An Efficient Coordination Architecture for Autonomous Robot Teams

Aaron Khoo and Ian Douglas Horswill

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

Abstract

Most physically implemented multi-robot controllers are based on extensions of behavior-based systems. While efficient, such techniques suffer from weak representational power. Symbolic systems, on the other hand, have more sophisticated representations but are computationally complex and difficult to keep synchronized with changes in the environment. 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

More hands obviously make light work. For this reason, multi-robot control systems have been a hotbed of research activity. However, the continually changing world is ultimately a cruel place for robots. It is difficult enough to keep one robot's world model synchronized with the outside world; synchronizing the shared situational awareness of an entire team of robots is an even more difficult issue.

Behavior-based systems

Many physically implemented multi-robot systems focus on extending traditional behavior-based techniques (Arkin 98) to a team setting. Some examples :

- (Balch and Arkin 98) describe a schema system that maintains military formations such as a line, wedge, diamond or column.
- The Alliance architecture (Parker 98) uses spreading activation to choose between sets of behaviors that achieve different goals
- (Goldberg and Mataric 00) presents several behavior-based multi-robot controllers used in a collection task.

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 networks of simple finite-state machines. This allows them to completely recompute sensor, model, and control decisions from moment to moment, which 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, that is, representations without predicate/argument structure. Propositional representations make reasoning and planning more difficult and clumsy since they require redundant copies of the system for each possible argument to a predicate or action (Maes 90). Since most multi-robot controllers are extensions of behavior-based techniques, they inherit the same issues from the basic underlying architecture.

Symbolic Techniques

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 process is sufficiently complicated that we

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.

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 symbolic system to update specific aspects of its 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 97, Firby *et al.* 98, Connell 92) can mitigate the computational complexity issues 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 both with the world and with one another. Failure to properly coordinate the knowledge bases will ultimately result in failure to coordinate activity. Excellent work has been done on communication protocols for insuring consistency (Cohen and Levesque 91, Tambe 96). However, it is telling that physically implemented multi-robot systems almost always use purely behavior-based architectures and/or shared, centrally controlled world models created using an overhead camera (Stone and Veloso 99).

HIVEMind

HIVEMind (Highly Interconnected Verbose Mind), is a multi-robot control architecture that supports very efficient sharing of symbolic information between team members. Based on role-passing (Horswill 98), a variant of deictic representation (Agre and Chapman 87), HIVEMind can efficiently implement quantified inference over unary predicates and provide hard real-time guarantees for synchronization of knowledge bases between team members and the world. Objects in a team member's working memory are bound to a small, fixed set of roles such as *agent, patient, source, destination*, etc. When a role is bound to an object, a tracker is dynamically allocated to it and tagged with the name of the role. Since the number of roles is relatively small, we can represent the extensions of unary predicates as bit-vectors, one bit per role.

This representation allows inference to be performed using bit-parallel operations in a feed-forward network. Alternatively, for commodity serial hardware, we can represent a unary predicate extension using a single machine word. Inference rules can then be compiled into directly into straight-line machine code consisting only of load, store, and bit-mask instructions (Horswill 98). While more limited than a full logic-programming system, it does allow us to express much of the kinds of inference used on physical robots today. The inference rules can be completely rerun on every cycle of the system's control loop, allowing the robots to respond to contingencies as soon as they are sensed. The compiled code is sufficiently efficient that inference is effectively free – 1000 Horn rules of 5 conjuncts each can be completely updated at 100Hz using less than 1% of a current CPU.

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 a single machine word. Function values are represented using small arrays indexed by role. For the kinds of tasks currently implemented by multi-robot teams, this representation is sufficiently compact to allow all function and predicate values of a robot to fit into a single UDP packet. Robots can therefore share information by periodically broadcasting *the entire knowledge base*, or at least all those predicates and functions that might be relevant to other team members.

Knowledge-base broadcast is a simple communication and coordination model that provides each robot with transparent access to every other robot's state, a kind of "group mind". It 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 are immediately informed and respond in O(1) time without the need for negotiation protocols. Moreover, since HIVEMind systems are based on role-passing, multi-robot controllers implemented using this architecture have greater representational power and flexibility than behavior-based systems with propositional representations.

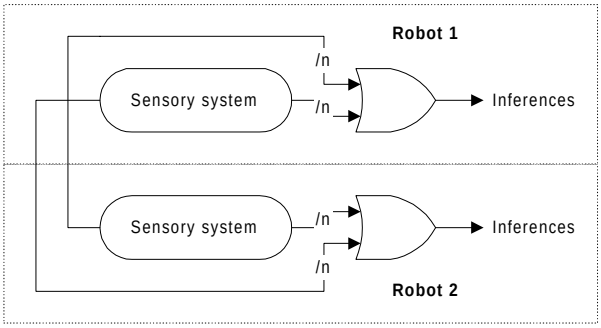

Figure 1 : Abstract view of HIVEMind

Figure 1 shows a HIVEMind configuration for a two-robot team. Each team member has its own inference network. The network 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 multi-phase commit protocols used in distributed database systems. Since communication bandwidth is a scarce resource and inference in our system is essentially free, it is redundant computation is more efficient than additional communication.

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,

- If the team is assigned to scout an area and report the number of enemies observed, each team member is likely to report a slightly different troop strength., so averaging the different reports might make sense.
- On the other hand, if the team is to converge on a target, each robot will have a slightly different estimate of both its position and the target's position. Rather than averaging, each robot should then trust its own sensors unless 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. Presently, broadcasts are made at 1Hz. Slower rates could be used when bandwidth or stealth are more critical.

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.

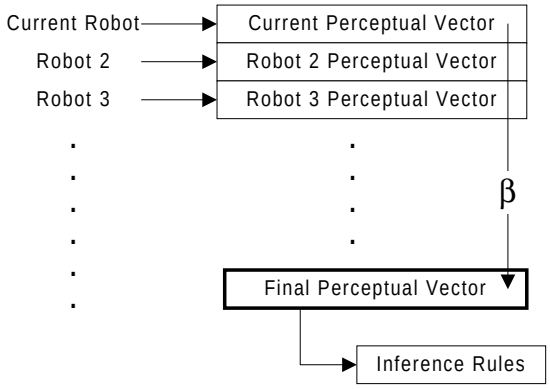

Figure 2 : Implementation of HIVEMind Wires

Figure 2 shows how aggregation is performed in the actual system. As packets arrive 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 most recent inputs from each robot and reruns the inference and control rules on the result. These rules 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 performs a systematic search of a known environment for an object. 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.

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.

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.

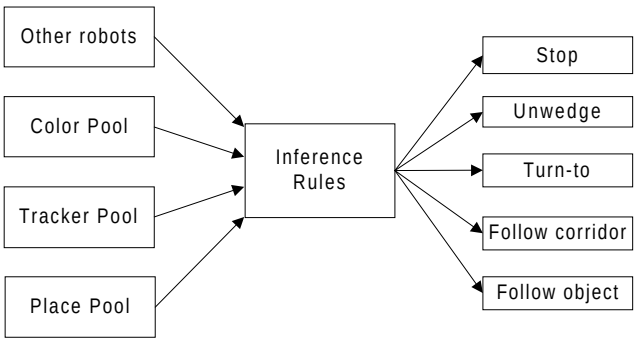

Figure 3 : High-level view of the ball finding system

Figure 3 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.

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 `goal(near(patient))`. The binding and goal are then automatically passed over the network to the robots. The color pool presently contains descriptions 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 Computer Science Department building.

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 is effectively a table of the two nearest landmarks, if known, for each 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 (see below).

Inference rules

The inference rules for this task are simple, in part because the continual recomputation of inferences alleviates the need for some error detection and recovery logic that would otherwise be necessary. The rules in the current system are:

For all objects `X`:

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 rudimentary natural language interface for the human user and allows commands such as "find green ball," "find red ball," or "patrol corridor," to be entered. The console appears as another robot to other team member, albeit one that is not doing any physical work. It reads commands, parses them, and binds their component nouns and verbs (e.g. the color of the object to search for) to the appropriate roles (e.g. *patient*) and begins broadcasting them to the rest of the team. It displays the shared knowledge base broadcast by the team in windows on the screen to provide status information to the human commander. 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".

Results

We have tested the system with a three-member team consisting of two robots and the command console. The team was tested in the west wing of the 3rd floor of the Computer Science Department building. The wing consists of a network of six corridors spanning an area approximately 6m×20m with an aggregate path length of 50m. The wing is represented by 12 landmarks in the topological map showing the locations of features such as corners and intersections. The robots drive at approximately 1m/s on straightaways, although stopping for ballistic turns at corners and intersections somewhat reduces their mean velocity. Sensing, inference and control decisions are each performed at 10Hz.

In the experiments, all team members were started from a central point at the extreme East end of the wing. The goal object, a green ball, was placed out of view, 15-20m from the starting point. The object was always at least two corridors and three landmarks away from the starting point. When the command "find green" was entered on the command console, the robots begin a systematic search of the wing for the goal object. Unlike stochastic search techniques such as foraging, the systematic search guarantees that each landmark is searched at most once and that all landmarks are guaranteed to be searched, if necessary. Using a greedy algorithm for landmark selection, the team was consistently able to find the landmark within 30 seconds provided that there were no catastrophic failures of the place recognition system. On typical runs, the team found the object in approximately 20 seconds.

Figure 4 : Two robots leaving on their search task

Figure 5 : One member of the team finding the ball

The place recognition system is the weak point of the current implementation. Minor errors are common and occasional catastrophic failures can cause one of the team members to think that it has searched the true location of the goal when in fact it has not. While we are working on improving the place recognition system, it should be stressed that the actual control and coordination architecture worked without error.

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, while remaining efficient enough in both inference speed and bandwidth consumption to be usable on physical robotic teams. HIVEMind provides multi-robot system designers with more powerful representations than behavior-based systems, and has a simple, efficient model for group coordination. 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. 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(2000) *Robust Behavior-Based Control for Distributed Multi-Robot Collection Tasks*, USC Institute for Robotics and Intelligent Systems Technical Report IRIS-00-387

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.

Maes, P. 1990. "Situated Agents Can Have Goals," *Robotics and Autonomous Systems*, Vol. 6, pp. 49-70.

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.