

Application of Mobile Agents to Robust Teleoperation of Internet Robots in Nuclear Decommissioning

Liam Cragg and Huosheng Hu

Department of Computer Science, University of Essex
Wivenhoe Park, Colchester, Essex, C04 3SQ, U.K.
Email: {lmcrag, hhu}@essex.ac.uk

Abstract– Nuclear decommissioning involves the characterisation of hazardous and contaminated environments. Robot characterisation systems have been developed to reduce the risk to human operatives, however their efficiency is limited. Coming decades will see a substantial increase in decommissioning globally as a large number of nuclear facilities are due to reach the end of their useful life. It is desirable that robot characterisation systems meet this increase in demand by becoming more efficient. This paper describes an architecture that makes use of advances in computer science including mobile agent technology, which we believe will offer improved efficiency over existing robot characterisation systems.

1 Introduction

Nuclear energy production produces radiation that can damage the structure and normal function of biological cells and is extremely hazardous to living organisms. When nuclear facilities reach the end of their useful life it is necessary to remove these materials carefully, so that the site that they occupy can be returned to the natural environment or re-used for non-nuclear purposes. The process of converting an operational nuclear facility to a safe non-operational state is known as decommissioning. Decommissioning involves a number of processes including characterisation, decontamination, dismantling/demolition, and waste management.

Characterisation involves measuring, sampling, analysing, and characterising the type and location of radioactive substances in a facility using a variety of radiation measuring tools. Characterisation is extremely important because it allows the most appropriate and effective methods of decontamination, dismantling/demolition, and waste management to be subsequently applied and therefore affects decommissioning safety and cost. Characterisation may occur in unstructured environments with restricted access, and unknown geographical layout, physical, or radiological contents.

Existing robot characterisation systems include the Internal Duct Characterisation System (IDCS) [2], Remote Underwater Characterisation System (RUCS) [3], Mobile Autonomous Characterisation System (MACS) [4], Pioneer [5], and the Robotic Gamma Locating and Isotopic Identification Device (RGL&IID) [6]. These systems are mainly electrically powered, tethered and tele-operated by human operators although MACS uses Radio Communication/Autonomous Navigation and RGL&IID Wireless LAN Communication. These systems can be used to sample, measure and characterise radioactive substances in a variety of real-world environments: smooth/uneven

surface, in-pipe and underwater as they have different drive mechanisms, i.e. wheels, tracks, propellers.

They perform the characterisation process by either allowing visual inspection of hazardous environments (using sensors, especially cameras that also provide human operators with a remote view) or in the more sophisticated systems (e.g. Pioneer and MACS) by collecting data that can be used to construct detailed 2 or 3D computer models of examined areas. Although some of these systems have been used in nuclear facility decommissioning they are in general limited by the following factors: (i) very limited robot autonomy; (ii) low sensor intelligence and online characterisation functionality.

By 2010 it is anticipated that approximately 300 nuclear power plants in global operation will require decommissioning [1]. This will require a substantial increase in decommissioning. Advances in computer science techniques may help to improve efficiency, cost and effectiveness of existing robot characterisation systems, however challenges must be overcome in order to meet the needs of this domain. This paper examines these challenges and describes a novel approach employing Mobile Agent technology.

The rest of this paper is organised as follows. Section 2 describes available technologies that might aid in the development of a novel robot characterisation system architecture. Section 3 describes our proposed architecture. Section 4 describes preliminary experiments to support or design selection. Section 5 provides conclusions and describes future work.

2 Available Technologies

This section examines available technologies for development of a multiple robot characterisation system.

2.1 Distributed Computing Architectures

A number of alternative architectures can be used in distributed computing applications. These include:

Client/Server Architecture – As shown in Fig. 1(a), a Client doesn't contain knowledge (K) or resources (R) to perform a task. A Server contains both the knowledge and resources to perform the task. The Client sends a message to the Server requesting the performance and result of the task. After performing the task using its local knowledge and resources, the Server returns the result to the Client.

Remote Computation Architecture – As shown in Fig.1(b), a Client contains knowledge but not the resources to perform a task. A Server contains resources but not the knowledge to perform the task. In order for the Client to obtain the result of the task, it sends its knowledge to the Server. After performing the task using the Client's

knowledge and its local resources, the Server returns the result to the Client.

Code on Demand Architecture – As shown in Fig. 1(c), a Client contains resources but not the knowledge to perform a task. A Server contains knowledge but not the resources to perform the task. In order for the Client to obtain the result of the task, it sends a message to the Server requesting that it sends it knowledge to perform the task. Using the Server’s knowledge and its local resources, the Client performs the task and generates a result locally.

Mobile Agent Architecture – As shown in Fig. 1(d), an Agent Server is host to a Mobile Agent, which contains knowledge and results from previous tasks, but not the resources to perform a new task. Another Agent Server contains resources to perform the task the Mobile Agent would like to perform. Instead of forwarding knowledge alone to the new Agent Server, the Mobile Agent itself is able to move in order to access the resource. When the Mobile Agent has completed its task it can return to its original Agent Server or continue to another Agent Server, carrying its knowledge and the result of executing its task.

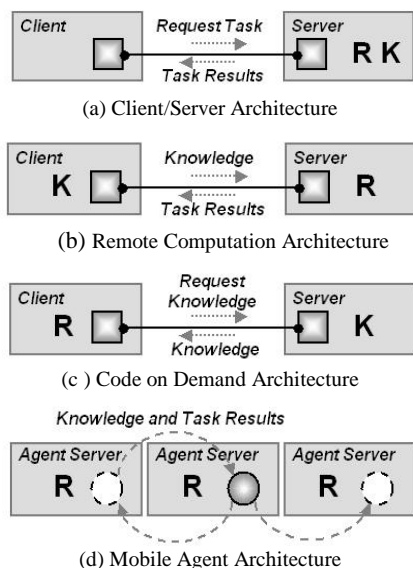


Fig. 1 Distributed Computing Architectures

2.2 Autonomous Multiple Robot Architectures

This section examines architectures developed for the control of autonomous multiple robot teams. Two such architectures are ALLIANCE [7] and MURDOCH [8] both of which were designed for the efficient fault tolerant allocation of tasks.

ALLIANCE – employs artificial motivation for task allocation. Robot team members contain artificial motivations that include characteristics such as impatience and acquiescence to control a robot’s underlying behaviour. Artificial motivations are affected by robot sensor input, the behaviour a robot is executing and the behaviour of other robots (in addition to other parameters).

Of the inputs each robot receives, the behaviour of other team members has most significance when a robot determines its own behaviour. Robots inform each other via inter-robot communication of their current activity. This leads robots to choose (in general) alternative tasks

from each other to perform, thereby distributing task allocation of simultaneously executable behaviours across a team. ALLIANCE is distributed, applicable to heterogeneous robot teams, uses one-way broadcast communication, recovers from failures in robots and communication, allows new robots to be added at any time and allows adaptive action selection in dynamic environments. It eliminates replication of effort when communication is available, provably terminates for a large class of applications and will scale to large missions, but is restricted to missions in which robots employ loosely coupled subtasks.

MURDOCH – employs an auction based task allocation mechanism, in which a task allocator robot announces a task to a team of robots. Robots assess their fitness to conduct the task by performing a metric evaluation. They then return the result of their evaluation as a bid in a bid submission to the allocator robot. The allocator robot closes the auction and evaluates the bids. It then allocates the task to the robot with the best bid. The allocating robot continues to monitor the execution of the task by its selected robot and renews its execution, while the robot is performing the task successfully. Although task allocation occurs from a central location, the task allocator may be located on any robot thereby ensuring the system is both distributed and fault tolerant. MURDOCH is distributed, applicable to heterogeneous robot teams, employs robot-robot communication in which messages may be lost, can handle partial or complete robot failure, and doesn’t need a model to describe the sequence in which tasks are generated.

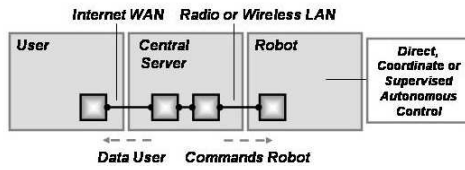
2.3 Internet Robot Architectures

Internet Robot Systems allow a user to control a robot at a remote location using the Internet as a communication mechanism. They often employ the TCP/IP Internet Communication Protocol for this purpose.

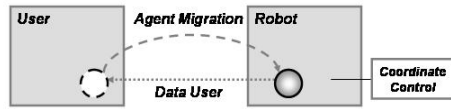
Single Internet Robot Architectures – Most Internet Robot Systems employ a Client/Server Single User/Single Robot Architecture as shown in Fig. 2(a). In such an architecture, a user sends commands to a remote computer (**Central Server**), which then forwards these commands to a robot and its control computer (remote computer-robot communication can occur via Radio or Wireless LAN). In return robot data is sent from the robot to the user. Single Internet Robot Systems have been developed that employ Low-Level (Direct) Control e.g. Essex Telerobot [9], Medium Level (Coordinate) Control e.g. UWA Telerobot and High-Level (Autonomous) Control e.g. Xavier [10].

In Single Internet Robot Systems, consideration for the effects of Internet delay on control are either ignored, with adjustment of control strategy due to delay left to the user of the system e.g. KhepOnTheWeb [10], or may dynamically adapt the ratio of human/robot control as delay varies, as found in P2PB Telerobot [11].

An alternative architecture for a Single Internet Robot System is shown in Fig. 2(b). This architecture employs Mobile Agents. In this system, described in [12], a Mobile Agent migrates to a remote robot via the Internet. It conducts a sequence of planned i.e. declarative actions locally on the robot and transmits data back to the user.



(a) Client/Server Single User/Single Robot Architecture



(b) Mobile Agent Single User/Single Robot Architecture

Fig. 2 Internet Robot Architectures

The system overcomes command execution delay by dynamically moving all commands to the robot to execute locally thereby avoiding the limited bandwidth and indeterminate transmission delay characteristics of the Internet.

Multiple Internet Robot Architectures – A Client/Server Single User/Multiple Robot Internet Robot Architecture can be found in [13]. In [13] a user can control a team of robots in a foraging mission. The user can control one robot at a time using Low-Level (Direct) Control commands, while other robots engage in High-Level (Autonomous) foraging. A Multiple User/Multiple Robot Internet Robot Architecture (RobOnWeb) can be found in [10]. In this architecture multiple users may control one and only one of multiple robots using Low-Level (Direct) Control.

The majority of Internet Robot Systems employ a Client/Server Single User/Single Robot Architecture although some Multiple Robot and Mobile Agent Based Architectures have been developed. In these architectures, the behaviour of robots is in general not complex, with communication occurring with simple message passing over Internet communication channels. Most system developers have focused on the construction of required hardware/software functionality for Internet control i.e. connection of web browser to a Central Server and mechanisms for feeding commands to a robot and receiving data feedback for a user, rather than complex robot functionality.

2.4 Discussion

Distributed Computing Architectures include Client/Server, Remote Computation, Code on Demand and Mobile Agent Architectures. Unlike the other methods, which use extensive inter-location communication, Mobile Agents move to the source of a resource and conduct communication locally in order to make an application function. In a low bandwidth, indeterminate transmission delay communication mechanism like the Internet this should allow more successful application execution.

Autonomous Multiple Robot Architectures including ALLIANCE and MURDOCH are effective for fault tolerant allocation of tasks to members of multiple robot teams. Both have been extensively tested using real robots in a number of practical applications. However both employ simple message passing mechanisms not designed

for large-scale robot-robot or robot-user data transfer and were originally designed for task allocation in autonomous robots, not for teleoperation.

Internet Robot Architectures including University of Essex Telerobot, KhepOnTheWeb, and UWA Telerobot have been successfully developed to explore the required hardware/software functionality for Internet control. However such systems generally employ simple message passing mechanisms and lack the use of complex functionality found in other areas of robotics.

3 Proposed Architecture

This section begins with an overview of the proposed architecture. A detailed description of architectural components is then provided in subsequent sections.

3.1 Proposed Architecture Overview

The overall structure of the proposed architecture contains: User Computer (U), Multiple Robots (R^1-R^n), Control Agents, Planning Agents, Mapping Agents, User-Robot (TCP/IP) Communication, Robot-Robot (Agent Messaging) Communication, and Mobile Agent Migration. The User Computer allows a user to control or supervise one or more robots, as shown in Fig. 3

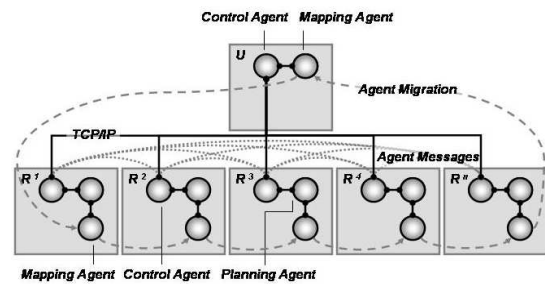


Fig. 3 Proposed Architecture Overview

3.2 Control Agent

Control Agents determine the current activity of robots on which they are located. The internal structure of a Control Agent is shown in Fig. 4.

Control Agent (Structure) – For task allocation Control Agents make use of the ALLIANCE architecture. Control Agents encapsulate an ALLIANCE architecture in which the following Motivational Behaviours (MB) and Behaviour Sets (BS) are located: User Controlled (Safe-Stop ($Safe$)); Low-Level (Direct) Control ($LLDC$); Medium-Level (Coordinate) Control ($MLCC$) and Autonomous Robot Controlled (Mapping (Map), Recharging ($Rech$), Homing ($Home$) and Stationary Behaviour ($Stay$)). The motivation to conduct these behaviours is based upon both internal and external parameters, including User, Inter-Robot ($InterR$) and Intra-Robot ($IntraR$) communication as well as Sensor feedback.

Control Agent (Behaviour) – Safe-Stop shown in Fig. 5 allows a user to send a stop command using Mobile Agent Messaging from the User Computer to all robots simultaneously. Receipt of a Safe-Stop Command by any robot will motivate it to select its Safe-Stop Behaviour Set causing it to cease all activity.

Low Level (Direct) Control shown in Fig. 6 allows a user to control a single robot at a time using Low-Level (Direct) Commands e.g. left, right, forward etc. The Control Agent on each robot will maintain an open ServerSocket using a known Port.

If the user wishes to control a specific robot, a TCP/IP connection will be made between the user and the robot using the unique IP Address/Port Number combination to identify the robot. If a TCP/IP connection to a robot is created this will motivate the associated robot to select its Low-Level (Direct) Control Behaviour Set.

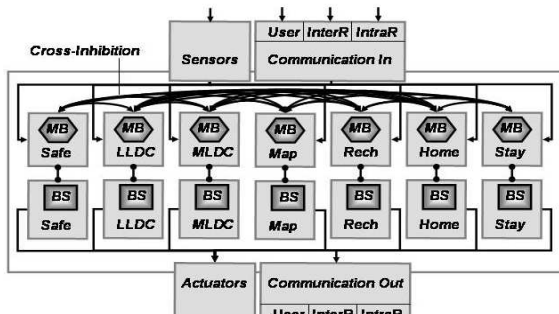


Fig. 4 Control Agent

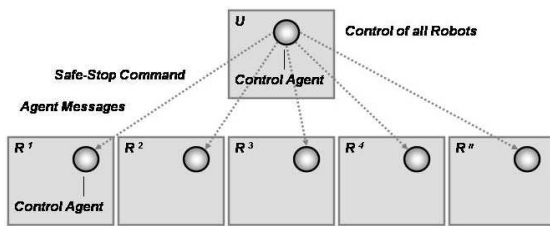


Fig. 5 Safe-Stop

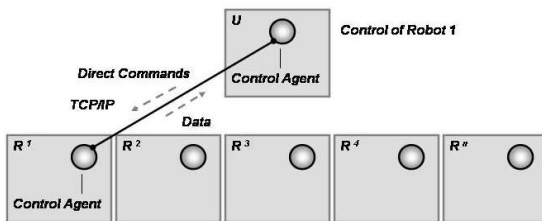


Fig. 6 Low Level (Direct) Control

Medium Level (Coordinate) Control shown in Fig. 7 allows a user to control one or more robots using Medium Level (Coordinate) Commands e.g. x, y, th. The user can send Coordinate Commands in list form to robots using Mobile Agent Messaging. These Coordinate Commands will be executed sequentially on the robot/s to which they are sent. Receipt of a list of Coordinate Commands by a robot will motivate it to select its Medium Level (Coordinate) Control Behaviour Set.

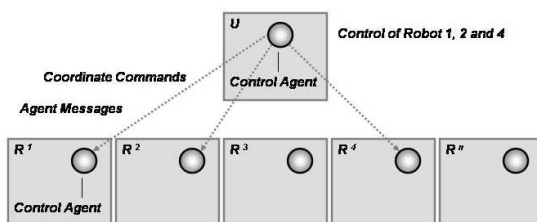


Fig. 7 Medium Level (Coordinate) Control

High Level (Autonomous) Control shown in Fig 8 allows a robot to select from one of four Behaviour Sets, these are Mapping in which a robot Maps/Explores (Characterises) the environment; Recharging in which the robot returns to a recharging station to recharge; Homing in which a robot returns to a home location (for maintenance or inspection etc); or Stationary in which a robot remains stationary in appropriate circumstances. The motivation for a robot to select a particular High-Level (Autonomous) Control Behaviour Set will depend upon the control requirements of a user, determined by User-Robot Communication, the activity of other robots, determined by Inter-Robot Communication, and internal parameters determined by Intra-Robot Communication.

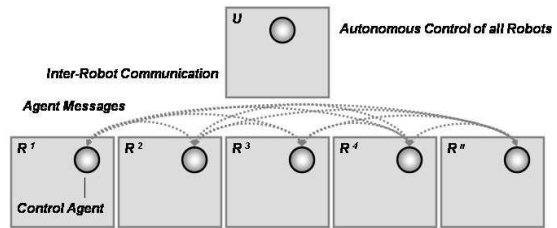


Fig. 8 High Level (Autonomous) Control

3.3 Planning Agent

Planning Agents help robots to determine an effective long-term behavioural strategy. The internal structure of a Planning Agent is shown in Fig. 9.

Planning Agent (Structure) – Planning Agents contain a Short Term Map (*STM*), a Long Term Map (*LTM*), a Mapping Planning Module (*MPM*), a Recharging Planning Module (*RPM*), and a Robot Status Module (*RSM*). These modules interact with each other and with Control/Mapping Agents via Intra-Robot Communication. Short Term Maps contain data on a robot's recently explored environment provided by a Control Agent. Long Term Maps contain data on a robot's wider environment provided by a Mapping Agent. The Mapping Planning Module establishes if and in which direction a robot should explore based on its Short and Long Term Map (using Occupancy Grid based Maps that will allow Frontier-based Exploration Algorithms [14] to be employed).

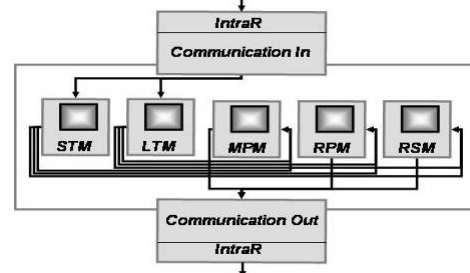


Fig. 9 Planning Agent

The Recharge Planning Module establishes if and how a robot should return to a recharging station. Its decision to recommend recharging or not will be based on a calculation of a planned path back to the recharging station. This module will employ Reinforcement Learning to establish the optimum time to recharge, taking into account distance from recharging station and battery power level.

Fig. 10 shows an example finite Markov Decision Process (MDP) Transition Graph showing probabilities and rewards of action selection/state in relation to this activity. The Robot State Module is used to assess the gross functionality of the robot based upon the expected versus actual behaviour of a robot. This module will contain information including approximate state of robot sensors, actuators and battery.

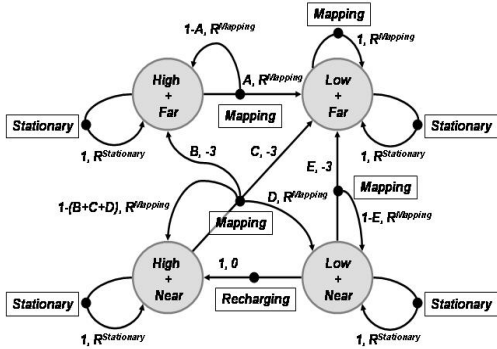


Fig. 10 Finite MDP Transition Graph for Recharge Planning Module

Planning Agent (Behaviour) – Planning Agent Behaviour is shown in Fig. 11. The Planning Agent remains stationary for most of its life and conducts Intra-Robot Agent-Agent Communication with local Control and Mapping Agents

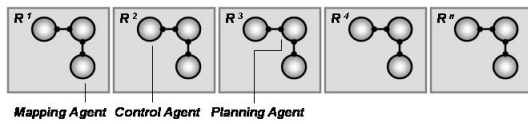


Fig. 11 Planning Agents Intra-Robot Communication

3.4 Mapping Agent

Mapping Agents allow a consistent Long Term Map to be constructed in all robots, and provide a user with a Characterisation Map developed On-line. The internal structure of a Mapping Agent is shown in Fig. 12.

Mapping Agent (Structure) – Mapping Agents contain a Short Term Map for each robot they visit (R^1M-R^nM), a Robot Status Module for each robot they visit (R^1S-R^nS), a Map Integration Module (MIM) and a Migration Planning Module (MPM). These modules interact with each other and with Planning Agents within a robot using Intra-Robot Communication.

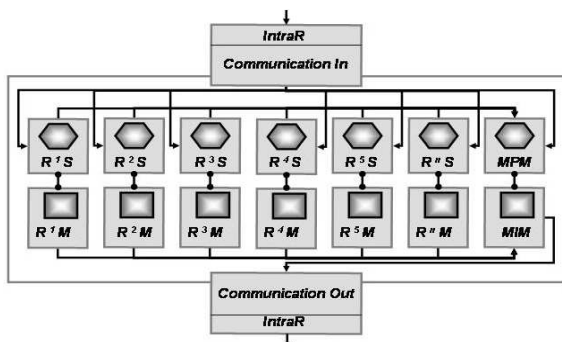


Fig. 12 Mapping Agent

The Map Integration Module allows the Mapping Agent to update the Long Term Map of each robot it visits based

on the Short Term Maps for each robot that it has stored. Because maps will be Occupancy Grid Based, Bayesian or Dempster-Schafer Sensor Fusion Algorithms can be applied [14]. This will allow the sensor information from each distributed robot to be fused together into a cohesive map that should be more accurate than that which can be developed using a single robot. Bayesian and Dempster-Schafer Algorithms can be used to fuse together sensor information from different sensor mechanisms e.g. sonar, laser, or vision.

The Migration Planning Module allows the Mapping Agent to determine which robots to visit based on the Robot Status Modules for each robot that it stores. The Mapping Agent will learn to adjust its migration behaviour at runtime. Using Reinforcement Learning the Mapping Agent will be able to ignore or leave failing robots allowing it to optimise its behaviour. Fig. 13 shows an example finite Markov Decision Process (MDP) Transition Graph showing probabilities and rewards of action selection/state in relation to this activity.

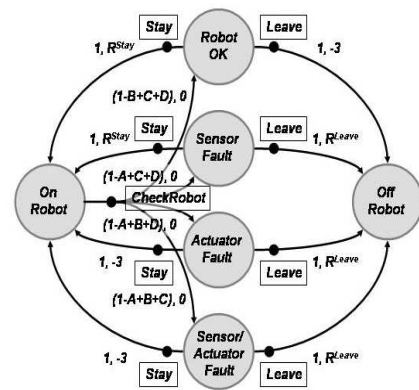


Fig. 13 Finite MDP Transition Graph for Migration Planning Module

Mapping Agent (Behaviour) – Is shown in Fig. 14. A Mapping Agent will successively migrate between user and robots collecting recent Short Term Map and Robot Status Data and updating robot's Long Term Maps.

The Mapping Agent will clone itself successively as it migrates, leaving a copy of its most recent incarnation behind. This produces a distributed and fault tolerant architectural structure.

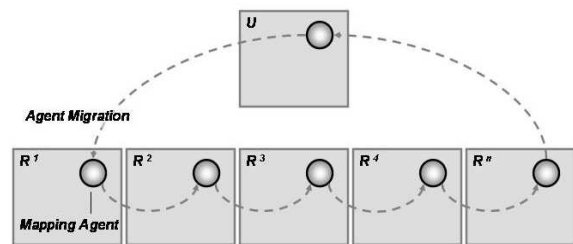


Fig. 14 Mapping Agent Migration

3.5 Discussion

The architecture described in this section provides fault tolerance at a robot mission level through the use of ALLIANCE for action selection, and architecturally through the use of Mobile Agents, using the migration, cloning and persistence (storage to persistent media) characteristics of Mobile Agents to retain information vital to the robot mission (in this case characterisation data).

4 Preliminary Experiments

We conducted some preliminary experiments in order to assess the effectiveness of alternative robot control mechanisms under varying levels of communication delay.

We developed a system in which a user could control up to five simulated robots simultaneously using Low Level (Direct) Control (left, right, forward etc), Medium Level (Coordinate) Control (x, y, th), or High Level (Autonomous) Control (foraging). The simulated robots were constructed using TeamBots™ Robot Simulator [16].

Foraging and mapping both require physical robot action at distributed locations and benefit from distributed action at a distance. Foraging was selected because it is simpler to record the number of objects collected from, (than calculate the area of), a mapped environment.

We conducted a number of experiments in which we employed teams of 1, 3 and 5 robots. We introduced communication delay between user and robots. We employed no, low (400ms), medium (2000ms) and high (4000ms) levels of communication delay, to simulate Internet communication delay variations. We controlled the robot/s using all three control types, individually, and in combination. We conducted 20 trials for each control type/delay combination and recorded the number of objects (attractors) collected from the environment. The results are shown in Figure 15.

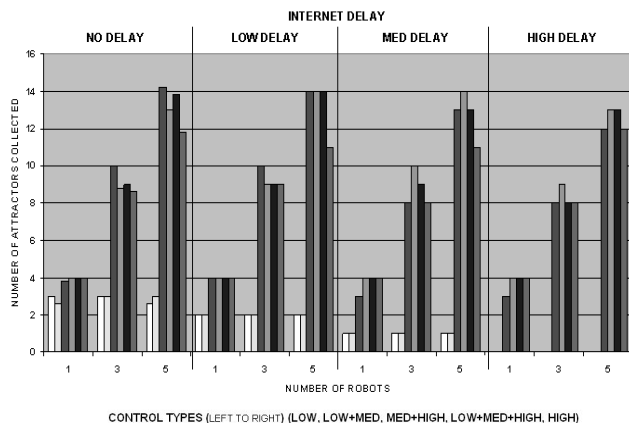


Fig. 15 Preliminary Experiment Results

These results show High Level (Autonomous) Control to be more efficient than Low Level (Direct) Control in all delay conditions. Low Level (Direct) Control can only be effectively employed in low delay conditions. As the number of robots employing High Level (Autonomous) Control increases the number of attractors collected increases. As the number of robots employing Low Level (Direct) Control increases the number of attractors collected does not increase. This occurs because a single user can only effectively control a single robot at a time using Low Level (Direct) Control. For optimum task efficiency High Level (Autonomous) Control mechanisms are required even in low delay conditions. The result of these experiments supports our intention to employ more High Level (Autonomous) Control functionality in our robots than found in existing robot characterisation systems.

5 Conclusion and Future Work

This paper has proposed a novel architecture to allow limited functionality of existing robot characterisation systems to be extended to meet the needs of increases in future decommissioning. The architecture combines the strengths of available Distributed Computing, Autonomous Multiple Robot, and Internet Robot Architectures while overcoming their individual weaknesses in this domain. Such an architecture will be fault tolerant, scalable, and adaptable, overcome the impact of delay and limited bandwidth characteristics of internet communication, resolve sensor data conflicts between robot team members, provide a more accurate characterisation map than possible with a single robot and be more efficient. Future work will involve experimentation to assess functional characteristics of the architecture in experiments with real robots.

References

- [1] European Website on Decommissioning of Nuclear Installations, "Decommissioning in Short", 2003. <http://www.eu-decom.be/about/initabout.htm>
- [2] US Department of Energy, "Internal Duct Characterisation System", Innovative Tech. Summary Report (Tech ID 42), 2002. http://www.robotics.ost.doe.gov/reports/idcs_itsr.pdf
- [3] W.D.Willis et al., The Remote Underwater Characterization System, Proceedings of the American Nuclear Societies 8th Topical Meeting on Robotics and Remote Systems, 1999.
- [4] B.S.Richardson et al., A Mobile Automated Characterization System (MACS) For Indoor Floor Characterization, Proc. the American Nuclear Societies 6th Topical Meeting on Robotics and Remote Systems, 1995.
- [5] A.Slifko et al., "Pioneer: A Robot for Structural Assessment of the Chornobyl Shelter", In Proceedings of the American Nuclear Societies 8th Topical Meeting on Robotics and Remote Systems, 1999.
- [6] M.Anderson et al., "Demonstration of the Robotic Gamma Locating and Isotopic Identification Device", Proceedings of the American Nuclear Society Spectrum, 2002.
- [7] L.E.Parker, "ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation", IEEE Transactions on Robotics and Automation, 14 (2), 1998.
- [8] B.P.Gerkey and M.J.Mataric, "Sold!: Auction Methods for Multi-Robot Coordination", IEEE Transactions on Robotics and Automation, 2002.
- [9] L.Yu et al., "A Web-based Telerobotic System for Research and Education at Essex", IEEE/ASME Int. Conference on Advanced Intelligent Mechatronics, 2001.
- [10] K.Goldberg and R.Sieglwart, "Beyond Webcams: An Introduction to Online Robots", MIT Press, 2001.
- [11] P.X.Liu et al., "A Study of Internet Delays for Robot Teleoperation using Biologically Inspired Approaches", International Journal of Robotics and Automation, 2002.
- [12] L.M.Camarinha-Matos et al., "A Mobile Agents Approach to Virtual Laboratories and Remote Supervision", Journal of Intelligent and Robotic Systems, 2002.
- [13] P.W.Tsui and H.Hu, "A Framework for Multi-robot Foraging over the Internet", Proceedings of IEEE International Conference on Industrial Technology, 2002.
- [14] R.R.Murphy, "Intro. to AI Robotics", MIT Press, 2000.
- [15] R.C.Arkin, "Behaviour-Based Robotics", MIT Press, 1998.
- [16] TeamBots™, "TeamBots Homepage", Sept 2003. <http://www.teambots.org/>