Grounding Inference in Distributed Multi-Robot Environments

Aaron Khoo and Ian Horswill

Computer Science Department, Northwestern University
1890 Maple Avenue
Evanston, IL 60201
{khoo, ian}@cs.northwestern.edu
Fax : (847) 491-5258

Keywords : Autonomous Mobile Robots, Multi-Robot Teams , Behavior-based control

Abstract

Systems embedded in a dynamic environment face the problem of grounding the inference used in their reasoning system to actual physical objects. Traditional symbolic reasoning systems are typically built on a transaction model of computation, which complicates the process of synchronizing their world models with changes in the environment. While some progress has been made grounding inference in tiered architectures for the single robot case, physical multi-robot systems invariably utilize purely behavior-based control techniques. We believe this is due to the complexities of synchronizing multiple distributed knowledge databases located on wireless platforms in real time. In this paper, we describe an inference grounding and coordination mechanism for small cooperative robot teams based on an extension of tagged behavior-based systems. Tagged behavior-based systems support a large subset of classical AI architectures while allowing object representations to remain distributed across multiple sensory and representational modalities. They provide a novel representation based on bit-vectors that allow team members to share intentional, attentional and sensory information using relatively low-bandwidth connections. We illustrate our approach on two problems involving systematic spatial search.

Introduction

Autonomous robots that reside in a complex, dynamic environment face the issue of anchoring the abstract representations they use to actual physical objects. The world around the robot continually changes, and its sensory systems must track those changes. In turn, 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.

Traditional symbolic reasoning systems are typically built on a transaction-oriented model of computation. Knowledge about the world, or the “world model”, is stored in a database of assertions in some logical language, indexed perhaps by predicate name [21]. Populating this database from a dynamic environment is a non-trivial issue; see [14] for an exposition on the difficulties involved. Changes in the environment occur often, so the database must also be updated fairly frequently, or risk the reasoning system operating on stale data. Additionally, assertions in the database can be dependent on other assertions. For example, the assertion that an area is safe could depend on the assertion that the robot does not currently observe any predators in the area. If the latter assertion is withdrawn, then the former must be too. Hence, each update from the perceptual systems can trigger a cascade of further transactions, resulting in additional load on the knowledgebase subsystem.

Although some recent work has been done towards the development of a formal framework for anchoring [8,9], most implemented physical systems instead equip the symbolic system with a set of domain-dependent epistemic actions that fire task-specific perceptual operators to update specific parts of the knowledgebase. The programmer designing the knowledgebase is responsible for ensuring that the proper updates are done, i.e. the right epistemic actions are fired at the appropriate times. This alleviates some of the difficulties of getting information into the knowledgebase in a timely manner. However, any mistakes by the programmer will lead to inconsistencies between the knowledgebase used by the symbolic system and the external environment. Tiered architectures, such as [1,5,7], that combine symbolic and behavior-based systems inherit these model coherency issues, because their symbolic layer still relies on a database-driven world model for its reasoning process.

Keeping the knowledgebase synchronized with the external environment becomes even more difficult in cooperative activity. In this analysis we assume a team model where members are fully autonomous entities with independent decision making ability. Furthermore, team members may lose sight of each other during execution, so coordination has to be achieved through explicit communication. Under these constraints, rather than one robot with a single knowledgebase, we now have *n* robots with *n* knowledgebases to keep consistent both with the world and with one another. Failure to properly coordinate the knowledgebases will ultimately lead to *system delusion* [13], i.e. the databases are now inconsistent, and there is no obvious way to repair them, resulting in failure to coordinate activity. There has been some excellent work done on coordination protocols for cooperative agents [6,22]. However, an analysis of asynchronous peer-to-peer replicated databases by Gray et al. [12] suggests a potential problem :

A *conflict* occurs when two different databases attempt to update the same object, or race to install their updates at other databases. Whenever conflicts occur, the replication mechanism must detect this and somehow reconcile the two transactions so that their updates are not lost. Under the following simple assumptions --

- The databases are updated through lazy group replication, i.e. the originating database updates its entries, and then propagates the update to other replicas asynchronously.
- Each node updates any other database location with equal probability
- All nodes impose an equal load on the system
- There are a fixed number of objects per transaction

Gray et al. were able to show that the conflict rate per second is approximately :

$$\frac{TPS^2 * Actions^3 * Action\_time * nodes^3}{2 * DB\_Size}$$

where *TPS* is the number of transactions per second initiated by each node, *Actions* is the number of locations updated per transaction, *Action_time* is the time required to complete an update, *DB_size* is the number of distinct entries in the database and *nodes* is the number of nodes (which, in our case, is equivalent to the number of robots) in the system. The critical point here is that the number of conflicts encountered by the system increases with the third power of the number of nodes or robots. As Gray et al. point out, “having the reconciliation rate rise by a factor of a thousand when the system scales up by a factor of ten is frightening”. While the two models are not exactly analogous, there is sufficient overlap between the problem of synchronizing different knowledgebases and the issue of distributed database replication to elicit concern.

Furthermore, note that message propagation times are not presently part of the conflict model as presented above. If message delays were added to the model, then each transaction would last longer, hold more resources and generate more conflicts. Moreover, mobile robots necessarily communicate via wireless links, which are well-known to have higher error rates [10,24], and hence higher message delays, than their wired counterparts. This analysis suggests that we could potentially face serious scalability issues for any physical multi-robot system with a database-driven knowledge model. The work necessary to reconcile the conflicts that could arise as team members tried to communicate knowledge to other members could eventually overwhelm the robots, or leave them badly out of synch.

We feel that these knowledgebase synchronization issues has led to a paucity of physical multi-robot systems utilizing symbolic reasoners. Instead, most existing multi-robot controllers implemented on physical systems focus on extending traditional behavior-based techniques [1] to a team environment (For example see [4,11,20]). Traditional behavior-based systems obey circuit semantics [19], which means their control programs are generally implemented as feed-forward circuits. This allows rapid response to changes in the environment due to tight sensor-actuator integration, and also simplifies the communication structure necessary to maintain coordination between team members. Essentially communication in behavior-based multi-robot controllers is reduced to virtual wires connecting the appropriate circuitry on one team member to another’s (see figure 1). The wires carry relevant information from a robot to its counterparts. Conversely, each robot views its teammates simply as additional sensory input, and integrates the incoming information as appropriate. Conveniently, virtual wires can be simulated on physical robots using a broadcast communication mechanism such as UDP.

However, this convenience is not without cost. The strengths of the behavior-based approach are also its weakness. Circuit semantics impose a propositional representation on the reasoning system, i.e. representations without predicate/argument structure. Propositional representation makes most reasoning and planning tasks both difficult and clumsy since they require redundant copies of the system for each possible argument to a predicate or action [18]. Since most multi-robot controllers are extensions of behavior-based techniques, they inherit the same issues from the basic underlying architecture.

We should point out that there are some multi-robot systems [23] that utilize a traditional symbolic reasoner by relying on a shared, centrally controller world model. In this case, reasoning is performed at a central server location where the master knowledgebase is located, and then actions are transmitted to the individual robots. Little, if any, reasoning is done on the client side. In this paper, however, we are only considering multi-robot systems where the team members are independent of a central reasoner, i.e. each robot maintains its own reasoning system.

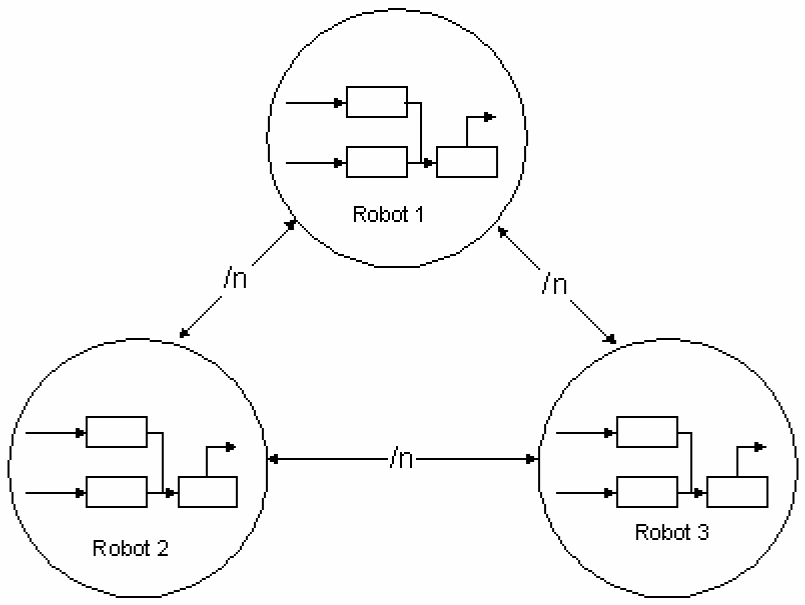

Figure 1 : Communication via Virtual Wires

HIVEMind

While designing our current multi-robot control architecture, we wanted to utilize as many useful features of traditional symbolic AI systems as possible on our robots. Specifically, we would like to have the ability to utilize predicate/argument structure in our representations. However, we also wanted to avoid importing the model coherence and database synchronization issues that symbolic systems encounter. That is, the symbols utilized in our inference rules should be tightly anchored to updates from the sensory systems as well as incoming information from other team members.

Our efforts in this direction have resulted in HIVEMind (Highly Interconnected Verbose Mind), a multi-robot control architecture that supports very efficient sharing of symbolic information between team members. The HIVEMind architecture is built on role-passing [14], a type of tagged behavior-based system [17]. Role-passing provides the developer with a limited set of domain-independent indexical variables (called 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, with one bit representing each 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 directly into straight-line machine code consisting only of load, store, and bit-mask instructions [14].

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 clauses of 5 conjuncts each can be completely updated at 100Hz using less than 1% of a current CPU. In short, role-passing affords us the ability to implement traditional inference rules using circuit semantics.

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. Function values are represented using small arrays indexed by role. This compactness, combined with the circuit semantic nature of role-passing, allows us to take full advantage of simplified communication mechanism described previously, i.e. virtual wires connecting team members. In fact, for the kinds of tasks currently implemented by multi-robot teams, the representation we use 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 their entire knowledge base, or at least all those predicates and functions that might be relevant to other team members.

Knowledge-based broadcast is a simple communication and coordination model that provides each robot with transparent access to every other robot's state, establishing 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 pure behavior-based systems with propositional representations. That is, our communication is not based on passing propositional values such as `see-blue-object` or `see-red-object`, but rather predicates such as `see-object(X).` Furthermore, since all relevant team knowledge is continuously being rebroadcast, each member's knowledgebase converges to the same state within O(1) time of joining the Hivemind. This means that team members can be brought online and integrated into the Hivemind very easily, allowing us to add or subtract team members dynamically. This also implies that, should communication fail for some time, the team would very rapidly return to a common state when it is restored.

Figure 2 shows an abstract 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 incoming data from the other team members. Outputs from the current robot's sensory systems and inference rules are fed into aggregation functions on other team members. The output from those aggregation functions is then fed into the inference rules which drive the motor behaviors.

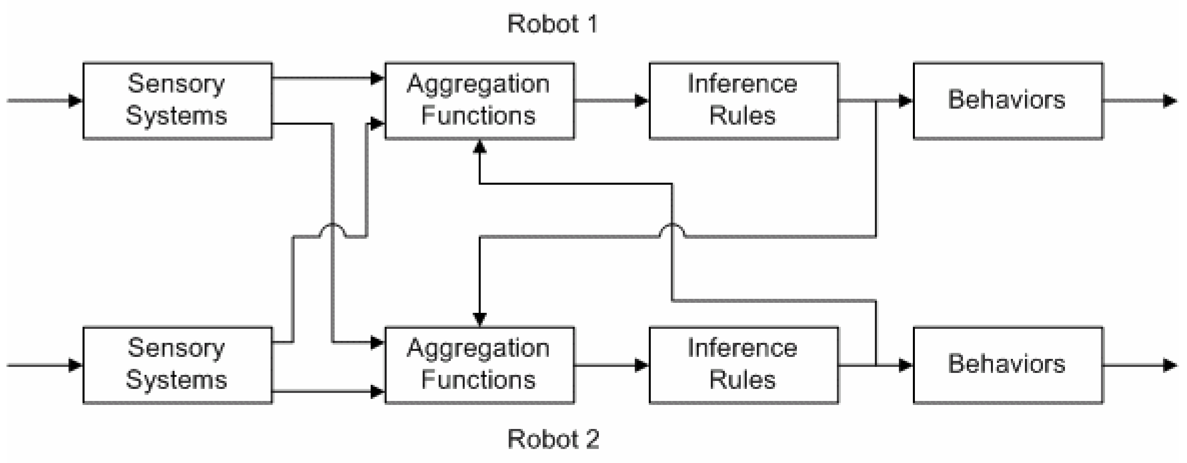

Figure 2 : Abstract View of HIVEMind

The aggregation functions are used to combine information from teammates and sensors into a single coherent output for the inference rules to reason over. 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 if and only if 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.

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).

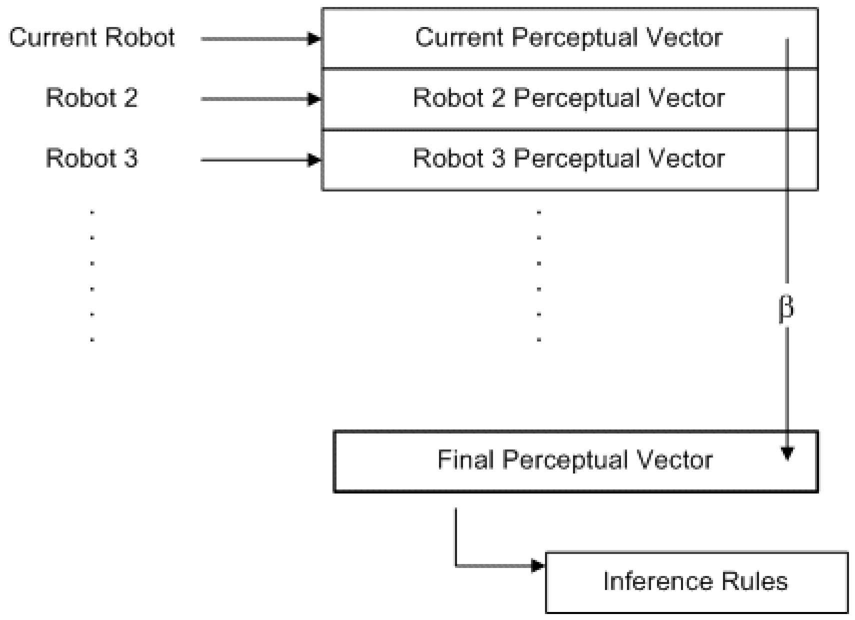
Current Robot
Robot 2
Robot 3
Current Perceptual Vector
Robot 2 Perceptual Vector
Robot 3 Perceptual Vector
β
Final Perceptual Vector
Inference Rules

Figure 3 : Implementation of HIVEMind Virtual Wires

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. 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 every second. Faster or slower rates could be used when latency is more or less critical. However, 1Hz has worked well for our applications. To reiterate, 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.

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 [6] 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 more efficient for HIVEMind robots to perform redundant computation.

Implementation

Overview

We have implemented the HIVEMind system on a robot team that performs two tasks :

- Find Object
  The team systematically searches for a brightly colored object in a known environment. Team members explore the environment in a systematic manner until one of the team members locates the object or all searchable space is exhausted. When the object is found, all team members converge on its location.
- Town Crier
  This task involves making announcements in the same known environment. The team cooperatively travels to each landmark on a map and makes an announcement at every landmark.

In both cases a human user is responsible for indicating the current task to perform and supplying any required parameters for that task, e.g. the properties of the object to be found in the former task. The human interacts with the team through a user console, which appears as an additional, albeit non-performing, member of the team. When user input is entered into the console, that information is passed through the virtual wires to all team members. We have tested both tasks with a two robot team. The code for this system was written in a combination of GRL [15] and Scheme, although low-level vision operators were written in C++.

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 Lucent Orinoco Silver wireless Ethernet cards that feature an 11Mbps data transfer rate under the IEEE 802.11b standard.

User Console

The Command Console for the HIVEMind team is based on the Cerebus project [16]. It provides a natural language interface for the human user and allows commands such as ‘find green ball’ or ‘announce “talk at 7!”’ to be entered. The task is bound to the `activity` role, and any arguments are bound to other appropriate roles, e.g. green would be bound to `object` in the former example. The current bindings are represented in alist form and transmitted on a virtual wire to all members to the team. The console appears as another robot to other team members, albeit one that is not doing any physical work. The user 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".

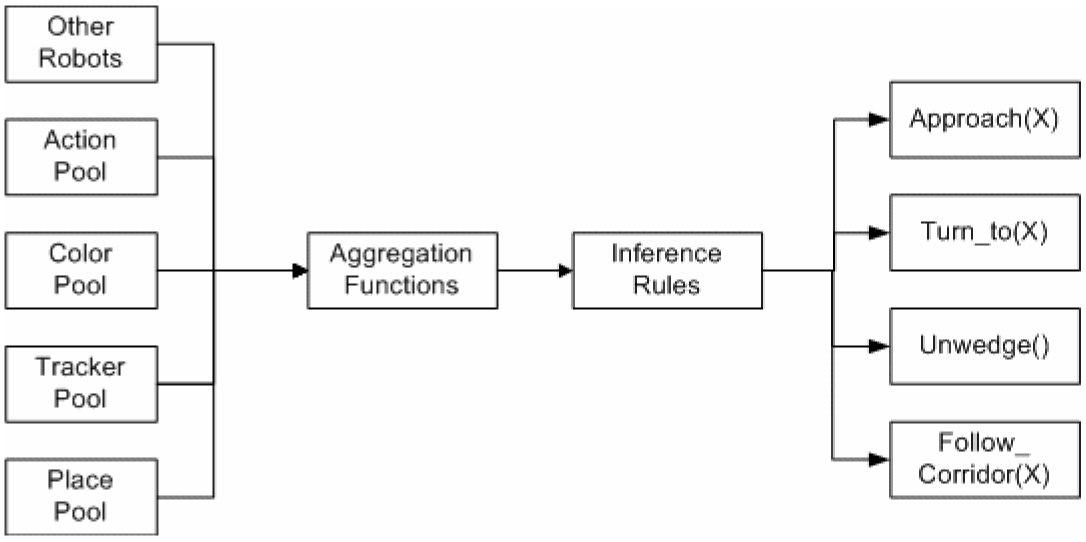

Figure 4 : Control flow from sensors to behaviors in a single robot

Perceptual Systems

The sensory and memory systems are divided into "pools", which are useful abstractions for grouping perceptual systems or descriptions of objects. Note that we do not make any unique claims about pools; they are simply convenient abstractions for implementing role-passing. The pools drive the inference rule network, which in turn drives the low-level behaviors that actually control the robot. Figure 4 shows a high-level view of the system. The action pool stores a set of reified user-defined plans that can be bound to roles at runtime. These plans can then be run by calling the role to which they are bound. For example, the `find` plan is bound to the role `activity` when the user enters "find green ball" at the console. The binding is passed via virtual wire to the individual team members. So, when the control system calls `activity`, it would run the `find` plan. There are currently two plans in the action pool : `find` and `announce`.

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. For example, when the user directs the team to seek a green ball, the term `green` is bound to an appropriate role. The bindings are then automatically passed over the network to the robots. The color pool presently contains descriptions for red, green, and blue objects, and is only used for the find object task.

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. The trackers are used only in the find object task.

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

Both tasks require communication of the following :

- The current role bindings, including bindings for the current activity or task, and any bindings for pertinent arguments
- A bit-vector specifying the set of landmarks that the robot has personally visited
- The bit vector for the `see-object(X)` predicate
- An array representing the `location(X)` function, which give the two nearest landmarks, if known, for any role X

All of these are low-level outputs of the various pools, except for the current role bindings, which has to be stored on a separate latch on the user console. When the team is performing the town-crier task, the latter two communication structures, i.e. `see-object(X)` and `location(X)`, are not utilized for reasoning.

Inference Rules

The inference rules for both taks are fairly simple. This is partly due to the continual recomputation of inferences, which alleviates the need for some error detection and recovery logic that would otherwise be necessary. The inference rules for the find object task are :

1. If `see-object(X)` is true, then `goto(X).`
2. If `location(X)` is known, and `see-object(X)` is false, then `goto(location(X)).`
3. If `location(X)` is unknown, and `see-object(X)` is false, then `goto(next-unsearched-location()).`

The inference rules for the town-crier task are :

1. If `at-landmark(X)` and `not-announced-at(X),` then `speak-string().`
2. If true, then `goto(next-unsearched-location()).`

The function `next-unsearched-location()` returns the current location if there are no new locations to travel to. `Goto()` is a polymorphic action keyed by the type of the argument passed to it. If the argument is bound to a location, then the robot will navigate to that landmark. If the argument is bound to a color in the color pool, then the robot approaches the largest object matching that color in its view. `Goto()` activates the four behaviors described below as necessary to accomplish its current task.

Behaviors

There are four motor behaviors that drive the robot :

- *Approach* drives to an object specified by role. It attempts to keep the object in the middle of its visual image.
- *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.
- *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.
- *Follow-corridor* navigates the hallways. It tries to remain centered in the middle of the corridor to facilitate easy recognition of environmental features.

The behaviors are arbitrated strictly through a priority stack. Behaviors that are higher on the stack have higher priority, and, if active, will be chosen to run over those of lower priority. Since HIVEMind always ensures that all team members are up-to-date on the current situation, each robot always knows the appropriate behavior to activate for the current situation and no conflict between team members arises.

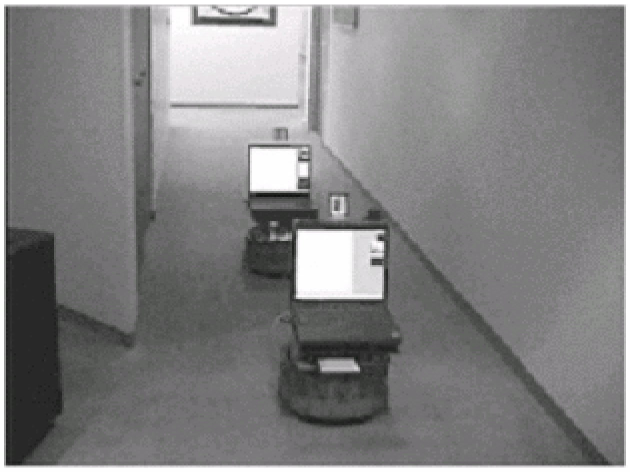
Figure 5 : Two robots leaving from start point to perform a task

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 6mx20m with an aggregate path length of 50m. The network of corridors 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.

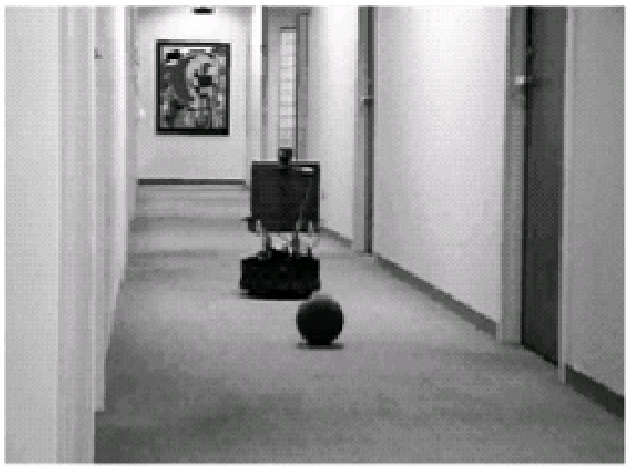

Figure 6 : One member of the team locating the target in the find object task

In the find object 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.

For the town-crier task, team members were again started from a central point at the extreme east end of the wing. The objective was for the robots to go through each landmark at least once, making the announcement at each landmark that the robots passed through. If a robot had already spoken at a particular landmark, then no further announcement should be made there, since we do not wish to inundate any nearby offices with multiple announcements. Again, barring any catastrophic failures of the place recognition system, the team was able to complete the task successfully.

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 traversed its intended destination 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.

Conclusions

Grounding inference is a complex but unavoidable issue for systems embodied physically. Traditional symbolic reasoning systems face the issue of maintaining a world model that is coherent with the dynamic world. This issue is exacerbated in multi-robot systems, as we now have *n* knowledgebases to synchronize with each other as well as the external environment. The multi-robot case shares some similarity to the problem of replicating *n* distributed databases; a problem which others have shown to very challenging, since the number of conflicts during attempted updates rises with the third power of the number of participating robots. We offer an alternative architecture that supports the useful features of a traditional symbolic reasoning system, in particular the ability to utilize predicate/argument structure, while avoiding the model coherence and database synchronization issues that traditional symbolic and tiered systems encounter.

The HIVEMind architecture allows behavior-based systems to abstract over both objects and sensors, while providing an anchoring approach that is efficient enough in both inference speed and bandwidth consumption to be usable on physical robotic teams. It presents multi-robot system designers with more powerful representations than behavior-based systems, and has a simple, efficient model for group coordination that consumes very little bandwidth while allowing team members to react to opportunities or contingencies within O(1) time. 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. We have currently implemented two tasks utilizing HIVEMind : one that finds object, and another that makes announcements. Our current goal is to implement a system which finds and traps evading targets such as other robots. This is an especially interesting task because it requires non-trivial spatial reasoning about containment and visibility.

References

1. P.E. Agre and D. Chapman. *Pengi : An Implementation of a Theory of Activity*. In Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 268-272. Seattle, Wa., 1987
2. R.C. Arkin. *Behavior-Based Robotics*. MIT Press. Cambridge, MA. 1998
3. R.C. Arkin and T.R. Balch. *Aura: principles and practice in review*. Journal of Experimental and Theoretical Artificial Intelligence, 9(2), 1997
4. T. Balch and R.C. Arkin. *Behavior-based formation control for multirobot teams*, IEEE Transactions on Robotics and Automation, vol. 14, no. 6, pp. 926--939, December 1998.
5. P. Bonasso, R.J. Firby, E. Gat, and D. Kortenkamp. *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, 1997. Taylor and Francis, Ltd.
6. P.R. Cohen and H.J. Levesque. *Teamwork*, Nous, Special Issue on Cognitive Science and AI, 25, 4, 1991, pp. 487-512

7. J.H. Connell. 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.
8. S. Coradeschi and A. Saffiotti. *Anchoring Symbols to Sensor Data: preliminary report*, Proceedings of the 17th AAAI Conference, pp. 129-135. Austin, Texas, July 2000.
9. S. Coradeschi and A. Saffiotti. *Perceptual Anchoring of Symbols for Action*. Proceedings of the 17th IJCAI Conference, pp. 407-412. Seattle, WA, 2001.
10. D. Eckhardt, P. Steenkiste. *Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network*. In Proceedings of the ACM SIGCOMM '96, pp. 243-254, August 1996.
11. D. Goldberg and M.J. Mataric. *Robust Behavior-Based Control for Distributed Multi-Robot Collection Tasks*, USC Institute for Robotics and Intelligent Systems Technical Report IRIS-00-387, 2000
12. J. Gray, P. Helland, P. O'Neil and D. Shasha. *The Dangers of Replication and a Solution*, Sigmod, 1996
13. J. Gray and A. Reuter. *Transaction Processing : Concepts and Techniques*, Morgan Kaufmann, San Francisco, 1993.
14. I. Horswill. *Grounding Mundane Inference in Perception.* In Autonomous Robots, 5, pp. 63-77, 1998.
15. I. Horswill. *Functional programming of behavior-based systems*. In Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation 1999.
16. I. Horswill, R. Zubek, A. Khoo, C. Le, and S. Nicholson(2000) *The Cerebus Project*. In the AAAI Fall Symposium on Parallel Cognition and Embodied Agents, 2000
17. I. Horswill. *Tagged Behavior-based Architectures: Integrating Cognition with Embodied Activity,* IEEE Intelligent Systems, September/October 2001, pp 30-38. IEEE Computer Society, NY.
18. P. Maes. *Situated Agents Can Have Goals*. Robotics and Autonomous Systems, Vol. 6, pp. 49-70.
19. N. Nilsson *Toward Agent Programs with Circuit Semantics*, Technical Report STAN-CS-92-1412, Stanford University Computer Science Department, 1992.
20. L.E. Parker *ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation*, IEEE Transactions on Robotics and Automation, Vol. 14, No. 2, April 1998.
21. S. Russell, and P. Norvig. *Artificial Intelligence : A Modern Approach.* Prentice Hall, 1995.
22. M. Tambe. *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.
23. M. Veloso, M. Bowling, S. Achim, K. Han, and P. Stone. *The CMUnited-98 Champion Small Robot Team.* "RoboCup-98: Robot Soccer World Cup II," M. Asada and H. Kitano (eds.), 1999. Springer Verlag, Berlin.
24. G. Xylomenos and G.C. Polyzos. *Internet Protocol Performance over Networks with Wireless Links*. IEEE Network 13**,** 1999. pp. 55–63