Modular Behaviors in Heterogeneous Groups of Mobile Robots

Göksel Dedeoglu and Gaurav S. Sukhatme

dedeoglu,gaurav@robotics.usc.edu

Robotics Research Laboratory
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781

ABSTRACT

We provide an overview of the software components underlying four different tasks performed by a heterogeneous group of mobile robots. These tasks are are drawn from three domains 1. Robot Competitions (Robot Soccer and 'Find Life on Mars'), 2. Security and Surveillance (Perimeter Protection) and 3. Building Environmental Models (Multi-Robot Navigation and Mapping). Once decomposed as a set of cooperating behaviors, we show how these (seemingly unrelated) tasks lead to similar solutions as far as their modular breakdown is concerned, thereby yielding high reusability. Although our collection of robot platforms is notably diverse in terms of mechanics, sensory and computational capabilities, cross-platform migration and extension of existing behavior assemblages require minimal programming effort.

Keywords: Behavior-based robotics, Robot programming, Mobile robots, Autonomous robots, Modular behaviors

1. INTRODUCTION AND RELATED WORK

Behavior-based systems have become popular in the past decade as a means of mobile robot control. We present four examples of behavior-based, multi-robot systems which we have constructed over the past two years. These systems are modular, extensible, reusable and heterogenous making them useful for a sustained program of research and application.

Behavior-based systems have been extensively reported in the literature[1,2] and have been applied to a diverse collection of tasks such as navigation, terrain mapping, distributed group foraging, collective coordinated pushing etc. Operationally, we think of behaviors as distributed processes that have direct connections to each others or to the sensors and actuators on the robot. This representation has been used by others[3,4] with considerable success.

Our approach (a generalization of contemporary behavior-based systems) focuses on encapsulating certain competencies into processes that communicate with each other. Behavior-based approaches have their genesis in the Subsumption approach which advocated a fixed layered structure.[5] However subsequent research has established their ability to provide representation[4] and integrate low-level reactive behaviors ("avoid obstacle") with goal-directed behaviors ("follow waypoints to target") and adaptive behaviors ("learn features of hazardous obstacles") without any centralized control.

The key problem in multi-robot systems is to achieve the desired level of global coordination in a decentralized, distributed manner i.e. by using local rules. While a centralized planner may be suitable for coordinating the behavior of a largely static collection of robots and sensors, it remains a bottleneck. This results in a system that scales poorly. The other extreme is a collection of robots each of which is purely reflexive (or reactive). Such a system scales well and is useful for a dynamic collection of robots and sensors but may not achieve the desired level of global coordination. Other researchers have addressed the issue of achieving global objectives through local rules.[6,2,7–9]

The rest of this paper is organized as follows. Section 2 provides an overview of hardware and software architecture of the robots. Three different application areas are discussed in Section 3, where physical implementations and actual robot demonstrations were realized. In Section 4 we comment on our experience in behavior-based programming in general.

Figure 1. Pioneer-1 AT (left), RWI Urban Robot (middle), and Pioneer-2 DX (right) platforms

2. SYSTEM DESCRIPTION

2.1. Hardware Platforms

From a hardware point of view, our mobile robots are a heterogenous group: The Pioneer-1 AT (Figure 1, left) is designed for outdoor applications, and it handles rough terrain and grassy areas well. Wide, inflatable tires and electric drives without current limiters enable the Pioneer to produce high torques to overcome the modest bumps and pits encountered outdoors. All wheels are actuated, and there is strong mechanical coupling between tires on the same side. Its light-weight gripper is equipped with beam and contact sensors, allowing it to pick up objects automatically. The robot is equipped with seven Polaroid sonar sensors, facing the sides and the front. It also carries a Pan-Tilt-Zoom camera (Sony), feeding into a three-channel color blob detector (Cognachrome FastTrack Vision from Newton Labs) that works in real-time. The white box that the Pioneer-1 AT is seen carrying encloses a PC-104 stack (486DX4-133, 16 MB RAM, 40 MB solid state disk) responsible for all high-level computing, such as the behaviors discussed in this paper. The underlying operating system is QNX. Also present in this box is a gyroscope (Systron Donner, QRS14-64-109) that has been integrated using Kalman filtering to enhance the robot's odometry.[10] A DGPS receiver card and antenna provide global positioning accuracy of about 1 meter outdoors, and a digital compass is used in conjunction with this absolute positioning system for navigation purposes. The network connectivity of the robot is provided by a wireless ethernet setup (WaveAccess, DS132) at approximately 1.6 MBit/sec, with a range up to 100 meters line-of-sight.

The Urban Robot (Figure 1, middle) has been developed by RWI/IS Robotics to address issues encountered in hostile urban search and rescue scenarios: Its tracks offer good mobility, whereas its rotating arms, also actuated, allow it to climb stairs. It carries 10 sonar transducers distributed equally on the front and both sides of the robot. Robot control software runs on a Pentium-based computer located within its chassis, reachable via wireless ethernet (BreezeNet), and running the Linux operating system.

The Pioneer-2 DX (Figure 1, right) is an indoor robot platform, with a two-wheel differential drive and a caster. In addition to 16 sonar transducers facing almost all directions, it is equipped with a SICK laser scanner with a field of view of 180 degrees and range of 8 meters. Its electronic compass and wireless ethernet hardware are the same as the Pioneer-1 AT's described above. The robot body encloses a Pentium-based PC which runs the QNX operating system.

Nevertheless, from a control point of view, our robots use a common interface. Lower-level microcontrollers provide a set of commands that allow translational and rotational velocities to be set, and sensory information is updated periodically, both over a serial line. Moreover, since all robots use differential steering, motion algorithms can typically migrate from one to another, requiring only maximum/minimum speed limits to be translated.

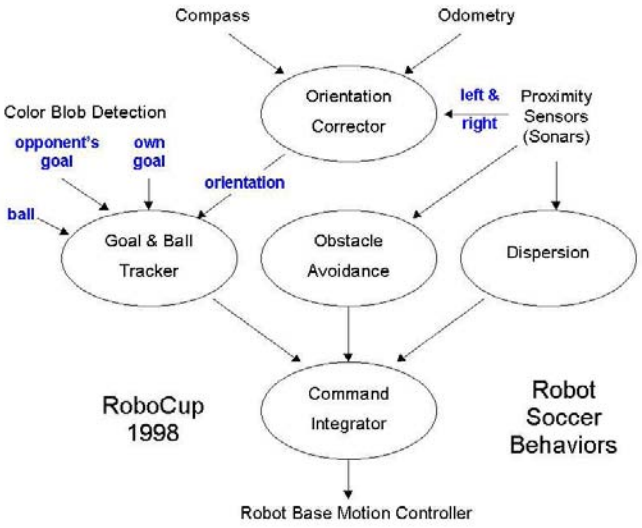

Figure 2. Robot soccer behaviors used in RoboCup '98

2.2. Software Architecture

All our robot control algorithms are implemented as a set behaviors,[11,5] which are specialized light-weight processes that run in parallel. Ports serve as communication channels between behaviors, and their connectivity is defined after the task at hand. At run-time, a behavior typically completes its sense-decide-act loop as follows: If input ports relevant to the task have not been updated by sensors or other behaviors, it simply suspends execution for a cycle. Alternatively, if some internal timer is expired or new input data is available, it proceeds with executing its computational task, posting the results at the output ports. The execution frequency of behaviors is set based on the requirements of the task, and ranges from 0.1 to 20 Hz in the applications presented in this paper.

The particular behavior-based programming language we have been using is Ayllu,[12] which is shipped as one of the programming tools with the Pioneer robots. It should also be noted that our inter-behavior communication is not limited to ports that supersede others. While priority-ordered ports are supported by the language, we typically program arbitrator behaviors that can make more complex decisions when necessary. Behaviors have access to local static variables and timers in addition to all other types provided by the C programming language.

3. APPLICATION EXAMPLES

This section presents four mobile robot tasks that we developed and demonstrated over the past two years. In the following figures, our software modules (i.e., behaviors) are depicted as blobs, with arrows indicating communication ports established between them. Sensory input or user command arrows are annotated with the modality in question.

3.1. Robot Competitions

In recent years robot competitions have become a popular part of the research community in mobile robotics. Competitions serve to involve young researchers and capture their attention and interest. They also provide incentives to researchers to incorporate recent technological innovations into their experimental work. Competitions also foster an integrated approach[13] towards design and problem solving in general.

Robot Soccer

The Robot World Cup Initiative (RoboCup) is an international research initiative that "aims to provide a standard problem where wide range of technologies can be integrated and examined".[14] Having chosen soccer as the application domain, it defines various categories (Simulation, Small/Middle/Full Size, Sony Legged, Humanoid and Commentator

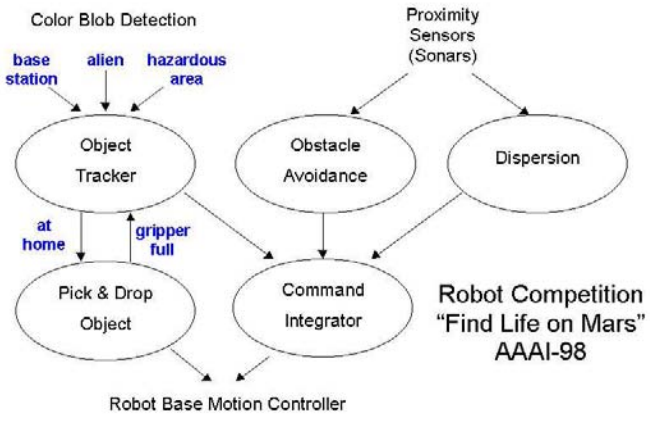

Figure 3. Behaviors used in AAAI-98 "Find Life on Mars" Competition

Leagues) in order to extend its reach to a variety of research communities. A USC Robotics Research Laboratory team participated in RoboCup '98 (Paris, France) in the middle-size robot category.[15] In essence, the algorithms that were implemented for this competition were the same as those described in,[16] except that they were coded in Ayllu[12] for the 1998 competition. Object tracking was the basic skill needed for the robot soccer task. The desired overall behavior of the robot consisted of pushing the ball in the direction of the opponent's goal, while avoiding obstacles and other robots.

Figure 2 shows the behaviors that provide a simple, yet effective solution to this task. The crucial component is **Goal & Ball Tracker**. The latter tracks the ball at a distance. Having gotten close to it, if the opponent's goal is in sight, it aggressively drives the robot towards the ball, trying to align its course with the goal. However, if the opponent's goal is not in sight, it generates a defensive maneuver[16] towards the ball, placing the robot between its home goal and the ball. Since the goals are not always visible to the robot, an orientation correction mechanism is introduced for keeping the robot heading information up-to-date: **Orientation Corrector** extracts line segments from the left & right sonar sensors' range measurements, and suggests incremental corrections for enhancing the heading odometry. In addition, the compass is used when no wall segments are visible by the robot for a long time, although its performance indoors is highly affected by magnetic fields of other robots, cameras, piping and fixtures present near the field.

Dispersion and **Obstacle Avoidance** behaviors allow the robots to keep a desired distance from other objects while pursuing the ball tracking task described above. For safety reasons, **Obstacle Avoidance** was always given the highest priority, whereas other behaviors adapted their speed profiles as a function of their state & perceptions (e.g. **Ball Tracker** issuing higher velocities in an attempt to kick the ball towards the goal).

Find Life on Mars

A USC Robotics Research Laboratory team participated in the AAAI-98 Mobile Robot Competitions organized in August 1998 in Madison, Wisconsin. This section presents the software architecture that competed in the "Find Life on Mars" event of the object manipulation category. The goal of this event was to "seek out new life forms, collect them, categorize them, and return them back safely to the Mars Lander".[17,18]

The behaviors that accomplish this task are very similar to those used in robot soccer: **Object Tracker** detects the aliens, home (Mars Lander) location and hazardous areas using the color blob tracking system. In the search-for-aliens mode, the camera is tilted down to enhance the detection of hazardous areas, which are basically colored patches on the floor that result in penalty points when overrun by the robot. The grippers close and lift automatically as an alien breaks the light beam integrated into the grippers, putting the robot in homing mode. The base location needs to be approached closely for **Pick & Drop Object** to release the grippers and go back to the search mode.

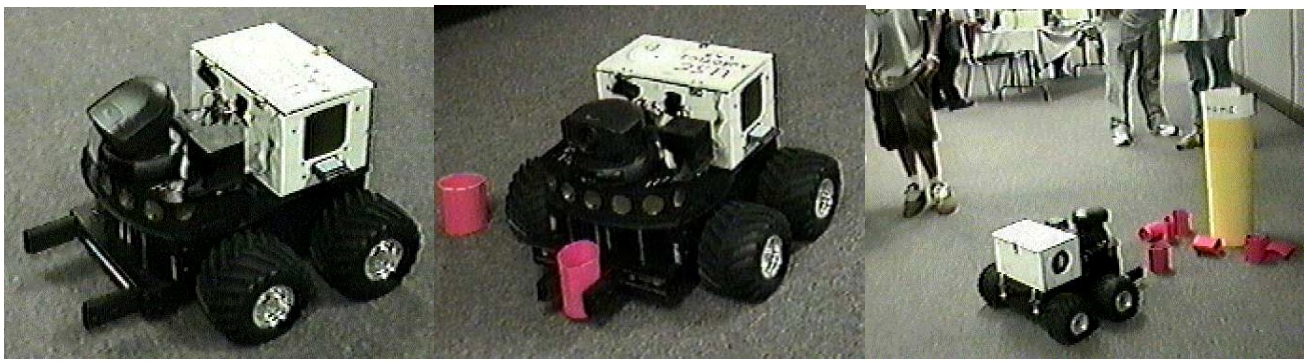

Figure 4. A Pioneer 1-AT robot demonstrates the "Find Life on Mars" concept in the Lancaster Fair '98, California. To the left, the robot is in search mode for "aliens", with its camera pointing down. Once it finds one and picks it up (middle), it starts searching for the base station where it will eventually drop it off. The rightmost picture shows the home location after a couple of minutes of "searching for aliens".

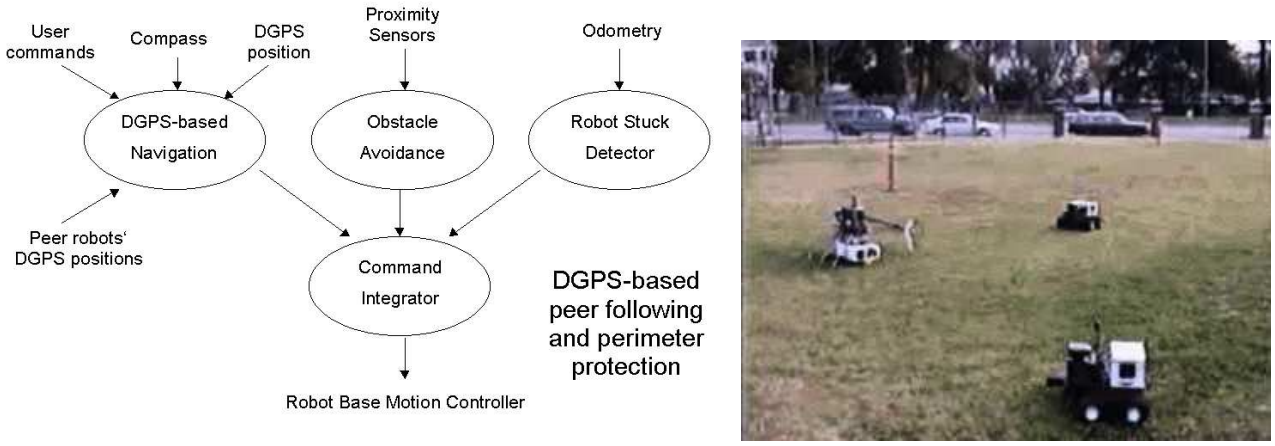

Figure 5. Behaviors implementing the perimeter protection task (left). Two Pioneer-1 ATs are on a mission with the AVATAR autonomous helicopter.

Dispersion allows the robot to follow the perimeter of the arena, which was found to be more effective than random navigation in this particular event.

3.2. Security and Surveillance

Security and surveillance are often cited as ideal applications for mobile robots, since they require intensive, round the clock, reliable sentries. Previous efforts along these lines[19,20] emphasized distributed processing for fault tolerance and the need for testing within a broad performance envelope. We describe below a modular, behavior-based, approach to a perimeter protection task which exploits robot heterogeneity.

Outdoor Perimeter Protection

This task involves cooperation between two ground-based mobile robots and a semi-autonomous robotic helicopter, AVATAR,[21,22] in order to perform automated surveillance and reconnaissance outdoors. A simulated sensor net detects intrusions into a protected area, upon which ground and air robots react by navigating to the location of activity and relaying a live video feed to the user. Based on the user's request, individual ground robots can be ordered to follow their peer or the helicopter,[23] or any known/stored sensor location.

DGPS-based Navigation is responsible for servoing to the goal location which is either stored statically, or updated in real-time when tracking other vehicles in motion. **Robot Stuck Detector** keeps a limited history of motor

commands and odometry readings in order to determine whether the robot is stuck, and generates sudden maneuvers that are typically sufficient in order to overcome the ground friction, bringing the robot back to its normal navigation.

3.3. Building Environmental Models

Topological Mapping and Indoor Navigation

As discussed in an earlier paper,[24] we developed an incremental, topological mapping and autonomous navigation scheme geared for time-critical indoor exploration tasks. The objective of our mapping system is to enable users to efficiently acquire an overall view of the main characteristics of the interiors of a building, consisting of basic topological features such as corridors, junctions, and corners. The incremental nature of the approach allows the user to view the most up-to-date map displayed at any time. As the robot proceeds to explore, it constantly estimates its position and orientation and updates the map in real-time This system has been demonstrated on all three robot platforms discussed earlier namely, the Pioneer-1 AT, the Pioneer-2 DX and the Urban robot.

The navigation strategy is relatively simple: a robot traverses corridors while avoiding obstacles, turning only when its path is blocked. Right turns are preferred to left turns when both are possible, and U-turns are performed at dead-ends. This results in paths which travel deep into the building rather than thoroughly mapping the nearby rooms. Just like any other behavior in the system, the exploration algorithm is independent of the other modules (basic navigation skills, mapping, etc.), and can be replaced easily. Figure 6 depicts the behaviors involved in the navigation system and their connections: **Safe Navigator** is responsible for avoiding obstacles based on readings from proximity sensors, i.e. objects too close to the robot cause it to move away from them. **Explorer**, on the other hand, only cares about successfully following corridors or moving along walls, which requires the corridor heading to be tracked. The **Point-To-Point Navigation** module accomplishes short-lived, open-loop navigation tasks solely based on odometry. The **Command Integrator** gives priority to **Safe Navigator** whenever the latter is active.

The behaviors for detecting topological features run at frequencies determined by the sensor modality in question. The building block of sonar-based feature detection algorithms is a line segment: Given the physical location and direction of sonar transducers, five separate buffers (one left, one right, and three frontal) are defined, each updated with projections of proximity readings onto the robot's two dimensional global coordinate system. If the buffers get full, or too much deviation is registered between consecutive sonar readings, a Least Squares line fitting algorithm is applied to the buffer in order to produce a line segment along with a quantitative measure of the fit. Line segments with poor fit rates are ignored. Also, a weak orthogonality condition is required between the buffer's average direction and the wall segment's heading. Extracted line segments are immediately sent to sonar-based feature detectors, and expire within seconds of their creation. This scheme does not rely on global accuracy of the underlying odometry capability, and is only meant to support local feature extraction.

The **Opening Detector** is designed to watch for discontinuities in consecutive line segments originating from left and right sonar buffers. A good candidate for corridor openings or open doors is a gap between two wall segments on the same side of the robot with matching headings. The **Corridor Detector** seeks sufficiently long, and overlapping right and left wall segments with approximately matching bearings, and outputs their average heading. The **End-Of-Corridor** detector looks for frontal wall segments that are orthogonal to the corridor being traversed. On the Pioneer-1 AT model, the **Door Detector** processes visual color blob information provided by the underlying vision system, examines the shape of perceived colored objects, and recognizes a closed door by its width and height. When the laser scanner is available, as is the case on our Pioneer-2 DX, a totally different **Door Detector** comes into play and analyzes complete scans to extract signatures caused by closed doors. Figure 7 (right) summarizes the organization of feature detection modules and their roles.

Figure 7 (left) depicts the behavior set responsible for generating a position estimate. Earlier work to enhance a Pioneer-1 AT mobile robot's odometry[10] resulted in substantial improvement in position estimation. A Kalman Filter incorporated wheel velocities and the output of a relatively inexpensive gyroscope using a calibrated kinematic model of the mobile platform. As explained in detail in,[24] we assume straight and orthogonal corridors. Corrections based on these constraints were suggested by the **Odometry Corrector** behavior in an on-line fashion, based on the discrepancies in most recently detected corridor stretches and landmarks.

More recently, we have extended this mapping scheme to the multi-robot case,[25] in which topological maps built independently by autonomous robots match and correctly combine their partial maps.

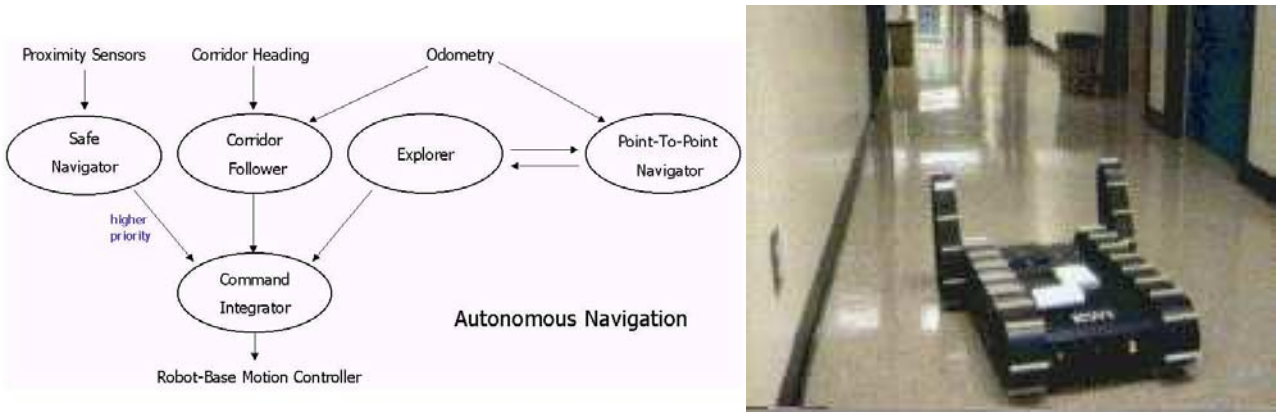

Figure 6. Autonomous navigation behaviors (left), and the Urban Robot in a hallway (right).

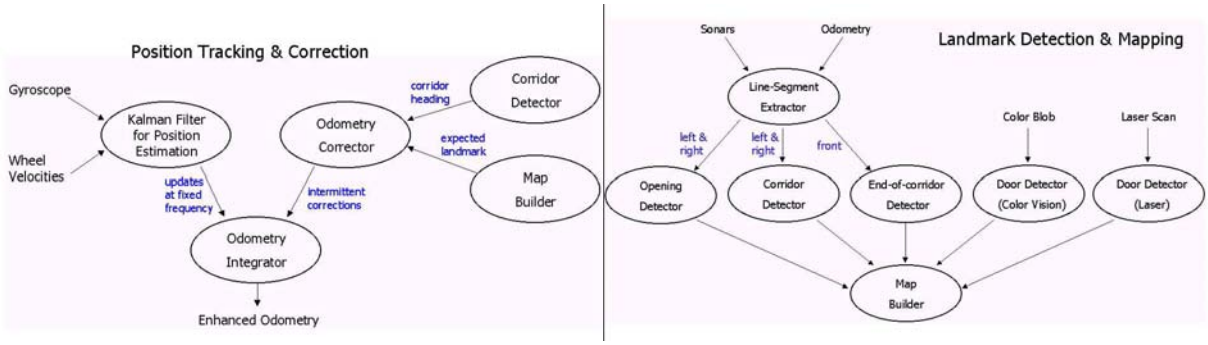

Figure 7. Position correction (left) and landmark detection (right)

4. CONCLUSION

We have described four tasks (drawn from three domains) performed by mobile robot systems. Based on our experience* with these systems, we make the following observations:

- Behavior-based systems are inherently **modular**. The behavior-based philosophy is to encapsulate functionality into small units called behaviors which are typically modeled as processes. The nature of the computation within each behavior can be arbitrarily complex. Behaviors are connected to each other through a limited set of ports.
- Modularity implies **reusability**. The systems we have described in this paper (and others we have built in the past) take code reuse seriously. We are able to easily take a behavior developed for one application and reuse it for a completely different application. For example, the obstacle avoidance behavior was initially developed on the Pioneer-1 ATs and was readily ported to the mapping and navigation system on the Urban robot and the Pioneer-2 DXs.
- Our systems are **extensible**. For example, the initial mapping system we built used sonar sensors to detect wall segments. When laser rangefinders became available, it was relatively easy to incorporate them into the system. This is because the feature detector behaviors were originally decoupled from the navigation and mapping behaviors.
- Modular design provides a good basis for coping with **heterogeneity**. From the examples discussed in this paper, it is apparent that our design philosophy allows us to cope with heterogeneous systems without too much additional overhead.

*The interested reader is refered to[26,27] for demonstration videos

ACKNOWLEDGMENTS

This work is supported by contracts F04701-97-C-0021 and DAAE07-98-C-L028 from DARPA under the Tactical Mobile Robotics (TMR) program. Thanks to LTC John Blitch for the loan of the Urban robot as part of the TMR program. The authors also thank Brian Ellenberger, Monica Nicolescu and Barry Werger (members of the AAAI and RoboCup teams) for their help.

REFERENCES

1. M. J. Matarić, "Behavior-based control: Examples from navigation, learning, and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence* **9**(2–3), pp. 323–336, 1997.
2. M. J. Matarić, "Issues and approaches in the design of collective autonomous agents," *Robotics and Autonomous Systems* **16**, pp. 321–331, December 1995.
3. R. A. Brooks, "Intelligence without reason," in *Proceedings, IJCAI-91*, pp. 569–595, (Sydney, Australia), 1991.
4. M. J. Matarić, "Integration of representation into goal-driven behavior-based robots," *IEEE Transactions on Robotics and Automation* **8**, pp. 304–312, June 1992.
5. R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation* **2(1)**, pp. 14–23, March 1986.
6. M. J. Matarić, "Behavior-based systems: Key properties and implications," in *IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, pp. 46–54, (Nice, France), May 1992.
7. M. J. Matarić, "Learning social behavior," *Robotics and Autonomous Systems* **20**, pp. 191–204, 1997.
8. M. A. Lewis and G. A. Bekey, "The behavioral self-organization of nanorobots using local rules," in *Proc. 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1333–1338, July 1992.
9. A. Agah and G. A. Bekey, "Sociorobotics: One to many robots," in *Proc. International Conference on Artificial Intelligence and Information-Control Systems of Robots 1994*, pp. 123–135, September 1994.
10. P. Goel, S. I. Roumeliotis, and G. S. Sukhatme, "Robust localization using relative and absolute position estimates," in *Proceesings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999.
11. R. C. Arkin, *Behavior-Based Robotics*, MIT Press, 1998.
12. B. B. Werger, "Ayllu: Distributed port-arbitrated behavior-based control," in *Proceedings of Distributed Autonomous Robot Systems 2000*, Springer Verlag, (Knoxville, TN), October 2000.
13. G. Sukhatme, J. Montgomery, and G. Bekey, "Implementing robots in hardware as a tool for integration," in *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1998.
14. RoboCup, "http://www.robocup.org/," WWW.
15. RoboCup, "http://www.robocup.org/games/98paris/312.html/," WWW, 1998.
16. B. B. Werger, "Cooperation without deliberation: A minimal behavior-based approach to multi-robot team," *Artificial Intelligence* **110**, pp. 293–320, 1999.
17. AAAI, "http://dangermouse.uark.edu/aaai-98/life-on-mars/," in *Find Life on Mars*, WWW, 1998.
18. AAAI, "http://www.cc.gatech.edu/ aaai98/," in *Robot Competitions*, WWW, 1998.
19. H. Everett and G. Gilbreath, "A supervised autonomous security robot," *Robotics and Autonomous Systems* **4**, pp. 209–232, November 1988.
20. H. Everett and D. Gage, "From laboratory to warehouse: security robots meet the real world," *International Journal of Robotics Research* **18**, pp. 760–768, July 1999.
21. A. H. Fagg, M. A. Lewis, J. F. Montgomery, and G. A. Bekey, "The usc autonomous flying vehicle: an experiment in real-time behavior-based control," in *Proceesings of the International Conference on Intelligent Robots and Systems*, pp. 1173–80, IEEE/RSJ, (Yokohama, Japan), 1993.
22. J. F. Montgomery, A. H. Fagg, and G. A. Bekey, "The USC AFV-I: A behavior-based entry in the 1994 international aerial robotics competition," *IEEE Expert* **10**(2), pp. 16–22, 1995.
23. G. S. Sukhatme, J. Montgomery, and M. Mataric, "Design and implementation of a mechanically heterogeneous robot group"," in *Proceedings of Mobile Robots XIV, Photonics East Conference*, SPIE, 1999.
24. G. Dedeoglu, M. J. Mataric, and G. S. Sukhatme, "Incremental, online topological map building with a mobile robot," in *Proceedings of Mobile Robots XIV, Photonics East Conference*, pp. 129–139, SPIE, (Boston, MA), 1999.
25. G. Dedeoglu and G. S. Sukhatme, "Landmark-based matching algorithm for cooperative mapping by autonomous robots," in *Proceedings of Distributed Autonomous Robotic Systems (DARS) 2000*, (Knoxville, TN), October 2000.
26. G. Dedeoglu, "Demonstration videos," http://www-robotics.usc.edu/~dedeoglu/videos.
27. J. Montgomery, "Demonstration videos," http://www-robotics.usc.edu/~avatar/video.html.