Robotica (2000) volume 18, pp. 235–250. Printed in the United Kingdom © 2000 Cambridge University Press

Mobile robot society for distributed operations in closed aquatic environment

M. Vainio, P. Appelqvist and A. Halme

Automation Technology Laboratory, Helsinki University of Technology, P.O. Box 5400, FIN-02015 HUT (Finland)
www.automation.hut.fi
E-mail: mika.vainio@hut.fi

SUMMARY
In this paper a multirobot system consisting of small size ball-shaped mobile underwater robots is introduced. Robots form a cooperative society operating together for a common goal. This is made possible by inter-member communication and control architecture allowing cooperation. The test environment is a closed aquatic process containing tanks, pipes, and a jet pump. The task considered is cleaning of biologically contaminated spots in the process. Detailed hardware structure of a robot-member as well as the control architecture are introduced. Behaviour of the cooperative system is demonstrated in a test environment where contamination caused by biological algae growth is emulated by infrared panels behaving like a living biomass.

KEYWORDS: Underwater robots; Robot society; Cooperation.

1. INTRODUCTION

The research results in the following are inspired by the societal structures developed in the biological world by evolution. In those systems, as is usually the case in Nature, the goal is never to find optimal or very accurate solution, but rather to come up with feasible solutions. Efficiency in energy consumption and utilization of materials along with amazing fault tolerance are obtained through distributed cooperative functions. Cooperative systems are formed by micro-organisms, cells, and animals like ants. In order to get successful cooperation, these systems need some kind of basic rules to minimize or at least reduce the unnecessary interference and competition between system members.[1] If these rules could be found and formulated, then, a real life-like society of robots with the same kind of astonishing robustness and adaptivity could be constructed. Such system is called robot society, see Halme et al.,[2] where a detailed presentation of the concept can be found. Robots in the society might be also called (physical) autonomous agents according to the meaning given to this term usually in the literature.[3,4] The term "robot society" is quite rarely used in the literature. Instead, terms like "robot colony", "distributed autonomous robotic systems", "cellular robotics", "collective robotics" are frequently used to describe systems, where multiple agents are working more or less together towards a common goal.[5,6] The effect of social relationships to the performance of the whole system can be significant.[7,8]

All functions of a robot society are obviously realized through its members. Members' behaviors are the results of their own needs as well as the stimulus received from the environment and other members. Constraints set by the society or directly by the operator define the frame of operation. In the case presented here, a member utilizes the information accumulated in the whole society. This is realized through the inter-member communication, which allows a member to update its environment model by utilizing observations done by other members.

Some valuable benefits are automatically built-in to the concept of robot society. If a large number of homogenous member are used at the same time for the same purposes, the level of redundancy becomes very high. The benefits and drawbacks of homogenous multi-robot systems compared to heterogeneous systems are under extensive study.[9,10] In homogeneous system, the volume of the society, *i.e.* the capacity of the system, can be adjusted simply by increasing or decreasing the number of robots in the system. No reconfiguration is needed, due to the implemented control and communication structures, which in fact does not have to recognize the number of members.

2. SUBMAR ROBOT SOCIETY

In the process industry the question of monitoring the internal state of the process in real-time and making local adjustments in mixing, flow or reaction conditions is one of the major problems. Normally the sensors used in monitoring are fixed and provide information only from certain areas of the process. Local adjustments are often difficult to implement, if not totally impossible, and thus the control policy is based more or less on the overall control of the system. This may lead e.g. to an extensive use of chemicals, which is both expensive and causes often unwanted residuals and pollution. To make sensors and actuators mobile inside a process has been one of the motivations for designing the SUBMAR robot society, illustrated in Figure 1.

2.1. Society member

The society consists of ball-shaped small-sized (diameter 11 cm) robots called SUBMARs (**S**mart **U**nderwater **B**all for **M**easurement and **A**ctuation **R**outines). These robots move passively along liquid flow or actively by controlling their specific weights. SUBMAR, shown in Figures 2a-2c, is

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

equipped with a micro-controller, several sensors, tank actuators and a short range radio. The figures represent the latest generation of SUBMARs.[11,12] The sensors implemented depend on the application at hand. Typically, sensors for temperature, pressure, pH, and conductivity are useful in many applications. Inertia sensors could be used for measuring accelerations and for navigation purposes. The tanks used for controlling the specific weight, taking samples, and carrying cleaning agent. Energy is carried in a battery back. Due to the fact that motion energy is taken mainly from process flow, consumption of energy is relatively small and the operation time long, in practice several hours. Energy and cleaning agent could be refuelled in a recharging station, which is, however, not realized in the test system.

Because of limited maneuverability SUBMARs move relatively stochastically. This affects their task performing capabilities, too. "Group power" included in the controlled behaviour of the multirobot society, however, compensates this weakness effectively. Tasks like collecting, harvesting, cleaning or searching and destroying are typical ones where group power can be utilised.[13–15] Nature has created many types of group behaviours with different social features for this purpose. Typical are social insect communities formed, for example, by ants or bees.[16,17] Even simple organisms, like bacteria, are known to form social communities with simple communication mechanisms.[18] The society formed

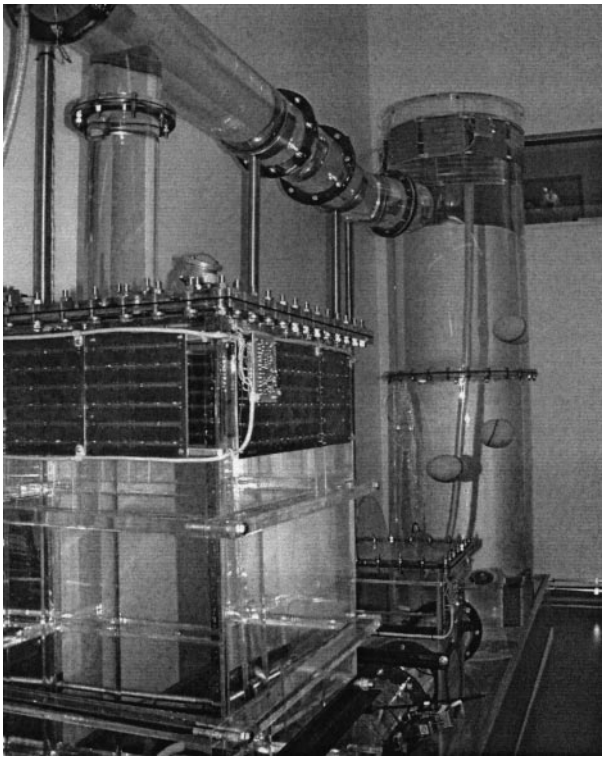

Fig. 1. SUBMAR robot society in test environment.

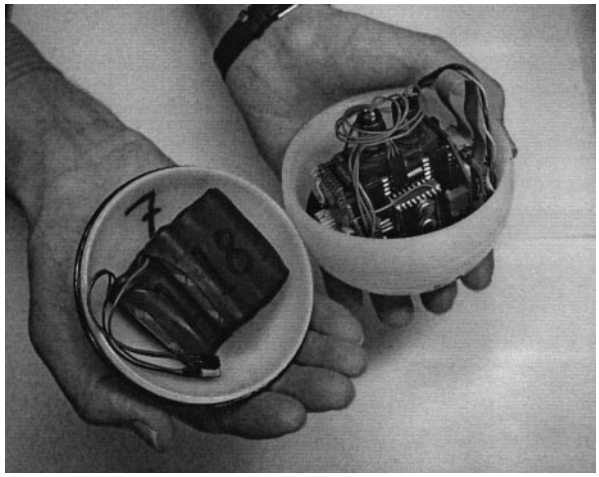

Fig. 2a. SUBMAR cover opened.

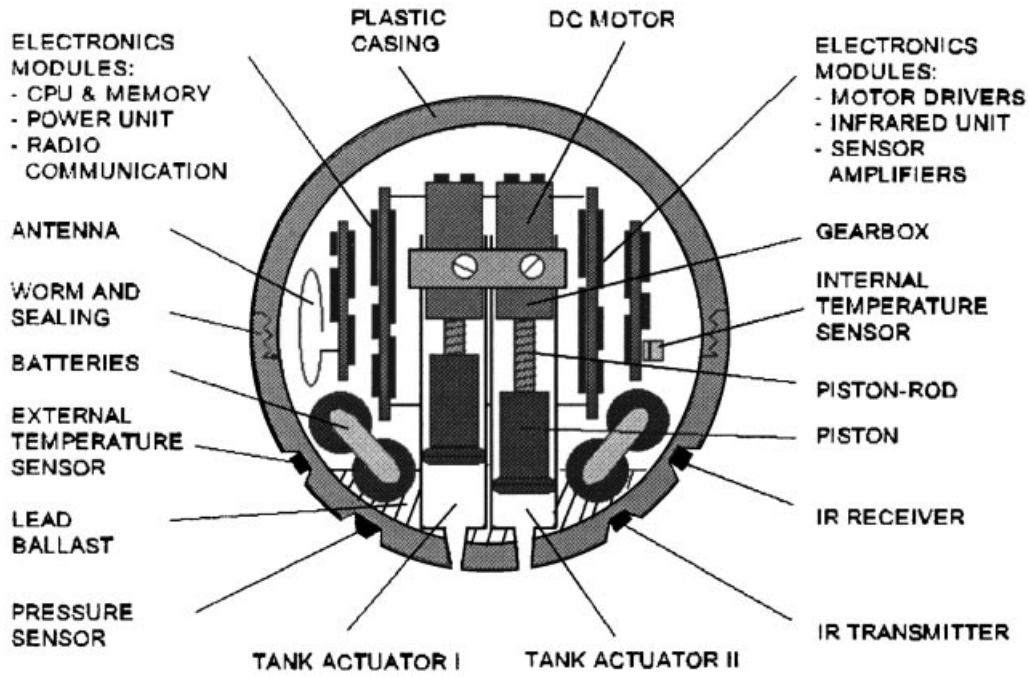

Fig. 2b. Cross-section view.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

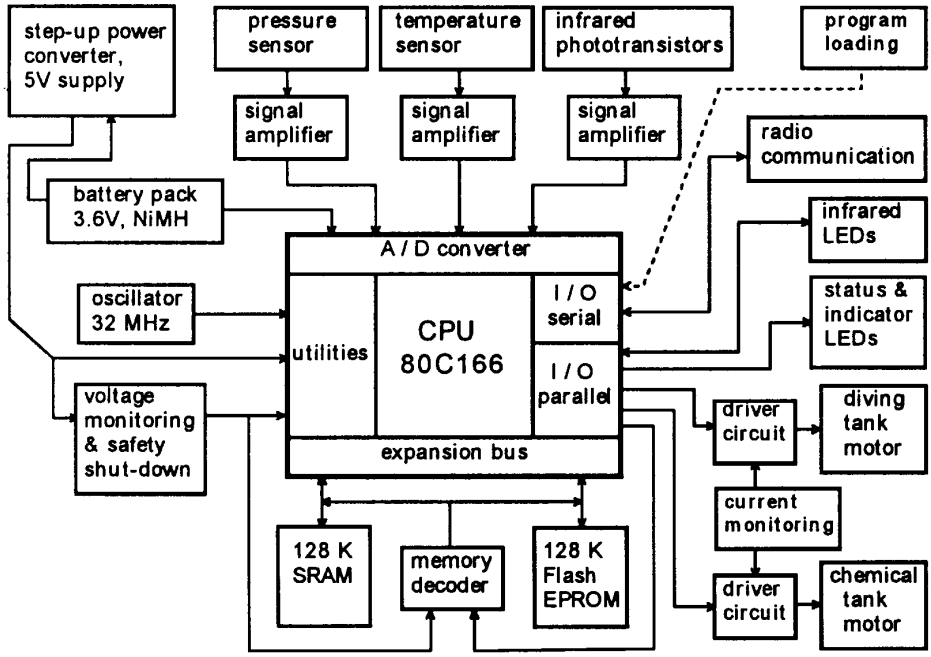

Fig. 2c. Electronics block diagram.

by SUBMAR robots is inspired by those biological communities, although communication by radio makes it possible to create more complex messaging system than in biological communities. The idea is, however, to follow minimalistic approach where communication as well as other features are kept at the minimal level.

2.2. Communication

Communication in a robot society can be divided into two categories; *communication between the operator and robots*, and *communication among the robots*. The former enables the user to control the society, while the latter is needed to turn a group of robots into a cooperative society. Since all data exchange in the society is carried out under the same frequency, some sort of communication protocol is essential. As a solution, an Ethernet (CSMA/CD) type of protocol was developed and implemented to SUBMARs. The protocol frame is presented in Figure 3.

Item	Bytes	Explanation
Message title :		
STX	1	Start byte
Version	1	Version number
Receiver ID	1	Receiver / Broadcast
Sender ID	1	Sender
Message type	1	CMD, INFO, ACK,...
Message ID	1	Used for ACKs
Data length	2	Length of payload
Title CRC	2	Title checksum
Payload CRC	2	Payload checksum
Payload :		
Data max.	65520	Actual message

Fig. 3. Message frame for communication.

Some inbuilt features in the protocol support an effective communication: messages can be sent only to certain members, or broadcast to all in the range. In addition, the most important messages can be sent with acknowledge request, to ensure that the message was received correctly. CRC checksums are calculated separately for the message title and for actual message, payload. To minimize the amount of transferred data, only a collection of fixed messages are used. The protocol does not determine the format of messages.

There are some parameters which can be adjusted to tune communication to the existing environment conditions. *Automatic resends* are used in context with info type of messages; the probability of correctly received messages increases radically, if each message is always resent a couple of times. Also, the number of *Acknowledge retries* can be set, as well as *Acknowledge time-outs*, the time before next retry. *Carrier signal detection* is used to prevent overlapping transmissions; in a very noisy environment it may be useful to switch it off.

Despite efficient communication protocol, it is evident that some part of the information is always lost. However, if the communication structure is well designed it does not harm the functioning of the society. In the worst case some actions are just delayed.

2.3. User interface

An operator's communication station is an important part of the system facilities. *Base station* software provides user interface, which allows the operator to control the society and obtain on-line information from the robots. Figure 4 shows a screenshot of the base station running in NT workstation. This software features protocol settings, different types of sendings, and logging of all the received data to files. Each robot has also its own log file for communication; this allows detailed performance analysis of

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

the communication network in the society during task execution. Physically, a base station communicates through a small half-duplex radio modem of the same type as used in robots. The radio module is connected to a PC via RS-232 port.

Automatic mission control is a software client for the base station software. It allows pre-programmed mission controls for task execution, i.e. the messages are automatically sent at a given time. A TCP/IP connection to the user interface enables also the running of the programs from distant location through the Internet.

2.4. Test environment

Practical testing of the SUBMAR society has been carried out in a special laboratory test environment, called here as *demo process*. This fully transparent process environment consists of typical process parts, like pipes and tanks (see Figure 1). The total volume of 700 litres is filled with water. In order to imitate process flow, water is circulated with a jet-flow pump. The instrumentation includes several temperature sensors, pressure sensors, and an ultrasonic flow-speed meter. Magnetic valves control hot and cold water inputs to generate plug flows and temperature gradients. Instrumentation is controlled from a PC-based standard automation system.

3. TASK DEFINITION

The task studied in the demo process is an exploration and exploitation task: searching and destroying of distributed targets. The distributed targets are supposed to be microbial algae growth spots inside the water system. Each taget has dynamic behaviour; if an algae growth is not completely destroyed in a certain location, it continues to grow. This leads to an interesting problem: what is the optimal strategy to complete successfully such task? Should the society first try to locate all the algae growth spots before starting to remove them, or should it operate immediately as soon as the first growth has been detected? Both environmental (only active vertical movement and strong currents with turbulences) and hardware constraints (energy and chemical resources) have their effect on the decision.

The cleaning task considered deserves some comments from the practical point of view. Algae growth occurs usually in a location where the water stands still. A standard solution to the problem is to insert as much cleaning agent as is needed to guarantee a minimum concentration over the whole volume of liquid. Needless to say, this approach is ineffective. The SUBMAR society provides an alternative solution where multiple small robot cooperate and transport the minimum amount of chemical needed straight to the areas where algae growths have been detected. This kind of

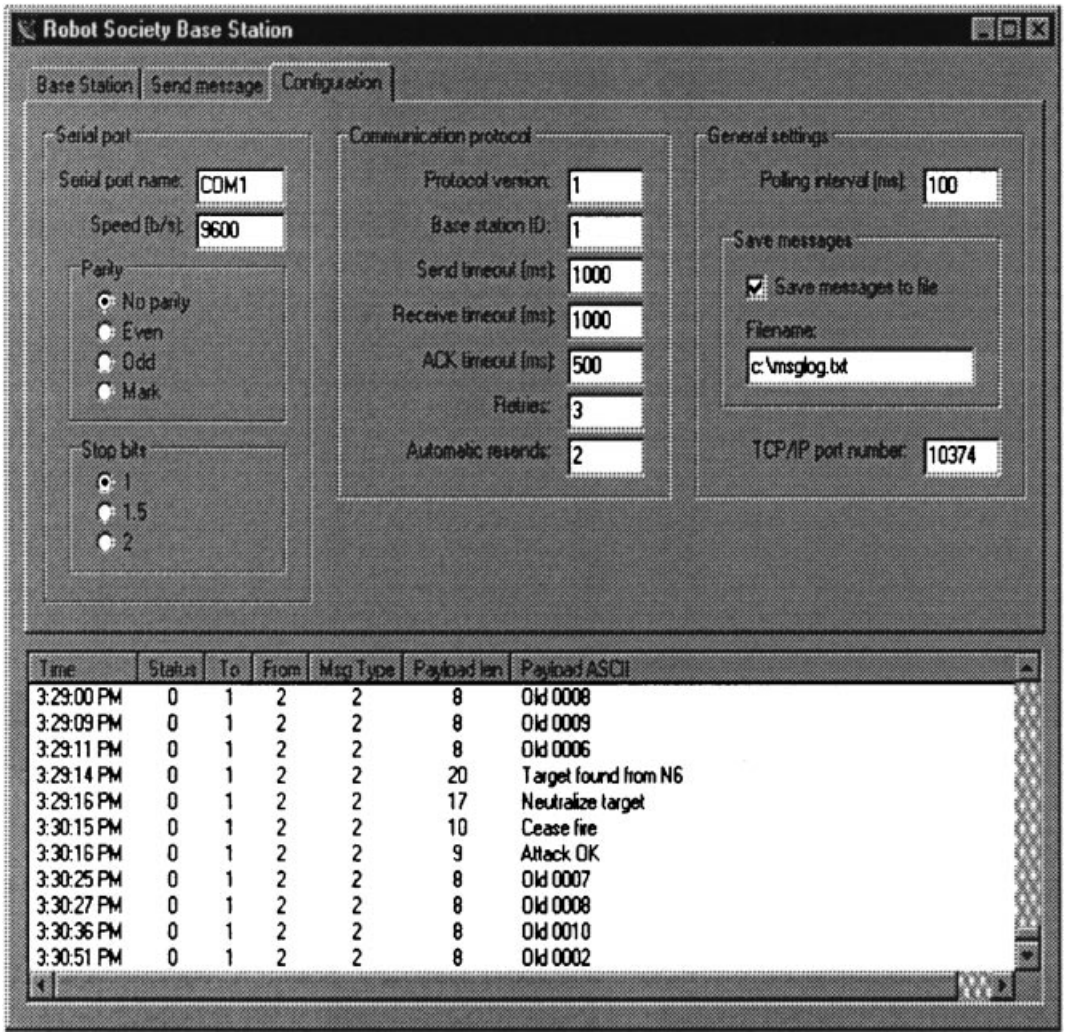

Fig. 4. User interface running in NT environment.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

policy can save a large amount of chemicals and it can be also very effective provided that the algae are not too widely distributed.

3.1. Emulated biomass growth

Because working with real algae in laboratory conditions is impractical, a special infrared system was developed for the demo process. The system can be used to describe the growth of biomass, e.g. bacteria, or algae, which tends to occur in closed water systems. In this system the growth spots are represented by infrared LED and phototransistor matrix panels, illustrated in Figure 5.

Light emitted from LEDs represent oxygen or some other gas produced by the microbial growth, which in reality could be detected by robots with an appropriate sensor. The emulated organism reacts to the presence of any cleaning agent through its phototransistors. Respectively, robots have infrared phototransistors as dissolved gas sensors, while lighting their LEDs robots emulates spreading of the cleaning agent in algae removal. There are three independent growth agglomerations installed to different locations in the demo process. This enables the analysis and evaluation of parallel multi-tasking in spatially distributed task execution. Each algae growth spot consists of four phototransistor panels. The electronics driving and sampling the actual emulated spots are controlled via I/O-card. Transmission of infrared light can be analog or frequency modulated. An analog case is more realistic, since a signal level is dependent of the measuring distance, which is a usual real-life situation.

Calculation of the current level of biomass and its growth rate in each spot is carried out in a remote PC. The behaviour of biomass is modeled with a generalized growth curve typical of most biological growth processes. The status of an algae growth, *A*, (*i.e.* the biomass) is based on formula which indicates how the derivative of the algae is related to the growth and natural death of the cells as follows:

$$\frac{dA}{dt} = (\mu - D)*A \tag{1}$$

where μ is the growth rate and D is the death rate of the organism. The value of μ depends on the limiting substrate (e.g. nitrogen). The death rate becomes meaningful when the age of the cells increase or when poisonous substrates (*i.e.* cleaning agent) are released into the environment. The actual equation used in our model, is discretized from (1):

$$A(t+1) = A(t)*e^{(\mu - D)\Delta t} \tag{2}$$

In case of a cleaning agent release, the value of D is related to the concentration of the cleaning agent (*Poison*). This value can be detected through the four inputs from the phototransistor panels. *Poison.max* is the maximum concentration of the cleaning agent in the liquid (*i.e.* the maximum value of the phototransistor panel ≈ 4.7 Volts). If *Poison* reaches *Poison.max* value, it indicates that the release of the agent had the maximum effect. The value of D is thus a function of *Poison* and *Poison.max*.

In a single robot attack the duration of this action is rather short. It equals the time that it takes for a robot to release the contents of its cleaning agent tank and for the agent to dissolve into the aquatic environment. If several robots release their chemical agents approximately at the same time, the value of *Poison* is naturally high, but the duration of the attack is short. On the other hand, if robots release their cleaning agents one after another, the value *Poison* is smaller, but the duration of the attack is respectively longer.

A generalized growth curve of a bacterial culture consists of four separate phases. These include a lag phase, exponential phase, stationary phase and death phase. These phases are shown in Figure 6, where the biomass (A) value, produced by the model, is plotted. During the exponential phase, there is an attack made by a single robot.

As a result, the value of biomass drops for a while, but it starts to grow immediately after the poison is dissolved. In

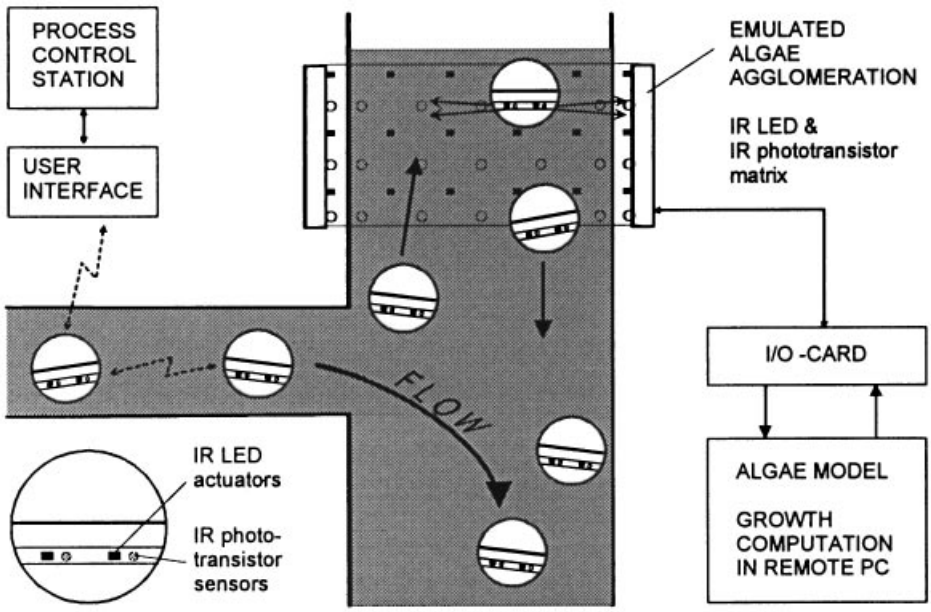

Fig. 5. Emulated algae system.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

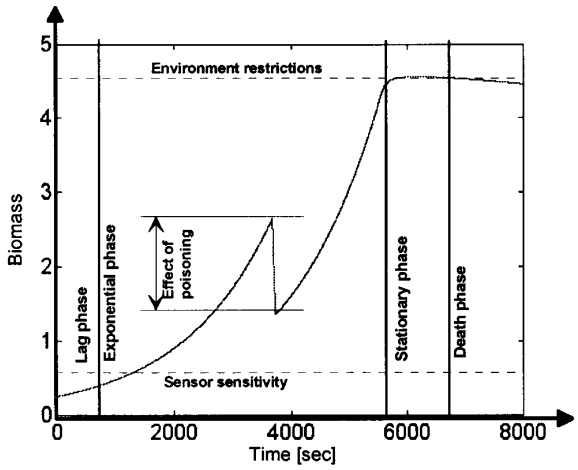

Fig. 6. Growth curve of the emulated biomass. It illustrates the four characteristic phases. Also, the effect of an attack is visible at the time of 3800 seconds.

our tests the lag phase is omitted, because during that phase the volume of the biomass is so small, that the robot cannot detect it with its sensors.

4. MAPPING OF THE ENVIRONMENT

In order to perform the task above, the members of the society, with a very limited ability to move actively, have to have some kind of spatial representation model of their environment. In a complex underwater environment any obsolute localization method is difficult to construct and requires some sort of an active beacon system or a highly sophisticated inertia navigation system.

The concept of robot society provides a simple way to solve this problem. An individual member's map need not to be highly accurate. It is enough that different process parts can be clearly recognized. The cooperation between society members will then make their maps more detailed. Mataric[19] pointed out, that even though fixing and finding of landmarks is usually based on vision, this isn't obligatory. Animals are known to use various types of landmarks, including tactile and auditory. Mataric presented a neurobiologically-feasible cognitive mapping implemented in an autonomous mobile robot, where landmarks were defined as combinations of the robot's motion and sensory inputs. The map produced by the robot contained nodes (*i.e.* landmarks) and topological links between different nodes, which indicate their spatial adjacency. We chose to represent the environment with a related method by using a directed graph.

4.1. Mapping algorithm

Mapping is based on processing a single variable, pressure. This value is used to detect events (*i.e.* nodes) where the motion character changes. An event indicates, that motion changes from downwards to certain level, from level to downwards, from upwards to level, or from level to upwards (see Figure 7). The data obtained by this way is naturally limited and open to errors.

To be able to deal with errors some kind of adaptive method had to be used. In literature[20] a concept called APN (Adaptive Place Network) was introduced; it provided a way to obtain a spatial representation and learning capability for a mobile autonomous robot. A modified version of this method was implemented to our system and it is shortly described below.

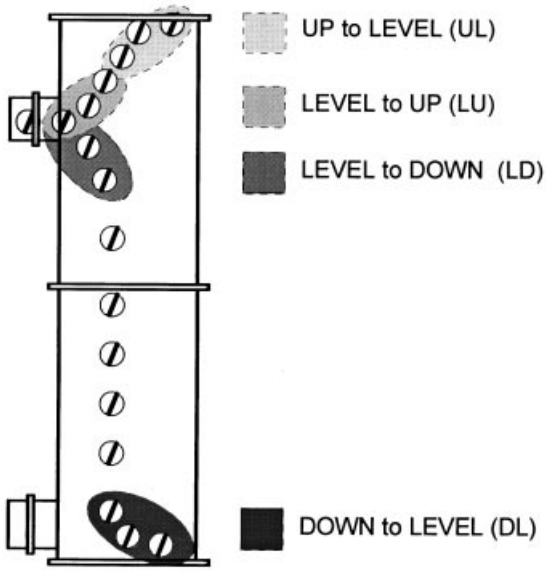

Fig. 7. Node types.

When a new link is created it is labeled with a confidence value $c \in [0, 1]$. This value basically estimates how realistic the link is, *i.e.* does it really exist, or has it occured due to a collision between robots. At the beginning of the lifespan of a link the value c is set to c_{birth}. If the robot travels through a link then the value of that link is increased with the following equation:

$$c_{t+1} = \lambda + (1 - \lambda) * c_t \quad (3)$$

where λ is the learning rate. After N_{nodes} nodes have been detected the values of all links are reduced according to the following formula:

$$c_{t+1} = (1 - \lambda) * c_t \quad (4)$$

When this value goes below a threshold (T_{kill}), a link disappears from the adjacency matrix. If all links connecting the node to the graph are deleted, then also the node is removed from the graph. Collisions with other members create nodes which are, however, eventually removed from the graph. The pseudocode of the *make.map*-algorithm[21] is shown in Table I.

4.2. Common environment representation

In a certain phase the map built by the *make,map*-algorithm in each robot reaches a stable form, where only small temporary perturbations can emerge. The size of the map stays within reasonable limits due to the reinforcement algorithm presented above. The maturation of the map can be easily detected by following the amount of new nodes *vs.* the old nodes. In a matured map the old nodes dominate clearly and only occasionally a new node is detected. At this point a robot will fix it's map and notify the operator. In order to be useful, every member of the society should have the same map. The combination of maps can be done in various ways, ranging from a fully automatic method to a map created by the operator (see Figure 8).

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

Table I: *make.map* -algorithm

1.	Take a new pressure measurement (p_i).
2.	Create input vector, i.e. three consecutive pressure measurements $I_j = [p_i, p_{i-1}, p_{i-2}]$.
3.	From I_i calculate: $S_i = (p_i - p_{i-2})$
4.	**IF** two consecutive S values are on the opposite side of threshold (T or $-$T) **THEN** a new node (a new event) has been found.
5.	The new node is compared to the linked nodes of the *current node*. **IF** matching node is found (i.e. the type of event is the same, and the difference between pressure values is smaller than T) **THEN** it is considered as the *current node.* **ELSE** the new node is compared to all known nodes. **IF** matching node is found **THEN** it is considered as the *current node*.
6.	**IF** on at least two previous algorithm cycles a link from the *current node* to the new node has been found **THEN** update *current node* data by averaging the pressure value and return the *current node* **ELSE** add a new node to the graph and return it.
7.	**IF** a link exists from the previous to returned node **THEN** increase the link confidence (Eq. 3) **ELSE** add a new link with confidence value c_{birth}
8.	Decrease all link confidences after every N_{nodes} found nodes (Eq. 4)
9.	**IF** a link confidence drops below T_{kill} **THEN** remove the link.
10.	**IF** a node is left without links **THEN** remove the node.
11.	Go to step **1**

The automatic combination is done through inter-robot communication. Several methods are available for this combination. For example, when all the members (or at least most of them) have reached the point where the map has matured, each robot transmits the map to the other robots. After receiving the maps a robot uses an algorithm that combines maps according to certain rules. The rules contain factors, like matching particular nodes, checking the confidence values for particular links, etc. As a result, each robot has the same map that contains the information from the distributed perception of the society. Another way to perform the map fusion is to allow the individual who obtains its basic map ready first to give its map to the others. This approach is fast and easy to implement, but it does not guarantee that the society has obtained the best possible map.

The map fusion task can be given also to the operator. After receiving the maps through radio, the operator can either use a special algorithm to do the combination or do it manually. The operator can also include some additional features to the map even though the robots haven't noticed them. Whatever of the previous method is used, after this phase, the society members have a common representation, a Common Basic Map (CBM), of the environment; this map enables cooperation. The quality of the map is verified time to time through a checking procedure. During the sequence each robot tracks how well CBM respresents the current situation of the environment. If lots of new nodes are detected, then a *make.map*-algorithm is run again and the old CBM is replaced with an updated CBM. In the case of the *demo process* a CBM has the form illustrated in Figure 9.

4.3. Using a Common Basic Map

The map is given to the robots in the form of a matrix L (links) and array N (nodes). Starting from a known location the robots are immediately on the map. After that they just follow the possible routes on the map. There are usually no more than three possible links from a particular node. When a node is detected the robot compares the node type and pressure value to the possible nodes in the map. Usually there is a match, and the robot stays on the map (*Old node* in Figure 10). Sometimes, however, no match can be detected. This means, that the robot has either collided with another robot or it has moved in a unusual way, e.g. interrupted a dive before it reached the node on the bottom of a tank. In case of a mismatch, the robot neglects also the following node, because non-matching nodes come always in pairs, as can be seen in Figure 10. After that the robot tries to match the next node to any node on the map. In most cases, the node is unambiguous (*New match* in Figure 10). There are some nodes, however, which are ambiguous. In those cases the robot has more than one possible match, and it cannot know for sure where it is, e.g. R2 initial location (*troubles*) in Figure 10. Based on a Common Basic Map, shown in Figure 9, when a robot arrives to node 2, there is no way to tell for sure to which node it will end up next, if it has emptied the diving tanks. Possible nodes include 3, 11 and 15. This is due to the water flow which may either suck it into the vertical pipe or let it rise up in some of the tanks. The problem is solved by checking the next node. The main principles of this algorithm called *follow.map* are illustrated in Figure 10.

5. NAVIGATION

Path planning of an individual robot is based on the fact, that a Common Basic Map is a strongly connected directed graph, *i.e.* from every node there is an access to all other nodes. The structure of the map makes it possible to use Floyd's algorithm[22] to calculate the shortest path between each node. These calculations can be done in advance based on the Common Basic Map's link *L*. The algorithm takes *L* as input and provides output in the form of same-sized matrix *D*. The matrix *D* is used in another algorithm, which calculates the shortest *node trail* from a start node to a goal node. This algorithm used depth-first search; it is also run in advance at the beginning of the mission after a Common Basic Map has been obtained. This information is stored in a struct called *ROUTE*.

The navigation method must take into account robot's limitations in maneuverability and must thus be very straightforward. It is based on the status of the diving tanks while traveling from one node to another on the map. There are three different modes for a link usage: *full(F), empty(E)* and *don't care(✗). Full* means, that the link represents diving. *Empty* indicates, that the robot is moving upwards going up based on its buoyancy. *Don't care* means that the status of the tanks is irrelevant for the use of the link. Such link represents, for example, vertical pipes, where a strong

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

flow moves the robot in any case. All this information is stored into matrix *L* indicating how the robot can move from one node to another.

The actual navigation is based on combined use of struct *ROUTE*, matrix *L* and array *N*. As an example, consider the case where the robot has to get from it's *current* node (6) to a *goal* node (12), see Figure 11. It fetches the route trail (6⇒12) from the struct *ROUTE*. The trail contains nodes 6, 7, 2, 3, 12. After that the robot uses matrix *L* to check what is the motor status from one node on the trail to the next. As a result, it obtains the following information (6⇒7: *F*), (7⇒2: *F*), (3⇒12: ⌀), and (3⇒12: *E*). Next, the robot tracks backwards from the *goal* node(12) to find out where the status of the actuators should change. In this particular case this happens when the robot reaches node 2. The link between nodes 2 and 3 is marked as *don't care(⌀)* and it can be considered to represent *E*. Hence, the navigation procedure for this case is to start with a dive in order to move from node 6 through node 7 to node 2. In node 2 the robot should empty the tanks; this should take the robot to the *goal* node 12.

If the robot detects that it has lost the trail, *i.e.* it finds itself in some other node than those listed in the node trail, it simply plans again starting from the *current* node. Losing the trail for a while is something that has to be accepted due to the limited maneuverability.

The navigation task (from node 6 to node 12) shown in Figure 11 was tested with a single robot. Twelve separate runs were performed and the duration and the routes of these runs were recorded. In five cases out of those runs, the

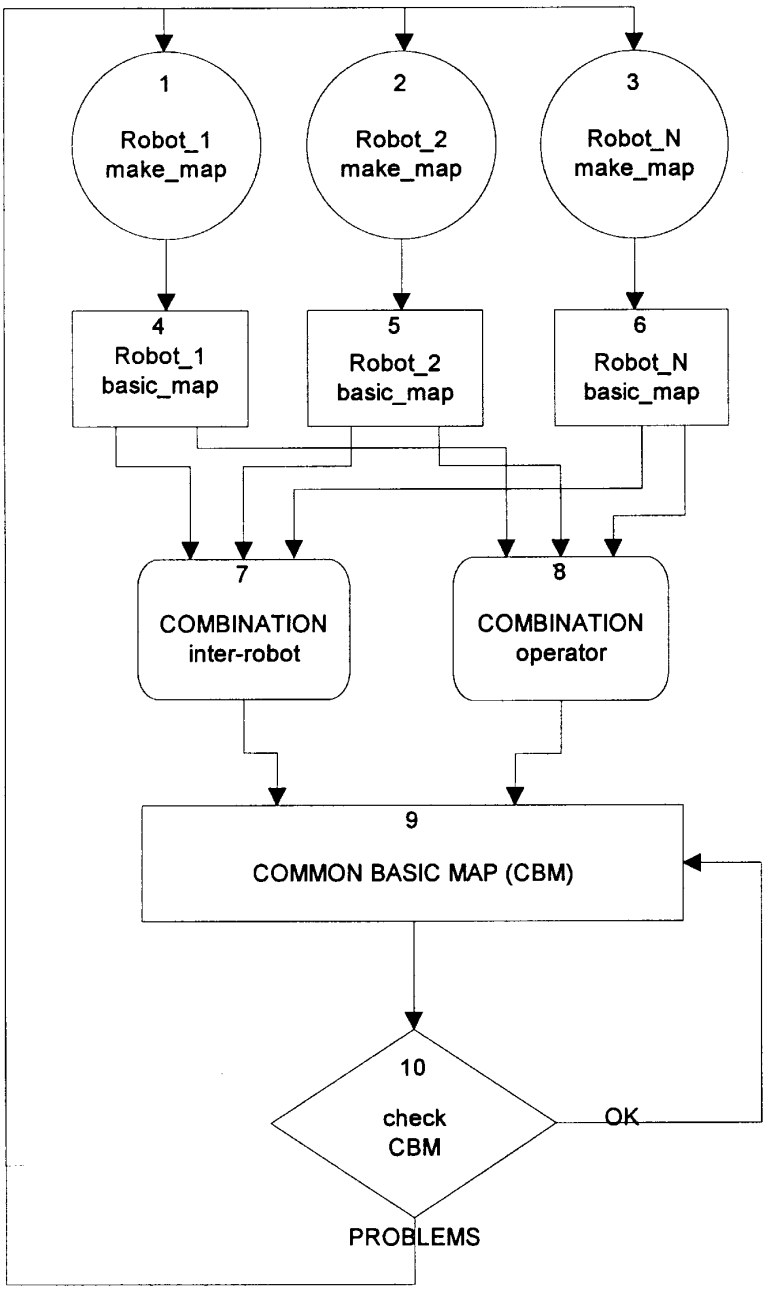

Fig. 8. Flowchart for the creation of a Common Basic Map (CBM). At first, individual robots use make.map-algorithm to get their basic maps. These maps are combined either through inter-robot communication or by the operator. As a result, every robot gets the same CBM.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

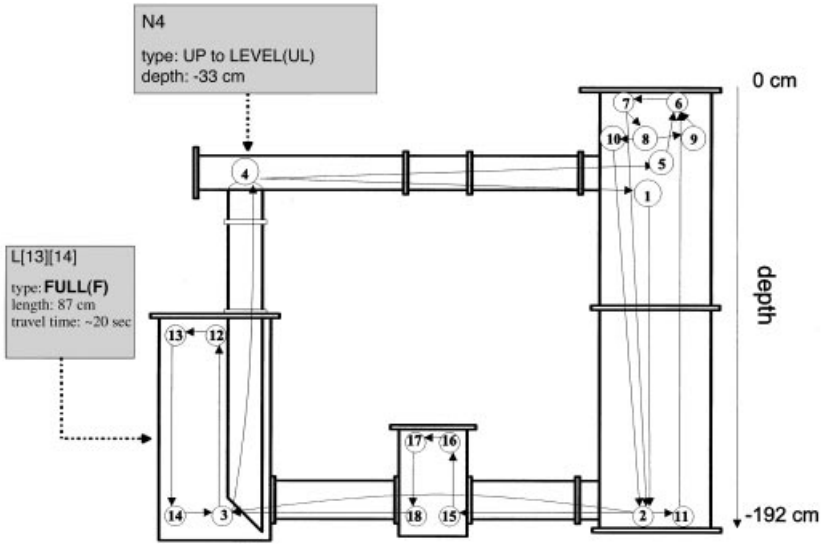

Fig. 9. The basic map drawn on top of the demo process lay-out. Nodes are represented as circles and arrow heads indicate the directions of links. Gray boxes describe data structures in node array N and link matrix L.

robot was able to navigate directly to the goal location. Three times the robot was unable to rise directly to target location due to strong current that sucked it into the vertical tube. After that, the robot replanned and succeeded in navigation, as is illustrated in Figure 12.

Three times the robot got stuck on the zero current zone of the tall tank. After it had ejected the water out of its tanks, it raised to the surface where it noticed a failure in the navigation task. After replanning, the navigation was successful, as is illustrated in Figure 13. The complete navigation algorithm is illustrated in Figure 14.

6. CONTROL ARCHITECTURE

The used control architecture for SUBMARs (see Figure 15) is a hierarchical three-layer architecture[23,24] which contains behavioural, task and cooperative layers. It is based on the FSA (Finite State Automaton) structure and behavioural action model. The theory of FSA is well-known and several studies using this method for describing relations in robotics have been done previously.[25,26] A FSA can be specified by the quadruple (Q, δ, q_0, F), where Q is the set of possible states for a robot, δ is the transition function mapping the current state q_i to the next state q_{i+1} using various types of inputs (sensors, communication, computations), $q_0 \in Q$ is the initial state, and $F \subseteq Q$ consisting of states indicating the completion of the task. In the model developed here each of these states includes a certain behavioural pattern, which the robot performs when it enters the state. The state transitions are determined by "desires" or "needs" of the robot. These, in turn, are conducted by single logical variables or more complicated performance evaluation functions.

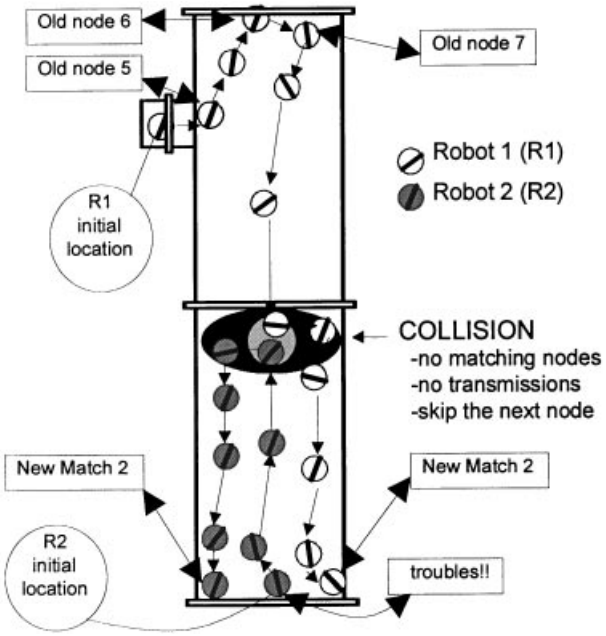

Fig. 10. Main features of *follow,map*-algorithm. The operation of the algorithm is based on the Common Basic Map (CBM). See text for details.

6.1. *Behavioural layer*

The lowest layer is the most vital for the robot's survival. The core of the layer is a FSA which determines how to respond to certain stimuli from the environment, operator, and internal sensors in order to keep the robot "alive". Furthermore, it also states for the actual task performing, but these states have a lower priority than those serving self-sufficiency and the operator's direct commands.

In the case of search and destroy mission considered here, the number of states included to the design of the behavioural layer FSA is rather small. There are six states, Q = {*recover*, *explore*, *exploit*, *load*, *notepad*, *end*} as shown in Figure 16. States *load* and *recover* represent low level behaviours with the highest priority. The actual task achieving behaviours include only *explore* and *exploit* states. The *recover*-state is the initial state, q_0, where the robot enters, when the power is turned on. It is also a kind of an emergency state, which is active when the robot detects that something is wrong with its mobility, e.g. its location has not changed even though it should have been. In this state the robot starts to use its actuators extensively.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

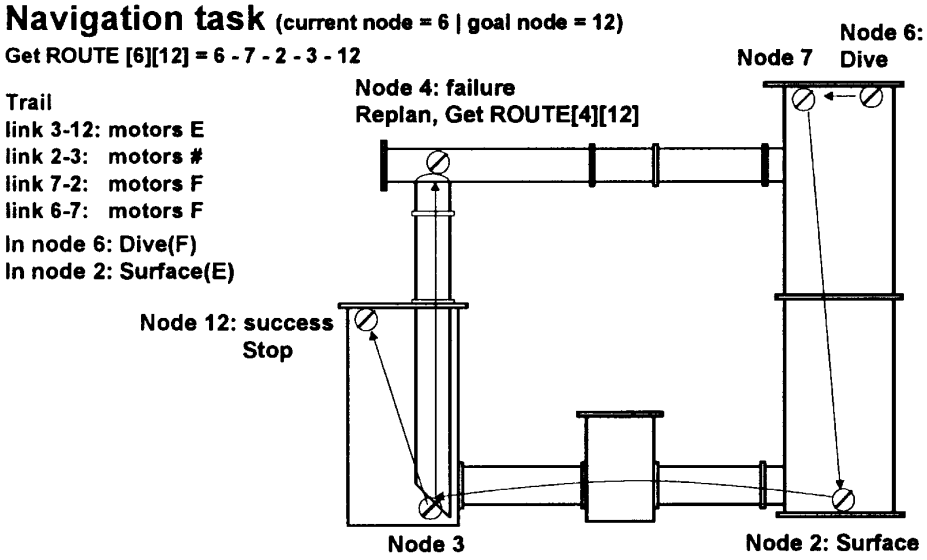

Fig. 11. Navigation is based on the use of tank actuators in particular nodes.

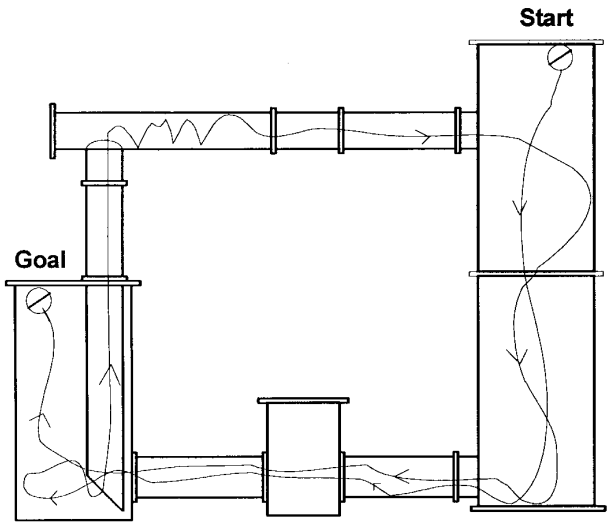

Fig. 12. The robot was forced to do an extra round.

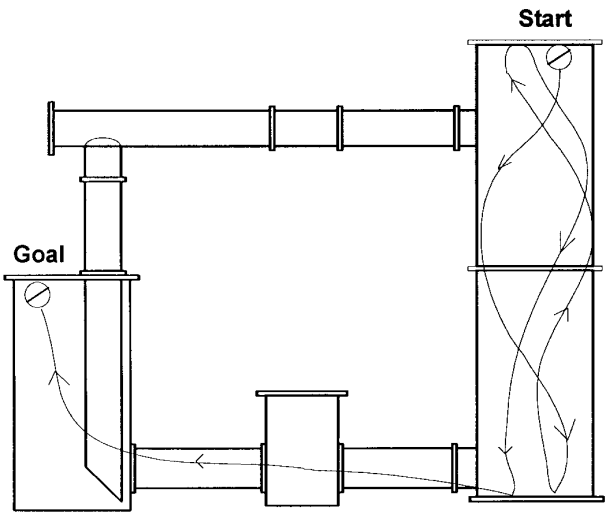

Fig. 13. Zero current conditions in certain area of the bottom of the largest tank caused the robot to miss the target at the first attempt. A second attempt was successful.

The robot changes its status to *notepad*, if *recover* has failed to make it mobile again. In this state the robot does not move anymore, and its only useful feature is to operate as a kind of beacon or message mediator until its energy resources are used up (*i.e.* abnormal termination). The robot enters the *load*-state, when detecting, that either its energy or poison level has reached some threshold value. In the *end*-state, the robot has completed the mission, or the operator has given the command to abort the mission (*i.e.* normal termination). Then the robot navigates to a defined location waiting to be removed from the *demo process*.

6.2. Task layer

In certain states of the behavioural layer the robot performs the actual tasks. These states are determined more precisely at the higher, so-called task layer. Functions in this layer define how the robot should proceed when doing its tasks. At this level the robot tries to optimize its work by choosing the most plausible strategy. In this case, two tasks are described on the task layer: *explore* and *exploit*.

In the *explore*-state the robot moves around the process trying to locate algae growth spots. The robot uses a Common Basic Map of the process and information about the location of the recharging site included to the map. Exploring evaluation is done by using CBM. A robot detects when it passes through a node in the map. By following the sequence of visited nodes it can estimate its exploring performance according to a piece-wise equation:

$$\text{explore}(t) = w_1 * a\_nodes(t) + w_2 * diff(t) \qquad (5)$$

w1, w2 coefficients

a_nodes(t) the total number of visited nodes on the Common Basic Map

$$diff(t) = \begin{cases} 0 & \text{if a\_nodes(t)} = \text{n\_nodes(t)} \\ \dfrac{1}{a\_nodes(t) - n\_nodes(t)} & otherwise \end{cases}$$

n_nodes(t) the number of consecutive nodes traveled on CBM

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

The number of found nodes on the Common Basic Map, *a_nodes*, increases initially, but at some stage this value starts to remain basically constant. The other parameter *n_nodes*, on the other hand, stays at zero for quite a while until it starts to grow. A collision with another robot drops the robot from the map, but gradually the value of this parameter starts to increase and finally passes *a_nodes*. At this point the robot is traveling through the detected CBM nodes.

Due to long time delays in the system, the *performance evaluation cycle* is rather long. Fortunately, this also prevents unwanted oscillations between different operational states. The time step is set to be equivalent to an average time it takes for a robot to circulate the process environment passively with the flow (~ 60 seconds). Hence, the actual explore performance evaluation function is the following:

$$\Delta\text{explore}_{\text{robot}} = \text{explore}(t) - \text{explore}(t-1) \qquad (6)$$

This equation rewards exploring of the environment, but when this task has been accomplished and the robot starts to go around through the same locations, the value of the function turns to negative. This indicates the fact that the robot should change its state to another task performing state, *exploit*.

In the *exploit* state the robot performs the poison release to a specific target location. While operating in the *explore* state a robot can detect an arbitrary number of growth spots. Each spot is registered with information field containing its location on the Common Basic Map and its volume. The behaviour of the robot is based on a selected strategy. Naturally, there exist several possible operational strategies, but here only four plausible strategies are considered: attacking the first detected algae growth (S1), attacking the fastest dying algae growth (S2), attacking the fastest growing algae growth (S3), and attacking the fastest dying algae growth (S4). S1 acts as a reference case. In this strategy, a robot releases its chemical agent to the first algae

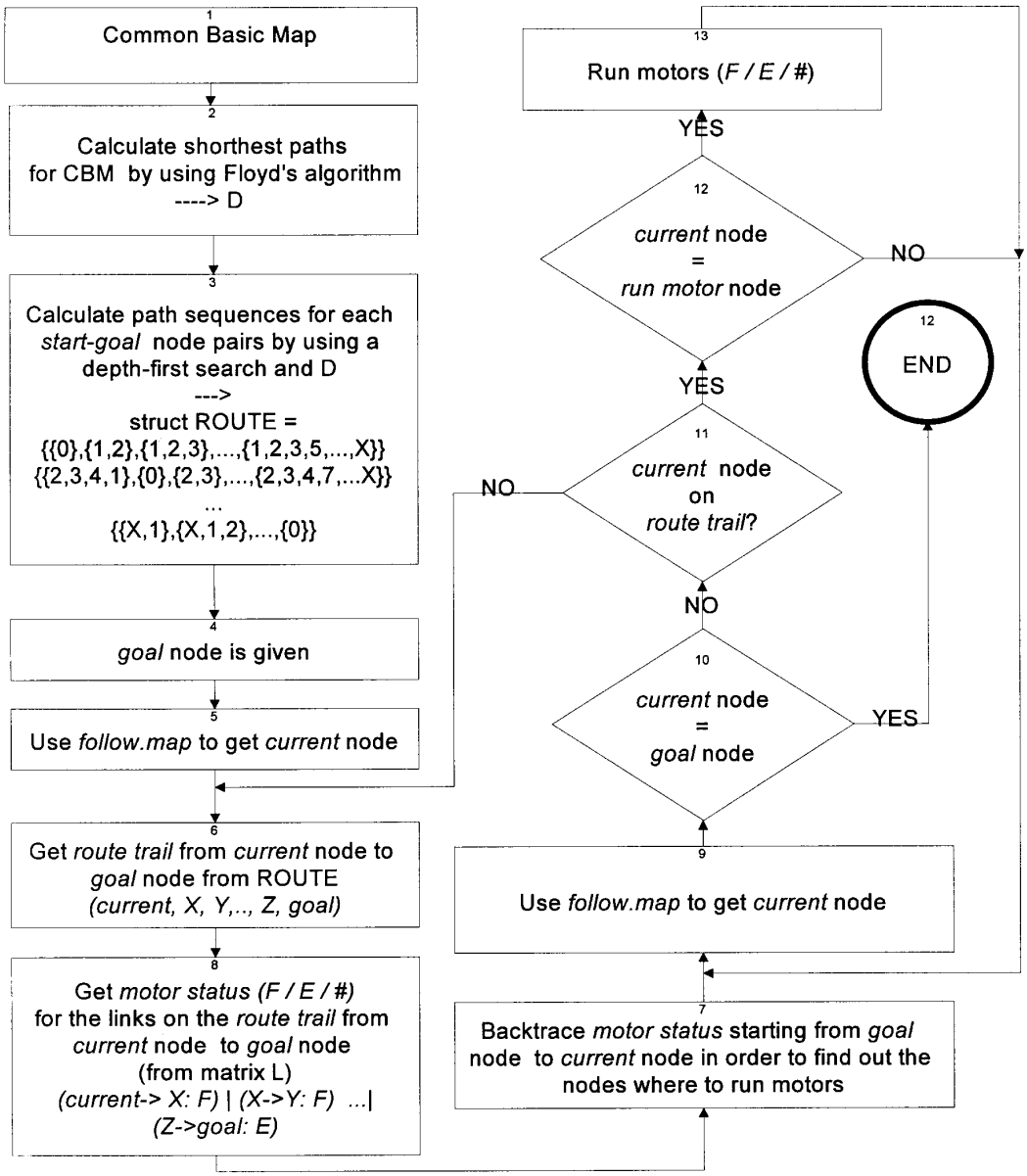

Fig. 14. Path planning and navigation algorithm is based on the Common Basic Map and route information available from it. It also uses follow.map-algorithm to keep robot on the map and to recognize when the target has been reached.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

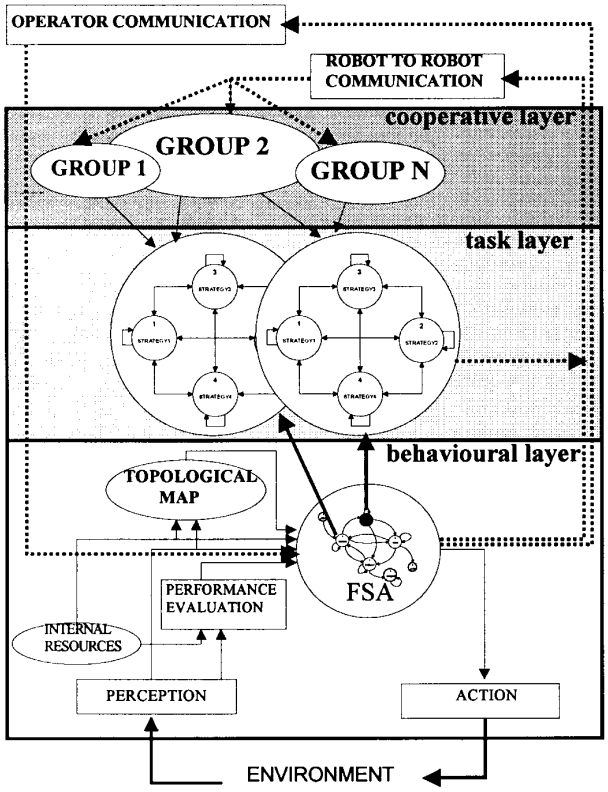

Fig. 15. Simplified illustration of the control architecture.

growth after the decision to shift to the *exploit* state. S4 is also a special strategy, because the total removal of a growth is very appealing. On the other hand, S2 and S3 are obviously also feasible selections, because large growths are naturally difficult to eliminate with small doses of a chemical. In the *exploit* state the robot monitors the status of chosen algae by storing the sensory value of the target algae growth, i.e. exploit = $\text{algae}_{\text{node(i)}}$. The location of the growth is always connected (directly to some node or temporally related to two consecutive nodes) to the basic map. Thus the exploit performance evaluation function is as follows:

$$\Delta\text{exploit}_{\text{robot}} = (\text{exploit}(t-1) - \text{exploit}(t))/dt \qquad (7)$$

In other words, the robot follows how a selected algae growth behaves during the *performance evaluation cycle*. At the beginning of the mission, the robot tests all available strategies in a row, it receives Δexploit value for each of the strategies. After this the robot chooses the most profitable one as its next strategy. Each strategy's Δexploit value is updated after testing it. Furthermore, if the chosen algae growth is already terminated, the robot forgets the chosen strategy and selects the next best strategy instead.

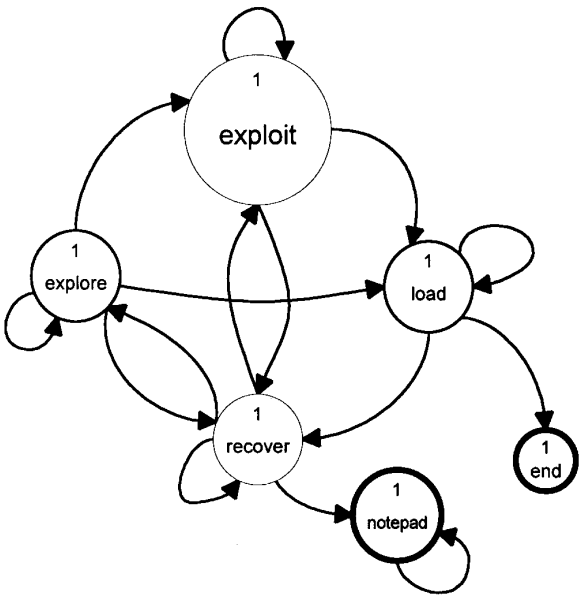

Fig. 16. The behavioural FSA.

6.3. *Cooperative layer*

Even though there would be no direct communication between the robots in the society, the members can still obtain information about the others through the environment. For example, if a robot detects that the volume of an algae growth is decreasing without its own active operation upon it, it concludes that there must be other robots performing the removal task to the same growth. This kind of indirect communication is common in Nature.[27] Nevertheless, when an active communication, no matter how simple, is allowed the performance of the society improves through cooperation.[28] An individual robot uses task related information received from the others. The communication used in the system is critical for the distance between sender and receiver, and locations of the robots in the process. The robot either receives the whole message or otherwise it is omitted due to the protocol. An inherent property of the society concept is incomplete communication between members.

In this research, the message between members contains only two types of information: strategy information and corresponding Δexploit value. Robots in certain strategy (S_i) are considered to form a group(i). Thus there can be as many groups as there are available strategies, and the number of robots (N_i) in a group can vary from one to the total number of robots in operation. For each of these groups the robot is aware of, an average value of performance is calculated. The equation form is

$$\Delta\text{exploit}_{\text{group(i)}} = (\Delta\text{exploit}_{\text{j}})/N_i \qquad (8)$$

This way the robot has a comparable value for each of the known active states. The bigger the strategy's $\Delta\text{exploit}_{\text{group(i)}}$ value is, the better the group in that strategy has performed. Thus the strategy with the highest $\Delta\text{exploit}_{\text{group(i)}}$ value is considered to represent the society's strategy as follows:

$$S_{\text{society}} = \max(\Delta\text{exploit}_{\text{group(i)}}) \qquad (9)$$

Next the individual robot compares this best group's (*i.e.* strategy's) average performance $\Delta\text{exploit}_{\text{group(i)}}$ (8) to all of its own $\Delta\text{exploit}_{\text{robot}}$ (7) values. If $\Delta\text{exploit}_{\text{group(i)}}$ is bigger than all the $\Delta\text{exploit}_{\text{robot}}$ values then the next strategy chosen is S_{society}. Otherwise, the robot chooses the next strategy based on its own highest Δexploit value.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

7. RESULTS

7.1. Simulations

The described control architecture was implemented first into a 3D-simulator, shown in Figure 17. The simulator was coded with Open GL in Silicon Graphics Indigo2 to represent the *demo process* with a complex flow and diffusion dynamics. Due to the simple environment sensing (*i.e.* only pressure and algae detection sensors) the "reality gap" between simulation and real world does not grow too wide. The results are quite comprable.[29]

The aim of the simulation studies was to study how the performance of the society changes when the size of the society is varied. The performance of the society was evaluated based on three parameters: *status of the algae growth*, *survival of the robots*, and *duration of the mission*. These parameters give the operator a possibility to guide the society into a preferable action by altering the weight of the parameters. In some missions the accomplishment of the task has the utmost importance. In some other "hard to reach" application (e.g. space, underwater), the unit cost for a robot is so substantial both financially and mission-wise that every robot must stay functional throughout the mission. Furthermore, in some other cases time is the most valuable resource, e.g. in nuclear power plant accidents.

Throughout testing the environmental conditions (initial algae volumes and growth rates, distance related communication probabilities, flow, etc.) were kept constant. The interference caused mainly by the competition of space was studied by varying the size of the society (3, 5, 7, 10, 15 robots). Three robots were clearly too small a group to finish the task but, on the other hand, 15 always caused a deadlock situation. This "traffic jam" was caused by a large group of robots simultaneously trying to enter an up-going narrow pipe. When comparing societies with 5, 7 and 10 members, it became obvious, that there were no significant differences in task performing nor in elapsed time, as can be seen in Figure 18.

Nevertheless, the increase in the number of robots in operation had a correlation to the untimely "death" of some robots. Consistently, out of 10 robots 2–3 run out of energy, and out of 7 robots 1–2 "died". When the size of the society was 5, all survived and performed the required task. Considering *the status of biomass*, *the survival of the robots*, and *the duration of the mission* it seems that the optimum number of robots for the task in hand would be 5.

7.2. Tests with real robots

Even though the simulator was considered to give comparable results, the real value would only be obtained through testing with a physical robot society. When writing this paper, there are 7 robots in operation. Full scale testing of ideas presented in this paper is yet to be done, but most of the features have already been verified. The mapping and navigation systems are in full use as well as the emulated algae growth system. Actually, the only part, which has not

Fig. 17. The simulator.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

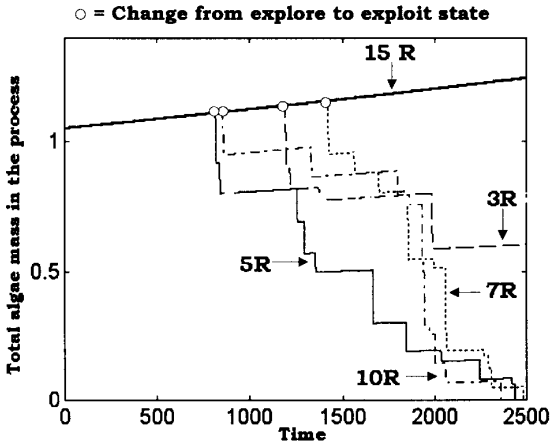

Fig. 18. The task performing of various sized societies. It illustrates how too large (15 robots) or too small (3 robots) societies fail in the exploiting task, whereas medium size societies (5, 7, 10) are able to finish the task approximately at the same time. Time scale is in seconds.

been fully implemented, is the cooperative layer of the model.

In the preliminary tests shown below, the members of the society have obtained their *basic maps*. These maps are then combined by the operator into a *Common Basic Map*, which is then given to the members. After that the robots use CBM and *follow.map* -algorithm in order to perform the *exploit* task. In Figure 19 a member is shown in one of the emulated algae growth spots (Node 6 in CBM). After detecting the growth, the member releases its cleaning agent to the environment, *i.e.* lights its IR LEDs. The result of this attack is shown in Figure 20. As can be seen, the volume of the biomass drops quite fast, but due to the limited amount of cleaning agent onboard the member cannot wipe out the whole growth.

In order to find out whether a member of the society could single-handed remove a medium size algae growth, a long term test was performed. In this test, the robot attacks only one growth spot. After releasing the cleaning agent, the robot leaves the growth spot and goes to a location, where its cleaning agent storage is refilled. The results of the test

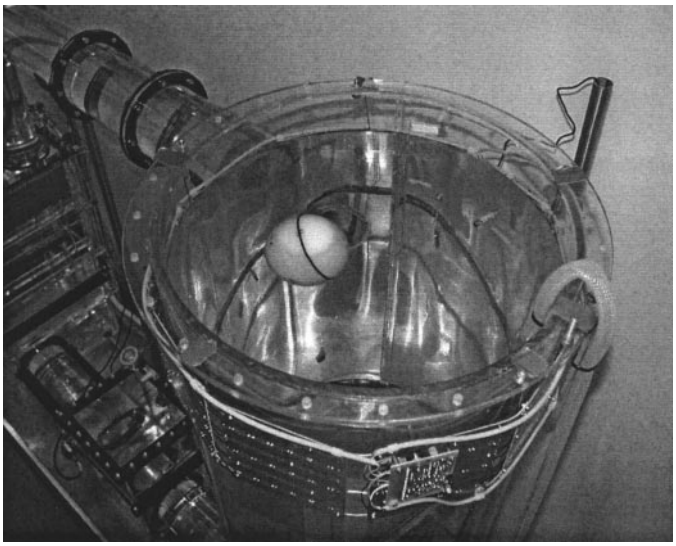

Fig. 19. A single member has found an emulated algae growth.

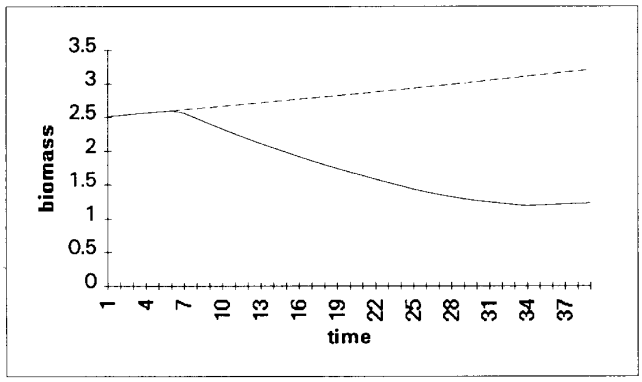

Fig. 20. The effect of attack made by a single member. The dashed line illustrates the normal growth of the biomass.

are shown in Figure 21. It can be clearly seen, that a single robot cannot destroy the algae growth. At the beginning of the test, just after the first attack, there is a long delay before a new attack. This is due to the poor maneuverability of the robot, *i.e.* it could not get back to the algae growth location at the first try. Delay in navigation resulted to a large growth in algae volume. After that, the robot was able to repeat attacks. Even though there was an extremely successful attack at time 800 seconds, and another soon after that, no complete elimination could be demonstrated.

Next, a group of three robots were used to attack the same growth spot, shown in Figure 22. The initial volume of the biomass and growth rate were naturally identical as in the single robot case. The results shown in Figure 23 demonstrate that three robots are enough to kill the growth totally. In the case shown here, the three robots released their cleaning agents sequentially, one after another. Small delays between attacks are visible in the form of small growing phases in an otherwise down-going trend.

The results shown here are only preliminary, and not from a complete system; they nevertheless indicate that the system is operational and ready for the final testing. Final results will be published in near future in the form of two doctoral theses

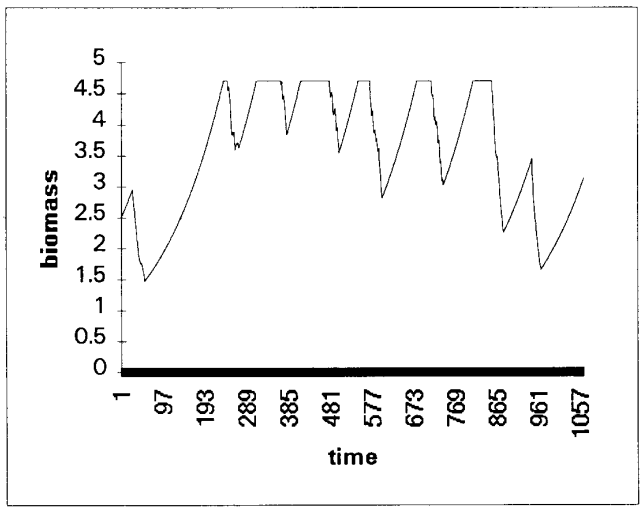

Fig. 21. Long-term test with a single robot. The robot fails to destroy the algae growth.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

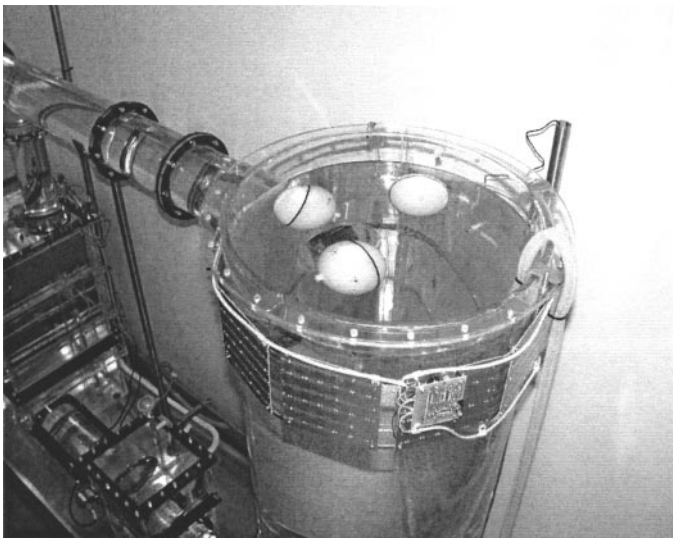

Fig. 22. Group power. Three robots attacking the same algae growth location.

8. CONCLUSIONS

The robot society formed by small-scale underwater SUBMAR robots and a generic three-layer control architecture for their cooperative functioning have been developed. To enable practical testing and optimization of the control structures, emulated algae removal was defined as their task. Performance evaluation of the system has been carried out both as a simulator study and by experimenting with real robots in a test environment.

The simulated results demonstrated that in a closed, restricted environment, including narrow pipes, junctions and tanks, the robots having minimalistic mobility and perception, were able to complete the task using their control structure, but spatial interference of multiple robots affect their task performing strongly. This indicates, that in a distributed system an optimal number of robots for a given task could be found.

Testing with real robots has not reached a fully operational level. Two levels out of three (i.e. *behavioural*- and *task performing level*) in the control architecture structure have been implemented and tested. In practice, mapping of the environment, navigation, task execution against multiple targets, and behaviour of the emulated algae growth have been verified with real robots already. In the near future, the *cooperative layer* will be completed. This allows full-scale testing of the robot society concept.

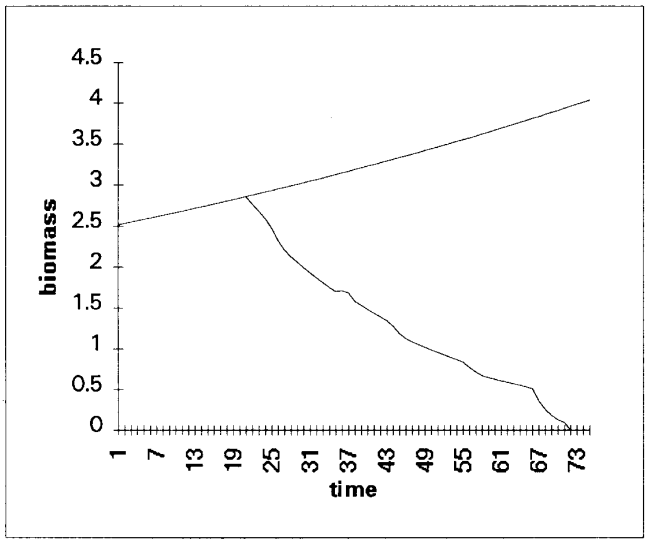

Fig. 23. The effect of an attack made by three robots in a row. The mission is completed and the algae growth is eliminated.

References

1. M.S. Fontan and M.J. Mataric, "A Study of Territoriality: The Role of Critical Mass in Adaptive Task Division", *Proc. From Animals to Animats 4* (4th Int. Conf. on Simulation of Adaptive Behavior) (P. Maes, M. Mataric, J.-A. Meyer, J. Pollack and S.W. Wilson; eds.) (MIT Press/Bradford Books, 1996) pp. 553–561.
2. A. Halme, P. Jakubik, T. Schönberg and M. Vainio, "The concept of robot society and its utilization", *Proc. IEEE/ Tsukuba Int. Workshop on Advanced Robotics*, Tsukuba, Japan (1993) pp. 29–35.
3. M.J. Mataric, "Issues and Approaches in the Design of Collective Autonomous Agents", *Robotics and Autonomous Systems* **16**, Nos. 2–4, 321–331 (Dec., 1995).
4. P. Maes, "Modeling Adaptive Autonomous Agents", **In**: *Artificial Life- An Overview*, (c. Langton, Ed.) (The MIT Press, Cambridge, Mass, 1995) pp. 135–162.
5. T. Fukuda, S. Nakagawa, T. Kawauchi and M. Buss, "Structure Decision Method for Self Organizing Robots based on Cell Structure - CEBOT", *Proc. IEEE Int. Conf. on Robotics and Automation* (1989) pp. 695–700.
6. H. Asama, A. Matsumoto and Y. Ishida, "Design of an Autonomous and Distributed Robot System: "ACTRESS"", *Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, Tsukuba, Japan, (1989) pp. 282–290.
7. K. Dautenhahn, "Getting to Know Each Other- Artificial Social Intelligence for Autonomous Robots", *Robotics and Autonomous Systems* **16**, Nos. 2–4, 333–356 (Dec., 1995).
8. M.J. Mataric, "Learning to Behave Socially", *Proc. of From Animals to Animats 3*, (3rd Int. Conf. on Simulation and Adaptive Behavior), (D. Cliff, P. Husbands, J.-A. Meyer and S.W. Wilson; Eds.) (MIT Press, (1994) pp. 453–462.
9. L.E. Parker, "Heterogeneous Multi-Robot Cooperation", *Ph.D. Thesis* (Massachusetts Institute of Technology, Artificial Intelligence Laboratory, MA. MIT-AI-TR 1465, 1994).
10. T. Balch, "Behavioral Diversity in Learning Robot Teams", *Ph.D. Thesis* (College of Computing, Georgia Institute of Technology, 1998).
11. P. Appelqvist, A. Halme, T. Schönberg, M. Vainio and Y. Wang, "Designing simple cooperative sensor/actuator robots for liquid process environments", *Proc. IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (CD-ROM)*, Tokyo, Japan (1997) pp. ???
12. P. Appelqvist, M. Vainio and A. Halme, "Mechanical design of underwater sensor/actuator robots for cooperative task execution", *Proc. Mechatronics '98* (The 6th UK Mechatronics Forum Int. Conf.), Skövde, Sweden (1998) pp. 249–254.
13. C.R. Kube, "Collective Robotics: From Local Perception to Global Action", *Ph.D. Thesis* (Computing Science, University of Alberta, 1997).
14. D.C. MacKenzie, R.C. Arkin and J.M. Cameron, "Multiagent mission specification and execution", *Autonomous Robots* **4**, No. 1, 29–52 (1997).
15. S. Goss, J.-L. Deneubourg, R. Beckers and J.L. Henrotte, "Recipes for Collective Movement", *Proc. 2nd European Conf. on Artificial Life* (preprints) (1993) pp. 400–410.
16. J.H. Sudd and N.R. Franks, *The Behavioural Ecology of Ants* (Chapman and Hall, New York, USA, 1987).
17. T.D. Seelay, *Honeybee Ecology – A Study of Adaption in Social Life* (Princeton University Press New Jersey, USA, 1985).

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press

18. J.A. Shapiro, "Bacteria as Multicellular Organisms", *Scientific American* **258**, No. 6, 62–69 (1988).
19. M.J. Mataric, "Navigating With a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation", *Proc. From Animals to Animats* (1st Int. Conf. on Simulation of Adaptive Behavior) (J.-A. Meyer and S. Wilson; Eds.) (MIT Press, Cambridge, MA, 1991) pp. 169–175.
20. B.M. Yamauchi, "Exploration and Spatial Learning in Dynamic Environments", *Ph.D. Thesis* (Case Western Reserve University, 1995).
21. A. Halme, P. Appelqvist, P. Jakubik, P. Kähkönen, T. Schönberg and M. Vainio, "Bacterium robot society – a biologically inspired multi-agent concept for internal monitoring and controlling of processes", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Osaka, Japan (1996) pp. 1707–1714.
22. R. Sedgewick, *Algorithms (2nd edition)*, (Addison-Wesley, New York, USA, 1988).
23. M. Vainio, P. Appelqvist, T. Schönberg and A. Halme, "Group behavior of a mobile underwater robot society destroying distributed targets in a closed process environment", *Proc. 3rd IFAC Conf. on Intelligent Autonomous Vehicles*, Madrid, Spain (1998) pp. 112–117.
24. M. Vainio, P. Appelqvist and A. Halme, "Generic Control Architecture for a Cooperative Robot System", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Victoria, Canada (1998) pp. 1119–1125.
25. R.A. Brooks, "Robust Layered Control System For a Mobile Robot", *IEEE Journal of Robotics an Automation* **RA-2**, No. 1, 14–23 (1986).
26. R.C. Arkin and D.C. MacKenzie, "Temperal coordination of perceptual algorithms for mobile robot navigation", *IEEE Transactions on Robotics and Automation* **10**, No. 3, 276–286 (1994).
27. R. Becker, O.E. Holland and J.L. Deneubourg, "From Local Actions to Global Tasks: Stigmergy and Collective Robotics", *Artificial Life IV* (R.A. Brooks and P. Maes; Eds.) (MIT Press, Cambridge, MA, 1995) pp. 181–189.
28. T. Balch, R.C. Arkin, "Communication in reactive multiagent robotic systems", *Autonomous Robots* **1**, No. 1, 27–52 (1994).
29. M. Vainio, A. Halme, P. Appelqvist, P. Kähkönen, P. Jakubik, T. Schönberg and Y. Wang, "An application concept of an underwater robot society", *Distributed Autonomous Robotic Systems 2* (H. Asama, T. Fukuda, T. Arai and I. Endo; Eds.) (Springer-Verlag, Tokyo, 1996) pp. 103–114.

https://doi.org/10.1017/S0263574799002222 Published online by Cambridge University Press