

Agent Architecture for Multi-robot Cooperation over the Internet

Huosheng Hu, Pui Wo Tsui, Liam Cragg, Norbert V?lker

*Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, UK
Tel: (+44)-1206-872297; Fax: (+44)-1206-872788*

Email: hhu@essex.ac.uk, pwtsui@essex.ac.uk, lmcrag@essex.ac.uk, Norbert@essex.ac.uk

Abstract

In real-world applications, Internet or online robots require a high degree of autonomy and local intelligence to deal with the uncertainties in both their environments and the Internet. This paper describes our progress in building a new framework for remote cooperation of multiple online robots, which can be accessed through any Java-enabled browsers. We present an approach to the cooperation of multiple robotic agents remotely through the Internet using a low cost and portable solution. A simulated experiment on multi-robot foraging is discussed. The experiment results show that the proposed approach is easily extended to include more functionality and more robotic agents. The local intelligence and communication of remote robotic agents provides simpler control, stability and steady performance.

Keywords: Internet/Online Robots, Web-based systems, Multi-robot Cooperation, Teleportation, Mobile Agents

1. Introduction

Internet technology provides us with a convenient way to develop the diversified applications of online robotic systems. Recently, more and more intelligent devices or systems have been embedded into the Internet for service, security and entertainment, including distributed computer systems, surveillance cameras, telescopes, manipulators and mobile robots. These Internet based devices or systems [17] have captured the interests of many researchers worldwide. Apart from operations in hazardous environments that are traditional telerobotic areas, Internet or online robots have opened up a completely new range of real-world applications, namely tele-manufacturing, tele-training, tele-surgery, museum guides, traffic control, space exploration, disaster rescue, house cleaning and health care [13][30].

Although the Internet provides a cheap and readily available communication channel for tele-operation, there are still many problems that need to be solved before successful real-world applications can be achieved. These problems include its restricted bandwidth and arbitrarily large transmission delay, which influence the performance of the Internet-based telerobotic systems. Therefore, it is necessary to remove human operators from the feedback control loop, and equip the robots with a high degree of local intelligence in order for them to autonomously handle the uncertainty in the real world and the arbitrary network delay. Also, an intuitive user interface is required for inexperienced people to control the robot remotely. The reliability of the system should be guaranteed so that Internet users can access the Internet robotic system 24 hours a day with minimum human maintenance.

Until now, most of Internet or online robotic systems contains a single robot that is controlled remotely by the web users. However, many real-world applications require multiple Internet robots to accomplish particular tasks. As the number of robots in a system increases, planning, control and communication of the system becomes increasingly complex. Therefore a mechanism for cooperation and fault tolerance is required.

To realise the cooperation of multiple mobile robots, four main issues need to be addressed: (i) how to appropriately divide the functionality of the system into multiple robots; (ii) how to manage the dynamic configuration of the system in order to realise cooperative behaviours; (iii) how to achieve multi-robot cooperation over the Internet; and (iv) how to achieve fault tolerance in a distributed manner. This paper is to address these issues using agent technology.

The rest of paper is organised as follows. Section 2 gives a brief overview of previous work on Internet or online robots. Section 3 presents a new framework for multi-robot cooperation over the Internet. Section 4 proposes the agent architecture for the implementation of the proposed multi-robot Internet robotic systems. Section 5 presents the implementation of the proposed framework in a simulated environment at this stage of

research. Experimental and preliminary results are given in section 6 to show the feasibility of the framework. Finally, a brief conclusion and future work are presented in section 7.

2. Previous Work

Since the first networked device "Cambridge coffeepot" appeared on the Internet, the rapid growth of the WWW over the past several years had resulted in a growing number of telerobotics sites and web accessible devices on the Internet.

The 1st generation of Internet robotic systems came into existence in 1994. The Mercury project was first carried out at that time to build a 6 DOF tele-manipulator, allowing users to pickup and manipulate various objects within its reach [11]. The telerobotic garden extended this idea to tending a garden situated around an Adept 6 DOF arm to allow users to dig and water the plants [12]. Various other devices have become available over time, such as the Bradford robotic telescope [9], the NETROLAB at Reading [21], a Web-based tele-manipulator [22], an interactive 3D art viewing system [14], the VISIT telerobot system via computer network [19], Internet-based remote teleoperation [8], and the "MAX" wireless teleoperation [10]. This generation of Internet robots is mainly based on robotic arms or simple mobile robots that are directly controlled by human operators. In other words, a human is in the control loop. These telerobots operate within a well-structured environment with little uncertainty and have no local intelligence.

In contrast, research on the 2nd generation of Internet robots has recently begun to focus on autonomous mobile robots that navigate in a dynamic and uncertain environment, including the Xavier -- an office exploring robot at CMU [23], the Khep-on-the-Web project for open access to a mobile robot on the Internet [25], and the Museum tour-guide robot [26]. The key features of this generation of Internet robotic projects are their autonomy and reactive behaviours which enable them to navigate and cope with uncertainty in the real world. Supervisory control is the main focus in building this generation of the networked robots. However, how to handle multi-robot co-operation and remote robot programming remain the major challenges.

Internet robotics involves controlling robots or devices from a web browser remotely and differs from traditional teleoperation in several aspects:

- ❑ The delay and the throughput of the Internet are highly unpredictable, unlike traditional teleoperation where the interfaces have fixed and guaranteed delays.
- ❑ Web-based teleoperation requires a high degree of tolerance to possible data-package loss due to packet discard when there is no existing remedy.
- ❑ Internet robots need innovative mechanisms for coping with shared control among multiple web users with different applications in mind.
- ❑ Internet robots are remotely operated by many people with little expertise and few skills. In contrast, traditional tele-robots were handled by trained operators.
- ❑ Since web users are a central part of the control loop in Internet robots, their behaviour becomes an important consideration in the system design.

All of these aspects contribute to the fact that there are not yet any commercial services provided with such technology. This in turn poses many new challenges since Internet robots need to deal with a complex and dynamic environment and the unpredictable delay in network communications.

3. A Framework for Multi-robot Cooperation over the Internet

We are interested in building a networked Telerobotics system consisting of multiple mobile robots so that Internet users, especially researchers and students, can control these mobile robots to explore dynamic environments remotely from their home and share this unique robotic system with us. The long-term goal of our research is towards real-world applications such as tele-manufacturing, tele-training, and tele-service. The work in this paper is focused on the realization of some of the features, namely a uniform interface for easy integration of different robots into the system; a networked and intuitive user interface and adequate feedback; a low-cost and easily extendable system for the addition of more complex functionality; cooperative behaviours to implement complex tasks that cannot be implemented by a single robot; and high degree of local intelligence to deal with the problems caused by the low bandwidth and transmission delay of the Internet. Figure 1 shows the overall framework proposed in this work, which can be separated into three main parts as follows.

Robotic Agents

A primary aim in the development of teams of cooperative Internet robots is to synthesis their cooperative behaviours in order to accomplish missions that cannot easily be achieved with individual robots. At the same time, low-level behaviours developed in individual robots should be retained in a multi-robot setting in order to protect a robot from colliding with other robots or its environment [15]. With the low-level behaviours, multiple Internet robots are able to work closely in a mission if necessary.

Based on agent technology, the control system of the Internet robots consists of a number of agents each of which plays a specific role in the system. Some agents are designed to implement high-level behaviours for cooperation such as communication and formation behaviours. Others are used to achieve the low-level behaviours, e.g. avoid obstacle, move to target, grab objects, etc, based on behaviour-based approach [3]. To handle the failure, some of agents are designed to be able of emigrate from one place to another to implement recovery jobs. More details are in the next section.

Communication

Communication forms an important part in coordinating all parts of the framework (i.e. robotic agents, communicators, etc.) in order to maximize performance and efficiency. In consideration of possible future extensions and different applications, the communication framework has been design with simple extendibility in mind. As the system expands over time, it is very likely that we would be require to cope with different communication needs. For this, a simple messaging system is adopted.

An improvement was made over the TCP-based single robot control as presented in our previous paper [30]. Communicator agents were added in order to facilitate communication between all parts of the system. In other words, each component in the system is assigned a communicator agent that provides the necessary communication service for interacting with other parts of the framework. The Communicator agents are also responsible for interpreting the user's commands and translating them into a form understandable by the other agents. Unknown commands will be discarded and reported so that no unnecessary communication is transmitted in the framework's communication channel. The addition of Communicator agents also simplifies the extension of our existing system to incorporate more agents into the framework by providing a unified messaging system.

User Interface

The user interface is designed to be simple and intuitive since most of users of Internet or online robots are non-specialists. Providing a lot of extra functionality will only increase the user's stress level.

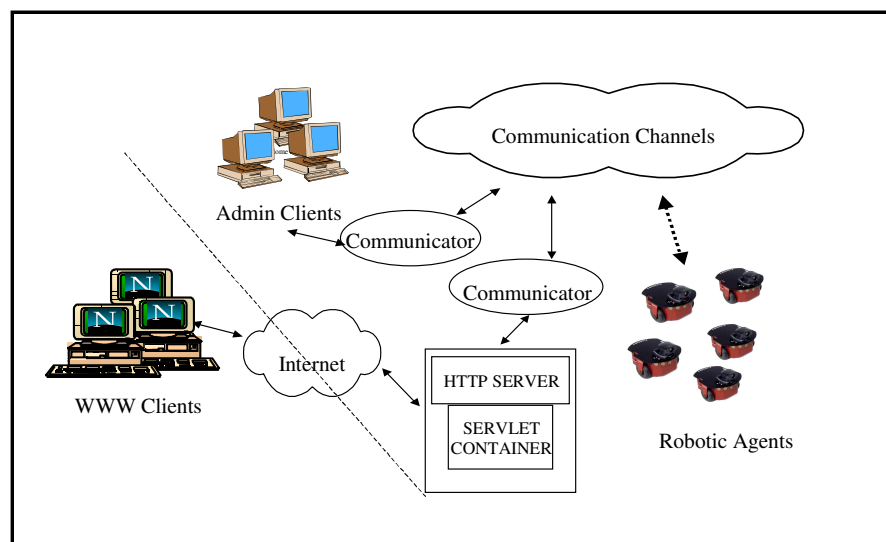


Figure 1 Communication Framework

4. Implementation of Agent Architecture

The proposed framework can be implemented as a multiple Internet telerobot system as shown in Figure 2, which contains a human user computer, a central hub computer, and a number of robots. In such a system a human user using the human user computer has control of one or more remote robots via the central hub computer. Control commands are sent from the human user computer over an Internet WAN connection to the central hub computer and then passed on to robots. Central hub computer-robot communication and inter-robot communication occurs via Internet LAN connections. Several potential points of failure exist in a multiple Internet telerobot system; these include human user, central hub, or robot computer failure or Internet WAN or LAN communication failure between computers.

The human user computer (Figure 3) hosts a web page interface that can be operated in a web-browser on a computer with an Internet connection. The web page interface provides relevant information and control features to allow users to control multiple robots, such as the ability to directly control each robot, the function to allocate and prioritise autonomous behaviour in each robot, information both on the activity each robot is engaged in and on the functional status of each robot, as well as visual or sensory information describing robots in their environment

More specifically, in the human user computer or WWW Clients:

- visual or sensory information is passed from the central hub computer to the interface for display to the human.
- robot functionality information is passed from the central hub computer to the interface to update functionality of each robot and determine which behaviours robots can perform.
- direct control commands are passed from the direct control section of the interface to the central hub computer.
- supervisory control commands are passed from the supervisory section of the interface to the central hub computer.

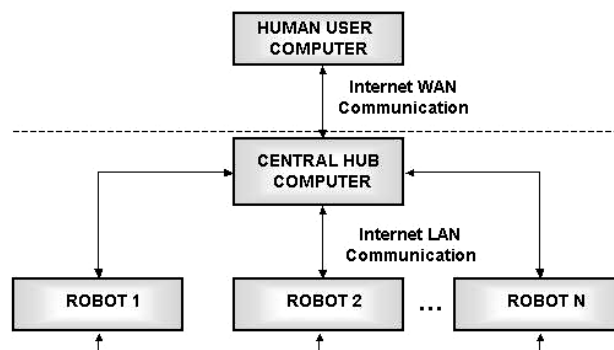


Figure 2 Block diagram of a multi-robot system over the Internet

The central hub computer (Figure 4) acts as a link between user computers and robots. The central hub computer:

- passes direct control commands received from the user to appropriate robot/s via a direct control agent (information passed are simple string commands e.g. forward, left, right, stop), used for low-level direct control of robots.
- creates behaviour agents for supervisory control which migrate with behaviour code to robots in order to perform high level behaviours (the supervisory control buffer receives simple string commands e.g. <"Robot 1-Wall Following-Priority 2"> it checks the required behaviour against current robot functionality and if functionality is available generates a mobile agent from a library of predefined mobile agents with the appropriate behaviour requested. It attaches the destination address and priority of the behaviour to the mobile agent. The mobile agent is generated into a place on the central hub computer from where it autonomously migrates to the correct robot).
- passes data received from robots via a data agent to the human user (information passed is a simple string, containing information e.g. robot co-ordinates, functionality etc).

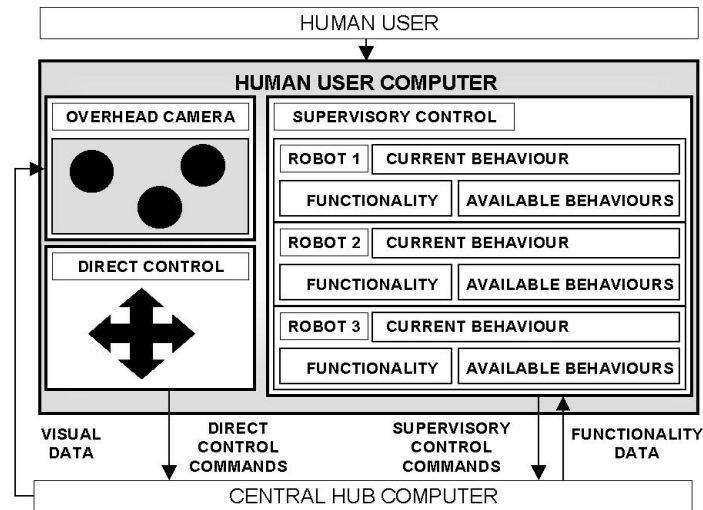


Figure 3 A web interface for users

In addition to the data passing stated above, the central hub computer operates a communication gateway. The communication gateway monitors communication connection and possible delay between users computer and robots. In the event that the central hub computer fails, control could move to one of the robot's while the central hub is unavailable subsequent moves could be made between robots, if the host robot fails. This would allow robot groups to operate autonomously in the event of central hub computer failure. In addition to fault tolerant mechanisms provided by mobile agents for communication or computer failure, mobile agents could also be used to increase fault tolerance at a lower level within robots.

The agent architecture for individual robots is shown in Figure 5. Each robot contains an agent place that contains: a single data agent, a single direct control agent, and one or more behaviour agents. The data agent performs the following functions on a robot:

- contains a list of the robots current functionality.
- performs arbitration between other agents at the place for control of the robot.
- passes data information to the central hub computer.

In addition to the functionality stated above, each robot operates a communication gateway. The communication gateway monitors communication connection and possible delay between the central hub computer and other robots.

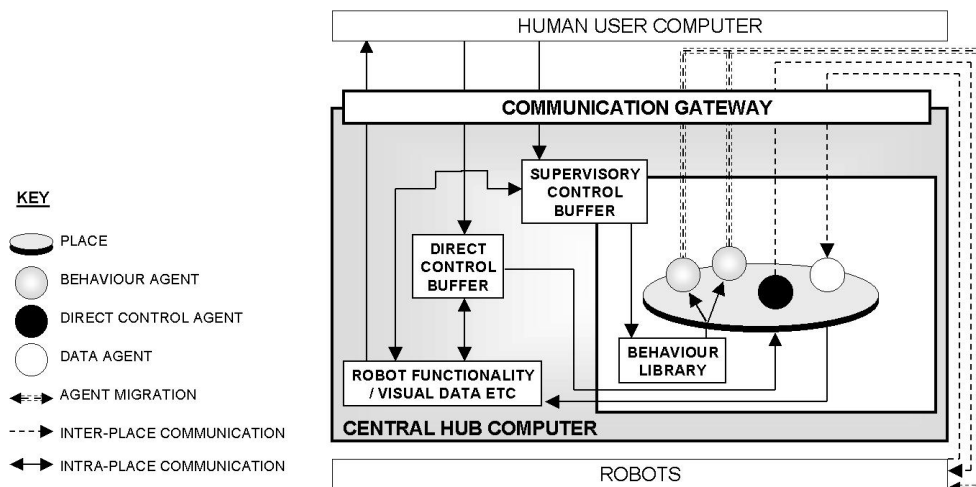


Figure 4 Agents on the central hub computer

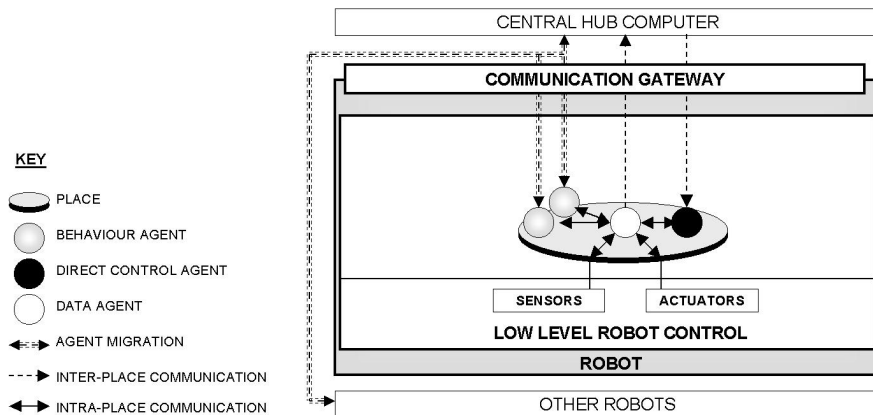


Figure 5 Agents on the individual robots over the Internet

5. Simulation

In implementing the framework, we have chosen to use readily available and open source resources at the current stage. This reduces the time to complete and lower the cost incurred in the system development. A real-world implementation will be carried out in the next stage. The resources used are a Pentium II 500 PC with 128MB RAM, Windows 98/2000, Redhat Linux 7.1, Apache HTTP Server v1.3.22 [1], Tomcat v3.2 Servlet Engine [17], Teambots [5], and Java2 SDK v1.3.1 [27]

5.1 Simulation Environment

For this experiment, a simulation environment is used so that preliminary results can be gathered quickly. Hence, Teambots Simulation Environment [5], Teambots in short, is used for implementing the experiments. Teambots is written entirely in Java, except for some C code which is used to interact with real robot hardware (including Nomad 150, Cyb). It contains a collection of Java packages that simplify the prototyping, simulation and execution of multi-robot control systems. Teambots was designed in a way that would allow the control program written for the simulator (i.e. TBSim) and be able to execute on a real robot (using TBHard). Figure 6 shows a snapshot of Teambots simulation environment running.

Teambots utilises the Motor Schema approach for reactive robot control [2]. Motor schemas in Teambots are defined as *Nodes*. A node has only two functions:

- Constructor for initialisation
- Value () it is called repeatedly during runtime.

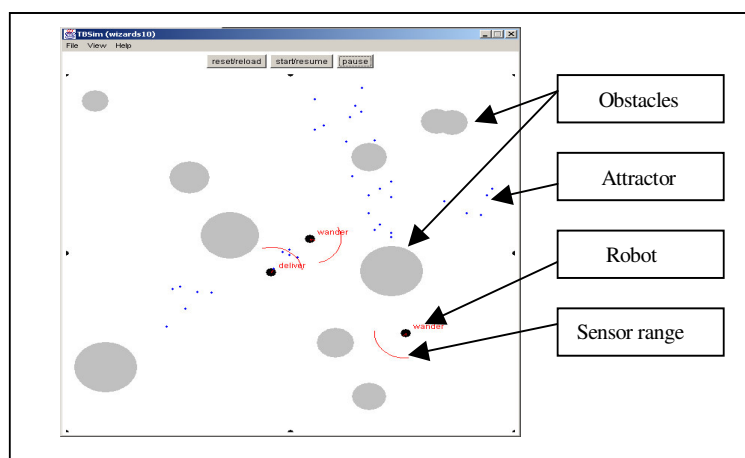


Figure 6 Teambots Simulation Environment

Each node corresponds to a robot schema. Some schemas can be grouped together to form assemblages and sequences using finite state machines (FSA). Teambots comes with a rich set of nodes packaged together for the Clay toolkit ([6]). Because the source code is freely available, we can customise and design new nodes for our own use.

5.2 HTTP Server & Communication Channel

The HTTP server acts as a gateway for the users to access the remote robots through the Web. Java servlets are widely used while the HTTP server serves up static html pages (e.g. Project home page, project information, contact, etc.). Two servlets were written:

- *CommandServlet* this servlet is a wrapper for the Communicator agent that handles command scripts transmitted through the User interface (presented in next section) and forwards the command to the appropriate receiver.
- *FeedbackServlet* this servlet is also a wrapper of the Communicator agent, but functions as a feedback/perceptual data collector for the user. The client side on the applet is still under development. At the moment, it only returns the last user command transmitted to the local communication channel.

In this work, Teambots RoboComm communication server was used as the communication channel for the robotic agents and the servlets reside in the HTTP server. It is a lightweight implementation of a messaging server, which simplifies robot's communication.

5.3 User Interface

There are two modes of user interfaces used in the system:

- **User** In this interface, the user is provided with the interface through a Java-enabled WWW browser. The design is simple and is aimed at non-specialist users.
- **Administrative** Still in command-line (no graphical interface), this is used for administrative task.

To enable the interaction between the user and the agents, a simple user mode control console is implemented using Java Swing, as shown in figure 7.

The button's functions are

- *CONNECT/DISCONNECT* send request/release teleoperation signal
- *LEFT/RIGHT* send directional command (left/right 20 degrees)
- *START/STOP* start/stop robotic agent from moving
- *Robot* (the menu) used to choose which agent(s) would receive the command.

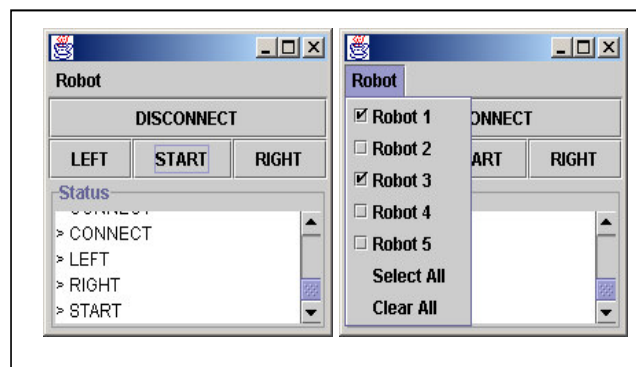


Figure 7 A simple user control console

5.4 Robot Agents

Finite State Automata (FSA) is used to design and implement task-level behaviours (e.g. foraging) of the robotic agents. FSA is used because it is simple to extend, easy to trace the execution sequence, and easy to implement. A robotic agent consists of five components (Figure 8):

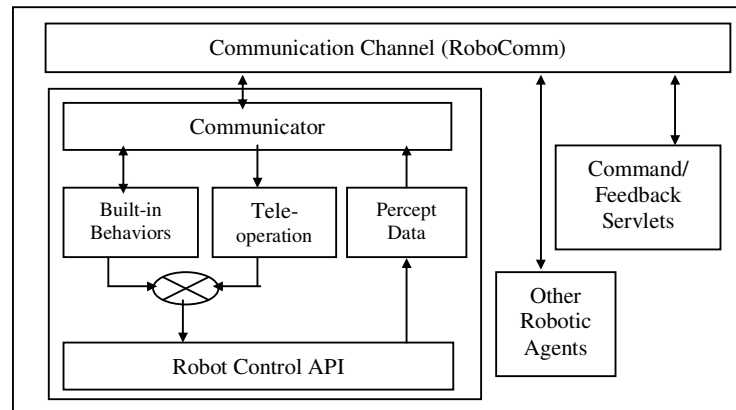


Figure 8 Robotic agents software structure

- 1) **Built-in Behaviours** this component contains the built-in behaviours for the robotic agents to act autonomously (e.g. foraging). In the future, we plan to extend this component so users can send scripts from remote computers to configure the FSA in the robotic agent. (e.g. request loading or removal of behaviours, change behaviours priority/weight to change the overall behaviours of the FSA, etc.)
- 2) **Teleoperation** (in this experiment: direction control) is designed as another motor schema, which would contribute to the overall behaviour of the robotic agent. This approach, based on TELOP ([4]), ensures seamless integration and avoids undesirable interruption to the robotic agent; its inner workings (this problem is similar to the locking problems as seen in concurrent programming).
- 3) **Communicator Agent** enables the robot to communicate through the communication channel.
- 4) **Perceptual Data** this component extracts perceptual data through the Robot Control API that in turn is used by other components for processing (e.g. sensory input to behaviours, sharing of perceptual data, etc.)
- 5) **Robot Control API** API used to communicate with the underlying hardware.

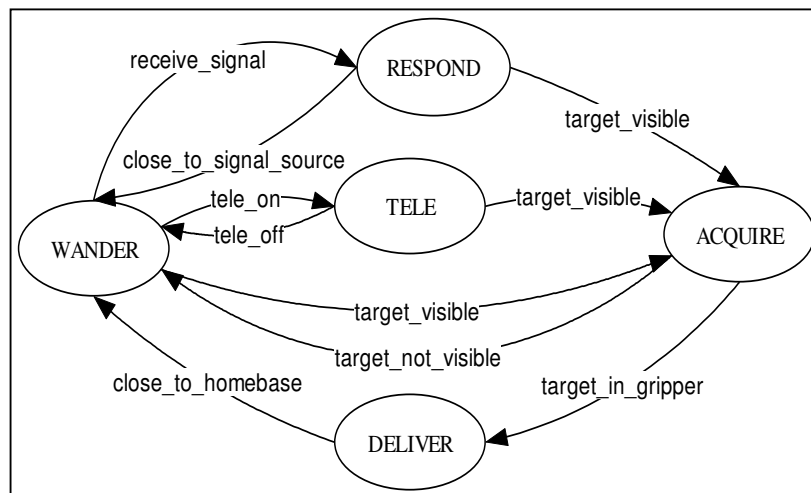


Fig 9 Finite State Automata (FSA) for each agent

6. Experiment & Preliminary Results

6.1 Some Assumptions

Some assumptions were made in this experiment:

- Robots do not have global positioning information in the simulation. Instead, robots rely on the information gathered from the simulated sonar in order to avoid obstacles. Although the map is available, it is not accessible to the simulated robots.
- Robots are assumed to have a relatively accurate positioning capability so that each robot is able to tell others its current location relative to the home base.
- Robots have a limited and circular communication range. In our simulation experiment, the communication range can be set as a parameter.
- Robots can accurately identify whether an object is an attractor (e.g. interesting/intended objects), another robot or an obstacle. Based on this assumption, robots can take actions corresponding to the information gathered.

6.2 Foraging

Foraging is a well-known and well-studied problem in robotic navigation [3]. Foraging emulates the real-world situation that multiple robots cooperate together to search for and analyse an unknown environment.

This task consists of a group of robots moving away from a home base and looking for predefined targets or attractors. The objective of the robots is to find an attractor object in the environment, collect it, and finally transport the attractor back to the home base. These three steps are repeated until there is no more attractor in the environment. These steps can be represented by three simplified FSA states as shown as part of figure 9.

- **Wandering** wandering randomly around the environment to search for attractors.
- **Acquire** approach to the detected attractor.
- **Deliver** transport the attractor to home base

These three states form the basis of the built-in behaviours detailed in the following sections.

6.3 Built-in Behaviours

The built-in behaviour's design at this stage forms the foundation of the robot controller where subsequent features/extension is put on top. The design of the behaviours will follow the standard three states as shown in the lower part of the FSA in figure 8: *WANDER*, *ACQUIRE*, and *DELIVER*. Since Teambots comes with an example program implementing this design, it is extended and used in the experiment instead of writing another one from scratch. Some addition and modifications were made to the example program so that it meets our requirements.

- **Logging** Record the number of attractors collected by the foraging agents every time an agent drops an attractor at home base. This is solely for statistical purposes.
- **Timeout** An earlier attempt in the experiment shows that sometimes several agents may attempt to collect the same or close by attractors, which causes a stagnation condition ([3]). Adding timeout can force the agents to stop moving towards the problem attractor for them to have more time to avoid teammates/obstacles to break this condition. Fig. 8 shows an example of a stagnation condition captured from one of the experiment trials.
- **Adding Communicator Agent** additional features to handle communication through Communicator (see section 6.4 for details).
- **Adding Teleoperation Control** Adding teleoperation control to the foraging agents. (see section 6.5)

6.4 Communication Strategy

In this experiment, we have chosen to use Goal-Communication ([7]), where the location of an attractor is communicated to other foraging agents. The reason behind this strategy is to exploit the possibility that more attractors may be nearby. This feature is added to the foraging agent's FSA by adding a state *RESPOND* that is triggered by a signal received from other agents when the agent is in *WANDER* state. For this to work, a Communicator agent is integrated into the agent. When the foraging agent detects an attractor and it is in *DELIVER* state, it will broadcast a detected signal, attaching its current position, to the other agents so that they can help search for possible attractors nearby. Other foraging agents not currently working (i.e. in *WANDER* state), and within the communication range will respond to the signal by moving towards the position attached with the detected signal.

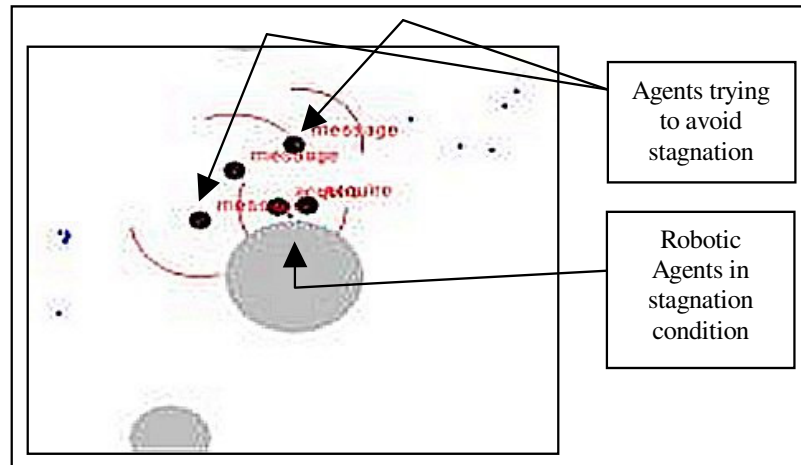


Figure 10 Stagnation Condition

6.5 Teleoperation Control

Following Arkin's TELOP [2, 3, 4] concepts, a new motor schema *TeleControl* node is implemented and introduced into the foraging agents as a new FSA state. The main function of this schema is to accept a directional command from the operator and generate a vector pointing at the direction indicated by the command. Another new NodeBoolean *tele_on* is used to listen to operator's request/release tele-control signal and triggers state changes between *WANDER* and *TELE* accordingly.

6.6 Preliminary Results

This experiment is conducted over three different settings. Each setting is presented with the same map and with randomly placed attractors (attractors' number: 30). The results gathered are 20 trials in each setting. The average simulation performance is in figure 11. Following are the observations for each setting.

A) 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) Built-in Behaviours + Teleoperation

In figure 11, 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).

A more interesting observation is in figure 12. As the number of foraging agents increases, the performance becomes unstable. 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.

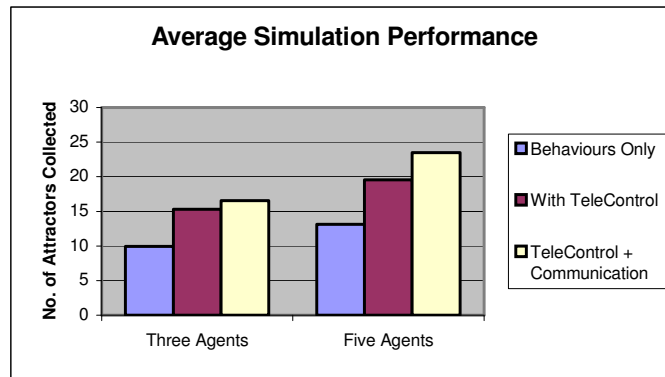


Figure 11 Average Performance of three different modes

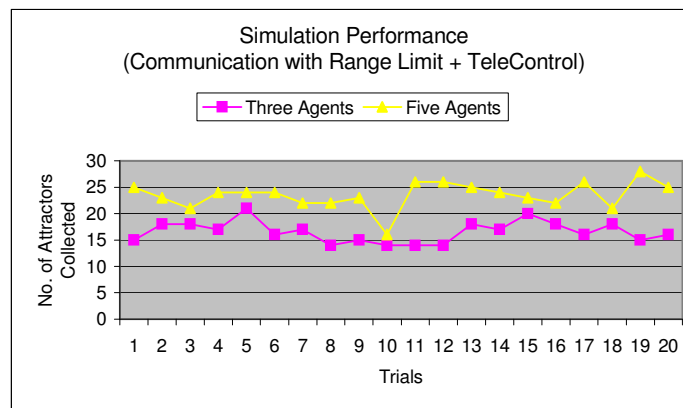


Figure 12 Simulation performance for Built-in Behaviours + Teleoperation

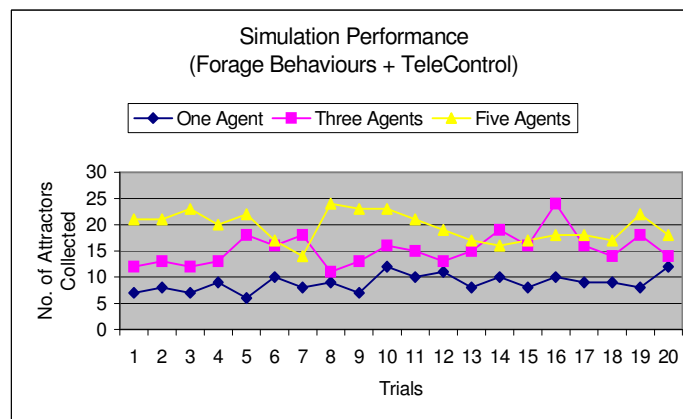


Figure 13 Simulated performance for Built-in Behaviours + Teleoperation + Communication Strategy

C) Built-in Behaviours + Teleoperation + Communication Strategy

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 13). 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 nearby to follow. This reduces the operator's control complexity.

One problem encountered in this approach is, sometime several foraging agents respond to the same help signal and will try to approach the same target, causing a stagnation condition (figure 10). One way that we employed to tackle this problem early on, is to change the weights in a state *RESPOND*. By setting the weight for 'avoid collision' higher than that for movement towards the target position, the foraging agents will be free more quickly from a stagnation condition. Another method commonly used in literature is *impatience* which basically uses a timeout in certain situations, such as *ACQUIRE*. When the foraging agent fails to collect the attractor after the predefined time, it will time out and go back to the *WANDER* state.

7. Conclusions and Future Work

This paper presents a new framework for multi-robot cooperation over the Internet. The framework consists of a number of agents that implement different tasks at the same time and some of them are designed to move from one robot to another for faulty detection and recovery. The experiment results show that employing multiple agents does improve the system performance but it also increases control complexity for the operator. Simple sensor information sharing through communication can help to increase the efficiency of the forage agents, thus simplifying control complexity. However, such a communication strategy causes a stagnation condition, which affects the performance of nearby agents. Currently this is addressed by using timeouts. A recent finding in using Teambots appears to show that the random seed used in generating the noise vector could affect the performance of the agents. It warrants further investigation since it might affect the experimental results presented here.

The experiment results presented in this paper seem to suggest that agents forming subgroups are able to work more efficiently than acting individually. The next step of the research may be focused on group behaviours with possibly more challenging experiment platforms, including distributed mapping, herding, etc. On the implementation side, the user interface currently can only send commands to the agents. Some feedback mechanism should be introduced to form a complete closed-loop system. The control buttons could be changed into an image map for simple control.

The proposed framework will be implemented on three Pioneer mobile robots in our Brooker Laboratory that have wireless network connections. More detailed implementation of mobile agents is currently under investigation.

References

- [1] Apache HTTP Server, <http://httpd.apache.org/>, Apache Software Foundation.
- [2] R.C. Arkin, Reactive control as a substrate for telerobotic systems, Proc. IEEE Aerospace and Electronics Systems Magazine, pages 24-31, 1991
- [3] R.C. Arkin, Behaviour-based robotics, MIT Press, 1998
- [4] R.C. Arkin and K.S. Ali, Integration of reactive and Telerobotic control in multi-agent robotic systems, Proc. The 3rd Int. Conf. On Simulation of Adaptive Behavior, Brighton, UK, 1994
- [5] T. Balch, Teambots: <http://www.teambots.org/>, 2000
- [6] T. Balch, Clay: Integrating motor schemas and reinforcement learning, College of Computing, Georgia Institute of Technology, 1997
- [7] T. Balch and R.C. Arkin, Communication in reactive Multiagent robotic systems, 1994
- [8] K. Brady and T.J. Tarn, Internet-based Remote Teleoperation, Proc. IEEE International Conference on Robotics and Automation, pages 65-70, Leuven, Belgium, 1998
- [9] M. J. Cox and J.E.F. Baruch, Robotic Telescopes: An Interactive Exhibit on the World-Wide Web, Proc. 2nd International Conference of the World-Wide Web, Chicago, October 17-20th 1994
- [10] A. Ferworn, R. Roque and I. Vecchia, MAX: Wireless teleoperation via the World Wide Web, Proc. IASTED Conf. on Robotics & Applications, pages 158-162, Santa Barbara, 28-30 Oct. 1999

- [11] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, Desktop tele-operation via the World Wide Web, Proc. IEEE Int. Conference on Robotics and Automation, pages 654-659, 1995
- [12] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, J. Wiegley, and E. Berger, The Telegarden, Proceedings of ACM SIGGRAPH, pages 135-1140, 1995
- [13] K. Goldberg and R. Siegwart, Beyond Webcams -- An Introduction to Online Robots, ISBN 0262072254, 2000
- [14] S. Goldberg, et al., DIGIMUSE: An interactive telerobotic system for remote viewing of 3D art objects, IROS'98: Workshop on Web Robots, pages 55-60, Victoria, Canada, 12-17 October 1998
- [15] H. Hu and M. Brady, π° Dynamic Global Path Planning with Uncertainty for Mobile Robots in Manufacturing, IEEE Transactions on Robotics and Automation, Vol. 13, No. 5, pages 760-767, 1997
- [16] H. Hu, D. Gu and M. Brady, A modular computing architecture for autonomous robots, International Journal of Microprocessors and Microsystems, Vol. 21, No. 6, pages 349-362, March 1998
- [17] IEEE Society of Robotics and Automation Technical Committee on: Internet and Online Robots, <http://www.ieor.berkeley.edu/~goldberg/tc/>, 2001
- [18] JakartaTomcat, <http://jakarta.apache.org/tomcat/index.html>
- [19] K. Kosuge, J. Kikuchi and K. Takeo, VISIT: A Teleoperation system via computer Network, Proc. IROS'98: Workshop on Web Robots, pages 61-66, Victoria, Canada, 12-17 October 1998
- [20] S. Levington, "HIGHVIEW: High-quality Resilient Video Over Existing Networks", Department of Computer Science, University of Essex, Colchester, England, <http://cswww.essex.ac.uk/research/>
- [21] G.T. McKee, and Barson, R. NETROLAB: providing access to robotics technology using the Internet. Robotics and Machine Perception. Special issue: Networked Robotics. USA: SPIE, 5, 1, 1996
- [22] E. Paulo and J. Canny, Delivering real reality to the world wide web via telerobotics, Proceedings of IEEE International Conference on Robotics and Automation, pages 1250-1256, 1996
- [23] R. Simmons, XAVIER: An autonomous mobile robots on the Web, Proceedings of IROS'98 Workshop on Web Robots, pages 43-48, Victoria, Canada, 12-17 October 1998
- [24] R. Siegwart, C. Wannaz, P. Garcia and R. Blank, Guiding Mobile Robots through the Web, Proceedings of IROS'98 Workshop on Web Robots, pages 1-6, Victoria, Canada, 12-17 October 1998
- [25] P. Saucy and F. Mondana, KhepOnTheWeb: Open access to a mobile robot on the Internet, IEEE Robotics and Automation Magazine, pages 41-47, March 2000
- [26] D. Schulz, W. Burgard, D. Fox, S. Thrun, and A.B. Cremers, Web interface for mobile robots in public places, IEEE Robotics and Automation Magazine, pages 48-56, March 2000
- [27] Sun Microsystems, I. Java 2 SDK v1.3.1, Sun Microsystems Inc., 2001
- [28] K. Taylor, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, et al., MINER VA: A second-generation museum tourguide robot, Proc. Int. Conf. on Robotics and Automation, pages 1997-2085, 1999
- [29] S. You, T. Wang, Q. Zhang, Internet-based Multimedia Interaction in Teleoperated Robotic System, Robotics Institute, Beijing University of Aeronautics and Astronautics, 2000
- [30] L. Yu, P.W. Tsui, Q. Zhou, H. Hu, A Web-based Telerobotic system for research and education at Essex, Proc. IEEE/ASME Int. Conf. On Advanced Intelligent Mechatronics (AIM'01), Colorado, 2001