"Processing Modules for Hybrid, Real-Time, Distributed Robot Architectures", Proc SPIE Intelligent Systems and Automated Manufacturing, Pittsburgh, USA Oct 1997

PROCESSING MODULES FOR HYBRID, REAL-TIME,DISTRIBUTED ROBOT ARCHITECTURES

Paul Chernett, Martin Colley, Vic Callaghan[†]

Department of Computer Science, University of Essex, Colchester, UK.

ABSTRACT

New experimental hardware for research into architectures for distributed intelligent embedded systems is proposed that will provide a wide range of communication media including non-deterministic broadcast such as Ethernet, deterministic broadcast such as CAN and processor busses such as VME. The emphasis is on large scale system integration rather than provision for individual capabilities.

A prototype implementation of some of the proposed hardware and software modules is described together with their use in several research projects.

Keywords: Hybrid architecture, distributed systems, autonomous vehicles, real-time, agricultural vehicles, subsea vehicles

1. INTRODUCTION

This paper describes new experimental hardware for research into architectures for distributed intelligent embedded systems in the University of Essex Brooker Laboratory for Intelligent Embedded Systems[1]. This laboratory, started in 1995 brings together much of the research of the university's Computer Science Department and is centered around autonomous vehicles although other applications (most notably Intelligent Buildings) are also being investigated. In recent years work has focused on the Agricultural and Sub-sea vehicles.

Previous work has used a group of small vehicles based on a half-height VME bus. Although in some ways more powerful than some other small-robot systems the slightly out of date technology used has led to a single processor architecture using the VME bus solely for communicating directly with relatively "dumb" sensors and actuators.

A principal aim of the work has been to integrate the achievements of the different research groups in the department including logic-based planning, vision processing, and machine learning as well as behavior-based approaches. We have found that while the old architecture was able to accommodate experiments in any one of these fields it proved inadequate when attempts were made to integrate all these capabilities on board a single vehicle.

We have therefore decided to move our emphasis away from providing individual capabilities to towards the integration of the existing state-of-the-art across several research fields covered by the department as a whole. The paper will describe some of these research fields, some of the application areas that we intend to work in, the design requirements for the new hardware and finally will describe the modules implemented so far.

2. RESEARCH AIMS

The hardware described in this paper is intended to provide experimental facilities for research in several different robotic, engineering and computer science traditions. We believe that each of these traditions has been successful in providing solutions and methodologies to some of the problems of building intelligent systems that need to operate in real environments and to be useful in real human applications. However our goal is now to find ways to integrate these achievements into complete working systems.

[†] email cherp@essex.ac.uk, martin@essex.ac.uk, vic@essex.ac.uk

We shall now summarize some of these traditions before conjecturing on the problems we expect to face when attempting to integrate them in a single system.

2.1 Behavior-based approaches.

Even though we are mostly concerned with systems that retain a human in the control loop, it is evident that it is advantageous for advanced vehicles to posses a high degree of autonomy. In the case of remotely operated vehicles (for example a subsea vehicle being controlled from he shipboard end of a 2000 meter umbilical) the remote operator often has a restricted view of the vehicle's environment, there is often significant latency, temporal indeterminacy and low bandwidth in the communications links (for example inter-planetary exploration) . It is vital when remotely controlling expensive and maybe dangerous vehicles that they are able to protect themselves and their environment without operator intervention. In the case of operating highly complex vehicles, such as modern agricultural machines, autonomy is useful in reducing the cognitive load on the operator. By taking over some tasks, for instance maintaining correct cutter height, an "intelligent" vehicle can allow an operator to maintain higher speeds with lower stress levels both for the operator and machine.

The behavior-based approaches, often inspired by the biology and ethology, and pioneered by Brooks[2], Steels[3], Maes[4] and Mataric[5] have provided both a developmental and practical implementation methodology for building and combining general "competencies" in robotic vehicles. In outline, behavior-based robotics avoids a central complex world representation, in which a central complex computational engine plans future action and then directly commands "dumb" actuators. Instead it proposes a highly distributed model in which the robot system is synthesized from a number of "behaviors". Each behavior is implemented as an asynchronous locus of control that directly maps sensor input to actuator output. There is no central memory shared between behavioral modules all memory being local to them. There are various schools of thought about inter-behavior communication and the references given above should be consulted for the details. The behavior-based robotics scene in the UK is currently very active.

Our work in this area has concentrated on behaviors that are continuously active but at varying levels of activation. Outputs shared between behaviors are summed and behaviors can output to other behaviors forming networks. We implement behaviors as proportional (or PID) controllers allowing the application of conventional control theory. The results are tending to confirm that this approach provides an effective programming methodology for the defining and combining of basic competencies that lead to predictable, stable, robust complex behaviors with smooth and efficient trajectories[6].

Autonomous competencies which we are fairly confident of providing reliably, in an agricultural environment say, include obstacle avoidance, orientation to goals, wall (or hedge, fence, ditch) following, following other vehicles (or people), maintaining desired distances, e.g. cutter heights, distance from a lead vehicle and maintaining pressures, speeds etc. of mechanical sub-systems.

2.2 Logic based approaches (planning)

While behavior-based approaches provide robust and smooth control in the rapid (sub 1Hz, say) range they will always fail at some level at local minima in the behavior space - i.e. where a combination or sequence of behaviors lead to a return to a previous state and thus to cyclic behavior that ceases to make progress towards a goal. An example of this from the animal kingdom is the bird that flies in through an open glasshouse door and sees her distressed fledgling through the glass at the closed end. Although the bird is able to learn very quickly to avoid the glass between herself and her objective she is unable to overcome her maternal instincts sufficiently to fly directly away from it in order reach her goal. Solving such problems requires the construction of a world model and the ability to use it to predict the results of future planning decisions that may involve the deferral or suppression of some behavioral impulses.

Traditional approaches to planning, when applied to real-time fields such as robotics suffer from a problem that is almost the direct corollary of the "local minima" problem of the behavior-based school. In the traditional approach a database of facts about the world and rules that produce a plan of actions given a goal to achieve. The plan is then executed to the point of success or known failure, the database is then updated with new knowledge gained from sensors and the cycle restarted. This algorithm is characterized as "plan-then-execute". In a static world this approach can be highly effective but in real rapidly changing worlds plans are invalidated part way through execution leading to poor robustness and efficiency. More recent approaches strive to interleave planning and execution or even perform both in parallel[7].

We hope that the distributed architecture of our test bed will allow experiments in hybrid approaches that integrate planning and behavior-based methods. One approach will be to view the inputs to the behaviors or behavior systems as the primitive actions of a centralized planning process. Sensory input is passed both to the individual behaviors and is also used to update the planning database. New plans are generated at as high a rate as possible while the behavior based system autonomously executes the current plan.

A more radical approach will be view planning as another set of behaviors - but behaviors with relatively large (and long!) memories. Rather than a single central planner multiple planners will each be equipped with sufficient knowledge and fed with just that sensory data necessary to fulfill some planning sub-task. However we are at an early stage in this inquiry and many problems need to be resolved before we can move to the experimental stage.

2.3 Vision-based Systems

Vision based systems have long been thought to hold out the promise of increasing the generality of competence in autonomous and semi-autonomous vehicles. The machine vision research communities have made great progress in recognizing, tracking, and locating objects in the visual field. The cost of the necessary hardware from CCD cameras, digitizers and compute engines has fallen and continues to fall rapidly. A principal barrier to progress is now one of being able to perform the massive amount of computation necessary at rates fast enough for real-time control and at low enough cost to be commercially viable. We will be exploring the use of highly parallel programmable hardware to overcome both of these difficulties[8].

3. APPLICATIONS

3.1 The Computer "Co-Pilot"

When operating highly complex vehicles, it is usually beneficial to reduce the cognitive load on the operator. By taking over some tasks, such as maintaining correct cutter height, an "intelligent" vehicle can allow an operator to maintain higher speeds with lower stress levels both for the operator and machine. Ideally, a driver would only have to provide high level commands leaving the second-to-second management of the machine to a computer. Already we have systems working within a laboratory that allow operators to provide a set of way points (such as the end points of ploughing or cutting lines) leaving path planning and obstacle avoidance to the computer.

The principle question we seek to answer is "which vehicle tasks would be appropriate to delegate to a computer and which should be left to the operator?"

3.2 Remote Control (teleoperation)

Vehicles can be remotely controlled either from positions sufficiently close to the vehicle to see and safely control them or at a distance where transmission of CCTV images is required. An example of the former might be a farmer locally controlling one or more vehicles in a field (maybe keeping at a safe distance for spraying) whilst an example of the latter might be control of a sub-sea vehicle from the ship-board end of a 2000 meter umbilical.

Even though this research is concerned with systems that retain a human in the control loop, it is evident that it is advantageous for such systems to possess a high degree of autonomy (of a similar type to the co-driver system). In the case of remotely operated vehicles the remote operator often has a restricted view of the vehicle's environment, there is often significant latency, temporal indeterminacy and low bandwidth in the communications links. It is vital for remotely controlled, expensive and maybe dangerous vehicles to be able to protect themselves and their environment without operator intervention.

We have already successfully demonstrated remote control of robots over low-bandwidth, high latency networks[9] have current projects to extend this to agricultural and subsea environments.

3.3 Find, Fetch and Carry

Locating and collecting objects from fields is a common task in farming (e.g. collecting boxes of fruit/vegetables, hay/straw bales) and many other applications of autonomous and semi-autonomous vehicles. One of the first goals for our experimental vehicles is to produce a demonstration of a land vehicle locating and collecting hay bales. There are a number of possible

approaches to solving this problem such as utilising GPS information from Harvesters as they drop bales to identify their approximate location and then plan a route to most effectively visit and collect the bales.

Most applications in this category rely in some part on machine vision systems. Thus, most of the work completed at Essex to date has focused on machine vision and the recognition of objects in their natural environment. Currently, single processor machines are being used to identify a set of bales in static images. This work will be extended to deal with dynamic images and to these ends the vehicle is equipped with VME racks that can house many high performance processor boards. We have also developed hardware to perform some of the simpler, but computationally intensive vision processing operations, such as frame differencing and compression for transmission over low bandwidth links.

3.4 Multi-Vehicle Co-operation

There are persuasive arguments in favour of replacing large expensive machines by numerous cheaper machines. This strategy promises increasing reliability - the system continues to function despite one machine failing - and scaleability - large farms would simply have more mini farm-robots than a smaller farm (but the small farmer would still benefit from the economy of scale associated with the overall market). Ideas and work are still at a rudimentary stage but initial simple applications could include the creation of “driverless trailers” that would form themselves up in to loose “trains” and follow a lead tractor while getting themselves through narrow gates, avoiding low bridges, passing ramblers and errant sheep. Other possibility is the use of a large number of relatively simple and cheap “browsers” to perform some task like root vegetable picking or mowing while being “supervised” by a small number of more intelligent “herders”.

4. REQUIREMENTS

Each application area or control methodology will require a different architecture, and different disposition of computational and communication resources. What we are aiming to design is a set of building blocks derived from our past experiences and which are currently being used (or shortly will be used) in ongoing projects.

As previously described, the experimental vehicles we are particularly interested in are autonomous (or at least semi-autonomous) with some form of distributed processor system. This raises some interesting architectural issues which are addressed in the following subsections.

4.1 Real-time Control Issues

When using remotely operated vehicles(ROVs) with no autonomous capability (such as most current subsea ROVs) real-time control issues are dealt with by the human operator. The vehicle just provides a continuous stream of sensor information and awaits new instructions. However, autonomous and semi-autonomous vehicles present a completely different control problem. The decisions required to control the vehicle must now by made by the vehicle itself. As most of these decisions must be made in something approaching real-time the designer of the vehicle is faced with two approaches; either to limit the amount of autonomy, given to the vehicle, or to provide a control system capable of producing real-time responses. In most cases the level of autonomy will also be dependent on the amount of computing resources available, either single or multi processor.

Limiting the amount of autonomy given to the vehicle may well provide real-time responses, but at the expense of vehicle flexibility. Such vehicles will most likely utilize single processors and have their autonomy limited to simple actions such as moving to a given location, following a navigational beacon or avoiding obstacles. It is unlikely that all of these actions could be carried out simultaneously. In some environments this may be exactly what the operators wants. A list of actions to be performed is given to the vehicle, which will then execute them making minor corrections if necessary, but stopping and reporting back to the operator if a more serious problem occurs.

Where a higher degree of autonomy is required it is inevitable that some form of multi processor configuration will be required. Although this could be configured as a central, more powerful resource replacing a uniprocessor, it could also be used in a distributed configuration. The choice which multiprocessor configuration is used will be dependent on the amount of shared

information required by the application. In general compute intensive tasks such as vision processing are going to require a shared memory multiprocessor configuration, whereas control functions can be distributed in a more loosely coupled fashion.

We see our own needs as being a hybrid distributed system, using shared memory multiprocessor systems to perform compute intensive tasks such as vision processing and forward planning together with a network of loosely coupled distributed processors for the vehicle control systems.

4.2 Communication Issues

Perhaps the most contentious area of distributed real-time system design is that of the communications system. The principle problem of communicating in a real-time environment is guaranteeing that the data arrives within the specified time constraints. Two factors will determine whether or not this can be achieved; the bandwidth of the communications system, and the degree of determinism provided. Clearly there will be some trade-offs in this area, but whatever mechanism is used it must guarantee to deliver the data within a specified time limit.

Tightly coupled, shared memory systems are fairly well defined, they have a very high bandwidth and depending on the access arbitration system used can also offer a very high degree of determinism. However, the same is not necessarily true for network based systems. Although the overall bandwidth of a network based system can be modified fairly easily, changing the degree of determinism usually requires a change in topology.

The highest degree of determinism can only be obtained where an upper bound can be placed on the transmission time of the data. From the network view point this implies a rigidly controlled access mechanism, where it is possible to determine when and how much each node will transmit. This is likely to be based on a token access system such as token ring or token bus, where it is possible to calculate the worst case transmission delays in advance. We would expect that this type of network topology would only be used critical data where guaranteed arrival times are required.

The next level of determinism is provided by the prioritized broadcast based systems such as CANbus. Using this type of network we are able to guarantee that the highest priority data will always be delivered first, but within the priority level no delivery guarantees can be given. It is possible to envisage a system where the majority of data is sent at a low priority, with the higher priority levels only being used to deliver critical data. We would expect that this type of network would be used for vehicle control. Indeed this is why CANbus was developed.

At the lowest level are the non-deterministic broadcast networks such as Ethernet. Using this type of network we are not able to provide any kind of deliver time guarantee, except to say that the data will be delivered at some time. For any kind of real-time or control system this is unlikely to be acceptable. However, we should not rule out using this type of network as there will be classes of data for which this level of delivery guarantee is still acceptable.

5. PROPOSED SOLUTIONS

As stated in the previous section, we are interested in the use of distributed architectures for robotic vehicles, making use of both closely and loosely coupled systems. This section outlines some of the solutions we are proposing. These solutions in no way imply this is the best or only approach, they are used as an example to illustrate a general approach to the problem area.

5.1 General Architecture

Within the environment of a robotic vehicle there are going to be a range of different control and computational tasks that need to be performed. It is unlikely that a single design of processing element will be suitable for all tasks. We therefore propose to use a number of heterogeneous processing elements, each suited in some way to a particular task. As we are going to provide a distributed environment such an approach becomes feasible.

The general architecture will be to provide one or more large shared memory compute resources. These will be used to perform planning operations, vision processing and other compute intensive actions. Together with a number of distributed embedded processors that will be used to perform local control functions, such as steering or engine control. The intention is to provide an appropriate level of processing capability at the location it is needed, rather than centrally.

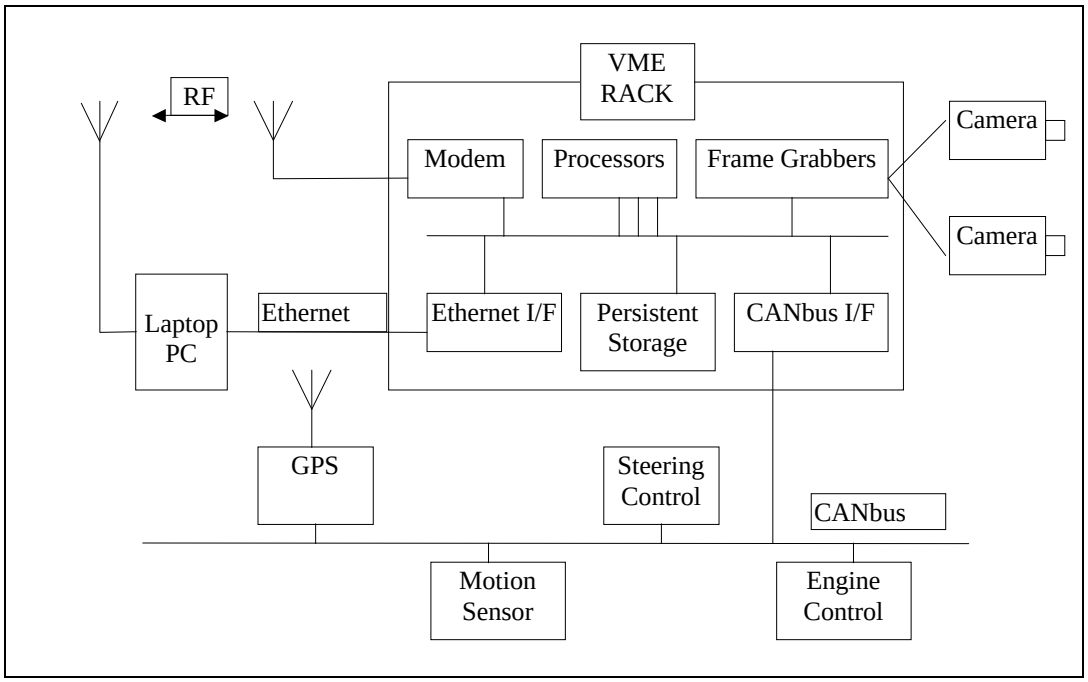

Figure 1 A typical configuration using all the processing elements

This architecture will also determine the types of interconnection strategies that can be used between the various computing components. As with the processors, the type of communications system used will be tailored to the type, and amount, of data that needs to be transferred. The use of common interconnection mechanisms of this type allows the overall control structure of the vehicle to be simplified and made more modular. A typical configuration is shown in Figure 1.

6. PROCESSING ELEMENTS

We are proposing to use three different types of processing element to construct the general architecture outlined previously, some of which are commercially available and others which we have designed and constructed ourselves. The main criterion for using a particular processing element is that it provides the required functionality in the most convenient way.

6.1 Motorola MVME167/147 processing elements.

These are commercial 9U VME boards with a 68040/68030 processor with 8Mbytes of memory and serial/Ethernet connections. We intend to use one or more VME rack based systems containing one or more of these processors as our main compute engines. These will be used to perform functions such as planning, vision processing and other control functions which need to be centralised.

6.2 Motorola MC68306 based processing elements

These are designed at Essex and are built as a 3U sized module with VME-like interface. Each contains a 16MHz MC68306 processor, this is a MC68000 CPU core rated at 2.4 MIPS, together with an extensive range of on chip peripherals. This processor is intended for use as an embedded systems controller requiring a minimum of external logic and interface components. The on chip peripherals that are provided are a dynamic ram controller, an interrupt controller, a chip select/bus time-out generator, a MC68681 compatible DUART, two 8-bit parallel ports and a 16-bit counter/timer. The boards are also equipped with 512Kbytes of ROM, 4Mbytes of dynamic RAM, a time of day clock/calendar, 56 bytes of battery backed up RAM and an ethernet interface (either thinnet or UTP). We intend these to be used where a modest amount of computing power is required, perhaps in conjunction with a specialised interface board.

6.3 Philips 80C592 based processing elements.

These are also designed at Essex and are built as a small standalone module. Each contains a 16Mhz Philips 80C592 processor, this is a 80C51 CPU core together with an extensive range of on chip peripherals. As with the MC68306 processor this is designed to be used as an embedded controller with a minimum of external logic and interface components. The on chip peripherals that are provided are 256 bytes of RAM, three 16-bit counter/timers, a 10-bit ADC with eight multiplexed inputs, two 8-bit resolution PWM outputs, 20 parallel I/O lines, 1Mbps CANbus interface and a serial UART interface. The boards are also equipped with 64Kbytes of ROM and 64Kbytes of static RAM. We intend these to be used to perform the control functions, such as steering or engine management, within the vehicle. The inclusion of a CANbus interface within the design allows us to use these processing elements in a loosely coupled, distributed, fashion, while still maintaining deterministic properties.

7. COMMUNICATION MECHANISMS

We are proposing to use two principle forms of communications mechanism between the various processing elements within the system. One applicable to tightly coupled configurations, and one more applicable to loosely coupled configurations.

For the tightly coupled, VME based multiprocessor system it is envisaged the shared memory, or some other form of backplane communications system will be employed. For the types of problem area utilising these systems a high bandwidth, low latency communications system is essential to ensure rapid access to potentially large shared data sets.

For the distributed vehicle control processors we intend to use the CANbus as the broadcast mechanism. CANbus was designed as a control bus for distributed embedded systems within the automotive industry and allows short command sequences to be multicast to all the controllers on the network. Each message has a priority level assigned to it and should any contention occur during the transmit phase the highest priority message is sent, with the lower priority transmitter backing off and trying again later. The fact that messages are multicast rather than directed allows communications to be subscription based, rather than the traditional peer to peer model. As we see the control structure of a vehicle being based short command sequences to intelligent controllers, this form of communications system would seem ideal.

It is inevitable that at some stage we will need to consider Ethernet connections. However because of their nondeterministic behaviour any use made will be limited to non-critical communication links, such as the initial download of program data. Although it is possible that some kind of radio Ethernet link will be used for remote control of a vehicle. If this is the case we will ensure it is as contention free as possible so as to reduce the nondeterminism to a minimum.

8. CURRENT STATUS

At the time of writing (July, 1997) both the Essex designed boards were at the final soak-testing stage before being replicated. We are currently collaborating with Writtle Agricultural College, Chelmsford, Essex, UK in the construction of a large agricultural vehicle and the hardware described in this paper will be installed as soon as the modules are available. The MC68306 boards will also be used in a “flock” of at least 10 small vehicles to be used in undergraduate courses scheduled to start in January, 1998.

So far the modules have proven to be cheap and reliable and have performed entirely up to specification. We are looking forward to an exciting period of experimentation in which we expect to see much promising work reach its full potential in complete and practical working systems.

REFERENCES

1. Callaghan, V., Chernett, P. & Lyons, D., “The Brooker Lab for Intelligent Embedded Systems”, *Improving the Quality of Teaching and Learning in Computing,* pp11-23, SEDA , July 1995.
2. Brooks R., “Intelligence Without Reason”, Proc. IJCAI-91, Morgan Kaufmann, San Mateo Ca. pp569-595, 1991.
3. Steels L, Brooks R., "The Artificial Life Route to Artificial Intelligence: Building Embodied Situated Agents", Lawrence Erlbaum, 1995.
4. Maes, P. & Brooks, R.A.,“Learning to Co-ordinate Behaviours”, Proc AAAI-91, Boston, MA, pp. 796-802, 1991.

5. Mataric, M.J., “Interaction and Intelligent Behaviour”, PhD Thesis, Massachusetts Institute of Technology, May, 1994.
6. Voudouris, C., Chernett, P., Wang, C.J. & Callaghan, V, “Hierarchical behavioural control for autonomous vehicles”, Proc. 2nd IFAC Int. Conf. on Intelligent Autonomous Vehicles IAV-95, pp267-272, Helsinki University of Technology, Espoo, Finland, Pergamon Press, June 1995.
7. Tate, A., *Readings in Planning*, Morgan Kaufmann, San Mateo, Ca., 1995.
8. Freeman, S., Spacek, L., Callaghan, V. & Chernett, P., “A programmable-logic based multiprocessor engine for real-time vision preprocessing”, Proc. IEE Colloquium on High Performance Applications of Parallel Architectures, pp1-6, London, England, February 1994.
9. Heron G., “Telepresence Robot Control Using Java & WWW”, Final Year Project Report, University of Essex, 1997.

