A Hybrid Behavioural/Knowledge-based Approach to Robotic Assembly

CHRIS MALCOLM
Dept. of Artificial Intelligence
University of Edinburgh
5 Forrest Hill
Edinburgh EH1 2QL
Scotland, UK
Telephone: +44 (0)131-650 3085
E-mail (JANet): Chris.Malcolm@uk.ac.edinburgh

1 Introduction

This paper describes the main current research project in assembly robotics at the Department of Artificial Intelligence, Edinburgh. This is a relatively recent research programme. The idea to pursue this line of research was first conceived in the summer of 1986. An exploratory implementation was done as a student project during 1987 producing the system later referred to generically as the SOMASS system (from "soma assembly"), this first version being the SOMASS1 system. This project was jointly awarded the Xerox prize for best MSc project of the year [Malcolm 87]. On the basis of this success a successful application for a research grant was made to SERC/ACME in 1988, to extend the SOMASS system, and to try out the insights and techniques developed in that extension on a simple family of industrial assemblies. External funding for this project started in January 1989 (SERC/ACME grant GR/E/68075), to run for 3 years.

The purpose of the funded research project was to "address the restricted usefulness of current industrial assembly robots by investigating the planning, programming, and execution of sensor-based robot assembly tasks in terms of Behavioural Modules[1]". Our longer term research programme, of which this forms a part, is of wider scope. We derive some of our ideas from behaviour-based work in mobile robots [Brooks 86], and here at Edinburgh this assembly project is part of our *behaviour-based systems research group* which also works with mobile robots [Malcolm Smithers and Hallam 89].

Most research work in assembly robotics, including our own previous work in this Department, follows from a particular attitude to the problems of assembly which we now call the *classical approach* or the *Newtonian approach* to distinguish it from our new approach, which we call the *behaviour-based approach*. We think that some of the difficult problems of current methods of programming assembly robots are artefacts of the Newtonian approach. These problems become particularly evident when trying to generate assembly programs off-line from some general specification of the task, and when trying to use sensors to handle uncertainty of location and dimension. It was Brooks's perception of these problems when designing the TWAIN [Lozano-Perez and Brooks 85] high level assembly planner at MIT that led him to devise the behaviour-based robot research programme at MIT, and our perception of these problems when developing the RAPT feature-based assembly programming ([Ambler, Bellos, and Popplestone 75]) language to handle uncertainty which led to our behaviour-based assembly robotics research programme at Edinburgh.

2 The Principles of the Behaviour-based Approach

As we argued in [Malcolm Smithers and Hallam 89], there is sufficient agreement between the loose and argumentative bunch of researchers who claim to be following a behaviour-based approach

[1]original grant proposal abstract.

to identify a set of common principles and a paradigm. In fact, there is as yet insufficient theoretical justification for these principles, so strictly speaking they should not be called principles, but rather guidelines. This lack of principled support is a criticism often levelled against behaviour-based roboticists by classical roboticists. They do have principled support for their designs, in the physics of the local world of the robot (Newton), the control theory of their servo controlled actuators (Clerk Maxwell), and the principles of computer science (Turing) and knowledge-based artificial intelligence (Frege)[2]. In fact, this principled support of the classical approach is based on an important unstated presumption, a presumption which is denied by behaviour-based roboticists, the presumption that the formal scientific description of some piece of behaviour is naturally an important ingredient in implementing that behaviour. There is of course no doubt that you *can* implement a certain behaviour by using its formal description as an ingredient of the implementation. The question is, is it the *best* way?

It can be seen that biological systems do not use this approach. For example, we do not catch balls by plotting the ball's trajectory, curve fitting Newton's equations of motion, and so on. And as Churchland has argued, the methods by which biology solves this kind of multi-dimensional co-ordination problem is much more computationally efficient than doing it by solving the mathematical equations [Churchland 89].

I would argue that this lack of principled support for the behaviour-based approach is due to the fact that the science from which it would naturally draw this support does not yet exist. That science is the general science of complex systems. There was an attempt to construct this science in the 1940s and 50s, under the various headings of General Systems Theory or Cybernetics. It was supposed to encompass everything from information theory through computer science and control theory to biology, psychology, sociology, and ecology — anywhere where there were to be found complex enduring

[2] I cite the early pioneers to emphasise the (apparent) historical depth and solidity of the support for this approach.

systems whose stability and adaptiveness depended on the the use of information. Unfortunately none of these sciences were developed enough to produce the general laws and theorems required by General Systems Theory, and General Systems Theory itself didn't have enough theory to develop its own. So the attempt to start such a science failed, not because it was a foolish idea, but because it was premature.

However, without further discussion I will briefly present the general principles (or guidelines) of the behaviour-based approach. Detailed argument in support of them can be found in [Malcolm Smithers and Hallam 89].

Intelligence Emergent, not Implemented Intelligence is an emergent non-modular function of the interaction of the right ingredients.

Low-level Amalgamation of Sensing and Action Sensing and acting should be amalgamated at low levels in the system, and combined synergistically. Sensing is tied into action to produce higher level competences, task-achieving behaviours; action is tied into sensing to simplify the required perception.

Parallelism and Process Where possible parallel implementation should be favoured, with the items running as processes rather than as functions delegated from a central control. Where possible central control should be avoided, and where it exists, the control should preferentially be passed by signalling rather than delegation.

Ignorance As little use as possible should be made of prior knowledge, world models, symbolic structures, propositions, etc.. The behavioural modules should use as little knowledge as possible, and where it is used, it should preferentially be local. Every extra item of knowledge is something that the system could be wrong about; and everything which the system need not know is something that can change or be changed without upsetting the system, or requiring it to be changed.

Active Use of the World This is as a preference for finding things out from the world, instead of remembering them. This is sometimes called "using the world as its model". It reduces the memory load, and simplifies control. Instead of the delegation of control via procedural hierarchy and parameter passing typical of modern programming practice, there is a preference for the atomic processing routines to be activated and controlled by sensed environmental triggers. This simplifies the computational requirements (or saves wires), and makes the implementation more adaptable, by tying it to environmental cues and parameters. It also helps to keep the system *ignorant* (see above).

Minimalism Preference should be given to simple and computationally economical implementations.

Behavioural not Functional Modularity This slogan is due to Brooks [Brooks 86]. This may point to the most fundamental difference between the classical (information functional analytic) and new (behavioural synthetic) paradigms. The classical paradigm divides the task of implementing intelligent behaviour along functional lines, where the functions are those that need to be applied to the input data to transform it into the output data. This is a chain of processing, and as such is only as strong as its weakest link. This is contrasted with the decomposition of the required task into behaviours. Because these implement purposeful competences, and run in parallel, all that happens if one of them doesn't work is that that particular competence (and some others which subsume it) is lost; the rest of the system carries on doing something sensible — graceful degradation. Especially important to Brooks, who is deliberately emulating the progress of evolution in his methods of robot development, is that this architecture lends itself to incremental development.

Rolf Pfeifer of the University of Zurich recently independently derived a very similar set of general principles in the domain of Artificial Life, using different arguments [Pfeifer 96].

3 The SOMASS System

This system has played such a central part of the project, being for some time the main experimental crucible in which new ideas were tested, that a general description of the ideas behind it deserves a section on its own [Malcolm 87].

3.1 THE SOMA ASSEMBLY DOMAIN

When faced with the problem of constructing something so complicated as to be currently beyond the scope of our technology and understanding — such as constructing a fully automated task-level sensor-based robotic assembly system — there are two possible general lines of approach. One is to break the problem down into subproblems, and so on, until problems of solvable scope have been identified. Once enough of these have been found, the problem of integrating them can be tackled. The other general approach is to simplify the domain of the problem (e.g. try two dimensions instead of three) until a sufficiently simple version has been arrived at that a complete system has come within achievable scope. Once this has been done, one tries to increment the domain complexity, extending the system to cope, until one either reaches the original problem domain, or has developed sufficient understanding to tackle it with good expectations of success. These two approaches are analogous to the top-down and bottom-up approaches to software design and, like them, are the polar extremes of a spectrum of mixed strategies[3].

An important component of the behaviour-based approach to robotic assembly research is the insistence on testing *complete* systems, the second of the above two research strategies. The case for this tactic has been argued extensively elsewhere, e.g. [Moravec 84,

[3]Yourdon and Constantine claim that an analysis of the best large software systems shows that the optimum approach is a combined method, where bottom-up and top-down design are employed simultaneously (i.e. cyclically), both approaches being guided by the iterative refinement of the definition of the interface between them [Yourdon and Constantine 79]. In the case of behaviour-based assembly this interface is the virtual assembly machine defined by the behavioural operators.

Brooks 86, Smithers and Malcolm 89]. Since we are employing this complete-system research tactic, we needed to identify a suitably simplified experimental domain, ideally as simple as possible without losing the essential characteristics of the problem, and having nicely graded increments of complexity. We settled on what we call the *soma world*. It is, we believe, the simplest possible experimental domain in which to explore the problems of robotic assembly. This permitted us to implement a fully integrated and completely automated task-level assembly system which plans its own assemblies from the most general starting point. Although very simple, it is not so simple as to lose the important characteristics of the assembly task. For example, the MIT "blocks world" of rectangular bricks is too simple, since the parts simply abut one another, lacking the shape-dependent part-fitting which we consider to be an essential characteristic of assembly (and which is preserved in the soma world).

The soma world is well known to mathematicians as a puzzle domain characterised by a fiercely combinatorially explosive search space [Gardner 61]. In its original form (the soma4 world) it consists of all the different irregular[4] shapes which can be made by joining at most *four* cubes together, face to face.

There are only seven different shapes in the soma4 world, comprising in total 27 cubies[5]. It so happens that this part set can be made into a 3x3x3 cube (the original discovery of Piet Hin, the Danish mathematician who invented and named the soma worlds) in 240 different ways (proved in [Berlekamp, Conway and Guy 82]), each one of these different ways constituting a distinctly different assembly problem for the robot system. In fact, since the figure of 240 excludes rotations, but the direction of gravity is an important component of the assembly problem, the number of distinctly different assembly problems in building this cube shape alone is 1440 (6x240). Thus, just the simple problem of building a cube from the soma4 set of parts has considerable scope for exercising an assembly

[4] i.e. containing a concavity.

[5] "cubies" are the small cubes from which the soma parts are constructed.

system.

The assembly task was seen as an ordered set of /em part phases. Each part phase consisted of all the operations specific to a part: acquiring it; if necessary orienting and regrasping it; and fitting it into the assembly. Planning the assembly consisted of deciding in what order to put the part phases, and then detailing the operations of each part phase. The difficulty (as in all assembly planning) was that the ordering of the part phases depended in part on the details of the part phase operations; and the details of the part-phase operations depended in part on the ordering. Thus there is no way of solving this without a general backtracking search.

The system was divided into two main parts. The first part, the assembly planner, solved the above problem, and produced an assembly plan. This was essentially a program to be interpreted by the second part of the system, the assembly agent. The planner was a classical symbolic planner written in PROLOG. The agent was a behaviour-based plan interpreter written in VAL2. The novelty of the system was its hybrid architecture: that a behaviour-based interpreter was used to execute the plans devised by a symbolic planner. The behaviour-based interpreter permitted some problems (mainly coping with uncertainty, with and without sensors) which were usually considered to be the domain of the planner to be solved in the agent. Classical assembly planners have been designed top-down to produce plans (programs) in the language of the robot controller (such as VAL2). Here the recipe of Yourdon & Constantine was followed. The interface between the agent and the planner is the plan, and the terms in which the plan is to be constructed are set by the capabilities of the agent. The behaviour-based plan execution was implemented bottom up to have as much useful capability as possible, where a useful capability was one which looked as though it would simplify the design of the planner. Similarly, the planner was designed top down towards this interface, clarifying the nature of useful capabilities at which the behaviour-based system should aim. This design method greatly reduced the complexity of the planner, increasing the complexity of the agent much less than this reduction, and thus reduced the overall system complexity. It also produced a robust system, capable of executing novel plans reliably despite the deliberate presence of considerable uncertainty.

This was the scope of the original (SOMASS1) system. The Behavioural Modules[6] consisted essentially of generic part *acquire*, *regrasp*, and *fit*. Written in VAL2 (the control language of the ADEPT robot) as subroutines, these in effect constituted an interpreter of a higher level language for soma assembly work. The output of the assembly planner was a sequence of parameterised calls to these subroutines, which would accomplish the assembly. Of course, since (in the SOMASS1 system) there were no sensors this interpretation was completely determined, i.e. the planner might just as well have *compiled* the complete assembly program.[7] The important difference between *interpretation* and *compilation* occurs in systems with sensors, and is the way in which sensor usage is concealed from the planner[8].

Of course, it is also possible to make other shapes with these soma4 parts. The extensive soma4 literature contains dozens of examples of these. These provide some extra complications for the

[6] A Behavioural Module is the implementation unit by means of which a particular behaviour is implemented. In this case they are the subroutines in the VAL2 programming language of the Adept robot, plus the various electrical and mechanical accessories which provide the effective connection with the physical world.

[7] Although the motions of the *robot* are completely determined, the motions of the parts are not — use is made of strategic "funnels" of constrained motions which bring parts from a state of considerable uncertainty to a state of known position and orientation.

[8] We often speak of the "sensorless" assembly robot. This is strictly speaking incorrect, since they do have joint sensors. These joint sensors are, however, not fully accessible to the programmer, since they are encapsulated in the position feedback control loops of the joints, which form the basis of the position control language in which the robot is programmed. The Behaviour-based approach to robot programming intends to do the same sort of thing with the other sensors: to hide them within task-achieving modules which host a higher level language in which the sensors themselves are not visible. In fact, although they were not implemented with this in mind, the end-effector position control systems of today's assembly robots can be seen as good examples of behaviour-based systems, including the fact that they host a higher-level language within whose functions they are concealed.

planner and for the robot, such as overhangs. The assembly domain consisting of all the shapes which can be made from one set of soma4 parts we refer to as the "single soma4 world", usually shortened to the "soma4 world". It so happens that *every* assembly in this world can be constructed using one part fitting strategy, the q*put-brick-in-corner* strategy, i.e., no rotations are required in part fitting, and there are always three degrees of translational freedom.

3.2 CYCLIC ADJACENCY AND SUBASSEMBLIES

The multiple soma4 world (where more than one instance of a part shape can exist) contains one assembly problem where this part-fitting strategy will not work. This is the simplest possible example of a general assembly problem we call *cyclic adjacency*, i.e., where two or more parts fit together in such a way that one set of mating faces shows one part in *front* of another, whereas another set of mating faces shows it to be *behind* the other.

A set of parts in which such a cyclic adjacency occurs constitutes a subassembly which requires a special assembly strategy. Each cyclic adjacency in a particular subassembly can remove one degree of translational freedom in the fitting trajectory. Thus one cyclic adjacency creates a subassembly where there are only two degrees of translational freedom in the final part-fitting trajectory, i.e., a plane. Two orthogonal cyclic adjacencies, as in the square-peg-in-hole subassembly, leave only one degree of translational freedom, i.e., a line.

- In the single soma4 world all parts can be fitted with *three* degrees of translational freedom.
- In the single soma5 world some subassemblies have only *two* degrees of translational freedom.
- In the single soma6 world some subassemblies have only *one* degree of translational freedom.
- In the single soma7 world there are some dispositions of parts in space which cannot possibly be assembled, i.e., where there

are *no* degrees of translational freedom.

The business of identifying subassemblies is a well known difficult problem in assembly planning. In the soma world these cyclic adjacencies give us a nice computational handle on the problem. One of the things we would like to explore is whether they can also give us a computational handle on the different part-fitting strategies required.

As the order of the soma worlds increases, other problems also become more complicated. For example, only in the soma4 world has every single part at least two gravitationally stable and rotationally distinct ways of sitting with some faces in contact with the surface of a table. When this is not the case, then the planning and execution of a regrasp becomes more complicated. Thus, as the order of the soma worlds is increased, the assembly problems increase in variety and complexity. It is possible to find analogues of the well-known part-fitting problems which have exercised robot assembly researchers, such as peg-in-hole, tongue-in-slot, two-pegs-in-holes (plug-in-socket), etc.. This kind of incremental complexity is just what the ideal experimental domain for our research strategy requires.

3.3 THE ASSEMBLY PLANNER

An important aim in developing the SOMASS assembly planner was to keep it as ignorant as possible of the details of the assembly, on the general grounds that the less it needed to know, the more general would the resultant plan be. Of course it can only be kept ignorant of those things which the agent is capable of deciding for itself. Thus this aim could be rephrased as making the agent as capable of handling as much detail as possible.

In consequence the planner knows nothing about the layout of the work cell, the size of the parts, or the dimensions of the fingers. This information is supplied at run time. Thus different sized parts can be used, in a different layout, without requiring a new plan. This information can be supplied by sensors at run-time, without the planner caring about it.

Many proposed assembly planning systems have taken it for granted that the output of an assembly planner — the assembly plan — should of course consist of an assembly program in the particular programming language of the robot concerned. What all these programming languages have in common is that the assembly is in effect described in terms of motions of the robot: the assembly is performed as a side-effect of these motions. An inevitable implication of this decision is that the planner must therefore be concerned with anything which might affect the motions of the robot, e.g. sensory input. In other words, the planner must concern itself with uncertainty, the uses of sensors, and strategies to reduce uncertainty. This makes the planner formidably complicated. To avoid this we decided that the plan would not be in terms of *robot* motions, but in terms of *part* motions. It would be the job of the run-time system to interpret these required part motions into the robot motions (and maybe sensings) which would accomplish these part motions. So in this case the planner is *not* concerned with uncertainty or the uses of sensors.

The planner is hierarchical, in the sense that it forms a complete plan at a certain stage of abstraction, before adding more constraints and forming a complete plan at a lower level of detail. This is a well-established method of improving search times in combinatorially explosive search spaces. Adding the constraints of a further level of detail frequently turns out to make the assembly impossible, in which case the planner backtracks to find an alternative higher-level plan. Where a great deal of wasteful backtracking would be required to escape from some problem, because the conditions causing failure had been set up at a much higher level, the planner uses failure-directed backtracking to avoid this[9].

[9] What distinguishes this failure-directed backtracking from the more common dependency-directed backtracking is the point at which the backtracking information becomes available. In dependency-directed backtracking the dependencies which identify the backtracking point are known at the time of making the original choice. In failure-directed backtracking the dependencies and hence the backtracking point cannot be known until an analysis of the reasons for failure has been made.

Added to the front of the assembly planner proper is a soma puzzle solver, which searches for a disposition of parts within the boundary of a given shape, and constitutes the first level of the planning hierarchy. This is not of course part of the general problem of planning assembly for robots: one does not give a set of parts to an assembly system (or person) and require that it guess how to construct the assembly! Such a front end is, however, very useful in a research prototype. It greatly simplifies using the system, since one does not need to describe how the parts are disposed in the assembly — a tedious and error-prone task — but merely the outline of the shape they are intended to construct. From this information, which is much more simply described than the disposition of the parts in the assembly, the system finds a disposition of the parts, i.e. it identifies and describes a candidate for assembly planning. More important, it permits the system to be a tireless inventor of its own test data, since the planner will automatically crank out one assembly after another, until (for example) all 1440 different methods of cube assembly have been tried (in fact, the poor robot, not being very dexterous, is currently only able to assemble about half of the 1440 theoretically possible cube assemblies[10]). For example, as part of the reliability trials of the SOMASS1.1 system ([Malcolm and Smithers 88] we used 39 different ways of assembling the cube, which were the first 39 robot-possible assemblies out of the first 85 candidate dispositions. It would probably take a specially ingenious and methodical person weeks (at least) of very tedious labour to prepare that amount of test data[11]. This took only 7.5 hours in 1987 with the SOMASS1.0 system on a 4 Mbyte Sun3/160. The later SOMASS1.3b system on a 12Mbyte Sun3/180 was an order of magnitude faster.

The output of each stage in a hierarchical planner corresponds to what would be a complete executable plan, if only the requisite operators to accomplish that plan existed. Thus the descent of the

---

[10] A person could do all 1440 assemblies.

[11] Many people would simply find the task impossible. The very gifted mathematicians and puzzle-solvers Conway and Guy might be able to do it in less than a week [Berlekamp, Conway and Guy 82].

hierarchy can be seen as a recursive elaboration of plan detail until the level of operators executable by the robot system is reached. Note that it is a fundamental principle of our behaviour-based assembly planners that we do *not* descend to the level of detail of *robot* motions. The planner stops at the level of motions of the *parts*. It is the job of the behaviour-based plan execution system to *interpret* these part motions into robot motions. The stages of the hierarchy, and the virtual operators which correspond to them, are listed below.

1. Find a disposition of the parts within the assembly (usually many solutions — in the cube 1440). This level corresponds to a "materialisation" operator, i.e., this would constitute an assembly plan if it were possible to "materialise" each part directly into its final position (and the parts were perfect in form). This will backtrack through all possible dispositions.

2. Find a gravitationally stable ordering of the above in which each part can move into its final position along an unobstructed vertically downwards trajectory (always many solutions — in the cube about 70 per disposition). This level corresponds to a telekinetic operator, i.e., this would constitute an assembly plan were it possible to command the parts to fly through space (and the parts were perfect in form). This backtracks through all possible stable orderings.

3. Find a way of gripping each part as it is put into the assembly (mostly fails). This level corresponds to an operator which causes the part to materialise in the required attitude in the robot's gripper.

   Because a failure here is always due to some other previously placed part getting in the way, this part of the search is pruned by identifying the obstructive part and using dependency-directed backtracking to leap directly to the next stable ordering in which that part has been re-ordered in the assembly sequence, i.e. the planner attempts to re-order the obstruction out of the way.

4. Find either a get-grasp which matches a put-grasp, or else a regrasping strategy (sometimes fails). This corresponds to the

final assembly plan in the sense that all of the operators used here (pick up, regrasp, etc.) correspond to Behavioural Modules in the assembly system. Planning proper is now finished, in the sense that neither of the two subsequent stages can fail, i.e., the searching which is characteristic of planning has concluded. Execution of this plan would always fail, since it is still based on the assumption that the parts are perfect in form and can be fitted into the assembly using ideal fitting trajectories with zero clearance between the parts.

5. Calculate the offsets necessary between the parts to allow spaces between all the horizontally adjacent faces (this always succeeds). This is known as *spacing out*, and is an important part of a tacit strategy for handling uncertainty[12]. These spaces accommodate the errors of form of the parts, plus the cumulative errors of location, which together comprise the presentation uncertainty at this point of fitting the part into the assembly. The errors are "trapped" (hygienically isolated) in these spaces during the assembly process. The robot concludes the assembly process by nudging the assembly together so as to close up the spaces[13].

6. Translate this plan into VAL2 code for the robot system (this always succeeds). This is a simple context-free syntactic transformation. The code consists largely of parameterised sub-

[12] i.e., "unknown" to the planner because built into the design of the entire system. In describing kinds of representation in autonomous systems we use *explicit* to refer to directly encoded representations, *implicit* for those which the system can deduce as required (from the explicit representations), and *tacit* to refer to those embedded in the design of the system, and which therefore can't be reasoned about by the system [Malcolm and Smithers 89].

[13] Note that while this spacing-out calculation always succeeds, it involves a search process which just happens to be both independent of the other constraints and guaranteed to succeed. This is an accident of the single soma4 world, and does not hold in either the multiple soma4 world or the higher orders (soma5, soma6, etc.). Thus in the soma5 world, this stage will still be part of the search in plan-space. The reason for this is that the existence of cyclic adjacencies forbids the assembly strategy of which this "spacing-out" is a part (see the discussion in section 3.1).

> routine calls to Behavioural Modules[14].
>
> Making assembly plans for different robots is therefore a question of devising an appropriate set of behavioural modules for the robot, and recoding this section of the planner. The rest of the planner remains unchanged, much in the way that modern computer programming language compilers change only the code-generation back-end to adapt to different computer hardware[15].

The system so far described (the SOMASS1 system) used no sensors, and coped with the uncertainties of part location, grasping, and so on, by various constrained-motion strategies based on pushing and sliding. However, one of the major problems motivating this research was the problem of integrating the use of sensors into higher-level assembly planning systems, and one of the claims made of this hybrid architecture was that it would be easier to integrate sensing. For example, we had arranged that the planner did not need to know the locations of the parts. This information was provided to the on-line plan execution assembly agent by reading a file. The file contained approximate part locations. Given approximate part locations the part acquisition behaviour was capable of acquiring the part with sufficient precision to accomplish the assembly successfully. The important point was that since the planner did not need to know the part locations, the part locations could change without requiring a change in the plan (only the plan interpretation changed). Thus a sensor system (e.g. vision) could be used to discover the locations of the parts, and this system could be built into the agent relatively simply without requiring any change in the planner.

That was the claim. In addition to this general claim, we also expected that constructing a part-locating vision system in a behaviour-based manner would be likely to yield a different kind of system, with different properties, than one built according to the classical

[14] This strictly serial delegation of control is only possible in the sensorless SOMASS1.0 system. The SOMASS2 system involves parallel distributed behaviours, with control mediation via the environment.

[15] And just as in the case of compilers, it is possible (without prejudice to the useful generality of this approach) to devise a bizarre hardware which would necessitate changes further up the system.

(i.e. knowledge-based) paradigm.

The next step in the research programme was to test these two claims in practice.

3.4 UNCALIBRATED BEHAVIOUR-BASED VISION

Consider the problem of a robot, a part which it is to pick up, and two overhead cameras which are to be used to help it do this, because the position of the part is known approximately, but not well enough to grasp it successfully. The robot is a standard industrial assembly robot with a programming language which permits it to be directed to move to locations in space specified in Cartesian co-ordinates, in some absolute reference frame.

There are two distinct ways of approaching this problem. The first we call the classical or Newtonian approach[16]. To begin with we have a robot which can move its gripper to a position and orientation specified in some frame of reference (usually by default centred on the base of the robot). Thus half the problem is solved. What we need to know for a complete solution is the position of the part in the same frame of reference. We can use the two cameras to triangulate the part. Taking for granted that we have suitable vision software for finding the position of the part image in the camera image planes, we can use these positions, plus knowledge of the exact positions and orientations of the cameras, their lens parameters, and so on, to compute the position of the part in the world reference frame. The necessary calculations are complex but well known and many such systems have been successfully implemented.

Now let us consider the knowledge such a system requires. The positions and parameters of the cameras must be accurately known. In practice this is done by fixing the cameras very firmly to some especially stable support, carefully taking the required measurements,

[16]We call this the Newtonian approach because Newton was probably the first scientist who could have solved this problem, since his mechanics and optics are crucial parts of it, and to someone with an education in Newtonian physics, i.e. the average schoolkid studying physics, or the average engineer, this is obviously the right way to do it.

and then making sure that nothing moves. From this knowledge, plus the lens parameters, and analysis of the images, we can compute the location of the part. It is sometimes forgotten that there is also knowledge involved in being able to send the robot arm to the required position, because this knowledge is built in to the robot programming system by the robot manufacturer. This knowledge is the inverse kinematic model, based on the lengths and articulations of the joints of the robot. From these are computed the necessary angles of the robot's joints for the gripper of the robot to be at the required location in space.

If any of this knowledge is wrong, then the robot may fail to pick up the part. Cameras may move slightly, through vibration, or being knocked. Any sizeable lengths of metal involved (such as the joints of the robot, or scaffolding supporting the cameras) will be liable to change their length with temperature. This in fact is a real problem with assembly robots — it is sometimes found that programs which work well when the factory is warm do not work so well when the factory is cold, e.g. on Monday morning in the winter. So, because it is so important that this built-in knowledge is correct, in practice some engineering expense is devoted to ensuring that it is likely to be correct. The robot will be made of very stiff substantial links. The cameras will be mounted very strongly and rigidly. And despite all of these precautions, it will probably be necessary every now and then to recalibrate the system anyway.

This critical dependence on knowledge is why this approach is also sometimes called the knowledge-based approach. The general presumption of the knowledge-based approach is that intelligent behaviour is knowledgeable behaviour, so the problem of making something behave "intelligently" becomes the problem of deciding what it needs to know in order to behave like that, and then deciding how to encode, access, and utilise that knowledge. The knowledge-based presumption leads naturally to the implementation of classical or Newtonian robot systems, as described above.

Now let us consider the Behaviour-based approach. It is part of the philosophy of behaviour-based systems to avoid (where possible) the use of explicit models, using direct perception of the world instead. In other words, we should try to use cameras of unknown location and lens parameters, to calibrate them no more than is actually necessary, and to do this while using them rather than beforehand.

In general it is possible to use motion of the camera and assembly robot to calibrate the system from a position of complete ignorance. Establishing a *complete and accurate* model would be difficult and computationally expensive, but it is unnecessary: we perform no more calibration than is needed for the task in hand — guiding the motion of the robot to the precision required.

The robot first moves to a position above the nominal location of the required part. Let us say that the error in position is a few cm., whereas in order to grasp it properly we need an error of no more than 1mm. The cameras take a picture. The robot closes its fingers. The cameras take another picture, and subtract it from the first, thus identifying — for each camera — the position of the gripper in the image plane (only the fingers have moved between the two pictures, and thus are the only feature left in the subtracted image). For each camera the vision system then projects a rectangle beneath the fingers in the image plane, within which the part is to be found. Having found the part, the vision system then locates the top visible edge of the top cubie of the part: the cubie to be gripped[17]. The system makes an estimate of how each image plane maps locally onto the robot joint angles, and commands the robot to move closer to the part[18]. The difference between the commanded motion and the observed motion permits the system to improve its guess concerning this local mapping. The initial estimate is based on what worked last time. If it is completely wrong, i.e., the robot moves in quite the wrong direction, then this false move simply leads to a better move next time. Each camera is trying to reduce the error between the observed relation between the gripper and part,

[17] The parts in question are soma parts, made from a few cubes glued together. These individual "atomic" cubes are known as *cubies*.

[18] The mapping is local because perspective varies it through the image plane, and it also changes as the robot changes in kinematic attitude.

and the desired relation. While each camera will have ambiguity of robot position along its line of sight, provided both cameras have sufficiently different view angles, when both images look right, the gripper will be in the right position relative to the part. Note that it is not necessary for the local mapping between image parameters and joint angles to be correct: it merely has to be more right than wrong, since then by iteration any errors can be minimised to any required degree.

In this sort of way the robot is brought into an appropriate position from which to grasp the part. The only essential initial requirement is that the required part is in view from some direction vaguely middling in the required octant of the part's nominal frame of reference. The distance, exact angle, length of lens: all these are irrelevant, and the system never bothers to compute them. In practice we found that 5-7 iterations sufficed to bring the gripper from an error of a few cm to an error of 1mm. In order to achieve this sort of error we required that the gripper and the part filled about half the camera image, i.e. so that a few pixels in the image corresponded to 1mm at the distance of the robot and part.

The system as described so far will guide the robot gripper into a gripping position, given that the cameras already have the part and the gripper in view. The next problem is how we get the cameras into this position, since the correct position for one part grasp will not do for another: the cameras must be pointed at each part in turn. To accomplish this the cameras need to be mounted on a motor-driven pan/tilt mount, or held by another robot.

For a classical system this is no problem at all: since everything is already calibrated into a global reference frame, it is trivial to calculate where the camera needs to be pointed, and to command the camera-carrying robot or pan/tilt mount to move appropriately. This is not an easy problem for a behaviour-based system, however, because we are trying to avoid using the kind of knowledge which solves the problem for the classical system.

Why not have the camera system search the area and recognise the wanted part? That would be one way of solving the problem, but at the cost of making the vision system a lot more complex — for example, this would require part recognition to be added. The challenge is to solve a problem which classical robot systems can already solve *without* making the system more complex.

The problem was solved by equipping the camera system with a motion-following behaviour, which simply tracked a moving object, trying to keep it in the centre of the view. A synchronisation position was used, a position which the robot went to and which the cameras knew how to point at. A convenient position was just above the assembly, a natural position for the robot when it had just finished fitting a part into the assembly. So after fitting a part the robot paused above the assembly. The cameras were pointing at this position waiting for something that looked like a stationary robot. The robot was equipped with a brightly coloured sign to make this recognition easy. Once the cameras had spotted the robot they switched to tracking behaviour, and followed it as it moved to (approximately) above the next part to be picked up. In a sense the robot pointed out to the vision system the part it wished the vision system to look at, and indicated that it was doing so by pausing and closing its fingers. This triggered the vision system into identifying the part, and from this point on the vision-servoed motion took over and the part was acquired as before.

Note that the system actively discovered its parameters from the world (in this case by following the moving gripper) rather than being passed parameters from an internal model.

The benefit of this approach is its robustness: it is insensitive to all those things it doesn't need to know. For example, a camera bracket can be bent, a different lens installed, different parts of different sizes substituted, the parts be used by the robot in a different order, the robot arm itself can be bent, etc. We have run the system with two different models and shapes of robot arm (Adept and RTX), without having to tell the system about the change of arm. None of this affects the working of the system. It is protected by its ignorance, and the fact that what it does need to know, it finds out actively from the world.

This system nicely illustrates the several characteristics of behaviour-based systems. It also shows clearly how the behaviour-based philosophy, as applied to sensing systems, produces quite a different kind of sensing system than would the classical approach, which regards the use of formal mathematical models in the *implementation* as the *sine qua non* of a principled scientific approach[19].

This work is described in more detail in [Chong and Conkie 90, Chong and Conkie 92, Conkie and Chong 90].

3.5 SHAPE MATCHING

Chongstitvatana's uncalibrated vision described in the last section showed how vision could be used in a behaviour-based way, using minimal prior knowledge, to guide the gripper to a part. We exploited the simplicity of the soma parts to avoid the shape matching problem. In a practical application it would be necessary to do some kind of recognition, either to distinguish one part from another, or to recognise where the part was to be grasped.

To tackle this problem Harris developed the shape attractor. This operates on two sets of straight lines, a target and a source. The goal is to translate and rotate the source set of lines so that it best matches, by superimposition, the target set of lines. Although these sets of lines do not have to be connected in the way that the image boundary or edge map of an object would be, since this is the intended application I shall from now on refer to the sets of lines as shapes. We wanted to avoid having to have perfect models, so we wanted matching to be good in the presence of imperfections, missing bits, and so on, and we wanted to avoid the well known computational expense of doing an exhaustive search for the best match. Harris's idea was to attract the shapes together iteratively using an energy function. Every line in the source shape would be attracted to every line in the target, more strongly the closer it got, and more strongly if it matched in length and orientation. Such an energy function requires careful tuning if it is to behave well.

---

[19] We have no objection to the use of mathematical models for analysis; it is their use as ingredients in implementations of which we are deeply suspicious.

Clearly such a method would be liable to get stuck in the wrong local maximum when there was a rough symmetry to the shape, but if it was fast enough, and this was a problem, then we could try ways of getting past it, such as rotating the source and re-attracting. This is an example of the behaviour-based approach: don't try and solve a problem perfectly, try and solve it approximately as quickly and efficiently as possible. Then if the solution is not good enough, improve it until it is. In practice perfect solutions are not often required, and are often not worth the cost even when desirable.

We found that it was possible to build a shape attractor that was surprisingly good. It did get stuck on false matches in predictable ways on awkward shapes, but it was very fast, and coped unexpectedly well with imperfections and missing bits [Harris 90]. Apart from trying it out in images, we used it, with Chongstitvatana's iterative gripper approach method, to get a robot to solve the child's posting box problem. This is a toy which involves a set of prisms with simple cross sections, such as circle, square, triangle, star, rectangle, which have to bu put into a box by selecting the appropriate shaped hole. In this case the shape attractor was used in two different ways. The robot first picked up a shape, and presented it to the camera system beside the box. Simple boundary tracking image software identified the outline of the part in the robot gripper, and the outlines of the holes in the lid of the box. The shape attractor matched the shape by attracting it to each hole in turn, and taking the one with the closest fit to be the right one. Having identified the target hole, the shape attractor was then used to guide the shape in the gripper into correspondence with the hole, using the same iterative approximation method as in the previous section, using linear approximations of the relations between joint movements and displacements in the image. Because of the way the important parameters are discovered by active exploration during the activity, the system which used robot A to put the shapes into one child's posting box toy work equally well, without needing to change the software, with another robot B, using a different version of the posting box toy, with differently sized and shaped parts.

Building on this success, Balch showed that the shape attractor could also be used to correct small rotational misalignments about the other two axes, just using the two dimensional image, by using the shape attractor to pull not a set of lines, but an underlying three dimensional model, from which the lines were derived by perspective projection and hidden line removal [Balch 92a].

We do not propose any of this sensory work as paradigmatic, but simply as showing that there are ways to solve these sensori-motor guidance problems in a behaviour-based way, avoiding the architectures and knowledge requirements of the classical approach, and gaining extra robustness and generality thereby.

3.6 EXPLORATORY USE OF FORCE SENSING

Having successfully established that in one particular sensory problem we could show the virtues of the behaviour-based approach it was time to try another one. We chose force sensing because it is as different as possible from vision. We used a standard six-axis (three force and three torque) force sensor which could provide readings in any frame of the user's choice. We mounted it beneath a small assembly work-table, so that it could measure the forces applied to the assembly, while being relatively immune to motor and motion noise from the robot. The assembly task we chose was the assembly of a solenoid-driven stackable double-pole double-throw switch. This consisted of a plastic housing base into which some awkwardly shaped bent pieces of metal (the contacts) had to be fitted. Then the top had to located and snap-fitted on.

The robot chosen for this task, an RTX, was quite incapable of the accuracy needed, and only just had the necessary resolution. In other words, the task was impossible without sensing, and only possible if the sensing were capable of accurately utilising the approx. 1mm resolution of the robot's joint motors.

The strategy employed was to assume as little as possible about the dimensions and location of the parts, and to discover as much as possible by using exploratory guarded motions[20]. A few basic

[20] A guarded motion is a robot motion which can be terminated early on detection

guarded motions were elaborated into more complex behaviours, such as peg-in-hole, follow-surface, find- edge, etc.. We were surprised at how few, how general, and how neatly composable, these basic behaviours turned out to be. For example, many exploratory tasks turned out to be variants of peg-in-hole, such as screwing a nut onto a bolt, which is in effect a helical peg-in-hole.

Once a good well-generalised basic behavioural repertoire had been established, developing a program for a particular assembly became largely a question of selecting exploratory strategies which would discover the needful locations, dimensions, and so on, and accomplishing these by appropriate combinations and parameterisations of the basic behaviours.

This development was successful in the sense that the robot accomplished the assembly reliably, and the architecture and assembly program development strategy turned out to be sufficiently general that it inspired a number of further sensory developments. The details of the original development can be found in [Balch 92]. It was later adapted for use with PVDF touch sensors developed here [Kim 92], later extended for preliminary experiments in sensor fusion [Kim 96]. Since sensor fusion is classically accomplished by integrating the sensor data at the level of the geometric world model, and our behaviour based systems avoid the use of these models, handling sensor fusion tasks is clearly an important test for the behaviour-based approach. This is an area in which we propose to do more research.

4 Are we using the Right Kind of Robots?

The force sensing experiments described above were not as satisfactory as the vision experiments. The behaviour-based vision system succeeded very well in meeting or even exceeding our initial hopeful expectations, and is a model illustration of the behaviour-based principles. The force-sensing and sensor-fusion experiments worked, and did illustrate the principles, but not as clearly. Where the vision

of a specific sensory signal.

system naturally fell into an excellent illustration, the force sensing systems sometimes seemed artificially contrived to meet the less clear illustrations claimed of them.

This kind of difference between vision sensing and force sensing has been noticed before. For example, Yin's assembly robot vision work [Yin et al 84] was an excellent illustration of RAPT-based vision, whereas the attempt to integrate force sensing was abandoned due to ever-increasing theoretical and practical complexity. This is at least partly due to the fact that, considered as human senses, vision is integrated much better into our conscious awareness, our linguistic abilities, and consequently our formal scientific models. While we are very adept indeed at constructing sensori-motor force-based strategies for doing things, such as for throwing things, jumping, and riding bicycles, we find if very much more difficult to be aware of what we are doing, or how we do it, and consequently we find it much harder to talk about force-based strategies, and consequently we have not developed mathematical models to the same extent as for vision. It may have something to do with the fact that we more easily share what we can see than what we can feel. Whatever the reason, as human beings we find force sensing, despite our practical expertise, much more difficult to understand and communicate.

It is nevertheless quite clear that when we — people as opposed to robots — do assembly work, we make very considerable use of force sensing. Indeed, because our sense of dead reckoning of the position of our hands is so poor compared to typical assembly robots, and because our arms and fingers are hopelessly compliant compared to robots, we make much more use of force sensing than do typical industrial assembly robots. In fact, unlike robots, we can easily learn to do assemblies without sight, but we could not possibly do assemblies without touch and force sensing. Could it then possibly be the case that our easier formal grasp of the visual world had led our scientists and engineers to construct the wrong kind of robot? Could it be that some of the awkward problems of assembly robotic research are artefacts of trying to do it the wrong way, where the right way is the way biological evolution has equipped us to do it, despite the fact that we find it difficult to understand?

It was these kind of considerations which led Graham Deacon to explore the possibilities of an assembly robot which did not use position controlled joints at all, but force-controlled joints.

5 Deacon's Force-Controlled Robot

Many researchers have taken a standard assembly robot with position controlled joint servos, and added a six-axis force sensor in the wrist. This allows the robot to use force-feedback to control its motion, but the force feedback is in terms of the forces experienced in the wrist frame, and consequently get back to the joints via the inverse kinematics of the arm. This inescapably slows the response of the system and also introduces stability problems when in contact. Others have tried to separate the position controlled axes of motion from the force controlled axes by lining up joints with the force controlled axes, such as in the SCARA configuration of robot. These are all ways of trying to integrate force control into a fundamentally position controlled device. Deacon's approach was more radical. He argued that since the problems of uncertainty which plague assembly robotics are due to the fact that the fundamental robot operation is to move to a precise position, *if* it were possible to avoid position control entirely, then the uncertainty problems would disappear.

For example, if it was possible to devise an assembly robot which was controlled purely in terms of force, and it was possible to program such a robot to perform (say) a peg-in-hole operation, then this program would be naturally general with respect to position, size, and orientation, of the peg-in-hole. If the program worked with a small peg-in-hole over here, then the same program would naturally work with a big one over there. The programs would exploit the natural constraints on motion which the parts exerted on each other. The program would in effect consist of "feeling" the peg into the hole, much as we would if we could see nothing.

Was it possible to build such a robot and program it in such a way? This was the topic of Deacon's PhD thesis project, partly funded by EPSRC, partly by BAE (British Aerospace). The experimental investigation used a planar robot with three degrees of freedom, x, y, and rotation about z. The answer was yes, it was possible. Programming the robot to do the task, however, was a highly skilled esoteric task. This was clearly not going to be a useful assembly robot for the factory floor unless a much simpler way of programming it could be found.

Programs for this robot consist of networks of force-situations, connected in a graph to other force-situations, where the arcs of the graph consist of applying a certain a certain kind of force strategy. For example, if the robot was enclosed in a square box, and it could feel around the walls and corners of the box, each corner would comprise a force-situation, and each side would comprise a force-situation. When in a side-situation it can move to left or right by applying a suitable force, whereupon it will slide along until it hits a corner. The hit, a change of sensed force, signals a new force-situation, in this case a corner.

Although we find force sensing tasks so hard to understand consciously and formally, we are nevertheless very skilled at learning how to do them. Would it be possible somehow to harness this non-conscious human skill to program the robot? For example, if the robot were equipped with suitable controls so that people found it easy to drive it around, then they would quickly and naturally become skilled at getting it to do tasks under manual control. If it proved possible somehow to capture the important control features of the human manual operation of the robot, then the robot system could use these to program itself to perform the task on its on.

In order to investigate this method of programming a three-axis robot motor control joystick has been built, which has built-in motors to provide force feedback from the robot's six-axis force sensor. Thus the human operator applies forces to the joystick to produce corresponding forces at the end-effector of the robot, and feels the contacts made by it. In a year or so we should have discovered whether the robot can be programmed in this way. In fact there is little doubt that one way or another it can be done. The crucial questions will be how difficult it is to do, and how simple the user interface can be made.

The work so far has concentrated on motion in contact, because this is the only kind of motion to which this kind of control applies. To be a complete assembly robot system there will also have to be a way of moving the robot through free space, in order to fetch parts, and carry them into the part-mating motion-in-contact situations. This will have to be carefully done in order not to introduce the dependence on absolute position whose avoidance results in the virtues of this robot.

If this line of research proves fruitful, quite apart from the development of a better kind of assembly robot, it may give us some of the representational formalisms and fundamental principles which so far are lacking in the use of force-sensing in assembly work.

Unfortunately there currently exists not other description of this work than the PhD thesis, which has only recently been submitted [Deacon 97]. Once this has been published, other publications will follow.

6 Relation to Other Work

There are two important points worth making concerning our relationship to other work. The first of these is that we see our underlying philosophy as being closely related to that of Brooks' research programme at MIT. At first sight there is little in common between the automated planning and control of industrial assembly tasks and the *subsumption architecture* on which Brooks bases his mobile robots, especially since Brooks strongly eschews any form of planning, representation, or symbolic computation. Although we do not (in our assembly robots) use Brooks' subsumption architecture, the general principles behind our behaviour-based assembly agent are similar to those behind the subsumption architecture. The important difference is in our use of behavioural modules.

Brooks's subsumption architecture sees behaviours as functional capabilities which may emerge in interactions between system components. Thus although they may be viewed externally as units of behaviour, internally they need not be implemented as units. In other words, behaviours are (sometimes) tacit in subsumption systems. We here insist on implementing behaviours as explicit behavioural modules — enough to form a complete virtual assembly system. Since assembly robots must be able to follow an assembly plan, and since in our hybrid architecture the terms in which the planner makes its plans are the capabilities of the behavioural modules, it is necessary for the behavioural modules of the assembly agent to form a complete virtual assembly system. Thus at least some of the behaviours must be implemented as explicit behavioural modules. This makes them not only units of functional capability both externally and internally, but also abstraction devices which enable the existence of a tractable representation of the world, and which by means of their practical capability connect (ground) that representation to the world. This is the crucial feature of the architecture which gives the planner its simplicity and reliability. And since this gain in simplicity outweighs (so we claim) the extra complexity of the assembly agent, the overall system complexity is reduced.

The kind of representation of its world used by the SOMASS system is similar to the indexical-functional representation which Agre and Chapman developed at about the same time, and argued on quite separate grounds should be both specially robust and computationally economical [Chapman and Agre]. They base their argument on a Heideggerean critique of classical AI planning systems[21], and claim to have demonstrated its virtues in their PENGI implementation. This has been much criticised on the grounds that it is trivially simple and evades by sleight-of-simulation many important robotic issues. In practice this simple notion of indexical-functional relationship needs to be augmented by relating it to goal-seeking mechanisms, as sketched in [Hallam and Malcolm 94], but it remains a sound idea with good provenance.

[21] This is not new — Dreyfus long ago mounted a Heideggerean critique of Minsky's frame systems.

So in summary the important architectural difference between MIT behaviour-based mobile robot systems and Edinburgh behaviour-based assembly robot systems is our hybrid architecture, a hybridisation of a behaviour-based agent with a classical ideal-world planner. For that reason we (Edinburgh) insist that our behaviours are implemented as explicit modules, and we use the behaviour-based system to define the classical ideal-world planner for which it is the execution agent.

7 A Philosophical Viewpoint

These kinds of philosophical consideration are of little interest to the industrial user or development engineer, but they do matter to the serious researcher in Artificial Intelligence, so I shall briefly explain the philosophical implications of this hybrid behaviour-based approach here. The branch of philosophy is philosophy of mind, and the relevance to Artificial Intelligence and robotics is that in some rather limited and specialised way we are putting some kind of a "mind" into a machine. In biology we can see that there are many different ways of building animals that walk about on the land, and we can see that some of them are capable of considerable development, and some are strictly limited. For example, a basic design choice is whether to have the skeleton outside, or inside. You are much more resistant to damage if you have your skeleton outside, but there are strict limits to how big an animal with an exoskeleton — an insect — can get. The future lies with endoskeletons, because they can grow much bigger, grow big brains, and so on.

Similarly there are many different ways of trying to build minds into machines. Three which are currently in vogue in artificial intelligence are knowledge-based systems, connectionist systems, and behaviour-based systems. It may be the case that one or more of these is like an insect, a dead-end which can only be developed so far, and no further. In fact that is exactly what some philosophers of mind have been saying for many years. For example, Searle, with his Chinese Room argument [Searle 80], claims to show that it is not possible to produce real understanding in a machine by using any kind of computer program at all. He doesn't rule out making a machine understand, what he does claim is that it can be shown that a computer program is the wrong way, a way which which cannot succeed. And Dreyfus has similarly been arguing that artificial intelligence in general is going about the creation of artificial minds in the wrong way [Dreyfus 92]. Whereas other philosophers, sympathetic to artificial intelligence, such as Boden, Dennett, and Sloman, have been pointing out the errors in the arguments of Searle and Dreyfus.

Without going into the messy details, a view which is held by many of those who espouse the behaviour-based approach to robotics, is that the criticisms of Searle and Dreyfus do in part apply to the knowledge-based paradigm in AI, and to a lesser extent may apply to connectionist systems, but do not apply to behaviour-based robotics. Harnad has identified the crucial problem as what he has called *symbol grounding*, i.e., making sure that the symbols used by a "thinking" machine are actually properly connected to the thing in the world they are intended to be symbols of, and that they remain properly connected [Harnad 90]. The architecture of a behaviour-based robot is designed to ensure just this kind of symbol grounding.

At a deeper level it can be pointed out that the crucial difference between things with minds (such as people) and things without minds (such as rocks), is that in the case of people we explain their behaviour in teleological terms, i.e., in terms of purposes, desires, obstacles, and so on, whereas in the case of rocks we explain their behaviour in terms of mechanistic causality. Therefore, if we want a computer system, or robot, to be properly describable as having a mind of its own, then we must buid it in such a way that the natural and appropriate way of describing its behaviour is teleological. In other words, it must (at least) have purposes. And in order to have purposes, a thing must exist in some world which it has some knowledge and perception of, and it must have a history and a future. The design of behaviour-based robot systems is intended to provide this, too. In fact, the roots of the behaviour-based approach to building robots lie in cybernetics, further back than artificial intelligence, and the cyberneticists (e.g [Wiener 48], [Sayre 76]) felt very strongly about the need to incorporate teleological explanation in biology and the information sciences. They saw the crucial difference between the life sciences and the hard sciences as the need to use teleological explanations, and the justification of teleological explanation derived from the fact that any living thing was built from a complex array of goal-seeking sub-mechanisms. Thus the simplest explanation of its behaviour was teleological; and the explanation in terms of the generative mechanisms was teleological. In other words the teleological explanation was the most scientific[22].

With respect to devices with their own intrinsic purposefulness, i.e., which properly require teleological explanations of their behaviour, note the special contribution made by our hybrid architecture. The insistence on unitary modular implementation of the the important behavioural capabilities, so that they can be exploited by the planner, is the way in which the system is given knowledge of its own capabilities. Thus these parameterised goal-seeking behavioural modules create, via the planner interface, grounded symbolic references to their capabilities, goals, and controlling perceptions. In this case the symbols are few are the "language" is so trivial as to be virtually non-existent, yet this is a secure and extendable platform on which more complex symbolisations can be erected.

It is from the early cybernetic roots, rather than artificial intelligence, that behaviour-based robotics gets its insistence on the low-level coupling of sensing and action in local goal-seeking activities[23]. The immediate reason for this design feature is that it promotes computational simplicity and robustness. The longer term consequence of this design feature, to which I'm drawing attention here, is that it renders behaviour-based robotics immune to the common criticisms of hostile philosophers of mind, and is keeping open an important

[22]I am restricting "teleological" here to its weakest non-conscious form. There is no implication that the device or creature is *aware* of these "purposes", any more than a thermostat is aware of what it is trying to do.

[23]Many of the early cyberneticists are still active in AI and related disciplines, and while cybernetics is now a very low-level, almost amateur discipline, it still exists, and is still influencing thought in related areas.

door to the future development of much more elaborate kinds of artificial “mind” [Hallam and Malcolm 94].

8 Application Areas

The target application area of this research work is robotic assembly. Our long term goal is to develop a methodology providing simpler methods of programming assembly robots, more reliable programs, and programs which are more general in the sense that they encompass wider variations in the assembly task. This generality is largely a natural consequence of the architecture and, although aimed at simplifying the integration of sensing, is also usefully applicable to sensorless assembly.

Although we now suspect that a full development of the implications of the behaviour-based programming of assembly robots will result in a radical redesign of the methods of robot control which underlie robot programming languages (e.g. [Deacon 90], [Deacon 97]), nevertheless we expect that considerable benefit can be obtained (in terms of reliability, ease of programming sensor use, etc.) by using behaviour-based programming methods with current robots and programming languages.

In addition there is the area of mobile robots with manipulative competence, both in those cases where the tasks are known in considerable detail beforehand (such as assembly), and in those cases, such as exploration and maintenance, where the robot has more general goals, and must devise its own manipulation strategies when it encounters the problems. As a consequence of one of our ex-students working in the area, it has been brought to our attention that current research work in the automation of building construction work, and in particular the use of robots to build walls, could benefit from our research. The benefits here derive from the establishment of a substratum of useful competences, devised in such a way as to simplify any “thinking” (such as planning) which the robot — or the robot operator — may need to do, and substantially reducing the need to engineer the environment into conformity with over-precise models[24].

We expect that the high-level competences offered by a behaviour-based interface will also be useful in cases of robot tele-operation where there are restrictions on the interface capacity. For example, there are bandwidth problems in communication with underwater mobiles which restrict the use by the operator of high-bandwidth sensing (such as vision). Bandwidth is also restricted by time delays, such as in terrestrially controlled space robots, which preclude the use of the operator in low-level control loops, such as sensor-guided motion. A robot interface engineered in terms of *behavioural operators* is inherently low in bandwidth and undemanding in response time, and is thus particularly suited to applications with these kinds of problems.

9 Conclusion

This hybrid behavioural/knowledge-based approach to robotic assembly shows promise in the short term of finding more robust and computationally economical ways of handling uncertainty and integrating sensing. In the medium term it may have implications for the design of robot assembly arms, e.g., removing position control as the primary control, and movements of the arm in space as the primary atomic operation.

It has already shown that the architecture can (in a toy experimental domain) support a complete integrated planning and execution system operating from the highest level of generality, and has shown itself capable of further development. To the best knowledge of the author, it was the first completely integrated assembly planning and execution to be implemented[25].

Finally, in the long term it shows promise of keeping open a development route for the implementation of more advanced forms

[24] For example, we note that in many automations of brick laying, the use of specially accurate bricks is required, whereas we would regard that as a symptom of using the wrong (Newtonian or classical) kind of approach.

[25] The later Handeye [Lozano-Perez et al 92], while more ambitious and capable in the lower levels, did not reach the level of generality of the SOMASS system.

of artificial intelligence, as indicated by its implementation of an elementary level of grounded symbolisation, and immunity from the defects alleged to damn other AI research paradigms by critical philosophers of mind and cognitive scientists.

References

[Ambler, Bellos, and Popplestone 75] Ambler, A.P., and Popplestone, R.P., *Inferring the positions of bodies from specified spatial relationships*, Artificial Intelligence 6, 2 (1975), pp 228-233.

[Balch 92] Peter Balch, *A software architecture for robot assembly* Edinburgh University Dept of Artificial Intelligence Working Paper 235, 1992

[Balch 92a] Balch, Peter, *A 3D shape attractor*, Edinburgh : Edinburgh University, c1992 SERIES :DAI research paper ; no.573

[Berlekamp, Conway and Guy 82] Berlekamp, E.R., Conway, J.H., and Guy, R.K., vol 2 of *Winning Ways*, Academic Press, 1982.

[Brooks 86] Brooks, R.A., "Achieving Artificial Intelligence Through Building Robots", AI Memo 899, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, May 1986.

[Chapman and Agre] Chapman, David, and Agre, Philip, *Abstract Reasoning as Emergent from Concrete Activity*, in "Reasoning about Actions and Plans", eds Georgeff and Lansky, Morgan Kaufmann 1986.

[Chong and Conkie 92] Chongstitvatana, Prabhas, and Conkie, Alistair, *Active mobile stereo vision for robotic assembly*, Edinburgh University, DAI research paper no.590, 1992.

[Chong and Conkie 90] Chongstitvatana, Prabhas, and Conkie, Alistair, *Behaviour based assembly experiments using vision sensing* Edinburgh University, DAI research paper no.466, 1990.

[Conkie and Chong 90] , A Conkie and P Chongstitvatana, *An Uncalibrated Stereo Visual Servo System*, Proc British Machine Vision Conference, Oxford 25-27 September 1990; also Edinburgh University DAI RP 475.

[Churchland 89] Churchland, Paul M., *A neurocomputational perspective : the nature of mind and the structure of science*, Cambridge, Mass. London : MIT Press, 1989

[Deacon 90] G E Deacon, "Achieving Robotic Assembly by Sensing and Controlling Spatial Relationships", Proc IEEE International Workshop on Intelligent Motion Control, Istanbul, Turkey, 20-22 August 1990. (Also DAI Research Paper No. 485.)

[Deacon et al 93] Deacon, Graham E., Low, Poh Lian, and Malcolm, Chris, *Orienting objects in a minimum number of robot sweeping motions*, Edinburgh University, DAI research paper no.619, 1993.

[Deacon 94] Deacon, Graham E., Malcolm, Chris, *A robot system designed for task-level assembly*, Edinburgh University, DAI research paper no.687, 1994.

[Deacon 97] Deacon, Graham E, *Accomplishing Task-Invariant Assembly Strategies by Means of an Inherently Accommodating Robot Arm*, Ph.D. Thesis, Department of Artificial Intelligence, University of Edinburgh, submitted September 1995, to be published late 1997.

[Dreyfus 92] Dreyfus, Hubert L., *What computers still can't do : a critique of artificial reason,* Cambridge, Mass. London : MIT Press, c1992

[Gardner 61] M Gardner, *More Mathematical Puzzles and Diversions* 1961, Penguin 1966 edition.

[Hallam and Malcolm 94] Hallam, J.C.T., and Malcolm, C.A., *Behaviour, Perception, and Action — the View from Situated Robotics*, Phil. Trans. R. Soc. Lond. A (1994) **349**, 29-42, 1994.

[Harnad 90] Harnad, S., *The Symbol Grounding Problem*, **Physica D**, 1990.

[Harris 90] M Harris, *Attracting similar shapes towards each other*, Proc "Simulation of Adaptive behaviour: From animals to animats", Paris, Groupe de BioInformatique, ENS, France, September 1990.

[Kim 92] Kim, Taehee, *The development of vibration sensors as event signature sensors in assembly*, Edinburgh University, DAI technical paper no.14, 1992.

[Kim 96] Kim, Taehee, *The Development of Touch Sensors for Guarded and Directed Motion in Behaviour–Based Robotic Assembly Systems*, Edinburgh University, Department of Artificial Intelligence, PhD thesis, 1996.

[Lozano-Perez and Brooks 85] Lozano-Perez, T., and Brooks, R., *An Approach to Automatic Robot Programming*, MIT AI Lab, AI Memo 842, 1985.

[Lozano-Perez et al 92] Lozano-Perez, Tomas, et al, *Handeye : a robot task planner*, Cambridge, Mass. London : MIT Press, c1992

[Malcolm 87] C Malcolm, *Planning and Performing the Robotic Assembly of Soma Cube Constructions*, MSc Dissertation (shared Xerox prize for best project), Edinburgh University, September 1987.

[Malcolm and Smithers 88] C Malcolm and T Smithers, *Programming Assembly Robots in Terms of Task Achieving*

Behavioural Modules: First Experimental Results, Proc International Advanced Robotics Programme, **Second Workshop on Manipulators, Sensors, and Steps towards Mobility**, October 1988, Manchester, UK, pp15.1-15.16; also Edinburgh University DAI RP 410.

[Malcolm and Smithers 89] C Malcolm and T Smithers, *Symbol Grounding via a Hybrid Architecture in an Autonomous Assembly System*, in in "New Architectures for Autonomous Agents" ed Pattie Maes, MIT Press and North Holland; and in June 90 issue of "Robotics and Autonomous Systems", Vol 6 nrs 1 and 2 (Special Issue – Designing Autonomous Agents) Elsevier Scientific Press, Amsterdam, 30 pages; also available as Edinburgh University DAI RP 420.

[Malcolm Smithers and Hallam 89] C Malcolm, T Smithers, and J Hallam, *An Emerging Paradigm in Robotics*, Proc 2nd **Intelligent Autonomous Systems** conference in Amsterdam, December 13-15, 1989, Edinburgh DAI RP 447 Oct 1989.

[Moravec 84] H P Moravec, "Locomotion, Vision, and Intelligence", in "Robotics Research", eds Brady and Paul, MIT Press, 1984, p215-227.

[Pfeifer 96] Pfeifer, Rolf, *Fungus Eaters*, Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB'96), Cape Cod, MA, USA, September 9-13, 1996.

[Sayre 76] Sayre, Kenneth M., *Cybernetics and the philosophy of mind*, London : Routledge and Kegan Paul, 1976

[Searle 80] John Searle, *Minds, Brains, and Programs*, Behavioral and Brain Sciences, 3, p417-457, 1980.

[Smithers and Malcolm 89] T Smithers and C Malcolm, *Programming Robotic Assembly in terms of Task Achieving*

Behavioural Modules, **Journal of Structural Learning**, vol 2 no 10, 1989, 15 pages; also Edinburgh University DAI RP 417.

[Wiener 48] Wiener, Norbert, *Cybernetics or control and communication in the animal and the machine*, New York : J. Wiley, [1948]

[Yin et al 84] Yin, B., Ambler, A.P., and Popplestone, R.J. "Combining Vision Verification with a High Level Robot Programming Language", in Proceedings of the 4th International Conference, ROVISEC, 1984, 371-379.

[Yourdon and Constantine 79] E Yourdon and L L Constantine, *Structured Design*, Prentice-Hall, New Jersey, 1979.