MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A. I. Memo 1091 **February, 1989**

A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network

Rodney A. Brooks

Abstract. Most animals have significant behavioral expertise built in without having to explicitly learn it all from scratch. This expertise is a product of evolution of the organism; it can be viewed as a very long term form of learning which provides a structured system within which individuals might learn more specialized skills or abilities. This paper suggests one possible mechanism for analagous robot evolution by describing a carefully designed series of networks, each one being a strict augmentation of the previous one, which control a six legged walking machine capable of walking over rough terrain and following a person passively sensed in the infrared spectrum. As the completely decentralized networks are augmented, the robot's performance and behavior repetoire demonstrably improve. The rationale for such demonstrations is that they may provide a hint as to the requirements for automatically building massive networks to carry out complex sensory-motor tasks. The experiments with an actual robot ensure that an essence of reality is maintained and that no critical disabling problems have been ignored.

Acknowledgements This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the research is provided in part by the University Research Initative under Office of Naval Research contract N00014-86-K-0685 and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

Introduction

In earlier work, [1], [2], we have demonstrated complex control systems for mobile robots built from completely distributed networks of augmented finite state machines. In this paper we demonstrate that these techniques can be used to incrementally build complex systems integrating relatively large numbers of sensory inputs and large numbers of actuator outputs. Each step in the construction is purely incremental, but nevertheless along the way viable control systems are left at each step, before the next little piece of network is added. Additionally we demonstrate how complex behaviors, such as walking, can emerge from a network of rather simple reflexes with little central control. This contradicts vague hypotheses made to the contrary during the study of insect walking (e.g. [3], page 112).

The Subsumption Architecture

The subsumption architecture[1] provides an incremental method for building robot control systems linking perception to action. A properly designed network of finite state machines, augmented with internal timers, provides a robot with a certain level of performance, and a repetoire of behaviors. The architecture provides mechanisms to augment such networks in a purely incremental way to improve the robot's performance on tasks and to increase the range of tasks it can perform. At an architectural level, the robot's control system is expressed as a series of layers, each specifying a behavior pattern for the robot, and each implemented as a network of message passing augmented finite state machines. The network can be thought of as an explicit wiring diagram connected outputs of some machines to inputs of others with wires that can transmit messages. In the implementation of the architecture on the walking robot the messages are limited to 8 bits.

Each augmented finite state machine (AFSM), figure 1, has a set of registers and a set of timers, or alarm clocks, connected to a conventional finite state machine which can control a combinatorial network fed by the registers. Registers can be written by attaching input wires to them and sending messages from other machines. The messages get written into them replacing any existing contents. The arrival of a message, or the expiration of a timer, can trigger a change of state in the interior finite state machine. Finite state machine states can either wait on some event, conditionally dispatch to one of two other states based on some combinatiorial predicate on the registers, or compute a combinatorial function of the registers directing the result either back to one of the registers or to an output of the augmented finite state ma-

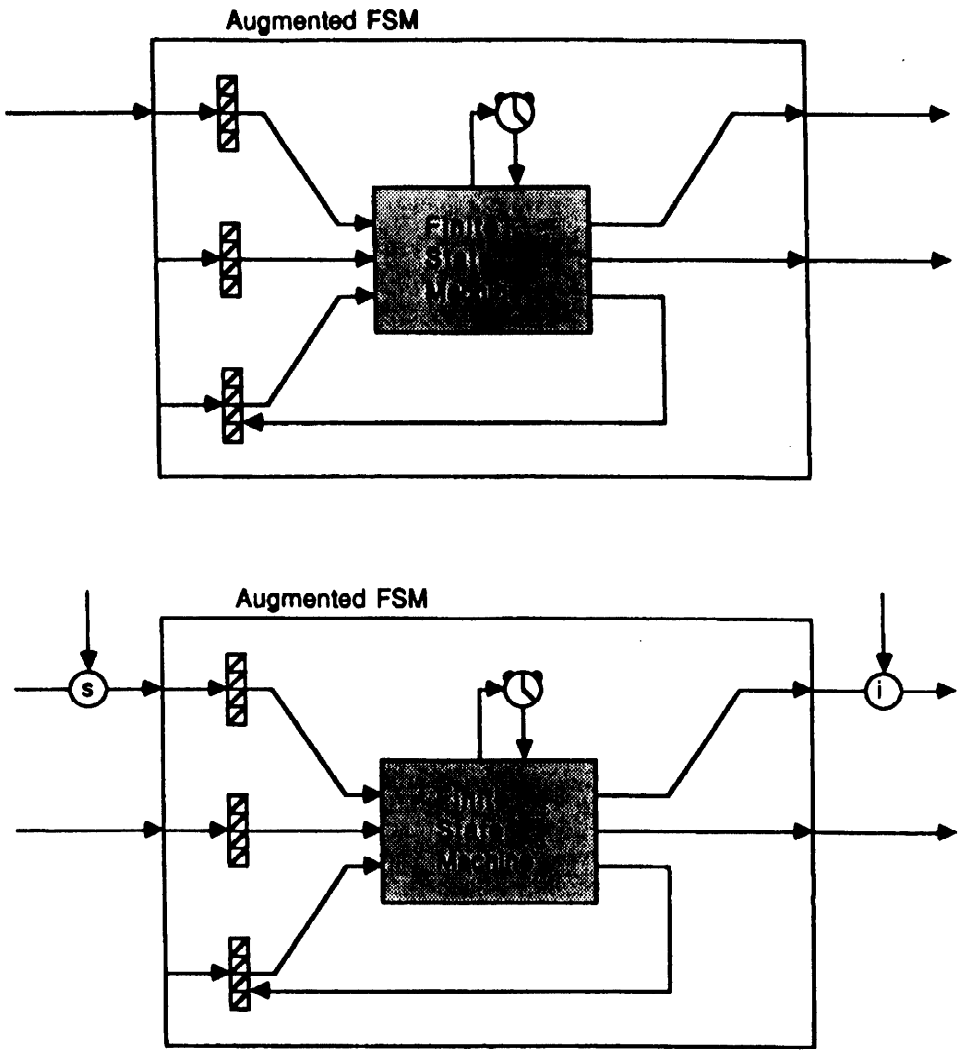

Figure 1. An augmented finite state machine consists of registers, alarm clocks, a combinatorial network and a regular finite state machine. Input messages are delivered to registers, and messages can be generated on output wires. AFSMs are wired together in networks of message passing wires. As new wires are added to a network, they can be connected to existing registers, they can inhibit outputs and they can suppress inputs.

chine. Some AFSMs connect directly to robot hardware. Sensors deposit their values to certain registers, and certain outputs direct commands to actuators.

A series of layers of such machines can be augmented by adding new machines and connecting them into the existing network in the ways shown in figure 1. New inputs can be connected to existing registers, which might previously have contained a constant. New machines can inhibit existing outputs or suppress existing inputs, by being attached as side-taps to existing wires (figure 1, circled 'i'). When a message arrives on an inhibitory side-tap no messages can travel along the existing wire for some short time period. To maintain inhibition there must be a continuous flow of messages along the new wire. (In previous versions of the subsumption architecture[1] explicit, long, time periods had to be specified for inhibition or suppression with single shot messages. Recent work has suggested this better approach [4].) When a message arrives on a suppressing side-tap (figure 1, circled 's'), again no messages are allowed to flow from the original source for some small time period, but now the suppressing message is gated through and it masquerades as having come from the original source. Again, a continuous supply of suppressing messages is required to maintain control of a side-tapped wire. One last mechanism for merging two wires is called defaulting (indicated in wiring diagrams by a circled 'd'). This is just like the suppression case, except that the original wire, rather than the new side-tapping wire, is able to wrest control of messages sent to the destination.

All clocks in a subsumption system have approximately the same tick period (0.04 seconds on the walking robot), but neither they nor messages are synchronous. The fastest possible rate of sending messages along a wire is one per clock tick. The time periods used for both inibition and suppression are two clock ticks. Thus, a side-tapping wire with messages being sent at the maximum rate can maintain control of its host wire.

The Networks and Emergent Behaviors

The six legged robot is shown in figure 2. We refer to the motors on each leg as an α motor (for *advance*) which swings the leg back and forth, and a β motor (for *balance*) which lifts the leg up and down.

Figure 3 shows a network of 57 augmented finite state machines which was built incrementally and can be run incrementally by selectively deactivating later AFSMs. The AFSMs without bands on top are repeated six times, once for each leg. The ASFMs with solid bands are unique and comprise the only central control in making the robot walk, steer and follow targets. The ASFMs with striped bands are duplicated twice each and are specific to particular legs.

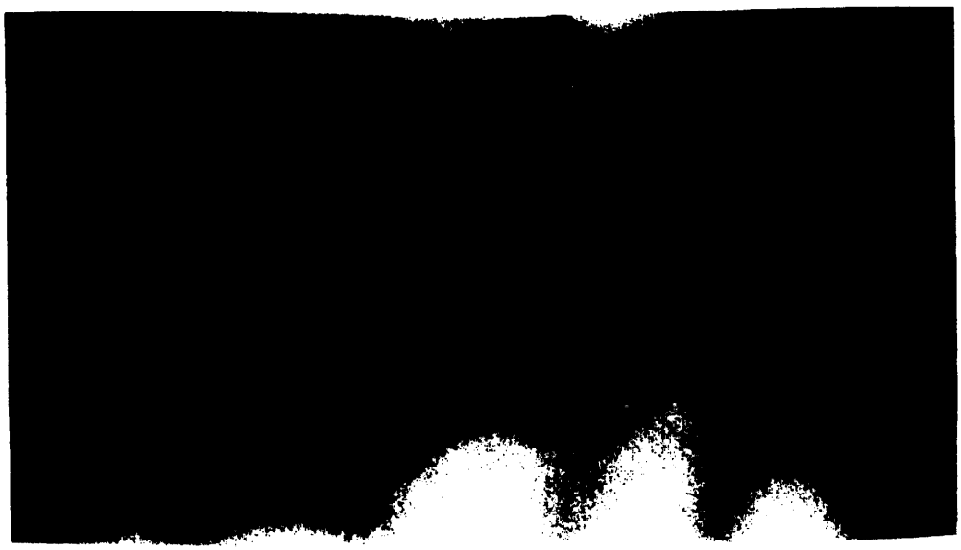

Figure 2. The six legged robot is about 35cm long, has a leg span of 25cm, and weighs approximately 1Kg. Each leg is rigid and is attached at a shoulder joint with two degrees of rotational freedom, driven by two orthoganally mounted model airplane position controllable servo motors. An error signal has been tapped from the internal servo circuitry to provide crude force measurement (5 bits, including sign) on each axis, when the leg is not in motion around that axis. Other sensors are two front whiskers, two four bit inclinomters (pitch and roll), and six forward looking passive pyroelectric infrared sensors. The sensors have approximately 6 degrees angular resolution and are arranged over a 45 degree span. There are four onboard 8 bit microprocessors linked by a 62.5Kbaud token ring. The total memory usage of the robot is about 1Kbytes of RAM and 10Kbytes of EPROM. Three silver-zinc batteries fit between the legs to make the robot totally self contained.

The complete network can be built incrementally by adding AFSMs to an existing network producing a number of viable robot control systems itemized below. All additions are strictly additive with no need to change any existing structure. Figure 4 shows a partially constructed version of the network.

1 **Standup.** The simplest level of competence for the robot is achieved with just two AFSMs per leg, *alpha pos* and *beta pos*. These two machines use a register to hold a set position for the α and β motors respectively and ensure that the motors are sent those positions. The initial values for the registers are such that on power up the robot assumes a stance position. The AFSMs also provide an output that reports the most recent commanded position for their motor.

2 **Simple walk.** A number of simple increments to this network result in

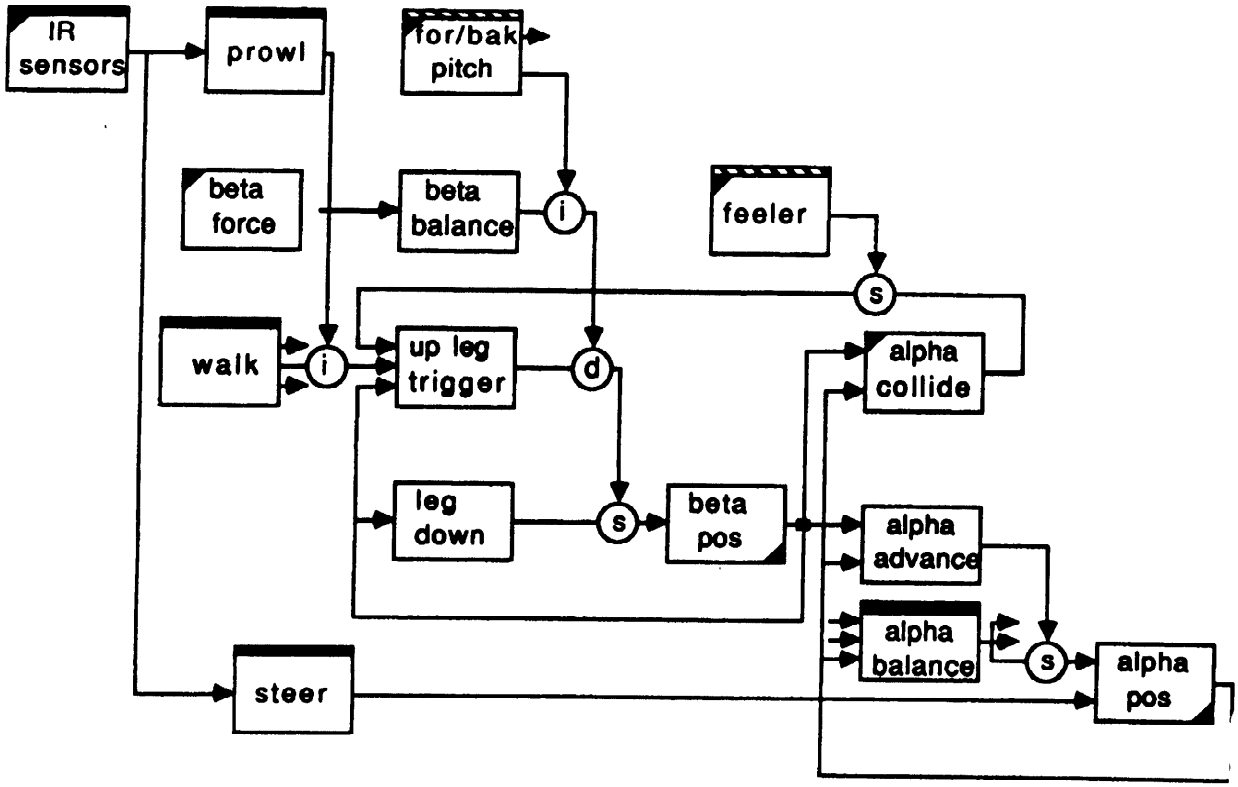

Figure 3. The final network consists of 57 augmented finite state machines. The AFSMs without bands on top are repeated six times, once for each leg. The ASFMs with solid bands are unique and comprise the only central control in making the robot walk, steer and follow targets. The AFSMs with striped bands are duplicated twice each and are specific to particular legs. The ASFMs with a filled triangle in their bottom right corner control actuators. Those with a filled triangle in their upper left corner receive inputs from sensors.

one which lets the robot walk. First, a *leg down* machine for each leg is added which notices whenever the leg is not in the down position and

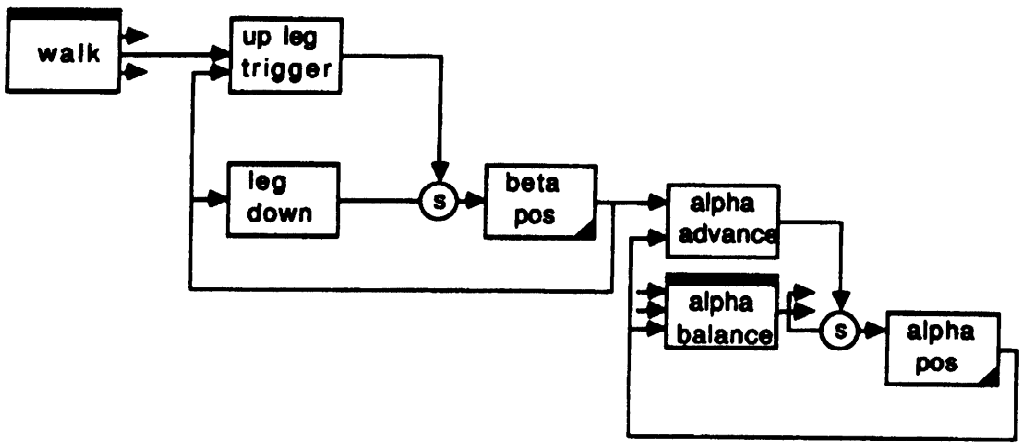

Figure 4. A strict subset of the full network enables the robot to walk without any feedback. It pitches and rolls significantly as it walks over rough terrain. This version of the network contains 32 AFSMs. 30 of these comprise six identical copies, one for each leg, of a network of five AFSMs which are purely local in their interactions with a leg. The last two machines provide all the global coordination necessary to make the machine walk; one tries to drive the sum of leg swing angles (α angles) to zero, and the other sequences lifting of individual legs.

writes the appropriate *beta pos* register in order to set the leg down. Then, a single *alpha balance* machine is added which monitors the α position.

or forward swing of all six legs, treating straight out as zero, forward as positive and backward as negative. It sums these six values and sends out a single identical message to all six *alpha pos* machines, which, depending on the sign of the sum is either null, or an increment or decrement to the current α position of each leg. The *alpha balance* machine samples the leg positions at a relatively high rate. Thus if one leg happens to move forward for some reason, all legs will receive a series of messages to move backward slightly.

Next, the *alpha advance* ASFM is added for each leg. Whenever it notices that the leg is raised (by monitoring the output of the *beta pos* machine) it forces the leg forward by suppressing the signal coming from the global *alpha balance* machine. Thus, if a leg is raised for some reason it reflexively swings forward, and all other legs swing backward slightly to compensate (notice that the forward swinging leg does not even receive the backward message due to the suppression of that signal). Now a fifth ASFM, *up leg trigger* is added for each leg which can issue a command to lift a leg by suppressing the commands from the *leg down* machine. It has one register which monitors the current β position of the leg. When it is down, and a trigger message is received in a second register, it ensures that the contents of an initially constant third register, are sent to the *beta pos* machine to lift the leg.

With this combination of local leg specific machines and a single machine trying to globally coordinate the sum of the α position of all legs, the robot can very nearly walk. If an *up leg trigger* machine receives a trigger message it lifts its associated leg, which triggers a reflex to swing it foward, and then the appropriate *leg down* machine will pull the leg down. At the same time all the other legs still on the ground (those not busy moving forward) will swing backwards, moving the robot fowards.

The final piece of the puzzle is to add a single AFSM which sequences walking by sending trigger messages in some appropriate pattern to each of the six *up leg trigger* machines. We have used two versions of this machine, both of which complete a gait cycle once every 2.4 seconds. One machine produces the well known alternating tripod [5], by sending simultaneous lift triggers to triples of legs every 1.2 seconds. The other produces the standard back to front ripple gait by sending a trigger message to a different leg every 0.4 seconds. Other gaits are possible by simple substitution of this machine. The machine walks with this network, but is insensitive to the terrain over which it is walking and tends to roll and pitch excessively as it walks over obstacles. The complete network for this simple type of walking is shown in figure 4.

3 **Force balancing.** A simple minded way to compenstate for rough terrain is to monitor the force on each leg as it is placed on the ground and back off if it rises beyond some threshold. The rationale is that if a leg is being placed down on an obstacle it will have to roll (or pitch) the body of the robot in order for the leg β angle to reach its preset value, increasing the load on the motor. For each leg a *beta force* machine is added which monitors the β motor forces, discarding high readings coming from servo errors during free space swinging, and a *beta balance* machine which sends out lift up messages whenever the force is too high. It includes a small deadband where it sends out zero move messages which trickle down through a defaulting switch on the *up leg trigger* to eventually suppress the *leg down* reflex. This is a form of active compliance which has a number of known problems on walking machines [5]. On a standard obstacle course (a single 5 centimeter high obstacle on a plane) this new machine significantly reduced the standard deviation, over a 12 second period, of the readings from onboard 4 bit pitch and roll inclinometers. Each inclinometer had a 35 degree range. The standard deviation of the pitch inclinometer fell from 3.592 to 2.325. The standard deviation of the roll inclinometer fell from 0.624 to 0.451. See figure 5 for details.

4 **Leg lifting.** There is a tradeoff between how high each leg is lifted and overall walking speed. But low leg lifts limit the height of obstacles which can be easily scaled. An eighth AFSM for each leg compensates for this by measuring the force on the forward swing (α) motor as it swings forward and writing the height register in the *up leg trigger* at a higher value setting up for a higher lift of the leg on the next step cycle of that leg. The *up leg trigger* resets this value after the next step.

5 **Whiskers.** In order to anticipate obstacles better, rather than waiting until the front legs are rammed against them, each of two whiskers is monitored by a *feeler* machine and the lift of the the left and right front legs is appropriately upped for the next step cycle.

6 **Pitch stabilization.** The simple force balancing strategy above is by no means perfect. In particular in high pitch situations the rear or front legs (depending on the direction of pitch) are heavily loaded and so tend to be lifted slightly causing the robot to sag and increase the pitch even more. Therefore one *forward pitch* and one *backward pitch* AFSM are added to monitor high pitch conditions on the pitch inclinometer and to inhibit the local *beta balance* machine output in the appropriate circumstances. The pitch standard deviation over the 12 second test reduces to 1.921 with this improvement while the roll standard deviation stays around the same at 0.458. Again see figure 5.

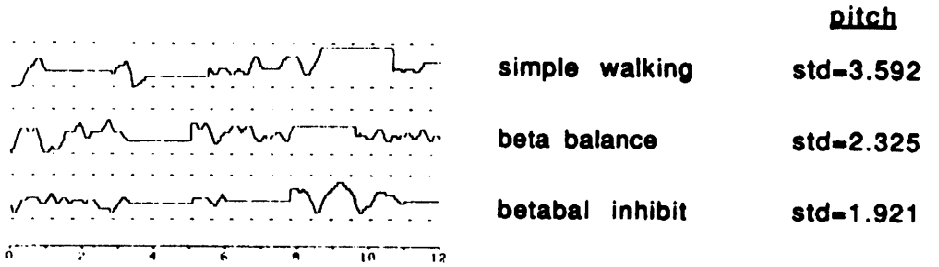

Figure 5. The robot was set walking across a plane witha a single 5cm high obstacle. The graphs above record the measured pitch during three trials each for a duration of 12 seconds. The middle of the range of each graph corresponds to a level body. The upper trace corresponds to simple walking with no force feedback. The middle trace corresponds to walking with the *beta balance* machine included. The lower trace corresponds to the walking machine with *beta balance* inhibited in high pitch situations. The standard deviations for the three trials are displayed at the right.

7 **Prowling.** Two additional AFSMs can be added so that the robot only bothers to walk when there is something moving nearby. The *IR sensors* machine monitors an array of six forward looking pyro-electric infrared sensors and sends an activity message to the *prowl* machine when it detects motion. The *prowl* machine usually inhibits the leg lifting trigger messages from the *walk* machine except for a little while after infrared activity is noticed. Thus the robot sits still until a person, say, walks by, and then it moves forward a little.

8 **Steered prowling.** The single *steer* AFSM takes note of the predominant direction, if any, of the infrared activity and writes into a register in each *alpha pos* machine for legs on that side of the robot, specifying the rear swinging stop position of the leg. This gets reset on every stepping cycle of the leg, so the *steer* machine must constantly refresh it in order to reduce the leg's backswing and force the robot to turn in the direction of the activity. With this single additional machine the robot is able to follow moving objects such as a slow walking person.

Conclusion

This exercise in synthetic neuro-ethology has successfully demonstrated a number of things, at least in the robot domain. All these demonstrations depend on the manner in which the networks were built incrementally from augmented finite state machines.

- Robust walking behaviors can be produced by a distributed system with very limited central coordination. In particular much of the sensory-motor integration which goes on can happen within local asynchronous units. This has relevance, in the form of an existence proof, to the debate on the central versus peripheral control of motion [7] and in particular in the domain of insect walking [3].
- Higher level behaviors (such as following people) can be integrated into a system which controls lower level behaviors, such as leg lifting and force balancing, in a completely seamless way. There is no need to postulate qualitatively different sorts of structures for different levels of behaviors and no need to postulate unique forms of network interconnect to integrate higher level behaviors.
- Coherent macro behaviors can arise from many independent micro behaviors. For instance, the robot following people works even though most of the effort is being done by independent circuits driving legs, and these circuits are getting only very indirect pieces of information from the higher levels, and none of this communication refers at all to the task in hand (or foot).
- There is no need to postulate a central repository for sensor fusion to feed into. Conflict resolution tends to happen more at the motor command level, rather than the sensor or perception level.

As a last observation, there are a few straight-forward engineering improvements which could be made to the current robot which should improve its performance markedly. The first is to add true position sensors to each motor that are accessible to the subsumption architecture. Currently it relies on a knowledge of most recently commanded positions of motors where (especially on rough terrain) this may not correspond well to the actual positions of motors. Secondly, true force sensing, using strain guages, would give much more reliable force readings than the rather indirect methid we use at the moment which is to coarsely time the switching time of the motor current as determined by a build in analog position servo system on each motor.

References

[1] "A Robust Layered Control System for a Mobile Robot", Rodney A. Brooks, *IEEE Journal of Robotics and Automation*, RA-2, April 1986, 14–23.

[2] "Asynchronous Distributed Control System for A Mobile Robot", Rodney A. Brooks and Jonathon H. Connell, *Proceedings SPIE, Cambridge, MA*, 1986, 77–84.

[3] "Neural Basis of Elementary Behavior in Stick Insects", Ulrich Bässler, Springer-Verlag, 1983.

[4] "A Behavior-Based Arm Controller", Jonathan H. Connell, MIT AI Memo 1025, June 1988.

[5] "Use of Force and Attitude Sensors for Locomotion of a Legged Vehicle", Charles A. Klein, Karl W. Olson and Dennis R. Pugh, *International Journal of Robotics Research*, Vol 2, No 2, 1983, 3–17.

[5] "Insect Walking", Donald M. Wilson, *Annual Review of Entomology*, Vol II, 1966, reprinted in "The Organization of Action: A New Synthesis", C. R. Gallistel, Lawrence Erlbaum, 1980.

[7] "Central and Peripheral Mechanisms in Motor Control, Emilio Bizzi, *Tutorials in Motor Behavior*, G. E. Stelmach and J. Requin (eds.), North-Holland, 1980.

Acknowledgements

Grinnell More did most of the mechanical design and fabrication of the robot. Colin Angle did much of the processor design and most of the electrical fabrication of the robot. Mike Ciholas, Jon Connell, Anita Flynn, Chris Foley and Peter Ning provided valuable design and fabrication advice and help.