System Architecture for Versatile Autonomous and Teleoperated Control of Multiple Miniature Robots*

Paul E. Rybski[†], Ian Burt[‡], Tom Dahlin[††]
Maria Gini[†], Dean F. Hougen[†], Donald G. Krantz[‡‡]
Florent Nageotte[†], Nikolaos Papanikolopoulos[†**], Sascha A. Stoeter[†]

Center for Distributed Robotics, Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455

Abstract

Using robots for surveillance and reconnaissance applications requires a versatile connection between the human operator and robotic hardware. Some application domains require a fully teleoperated system while others may benefit by giving robots more autonomy. This paper describes a robotic control architecture which merges both paradigms. The whole scheme is implemented using the miniature Scout robot and involves a suite of user interfaces that can be tailored to specific surveillance and reconnaissance missions. Hardware capabilities are presented and a visual servoing strategy, important for semi-autonomous Scout operation, is discussed.

1 Introduction

Remote surveillance and reconnaissance applications can be greatly facilitated through the use of a small semi-autonomous robotic sensor platform. A human operator equipped with a wearable computer interface can direct a small mobile robotic platform into a dangerous or difficult to reach area and gather information from within. Sensor information is returned to the human through a head-mounted display and commands are sent from a hand-held controller. Simply teleoperating the robotic platform can be useful in many situations, but the operator's attention must be completely dedicated to controlling the robot. This may be difficult in an emergency situation because of the high likelihood of distractions. To assist the human operator, the robot can be given more autonomy, allowing it to interpret higher-level commands from the human. If a task requires the use of multiple robots, a human operator cannot be expected to simultaneously control them all directly. Instead, the robots must be able to perform useful tasks when the operator's attention is called elsewhere.

A layered system, shown in Fig. 1, has been developed to facilitate this kind of multi-modal control. This system includes user interfaces for teleoperation clients and robust sensor interpretation algorithms for autonomous control clients. A distributed software control architecture dynamically coordinates

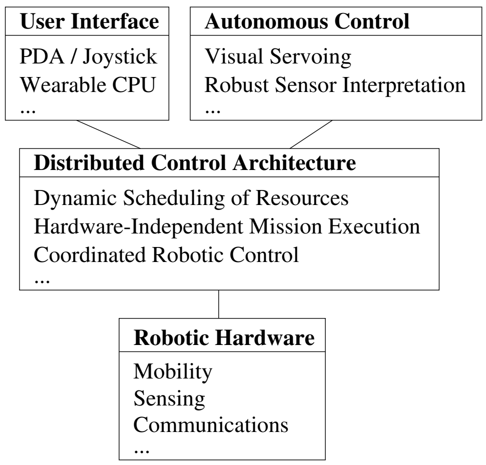


Figure 1: A layered system architecture for the control of multiple robots in both teleoperated and autonomous modes.

*This material is based upon work supported by the Defense Advanced Research Projects Agency, Microsystems Technology Office (Distributed Robotics), ARPA Order No. G155, Program Code No. 8H20, Issued by DARPA/CMD under Contract #MDA972-98-C-0008.

[†]Center for Distributed Robotics and Dept. of Computer Science and Engineering, University of Minnesota
[‡]Dept. of Mechanical Engineering, University of Minnesota
[††]Dahlin Consulting
[‡‡]MTS Systems Corporation
[**]Corresponding Author

hardware resources and shares them between the various clients, allowing for simultaneous control of multiple robots.

The paper starts with a description of the updated Scout robot (the hardware focus of the system) and continues with the presentation of the various designs and implementations of user interfaces. The distributed control architecture is then presented followed by a novel mobile robot visual servoing scheme.

2 Robots for Surveillance and Reconnaissance

A robot that is small enough to avoid detection and can access hard to reach areas is extremely useful for surveillance and reconnaissance applications. Such a robot should be able to transmit environmental information to a remote location for additional analysis. The robot should also be able to receive remote instructions to help guide its search. The Scout robot, shown in Fig. 2, is an innovative miniature robot developed specifically to address these requirements. The robot's small size (11 cm long and 4 cm wide) and light weight (approximately 200 g) allows it to be easily carried by a human or another robot.

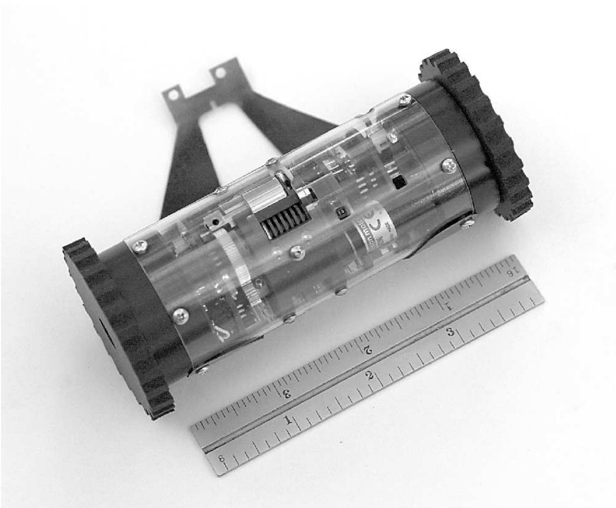


Figure 2: The Scout robot shown next to a ruler (in inches) for scale.

The Scout is able to navigate through most indoor environments using its differentially-driven wheels and can climb a 20° slope. The Scout can also jump over small obstacles, such as stairs, by winching its leaf-spring tail around its body and snapping it out. Fig. 3 shows the Scout jumping up a step.

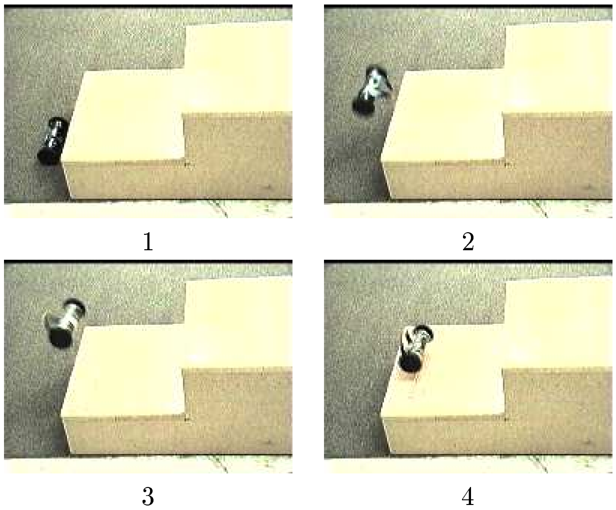


Figure 3: A Scout using its spring to jump up a stair.

Scouts carry a sensor payload, usually a small camera, used to broadcast environmental information over an analog RF transmitter. Scouts can also transmit and receive digital data over a separate RF modem using a custom network protocol. Due to their small size, Scouts have extremely limited on-board computational power, requiring the full capacity of their two CPUs for network communications and actuator control. Motion commands are transmitted from a remote source where they are received and executed by the robot.

2.1 Scout Hardware Updates

The Scout robots are in their second phase of development [7]. Nearly all of their hardware components have been redesigned to enhance their capabilities. The Scouts have a new radio, developed by Honeywell Labs Inc., which greatly increases the range and throughput of commands sent to the robots. The electrical system has been updated to make more efficient use of the batteries, greatly increasing the operational lifetime. The two CPUs have been upgraded to decrease power usage by a factor of ten while increasing the amount of available code space. The video transmission systems have been channelized to allow simultaneous broadcasts. Encoders have been added to the drive motors, allowing the robot to accurately control its speed and calculate the distance it has traveled. The jumping spring and winching motor have been improved to increase overall mechanical efficiency and the height of jumps. Finally, the internal components have all been modularized to decrease assembly and repair time from hours to minutes.

3 User Interfaces

A typical workstation-based Scout teleoperation controller is shown in Fig. 4. This interface runs on a Linux workstation or laptop, devices which can be difficult to carry around and use in the field.

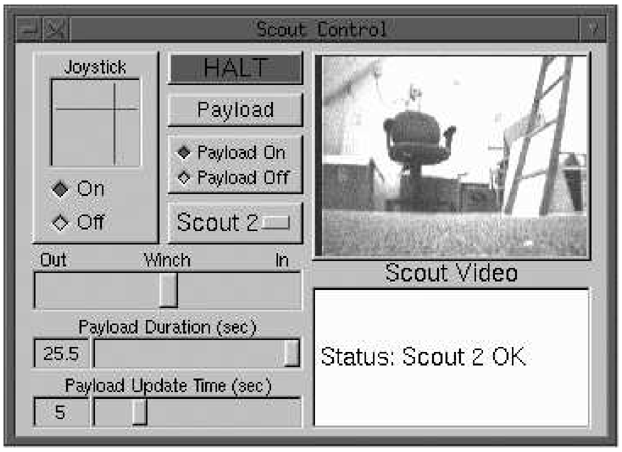


Figure 4: XWindow-based Scout controller.

To address the issue of portability, a wearable Scout package has been developed which contains a Scout radio, a video receiver, and an internal battery pack. To control the Scouts, a small commercial PDA is plugged into the portable radio pack. Control pads and buttons are displayed on the PDA, as shown in Fig. 5, and are activated using the touch-sensitive screen. Different robots can be controlled separately, allowing the user to manage a large team from one interface.

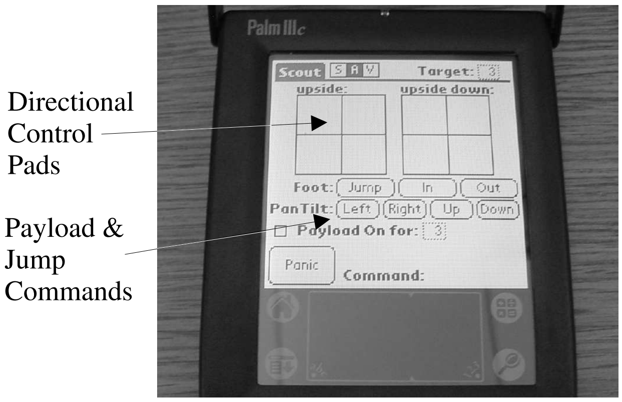


Figure 5: PDA interface.

An alternate interface based on a low-cost joystick for a commercial game console, shown in Fig. 6, has also been developed. This joystick has an ergonomic form which provides for more comfortable and intuitive use than the other interfaces. This joystick can have a Scout radio embedded within it to make it fully self-contained, or it can connect to the wearable Scout controller.

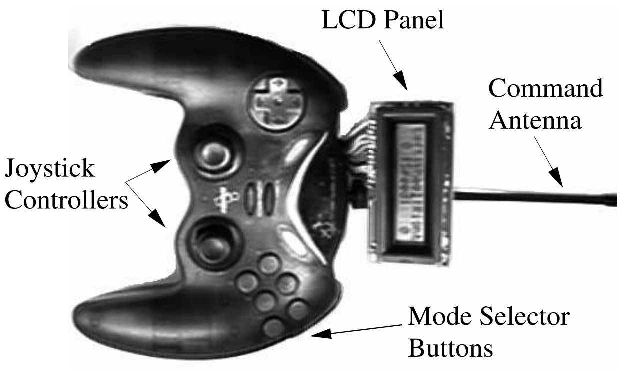


Figure 6: Joystick interface.

To allow for both autonomous and teleoperated control of a Scout from a wearable controller, a wearable PC with a framegrabber must be connected to the system. As shown in Fig. 7, the PC sits between the user interface and the wearable controller. The PC receives inputs from the joystick or PDA and sends the appropriate commands to the radio. Received video is captured by a framegrabber and is presented to the user via a head- or arm-mounted display. This captured video can also be used by autonomous Scout control programs running on the PC. If a wireless ethernet is added to the PC, the PC can communicate with other machines on the network and make use of the machine independence of the full software control architecture described in Sec. 4.

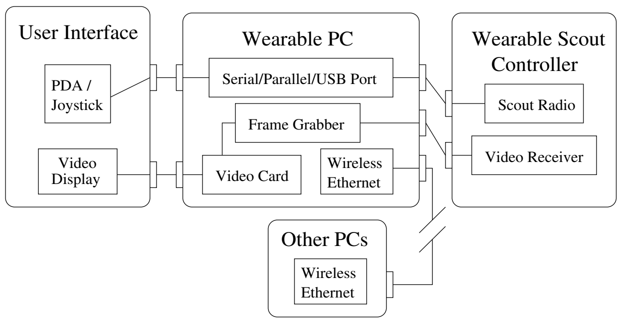


Figure 7: Connecting a wearable PC to the user interface and Scout radios.

4 Distributed Control Architecture

A distributed control architecture emphasizing the reusability of software components and the efficient utilization of resources has been designed (for a full description, see [13]). The objective is to support simultaneous control of multiple robots in both autonomous and teleoperated capacities. This architecture runs on a set of networked computers and can control an arbitrary number of robots. The software architecture consists of four subsystems: Mission Control, User Interface, Resource Pool, and Backbone.

All high-level robot control behaviors reside in the Mission Control subsystem. A mission is modeled as a high-level behavior that is recursively decomposed into simpler behaviors. Priorities are assigned which express a behavior's relative importance and are used to manage access to resources.

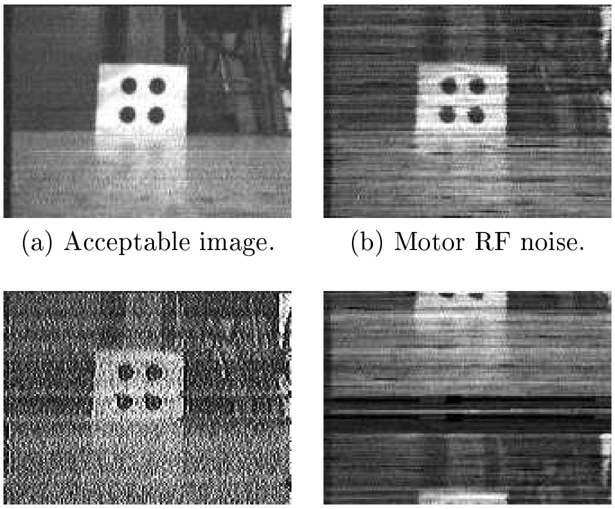
The User Interface subsystem serves two purposes: mission design and mission execution. In the mission design stage, it supports building complex behaviors from existing behavior primitives. During mission execution, an operator can view mission status, alter system parameters, and control robots from teleoperation consoles.

The Resource Pool subsystem provides access to resources (such as robots, radios, and framegrabbers) and ensures that the capacities of sharable resources are not exceeded. Multiple resource clients (teleoperation controllers or behaviors) that request simultaneous access to a resource will have their access controlled through a job scheduler.

The Backbone subsystem provides services which enable the other subsystems to work together. These services are capable of dynamically starting and stopping components, providing seamless communications across a variable number of networked machines, and trying to balance the system load over these machines. All inter-component communications are handled transparently by CORBA [4].

5 Visual Servoing

One of the challenges that must be addressed when writing autonomous behaviors for the Scout is how to handle signal corruption and noise in the image returned from the analog video link. Noise creeps into the system from the Scout's motors, multi-path reflections caused by the presence of obstacles around the robot, and weak signal strength caused by the large distance between the transmitter and the receiver. Fig. 8 illustrates how noise can affect the quality and clarity of the returned images. Finding ways to address this problem is extremely important as any perceptual system which operates in the real world must be able to recognize and correct for corrupted sensor data if it expects to operate correctly [6].



(a) Acceptable image. (b) Motor RF noise.

(c) Weak signal. (d) Loss of sync.

Figure 8: Effects of RF noise in Scout video.

In previous work [12], a simple frame averaging algorithm was used to reduce the effects of noise. This approach only dealt with the problem of spurious horizontal lines and white noise (Figs. 8(b) & 8(c))). If vertical synchronization was lost (Fig. 8(d)), averaging only compounded the noise. In this work, the navigation algorithms only used patches of light and dark areas as features.

Generally, the environments that the Scouts are expected to operate in will be noisy, cluttered, and highly unstructured. However, some aspects of the robot's immediate surroundings, such as a home base, or deployment platform, could be marked with an easy to detect landmark. The work described below allows a Scout to navigate to more structured landmarks, such as the four circles shown in Fig. 8(a).

The new navigation procedure consists of a target acquisition phase and a tracking phase. In the target acquisition phase, the Scout rotates in place to find a target. Grayscale images digitized from the Scout's camera are converted to binary images by applying a fixed threshold (dependent on the lighting conditions). The image is searched for a pattern which is consistent with a previously-seen target geometry.

Once a likely target is found, the Scout enters the tracking phase. Two models of different complexities are built which represent the positions of the target's features. The first model consists of a subimage containing the pixel values of the target. A second model consists of a subsampled set of pixels of the target. This set consists of the pixel values at the centers of the features as well as the pixel values surrounding each feature. This pixel set is referred to as the skeleton model of the target.

In the experiments, the target is a 2D image mounted perpendicular to the ground. The features that make up the target are arranged in a pattern such that there are at least two columns, each of which contains at least two features. The ratio between the distances between any two features in the columns and the sizes of the features is fixed. Each feature is the same size and shape as the others. As shown in Fig. 9, the target model is represented as a set of four circles which correspond to the positions of the target features. The "+"s show the positions of the pixels of the skeleton. The skeleton for this four-feature target consists of only seventeen pixels.

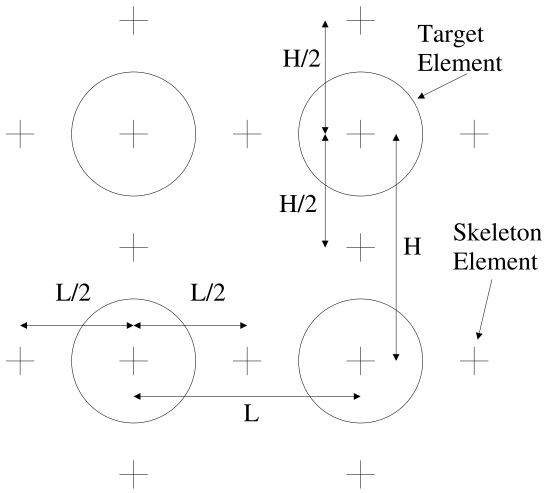


Figure 9: Elements of the Target and skeleton models.

In the tracking phase, the Scout moves towards the target. Because the camera's field of view is relatively narrow, the Scout can easily lose sight of the target if it rotates too far. Thus, it is imperative that the Scout tracks the target features in each successive frame. To achieve real-time performance, new frames are acquired as close to a rate of 30 fps as possible. This tracking speed is made possible by correlating only the skeleton model with each successive image. If a match is found, the target model is updated by extracting the target's new position from the image. If no match is found, the Scout stops moving and tries to correlate the last valid target model with the image (a more time consuming operation because of the larger number of pixels that must be compared). If the target model matches less than 70% of the image, it is assumed that the robot has turned such that its camera can no longer see the target and must return to the target acquisition phase.

Another difficulty that must be addressed is the problem of orienting the body of the Scout so that it faces the target head on. In many situations, the Scout needs to be aligned directly with the target. Such scenarios could include docking with a larger robot for pickup [8], or using a landmark for localization. If the angle of approach is not close to perpendicular, the robot will turn slightly and back up to get a better view before trying its approach again.

A sample run of the algorithm is illustrated in Fig. 10. In this run, the Scout starts to the left of the target. Its goal is to reach a point perpendicular to the target. Currently, experiments have been tried that start the robot upwards of two meters away from the target. Under uniform lighting, and barring battery and communications failures, the robot is able to reach the target each time.

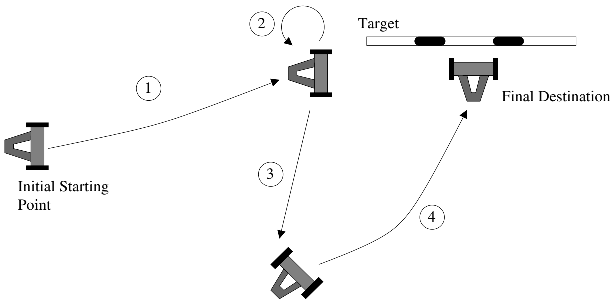


Figure 10: A sample multi-approach Scout trajectory. After a forward movement (1) which doesn't reach the desired position, the Scout rotates left (2) before moving backwards (3) to give it a better approach (4) to the target.

6 Related Work

Constructing robots that are small, easily deployable, and yet can do useful work and operate reliably over long period of times has proven to be quite difficult. Most miniature robots have wheels [2, 10], others can jump [5], fly [14], or swim [3]. So far, their use has been limited to research laboratories. Our Scout robots promise to be among the first miniature robots ready for field deployment.

To control a large group of robots, a software architecture must allow for distributed operations and facilitate allocation of resources. Architectures have been proposed to support fault-tolerant control of multiple robots [11], mission specification [9], and high-level task planning [1]. Much remains to be done before a general architecture is developed that is applicable to heterogeneous robots and to different tasks. The architecture we presented provides support for distribution of resources across robots, use of shared resources, and integrates in a seamless way autonomous and human-supervised control.

7 Conclusions and Future Work

We have presented a robotic system for reconnaissance and surveillance applications that is designed to operate in a semi-autonomous fashion. A human operator is able to remotely direct the robot to explore unknown areas as well as allowing the robot to do some of the tasks autonomously (such as returning to a marked pickup area). A set of different kinds of user interfaces has been described which allow humans to control one or more Scouts, depending on the complexity of the mission. A software architecture has also been presented which allows distributed communication to an arbitrary number of robots by teleoperation and autonomous control clients.

Future work will expand on the Scout's autonomous capabilities, which will include more advanced sensor interpretation and spatial reasoning techniques. The software control architecture is also being expanded to allow more types of hardware resources, such as larger robots, to be controlled.

References

[1] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-robot cooperation in the MARTHA project. *IEEE Robotics and Automation Magazine*, 5(1):36–47, Mar. 1998.

[2] G. Caprari, P. Balmer, R. Piguet, and R. Siegwart. The autonomous micro robot ALICE: A platform for scientific and commercial applications. In *Proc. of 1998 Int'l Symposium on Micromechatronics and Human Science (MHS'98)*, Nov. 1998.

[3] T. Fukuda, A. Kawamoto, and K. Shimojima. Acquisition of swimming motion by RBF fuzzy neuro with unsupervised learning. In *ALIFE V*, pages 31–37, 1996.

[4] O. M. Group. *The Common Object Request Broker: Architecture and Specification.* Object Management Group, 1998.

[5] A. Halme, T. Schönberg, and Y. Wang. Motion control of a spherical mobile robot. In *4th Int'l Workshop on Advanced Motion Control*, 1996.

[6] R. Hoffman and E. Krotkov. Terrain mapping for outdoor robots: Robust perception for walking in the grass. In *Proc. of the IEEE Int'l Conference on Robotics and Automation*, pages 529–533, 1993.

[7] D. F. Hougen, J. C. Bonney, J. R. Budenske, M. Dvorak, M. Gini, D. G. Krantz, F. Malver, B. Nelson, N. Papanikolopoulos, P. E. Rybski, S. A. Stoeter, R. Voyles, and K. B. Yesin. Reconfigurable robots for distributed robotics. In *Government Microcircuit Applications Conference*, pages 72–75, Anaheim, CA, Mar. 2000.

[8] J. P. Hyams and R. Murphy. Cooperative navigaion of micro-rovers using color segmentation. *Autonomous Robots*, 9(1):7–16, 2000.

[9] D. MacKenzie, R. C. Arkin, and R. Cameron. Specification and execution of multiagent missions. *Autonomous Robots*, 4(1):29–57, Jan. 1997.

[10] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Experimental Robotics III, Proc. of the 3rd Int'l Symposium on Experimental Robotics*, pages 501–513, Kyoto, Japan, Oct. 1993. Springer Verlag, London.

[11] L. E. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, Apr. 1998.

[12] P. E. Rybski, S. A. Stoeter, M. D. Erickson, M. Gini, D. F. Hougen, and N. Papanikolopoulos. A team of robotic agents for surveillance. In *Proc. of the Int'l Conf. on Autonomous Agents*, pages 9–16, Barcelona, Spain, June 2000.

[13] S. A. Stoeter, P. E. Rybski, M. D. Erickson, M. Gini, D. F. Hougen, D. G. Krantz, N. Papanikolopoulos, and M. Wyman. A robot team for exploration and surveillance: Design and architecture. In *The Sixth International Conference on Intelligent Autonomous Systems*, pages 767–774, Venice, Italy, July 2000.

[14] A. S. Wu, A. C. Schultz, and A. Agah. Evolving control for distributed micro air vehicles. In *Proc. IEEE Int'l Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, Nov. 1999.