HIVEMind : Grounding Inference in Cooperative Activity

Aaron Khoo and Ian Douglas Horswill

Computer Science Department, Northwestern University
1890 Maple Avenue
Evanston, IL 60201
{khoo, ian}@cs.northwestern.edu

Abstract

In this paper, we describe HIVEMind, a tagged behavior-based architecture for small teams of cooperative robots. In HIVEMind, robots share inferences and sensory data by treating other team members as virtual sensors connected by wireless links. A novel representation based on bit-vectors allows team members to share intentional, attentional, and sensory information using relatively low-bandwidth connections. We describe an application of the architecture to the problem of systematic spatial search.

Introduction

Robust autonomous robots are notoriously difficult to design. The world continually changes and the robot's sensory systems must continually track those changes. Its modeling systems must track the sensory data, and its control systems must be ready to alter plans and actions to suit the changing model.

Behavior-based systems (Arkin 98) solve these problems very effectively. In their purest form, behavior-based systems divide sensing, modeling, and control between many parallel task-achieving modules called behaviors. Each behavior contains its own task-specific sensing, modeling, and control processes. Behaviors tend to be simple enough to be implemented as feed-forward circuits or simple finite-state machines, allowing them to completely recompute sensor, model, and control decisions from moment to moment. This, in turn, allows them to respond immediately to changes in the environment.

Not surprisingly, the task-specificity and computational simplicity of behavior-based systems are also a weakness. We believe that their greatest weakness is the use of simple propositional representations, which makes most reasoning and planning tasks both difficult and clumsy.

Traditional symbolic reasoning systems, on the other hand, allow the manipulation of arbitrarily sophisticated representations at the cost of increased computational complexity. While that complexity need not always involve exponential-time or undecidable computations, it does generally involve highly serial computations operating on a large database of logical assertions. In principle, modifying such a system to track changes in the environment would require recording dependencies between stored assertions and their justifications such that when the perceptual system added or retracted an assertion, the reasoning system could enumerate and update the set of existing assertions affected by the change. This is a sufficiently complicated process that we know of no implemented physical robots that do it. Instead, the symbolic system is generally equipped with a set of "epistemic actions" that allow a programmer designing the knowledge base to force the reasoner to update specific aspects of the knowledge base at specific times. Any mistakes by the programmer will lead to inconsistencies between the models of the symbolic system and the other subsystems. While tiered architectures combining symbolic and behavior-based systems (Arkin and Balch 98, Firby *et al.* 98, Connell 92) can mitigate these problems by offloading short-timescale interactions from the symbolic system, they do not ultimately solve the problem of keeping the many fragmentary models of the different components in sync with one another and with the world.

These issues are exacerbated in cooperative activity. Rather than one robot with one knowledge base, we now have *n* robots with *n* knowledge bases to keep consistent with one another. Failure to properly coordinate the knowledge bases will ultimately result in failure to coordinate activity. While excellent work has been done on communication protocols for insuring consistency (Cohen and Levesque 91, Tambe 96), in practice, implemented multi-robot systems almost always use behavior-based architectures (Goldberg and Mataric 98, Arkin and Balch 99, Parker 98) and/or shared, centrally world models created using an overhead camera (Stone and Veloso 99).

Role-passing

One alternative to using tiered architectures is to generalize behavior-based systems to more abstract representations. Obviously, the simple parallel networks and finite-state machines used in behavior-based systems will never be

Support for this work was provided by the DARPA Distributed Robotics Program and the U.S. Army Soldiers Systems Command under award #DAAN02-98-C-4023/MOD P3.

able to do arbitrary first-order logic theorem proving. In practice however, neither do any existing robots. Given that, one might hope to use some suitable generalization of behavior-based systems to implement at least most of the representations and inference techniques that are actually deployed on present-day robots.

Since behavior-based systems effectively use propositional representations, variable binding and quantified inference are the main techniques that are hard to implement with them. Agre and Chapman (87) argued for the use of indexical constants as a surrogate for variable binding. Rather than putting a variable binding mechanism into the reasoning engine, they implemented a propositional reasoner whose inputs were driven by an active vision system. The outputs of the vision system directly measured the truth values of a fixed set of literals such as *near*(*the-tiger-that's-about-to-eat-me*). In the metatheory, *the-tiger-that's-about-to-eat-me* is an indexical name whose denotation is determined by the current attentional state of the perceptual system. By redirecting attention of the visual system from one object to another, the system could effectively treat the name as a variable and rebind it from object to object. However, since the reasoning system didn't represent the internal structure of the literals, it was purely propositional and so could be implemented as a feed-forward logic network. Moreover, since vision systems are generally attentive in practice, this "variable binding" mechanism was already present. No additional computations needed to be added to take advantage of it.

The problem with this "indexical-functional" or "deictic" style of representation is that there tend to be more indexical names that there are trackers in the vision system. In practice, it becomes necessary to decide at design time that a certain set of indexicals will share one tracker, while another set will use another tracker. The designer then hopes that there will be no situations in which two indexicals that share a tracker will need to be bound to distinct objects. In addition, there are some times when you just really want to use a quantified inference rule.

Role-passing (Horswill 98) is a variant of deictic representation that solves both these problems. Rather than a large number of task-specific indexicals, the designer uses a small set of domain-independent indexicals, specifically linguistic roles such as *agent, patient, source, destination*, etc. When an indexical is bound to an object, a tracker is dynamically allocated to it and tagged with the name of the indexical. Since the number of linguistic roles is relatively small, we can represent the extensions of unary predicates as bit-vectors, one bit per role. Inference can then be performed using bit-parallel operations in a feed-forward network. Note that this involves an unusual kind of closed-world assumption, in which quantifiers are restricted to the set of objects attended to by the robot at that moment.

Alternatively, on conventional hardware, we can represent a unary predicate extension using a single machine word. We can then compile Horn clause inference over unary predicates to straight-line code consisting only of loads, stores, and bit-mask instructions. While more limited than a full logic-programming system, it does allow us to express much of the kinds of control reasoning that people really implement on physical robots today. Since the reasoning process recomputes all inferences on every cycle of the system's control loop, the reasoning system is able to respond to contingencies as soon as they are sensed. And since the compilation process is very efficient, inference is effectively free – 1000 Horn clauses of 5 conjuncts each can be completely updated at 100Hz using less than 1% of a current CPU.

HIVEMind

In addition to allowing very fast inference, this representation allows for very compact storage of a robot's current set of inferences. Unary predicates are stored in one machine word. Functional values associated with indexicals are stored in an array whose size is equal to the total number of indexicals, a relatively small number. The representation is small enough to allow robots to share information by periodically broadcasting *all* their inferences, or at least all those predicates and functions that might be relevant to other team members. This provides each robot with transparent access to every other robot's state. The result is a kind of "group mind".

We have implemented this technique in the HIVEMind (Highly Interconnected Verbose Mind) architecture for cooperative robot teams. Its simple communication and coordination model allows the team to efficiently maintain a shared situational awareness and to provide hard real-time response guarantees; when a team member detects a contingency, other members immediately share its awareness and respond in O(1) time.

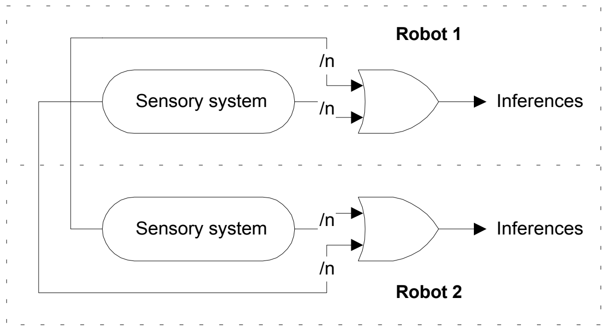

Figure 2 : Abstract view of HIVEMind

Figure 2 shows a HIVEMind configuration for a two-robot team. Each team member has its own inference network that is driven both by its own sensory system and by the sensory data of the other team members. The entire HIVEMind can be considered a single, parallel control network whose components happen to be distributed between the different robot bodies being controlled. Wires crossing between bodies are simulated using the RF broadcast mechanism, so that each member of the team is "connected" to every other member in a web-like structure of virtual wires.

It may seem inefficient for each robot to have its own separate copy of the inference network. However, to have a single robot perform each inference and share the results would require much more complicated coordination protocols (Cohen and Levesque 91) analogous to the multiphase commit protocols used in distributed database systems. Since communication bandwidth is a scarce resource and inference in our system is essentially free, it is more efficient for HIVEMind robots to perform redundant computation.

Aggregation of Data

In an n robot team, each robot's inference network has n distinct sets of inputs, one generated internally, and the rest received from the robot's teammates. These distinct inputs are first fused into a single set of inputs:

$$K = \beta(k_1, k_2, \ldots, k_n)$$

where the k_i are the tuples of inputs from each robot, K is the final fused tuple, and β is some *aggregation function* that performs the fusion. For example, if a particular component of the input was a proposition, the aggregation function might simply OR together the corresponding components of the k_i. Thus the robot would believe the proposition iff some robot had evidence for it. In more complicated cases, fuzzy logic or Bayesian inference could be used. Real-valued data is likely to require task-specific aggregation. For example,

- The team is assigned to scout an area and report the number of enemies observed. Each team member has a slightly different count of enemy troops. In this case, the best solution is probably to average the disparate counts.
- The task is "converge on the target". Each robot's sensors report a slightly different position for the target. In this situation, it appears to make sense that each team member rely on its own sensor values to track the target and only rely on other robots when the robot's own sensors are unable to track the target, e.g. the target is out of sight.

Communication

While the robots are conceptually connected by wires, in actuality, they communicate by RF broadcast. In our current implementation, each robot broadcasts its sensory data and state estimates in a single UDP packet at predefined intervals. In our current implementation, broadcasts are made every second. Faster or slower rates could be used when latency was more or less critical, however, 1Hz has worked well for our applications.

Again, we expect that currently implementable robot systems could store all the sensory inputs to the inference system in a single UDP packet (1024 bytes). As robots develop more complicated sensoria, it may be necessary to use more complicated protocols, perhaps involving multiple packets, or packets that only contain updates for wires whose values have changed since the last transmission. For the moment, however, these issues are moot.

Given the current single-packet-protocol, the aggregate bandwidth required for coordination is bounded by 1KB/robot/sec, or about 0.1% of a current RF LAN per robot. Thus robot teams on the order of 100 robots should be practical from a communication standpoint. However, hardware failure limits most current robot teams to less than 10 members, so scaling limits are difficult to test empirically.

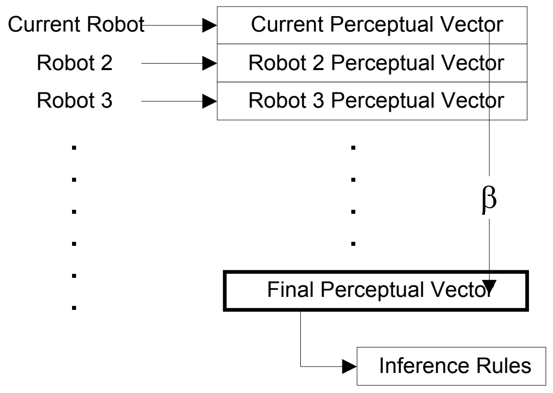
Current Robot
Robot 2
Robot 3
Current Perceptual Vector
Robot 2 Perceptual Vector
Robot 3 Perceptual Vector
β
Final Perceptual Vector
Inference Rules

Figure 3 : Implementation of HIVEMind Wires

Figure 3 shows how aggregation is performed in the actual system. As packets arrive on from other robots, they are unpacked into buffers for their respective robots, replacing whatever data had been stored previously for that robot. In parallel with this process, the main control loop of the robot aggregates the inputs from each robot and reruns the inference rules on the result. These inference rules then enable and disable low-level behaviors for sensory-motor control. Since the main control loop is performing real-time control, it runs much faster than the 1Hz update used for communication (10Hz in our current implementation).

Preliminary Implementation

Overview

As a proof of concept system, we constructed a simple robot team that searches for a brightly colored object in a known environment. A human user is responsible for entering the properties of the desired object to the system. The user console appears as an extra robot to the team.

When the user inputs the color of the desired object, this information is passed automatically to the other robots. Team members then systematically explore the environment until one of the members locates the object or all searchable space is exhausted. When the object is found, all team members converge on its location. We have tested the system with a two-robot team.

Figure 4 : Two robots leaving on their search task

Figure 5 : One member of the team finding the ball

Hardware

The robotic bases used in this experiment are first generation Real World Interface(RWI) Magellan bases. The Magellan provides sonars, infrared sensors and bump switches; a total of 16 each, arrayed around the circular base. Vision is provided by a ProVideo CCD camera, connected by a Nogatech USB video capture adaptor cable to a laptop. The laptops are Dell Latitudes with Pentium II 500Mhz processors, 384Mb of RAM and 11Gb hard drives. They run Windows98, and communicate with the base through a serial cable. Remote communication is provided by LinkSys WPC11 wireless Ethernet cards that feature an 11Mbps data transfer rate under the IEEE 802.11b standard.

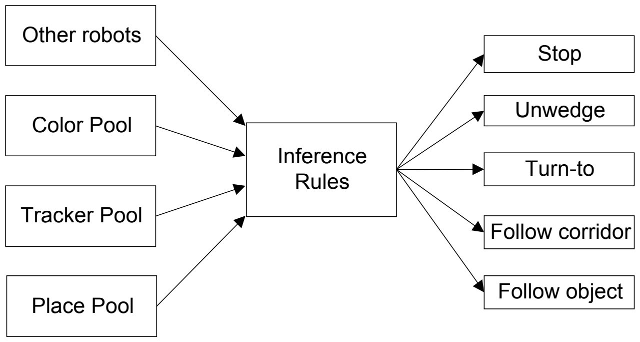

Figure 6 : High-level view of the ball finding system

Perceptual Systems

Sensory and memory systems are divided into *pools*, which can either be perceptual systems or passive information stores, e.g. descriptions of objects. These pools drive the inference network, which in turn drives the low-level behaviors that actually control the robot.

Figure 6 shows a high-level view of the system. The code was written in a combination of GRL (Horswill 99) and Scheme, although low-level vision operators were written in C++. In the following subsections, we will briefly describe each of the pools, the communicated predicate/function values, the inference rules and the control behaviors.

Color Pool The color pool stores color coordinates of different objects in a format suitable for use by the visual tracking system. The color of a desired object can be specified by binding a given color description in the pool to the role of the object. Thus the user, would direct the team to seek a green ball by binding the green color to patient and then asserting the goal near(patient). The binding and goal are then automatically passed over the network to the robots. The color pool presently contains coordinates for red, green, and blue objects.

Tracker Pool The tracker pool consists of a set of color blob trackers that can be allocated, and bound to a role. The trackers can drive low-level behaviors with image-plane coordinate of the objects they track. In addition, they generate the low-level predicates see-object(X) and near-object(X) for input to the inference network.

Place pool The place pool is a probabilistic localization system that uses a topological (i.e. landmark-based) map. Roles can be bound to landmarks and the system can determine the next appropriate waypoint in order to reach a landmark specified by role. The place pool also records the set of landmarks that have been visited with high probability and can determine the closest unvisited landmark. The current map contains 11 landmarks distributed over the west wing of the 3rd floor of the Northwestern Computer Science Department.

Communication

The task can be accomplished by sharing the role bindings for each color, the bit-vector for the goal(near(X)) predicate, the bit-vector for the see-object(X) predicate, a location(X) function, which give the two nearest landmarks, if known, to any role X, and a bit-vector specifying the set of landmarks that the robot has personally visited. All of these are low-level outputs of the various pools, except for the goal predicate which has to be stored in a separate latch on the control console.

Inference rules

The inference rules for this case are fairly simple. This is partly due to the continual recomputation inferences, which alleviates the need for some error detection and recovery logic that would otherwise be necessary. The rules in the current system are:

1. If near(X) is an unsatisfied goal and see-object(X) is true, then approach(X).
2. If near(X) is an unsatisfied goal and location(X) is known, and see-object(X) is false, then goto(location(X)).
3. If near(X) is an unsatisfied goal and location(X) is unknown, then goto(next-unsearched-location).

Behaviors

There are five motor behaviors which are activated by the rules as necessary.

- *Approach-object* drives to an object specified by role. It attempts to keep the object in the middle of its visual image.
- *Follow-corridor* navigates the hallways. It tries to remain centered in the middle of the corridor to facilitate easy recognition of environmental features.
- *Unwedge* activates when the robot becomes stuck in some corner unexpectedly. It swivels the robot in the direction in which it thinks has the greatest open space so the robot can continue moving.
- *Turn-to* swivels the robot to face a new direction. It is used when the robot arrives at a landmark and needs to turn in a new direction to reach another landmark.

Interface Console

The Command Console for the HIVEMind team is based on the Cerebus project (Horswill et al 00). It provides a natural language interface for the human user and allows commands such as “find green ball” or “find red ball” to be entered. The desired color is bound to the *patient* role and transmitted to members to the team. The console appears as another robot to other team member, albeit one that is not doing any physical work. The Command Console also provides status information in the form of display windows based on the broadcast knowledge it is receiving from other team members. Using this interface, the human commander can inject new information into the team, as well as receive data about the current state of the “group mind”.

Related Work

A number of behavior-based architectures for multi-robot systems have been successfully implemented. The Alliance architecture (Parker 98) uses spreading activation to choose between sets of behaviors that achieve different goals. Team members can become impatient if teammate’s task is not accomplished in a timely manner, or alternatively may abandon a goal if it is not making progress. (Goldberg and Mataric 99) present an approach utilizing augmented Markov models to learn probabilistic state transitions for a foraging task. (Balch and Arkin 98) describe a system that maintains military formations such as a line, wedge, diamond or column. Our approach, which behavior-based in spirit, differs from these architectures in its support for explicit inference, limited variable binding, and shared knowledge. It provides a useful set of symbolic operations while still retaining all the advantages of a behavior-based architecture. Although any of these architectures could likely have implemented the search task described here, our expectation is that HIVEMind’s support for explicit goals and variable binding will make it easier to build reusable, taskable systems.

Many simulated multi-robot systems that utilize a strong symbolic approach have also been built. The STEAM architecture (Tambe 96) is a particularly ambitious example based on the Joint Intentions framework of (Cohen & Levesque 91). The system uses a multiphase commit protocol to maintain coherence amongst team members. Although the present HIVEMind broadcasting strategy works well for current systems, it will eventually break down for very large teams or very large knowledge bases. More complicated protocols might fare better.

Conclusion

The HIVEMind architecture is an attempt to extend parallel reactive inference to a multi-robot environment. It allows behavior-based systems to abstract over both objects and sensors. We believe that the right set of representational choices can allow the kinds of inference presently implemented on robots to be cleanly grounded in sensor data and reactively updated by a parallel inference network. If so, then this also provides a clean mechanism for group coordination. By continually sharing perceptual knowledge between robots, coordination can be achieved for little or no additional cost beyond the communication bandwidth required to share the data. The effect is a kind of "group mind" in which robots can treat one another as auxiliary sensors and effectors. As a first step, we have implemented a proof-of-concept system to validate the architecture. The current system finds static objects in a known environment. Our current goal is to extend the system to find and trap evading targets such as other robots. This is an especially interesting task because it requires non-trivial spatial reasoning about containment and visibility.

References

P.E. Agre and D. Chapman(1987) *Pengi : An Implementation of a Theory of Activity*. In Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 268-272. Seattle, Wa.

R.C. Arkin (1998). *Behavior-Based Robotics*. MIT Press. Cambridge, MA.

R.C. Arkin and T.R. Balch(1997) *Aura: principles and practice in review*. Journal of Experimental and Theoretical Artificial Intelligence, 9(2).

T. Balch and R.C. Arkin(1998) *Behavior-based formation control for multirobot teams*, IEEE Transactions on Robotics and Automation, vol. 14, no. 6, pp. 926--939, December 1998.

P. Bonasso, R.J. Firby, E. Gat, and D. Kortenkamp (1997). *Experiences with an Architecture for Intelligent Reactive Agents*. In Journal of Theoretical and Experimental Artificial Intelligence, special issue on software architectures for physical agents, Hexmoor, Horswill and Kortenkamp, eds., 9:2-3. Taylor and Francis, Ltd.

P.R. Cohen and H.J. Levesque (1991) *Teamwork*, Nous 35.

J.H. Connell(1992) S*SS: A hybrid architecture applied to robot navigation*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 92), pages 2719--2724, Nice, France, 1992. IEEE Press, New York, NY.

R. J. Firby, P.N. Propopowicz, and M.J. Swain(1998), *The animate agent architecture*, in Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems, eds., D. Kortenkamp, R.P. Bonasso, and R. Murphy, AAA Press/The MIT Press.

D. Goldberg and M.J. Mataric(1999) *Coordinating Mobile Robot Group Behavior Using a Model of Interaction Dynamics,* Proceedings, The Third International Conference on Autonomous Agents (Agents '99), Seattle, Washington, May 1-5

I. Horswill(1998). *Grounding Mundane Inference in Perception.* In Autonomous Robots, 5, pp. 63-77.

I. Horswill(1999). *Functional programming of behavior-based systems*. In Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation

I. Horswill, R. Zubek, A. Khoo, C. Le, and S. Nicholson(2000) *The Cerebus Project*. In Proceedings of the 2000 AAAI Fall Symposium on Parallel Cognition and Embodied Agents.

L.E. Parker(1998) *ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation*, IEEE Transactions on Robotics and Automation, Vol. 14, No. 2, April 1998.

P. Stone and M. Veloso (1999) *Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork* Artificial Intelligence (AIJ), volume 100, number 2, June 1999.

M. Tambe(1996) *Teamwork in real-world, dynamic environments.* In Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96), Menlo Park, California, December 1996. AAAI Press.