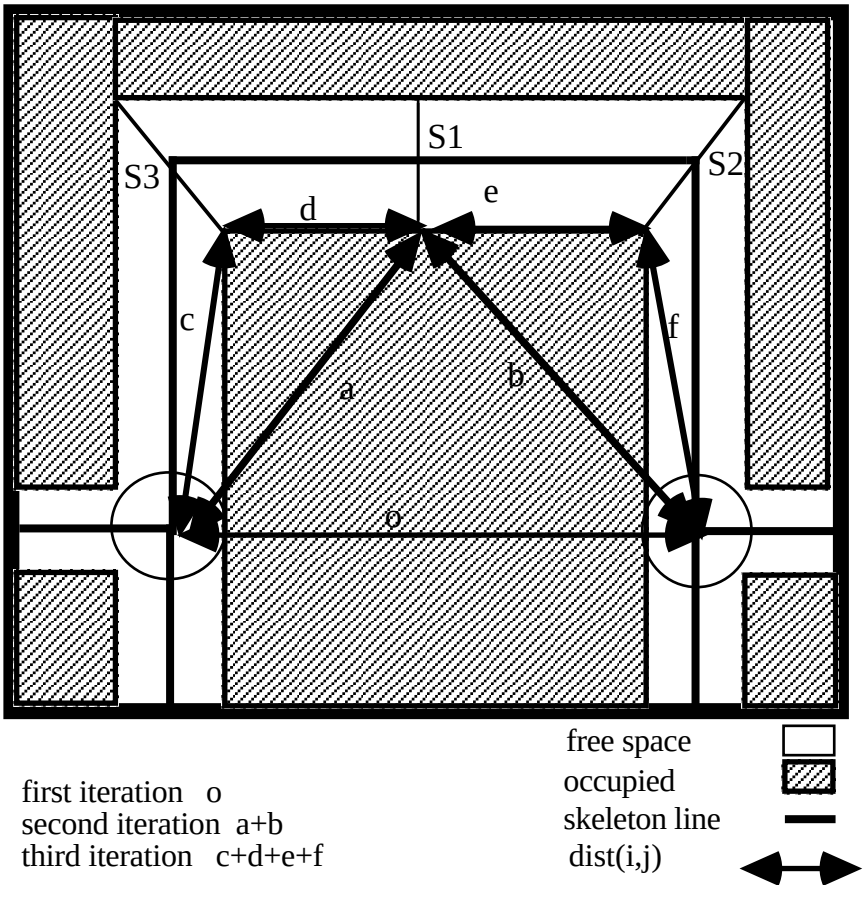


Figure 5.1 Calculation of Link Lower Bound for the link that pass through the sections S1 S2 S3.

For the computation of the *LLB* we introduce a method that produces an order depending value. Up to a certain limit, which varies from link to link, the higher the

order the closer the LLB is to the value of the shortest link length. One should notice that the LLB is a virtual path that not respects the physical limitation.

Any path passing through a link and connecting the 2 junctions at the link ends, including the shortest path, must include at least one point of any section. Consequently, the length of path segment between 2 sections can't be shorter than the distance between these sections.

A lower bound of order n is computed as follow:

1. we choose n section along the link
2. for each adjoined pair of section (including the 2 junctions at the link ends which we denote S_0 and S_{n+1}) we find the shortest distance $dist(i, i+1)$.
3. $\sum_{i=1}^{n} dist(i, i+1)$ is a lower bound of order n for the link minimal length.

An order 0 LLB will be the straight line between the 2 junctions which had been already defined by Euclid as the shortest distance between 2 points. The simplest, and normally sufficient, approach is to distribute uniformly the sections along the link. A little bit more complex method is to search for the link's points which are the most remote from the straight line, and locate the section in these points. This method requires more computation effort but produces better results. Figure 5.1 illustrates the calculation of the *LLB*. When using a simple strategy of using at each iteration the mid point of the line segment, the third iteration gives a real LLB for the link.

5.4 Upper and Lower Bounds for Path Length

5.4.1 Path Length Upper Bound *PUB*

A nominal length of a path passing through several junctions is defined as the sum of all the nominal length of the links connecting the junctions that the path consists of. Since this path is possible, by definition of the skeleton creation, if it is not the shortest path possible, the shortest path would be shorter and therefore we can use this value as a path upper bound PUB for the minimal path length.

5.4.2 Path Length Lower Bound *PLB*

At each junction any path should pass at a distance smaller than the junction radius, so the maximum distance any of the links of which the path consists can 'gain' depends on radius of the 2 junctions it connects and the angles between the link and its ancestor and successor. A lower bound for path length is defined as follows (see figure 5.2):

AJ and BJ are straight segments of $L1$ and $L2$ respectively which are connected to the junction J and α is the angle between them.

R_j is the radius of junction J.

Any path from A to B should pass at least in one point at a distance R_j from J, we denote this point C.

CD and CE are perpendicular to AJ and BJ therefore:

$$AD = AJ - R_j \text{ *Cos } \alpha 1$$

$$BD = BJ - R_j \text{ * Cos } \alpha 2$$

ADC and BEC are triangles with right angle in which the hypotenuse is equal or greater than any of the other sides:

$$AC \geq AJ - R_j * Cos\alpha 1$$

$$BD \geq BJ - R_j * Cos\alpha 2$$

Combining these equations we obtain:

$$AJ + BJ - R_j * (Cos\alpha 1 + Cos\alpha 2) \leq AC + BC$$

$$AJ + BJ - 2R_j * \left(Cos\frac{\alpha 1 + \alpha 2}{2} \right) * \left(Cos\frac{\alpha 1 - \alpha 2}{2} \right) \leq AC + BC$$

$$AJ + BJ - 2R_j * \left(Cos\frac{\alpha}{2} \right) * \left(Cos\frac{\alpha 1 - \alpha 2}{2} \right) \leq AC + BC$$

Since $\left(Cos\frac{\alpha 1 - \alpha 2}{2} \right) \leq 1$ the following is also true:

$$AJ + BJ - 2R_j * \left(Cos\frac{\alpha}{2} \right) \leq AC + BC$$

and we can calculate a lower bound for path length as:

$$PLB = \sum LLB - \sum_j R_j * Cos\frac{\alpha_j}{2}$$

LLB link lower bound
Rj junction radius
and *j* is going over all the junction of the path.

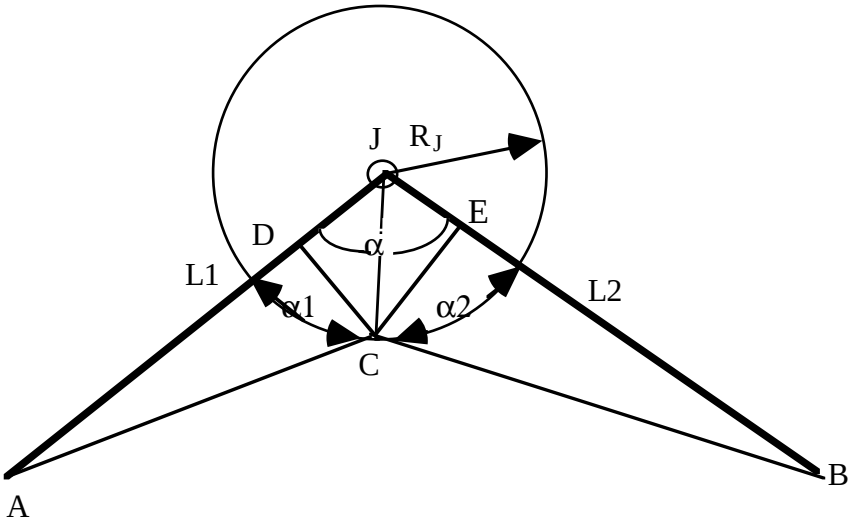


Figure 5.2 Geometric definition

5.5 Search Algorithm

The path finder is based on A* graph search subjected to the following heuristic and rules:

1. the remaining distance from a junction to the target is estimated as the length of the straight line connecting them.
2. the junction value is bounded between the following 2 sums:
   2.1 the *PUB* + remaining distance
   2.2 the *PLB*. + remaining distance

3. at each stage all the junctions for which the lower bound of their value is smaller than the minimum of the current junctions upper value are developed.
4. if more than one path lead to the same junction, any path for which the PLB is bigger than the minimal PUB of the paths leading to that junction is eliminated.

The search is performed in iteration, at each iteration we apply the search only to the paths that have 'survived' the preceding search. At each iteration the order of the LLB is augmented by 1.

The process terminates when either only one path has survived the preceding search, or if all the paths have reached their maximal *PLB.*

5.6 Discussion

5.6.1 Comparison to other methods

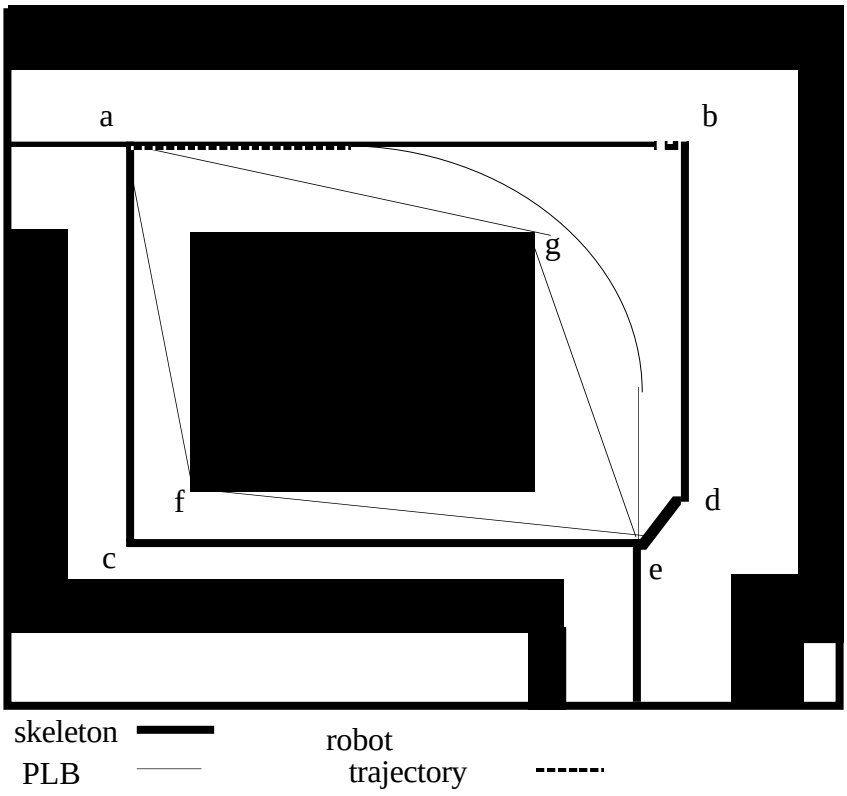


Figure 5.3 Comparing to Barraquand-Latombe

The results of the near optimal path finder in simple case are illustrated in figure 5.3 When comparing to the planning system presented by Barraquand and Latombe we see that when the shortest skeleton line does not correspond to the shortest optimal path the latter system will choose the shortest skeleton (a c e in figure 5.3). Our planning system will suggest the two possibilities but will recommend the other path (a b d e) because its PLB is shorter.

5.6.2 Stand-alone

The method presented is complete in the sense that it will always find a path (if it exists) and the optimal path will be one of the recommended paths. As a stand alone method it can be used by applying some educated guess (for example the path with minimal *PLB*), to choose a path among those recommended by the system. In that case the deflection from the optimal solution is bounded and the performances of the system can be easily evaluated.

In a constructed and crowded environment, as it is the case in many navigation problems, where the passable area occupies little of the space and mostly the width of the passes is very little compare to their length, the system is very efficient and more than likely will produce only one or two solutions.

Mostly it is not of such an importance to find the shortest path, An intelligent system can decide in each case, knowing the maximum length it can gain since the differences among the possible solutions are bounded, whether to require some more sophisticated methods or to 'gamble'.

5.6.3 Combination with exact methods

This method can be regarded as a filter that screens out the most, if not all, non shortest paths and bounds the length difference among the rest. It leaves to the user, or a sophisticated system, the choice between turning to some exact method with the advantage of applying to a smaller number of possible paths, and choosing a path among those recommended by the system (see stand-alone).

Even in the case of applying an exact or more accurate geometric method, the advantage of using this method as a pre-filter is obvious. Reducing the number of possible paths that should be checked, reduces significantly the amount of time and effort required since these exact methods are known to be complex and time consuming.

5.6.4 Any-time algorithm

Either as stand-alone or in combination with more accurate or exact methods this method can be used as an any-time algorithm. The probability that the path recommended by the method is the optimal is augmented with the time invested in computation. In a combined algorithm the augmentation will be very fast at the beginning of the search, while using our fast method, and then will decrease using exact but time consuming method.

6. The Navigation System

6. The Navigation System

The navigation system is a realisation of the theoretical approaches presented in chapters 4 and 5. The realisation of the theories introduced some techniques and precision as shown in figure 6.1. The map of figure 4.1 (theoretical) realised in probabilistic occupation grid. The sensors are now real sensors (sonar, infra-red, compass, odometry and bumpers). The planner is realised as the near optimal path finder from chapter 5 and a local mapping and skeletoniser were added to the system. The navigation system is composed of two subsystems:

(1) the global presentation and planning
(2) the local presentation and execution

Both subsystems share much the techniques they use. §6.1 describes these techniques.

The global subsystem uses the probabilistic occupancy grid and a graph based on the Voronoi diagram for presentation. The planning is a realisation of the near optimal path finder techniques that is described in chapter 5. §6.2 describes this subsystem.

The local subsystem is based on the extraction of the local graph. It extracts the local graph of the skeleton from a probabilistic occupancy grid that holds the latest information gathered by the sonar and infra red distance detectors. §6.3 describes the local subsystem.

The navigation system was developed on Sun Sparc Station and on Macintosh IIFX. The Nomad 200 mobile robot and a simulation of the Nomad 200 were used to embody and testing the navigation theory. A description of the experimental system and some experimental results is given in §6.4.

6.1 Techniques used by the Navigation System

There are several techniques and modules that are almost identical for the global and local subsystems. These techniques are described hereafter. The differences in nuances are referred to in the description of each subsystem.

6.1.1 Probabilistic Occupancy Grid

The occupancy grid is a direct geometrical representation of the working space (see 2.2.2 and 2.2.3). In our system we use a probabilistic occupancy grid based on the approach of Moravec and Elf [Mor-Elf 85] [Elf 85,89] that is described in details in § 2.2.3.

The probabilistic occupancy grid permits updating the map using sensors' information and dealing with sensory and localisation uncertainty (see the description of mapping process below).When no a priory information is available, the system creates the global map via the mapping process described below starting from a blank grid (all the cells have the value 0.5).

The cell size controls and defines the grid quality and performances. It defines the resolution of the grid and the number of cells required to cover a given area and hence affects the speed of treating, depends on the information contained in the grid. When choosing the cell size we should consider the accuracy of the available information and the system needs. It is useless to define cell size 1 cm to navigate a robot of 1 meter radius as well as to introduce information obtained from sonar where the accuracy is in the range of 10 cm or even less. The cell size should be small enough compared to the robot size to enable efficient navigation and not too small in order not to overcharge the system. The number of cells that cover a given area is depend on the inverse of the square of the cell size while the treatment of the grid is at least linear with the number of cells [Lat 91] [Meys 91].

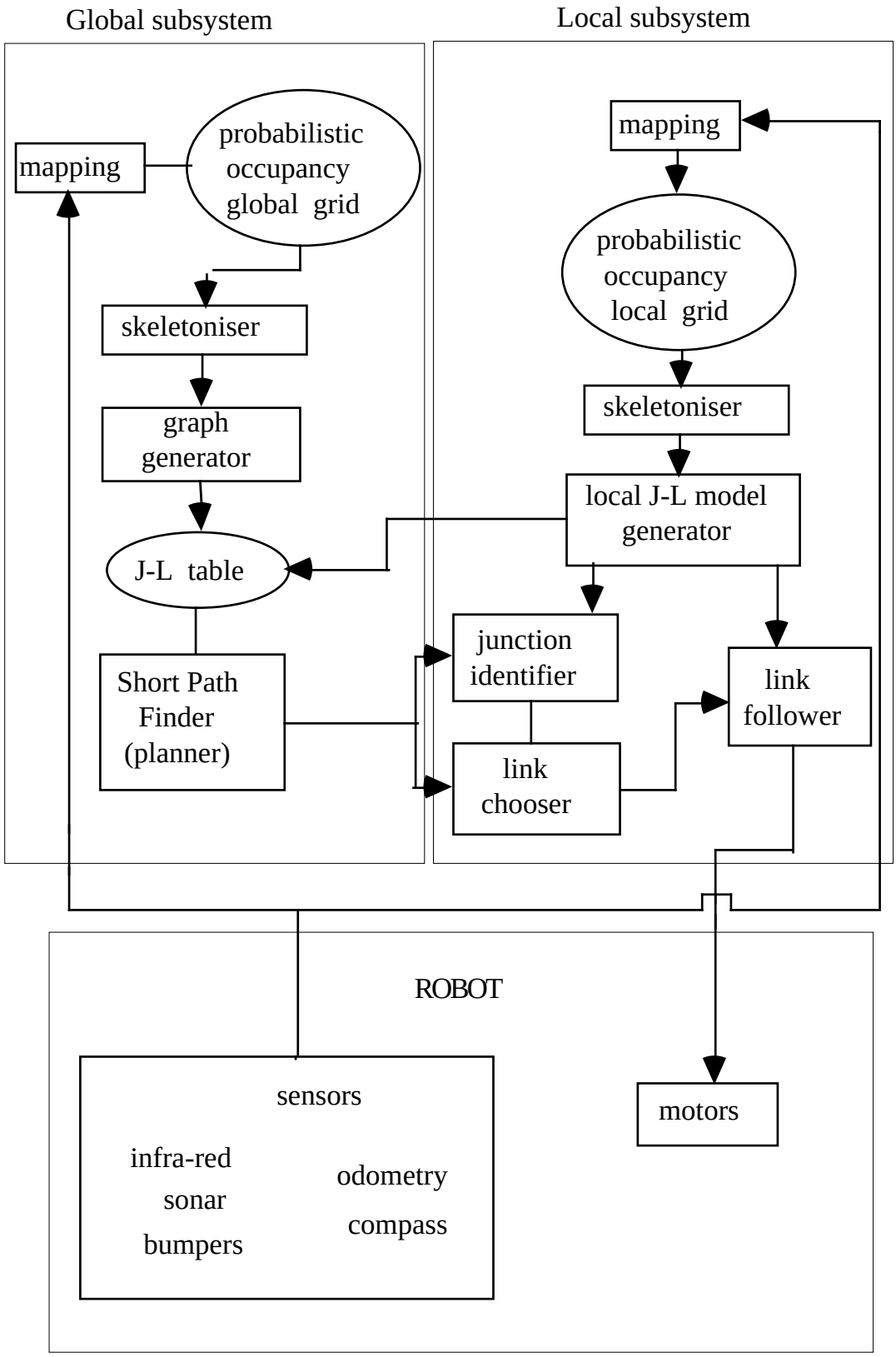


Figure 6.1 Schema of the navigation system

In our system, based on the experience we have, we have found that the a good cell size is about 20 - 30 cm (between a quarter and a half of the robot diameter).

For a real sensor we have to calculate or more often to measure the probability profile. In our case we measured the performances of the sonar and derive the approximate characterisation which is composed of $p[s(x) = OCC \mid x < r]$, the probability that a cell at a distance x smaller than the sonar reading r, would be occupied when the sensor reading says empty, and $p[s(x) = OCC \mid r]$ the probability that the cell is occupied at the reading distance r. The two correspond to what we call ghosts, reading of something which does not exist, and reflections. We have found the probability to depend mainly on distance, and derived a distribution function for the occupancy based on r. For practical reasons we linearised and defined it by the constant a as we can see in figure 6.2.

$p[s(x) = OCC \mid x < r]$ and $p[s(x) = OCC \mid r]$ are listed in tables that were derived manually from experiments with the sonar and the IR distance detector and then were tested and modified during robot operation till the performances became sufficient.

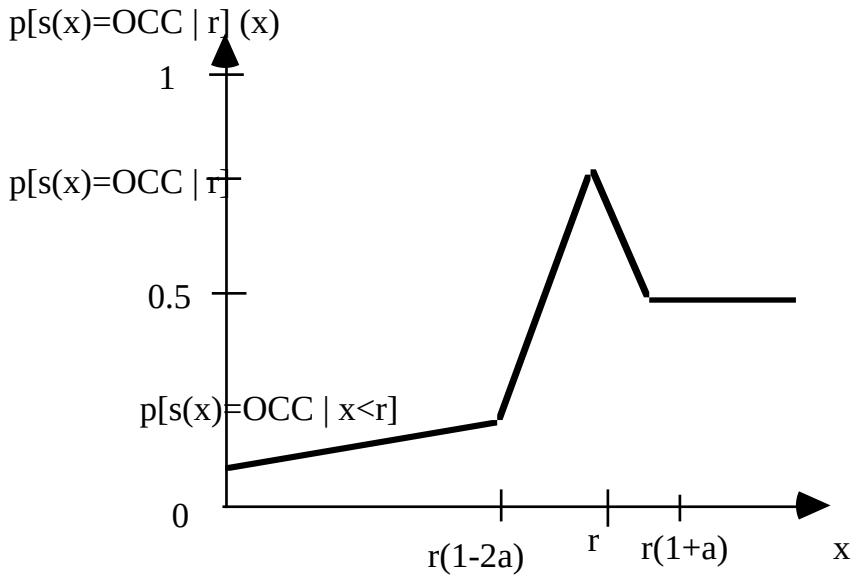


Figure 6.2 occupancy probability profile for real sensor

6.1.2 Extraction of the Skeleton

The skeleton is extracted from a deterministic occupancy grid that is derived from the probabilistic occupancy grid. The extraction of the skeleton is made in a 2 steps' process. The first step is based on the grid potential approach that was developed by Barraquand and Latombe [Bar-Lat 89] [Lat 91 pp 319 - 334]. The outcome of this step is a grid in which each cell is given the value of its distance from the boundary of the free space, and lists of the free cells, each one contains all the equi-distance cells.

In the second step the lists of the equi-distance cells are screened to eliminate all the cells that do not belong to the skeleton. The screening is based on the mathematical morphology approach of Jean Sera [Ser 82] [Ser 88]. Starting from the lowest potential

list each list is screened and each cell's neighbourhood is checked to comply with one of the typical forms of a skeleton cell. The non skeleton cells are eliminated from the lists and marked as non skeleton cells on the grid. The screening process is repeated until all the cells in the list belong to the skeleton. During the screening process the junctions are found and listed. The result of this process is a list of the skeleton cells and a list of the junctions.

6.1.2.1 The Deterministic Occupancy Grid

A deterministic occupation grid is derived from the probabilistic occupation grid. Each cell with an occupation probability value greater than $0.5 + \varepsilon$ (typical 0.6) is considered occupied and denoted the value 1, each cell with an occupation probability less than $0.5 - \varepsilon$ (typical 0.4) is considered empty and denoted the value 0, all the other cells are considered to have an occupation unknown and denoted the value 0.5. Lists of the empty, occupied and unknown cells are generated during this process. A smoothening process is performed next, each unknown cell with more than a given number of empty neighbours (typically 4, half of the neighbourhood) is considered empty. The condensing process is the next step, each cell within a given radius (typically the robot radius) around each occupied cell is considered occupied and if was not occupied before it is added to the occupied cell list as well as denoted the value 1 in the grid. The result of the latter step is the 2D configuration space for a circular robot. The deterministic grid and the list of the occupied cells are used in the next step to create the potential grid.

6.1.2.2 The Potential Grid

In the potential grid each cell is denoted a value that corresponds to its distance from the nearest border of the free space. The results are similar to a potential field which is imposed on the occupancy grid. This approach was introduced by Barraquand and Latombe [Bar-Lat 89] [Lat 91] to derive a potential field for robot motion planning which is guided by the Voronoi diagram. In our case we use the potential grid as a reference help to Voronoi diagram extraction system which is based on the morphology of the cells and described below. The potential grid is created identically to the method described by Latombe. At the first step, each occupied cell in the occupied cells list is checked for its immediate neighbours, to each neighbour which is not occupied or has already a potential value we assign the value 2 and the cell is added to the list of potential value 2. The step is terminated when all the occupied cells have been checked. The list of potential value 2 is transferred to the next step. At the second steps all the cells in the potential value 2 are checked for their neighbours, each neighbour that has no value is assigned the value 3 and added to the potential value 3 list. When the step is terminated the list is transferred to the next step. The process continues until all the cells of the grid have a potential value. In order to economise the memory the potential grid is kept in the same array as the deterministic grid above.

$$\begin{array}{ccc} C(7) & C(0) & C(1) \\ C(6) & C & C(2) \\ C(5) & C(4) & C(3) \end{array}$$

Figure 6.3 neighbourhood of C

Rosenfeld and Kak [Ros-Kak] obtain the same results by calculating the distance of a cell from the borders using a 2 passage algorithm from top and left to bottom and right in the first passage, and from bottom and right to top and left in the second passage. In the first passage the minimal distance from the upper and left borders is calculated for each cell by adding 1 to the value of each of its upper and left neighbours. In the second step the value of a cell is defined as the minimum of the first passage value and a value calculated as in the first stage but using the lower and right neighbourhood instead.

The description of the algorithm is given by the formula below
When:
S is the region which belongs to the image (free space in our case)
$N_1 = \{C(6), C(7), C(0), C(1)\}$ and $N_2 = \{C(2), C(3), C(4), C(5)\}$ are the left upper and the right lower neighbourhood respectively (see figure 6.3).
f and f' are the first and second passage distance functions.

$$f(C) = \begin{cases} 0 & C \notin S \\ \min\left[f(C(i)) | i = 0,1,6,7 \right] + 1 & C \in S \end{cases}$$

$$f'(C) = \min\left[f(C), f'(C(i)) | i = 2,3,4,5 \right] + 1$$

6.1.2.3 Thinning

The extraction of the skeleton is based on the morphology of the neighbourhood of each cell. This approach is inspired by the method presented by Jean Sera in Image Analysis and Mathematical Morphology [Ser 82] [Ser 88].

There are several methods to extract the skeleton which are based on the cell and its neighbourhood morphology and hereafter 3 of them are presented briefly.

The Rosenfeld and Kak (RK) thinning algorithm uses 4 different equations to check the borders' cells at each direction (i.e. upper border, bottom border, left border and right border) and apply the corresponding equation to each side of the free space (image) to find whether its deletion will modify the connectedness of the space [Ros-Kak]. Any cell (pixel) which has no effect on the connectedness of the image is called simple. At each passage the (RK) algorithm deletes the border cells at each side of the image provide they are simple and not end cells. The equations below check whether the cell C has neighbours arranged in a form that the elimination of C would disconnect two of its neighbours. In the set of equation below the second part is the same for all and verifies that the form of a star or partial star centred in C does not exist. If for example $C(1)$ belongs to the image and $C(0)$ and $C(2)$ on both sides do not the elimination of C would disconnect $C(2)$. The first part depends on the border. It verifies that the cells on both sides of C parallel to the border are connected only by C. Looking at $U(C)$ one can see that if $C(2)$ and $C(6)$ belong to the image when $C(4)$ does not the elimination of C will disconnect them. In this case the first part of $U(C)$ will give it the value 1 and make it non simple.

$U(C)$:

$$C(2)\overline{C}(4)C(6) + \sum_{k=0}^{3} \overline{C}(2k)C(2k+1)\overline{C}(2k+2) = 0$$

$B(C)$:

$$C(2)\overline{C}(0)C(6) + \sum_{k=0}^{3} \overline{C}(2k)C(2k+1)\overline{C}(2k+2) = 0$$

$R(C)$:

$$C(0)\overline{C}(6)C(4)+\sum_{k=0}^{3}\overline{C}(2k)C(2k+1)\overline{C}(2k+2)=0$$

$L(C)$:

$$C(4)\overline{C}(2)C(0)+\sum_{k=0}^{3}\overline{C}(2k)C(2k+1)\overline{C}(2k+2)=0$$

Where the overbars denote the complementation and C and its neighbourhood are as defined in figure 6.3. Note that C stands as well for the value of the cell which is 1 for image (free space) cell and 0 for background (occupied) cell. U, B, R and L are the conditions that should be checked for the upper lower right and left side border cell C according to the border it belongs. If the above conditions are respected the cell C is simple and can be deleted.

The Wang Zhang algorithm (WZ) [Wan-Zha 89] checks if a cell belongs to a border and whether it is an end cell. If not, it runs 3 checks:

(1) *A* (*C*) the number of background to object (equivalent to occupied to free) transitions in a clockwise walk around the cell. If there are more transitions than one that means that the *C* connects two cells that will be disconnected by its deletion.

(2) *G* (*C*) special forms of neighbourhood around the cell which are deletable even though there are more than one transition from background to object. This is true because *C(6)* will continue to be connected to *C(0)* even when we delete *C* as is the case of *k*=3. The same is true for *C(0)* and *C(2)* when *k* =5.

$$G(C)=\begin{cases}1 & if \quad C(k-1)+C(k)+C(k+1)+C(k+4)=0 \\ & \forall C(k+3)=1 \forall C(k+5)=1 | k=3,5 \\ 0 & otherwise\end{cases}$$

(3) *B(C)* the number of neighbours that belong to the image. If *B*=1 the cell is an end cell. If *B*=7 or 8 then the cell is interior or in a notch and its deletion can change the skeleton of the image.

The WZ algorithm can be summarised as follow:

for k=1 : 7
delete p if
1. 1 < B < 7 ; B the number of occupied neighbours.
and
2. A(C) = 1 or G(C) = 1
and
3.1 mod (k,2) = 1 and (C(k+2) + C(k+4)) C(k) C(k+6) = 0
or
3.2 mod (k,2) = 0 and (C(k) + C(k+6)) C(k+2) C(k+4) = 0

Condition 3.1 prevents the deletion of a cell for which more than two corners' cells. Condition 3.2 prevents the deletion of a cell for which more than half of a cross belongs to the image.

It is sufficient to apply the WZ algorithm to the border cells (pixels) at each iteration.

6.1.2.4 A Simplified Controlled Thinning Algorithm

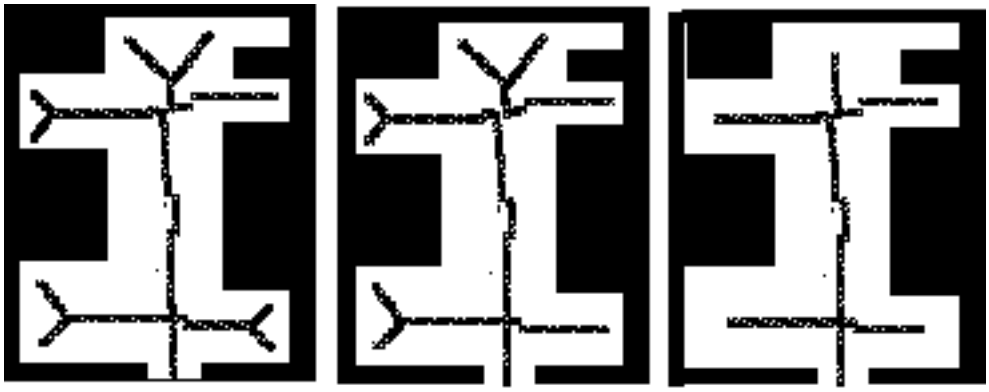
At first we have tried to develop a system which was based on the idea of looking for the local minimum as it has been proposed by Barraquand and Latombe and

by Rosenfeld and Kak, the results were quite poor. The connectivity of the free space which is, for navigation, the most important characteristic of the world was not conserved, something which was admitted later by Latombe [Lat 91, pp. 323-325] as he changes from the local minimum approach [Bar-Lat 89] to a method which is based on the meeting points of 2 expansion waves with different origins.

The final formulas are very close to the WZ method. The idea is to eliminate a cell whenever the elimination would not change the connectivity of the free space. Each cell neighbourhood is checked, if all the neighbours which belong or can belong to the skeleton are connected among themselves, and the cell is not an end point, the cell can be eliminated. There are two checks that we use. The first is the number of transition from free to occupied around the cell and the second is the number of non skeleton and occupied cells.

We've found that when this is executed from the borders inward, the cells that eliminated by the function G in the WZ algorithm and not by A will be deleted in the next passage. That makes the other checks used by WZ including g not necessary.

The level of the skeleton details is controlled by the function B (C) which determines the number of neighbours that belong to the skeleton. Each cell which is free and has not yet being eliminated is considered as a skeleton cell. The thinning process is perform in parallel with the calculation of the potential field as described above.



W-Z algorithm B_{min} 0 B_{max} 6 B_{min} 1 B_{max} 7

Figure 6.4 Comparison to W-Z algorithm

The algorithm is defined by the following formulas:

(1) $A(C)$ the number of occupied to free transitions in an anti-clockwise walk around the cell

(2) $B(C)$ the number of neighbours which do not belong to the skeleton.

A cell C is eliminated iff:

1. $A(C) =< 1$. and
2. $b_{min} < B(C) < b_{max}$; $(0 < b_{min} =< b_{max} < 8)$

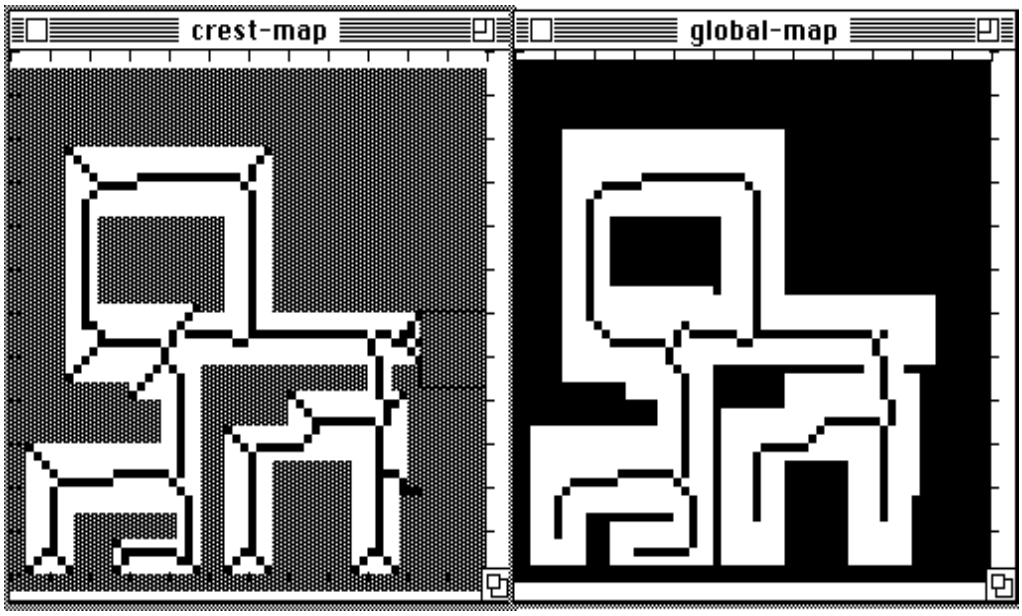
b_{min} and b_{max} control the level of the skeleton details, at the first iterations (close to the borders) we use a little b_{max} in order to have a detailed skeleton and a bigger b_{max} to filter noise and to smooth the borders (see figures 6.4 and 6.5.).

The algorithm generates a junction list during the thinning process. Each cell for which $A(C) > 1$ is listed automatically as a junction.

The thinning method which has been developed has two advantages over the WZ method: (1) It uses only two not complicated formulas which can be calculated at the same checking step for each cell, and (2) The level of details is controllable, hence

noise and unuseful details which disturb a lot in navigation can be dealt with in the lower immediate level of the thinning process.

The complexity of the algorithm is linear with the number of the grid cell.



a. B_{min} 0 B_{max} 6 b. B_{min} 1 B_{max} 7

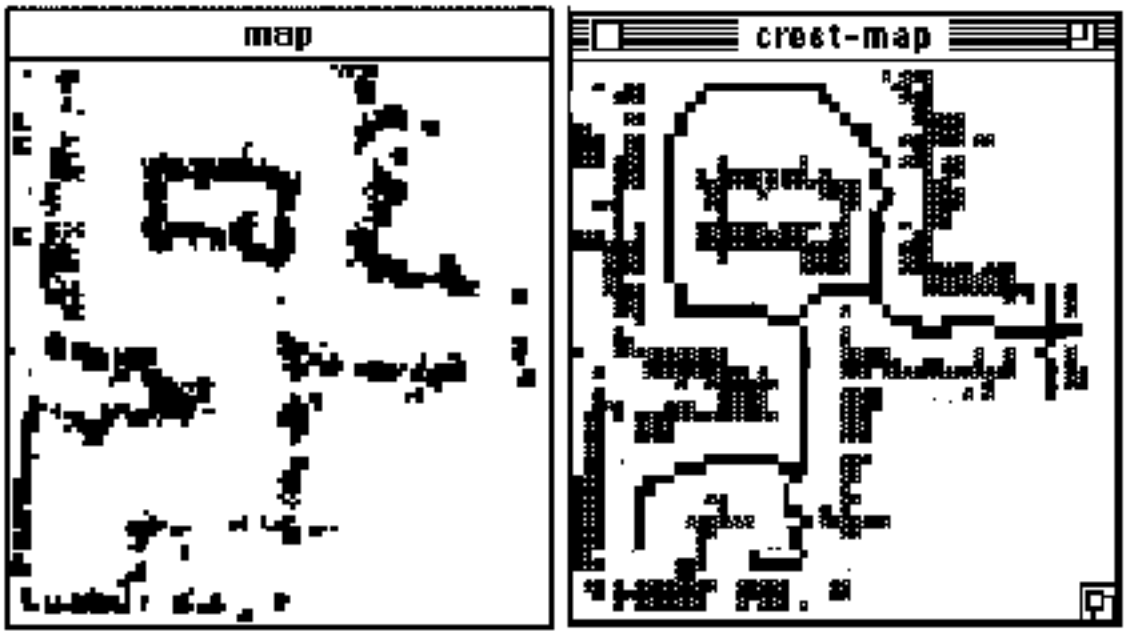
Figure 6.5 results of the thinning algorithm.

6.2 The Global Subsystem

The global navigation subsystem uses a global occupancy grid to generate a graph description of the world. The occupancy grid can be a deterministic grid given as a priori knowledge or a probabilistic grid that is created by a mapping process described in §6.1. The skeleton of the free space is extracted using the thinning process that is described above. From the skeleton the system generates a graph within which it performs the planning. In this chapter we described the generation of the graph and the planning.

6.2.1 Global Maps

The global map covers all the working space, holding all the necessary information for path planning and execution. The system maintains two levels of the global map: (1) occupancy grid (see chapter 2) and (2) Junctions and Links graph. Each of the levels can be constructed from the other hence they can be considered equivalent from the information point of view.



a. Probabilistic occupancy grid b. Skeleton after grid treatment

Figure 6.6 Results of global mapping process made by the robot

6.2.2 The Global Graph

The global graph is a symbolic description of the working space which is based on the Voronoi diagram of the free space. The graph consists of junction as nodes and links as branches (see chapter 3 and 4 for mathematical definition). Geometric information that the navigation system uses for planning (see chapter 5) and dealing with errors and noise, is attached to each junction and link. The graph is contained in the junction table for which the entries are the junctions' names. A links table that contains all the links connected to the junction is attached to each junction.

6.2.2.1 Junctions table

The junctions' table contains all the junctions in the working space. Each junction is a separate entry to the table accessible by the name that it has been given during the graph creation or updating.

Each entry contains the following information:

1. Junction name - a number between 1 to 99 (99 is a arbitrary limit which has been found more than sufficient to our purpose and facilitates the treating of the information).
2. Adjacent junctions - the list of the junctions' names which are connected to the above junction by a link. To each of which is attached the name of the connecting link.
3. Junction position - the global co-ordinates of the junction
4. Junction radius
5. Junction type - the number of links that create the junction
6 Junction links - a table of the links that are connected to the junction

For example let consider the junction "1" that is connected to the junctions "2", "3" and "4". The links that connect these junctions are links 1, 2 and 3 of junction 1 accordingly. The junction "1" is at x = 150 and y = 200 and has a radius of 5.
The junction entry will look like the following:
"1 adjacent: '((2) (3) (4)) position: '(150 200) radius: 5 type: 4" links: table .

For simplicity and practical reasons a knee (a point in which the link changes direction for 90 degrees or more) in a link or a dead end are considered as junctions of type 2 and 1 respectively. The dead ends are named after the junction to which they are connected multiplied by 100. For example two dead ends connected to junction 3 will be named 301 and 302.

6.2.2.2 Links' tables

The links' table contains the links that create the junction to which it is attached. To facilitate the planning each link appears two times, once as the link leading from junction J_i to junction J_j in the links' table of J_i and once as the link leading from J_j to J_i in the links' table of J_j.

Each entry contains the following information:

1. The link - the link number at the junction.
2. The link end - the name of the junction at the end of the link
3. Link direction - the general global direction of the link
4. Link length.
5. Geometric description - a list of the cells which create the link, each cell is described by its x and y co-ordinates and its radius.

6.2.3 Generating the Graph

The grid that holds the skeleton is used to generate the graph. Each grid cell that belongs to skeleton is defined by a value correspond to its distance from the borders of the free space. A junction has a value which is the original cell value plus 100, hence it becomes detectable at the grid level. All the other cells have the value 0. A junction is chosen arbitrarily and each line which is connected to it is explored cell after cell. Whenever a new junction occurs, it is developed line by line. This recursive process terminates when all the cells which belong to the skeleton were visited and all the lines and junctions where noted in the tables.

The following algorithm describes the process. The **main** chooses the junction to start from and calls the **develop-junction** which searches the links connected to the junction. When **develop-junction** finds a link it names the link and calls **explore-line** that follows the link. **develop-junction** returns to the procedure that called it when it finishes the full circle search. **explore-line** follows the link to its end that can be a junction or a dead end. When it is a dead end it returns to the **develop-junction** that called it. When the end is a junction **explore-line** calls **develop-junction** and when the latter returns it returns to the **develop-junction** that called it.

```
main
  choose arbitrarily junction (j)
  develop-junction (j)
end
```

```
develop-junction (j)
  l = 0
  do i 8
    when neighbour(j i) > 0
         explore-line (j l)
         l = l + 1
    end
  end do
end
```

```
explore-line (j l)
  if line was explored
    ln = read link name
    write inverse-link(ln) as link j.l
  else
    link-list = current-cell
    do while next-cell not a junction
          add cell to link-list
          search next cell
    end-do
    calculate link direction
    write link in links table as j.l
    write link name in current-cell
    develop-junction (next-cell)
  end if
end
```

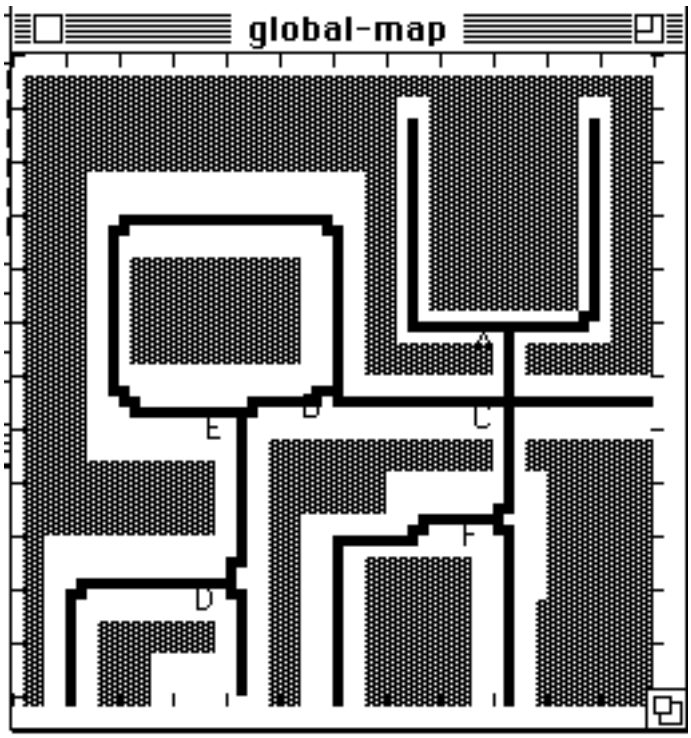


Figure 6.7 the graph of the lab and the planning results

6.2.4 Path Planning

The path planning is based on the approach that was presented in chapter 5. A target can be any of the junction in the global graph. For any other target the nearest junction will be the target for planning. From the nearest junction to the target itself another method of navigation should be used.

When given a target while on a junction the planning module plans a path from this junction to the target. If on a link a preliminary step of finding the nearest junction is executed.

The algorithm described in §5.5 is used to find a near optimal path that connects the current junction to the target.

6.2.4.1 Search Algorithm

The following algorithms describe the process of path search. These algorithms are based on the Near Optimal Path Finder that is described in chapter 5.

```
path :PLB :PUB :junctions :end
        PUB and PLB as they are defined in chapter 5,
        Junctions is the list of the junctions the path pass through,
                each of its elements is a list that consists of the junction name
                and the link that leads to the next junction.
        End is the last junction of the path

find-path (start target)
        paths = develop (start)
        PUB-min = find-min-PUB (paths)
  while paths
        path = pop (paths)
        if
                        * whenever a path reach the target it is added to the list
                        * of possible paths
                (path-end (path) = target)
                        add possible-paths path
        else if
                        * when the path PLB is smaller than the minimum PUB
                        * known at the moment the developing of the path is
                        * continued
                (path-PLB (path) <=  PUB-min)
                        add paths develop (path-end (path))
        else if
                        * when the PLB is bigger than the minimum PUB the
                        * path is not developed and added as is to the paths list it
                        * will be developed if and when its PLB will be smaller
                        * than the minimum PUB
                (path-PLB (path) > PUB-min) and (not possible-paths)
                        add paths path
        end if
end
```

The procedure develop finds the junctions which are connected to the junction it receives, and create the new paths that it adds to the paths list. It also updates the PUB-min which is the minimum PUB of all the paths in the list paths.

The result of find-path is a list of all the paths that can be the shortest paths from the junction 'start' to the junction 'target'. From this list we take the path of the lowest average of PUB and PLB but one can think of other criteria to choose the most promising path.

6.3 Local Subsystem - Path Following

The local subsystem creates a local map from the sonar and infrared distance sensors using the mapping process. From this map it generates the local skeleton and graph which it uses to localised itself and to generate a move command to the robot in order to follow the correct skeleton line (link).

6.3.1 Local Map

The robot uses a local map to identify links and junctions. The local map is a probabilistic occupancy grid of the immediate environment that the robot derives from its sensors. The execution part, the "follow a path module", extracts the local skeleton and activates one of the two possible behaviours, either "follow a link" or "turn in junction". The size of the grid as well as the size of the grid's cells is determined dynamically in a real time process which is based on the sensory input.

6.3.2 Mapping the Sensors' Reading

The robot's sensors' input is mapped directly on a local probabilistic occupancy grid. Each reading/processing cycle the robot creates a local grid map, for which the robot itself is at the centre. The size of the grid and the size of the grid's cells is determined from the sensors' readings (see bellow the description of the dynamic size determination). In order to compensate noises and reading's errors, the robot uses several sensors' readings to create the local map. Only relevant readings are mapped. The relevance is set by two relevancy tests:

1. Temporal relevance:- only the last $n_{r\text{-}max}$ readings are mapped, where $n_{r\text{-}max}$ is a controllable parameter of the system.
2. Spatial relevance:- only the reading which have been taken at a distance $d_r <= d_{r\text{-}max}$ are mapped, where $d_{r\text{-}max}$ is defined by the dynamic size determination system based on a controllable parameter.

The first test helps the robot to deal with dynamic changes in the world, only the latest information is used. The second test eliminates a lot of sonar malreadings that are due to distance and reflection angles.

The local map can incorporate an a priory knowledge that was derived from the global map, given a certain probability. This would be an equivalence of expectation at occupancy grid level. However at our case we use no a priory information. At each cycle we start from a blank grid. The initial value for all the grid's cells is 0.5, which means an unknown occupancy probability.

6.3.3 Dynamic Grid's Cell Size

While the robot moves in the working space the width of the passages is changed. To cover all the width of a passage which is necessary for the extraction of the skeleton a dynamic cell size is applied. The number of the grid cells is fix. After several experiments taking in consideration the time of processing such image and the robot speed the 25 x 25 grid was found to be a good compromise. The actual width of the passage is calculated from the last sonar reading. The distance between each pair of opposite sonar reading is the base for the calculation. The system takes the minimum distance as the actual width and calculates the cell size for which the free space width will cover half of the grid width.

For example if the minimum distance is 100" than the cells' size will be (integer (100 / 12) = 6". The cells' size is limited to a maximum of 15" (about half of the robot diameter) to prevent missing a link, and a minimum of 5" to prevent the creation of false skeleton lines (all the measurements are in inches). When the cells' size passes the maximum the grid size increases to keep the cells' size at the maximum. When the cells'

size passes the minimum the grid size reduces to keep the cells' size at the minimum, the latter increases the cycle speed in narrow passages.

6.3.4 Local J-L model

Every cycle the navigation system extracts the local skeleton from the local map using the thinning method described above (6.3.4.). The thinning parameter is 6 in order to eliminate the effects of sensors' noise. The result of the thinning process is a list of junctions and skeleton cells. A control parameter, the search-radius, defines the maximum distance for a junction to be considered.

From the considerable junctions it chooses the nearest to the robot. If no junction is available the system chooses the nearest skeleton cell as starting point. From the starting point it creates a local one node graph that represents the local J-L model. The node of the graph is the starting point and the branches are the links exiting the starting point. The links end either at the last skeleton point or at a junction on the other end of the link.

6.3.5 Following a Path

The following a path algorithm is described bellow. The execution module gets from the planning module the path list that includes the sequence of the junctions and links to be followed. It follows the list junction after junction. When there is a junction in the local J-L model it is checked and compared with the expected junction using the junction table.

If the check succeeds the system continues to the next junction on the list by choosing the right link to follow from the junction. If the check fails the search continues by following the link in the closest direction to the general direction defined in the links table. The search is limited to the expected length of the link plus a certain margin which is a control parameter of the navigation system. If the search failed the system use a recovery option. It turns back to the last identified junction and starts again.

```
follow-a-path (path-list)
        do-list (next-junction path-list)
                goto-next-junction (next-junction)
        end
end


goto-next-junction (next-junction)
        while (not junction-verified) and (travel < link-length)
                set (junction, travel) follow-a-link (next-junction)
                verify-junction (junction)
        end
        when (not junction-verified)
                goto-next-junction (preceding-junction)
end


follow-a-link (next-junction)
        read-sensors
        map-sensors
        set (junction, links) make-local-graph
        set link-end identify-link (next-junction)
        accumulate (travel move-to (link-end))
        return (junction, travel)
end
```

6.4 Experimental System and Results

6.4.1 Experimental Test-bed

The robot and computers on which the navigation system was implemented and tested changed three times during the five years of project MARS.

At the first stage that lasted about a year, the system consisted of a mobile robot Hero2000 of Zenith corporation and Macintosh IIFX that communicates with the robot and controls it. The basic behaviour of following the centre of a passage and the sonar mapping abilities were developed and tested at this stage. The Hero2000 provided poor control, communication and sensory characteristics and was therefore discarded and will not be described here.

In the second stage of the project a new participant, the Nomad 200 mobile robot came to play the main role. This robot that provides modern control, communication and sensory is described in the next section. Using the simulation environment that came with the robot the planning and the execution modules were developed, implemented and integrated to create the complete navigation system.

After about a year the project entered its third final stage using Sun Sparcs stations as host and control units. The navigation system was adapted to the UNIX/Sun environment to test and to demonstrate real world navigation abilities.

§6.4.3 describes and discusses some results of the second and third stages.

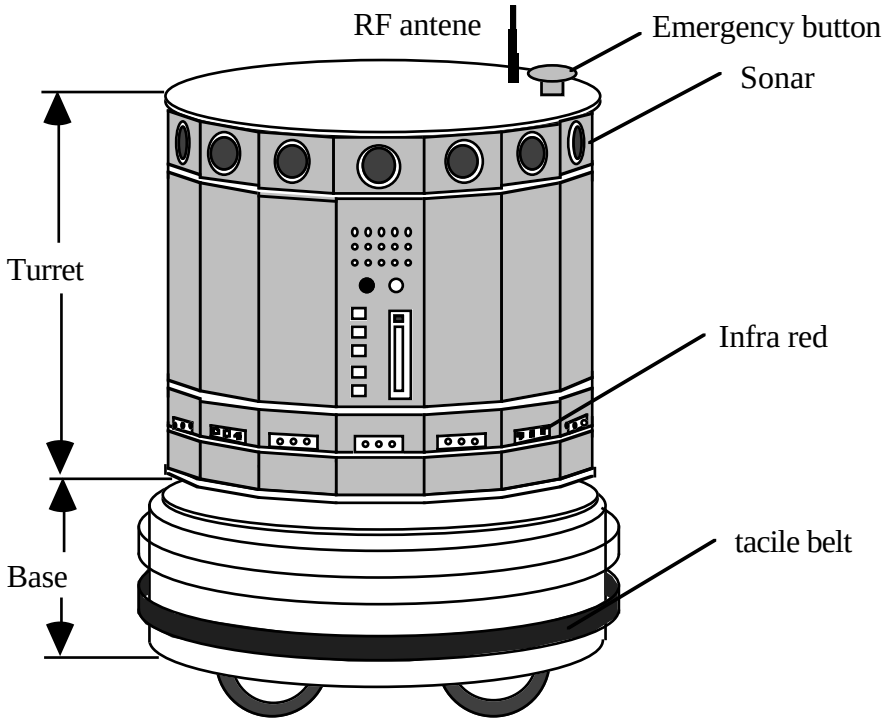


Figure 6.8. Nomad 200

6.4.2 The Nomad 200 Mobile Robot.

Nomad 200 (figure 6.8 above) is a 1 meter tall mobile robot. Its cylinder-shaped body is divided into two parts: a base and a turret. The base keeps its global attitude fixed and capable of translation move only. It is moved by a three wheels motion system that allows two degrees of freedom translation by rotating each wheel around its vertical axis. The turret is capable of a complete 360 degrees rotation independent of the wheels direction. Its sensors are arranged symmetrically around the body, each measuring a given sector of the surrounding environment. They are of three different types: 16 sonars near the top of the turret, 16 infrared range sensors at the bottom of the turret, 20 tactile sensors around the base and a digital compass mounted on top of the turret. Proprioceptive sensors include odometry.

An onboard PC compatible (Intel 486) controls the robot motors and sensory system. It communicates with the host computer via a serial RF modem.

On the host computer we find a fully simulated development environment that simulates the robot's activities, communication and environment. It enables direct shifting from the simulation to the real robot.

6.4.3 Experimental Results

Our laboratory at the institute was the environment in which we experimented the robot behaviours and the navigation system. Only little adaptation of the environment was needed. We had to put wooden board in front of the tables because the IR sensors' height prevents the detection of the table in close range.

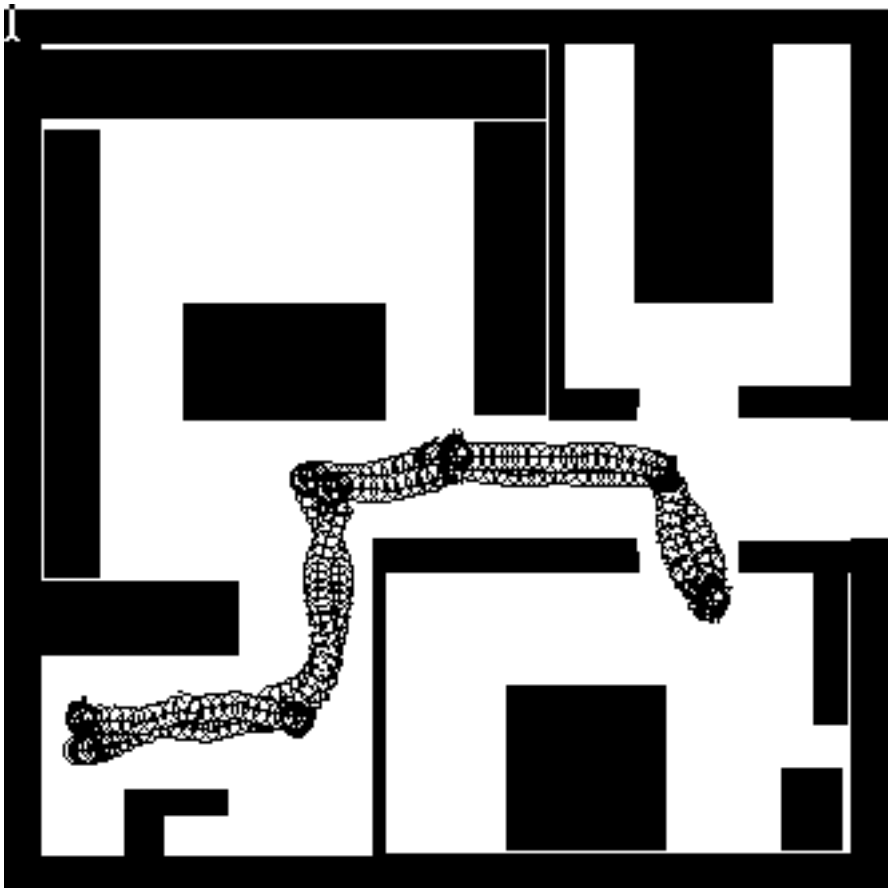


Figure 6.9 simulation results go and return of path DEBCF of fig. 6.7

The basic behaviours and the navigation system where first tested in the simulation environment that enable as well the introduction of sensors' and odometry errors. The results (see figures 6.9 6.10) were quite good. The robot followed more or less the same trajectory every time it repeated the same path, but as we can see it never repeated the exact trajectory due the different sensors' readings (see figure 6.9)

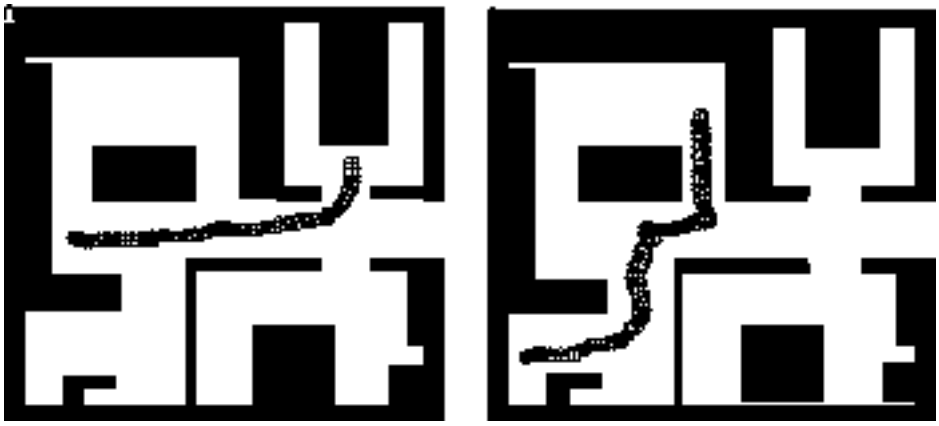


Figure 6.10 simulation results - following the path

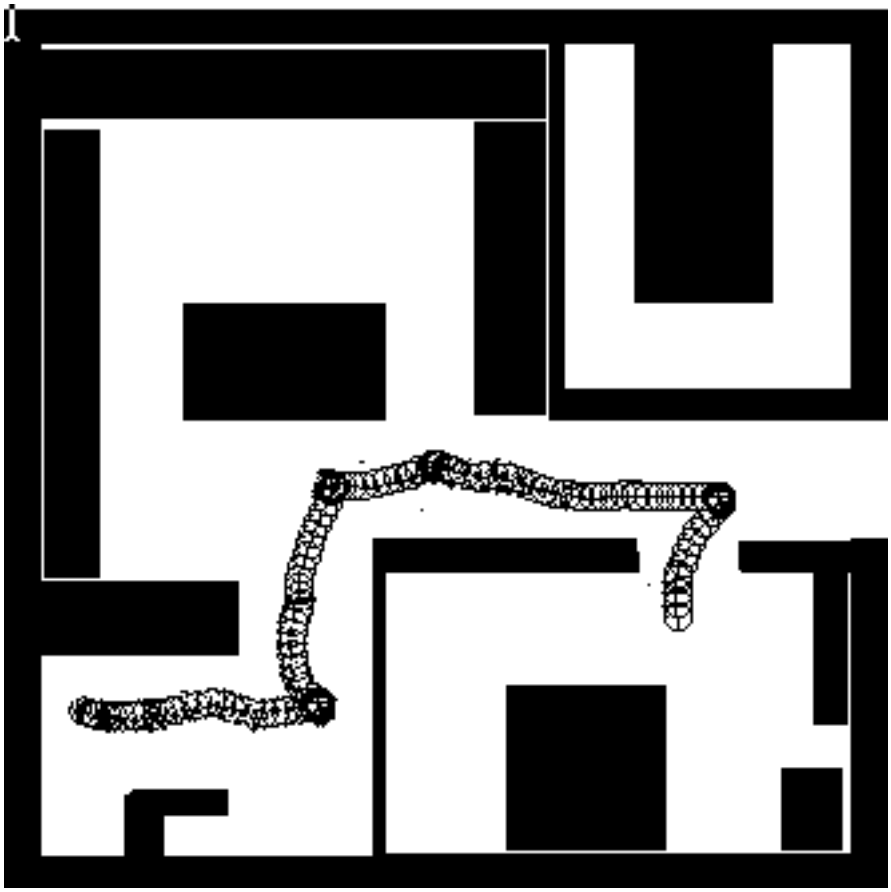
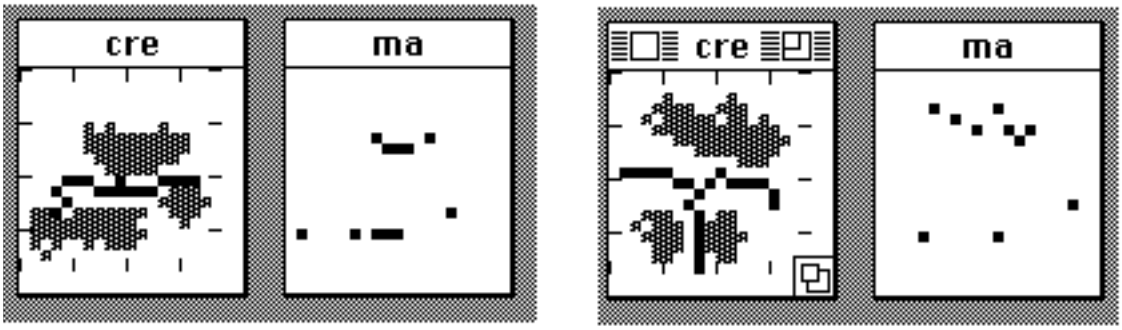


figure 6.11 real robot results following the path DEBCF

In figure 6.11 we see that when changing to the real world the robot follows almost the same trajectory of the simulation. A significant difference is the late recognition of junctions due to longer time cycle of reading and communication.



local skeleton local occupancy grid local skeleton local occupancy grid

Figure 6.12 Real robot - extraction of the skeleton from local sensor reading

6.5 Discussion

Above we have described a system that combines the behaviours as low level control and the classic planning. This is a real autonomous navigation system. Not like Toto of M. Mataric it can use a priory knowledge given as a regular normal map as well as mapping its environment. Not like in other systems (like AuRa see §2.4) the a priori knowledge should not be translated to symbols that the system can understand. It extracts the necessary information by itself and can match it to its treatment of its perception. It also relates the perceptions to actions reasoning about the results of an action to expect the new perception. We have created an equivocal representation in which the same graph represents the sensorial space and the action space.

We have shown that a navigation based on only one behaviour is possible and effective. The same approach and behaviours that combine the direction with the basic follow-a-link behaviour were used quite successfully by Miguel Rodriguez [Rod 94] in an autonomous robotics system. In this system the environment is mapped directly to a graph representing the sensorial perception in terms of the stimulated behaviours.

A by-product of the work is the developing of an improved thinning algorithm which is more efficient than the Wang Zhang algorithm and gives control of the skeleton level of details.

It is possible to develop other methods to extract the local J-L model directly from the sonar and infra red sensors. Some possibilities are neural network, pattern recognition by matching or the perceptron membrane.

7. Conclusions

7. Conclusions

7.1 Contributions

This work contributes at three levels to the field of robotics and autonomous intelligent systems.

At the theoretical abstract level I've presented a new look into behaviour based systems.

The second is the application level. I've developed a navigation system and world representation that bridge the gap between the classical planning approach and the behaviour based systems.

At the technical level I've developed two new technical solutions for already existing and well-treated problems. The first is a near optimal path finder that works entirely within the graph representation of the Voronoi diagram. The second is a simpler set of formulas to extract the skeleton from an occupancy grid.

7.1.1 Behaviours' Mapping

The theory of behaviours' mapping is a bridge that enables combining behaviours with classical planning and world modelling. The points of bifurcation, non continuity and non equivalence of the behaviours create landmarks which I called decision points. Those points enable building a world model that has the advantage of being entirely based entirely on the system's behaviours. The mapping of the behaviours onto a graph enables planning and reasoning. The graph compatibility with the actual behaviours results in a simple communication of the plan in terms of behaviours. A plan based on this model can be regarded as well as an answer to the problem posed by Agre and Chapman e.g. plan as advise. Instead of giving a command "execute now behaviour..." it says "at the next decision point execute one of the equivalence behaviours...".

The decision points give reveal another advantage of the system. Between them the system will act as what I described as a linear stable system, meaning that small perturbation creates small deviation and the system will arrive to the next decision point as long as the topology remains unchanged.

The meta behaviour that I've defined as the combination of all the behaviours the system possesses gives a compact and useful model of the world. I do hope that it complies with the citation of Einstein at head of this work.

7.1.2 The Navigation System

At the application level I presented a navigation system that demonstrates the feasibility and applicability of the behaviour mapping theory. It also demonstrates that navigation based only on one behaviour is possible and effective. It combines the behaviour execution with the geometric presentation and planning.

This navigation system answers the three basic questions of navigation in geometric terms as well as in behaviours' terms. A junction in the skeleton is a bifurcation point in behaviour terms. In contrary to other systems (like AuRA) the translation from the geometric model to the behaviour's based model and vice versa is immediate. Hence it enables to combine the classical planning within the geometric model with the stability and autonomy of behaviours' based systems.

The communication between man and machine becomes easier and accurate thanks to the geometry topology duality.

The behaviour follow-a-link gives the best result in structured and clustered areas. In open spaces other behaviours should be applied.

7.1.3 Path Planning

For the path planning I've developed a new approach that combines the heuristic A* graph search with a geometric information. It has the advantages of the simplicity and robustness of graph search while taking into account the geometry of the working space. It supports working with the uncertainty of path length and can be used as well as any time planning algorithm.

When comparing to the approach of Barraquand and Latombe it does not suffer from the local minima problem and once the graph is created there is no need to re-calculate any time the target is changed.

7.1.4 Skeleton Extraction

For the skeleton extraction I've reduced the set of conditions to be checked, hence the amount of calculations needed to verify whether a grid cell belongs to the skeleton or not. It has two advantages over the Wang-Zhang algorithm:

(1) The amount of calculation is reduced almost to an half.
(2) The details' level can be controlled.

As we have shown the second is quite important in navigation.

7.2 Future Work

My work leaves room for future work dealing with subjects and problems that I did not touch or did not completely develop.

There are three main fields that should be considered for a future work:

(1) Behaviour analysis and construction.
(2) Systems' analysis.
(3) Systems' construction.

At the behaviours' level:

- Using mathematical models of behaviours one can apply the theory of catastrophes or the theory of chaos to identify bifurcation points. It will enable direct behaviours' mapping into the model presented in this work.
- Construction and analysis of neural network based behaviours

Systems' analysis:

- Analysing and mapping of existing systems using the meta-behaviour approach
- Decomposition of system behaviour into its sub components

Systems' construction:

- Construction of multi-behaviours' systems
- Giving autonomous systems the ability to identify directly from their input the points of decision.
- Improving the self and external control of existing or new systems

7.3 General Conclusions and Some Private Reflections

7.3.1 Situation in the Autonomous Systems' Domain

In the introduction and the state of the art review I describe the two different basic approaches to the problem of creating an autonomous agent, the classic AI (§2.2 §2.3) and the reactive behaviour based systems (§2.5).

I do believe , as most of the AI community, that the classic pure logical way can't cope with the real world. This approach is as naive as the absolute determinism of Laplace saying that knowing the laws controlling the world and the exact conditions enable the prediction of the future of any system. We know now that this is not enough.

The Quantum Mechanics taught us the probabilistic nature of the physical world. The theory of the Chaos and the theory of Catastrophes show the non predictability and the great dependence on initial conditions of the non-linear systems. Hence lack of information and fuzzy input make the pure logic of the classic AI a nice mathematical model that can cope with chess but not with the real physical world.

On the other hand the reactive or situated actions systems that cope quite nicely with the real world reached their limits and can't go much farther. They lack the ability to plan and predict, therefore they can survive but doubtfully demonstrate an intelligent existence. To enable planning world models were needed so world models were introduced. M. Mataric introduced a behaviours' based model the system acquires by experience (§2.5.5). The model consists of the collection of paths the robot followed. This model enables repeating the same path to arrive to an already known goal but nothing more. R. Arkin presents the other extreme introducing a geometric model where the sites are connected by carefully specially developed behaviours (§2.5.6). The system can't create or modify the model by itself and uses a priori model made by its creator. Project MARS (§2.6) makes quite a step ahead. The model is based on the behaviours themselves and enables planning at the symbolic level and creating a new path based on the knowledge stored in the model. Adding a new behaviour is simple and doesn't change the nature of the model. The information is based on the robot experience and there is no way to treat and acquire a knowledge that comes in other forms.

The theory of behaviour space mapping that I presented in this work proposes a model that based not directly on the behaviours themselves but on the behaviours' bifurcation and behaviours' equivalence. These phenomena which are symbolic and common to all the behaviours enable creating a symbolic world model using only and in terms of behaviours. The model of the world used in project MARS is one of its derivations as well as the system of navigation described in §4. It enables acquiring knowledge and creating a world model either from experience or from geometric or physical maps and world models.

To conclude I do believe that I created a new bridge that connects the real world in which an intelligent system should be able to work to the abstract modelling in which an intelligent agent should be able to plan.

7.3.2 Quo Vadis Autonomous Systems

We can follow the line of behaviours' based and neural network systems and push it to the limit using better and faster sensors and computers and maybe one day in the future we will produce a replica of a human being made of metal and silicon instead of carbon compounds. We can follow the classic logic AI to the limits using brut computation force to create a machine that will be the world chess champion (we are not so far from that goal). Will we really create intelligence by doing so? I doubt. In the first way we will create the physics of a creature but we will not understand how it functions on the symbolic and intelligence levels. In the second way the brut computation force covers lack of intelligence. The machine will play chess (which is theoretically a solved game) better than any man or woman but will be worth nothing for anything else.

I do believe that to create an intelligent autonomous system that will serve us and will help us to understand the essential nature of intelligence we should combine the basic behaviours with models in terms that we as creators and the system can understand and communicate. I do hope that my work contributes a little in this direction.

Bibliography

References

[Agre-Chap 87] P.E. Agre D. Chapman, Pengi: An Implementation of Theory of Activity, Proceeding of AAAI 87 National Conference.

[Agre-Chap 90] P.E. Agre D. Chapman, What are plans for, Designing Autonomous Agents, edited by Pattie Maes, MIT press 1990.

[Alam-Chat 92] R. Alami, R. Chatila and M. Ghalab, Mission Planning And Control for an Autonomous Mobile Robot, LAAS-CNRS 1992.

[Akman 89] V. Akman and P.J.W. ten Hagen, The Power of Physical Representation , AI Magazine, Vol. 10 No. 3 Fall 1989.

[Allen 84] J.F. Allen, Toward a General Theory of Action and Time, Artificial Intelligence Vol. 23 1984

[And-Don 90] T.L. Anderson and M. Donath, Animal Behaviour as a Paradigm for Developing Robot Autonomy, Designing Autonomous Agents, edited by Pattie Maes, MIT press 1990.

[Ark 90] Ron C. Arkin, Integration Behavioural Perceptual and World Knowledge in Reactive Navigation, Designing Autonomous Agents, edited by Pattie Maes, MIT press 1990.

[Ark 92] Ron C. Arkin, Homeostatic Control for a Mobile Robot: Dynamic Replanning in Hazardous Environments, Journal of Robotics Systems, Vol. 9 No. 2 1992

[Aur 91] F. Aurenhammer, Voronoi Diagram - A Survey of a Fundamental Geometric Data Structure, acm computing survey volume 23 No. 3 September 1991.

[Ball-Bro] D.H. Ballard and C.M. Brown, Computer Vision, p 252-253

[Bares 89] John Bares et al, Ambler: An Autonomous Rover for Planetary Exploration, Computer, Vol. 22 No. 6 June 1989

[Bar-Lat 89] J. Barraquand and J-C Latombe, Robot Motion Planning: A Distributed Representation Approach, TR STAN-CS-89-1275.

[Bod 88] M.A. Boden, Computer Models of Mind, Cambridge Press 1988.

[Bourg 92] Paul Bourgine, Heuristique et Abduction, Intelligence Artificielle et Vie Artificielle, XXII session de l'Ecole Internationale d'Informatique de l'AFCET.

[Bourg-Var 91] Paul Bourgine and Francisco Varela, Toward a Practice of Autonomous Systems, Proceeding of the 1st European Conference on Artificial Life, Paris Dec. 1991.

[Brait 84] Valentino Braitenberg, Vehicules Experiments in Synthetic Psychology, M.I.T press, 1984

[Bring 77] Jean-Claude Bringuier, Conversations Libres avec Jean Piaget, Edition Robert Laffont, 1977.

[Broo 83] R. A. Brooks, Solving the Find-Path Problem by Good Representation of Free Space, IEEE Transaction on System, Man and Cybernetics, SMC-13(3), 190-197.

[Broo 86] R. A. Brooks, A Robust Layered Control System For A Mobile Robot, IEEE Journal of Robotics and Automation, Vol. RA-2, No 1, March 1986.

[Broo 86a] R. A. Brooks, Achieving Artificial Intelligence Through Building Robots, AI memo 899, AI Laboratory, MIT.

[Broo 87a] R.A. Brooks, Intelligence Without Representation, internal report, MIT.

[Broo 87b] R.A. Brooks, Planning is Just a Way of Avoiding Figuring Out What to Do Next, internal paper, MIT.

[Broo 90] R.A. Brooks, Elephants don't Play Chess, Designing Autonomous Agents, edited by Pattie Maes, MIT press 1990.

[Broo 91] R. A. Brooks, Intelligence Without Reasoning, IJCAI 91, Sydney, Australia, Aug. 1991.

[Broo 91a] R. A. Brooks, Artificial Life and Robots, Proceeding of the 1st European Conference on Artificial Life, Paris Dec. 1991.
[Chap 87a] David Chapman, Non-linear Planning: a Rigorous Reconstruction, Technical Report, AI Lab MIT 1987.
[Chap 87b] David Chapman, Planning for Conjunctive Goals, Artificial Intelligence, Vol.. 32, 1987
[Chap 89] David Chapman, Penguins Can Make Cakes, AI Magazine Vol. 10 No. 4, Winter 89
[Chap 91] David Chapman, Combinatorics and Action, AAAI Fall Symposium, Assilomar, Cal. Nov. 1991.
[Char-McDer 84] E. Charniak and D. McDermott, Introduction to Artificial Intelligent, Addison-Wesley 1984.
[Cohen 89] P.R. Cohen, M.L. Greenberg, D.M. Hart and A.E. Howe, Trial by Fire: Understanding the Design Requirements for an Agent in a Complex World, AI Magazine, Vol. 10 No. 3 Fall 1989.
[Conn 88] J.H. Connell, Navigation by Path Remembering, SPIE 1988.
[Crow 87] J.L. Crowley, The State of the Art in Mobile Robotics, The Fourth International Symposium on Robotics in Construction, Haifa Israel 1987.
[Crow 88a] J.L. Crowley, World Modelling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging, LIFIA (IMAG) I.N.P.G. Oct. 88.
[Crow 88b] J.L. Crowley, Symbolic and Spatial Reasoning in Navigation, LIFIA (IMAG) I.N.P.G. December 1988.
[Crow 89] J.L. Crowley, Ph. Bobet, K. Sarachik, Dynamic World Modelling Using Vertical Line Stereo, The First European Conference of Computer Vision, October 1989.
[Davis 93] R. Davis, H. Shrobe and P. Szolovits, What is a Knowledge Representation, AI Magazine, Vol. 14, No. 1 Spring 1993
[Dean 93] T. Dean and R.P. Bonasso, The 1992 AAAI Robot Exhibition and Competition, AI Magazine, Vol. 14, No. 1 Spring 1993
[Dech 90] R. Dechter, Enhancement Schemes for Constraint Processing: Back jumping, Learning and Cutset Decomposition, Artificial Intelligence Vol. 41 No. 3 Jan. 90 p. 273.
[Def 94] Guilliaume Deffuant, An Algorithm for Building Regularized Piecewise Linear Discrimination Surfaces: The Perceptron Membrane, Crest 1994
[Ding 91] H. Ding, L.L. Huang and Y.L. Xiong, Computer Aided Off-Line Planning of Robot Motion, Robotics and Autonomous Systems, Vol. 7 No. 1 March 1991
[Don-Jen 91] Bruce R. Donald and James Jennings, Constructive Recognizability for Task-Directed Robot Programming, Computer Science Dept. Cornell University, 1991.
[Drog 93] A. Drogoul, When Ants Play Chess, MAMAOW, Neuchâtel, 1993
[Drum-Cur 89] M. Drummond and K. Curri, Goal Ordering in Partial Ordering Plans, IJCAI 1989.
[Elf 85] A. Elfes, A Sonar Based Mapping and Navigation System, 1985 IEEE Conference on Robotics and Automation, St. Louis, March 85
[Elf 89] A. Elfes, Using Occupancy Grids for Mobile Robot Perception and Navigation, Computer Vol. 22 No. 6 Jun. 89 p. 46.
[Foul 92] D.E. Foulster, M. Li and Q Yang, Theory and Algorithms for Plan Merging, Artificial Intelligence, Vol. 57 No. 2-3, Oct. 1992
[Firby 87] R.J. Firby, An Investigation into Reactive Planning in Complex Domains, AAAI conference 1987.
[Gam 91] L. M. Gambardella, Simulation and Planning with Multiple representations. AI Simulation and Planning in High Autonomy System, Cocoa Beach, Florida, April 1991.

[Gaus-Zre 93] Philippe Gaussier and Stephane Zrehen, The Probabilistic Topological Map (PTM): A self Organizing and Fast Learning Neural Map That Preserved Topology, Annales du Groupe CARNAC No. 6 1993

[Gat E. 93] Erann Gat, On the Role of Stored Internal State in the Control of Mobile Robots, AI Magazine, Vol. 14, No. 1 Spring 1993

[Gat 92] Y. Gat , Near Optimal Path Planner Based on Voronoi Diagram, Proceeding of SGAICO Conference, Neufchatel, Swiss, Sept. 1992.

[Gat et al 92] Yoel Gat, Miguel Rodriguez, Jean-Pierre Müller, Enriched Sensitive and Perceptive Localisation, Proceeding of SGAICO Conference, Neuchâtel, Swiss, Sept. 1992.

[Gat-Müll 91] Y. Gat and J.P. Müller, Simple World Modelling for Reactive Navigation, AAAI Fall Symposium, Nov. 1991.

[Gat-Müll 92] Y. Gat and J.P. Müller, Reactive Navigation Based on Simple World Modelling, Proceeding of The 9th Israeli Symposium on Artificial Intelligence, Ramat-Gan, Israel, Dec. 1992.

[Gins 89a] Matthew L. Ginsberg, Universal Planning: An (Almost) Bad Idea, AI Magazine Vol. 10 No. 4, Winter 89

[Gins 89b] Matthew L. Ginsberg, Universal Planning Research: A Good or Bad Idea, AI Magazine Vol. 10 No. 4, Winter 89

[Giun-Walsh 92] F. Giunchiglia and T. Walsh, A Theory of Abstraction, Artificial Intelligence, Vol. 57 No. 2-3, Oct. 1992

[Gup-Nau 92] N. Gupta and D.s. Nau, On The Complexity of Block-World Planning, Artificial Intelligence Vol. 56, Aug. 1992.

[Gup-Yam 88] M.M. Gupta and T. Yamakawa, Fuzzy Logic in Knowledge-Based Systems, Decision and Control, North-Holland 1988

[Hayes 89] C. Hayes, Using Goal Interaction to Guide Planning, AAAI Conference 1987.

[Hayes-Roth 93] B. Hayes-Roth, P. Lalanda, P Morignot, K Pfleger, an M. Balabanovic, Plans and Behaviour in Intelligent Agents, Report No. KSL 93-43, Stanford University, May 93.

[Holland 80] J. Holland, Adaptive Algorithm for Discovering and Using general Patterns in Growing Knowledge-Bases, International Journal of Policy Analysis and Information Systems, Vol. 4 No. 3 1980.

[Holland et al 86] J. Holland, K. Holyoak, R. Nisbet and P. Thagard, Induction: Process of Infernce, Learning and Discovery, MIT Press, 1986.

[Hors 88] I.D. Horswill, Reactive Navigation for Mobile Robots, MIT 1030 May 88.

[Hutch 90] S.A. Hutchinson and A.C. Kak, SPAR: A Planner That Satisfies Operational and Geometric Goals in Uncertain Environments, AI Magazine, Vol. 11 No. 1 Spring 1990.

[Iyen-Rang 89] S.S. Iyengar and R.L. Kashyap, Autonomous Intelligent Machines, Computer, Vol. 22 No. 6 June 1989

[Iyen-Thom 92] S.S. Iyengar and D. Thomas, Autonomous Mobile Research at Louisiana state University, AI magazine, Vol. 13 No. 2, summer 1992

[Inoue 85] H. Inoue, Building a Bridge Between AI and Robotics, IJCAI 1985

[James 84] I.M. James, General Topology and Homotopy Theory, Springer Verlag 1984.

[Kael 91] L.P. Kaelbing, An Adaptable Mobile Robot, Toward a Practice of Autonomous Systems, Proceeding of the 1st European Conference on Artificial Life, Paris Dec. 1991

[Kael-Rosen 90] L.P. Kaelbing and S.J. Rosenschein, Action and Planning in Embedded Agents, Designing Autonomous Agents, edited by Pattie Maes, MIT press 1990.

[Kha 86] O. Khatib, Real-Time Obstacle Avoidance System for Manipulators and Mobile Robots, International Journal of Robotics Research, 5(1) 1986, p 90-98.

[Kris-Chen 90] Raghu Krishnapuram and Ling-Fan Chen, Iterative Neural Network for Skeletonisation and Thinning, Proc. of SPIE: Intelligent Robots and Computer Vision ix: Neural, Biological and 3-D Methods, Vol. 1382 Boston 7-9 Nov. 1990

[Lat 88] J-C Latommbe, Global Path Planning Approaches to Material Movements in a Worksite, NATO AWR on Advanced Information Technologies for Industrial Material Flow Systems, Grenoble 1988

[Lat 91] J-C Latommbe, Robot Motion Planning, Kluwer Academic Publishers (SECS 0124) 1991, ISBN 0-7923-9129-2.

[Lec 94] Jacque Lecommte, Etre Intelligent ce n'est pas Seulment Savoir Rflechir, Science Humaines No. 36 Feb. 94.

[Lev-Law 90] T.S. Levit and D.T. Lawton, Qualitative Navigation for Mobile Robots Artificial Intelligence Vol. 44 No. 3 Aug. 90 p. 305.

[Lig 90] A. Ligeza, Dynamic Backward Reasoning System, Artificial Intelligence Vol. 43 No. 2 may 90 p. 127.

[Lohm 92] K. J. Lohman, How See Turtles Navigate, Scientific American, Jan. 1992

[Maes 90] P. Maes, Situated Agents can have Plans, Designing Autonomous Agents, edited by Pattie Maes, MIT press 1990.

[Mah-Conn 92] S. Mahadevan and J. Connel, Automatic Programming of Behaviour-Based Robots Using Reinforcement Learning, Artificial Intelligent Vol. 55 No. 2-3 June 1992.

[Mas 90] P.R. Masani, Norbert Wiener, Vita Matematica Vol. 5, 1990.

[Mason 93] M.T. Mason, Kicking the Sensing Habit, AI Magazine, Vol. 14, No. 1 Spring 1993

[Mat 89] Maja J. Mataric, Qualitative Sonar Based Environment Learning for Mobile Robots, SPIE Mobile Robots IV Proceeding, Philadelphia, PA, Nov. 1989.

[Mat 90a] Maja J. Mataric and R. Brooks, Learning a Distributed Map Representation Based on Navigation Behaviours, MIT, Proceedings of the USA-Japan Symposium on flexible Automaton, Kyoto, Japan, July 1990, 499-506.

[Mat 90b] Maja J. Mataric, Environment learning using a distributed representation, MIT, Proceedings of the 1990 IEEE International Conference on Robotics and Automation.

[Mat 90c] Maja J. Mataric, Navigation With a Rat Brain: A Neurobiologically Inspired Model for Robot Spatial Representation, Proceedings of the First International Conference on Simulation of Adaptive Behaviours, MIT Press Bradford Books 1990

[Mat-Var 80] Humberto R. Maturana and Francisco Varela, Aoutopoiesis and Cognition, Boston Studies in the Philosophy of Science vol. 42, D. Reidel Publishing Co. 1980.

[McCor 79] Pamela McCorduck, Machines Who Think, 1979

[McDer 92] Drew McDermott, Robot Planning, AI magazine, Vol. 13 No. 2, summer 1992

[Meys 91] A. Meystel, Autonomous Mobile Robot - Vehicles with Cognitive Control, Series in Automation - Vol. 1, World Scientific Publishing Co. , 1991.

[MILL 89] David P. Miller, Execution Monitoring for a Mobile Robot System, Proc of 1989 Conference on Intelligent Control and Adaptive Systems, Vol. 1196, p. 36-43, Philadelphia, PA, Nov. 1989.

[MILL 90] David P. Miller, Rover Navigation Through Behaviour Modification, Proc. of the Space Operation Automation and Robotics Workshop, NASA, Albuquerque, NM, June 1989.

[MILL 91] David P. Miller, Levels of Sensing for Rough Terrain Navigation, AAAI Fall Symposium, Assilomar, Cal, Nov. 1991.

[MILL 91] David P. Miller, A Twelve-Step Program to More Efficient Robotics, AI Magazine, Vol. 14, No. 1 Spring 1993

[Mins 85 Marvin Minsky, The Society of Mind, Simon & Schuster, 1985
[Mitch 88] Joseph S.B. Mitchell, An Algorithmic Approach to Some Problems in Terrain Navigation, Artificial Intelligence Vol. 37 No. 1-3 Dec. 88 p. 171.
[Mor-Elf 85] Hans P. Moravec and Alberto Elfes, High Resolution Maps from Wide Angle Sonar, IEEE 1985
[Pav 77] T. Pavlidis, Structural Pattern Recognition, Springer-Verlag 1977.
[Piaget 67] Le Representation du Monde chez L'Enfent, Jean Piaget, Press Universitaires de France, 1967
[Pfeif-Verch 91] Rolf Pfeifer and Paul F.M.J. Veschure, Distributed Adaptive Control: A Paradigm for Designing Autonomous Agents, Zurich University, July 91.
[Pfeif-Verch 92] Rolf Pfeifer and Paul F.M.J. Veschure, Designing Efficiently Navigation Non-Goal-Directed Robot, Zurich Univercity, Nov. 92.
[Pour 91] B. Pourbabai, A Note on Avoiding Obstacles, Robotics and Autonomous Systems, Vol. 7 No. 1 March 1991
[Rag-Chen 90] K. Raghu and Ling-Fan Chen, Iterative Neural Network for Skeletonizing and Thinning, Proc. of SPIE: Intelligent Robots and Computer Vision IX, Vol. 1382, Boston Nov. 1990.
[Ram 90] F. Ramos, Une Contribution a la Planification de Chemins en Robotique Mobile, Universite de Franche-Comte No. 164.
[Rao 89] N.S.V. Rao, Algorithmic Framework for Learned Robot Navigation in Unknown Terrains, Computer Vol. 22 No. 6 Jun. 89 p. 37.
[Rice 89] James Rice, The advance Architecture Project, AI Magazine Vol. 10 No. 4, Winter 89
[Rod 92] Miguel Rodriguez, Modélisation du monde par extraction de circuits sensitifs, internal report, Institut d'Informatique et d'Intelligence Artificielle, UNI Neuchâtel, 1992.
[Rod 94] Miguel Rodriguez, Modélisation d'un Agent Autonome: Architecture et Representation, These, Institut d'Informatique et d'Intelligence Artificielle, UNI Neuchâtel, 1994.
[Ros-Kak] A. Rosenfeld and A.C. Kak, Digital Picture Processing, Academic Press, San Diego, 1982
[Roth-Jain 89] Y. Roth-Tabak and R. Jain, Building an Environment Model Using Depth Information, Computer Vol. 22 No. 6 June 1989
[Ser 82] Jean Sera, Image Analysis and Mathematical Morphology, Vol. 1, Academic Press 1982.
[Ser 88] Jean Sera, Image Analysis and Mathematical Morphology, Vol. 2, Academic Press 1988.
[Such 87] Lucy A. Suchman, Plans and Situated Actions, Cambridge Univercity Press, 1987.
[Sch 89] Marcel J. Schopers, In Defence of Reaction Plans as Cashes, AI Magazine Vol. 10 No. 4, Winter 89
[Sch-Shar 88] J.T. Schwartz and M. Sharir, A Survey of Motion Planning and Related Geometric Algorithm, Artificial Intelligence Vol. 37 No. 1-3 Dec. 88 p. 157.
[Smit 91] Tim Smithers, Taking Eliminative Materialism Seriously: A Methodology for Autonomous System Research, Proc. of the First European Conference on Artificial Life, Paris, Dec. 1991.
[Sorg-Holm 93] Amos Sorgen and Hans Holm, A New Approach to Robot Free-Path Problem by Stepwise Path Improvement, Engineering Application of Artificial Intelligence, Vol. 6 No. 1 1993
[Stee 90] Luc Steels. Exploiting analogical representations. Designing Autonomous Agents. Edited by P. Maes. MIT Press, 1990.

[Verch-Pfeif 92] Paul F.M.J. Veschure and Rolf Pfeifer, Categorisation, Representation and the Dynamics of System-Environment Interaction, Zurich Univercity, Nov. 92.

[Wan-Zha 89] P.S.P. Wang and Y.Y. Zhang, A Fast Flexible Thinning Algorithm, IEEE Transaction on Computers, vol. 38 no. 5 pp. 741-745, May 1989.

[Wiener 48] Norbert Wiener, Cybernetics, or Control and Communication in the Animal and the Machine, MIT Press 1948.

[Weis 87] C.R. Weisbin, G. de Saussure, J.R. Einstein, F.G. Pin and E. Heer, Autonomous Mobile Robot Navigation and Learning, 1987 IEEE Conference on Robotics and Automation, Raleigh N.C. March 87

[Weis 89] C.R. Weisbin, G. de Saussure, J.R. Einstein, F.G. Pin and E. Heer, Autonomous Mobile Robot Navigation and Learning, Computer Vol. 22 No. 6 Jun. 89 p. 29.

[Wins 84] P.H. Winston, Artificial Intelligence, second edition, Addison-Wesley, Reading, Massachusetts, 1984.