

BUILDING A FAULT TOLERANT ARCHITECTURE FOR INTERNET ROBOTS USING MOBILE AGENTS

Liam Cragg, Pui Wo Tsui and Huosheng Hu

Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ
Email: {lmcrag, pwtstui, hhu}@essex.ac.uk

Abstract: *In Internet robot systems it is necessary to overcome the challenges of the limited bandwidth, indeterminate transmission delay, and potential data loss in order to make use of its cost effective functionality. With most systems focusing on these challenges, fault tolerance issues have not yet been addressed systematically in Internet robotics. The work described in this paper focuses on the development of a fault tolerant architecture for Internet robots taking into consideration the challenges imposed by the Internet as a communication mechanism. We aim to overcome these challenges and develop fault tolerance in our system by employing the mobile agent-programming paradigm.*

Key words: Fault Tolerance, Internet Robots, Mobile Agents.

1 Introduction

The Internet has made a significant impact on society through its use as a communication and data transfer mechanism. It is anticipated that in the future its impact will be seen in areas including remote embedded devices [1] and mobile robots that can be teleoperated by Internet users. It is expected that these systems will play a key role in hazardous environments such as the nuclear industry, underwater, and space, and also serve us at home.

Over recent years a number of Internet robot systems have been developed. These usually employ a single industrial robot arm (The Mercury Project [2], Tele-Garden [2], Australia's Telerobot on the Web [2]) or a wheeled mobile robot (KhepOnTheWeb [3], MAX the telerobotic dog [4], University of Essex Telerobot [5], Minerva [6], Xavier [2], P2PB Telerobot [7]). Some examples of multiple robot systems employing wheeled mobile robots can also be found (RobotOnWeb [2], and Multi-Robot Foraging [8]).

In these Internet robots, data loss could be reduced using reliable communication mechanisms such as TCP/IP although this protocol increases delay. The delay and limited bandwidth problems could be improved by increasing intelligence in a robot and preventing a user from engaging in direct low-level control, including Minerva [6], Xavier [2] and Vieira et al.'s Mobile Agent Enabled Internet Robot [9]. On the other hand, some systems ignore delay and limited bandwidth allowing users to control a robot by adopting their own control strategy to compensate for delay, such as the MAX, a teleoperated robot dog [4] and the University of Essex Telerobot [5]. Finally some systems adjust the ratio of user/robot control available in the system based on the level of Internet delay, including the P2PB

Telerobot [7] in which more control is available to the user during periods of low delay, and more control employed by the robot during periods of high delay.

Traditionally fault tolerant architectures can be found including those robot systems such as individual robots (Hannibal [10], SFX-EH [11]) and multiple autonomous robots (Alliance [12], and Murdoch [13]) and for single non-Internet teleoperated robots (Tele-SFX [14]). These systems employ redundant hardware or redundant/adaptive software in order to generate fault tolerance. Redundant hardware employs superfluous components within individual robots, or additional robot units to generate fault tolerance while redundant/adaptive software employs additional or adaptive robot control software components to generate fault tolerance.

The systems described above all work effectively to provide control in the robot system for which they were designed, however they did not need to consider challenges found in Internet robot control. We intend to address this deficit by developing a new architecture using mobile agents in which we address the fault tolerant challenges in Internet robot control. We intend to employ mobile agents as redundant/adaptive software components in our control architecture.

The rest of this paper is organised as follows. Section 2 describes some background information related to mobile agents. Section 3 presents our proposed framework, i.e. a application-centric mobile agent architecture. Section 4 addresses challenges in Internet robots and their control strategies. A simulation environment is proposed in section 5 for the implementation of our system. Section 6 proposes some preliminary experiments that could be undertaken to assess our architecture. Section 7 provides some conclusions and future directions.

2 Background – Mobile Agents

Mobile agents are moveable software objects that are reactive, autonomous and goal driven [15], they can be programmed to support asynchronous and autonomous operation, and perform dynamic and flexible adaptation in changing environments [16], as well as communicate or learn.

Mobile agents operate in an agent platform. This is an execution environment that forms a software layer between agents and underlying computer software/hardware regulating the level of access agents have to underlying computer services. Most mobile agent platforms have been developed using the Java programming language; agents are therefore able to make use of Java's platform independence

and portability properties. Agent platforms can run across groups of networked computers, providing a uniform distributed computing environment within which mobile agents can migrate and operate.

Agent platforms often employ the concept place; a place is a location at which agents reside within the agent platform. Each place usually refers to a single computer as shown in Figure 1, although multiple places can be hosted at a single computer.

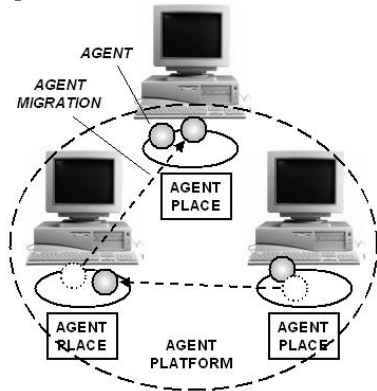


Figure 1 Mobile Agent Execution Environments

Mobile agents have characteristics that can make them useful in the Internet robot domain. More specifically,

- They can operate asynchronously and therefore lose contact with either a user or robot but remain functional aiding in overcoming delay or failure in an Internet WAN connection.
- They can examine their execution environment and adapt dynamically to change e.g. using services on local or remote hosts in a multiple robot setting a mobile agent could examine and make use of sensor or actuator functionality on multiple robots.
- They can be transferred across a network to process data at its source reducing communication bandwidth and latency and be used to encapsulate protocols, allowing teams of robots based on heterogeneous platforms to be used.
- Finally they can clone themselves or migrate away from points of failure in a system so that functionality can be maintained and alternative services sought, increasing system robustness.

3 Proposed Framework

Our aim is to employ mobile agents in an architecture so that we might use some of their beneficial characteristics. An application that uses mobile agent technology can be a platform based or application centric system [17]. This section is used to describe these in relation to the Internet robot context and explain our selection.

3.1 Platform Based Mobile Agent Systems

The architecture proposed by Vieira et al in [9] is shown in Figure 2. It employs a mobile agent platform encompassing two distributed locations. A user launches control and data

filtering agents from their local agent place. These then migrate to a remote agent place that has access to local robots functionality. Both agents conduct a number of operations locally at the remote site, one controlling the robot the other filtering data received from the robot that is then sent back to the user's location. The role of agent mobility in this instance is to reduce network communication, mobile agents move commands from the user to robot in one process.

A platform-based approach relies on the user installing an agent place at their location. This will prevent accessibility for many people who do not wish or are unable to install such a system. The use of mobile agents created externally and received at a local robot laboratory also potentially creates many security problems as mobile agents can arrive from any location and encapsulate potentially malicious functionality.

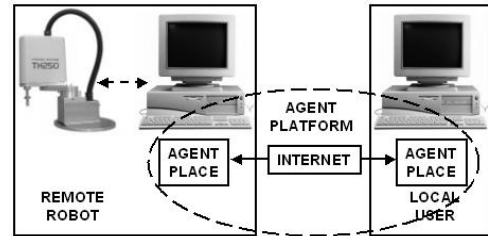


Figure 2 Platform Based Mobile Agent System

3.2 Application-Centric Mobile Agent Systems

We propose an agent architecture shown in Figure 3. It employs a mobile agent platform encompassing a number of distributed locations. However, unlike the platform based mobile agent system, not all application components are mobile agent enabled. Mobile agent places are created on robots and the server computer; however the user does not employ a mobile agent place at their location. The user does not launch mobile agents from their location, but sends commands, in the traditional client/server model seen in other Internet robot systems. When these commands reach the robot location, they are handled by agents.

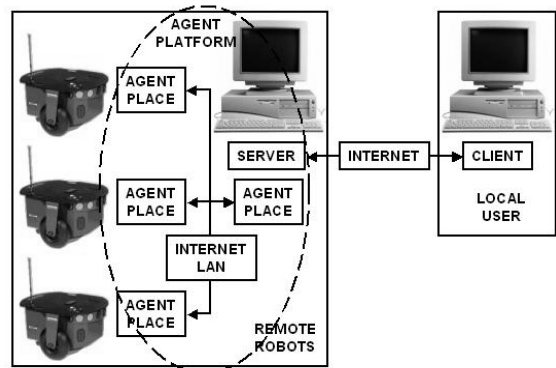


Figure 3 Application-Centric Mobile Agent System

Compact messages containing multiple commands can be sent to the agents that then process these locally into

individual commands. In the event that communication with the user is lost agents can employ these commands locally, they can also migrate between robots and the server computer. The role of agent mobility in this instance is to generate fault tolerance rather than reduce network communication load as in Vieira et al.'s work.

In the application-centric approach we propose there is no need for the user to install an agent platform with a user only needing to employ a standard web browser for control of robots as found in other Internet robot systems. In addition mobile agents are constrained from creation to termination in a local network therefore malicious agents which might try to migrate to the system from external sources could be prevented from entering the system or easily detected and removed.

4 Address Internet Robot Challenges

4.1 Fault Tolerance and Robot Control

In our system mobile agents are employed to provide fault tolerance in the system. These mobile agents are able to migrate or clone themselves between robot platforms at a remote site. This will allow control commands to robots and data from robots to be retained in the event that the functionality available in individual components in the system begins to degrade, including use of sensors and actuators across multiple robots.

The use of mobile agents is in keeping with the process of using more intelligence in remote robots in order to overcome the effects of delay and limited bandwidth in the Internet. In order to give a human user the opportunity of effective control of robots at a low level using their own intelligence e.g. for activities where fine control may be required.

A range of control methods should be made available in Internet robots, including both low-level and high-level control. Mobile agents are in general part of the control strategy that endows robots with more intelligence. It would be undesirable for mobile agent behaviour to conflict with user control; instead we want to create mobile agent behaviour which complements user control.

4.2 Control Strategy

It would be beneficial for an Internet robot to take into account both delay in the Internet and the level of control a user wishes to employ. In a system the level of control available to a user could be determined by measuring Internet delay in a similar manner to that found in the P2PB Telerobot [7].

In the event that delay is low the user could conduct low-level control, as delay increases so the user would be restricted to higher levels of control. This would appropriately shift intelligence in the system from the user to the robot as delay increases. As mobile agents are part of intelligence embedded in robots in the system so their activity should increase or decrease according to the level of intelligence required at the robot side. In the same way that

control adjusts to delay, so mobile agent activity should adjust to delay. As delay increases, so mobile agents should be able to take a more active role in providing fault tolerance in the system through adaptive migration, cloning, etc.

Mobile agent activities should reduce or provide the user with intelligent information to aid them in their decision-making. The behaviour of mobile agents should change automatically as Internet delay or the control mechanism employed by the user changes. Figure 3 shows what we perceived to be an appropriate general architecture for a mobile agent enabled Internet robot system.

Figure 4a shows the Client/Server part of a mobile agent enabled Internet robot system. This agent could provide two functions. Firstly it could monitor delay in the connection between Client and Server. Secondly it could distribute commands to robots. The server computer would supply the client with the user interface. The Communication Agent could therefore be used to update the user interface, thereby constraining controls as delay increases, and causing the user to control robots using mechanisms that are appropriate to the level of Internet delay.

Figure 4b shows the Robot/Server part of the system, and contains components that address fault tolerance behaviour challenge. It contains a number of agents, Robot Agents, Command Agents, Data Agents and Coordination Agents.

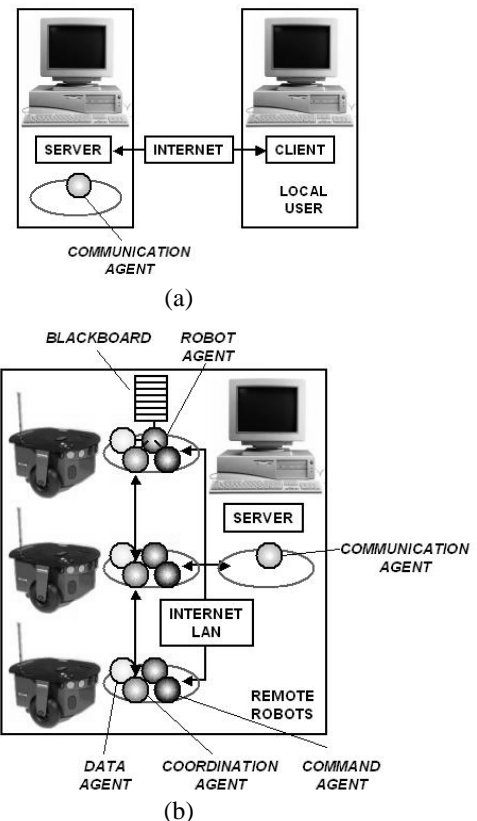


Figure 4 (a) Client/Server Components of Mobile Agent Internet Robot Architecture (b) Server/Robot Components of Mobile Agent Internet Robot Architecture

A Robot Agent handles command communication between the server and each robot. Commands are stored in a blackboard structure and passed on to the robot sequentially. The Robot Agent will send one command at a time and monitor feedback from the robot. This occurs for all commands in all levels of control. During normal execution data is passed back to the user via the server.

Coordination Agents can be used to facilitate behaviour in which the robots perform coordinated actions, in addition it allows robots to monitor each other's status to allow fault tolerant behaviour to be employed.

The main functions of Command and Data Agents are to generate fault tolerance in the system. Command Agents can be used to access the local blackboard of the local Robot Agent, then by copying data from the blackboard to other robots by generating clones of itself distribute commands to other robots. This would allow commands for a specific mission to be performed by at least one robot in a team, because it would ensure that all control commands required for all mission tasks would be distributed across the robot team at appropriate times. Data Agents can be used to store feedback data. In the event that communication with a user is lost or severely delayed the Data Agent could be used to process data for storage in a concise form that could later be relayed back to the user on reconnection or after a long delay to keep them updated as to activity in robots.

In a multiple Internet robot control system it is necessary to take into account, the role of the human user at run time, fault tolerance issues related to both robot functionality and the connection of the user to the system, and the challenges imposed by Internet delay.

5 Building a Simulation Environment

It is challenging to establish the exact nature of the behaviour of mobile agents within our dynamic distributed environment before practical examinations and development of the system are undertaken. Therefore our initial development has focused on the relationship between agents and how an agent platform should link robot to user; simulated robots have been used rather than real robots for ease of development.

TCP/IP communication is currently used for communication between all components. The user interface is not currently accessible via a web browser served from a web server but instead is employed locally as an application. The system currently comprised from three main components, the robot simulator, an agent platform and the user interface.

5.1 TeamBots™

The simulator we have employed is the TeamBots™ Robot Simulator [18]. This is a Java based simulator that allows simulation of multiple robots. Environments can be constructed containing obstacles, objects etc with which robots can interact. Teams of robots can be employed in the same environment with individual robots employing unique control programs.

Our system currently uses one or two robots in an environment. Each robot reads or writes strings to a dedicated port and maintains a TCP/IP connection to a local agent. An agent is allocated to communicate directly with each robot. The TeamBots™ simulator although representing up to two robots currently resides on a single computer. Agents with whom each robot can communicate reside on two additional computers.

The overall intention is that each real robot in a system would have an agent place in their control computer in order to host agents locally. The agent with whom the control program of each robot communicated would reside on the local computer.

Because it is desirable to test multiple robot teams in the same environment in which each robot recognises the existence of other robots a single simulation environment employing multiple robots on a single computer has been employed.

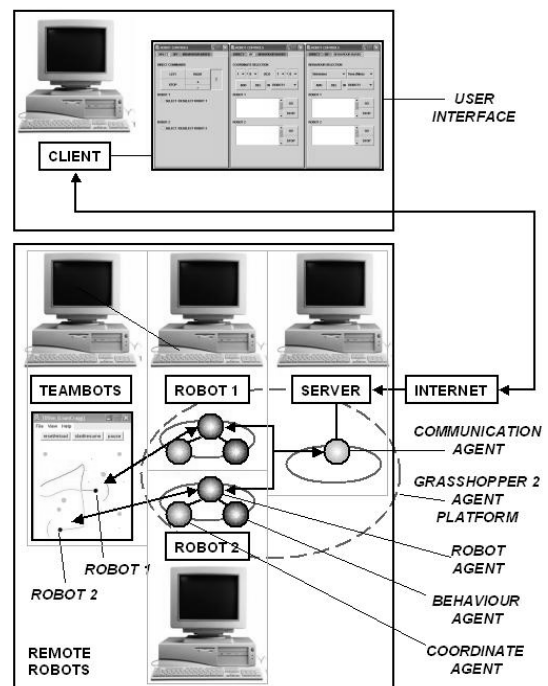


Figure 5 A Simulation environment

5.2 Grasshopper 2

The agent platform we have employed is Grasshopper 2 [19]. Grasshopper 2 is a Java based agent platform and can be used for creation of both mobile and static agents. Grasshopper provides the underlying execution environment in which agents can be created, and provides the underlying execution environment in which agents can be created. Grasshopper 2 provides a developer with built in functions for agent creation and manipulation. Some of these functions can be overridden allowing a developer to customise the behaviour of agents and employ their own control strategies within agents.

Built in functions allow a developer to move or clone agents over Grasshopper 2 enabled computers or store

agents to local persistent media such as a hard drive at regular intervals. This facility is described as persistence and in conjunction with mobility and cloning can aid in providing fault tolerant functionality.

5.3 Implementation

The overall structure of the current system can be seen in Figure 5. The location of the TeamBots™ Simulator and Grasshopper 2 Platform can be seen in this structure. A client currently has access to robot functionality via the user interface. This user interface offers a user one of three methods of control that also determine the autonomy of the robots in the system.

- Direct control allows a user low-level control of the robot using left, right, stop, speed up, and slow down buttons in which the robot has no autonomy. Therefore a tight control loop between user and robot is required. The control commands are forwarded directly to the robot by the Robot Agent and data returned via the Communication Agent to the user interface.
- Coordinate control allows a user medium level control of the robot in which a user can specify a list of Cartesian coordinates between which the robot should navigate. The robot is under supervised autonomy in this mode of control as it employs intelligence in reaching its goal and avoiding obstacles. In coordinate control a client may send an individual or list of coordinates to the robot. When these arrive at the Robot Agent they are forwarded to the Coordinate Agent that stores the list of coordinates and submits individual coordinate commands back to the Robot Agent to be forwarded to the robot. A control loop is maintained between robot, Robot Agent and Coordinate Agent as each coordinate is reached the Coordinate Agent is informed and sends each subsequent location. Data is also sent back to the user interface informing the user of robot progress in reaching each location.
- Behaviour based control allows a user high level control of robots in which a user can specify types of behaviour a robot should exhibit e.g. collect objects of a particular colour from the environment, in which the robot has near full autonomy. Each robot can only be controlled using one method at a time. Only one robot can be directly controlled at a time. It works in a similar manner to coordinate control with the Behaviour Agent storing a list of high-level behaviours submitted by the user. Once the user has submitted coordinate or behaviour lists to robots that can disconnect from the system, the coordinate or behaviour agents continue to submit commands on behalf of the client until all previously submitted commands have been executed or a robot fails.

Commands are sent from the client to a communication agent at the server every 100ms. The Communication Agent currently determines which robot should receive a particular command and forwards the command string on to the robot agent of the appropriate robot.

6 Preliminary Experiments

The system is not yet completed at this stage. Here, we show some preliminary experiments carried out using Teambots. The experiment is conducted over three different settings. Each setting is presented with the same condition and randomly placed attractors (attractors' number: 30). The results gathered are 20 trials in each setting. The average simulation performance is shown in Figure 6.

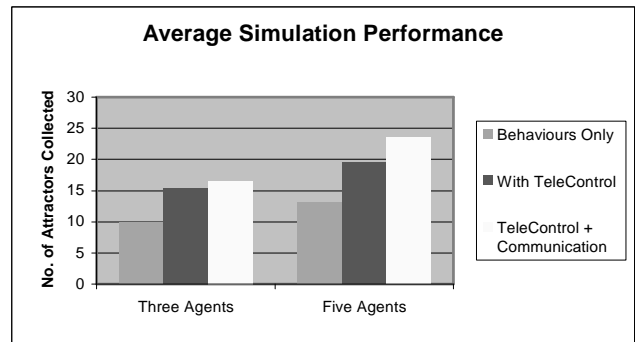


Figure 6 Average Performance of three different modes

A) Mode I: Built-in Behaviours Only

The goal of this experiment is to develop and gather the experiment results as references for analysis and comparison with later work, which is presented in the other two settings.

B) Mode II: Built-in Behaviours + Teleoperation

In Figure 7, we can see that there is a steady increase in performance. By taking a closer look, the change is possibly caused by the human factor. As the operator is getting more and more familiar with the control, he/she can easily achieve a much higher score than a hand-coded foraging agent.

Another possible cause is due to the fact that the operator is over looking the entire environment and has an advantage over the foraging agents. As our experience in developing the Internet telerobot system using Pioneer robots has shown it will be much harder with a local view (of which is the case for the foraging agents).

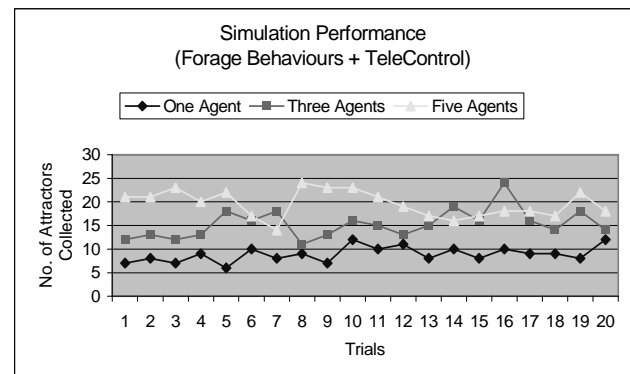


Figure 7 Simulation performance for Mode II

As can be seen from Figure 7, the system performance becomes unstable as the number of foraging agents increases. The rate of increased performance for each agents

group compared to results from behaviours only is: 7.900 (one agent), 0.537 (three agents), and 0.486 (five agents). This might be caused by the fact that operators have to look after several foraging agents, who may spread across the environment, simultaneously causing the control complexity increases.

C) Mode III: Built-in Behaviours + Teleoperation + Communication

By integrating communication strategy for collaboration, we can see a more stable increase in overall performance, when more foraging agents are inserted, compare to the approach using forage behaviours with TeleControl only (Figure 8). The forage agents naturally separate into subgroups by sharing sensor information with nearby agents, which causes them to possibly converge into separate groups in different locations. The operator can choose to control one robot to an attractor filled corner of the environment and cause the others near by to follow. This reduces the operator's control complexity.

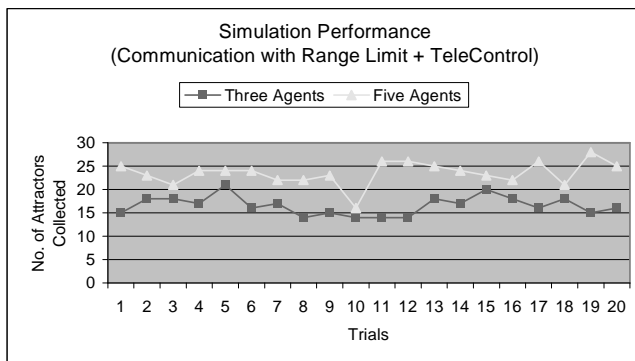


Figure 8 Simulated performance for Mode III

7 Conclusions and Future Work

The aim of the research described in this paper is to develop a fault tolerant Internet robot architecture that addresses issues related to Internet robot control, including limited bandwidth, indeterminate transmission delay, and potential data loss. The positive characteristics of mobile agents are adopted. The behaviour of the mobile agents and type of control available in the system will take into account delay inherent in the Internet communication mechanism, thereby ensuring that it addresses this Internet robot challenge.

The system is currently at a developmental stage using simulated environment rather than real robots. A number of mobile agents have been created for multiple robot control via simple user interfaces. Following stages of work include monitoring delay in the connection between user interface and agents, so that appropriate activity of the agents is performed due to changes in Internet delay or human control selection at run time.

Appropriate strategies for creating fault tolerance in the system also need to be solidified. When appropriate mobile agent behaviour and control strategies have been developed, the system needs to be made available to external users via a

web server as with other Internet robot systems. Following this experimentation needs to be conducted to assess that validity of the use of mobile agents for fault tolerance in multiple Internet robot systems.

References

- [1] D. Estrin, R. Govindan, J. Heidemann. "Embedding the Internet: Introduction", Communications of ACM, May, 2000
- [2] K. Goldberg and R. Siegwart, "Beyond Web Cams: An Introduction to Online Robots", The MIT Press, ISBN 0262072254, 2002.
- [3] P. Saucy and F. Mondada, "KhepOnTheWeb: Open Access to a Mobile Robot on the Internet", IEEE Robotics and Automation Magazine, March 2000.
- [4] A. Ferworn, R. Roque, and I. Vecchia, "MAX: Wireless Teleoperation via the World Wide Web", Proc. Canadian Conf. on Electrical & Computer Eng., Edmonton, Alberta, Canada, May, 1999.
- [5] H. Hu, L. Yu, P.W. Tsui, Q. Zhou, "Internet-Based Robotic Systems for Teleoperation", Int. J. of Assembly Automation, Vol. 21, No. 2, 2001.
- [6] D. Schulz, W. Burgard, D. Fox, S. Thrun and A. B. Cremers, "Web Interfaces for Mobile Robots in Public Places", IEEE Robotics and Automation Magazine, March 2000.
- [7] P.X. Liu, M.Q.-H. Meng, J. J. Gu and S. X. Yang, "A Study of Internet Delays for Robot Teleoperation using Biologically Inspired Approaches", International Journal of Robotics and Automation, Vol. 17. No. 4, 2002.
- [8] P. W. Tsui and H. Hu, "A Framework for Multi-Robot Foraging over the Internet", IEEE International Conference on Industrial Technology, Bangkok, 11-14 December 2002.
- [9] W. J. Vieira, L. M. Camarinha-Matos and L. Octavio Castolo, "Fitting Autonomy and Mobile Agents", 8th International Conference on Emerging Technologies and Factory Automation (ETFA 2001), Vol. 2, 2001.
- [10] C. Ferrell, "Failure Recognition and Fault Tolerance of an Autonomous Robot", Adaptive Behaviour, 2:4, 375-398, 1994.
- [11] R. Murphy and D. Hershberger, "Handling Sensing Failures In Autonomous Mobile Robots", International Journal Of Robotics Research, Vol. 18, No. 4, 1999, pp. 382-400, 1999.
- [12] L. E. Parker, "ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation", IEEE Transactions on Robotics and Automation, 14 (2), 1998.
- [13] B. P. Gerkey and M. J. Mataric, "Sold!: Auction Methods for Multirobot Coordination", IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, p758-768, 2002.
- [14] R. R. Murphy and E. Rogers, "Cooperative Assistance For Remote Robot Supervision", Presence, Special Issue On Starkfest, Vol. 5, No. 2, Spring 1996, pp. 224 -240, 1996.
- [15] D. B. Lange, "Mobile Objects and Mobile Agents: The Future of Distributed Computing?", Proc. of the European Conf. Object-Oriented Prog., 1998.
- [16] J. Baumann, "Mobile Agents: Control Algorithms", LNCS, Vol. 1658, Springer, June 2000.
- [17] P. Marques, P. Simoes, L. Silva, F. Boavida and J. Silva, "Providing Applications with Mobile Agent Technology", in Proceedings of IEEE Open Architectures and Network Programming Conference, 2001.
- [18] TeamBotsTM, "TeamBots-Homepage", World Wide Web, <http://www.teambots.org/>, Jan, 2003.
- [19] IKV++ Technologies AG, "Grasshopper 2 - Homepage", <http://www.grasshopper.de/>, Jan 2003.