A LAYERED ARCHITECTURE FOR COORDINATION OF MOBILE ROBOTS

Reid Simmons, Trey Smith, M. Bernardine Dias,
Dani Goldberg, David Hershberger, Anthony Stentz, Robert Zlot
Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213
{reids,trey,mbdias,danig,hersh,axs,robz}@ri.cmu.edu

Abstract This paper presents an architecture that enables multiple robots to explicitly coordinate actions at multiple levels of abstraction. In particular, we are developing an extension to the traditional three-layered robot architecture that enables robots to interact directly at each layer – at the behavioral level, the robots create distributed control loops; at the executive level, they synchronize task execution; at the planning level, they use market-based techniques to assign tasks, form teams, and allocate resources. We illustrate these ideas through applications in multi-robot assembly, multi-robot deployment, and multi-robot mapping.

Keywords: Multi-robot coordination, robot architecture, task-level control.

1. Introduction

An architecture for multi-robot coordination must be able to accommodate issues of synchronization and cooperation under a wide range of conditions and at various levels of granularity and timescales. At the concrete, physical level of sensors and actuators, the robots need to respond quickly to dynamic events (such as imminent collisions), while at the same time reasoning about and executing long-term strategies for achieving goals. Executing such strategies is likely to involve establishing and managing a variety of synchronization constraints between robots. For some tasks, such as distributed search, coordination may be loose and asynchronous; for others, such as cooperative manipulation of a heavy object, coordination must be tightly synchronized.

In designing a multi-robot architecture that allows flexibility in synchronization and granularity, one must inevitably negotiate the tension between centralized and distributed approaches. A centralized system can make optimal decisions about subtle issues involving many robots and many tasks. In contrast, a highly distributed system can quickly respond to problems involving one, or a few, robots and is more robust to point failures.

We are developing a multi-robot coordination architecture that addresses these issues, providing flexibility in granularity and synchronization, while also attempting to accommodate the strengths of both distributed and centralized approaches. The architecture is an extension of the traditional three-layered approach, which provides event handling at different levels of abstraction through the use of behavioral, executive, and planning layers. Our approach extends the architecture to multiple robots by allowing robots to interact directly at each layer (see Figure 1). This provides several benefits, including (1) plans can be constructed and shared between multiple robots, using a market-based approach, providing for various degrees of optimization; (2) executive-level, inter-robot synchronization constraints can be established and maintained explicitly; and (3) distributed behavior-level feedback loops can be established to provide for both loosely- and closely-coupled coordination.

Each layer is implemented using representations and algorithms that are tuned to the granularity, speed, and types of interactions typically encountered at each level (symbolic/global, hybrid/reactive, numeric/reflexive). The strengths of this architecture are its flexibility in establishing interactions between robots at different levels and its ability to handle tasks of varying degrees of complexity while maintaining reactivity to changes and uncertainty in the environment. This paper presents the major components of the architecture, together with case studies that illustrate their use in multi-robot applications.

2. Related Work

Our approach blends the advantages of both the centralized and distributed approaches to multi-robot systems. In the centralized approach, a centralized planner plans out detailed actions for each robot. For example, a planner might treat two 6 DOF arms as a single 12 DOF system for the purpose of generating detailed trajectories that enable the arms to work together in moving some object, without bumping into each other (Khatib, 1995). While this approach provides for close coordination, it does so at the expense of local robot autonomy. In particular, this approach usually employs centralized monitoring and, if anything goes wrong, the planner is invoked to replan everything. Thus, this approach suffers from single point failure and lack of local reactivity.

At the other end of the spectrum, in the distributed approach (Arkin, 1992; Balch and Arkin, 1994; Mataric, 1992; Parker, 1998) each agent is autonomous, but there is usually no explicit synchronization among the robots. Coordination (or, more accurately, cooperation) occurs fortuitously, depending on how the behaviors of the robots interact with the environment. For instance, in the

ALLIANCE architecture (Parker, 1998), robots decide which tasks to perform in a behavior-based fashion: They have "motivations" that rise and fall as they notice that tasks are available or not. While ALLIANCE can handle heterogeneous robots (robots can have different motivations for different tasks), it does not deal with the problem of explicit coordination.

(Jennings and Kirkwood-Watts, 1998) have developed a distributed executive for multi-robot coordination. The executive, based on a distributed dialect of Scheme, is similar to our executive language in the types of synchronization constructs it supports. As with our work, this enables robots to solve local coordination problems without having to invoke a high-level planner.

Several researchers have investigated economy-based architectures applied to multi-agents systems (Sandholm and Lesser, 1995; Sycara and Zeng, 1996; Wellman and Wurman, 1998), beginning with work on the Contract Net (Smith, 1980). (Golfarelli et al., 1997) proposed a negotiation protocol for multi-robot coordination that restricted negotiations to task-swaps. (Stentz and Dias, 1999) proposed a more capable market-based approach that aims to opportunistically introduce pockets of centralized planning into a distributed system, thereby exploiting the desirable properties of both distributed and centralized approaches. (Thayer et al., 2000; Gerkey and Matarić, 2001; Zlot et al., 2002) have since presented market-based multi-robot coordination results.

3. Approach

Our multi-robot architecture is based on the layered approach that has been adopted for many single-agent autonomous systems (Bonasso et al., 1997; Muscettola et al., 1998; Simmons et al., 1997). These architectures typically consist of a planning layer that decides how to achieve high-level goals, an executive layer that sequences tasks and monitors task execution, and a behavioral layer that interfaces to the robot's sensors and effectors.

Typically, information and control flows up and down between layers. The planning layer sends plans to the executive, which further decomposes tasks into subtasks and dispatches them based on the temporal constraints imposed by the plan. Dispatching a task often involves enabling or disabling various behaviors. The behaviors interact to control the robot, sending back sensor data and status information. The executive informs the planner when tasks are completed, and may further abstract sensor data for use by the planner. The executive also monitors task execution: In case of failure, it can try to recover or it can terminate the task and request a new plan from the planner.

We extend this architectural concept to multiple robots in a relatively straightforward way. Each robot is composed of a complete three-layered architecture. In addition, each of the three layers can interact directly with the same layer of the other robots (Figure 1). Thus, each robot can act autonomously at all

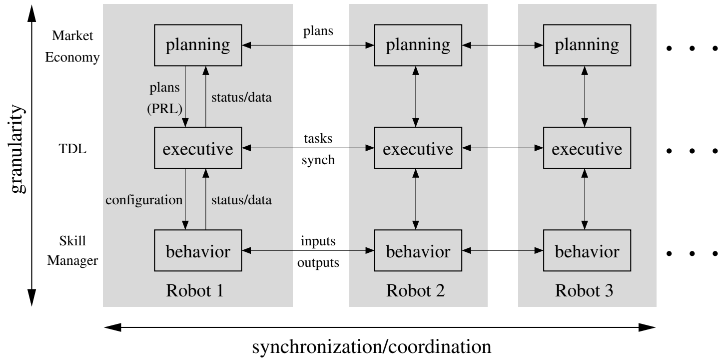

Figure 1. Layered multi-robot architecture

times, but can coordinate (at multiple levels) with other agents, when needed. By allowing each layer to interact directly with its peers, we can essentially form distributed feedback loops, operating at different levels of abstraction and at different timescales. In particular, the behavioral layer coordinates behaviors, the executive layer coordinates tasks, and the planning layer coordinates/schedules resources. In this way, problems arising can be dealt with at the appropriate level, without having to involve higher layers. This decreases latency and may increase robustness (since lower layers typically operate with higher-fidelity models).

The following sections describe each of the layers of the architecture in more detail. For each, we also provide a case study illustrating how interaction at that level can be used for multi-robot coordination.

3.1 Coordinated Behaviors

The behavioral layer consists of real-time sensor/effector feedback loops. By connecting the sensor behaviors of one robot to the effector behaviors of another, we can create efficient distributed servo loops (Simmons et al., 2000b). Similarly, by connecting effector behaviors together, we can create tightly coordinated controllers. For instance, two robots could coordinate their arm and navigation behaviors to jointly carry a heavy piece of equipment (Pirjanian et al., 2001).

A multi-robot behavioral layer that can support such capabilities needs several critical features. First, it must be possible to connect behaviors to one another, enabling sensor data and status information to flow between behaviors on different robots. The architecture should not place restrictions on the type of data that can pass between distributed behaviors. Also, it should be possible to connect behaviors transparently on different robots, in the same way that one connects them on the same robot. Finally, high-bandwidth, lowlatency communications is needed to achieve good performance in interacting, multi-robot behaviors.

Our implementation extends the Skill Manager of (Bonasso et al., 1997) to provide for both intra- and inter-robot connections. Skills are connected via input/output ports and operate in a data-flow fashion: When a new input value arrives on a port, the skill runs an action code that (optionally) produces outputs. For skills on the same robot, the connection is via function call; for inter-robot connections, data flows using a transparent message-passing protocol. Certain aspects of the skills, such as the action code and number and types of ports, are statically determined at compile time. Most aspects, however, can be dynamically configured at run time (either from the executive layer, or from a skill's action code). These include the ability to enable and disable a behavior, set the value of an input, set parameters of the skill, and create or destroy the connections between ports. The executive layer can also subscribe to skill outputs.

These ideas have been demonstrated in the context of distributed visual servoing for large-scale assembly using multiple, heterogeneous robots (Simmons et al., 2000b). The task, which is to move the end of a large beam into a vertical stanchion, uses three robots (Figure 2) – an observer (a mobile robot with stereo vision) and two controllers (an automated crane and a mobile manipulator).

The observer robot tries to maintain the best view of the fiducials on the beam, stanchion, and manipulator arm. It move the cameras and drives around the workspace to keep those fiducials it is currently tracking centered in the image and filling most of the cameras' fields of view. The observer uses stereo vision to compute the 6 DOF pose of the fiducials and outputs the differences between the poses to one of the controller robots, depending on which skill it is connected to at the time. In particular, the observer's visual tracking behavior (Figure 3) first helps direct the crane to move the beam near the stanchion, then aids the mobile manipulator in grabbing the beam, and finally helps the mobile manipulator to place the beam in the stanchion (5mm clearance).

3.2 Coordinated Task Execution

The executive layer has responsibility for hierarchically decomposing tasks into subtasks, enforcing synchronization constraints between tasks (both those imposed by the planner and those added during task decomposition), monitoring task execution, and recovering from exceptions (Simmons, 1994).

For coordinated multi-robot task execution, it should be possible to synchronize tasks transparently on two different agents, in the same way as if they were performed by a single agent. For instance, we may want to state that a robot should not start analyzing a rock until two other rovers have moved into place

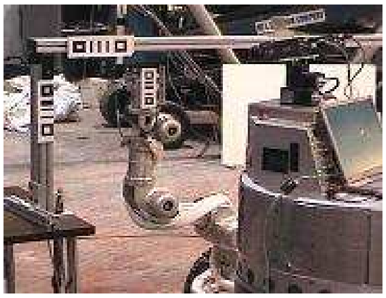

Figure 2. Heterogeneous robots for large-scale assembly

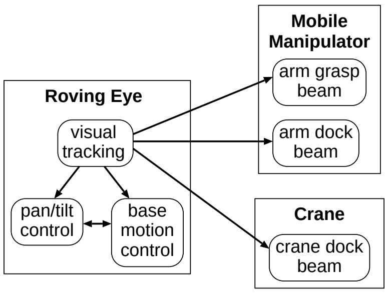

Figure 3. Coordinated behaviors for distributed visual servoing

to provide assistance. In addition, a distributed executive should facilitate one robot monitoring the execution of another robot and helping it recover from faults. Finally, it is desirable for one robot to be able to directly request that another robot perform a task (such as assisting it to perform visual servoing).

Our interface between planner and executive is based on PRL, a plan representation language that was developed for an earlier project (Simmons et al., 2000a). PRL represents plans as a hierarchy of tasks, with each task defined in terms of parameters, subtasks, and temporal constraints between the subtasks. We have recently extended PRL to enable specification of the resource utilization of a task and which agent should be executing it.

The executive is implemented using the Task Description Language. TDL is an extension of C++ that contains explicit syntax to support hierarchical task decomposition, task synchronization, execution monitoring, and exception handling (Simmons and Apfelbaum, 1998). Recently, we have extended TDL to handle task-level coordination between robots and to enable one robot to spawn or terminate a task on another. TDL transparently handles passing task data (function parameters) and synchronization signals between robots, using message passing.

We have used these ideas to perform coordinated deployment of multiple, heterogeneous robots (Simmons et al., 2000a). The executive receives a plan consisting of a set of deployments, where each deployment is an ordered list of locations, the robots that should to deploy to those locations, and a deployment "style." For instance, in the "group" deployment style all the robots navigate concurrently to the first deployment location, one stays behind while the others continue to the next location, and so on. The executive has procedures for decomposing each deployment style into primitive navigation behaviors. It is then responsible for coordinating the tasks, to ensure that robots do not move until others are in position. Figure 4 presents a simplified version of the TDL code for deploying the robots shown in Figure 5.

```
Goal GroupDeploy (DEPLOY_PTR deployList) {
  with (serial) {
    for (int i=0; i<length(deployList); i++) {
      spawn GroupDeploySub(i, deployList)
} } }

Goal GroupDeploySub (int phase,
                     DEPLOY_PTR deployList) {
  with (parallel) {
    for (int j=phase; j<length(deployList); j++) {
      spawn Navigate(deployList[i].location)
            with on deployList[j].robot;
} } }
```

Figure 4. TDL code (simplified) for "group" deployment strategy

Figure 5. Coordinated deployment of heterogeneous robots

3.3 Coordinated Planning

Our approach to task allocation and planning is based on a market economy. An *economy* is essentially a population of agents coordinating with each other to produce an aggregate set of goods. *Market economies* are those generally unencumbered by centralized planning, instead leaving individuals free to exchange goods and services and enter into contracts as they see fit. Despite the fact that individuals in the economy act only to advance their own self-interests, the aggregate effect is a highly productive society.

We have developed a market-based architecture in which tasks are allocated based on exchanges of single tasks between pairs of robots. A robot that needs a task performed announces that it will auction off the task as a buyer. Each seller capable of performing the task for a cost c, bids to do so for $c+\varepsilon$. The buyer accepts the lowest bid, as long as it is cheaper than doing the task itself. If a bid is accepted, the seller performs the task and pockets ε as profit.

This approach has been demonstrated in several simulated and actual robot applications. (Zlot et al., 2002) reports on multi-robot exploration of an unknown environment. Each robot generates a list of target points to visit, and orders them into a tour (using an approximate TSP algorithm). Next, the robots try to auction off tasks, one at a time (sequentially along the tour). When all its auctions close, the robot navigates to its first target point, incorporates map information, and generates new target points. Note that the robots are able to share map information via trades on the market, and that all communication is asynchronous and not assumed to be reliable. Figure 6 shows a map built by four robots in a highbay, and Figure 7 shows the paths they took.

More recently, our market-based architecture has been extended to support *leader* agents that engage in multi-agent, multi-task exchanges. This can lead to task allocations that outperform those produced by repeatedly exchanging one task at a time. A leader agent opens a *combinatorial exchange* in which agents can bid to buy or sell combinations of tasks. The leader chooses which

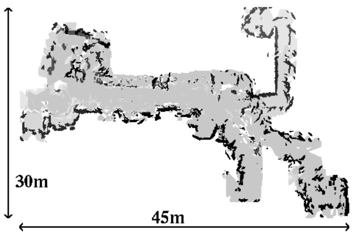

Figure 6. Map of highbay created by four robots (from (Zlot et al., 2002))

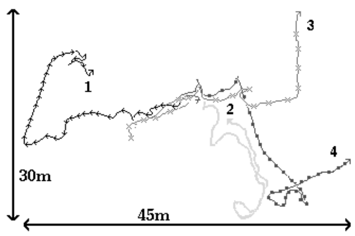

Figure 7. Paths taken by exploring robots (from (Zlot et al., 2002))

bids to accept and pockets the difference between the total revenue and cost of all the transactions. In order to earn a good profit, however, it must perform the computationally difficult problem of choosing which bids to accept in order to maximize efficiency. A leader opportunistically announces exchanges for those pockets of the overall problem that will net a good return on its computational investment (those that can produce the largest efficiency gain from its optimization).

3.4 Capability Server

A mechanism for providing up-to-date information on robots and their capabilities is necessary for highly dynamic multi-robot systems, where robot attrition, accession, and modification are common. The Agent Capability Server (ACS) is designed to handle this by providing a distributed facility for automatically disseminating agent information. The idea is that each robot in the system has its own local ACS that it can quickly query for the capabilities and status of other robots. This information can then be used in determining which robots can perform which tasks, planning efficient multi-robot strategies for a task, establishing the appropriate executive-level and behavior-level coordination mechanisms between robots, and accommodating non-responsive (possibly failed) robots. Our ACS is similar in purpose, though specialized by comparison, to middle agents that map capabilities to particular agents, and agent naming services that map agents to locations (Sycara et al., 2001).

Maintaining the consistency of information among the various Agent Capability Servers is a key concern, especially since communications range may be limited and influenced by geographic features. Thus, it cannot be assumed that a broadcast of new information will reach all robots. One method for boosting consistency in the face of uncertain communication is to have each ACS broadcast its data periodically. Thus, new information will eventually propagate throughout the group by transparently using robots as communication relays.

Each ACS monitors the periodic updates from the other servers. When a server has failed to provide an update for a sufficiently long interval, the robot is assumed to have failed or be completely out of communications range. In either case, its entry is removed from the ACS. If and when the missing robot is heard from again, it will be seamlessly re-incorporated.

4. Summary and Future Work

This paper describes a multi-robot extension to the traditional three-layered architecture, where each layer can communicate directly with its peer layers on other robots. This gives the robots the ability to coordinate at multiple levels of abstraction with minimal overhead in terms of inter- and intra-agent communication. We have described criteria and design decisions for each layer, and have presented case studies showing how layer-to-layer interaction enables reliable multi-robot coordination.

Much work still remains on the architecture. We need to integrate the planning and executive layers more fully. We need to generalize the market-based planning framework, especially by adding leader agents and their more sophisticated bidding structures. We need to have all levels of the architecture deal more fully with the loss of agents. We need to complete design and implementation of the Agent Capability Server. And, we need to demonstrate the architecture in rich domains.

Multi-robot coordination promises huge benefits in terms of increased capability and reliability. The price, however, is often added complexity. We believe that a well-structured, flexible architecture will facilitate the development of such systems, at reasonable cost.

Acknowledgments

This work has been supported by several grants, including NASA NCC2-1243, NASA NAG9-1226, DARPA N66001-99-1-892, and DARPA DAAE07-98-C-L032. Thanks go to Steve Smith, Jeff Schneider, Drew Bagnell, and Vincent Cicirello for their comments and advice on the architecture. David Apfelbaum implemented and helped design TDL.

References

Arkin, R. (1992). Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364.

Balch, T. and Arkin, R. C. (1994). Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52.

Bonasso, R., Kortenkamp, D., Miller, D., and Slack, M. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Artificial Intelligence Research*, 9(1).

Gerkey, B. P. and Matarić, M. J. (2001). Sold!: Market methods for multi-robot control. In *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems*.

Golfarelli, M., Maio, D., and Rizzi, S. (1997). A task-swap negotiation protocol based on the contract net paradigm. Technical Report CSITE, 005-97, University of Bologna.

Jennings, J. and Kirkwood-Watts, C. (1998). Distributed mobile robotics by the method of dynamic teams. In *Proc. Conference on Distributed Autonomous Robot Systems*.

Khatib, O. (1995). Force strategies for cooperative tasks in multiple mobile manipulation systems. In *Proc. International Symposium of Robotics Research*.

Mataric, M. (1992). Distributed approaches to behavior control. In *Proc. SPIE Sensor Fusion V*, pages 373–382.

Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. (1998). Remote agent: To boldly go where no ai system has gone before. *Artificial Intelligence*, 103(1–2):5–48.

Parker, L. (1998). Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240.

Pirjanian, P., Huntsberger, T., and Barrett, A. (2001). Representation and execution of plan sequences for distributed multi-agent systems. In *Proc. International Conference on Intelligent Robots and Systems*.

Sandholm, T. and Lesser, V. (1995). Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proc. International Conference on Multiagent Systems*, pages 328–335.

Simmons, R. (1994). Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43.

Simmons, R. and Apfelbaum, D. (1998). A task description language for robot control. In *Proc. International Conference on Intelligent Robots and Systems*, Vancouver Canada.

Simmons, R., Apfelbaum, D., Fox, D., Goldman, R., Haigh, K. Z., Musliner, D., Pelican, M., and Thrun, S. (2000a). Coordinated deployment of multiple, heterogeneous robots. In *Proc. International Conference on Intelligent Robots and Systems*, Takamatsu Japan.

Simmons, R., Goodwin, R., Haigh, K., Koenig, S., and O'Sullivan, J. (1997). A layered architecture for office delivery robots. In *Proc. First International Conference on Autonomous Agents*.

Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith, T. (2000b). First results in the coordination of heterogeneous robots for large-scale assembly. In *Proc. International Symposium on Experimental Robotics*, Honolulu Hawaii.

Smith, R. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

Stentz, A. and Dias, M. B. (1999). A free market architecture for coordinating multiple robots. Technical Report CMU-RI-TR-99-42, Robotics Institute, Carnegie Mellon University.

Sycara, K., Paolucci, M., van Velsen, M., and Giampapa, J. (2001). The retsina mas infrastructure. Technical Report CMU-RI-TR-01-05, Robotics Institute, Carnegie Mellon University.

Sycara, K. and Zeng, D. (1996). Coordination of multiple intelligent software agents. *International Journal of Cooperative Information Systems*, 5(2–3).

Thayer, S., Digney, B., Dias, M. B., Stentz, A., Nabbe, B., and Hebert, M. (2000). Distributed robotic mapping of extreme environments. In *Proceedings of SPIE: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*.

Wellman, M. and Wurman, P. (1998). Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, pages 115–125.

Zlot, R., Stentz, A., Dias, M. B., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *Proc. International Conference on Robotics and Automation*.