Journal of Experimental & Theoretical Artificial Intelligence

ISSN: 0952-813X (Print) 1362-3079 (Online) Journal homepage: https://www.tandfonline.com/loi/teta20

Software and hardware architecture of advanced mobile robots for manufacturing

Michael Brady & Huosheng Hu

To cite this article: Michael Brady & Huosheng Hu (1997) Software and hardware architecture of advanced mobile robots for manufacturing, Journal of Experimental & Theoretical Artificial Intelligence, 9:2-3, 257-276, DOI: 10.1080/095281397147112

To link to this article: https://doi.org/10.1080/095281397147112

Published online: 09 Nov 2010.

J. Expt. Theor. Artif. Intell. 9(1997)257–276

Software and hardware architecture of advanced mobile robots for manufacturing

MICHAEL BRADY and HUOSHENG HU

Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, UK
tel: (+44)-1865-273154; fax: (+44)-1865-273908

Abstract. Modern manufacturing requires flexible and autonomous systems for materials handling and transportation. For such applications, a succession of implemented mobile robots has been developed over the past seven years. In this paper, the questions of software and hardware architecture are discussed in light of experience gained so far. Also discussed are mobile robot projects that seem realistic over the next five to ten years.

Keywords: Mobile Robots, Architecture, Sensing, Real-time Control, Manufacturing.

1. Introduction

The mobile robots, sometimes called autonomous guided vehicles (AGVs), currently deployed in flexible manufacturing are used primarily for transporting material or for cleaning. The vast majority are wire-guided, typically following a signal-carrying wire, or tracking a visible line on the floor (Premi and Besant 1983, Tsumura 1986, Hollier 1987). These vehicles have enjoyed considerable success mainly because of the end-point accuracy that they can achieve. However, they can only follow predetermined paths that comprise an unchanging network. Such AGVs need, and have, very limited sensing capabilities and in the presence of obstacles can only stop. Installation of wire guides can be costly and disruptive; but, more significantly, modifications are comparably expensive.

In order to achieve the new heights of performance needed for routine and flexible deployment in manufacturing, advanced mobile robots should be able to navigate their environment, build or update maps, plan and execute actions, and adapt their behaviour to environmental changes. They should also be able to cooperate with other robots and human operators, as well as learn something new from their experience. To perform these complex tasks in the real world, especially in real time, demands not only high computing power but also computing power of the right sort; that is, a concurrent hardware and software architecture.

More generally, architectural issues are key to the design and construction of real robots, just as they are for any computer-controlled complex system that is subject to hard (time) constraints. Considered from several points of view, (mobile) robots pose a tough challenge:

0952–813X/97 $12·00

- *Robots are fast, multi-degree of freedom devices* for which the plant kinematics changes as fast as the device moves. This is particularly challenging for robot arm control, but is non-trivial for vehicle control too.
- *It is necessary to sense the environment in order to determine how to act*. The key advantage of mobile robotics is that they can potentially operate in an environment that is variable and uncertain, hence sensing is vital. As it is rarely the case that any single sensor suffices for all situations, sensor data-fusion is necessary (Brady *et al*. 1990). Since sensor data is intrinsically noisy, and the range, reflectance, and other parameters of real sensors are limited, the control system must be able to deal with uncertain information about the environment. Sensors typically have high bandwidths, and there is inevitable processing latency, reducing and jeopardising the stability margin of the control.
- *Practically useful mobile robots must operate in real time*. It is useful to distinguish a number of control layers, which in turn poses the challenge of a multirate, multilevel system. Typically, an obstacle avoiding control loop operates at about 25 Hz. A global path planner needs to be able to deliver set points to the low level control layers, and be able to perform replanning of paths in case obstacles are encountered. A typical path planner might operate at 1–5 Hz. The ability to replan or repair a plan that seems almost right is much more useful than an optimal path planner which typically degrades disgracefully when they are subject to noise and the need to replan (which is almost always the case in practice).
- *Systems that are fielded in practice must be fail-safe, hence have redundant sensing, processing and actuation*. By processing, reasoning, planning, recognition, and modelling is meant. To be fail-safe, it is well to make processing distributed.
- *Real systems are judged by performance not by papers*. Much mobile robotics research focuses on relatively theoretical—simulated environment—and often computationally expensive. In fact, it is not clear that simply implementing such solutions on fast hardware will make them practical or even work.

Over the past decade there has been a great deal of interest in mobile robotics, and a great deal of rhetoric about issues such as representation and architecture. Although it is true that 'when all is said and done, a great deal more is said than done, a number of actual mobile robots have been constructed, some of which worked, some fewer of which appear to have worked well, and far fewer of which were actually transferred into regular practical use. If our views have merit, it is because we belong to the latter group. Over the past seven years, we have built a series of mobile robots for practical application, in close collaboration with industry. Nevertheless, our work has always been aimed at addressing generic research issue. Our perspective on the questions posed for the AAAI workshop that led to the current paper reflect this background.

2. Motivations

Clearly, there are many motivations for working on mobile robots in particular, and robotics more generally. First, as in our case, one might approach a problem from the standpoint of engineering science. In that case, one starts with a range of specifications of actual practical problems, for example clearing a baggage handling area at an airport, acquiring palletized objects in the loading bay of a factory, etc, and sets about defining a system that can meet the specifications. Primarily, success is judged by whether or not one's system meets the specifications and, assuming that it does, to

what extent it can cope with likely problems, is easily fixed, is costed attractively, and can be adapted to similar problems.

A second motivation may be to shed some light on human cognition. Such work may be considered good if it provides an explanation of certain observed cognitive behaviour, for example dealing with conflicts in reasoning or goals, and more importantly if it leads to testable predictions whose results are not known in advance but which turn out to be as predicted. Other motivations we have encountered include the desire to study control systems, and the desire to study artificial intelligence.

We are sure that there is very good work, entirely sited within the study of human intelligence, and which is based on mobile robots, or components of such systems. For example, there has been some superb research on the control of attention and eye movements by motor physiologists, and some excellent work on motion perception by psychophysicists. Our difficulty is certainly not with such work, but with systems that claim or advocate a close relationship between engineering and studies of humans. The limiting factor in the design of real robot systems turns out to be humble components such as motors, transmission elements, gear boxes, and sensors such as TV cameras.

The human eye controller, for example, has many degrees of freedom actuated by a mass of muscles that transmit power through tendons bathed in sinovial fluid. Unfortunately, no motor currently available, or likely to be available for many years to come, even remotely matches the performance of muscles, and sinovial fluid, being almost frictionless, has a coefficient of friction a tiny fraction of that of WD40. From an engineering standpoint, it is folly to build systems with more than a minimal number of degrees of freedom, in order to maximize stiffness, minimize friction loss, and maximize the parameters (e.g. acceleration, torque) that determine the usefulness of the system.

This fundamental difference is inevitably reflected in higher-level functions such as trajectory planners, path planners, and mission planners. Examples abound. Exactly similar comments apply to sensors: the TV camera has nowhere near the performance of the human eye, nor will it have for many years. In turn, the peculiar characteristics of TV cameras, for example poor dynamic range, low speed, anisotropy, uniformity, yet linearity, predetermine strategies for motion computation, intensity change detection, and ultimately stereo and recognition. Of course, there have been useful interactions at the level of stereo, most notably the *PMF* algorithm, but that embodies a geometric constraint (disparity gradient limit) not a peculiar aspects of human vision.

3. Structure and coordination

Any robot system or autonomous mobile robot needs constantly to process large amounts of sensory data in order to build a representation of its environment and to determine meaningful actions. The extent to which a control architecture can support this enormous processing task in a timely manner is affected significantly by the organization of information pathways within the architecture. The flow of information from sensing to action should be maximized to provide minimal delay in responding to the dynamically changing environment. A distributed processing architecture offers a number of advantages for coping with the significant design and implementation complexity inherent in sophisticated robot systems. First, it is often cheaper and more resilient than alternative uniprocessor designs. More significantly, multiple, possibly

redundant, processors offer the opportunity to take advantage of parallelism for improved throughput and for fault tolerance. Note that we distinguish the design of processing structure (architecture) from its realization in hardware and/or software.

Over the past two decades, a good deal of thought and effort has been dedicated to the design of architectures to tame complexity and achieve new heights of performance. Two principal designs have been adopted: *functional* and *behavioural* decomposition. Functional decomposition follows the classical top-down approach to build systems. The entire control task of a mobile robot is divided into subtasks which are then implemented by separate modules. These functional modules form a chain through which information flows from the robot's environment, via the sensors, through the robot and back to the environment via actuators, closing the feedback loop. Most previous mobile robots have been based on this approach, including, for example, hierarchical (Daily *et al.* 1988) and blackboard (Harmon 1987) architectures; but both of these have inherent limitations, including poor use of sensory information, reduced bandwidth causing bottlenecks, and difficulty in dealing with uncertainty.

In contrast, behavioural decomposition is a bottom-up approach to building a system. A behaviour encapsulates the perception, explore, avoidance, planning, and task execution capabilities necessary to achieve one specific aspect of robot control. That is, each is capable of producing meaningful action, and several such can be combined to form increasing levels of competence (Brooks 1986). In other words, each of them realizes an individual connection between some kind of sensor data and actuation. The system is built step by step from a very low level, say from locomotion, to obstacle avoidance, to wandering. Successive levels can be added incrementally in order to enhance the functionality of the robot.

This design method has grown in popularity recently. In the subsumption architecture, for example, control is distributed among those task-achieving behaviours that operate asynchronously. Higher layers can subsume the operation of the lower ones when necessary, only one layer actually controlling the robot at any one time. Since each layer achieves a limited task, it requires only that information which is useful for its operation. It is claimed that the control system can respond rapidly to dynamic changes in the environment without the delays imposed by sensor fusion. But

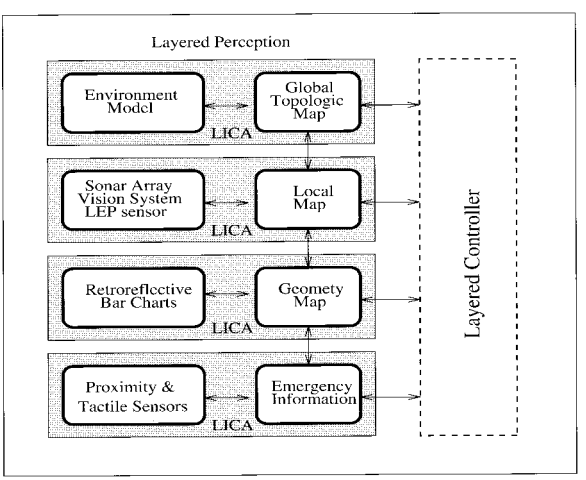

Figure 1. Concurrent sensing system.

the implementation of higher layers of competence still poses a problem. More careful initial design in specifying the communications and modularity is required. Moreover, the higher levels often rely on the internal structure of lower levels, thus sacrificing modularity.

Neither of these approaches suffices, since the control of a mobile robot is so complex that one cannot strictly adhere to one decomposition scheme while completely ignoring the other. Each has benefits and drawbacks. We have developed and

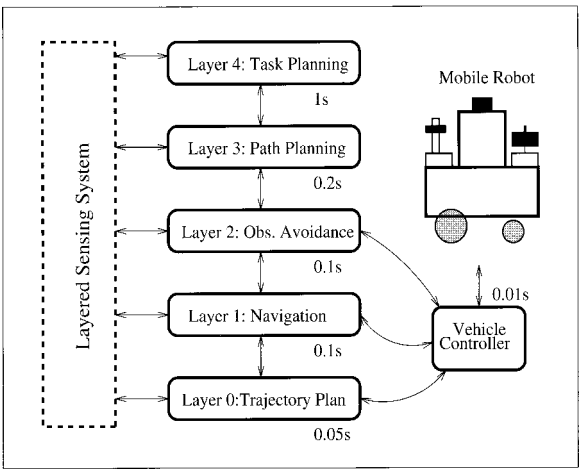

Figure 2. Concurrent control system.

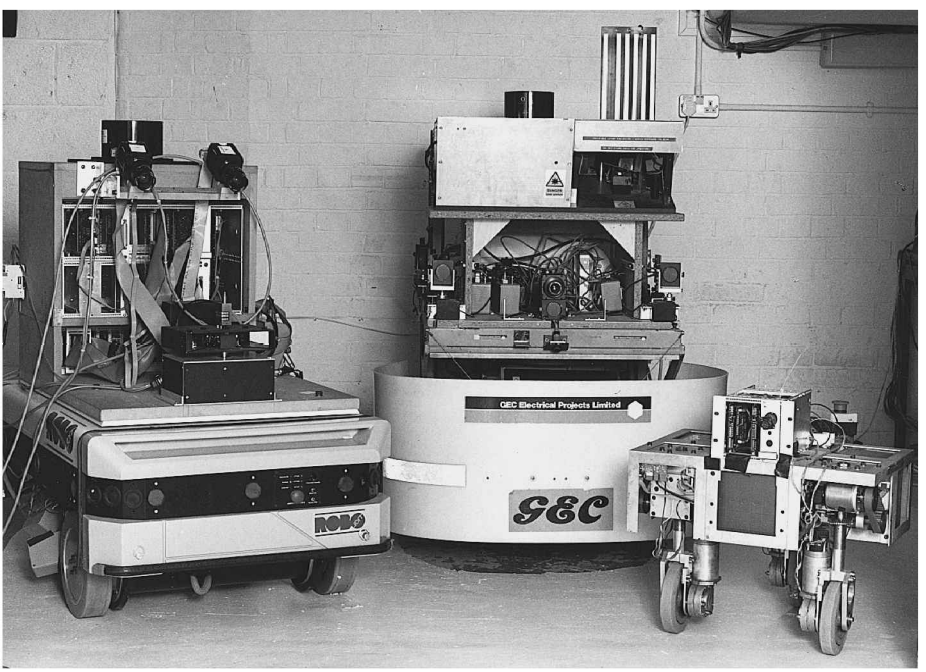

Figure 3. Oxford AGVs.

implemented an architecture that is, we contend, a blend of the best features of each. It consists of a distributed community of sensing, action and reasoning modes. Each of them has sufficient expertise to achieve a specific subtask, following a hierarchical decomposition scheme. Few of them are composed of a single task-achieving behaviour as in a behavioural decomposition scheme. The key point is that the control tasks of the mobile robot is distributed among a set of behaviour experts that tightly couple sensing and action, but which are also loosely coupled to each other. In this way, sensor data can be used directly in a corresponding layered control task to form a task-achieving behaviour. This differs from the functional decomposition in which the combination of subtasks is a single processing chain. To function effectively, each layer requires a significant amount of self-knowledge to allow it to decide what information it can supply to others; how best to recover from local errors, as well as what to do if no information is sent from other layers.

Basically, there are two subsystems in our design: a layered perception system shown in Figure 1 and a layered control system shown in Figure 2. The layered perception system is used for active sensor control and distributed sensor fusion to support a layered control strategy. The layers process raw sensor data from internal sensors (tachometer, encoder, resolver) and external sensors (sonar, vision, laser scanner, LEP sensor) to build up models in a bottom-up manner. All the sensing layers operate independently and are loosely coupled. Communication between them is used to realize sensor data fusion.

The design of the layered controller is based on the observation that different response times are demanded by different tasks. The lower levels perform simple, general tasks such as *smooth path guidance* and *avoiding obstacles* for fast reactivity. The higher levels perform more complex, situation-specific tasks such as *path planning* and *monitoring*. All layers operate in parallel (Hu *et al.* 1993). Such a design has been adopted in different mobile robots existed in our group to implement real-time sensing, planning, and control, three of which are shown in Figure 3. It should be noticed that each layer of our distributed real-time architecture consists of a control node and a sensing node. Each sensing node delivers a representation, which, in the context of a given task, causes a corresponding control node to generate commands.

4. Modularity and Scalability

There are several issues that concern the realization in hardware or software of distributed architectures such as those described above. For example, one key issue is granularity: a fine-grained architecture is generally considered to comprise many thousands of individual processing elements, while a coarse-grain architecture comprises just a few tens or hundreds. In every case, there is a trade-off between computation and communication. Granularity greatly influences software design. We have adopted a coarse granularity based upon a well-developed theory of inter-process communication (Hoare 1985) and commercially available processors to implement it.

4.1. *LICA concept*

To implement the complex system proposed above, we have fixed on a processing module that we call a *Locally Intelligent Control Agent* (LICA). It is based on the concept shown in Figure 4. The LICA design is a standard approach to the design and construction of a complex system and to maximize modularity. It is independent of any particular processor. Central to such a design is the concept of *Communication*

Sequential Processes (CSP) which was developed by Hoare (1985). Although we have already implemented the LICA concept using *transputer* technology (Hoare, 1985), we have begun to investigate the applications of other processors such as *SIEMENS 80C166*, *PowerPC*, and *DEC alpha* processors into the LICA architecture.

The transputer was designed to execute CSP efficiently and specifically for dedicated applications in embedded real-time systems. Its high-speed serial links (up to 20 Mbit/s) provide a convenient way to build a parallel processing computer architecture. It has outstanding capability for input and output, and an efficient response to the interrupts. These features are supported by a timesharing (multi-programming) operating system which is built into the chip. An application program can therefore be structured naturally as a collection of loosely coupled parallel processes, modelling the parallel structure.

Dozens of LICAs have been built for a variety of purposes, including high-level planning functions, low-level vehicle control, and for sensory data interpretation and fusion. Figure 4(a) shows the logical functions of a typical LICA, which includes drive interface to external sensors and motors, computing power and memory, as well as high-speed serial links. A pair of RS-422A drivers and receivers are included in a LICA for the point-to-point signals to be carried at a speed of 20 Mbit/s for up to a distance of 20 m. We have also build a range of different interface modules, such as a sonar TRAM, a linear camera TRAM, an A/D, a D/A, a digital I/O, a RS232, a counter/timer or any combination of these modules (Hu *et al.* 1993).

Each LICA contains its own localised function and its own localized knowledge source, as shown in Figure 4(b). Each LICA uses high-speed serial links to communicate directly with other LICAs. There is no shared memory. The LICAs run independently, monitoring their input lines, and sending messages on their output lines. Once a LICA has input data to work on, the data is processed and the results are held for a certain predefined time for other LICAs to read. If no valid data is available, the LICA gives up its input process and continues with other processing.

4.2. *A modular architecture*

The LICA encourages a modular approach to building complex control systems for intelligent mobile robots to perform navigation, planning, and control tasks in real time. The LICA-based modularity offers flexibility in changing the system

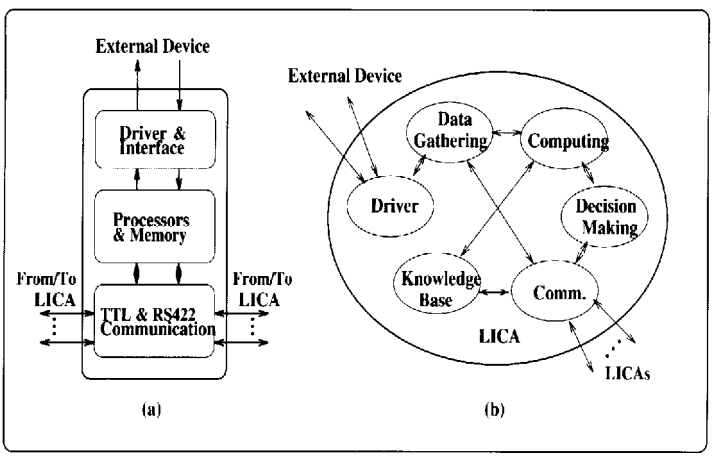

Figure 4. Diagram of LICA concept.

configuration between the different applications by exploiting point-to-point communications. It provides a degree of robustness and reliability, as well as a cost-effective solution to problems of designing complex robot systems.

Figure 5 shows that the proposed control system is built from 23 LICA boards. It should be noticed that the number of LICAs for each task-achieving behaviour such as path planning and obstacle avoidance is flexible, and can be easily changed. This depends upon the particular sensor and the system time constraints. The topology of the network can be reconfigurated flexibly by twisted link cables according to different experiments or applications.

Figure 6 shows the corresponding software structure. It is implemented by INMOS C which is a special configuration language able to distribute processes over a network of transputers and to create a multi-processor system very quickly. Note that there are eight main processes, each of which in turn consists of many subprocesses which are to exploit the support of parallelism in transputers. All the processes are correspondingly mapped into the LICA-based transputer network shown in Figure 5, and operate in parallel. A typical mapping for the active stereo vision head onto the LICA architecture is sketched in the next section to show scalability of the system. The mapping for other sensors and the layered controller can be found in Hu *et al.* (1993).

4.3. *The scalability of the system*

The LICA design not only provides modularity for building a complex robot system, but also provides scalability to the system to enable it to be used for different applications. Figure 7 shows that the LICA architecture can be scaled into three versions. The top part presents an ASIC version of LICA used to increase its interfacing capability to different sensors and actuators. The microprocessor 80C166 is currently used for this version, which can be changed to other processors as well. The middle part gives the normal version LICAs which compromise the power for

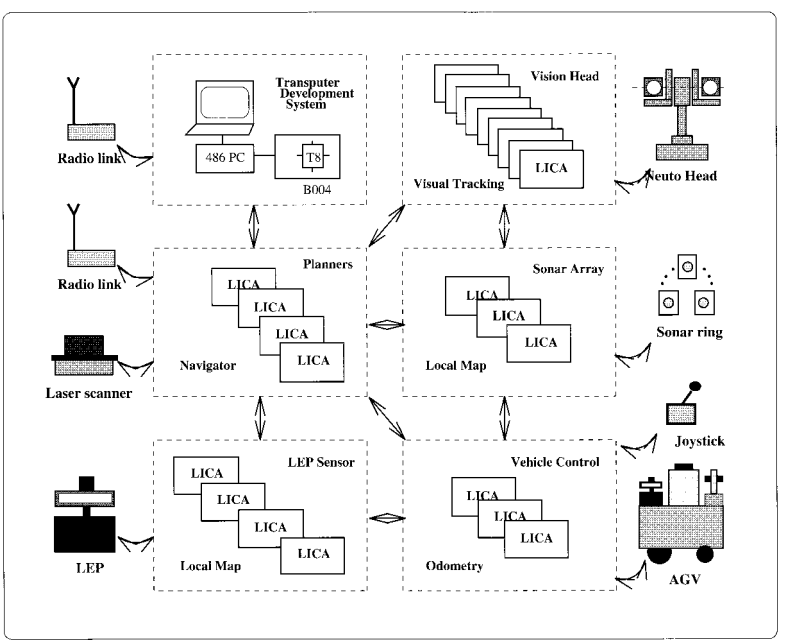

Figure 5. A LICA-based system architecture for our mobile robot.

computing and the power for interfacing external devices. However, the bottom part gives the LICAs with high computing power by adding Texas C40, IBM PowerPC, and DEC Alpha into LICA architecture. This version is required by high bandwidth system such as real-time vision systems.

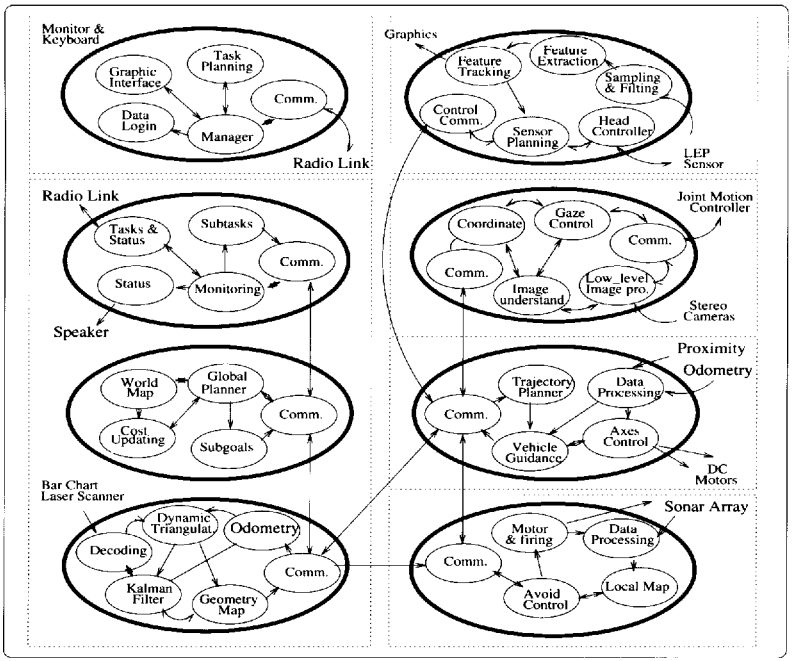

Figure 6. Software structure for the control system.

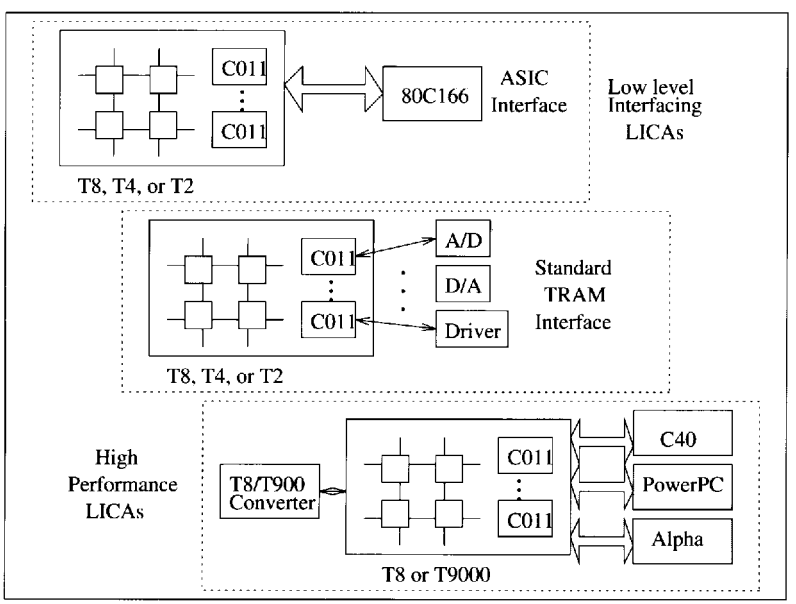

Figure 7. Scalability of the LICA-based architecture.

A LICA-based active vision system (Du and Brady 1994) is one example to demonstrate system scalability. The vision system enhances the visual navigation capabilities of the system when it is confronted by a dynamic environment and explores the potential of using active vision for visual navigation. Based on a four degree-of-freedom stereo robot head, the vision system is able to actively change its geometry to adapt to the task requirements and is able to plan and execute accurate camera motions (Hu *et al.* 1995). Figure 8 shows that the system has several functional blocks, namely:

- *The robot head* and *its real-time motion controller* are responsible for carrying out the desired joint motions.
- *The gaze controller* is responsible for generating and controlling a set of useful behaviours which are important for visual information processing.
- *The low-level image processing* is generally task independent. It provides on demand a set of low-level visual features and an estimate of visual motion, and transmits them to high-level image processing and the gaze controller.

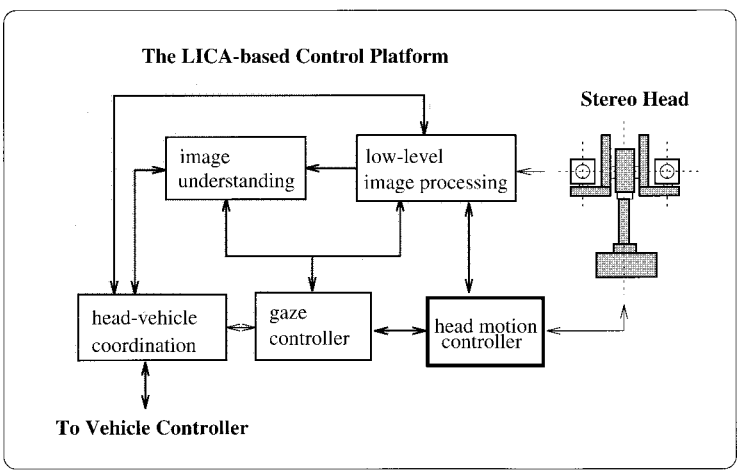

Figure 8. The active vision subsystem.

Table 1. The performance of the head.

Joint	*Verges*	*Elevation*	*Pan*
Resolution	0·0036°	0·0018°	0·0036°
Max. joint acc.	7000°/s^2	1000°/s^2	2000°/s^2
Max. joint rate	530°/s	200°/s	450°/s
30° slew time	0·14 s	0·34 s	0·23 s
90° slew time	0·25 s		0·4 s
Position error	< 1 count	< 2 count	< 2 count
Dead time	< 5 ms	< 10 ms	< 10 ms
Repeatability	0·0075°	0·13°	0·0075°
Min. speed	< 0·1°/s	< 0·1°/s	< 0·1°/s
Motion range	±60°	±75°	±90°

- *The high-level image processing* is used to interpret the information according to the task requirements. Head-vehicle coordination controls the overall behaviour of the active vision system and information exchange between vision and the vehicle controller. It must initialize appropriate behaviours and image processing algorithms to achieve its current purpose.

The joint motion control of the robot head is based on a compact, high-performance, real-time joint motion controller (which is implemented on a 6U Eurocard (160×230 mm^2). The real-time joint motion is controlled by PID controllers working at 2·93 kHz. The performance of the head is summarized in Table 1.

The computing/control platform of the active vision system is firstly implemented using a network of 32 T800 transputers on nine LICA boards. To maintain compactness, two transputer-based frame grabbers are used to capture the images. The architecture of the vision system is shown in Figure 9. That is, the network topology can be changed both locally and globally to implement specific tasks. For low-level image processing (monocular image processing for the left and right cameras) short length pipelines and intensive parallelism are used to provide high sample rate and short latency. These are key to achieving real-time control of actions (head motions). The data fusion represents middle-level image processing and perception correspond to high-level reasoning and image understanding. This architecture enables us to integrate various strategies and algorithms into the system.

The maximum tracking speed achieved by the stereo head in our experiments is about 50%. In a typical experiment, a torch held by a person is moving at a speed of

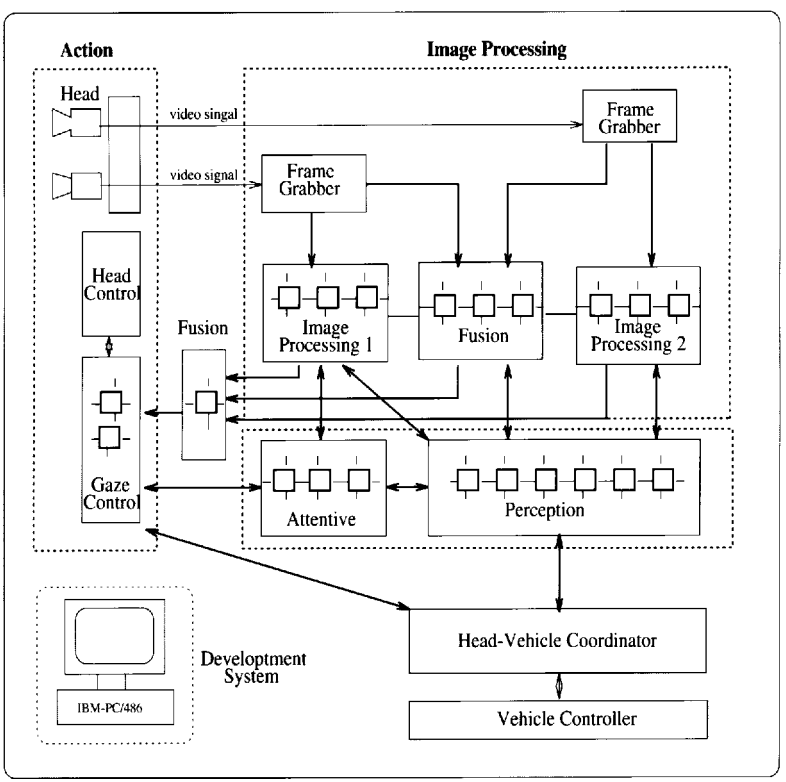

Figure 9. The architecture of the vision system.

about 10 cm/s. The vision head tracks its light and locates its position and angle relative to itself. Then such information is used to calculate the moving speed and steering for the mobile robot to pursue it. The mobile robot will follow the spot light within a certain distance, and will backtrack if the torch is too close (within 1·5 m safe distance).

5. Representation and planning

Planning is central to purposive behaviour. To plan is to formulate a strategy to achieve a desired goal. An inability to plan implies purely reflexive responses, with an agent stumbling aimlessly about in response to unexpected external events. Most reactive systems, especially MIT's creatures, are characterized by a stimulus–response type of relationship with the real world. The representation of the real world is deliberately avoided because of the belief that *the world is it's own best model* (Brooks 1988). However, for manufacturing applications, it is simply impossible for systems to do anything useful. In fact, world modelling plays a vital role and presents a significant challenge for efficient and flexible material handling implemented by autonomous vehicles.

However, planning has traditionally been treated as a purely predictive process where complete knowledge is assumed *a priori* about the environment either in the form of a qualitative or a quantitative model. A complete sequence of steps that solve a problem or reach a goal are determined before taking any action. Recovering from error and dealing with unexpected events is usually left as a subsequent execution stage. As a result, traditional planning systems are inherently open-loop and unable to handle unmodelled disturbances in the real world.

Planning embraces a broad range of abilities, and for manufacturing purposes it is useful to distinguish four: *mission planning* is the process of determining the requirements and constraints for the global tasks and obtaining an optimal schedule for multiple goals. *Global path planning* aims to find a collision-free path (as set of intermediate points) for a mobile robot to follow from the start position to the goal position. In contrast, *local path planning* generates relatively local detours around sensed obstacles while following a globally planned path. Finally, *trajectory planning* generates a nominal motion plan consisting of a geometrical path and a velocity profile along it in terms of the kinematics of the individual robot.

What we call mission planning is what is normally called planning in cognitive AI. For example, planning a journey to Koh Samui, subject to constraints (e.g. to get there within a week), probably involves flying via Bangkok rather than walking. Planning the assembly of a complex device such as a car engine inevitably involves hierarchical decomposition into substructures, each planned separately but respecting interfaces. Military mission planning involves the disposition of forces, continually monitoring enemy movements, and anticipating then countering threats.

Global path planning for a mobile robot in a known environment with known static objects has been studied extensively. Graph searching and potential field methods are the two main approaches used to solve the path-finding problem. The main feature of both methods is that the entire environment is modelled geometrically before the motion takes place. In the graph searching approach, a graph is created that shows the free spaces and forbidden spaces in the robot's environment. A path is then generated by piecing together the free spaces or by tracing around the forbidden area. In contrast, the potential field approach uses a scalar function to describe both objects

and free space. The negative gradient of the potential field precisely gives the direction to move to avoid obstacles. It offers a relatively fast and effective solution for safe paths around obstacles.

In the real world, both mission planning and global path planning have to react to unforeseen events, modify plans accordingly, contending all the while with uncertain information, and do so against constraints such as time. The earliest influential planning system, STRIPS (Fikes and Nilsson 1972), planned a sequence of moves through a set of rooms to reach a desired location. Unlike most subsequent planners, STRIPS was connected to a plan execution software module, PLANEX, which orchestrated the execution of the planned path devised by STRIPS on a sensing mobile robot SHAKEY. SHAKEY's sensors could detect unexpected obstacles, causing PLANEX to suspend its script, halting the robot, so that STRIPS could be involved afresh in the new situation. In this way, Shakey's behaviour alternated mindless script following and long periods of planning, which was regarded as a form of reasoning.

Recent work in planning has aimed to overcome this limitation. A variety of proposals have been made to develop the idea of a situated agent (Agre and Rosenschein 1995), a software object that is intended to be in continuous interaction with the world. Sadly, the vast majority of situated agents only inhibit a simulated world in which the problems of noise, uncertainty, and clutter are absent (but see Rosenschein and Kaelbling, 1986 for a notable exception).

Our work has concentrated on planning for a mobile robot to operate in a dynamic manufacturing environment autonomously and efficiently. The first issue addressed is how to model the dynamic, uncertain environment in a manner that makes it possible to provide a solution for an optimal path constrained by the real world. Since a single, centralized and complicated model of a dynamic environment takes a long time to compute, thereby reducing system response speed, we employ several simple models, each tuned to a different range and resolution of situations for different tasks, shown in Figure 10. In other words, this multi-model control architecture takes the real-time capability and the expected task-achieving behaviours into account.

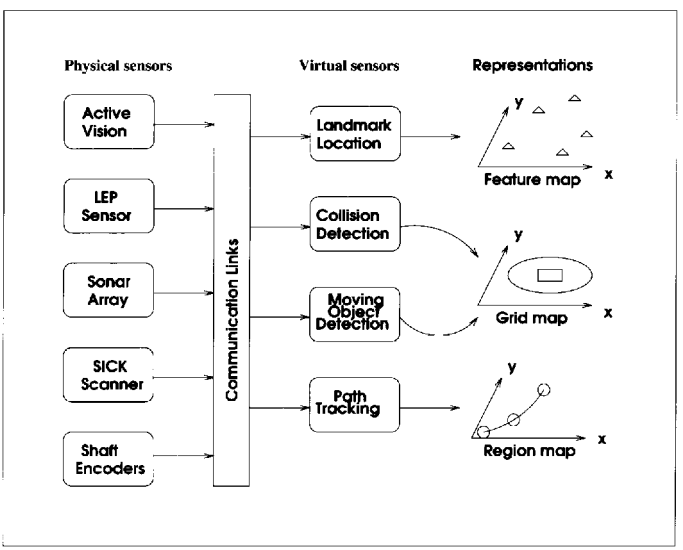

Figure 10. Multiple Sensors and Multiple Models.

In global path planning, we have proposed a probabilistic approach to address the uncertainty problem of mobile robots operating in a dynamic real world (Hu and Brady 1992). Two types of world knowledge have been used in our approach to achieve real-time performance. One is *a priori* information about the robot's environment which is considered relative static. The other one is updated information acquired by the sensors as the robot moves around, which is normally unpredictable and is dynamically changing.

Instead of using an uncertainty grid, we use a topological graph and roadway to represent the free space of the robot's environment, weighted by a scalar cost function, $C(\mathrm{x}(k))$, which represents the effort for a mobile robot to move from the state $\mathrm{x}(k-1)$ to the next state $\mathrm{x}(k)$. It consists of two parts: deterministic cost ($T(\mathrm{x}(k))$ and uncertainty cost $U(\mathrm{x}(k))$ as follows:

$$C(\mathrm{x}(k)) = T(\mathrm{x}(k)) + U(\mathrm{x}(k)) \tag{1}$$

The deterministic cost depends on path parameters such as length, width, and straightness which are easy to model. To quantify uncertainty, we have built the statistical models to form uncertainty costs for unexpected events. These uncertainty costs are updated by available sensor data when the robot moves around. Given an initial state x_s and the goal state x_g, an optimal path is found by choosing a sequence of admissible decisions:

$$\pi = \{\mathbf{d}(0), \mathbf{d}(1), \ldots, \mathbf{d}(N-1)\} \tag{2}$$

in such a way as to minimize the accumulated cost (expectation) expressed by the sum

$$J = E\{C[\mathrm{x}(N)] + \sum_{k=0}^{N-1} C[\mathrm{x}(k)]\} \tag{3}$$

subject to the system constraints

$$\begin{aligned} \mathrm{x}(0) &= \mathrm{x}_s \qquad & \mathrm{x}(N) &= \mathrm{x}_g \\ \mathbf{d}(k) &\in \mathbf{D} & \mathrm{x}(k) &\in \mathbf{X} \end{aligned} \tag{4}$$

Analogous to the comment made earlier about mission planners, in traditional planning systems global path planning and local path planning are sequential processes. In other words, the local planner is idle when the global planner is busy, and vice versa. This causes a delay for the whole system, since the local planner, whenever a preplanned path is blocked, triggers the global planner and then has to wait for an alternative path. Concurrent processing of both planners is crucial for avoiding delays. In our design, however, the global planner generates alternative subgoals dynamically when the robot is travelling along a preplanned optimal path. In other words, it is always assumed that the next node of the preplanned path may be blocked by unexpected obstacles. If this turns out to be true, the local planner can backtrack along these subgoals without delay. However, if nothing happens when the robot approaches the next node, the alternative subgoals provided by the global planner will be ignored. The system has been implemented and performs real-time local and global path planning and obstacle avoidance. Dynamic replanning is performed as necessary based on the decisions that are rooted in sensory information.

Any robot or animate system that operates continuously in the real world must rely on information provided by its sensors. It is not possible, regardless of how many sensors are used or how discriminating they are, to observe directly everything of interest, all the time, and with complete accuracy: the sensors available may not be

able to make direct observations of everything of interest. As we noted above, there is always a limit to the temporal sampling; and all sensors are subject to noise and error. Therefore, a planning system is not able to maintain a complete picture of the environment with complete certainty.

6. Coping with uncertainty

A system which plans under uncertainty must maintain multiple possibilities for the state of the world, and associate with each some degree of believe. Some alternative world states will be more likely than others; for example the system must allow for the possibility that the sensor data is incorrect. New sensor data about the world must be used to update these possibilities and beliefs; some alternatives may be ruled out, new alternatives generated, while others may become more or less likely.

Let us suppose, for example, that a mobile robot is commanded to travel along a path generated by a global path planner. While traversing the path, an unexpected obstacle appears and the possibility of a collision arises. Since sensors inevitably have limitations on their range and accuracy, they may not be able to tell exactly whether the gap between the obstacle and a known object is wide enough for a sidestep manoeuvre when the mobile robot is some distance away. A decision is required as to whether the robot should continue its motion along the path, to make further observations to manoeuvre (at a certain cost) or, alternatively, to follow a different path and incur a certain loss. If the cost of doing extra sensing is less than the cost of taking the alternative path, it may be worth persevering along the original path. But the robot may eventually find the gap impassable, incurring an overall cost greater than immediately abandoning the planned path and following the alternative. Hu and Brady (1994) adopt a Bayesian decision theoretic approach to this problem. First, a probabilistic model is formulated of the (sonar) sensory information available to the robot. A loss function is defined that provides the outcome of an action (e.g. sidestep) given the path state (e.g. passable. Then that action is chosen which minimizes the Bayes risk.

The Bayesian framework is but one of a number of approaches to uncertainty that has been explored in AI. Pearl (1988) makes the following classification of AI approaches to uncertainty: logicist, neocalculist, and neo-probabilist. Logicist approaches use non-numerical techniques for dealing with uncertainty, mainly non-monotonic logics. Neo-calculist approaches use a numerical representation of uncertainty, but invent new calculi, considering the traditional probability calculus inadequate; examples are Dempster–Shafer calculus, fuzzy logic, certainty factors (see Nicholson 1992 for references). The neo-probabilist school, which includes our work, remains within the traditional Bayesian framework of probability theory but adds the computational facilities required by AI tasks.

There have been many objections by the AI community to the use of probability, including the observation that people seem to be bad probability estimators. When the planning/navigating system asserts that 'The chances that an object in region X at time T will move to region Y is p', the important thing is not the precise magnitude of p, so much as the specific reason for this belief (the sensor data, its schedule, or its previous movement), the context or assumptions under which the belief should be firmly held, and the information which would lead to this being revised.

Belief networks allow the representation of causal relationships, and provide a mechanism for integrating logical inference with Bayesian probability. Belief networks are directed acyclic graphs, where nodes correspond to random variables, say world

states or sensor observations. The relationship between any set of state variables can be specified by a joint probability distribution. Evidence can be provided about the state of any of the nodes in the network. This evidence is propagated through the network using a bidirectional (message passing) mechanism, affecting the overall joint distribution.

We return to belief networks soon, but first we pause to consider an extremely useful first cousin, the Kalman filter. The Kalman filter maintains an estimate of the state x of a system and a covariance estimate P of its uncertainty. It assumes that the dynamics of the system can be linearized, so that the transformation from the state at the kth time step to the $k+1$th are given by a matrix (linear) operation on the state, but corrupted by noise. Its simplest form can be shown in Figure 11.

In our work, we have built on an industrial system that used a feature-based approach to localization based on artificial beacons, shown in Figure 12. The position of the robot is determined by integrating odometry $\mathbf{u}(k) = [d(k), \alpha(k)]$ with beacon information (x_i, y_i, ϕ_i) provided by a laser scanner.

Assume that the system model describes how the robot state $\mathbf{x}(k)$ changes over time in response to a control input $\mathbf{u}(k)$:

$$\mathrm{x}(k+1) = \mathbf{F}(\mathrm{x}(k), \mathbf{u}(k)) + \mathbf{w}(k) \tag{5}$$

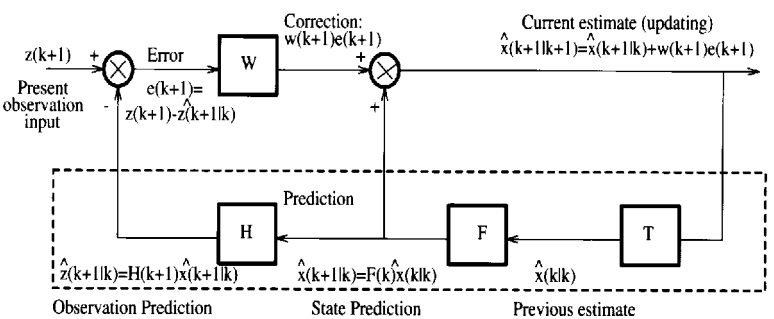

Figure 11. The information flow of the Kalman filter.

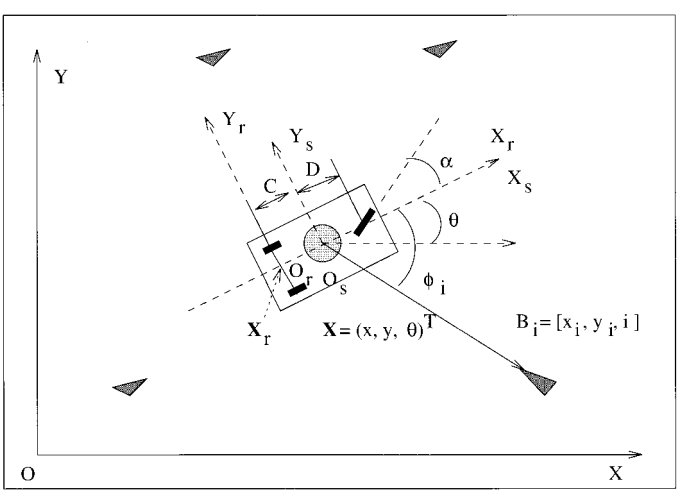

Figure 12. The localization system based on angle observations. Note that XOY is the world frame; $X_rO_rY_r$ is the robot frame; $X_sO_sY_s$ is the scanner frame.

where $\mathbf{w}(k) \sim \mathbf{N}(\mathbf{0}, \mathbf{Q}(k))$, and $\mathbf{F}(\mathrm{x}(k), \mathbf{u}(k))$ is the nonlinear function:

$$\mathbf{F} = \begin{bmatrix} x(k) + d(k)\cos\alpha(k)\cos(\theta(k)+\alpha(k)) \\ y(k) + d(k)\cos\alpha(k)\sin(\theta(k)+\alpha(k)) \\ \theta(k) + d(k)\sin\alpha(k)/(C+D) \end{bmatrix} \tag{6}$$

and $(C+D)$ (wheel base) is the distance from the steer point to the centre of the rear axle.

The measurement model is an angle observation ϕ_i in terms of the system state $\mathrm{x}(k)$ and the position of the beacon being detected, $B_i = [x_i, y_i]$, under the assumption of zero-mean Gaussian noise:

$$\mathrm{z}(k) = \mathbf{h}(B_i, \mathbf{x}(k)) + \mathbf{v}(k) \tag{7}$$

where $\mathbf{v}(k) \sim \mathbf{N}(\mathbf{0}, \mathbf{R}(k))$, and the measurement function $\mathbf{h}(B_i, \mathbf{x}(k))$ is also nonlinear, given by:

$$\mathbf{h}(B_i, \mathbf{x}(k)) = \arctan\frac{y_i - y(k)}{x_i - x(k)} - \theta(k) \tag{8}$$

Since both models are nonlinear, the Extended Kalman filter (EKF) algorithm is used. It operates in a 'prediction–observation–matching–correction' cycle to provide an estimate of the robot location.

1. *Prediction*

$$\hat{\mathrm{x}}(k+1|k) = \mathbf{F}(\mathbf{x}(k|k), \mathbf{u}(k)) \tag{9}$$

$$\mathbf{P}(k+1|k) = \nabla\mathbf{F}\mathbf{P}(k|k)\nabla\mathbf{F}^{\top} + \mathbf{Q}(k) \tag{10}$$

where $\nabla\mathbf{F}$ is the Jacobian of the transition function (6), and is obtained by linearization $\partial\mathbf{F}/\partial\mathbf{x}$.

2. *Observation*

The predicted observation, innovation, and innovation covariance are

$$\hat{\mathbf{z}}(k+1) = \mathbf{h}(B_i, \hat{\mathrm{x}}(k+1|k)) \tag{11}$$

$$\mathscr{V}(k+1) = \mathrm{z}(k+1) - \hat{\mathbf{z}}(k+1) \tag{12}$$

$$\mathbf{S}(k+1) = \nabla\mathbf{h}\mathbf{P}(k+1|1)\nabla\mathbf{h}^{\top} + \mathbf{R}(k+1) \tag{13}$$

where the Jacobian $\nabla\mathbf{h}$ is found by linearization $\partial\mathbf{h}/\partial\mathbf{x}$.

3. *Matching*

For each measurement, we use a validation gate to decide whether it is a match or not:

$$\mathscr{D}(k+1) = \mathscr{V}(k+1)\mathbf{S}^{-1}(k+1)\mathscr{V}^{\top}(k+1) < G^2 \tag{14}$$

where G^2 is the validation gate. If a measurement falls into such a gate, we get a successful match. Otherwise, we simply ignore it and look for next measurement.

4. *Correction*

The updated gain, state, and covariance matrix are

$$\mathbf{W}(k+1) = \mathbf{P}(k+1|k)\nabla\mathbf{h}^{\top}\mathbf{S}^{-1}(k+1) \tag{15}$$

$$\mathrm{x}(k+1|k+1) = \hat{\mathrm{x}}(k+1|k) + \mathbf{W}(k+1)\mathscr{V}(k+1) \tag{16}$$

$$\mathbf{P}(k+1|k+1) = \mathbf{P}(k+1|k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}^{\top}(k+1) \tag{17}$$

In this way, the robot's position and orientation are predicted by dead reckoning and corrected by sensed information provided by a laser scanner. This is implemented recursively to reduce the discrepancy between the planned and actual states of a robot which increase, as does the state uncertainty, when no sensor measurements are made. The Kalman filter has the property that, under certain reasonable assumptions, it is the optimal state estimator. Note that it is without problems in practice. Among the more severe of these are (i) the difficulty of computing a good *initial* state estimate; (ii) the difficulty of determining appropriate gain matrices; and (iii) the difficulty of identifying and approximating real plants in the simple form shown. The Kalman filter has much been used at Oxford (and elsewhere) to guide robot vehicles, tracking moving shapes, and compute egomotion.

The Kalman filter is related to the distributed processing discussed in the previous section. In a typical fielded system, a set of sensors make independent measurements of components of the state and report them, at each step, to a central processor that runs the Kalman filter. If one of the sensors ceases to operate, the system continues to run, albeit it with increased state uncertainty. If, however, the central processor ceases to operate, the whole system fails. An obvious alternative design is to distribute the Kalman filter among the sensors (in the current parlance this makes them 'smart' sensors) and enable them to communicate their state estimates among themselves. Rao *et al.* (1993) have shown that the equations of the Kalman filter can be partitioned so that such a fully decentralized system converges to the same global optimum as the centralized system. The system degrades gracefully as 'smart' sensors cease to operate, and upgrades gracefully as new 'smart' sensors are introduced into the system.

Dickson (1991) notes 'The correspondence between Bayesian networks and Kalman filters should come as no surprise, as both are rigorous applications of Bayes' theorem, each in their particular domain. While the Kalman filter gains its power and efficiency from the theory of linear transformation, no such simple mapping is available between the nodes of the Pearl network ... The condition independence which is the centre piece of Bayesian networks is of more ancillary importance to the application (but not the theory) of Kalman Filters'. Nicholson (1992) has explored methods for constructing a predict–measure–update cycle, analogous to a Kalman filter, but liberated from the latter's dependence upon a linear algebraic representation of state and state change. The choice of belief networks is well-founded since the connection between them and Kalman filters has been established by Kenley (1986).

7. Simulation

Simulation in robotics, control theory, and AI has mostly been a complete waste of time. Of course, there are certain cases in which simulation is inevitable, for example because it involves designing a system to operate in conditions that cannot, or one hopes will not, obtain in practice. These include designing a system to operate on a planet, say Mars, or in a hostile environment such as a battle, or in an abnormally operating nuclear power station. Even in such cases it is never necessary nor wise to rely on simulation alone. There are many landscapes on earth that resemble that on Mars, and systems may be asked to operate in a fallen down building to simulate the power station. Equally, it is of course a good idea to use simulations in the development of a system, e.g. to help debug hardware or programs: what is at issue is whether results 'demonstrated' using simulation *only* should be accorded worth. We think not.

Why has simulation proved to be of such little use in practice? Simply because it turns out to be very difficult indeed to simulate to any reasonable degree of approximation the real world. Much better to immerse a system in the real world. Simulated sensors rarely if ever behave like real sensors, simulated obstacles and simulated objects population the simulated world rarely resemble real objects. Simulated robots tend not to have friction, stiction, or to slip. To be sure, it is much easier to build simulated systems that live entirely inside the comfortable world of a workstation. For this reason, simulations have flourished but have not added much to the stock of knowledge of robotics.

8. Learning

Of course, the ability to learn is important in practice for a mobile robot. First, the robot itself changes, not least if it picks up objects for delivery or disposal. There is a well-developed theory of system identification and adaptive control, which has been applied to robots by Hollerbach, Atkeson and An. Equally, the robot's environment may change: to take a familiar example, the road between one's home and work may be subject to repair and so to unacceptable delays. We have built in to our Bayesian decision system the ability to uptake its environment model to cope with such changes, and to gradually forget them so as to try the original route once more. It is, in fact, desirable that the robot should learn its environment model, for at least two reasons. First, it is tedious to specify a model of even a moderately complex environment. Second, and more significantly, the environment model is determined by the sensors. A typical model of an indoor laboratory used in current mobile robotics imagines ideal planar walls. However, walls are rarely ideal planes; more usually there are holes in the wall surface, for example for electricity sockets. A mobile robot equipped with a sonar sensor will typically 'hear' such holes absolutely repeatedly and reliably, hence they can, and should, appear in the environmental model. Integrating such sensor models for a variety of sensors is a largely open problem.

9. Where next?

The use of mobile robots for parts handling and delivery is increasingly common in manufacturing industry. There are hundreds of fielded systems. Nearly all of them are tracked or line following, because currently such systems are far more accurate than 'free ranging' systems that locate their position using retroreflecting beacons. Major application opportunities are opening up in outdoor mobile robotics: (i) stockyard parts acquisition and delivery; (ii) subsea; (iii) dockside automation; (iv) inshore navigation; (v) civil construction; and (vi) parts delivery inside nuclear power stations. All of these are characterized by imperfect information generated by application specific sensors from which decisions must be made in real time. In each case, a fixed environmental model suffices, with occasional updates. Cases (i), (ii) and (iii) predominantly require advances in sensing and sensor-based obstacle avoidance; case (iv) primarily sensor data-fusion; and case (v) high accuracy (typically 5 mm over a range of 100 m). Applications that are often talked about, but which seem to have very limited likelihood of being fielded over the next decade, include object delivery in hospitals and offices; robot 'guide dogs', and sentries, e.g. in prisons. It is fun to imagine far out applications: measurable progress will come from work on real problems.

10. Acknowledgements

We wish to thank Penny Probert and Hugh Durrant-Whyte for many insightful conversations, and the staff at GCS for their continuing help and encouragement. Thanks to Fenglei Du (now at Amherst Systems Inc.), Nick Pears (now at Cambridge University) and many people within the Oxford Robotics Research Group for contributing towards the project. Thanks also to the EPSRC ACME for their financial support under grant $GR/K39844$.

References

Agre, P. E. and Rosenschein, S. J. (1995) *Special Issue of Artificial Intelligence* (The Netherlands: Elsevier Science B.V.).

Brady, J. M., Durrant-White, H., Hu, H., Leonard, J. J., Probert, P. J. and Rao, B. S. Y. (1990) Sensor-based control of AGVs, *IEEE Journal of Computing and Control Engineering*, **1**: 64–70.

Brooks, R. A. (1986). A robust layered control system for a mobile robot, *IEEE Journal Robotics and Automation*, **2**: 14.

Brooks, R. A. (1988) Intelligence without representation. Technical report, Artificial Intelligence Laboratory, MIT.

Daily, M., Harris, J., Keirsey, D., Olin, K., Payton, D., Reiser, K., Rosenblatt, J., Tseng, D. and Wong, V. (1988) Autonomous cross-country navigation with the alv, *Proceedings IEEE International Conference Robotics and Automation*, Philadelphia, USA, pp. 718–726.

Dickson, W. (1991) Image structure and model-based vision. PhD thesis, University of Oxford.

Du, F. and Brady M. (1994) A four-degree-of-freedom robot head for active vision, *International Journal of Pattern Recognition and Artificial Intelligence*, **8**.

Fikes, R. E. and Nilsson, N. J. (1971) Strips: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, **2**: 189–208.

Harmon, S. Y. (1987) The ground surveillance robot (gsr): an autonomous vehicle designed to transit unknown terrain, *IEEE Journal Robotics and Automation*, **RA-3**: 266–279.

Hoare, C. A. R. (1985) *Communication Sequential Processes* (London: Prentice Hall).

Hollier, R. H. (1987) *Automated Guided Vehicles* (London: IFS).

Hu, H. and Brady, M. (1992) Planning with uncertainty for a mobile robot, *Proceedings of the 2nd International Conferences on Automation, Robotics and Computer Vision*, Hyatt Regency, Singapore, 15–18 September, vol. 3, pp. RO-13·4·1–RO-13·4·5.

Hu, H. and Brady, J. M. (1994) A Bayesian approach to real-time obstacle avoidance for an intelligent mobile robot, *International Journal of Autonomous Robots*, **1**: 67–102.

Hu, H., Brady, J. M., Du, F. and Probert P. J. (1995) Distributed real-time control of a mobile robot, *The International Journal of Intelligent Automation and Soft Computing*, June: **1**, 63–83.

Hu, H., Brady, J. M. and Probert, P. J. (1993) Transputer architecture for seneor-guided control of mobile robots, *Proceedings of World Transputer Congress '93*, Aachen, Germany, September, 118–133.

Kenley, C. R. (1986) Influence diagram model with continuous variables. PhD thesis, Dept. of Engineering-Economic Systems, Standford University, CA.

Nicholson A. (1992) Monitoring discrete environments using dynamic belief network. PhD thesis, University of Oxford.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems* (Palo Alto, CA: Morgan Kaufmann).

Premi, S. and Besant, C. (1983) A review of various vehicle guidance techniques that can be used by mobile robots or agvs, *Proceedings 2nd International Conferences on Automated Guided Vehicle Systems*, Stuttgart, Germany, pp. 195–209.

Rao, B. S. Y., Durrant-Whyte, H. F. and Sheen, J. A. (1993). A fully decentralised multi-sensor system for tracking and surveillance, *International Journal Robotics Research*, **1**: 20–44.

Rosenschein, S. J. and Kaelbling, L. P. (1986). *The Synthesis of Digital Machines with Provable Epistemic Properties.* (San Mateo, CA: Morgan Kaufmann).

Tsumura T. (1986) Survey of automated guided vehicle in Japanese factory, *Proceedings IEEE International Conference Robotics and Automation*, San Francisco, CA, p. 1329.