The Society of Naval Architects of Japan

Application of a Cooperative Control System to the Collision Avoidance Guidance of AUVs

E. A. de Barros*, *Member* H. Maeda*, *Member*
S. Miyajima*, *Member*

Summary

A decentralized control system is proposed for the control architecture of an autonomous underwater vehicle (AUV). The software concept is based on the object-oriented model and its specialization, a "agent oriented model". Such control system was implemented on a network of transputers. First tests of such control system were related to the collision avoidance guidance for an AUV which follows the sea bed profile. The guidance system combines a pursuit guidance algorithm and a grid map built in real-time, which is based on the information acquired from a sonar system. First tests concerned simulations using a model of the vehicle Pteroa-150. Experiments were also carried out using the carriage system of a model basin adapted to a test bed for underwater robotics. A robotic system was achieved by using an automatic pursuing system aimed originally to experiments with free running ship models, and attaching underwater acoustic sensors to one of the carriages. Thus, automatic guidance in collision avoidance experiments was made possible for investigating the response of the cooperative control system when it faces the input of a real sensory system. Computer simulations and experimental tests have shown the usefulness and feasibility in real-time of the map assisted guidance system.

1. Introduction

Providing autonomy for an untetherered underwater vehicle implies in the implementation of a variety of functions inside it for attending its mission goals. Such functions can be classified relatively to concepts like environment perception, analysis, planning of actions, and execution. The software architecture responsible for controlling and coordinating these functions in real -time is the robot's control architecture.

In this paper, a control architecture based on distributed agents is discussed, the "cooperative control system"[1]. It differs from other related schemes proposed for AUV control by the computational and artificial intelligence frameworks it is based on, and their respective relationship. Moreover, it is applied to a real problem during an AUV mission, the collision avoidance guidance.

The vehicle considered was the Pteroa-150, for which a collision avoidance algorithm based on sensory data from a sonar system had been proposed before. Limitations on such algorithm motivated the development of a new one.

Simulations were used for comparing the performance of the collision avoidance algorithm itself with that of the cooperative control system. In addition, experimental tests of the cooperative control system that related real acoustic perception with maneuver decisions in real-time were carried out using a new test bed proposed for underwater robotics.

2. Introduction to a Cooperative Control System

The research on AUVs has been influenced by the recent advances achieved in mobile robotics. The progress in computer technology (hardware and software), together with new approaches proposed for robot control systems, provided simpler and more feasible implementations than before. Particularly, since the middle of the last decade, new concepts in the field of artificial intelligence have changed the design of control systems for mobile robots. Discussions on artificial intelligence paradigms and methodologies applied to the design of control architectures for mobile robots can be found in a number of references 1), 2), 3), 4).

The cooperative control system to be presented in this section, has been influenced by the "message passing paradigm" in computer science, and a kind of its specialization: the "agent oriented programming" in AI.

* Institute of Industrial Science, Univ. of Tokyo.

Received 10 th July 1995
Read at the Autumn meeting 16, 17 th Nov. 1995

NII-Electronic Library Service

The Society of Naval Architects of Japan

Originally, desired features in three basic areas of an AUV control system: hardware, AI model, and the software implementation, motivated the proposed control architecture.

2.1.1 Hardware Performance

A control system based on parallel distributed processing implemented on a transputer network was considered. A parallel distributed system can provide modularity and fast response for a number of concurrent tasks that need to be executed in real-time. Another inherent characteristic is the possibility of conferring graceful degradation to the control system, taking into account the possibility of occurrence of faults or erratic responses inside isolated control units.

2.1.2 Control System Conception

A set of control units is modeled as a community of agents. They are autonomous entities that coordinate their actions through communication protocols. The "agent" model, that is introduced in the section 5 can be viewed as an specialization of the *Actors* model[5),6)]. Thus, a more complex system than the hardwired approach of the subsumption control[3)] is used for the implementation of patterns of relationship among control units. However, communication protocols define some boundaries in the exchange of information and commands between the modules.

The proposed control system is supposed to be *cooperative* in the sense that control units help each other in some degree for performing their tasks better. Thus, one of the desired characteristics of the control architecture is the possibility of joint solutions to be achieved for solving global problems that must be faced during a mission. Cooperation can be achieved in a number of different ways, such as exchanging information, negotiating task assignments, and deciding on interactions. For the cooperation to be effective, control units should be coordinated. Inspired on the *Actors* model, coordination is achieved through predefined behavioral scripts, and communication protocols. Furthermore, based on the adopted "agent oriented' model, cooperation is implemented by explicit rules that are expressed through commitments between one control unit to another.

2.1.3 Software Implementation and Computational Modeling

Actors are related to object-oriented languages. Particularly, they can be considered as basic types of objects that can be analyzed and explored theoretically. The advent of object-oriented languages in the beginning of the decade of 1970, was one of the factors that motivated the research on these computational entities. Furthermore, *Actors* are basic components of concurrent systems. Computations are executed by Actors sending messages to one another. For receiving communications, each Actor has a reference address or *mail address*. Communications are asynchronous. All information, or knowledge is local to an Actor, and invisible to the others.

The action produced by an *Actor* is activated by incoming communications. For a given communication the correspondent action is determined by the *Actor* current *behavior*. Behaviors themselves are determined by internal states of the Actors.

In the original *Actor* formalism, messages are enqueued according to the order of arrival. An Actor can only accept messages placed at the head of the queue. Another limitation is the restriction in the messages exchanging mode. After sending a message, even if an Actor expects an answer, it proceeds its computation. Thus, the response will be one of the future incoming messages, and some way for distinguish it should implemented before sending the original message.

The ABCM/1, "An object-Based Concurrent Model 1"[7),8)] has evolved from the *Actors* model, and copes with such limitations in the original model. Moreover, an implementation based on the ABCM/1 was developed, the ABCL/1 language. This is one of the few object-oriented languages for parallel processing. In the cooperative control system, part of the message passing mechanisms proposed in ABCM/1 were implemented for exploring a more flexible communication structure.

The *Actors* model, and the parallel object orientation can be implemented to some extent using transputers, and the language Occam[1)]. Occam is derived from a theory for concurrent processes, the CSP[9)]: Communicating Sequential Processes. The CSP theory can be used as an algebraic tool for analyzing a discrete-event control system. The former application is related to the collision avoidance guidance system that we have developed. Moreover, we believe that many other aspects of a concurrent cooperative control system can be tested and developed using CSP and Occam.

3. Basic Features of the Control Architecture

3.1 Distributed Control

Each control unit is implemented in one or more processors. Communications between any two processors are only based on message passing. There is no shared memory like in Blackboard based architectures. There is no fixed priority for decisions of one control unit when compared to another one. The system is heterarchical. Decisions are taken based on shared perspectives, and cooperative criteria.

3.2 Decomposition Criterion

The units are designed for performing general internal functions like in the classical control architectures. Thus, for example, there may be units in charge of functions like guidance, navigation, map-building, etc. (Fig. 1).

3.3 Physical Grounding

Actions are specified in terms of implemented behaviors. However, behaviors are grouped inside functional units, as defined above, not decomposed in layers, like the system proposed by Brooks[3)]. The activation of a

NII-Electronic Library Service

The Society of Naval Architects of Japan

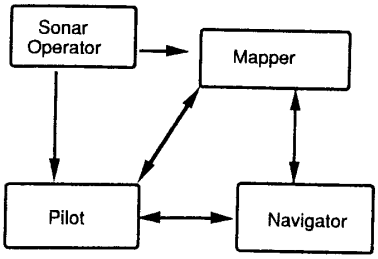

Fig. 1 An example of the Cooperative Control System. The arrows represent communication between control units through message passing.

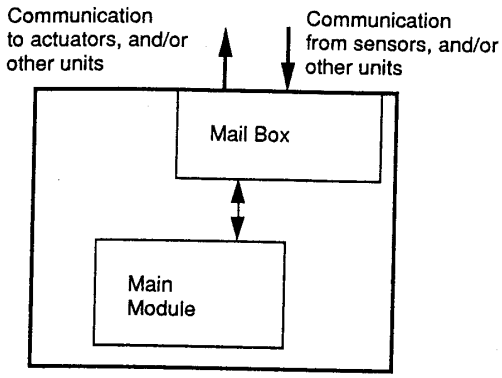

Fig. 2 Block diagram of a control unit.

particular behavior is decided by a control module inside the correspondent unit.

3.4 Use of Representations

World models are usually taken into account by employing representations close to the raw sensory data. One example used in this thesis is the occupancy grid-map. One exception concerns the case of missions in engineering controlled environments, where more abstract models can be considered. Other elaborated forms of representation would be used for modeling parts of the vehicle itself, like dynamic models of devices and/or expert systems. They are implemented following the object oriented modeling. Internal representations of other units can be also included inside control units, according to the agent based approach.

3.5 Implementation

The application runs on a transputer network using the language Occam, whose characteristics of concurrence and communication between processes are explicitly represented. Control units are formed by at least two objects: the main control module, and the "mail box". Other objects may be also considered depending on the function of the control unit. In order to permit asynchronous communication using Occam, control units need to be mapped into more than one Occam process. Therefore, a dedicated object has been developed for buffering and sending messages, the "mail box" (Fig. 2).

4. A Case Study: The Collision Avoidance Guidance System for an AUV

Concurrently to the investigation and development of a control architecture, the application of it to a practical problem during an AUV mission was considered: the sonar based navigation of an AUV, which has been developed at the University of Tokyo, the "Pteroa-150"[10]. The mission considered is the exploration of the sea bed by terrain following maneuvers in the vertical plane. A similar vehicle has been developed by the company KDD for the inspection of underwater communication cables[11].

The perception of the relative position between vehicle and terrain occurs through the use of range data from a sonar system. In simulation tests, a model of the vehicle "Pteroa-150" that includes 4 active echo sounders was used. The echo sounders are arranged around the gravity center of the vehicle aiming at directions spaced by 30 degrees from each other, from the body longitudinal axis, "x", to the transversal axis, "z", as shown in Fig. 3.

Guidance laws based on terrain following, and pursuit guidance are used together with the line of sight angle as basic tools for the local path planning system. They are based on the local and absolute reference systems shown in Fig. 4.

4.1 Terrain Following Algorithm: First Version

A collision avoidance guidance for "Pteroa-150" was first proposed by Ishitani et al.[12]. A similar algorithm was used in the first simulations[13], and in the experiments presented in this paper.

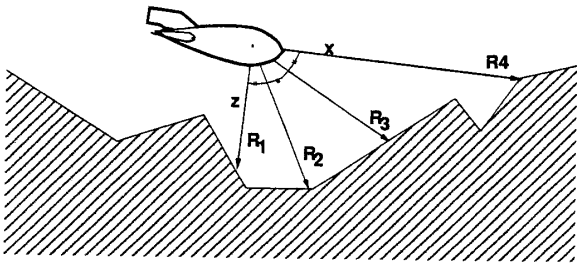

Fig. 3 Ultrasonic beams representation and the corresponding slant-ranges.

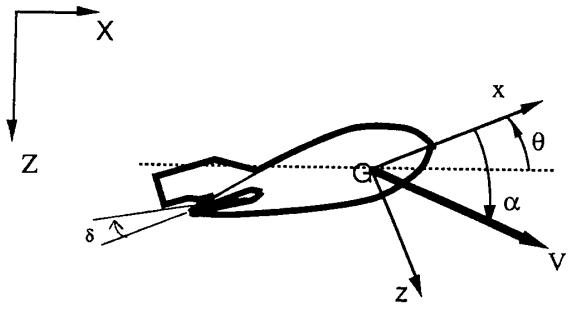

Fig. 4 Coordinate Systems: absolute, and body-fixed.

NII-Electronic Library Service

The Society of Naval Architects of Japan

The algorithm uses a piece wise approximation of the terrain profile obtained through the slant-ranges values that correspond to the acoustic beams. Based on the slant ranges related to two adjacent beams, local estimations of the terrain slope relatively to the vehicle can be estimated, as follows:

$$b_i=-\tan^{-1}((z_i-z_{i-1})/(x_i-x_{i-1})),\quad i=2,3. \tag{1}$$

, where

$$z_i=R_i^*\cos[(i-1)^*\pi/6]$$
$$x_i=R_i^*\sin[(i-1)^*\pi/6] \tag{2}$$

During terrain following, the main objective is to keep a minimum altitude "z_0" from the bottom, following its profile. The command pitch angle to be input into the control algorithm is given by:

$$\theta_{ref}=\theta_0+\theta+\Delta\theta \tag{3}$$

, where:

θ_{ref} is the command pitch angle,

θ_0 is the trimming angle,

$\Delta\theta$ is the pitch angle variation, whose calculation is based on the terrain perception by the sonar system.

Naturally, the main reference for the vehicle's altitude from the bottom is the distance "R_1" obtained from the echo sounder parallel to the "z" axis, i.e. directly bellow the vehicle. Using just this information, the pitch angle variation would be calculated as:

$$b_1=-\tan^{-1}((R_1-z_0)/\lambda) \tag{4}$$

, where λ is the reference distance in the "x" direction within which the error in the altitude should vanish.

A more explicit collision avoidance maneuver occurs in the presence of a forward obstacle within a distance bellow a defined limit, i.e. "$R_4<$limit". In such case,

$$\Delta\theta=\tan^{-1}(b_4) \tag{5}$$

where, $b_4=(R_1/R_4)$, using the terrain slope as reference, or $b_4=(z_0/R_4)$, as in the algorithm based on pursuit guidance (section 4.2).

In this guidance algorithm, it is imposed a restrictive criterion for using the information given by such angles: the information ahead of the vehicle is considered only when the referred terrain altitude is bellow the minimum required. Thus, the correspondent projection of the slant-ranges onto the "z" axis must be less than the required minimum altitude "z_0". Only the maximum value between "b_2" and "b_3" is used, which is compared to "k^*b_1". At the time of the first tests, a large value for "λ" was adopted. Thus, the gain "k" was increased (or "λ" decreased) for altitudes less than "z_0". This can be expressed as follows:

Let,

$$b_{23}=\theta+\max\{b_2, b_3\}. \tag{6}$$

If $z_2<z_0$, or $z_3<z_0$, then

$$\Delta\theta=\max\{k^*b_1, b23\}, \tag{7}$$

else, $\Delta\theta=b1$ (8)

where, $k>=1$, if $b1>0$, or, $k=1$, if $b1<0$.

4.2 Terrain Following Algorithm Based on Pursuit Guidance

An improvement in the former approach was achieved after adapting guidance methods commonly used in air missile guidance to our purpose. In this case, terrain following is viewed as the task of intercepting a imaginary target traveling at constant altitude from the terrain The basic element in the algorithm is the line of sight angle, relating the pursuer (the AUV in our case) to the target. Considering the absolute reference frame, the line of sight angle can be represented as:

$$\sigma_{abs}=\tan^{-1}\left(\frac{Z_{target}-Z_G}{X_{target}-X_G}\right) \tag{9}$$

, where, X_G and Z_G are the vehicle coordinates.

The pitch angle variation calculated using altitude range, can also be interpreted in the sense of the line of sight. Considering the expression (4), the desired downward range variation can be interpreted as a distance to the target in the direction defined by the local "z" axis. Similarly, "λ" is the distance difference between target and vehicle in the longitudinal direction "x". Thus,

$$\sigma_0=\tan^{-1}\left(\frac{z_{target}}{x_{target}}\right) \tag{10}$$

, where:

$$x_{target}=\lambda \tag{11}$$
$$Z_{target}=z_0-R_1. \tag{12}$$

Imposing the line of sight angle as an ordered pitch variation to the control system consists in a pure pursuit guidance. The idea is to keep the vehicle longitudinal axis always aiming to the target along the line of sight. Since the longitudinal axis and the velocity direction are not always coincident, i.e. the angle of attack is different from zero, guidance errors arise. Therefore, better results are obtained aiming the velocity vector to the target. This can be achieved by adding the angle of attack value to the line of sight angle. In this case, the guidance law changes to the so called velocity pursuit:

$$\Delta\theta=\sigma_0+\alpha \tag{13}$$

Assuming a linear model related to the vehicle forward velocity, and small values of the pitch angle, the angle of attack can be approximated by:

$$\alpha=\frac{\dot{Z}}{U}+\theta \tag{14}$$

, where:

$\dot{Z}$ is the vehicle depth rate,

U is the constant forward velocity.

A rough terrain corresponds to a maneuvering target. In such case, terrain following by pure altitude control may not provide response to the vehicle as fast as required. Besides the intrinsic limitation of the algorithm, the vehicle dynamics impose limitations to the guidance performance. This is a significant difference from the air missile case, where there is a high pursuer velocity related to the target, and fast control rate is allowed. Such limitation restricts the minimum value of λ for assuring the stability of the system "guidance+control+vehicle", as analyzed in 14).

Guidance performance can be improved by using information on the target velocity for anticipating its future position. In the terrain following case, the analogous information is the terrain slope. For estimat-

NII-Electronic Library Service

The Society of Naval Architects of Japan

ing the local terrain slope, the slant ranges relative to the beams at 30 and 60 degrees are used in "b_2", defined by (1), and "b_3'", defined as:

$$b_3' = -\tan^{-1}((z_3 - R_1)/x_3) \quad (15)$$

The slope calculated in this case is related to the intersection of the beam coincident with the "z" axis, and the terrain profile.

Employing again the missile guidance analogy, the algorithm proposed is a kind of deviated-pursuit lead guidance. The lead angle is proportional to the line of sight rate, that is calculated using the expressions for "b_2" and/or "b_3'". If both values are considered, the average value between them is adopted. The expression for the pitch variation is calculated as follows:

$$\Delta\theta = \sigma_r + \alpha \quad (16)$$

where:

$$\sigma_r = K_P * \sigma_0 + K_D * \sigma \quad (17)$$

$$\sigma = \begin{cases} b2, \text{ if } R2 \leqslant \lim 2, \text{ and } R3 > \lim 3 \\ b'3, \text{ if } R2 > \lim 2, \text{ and } R3 \leqslant \lim 3 \\ \dfrac{b2 + b'3}{2}, \text{ if } R2 \leqslant \lim 2, \text{ and } R3 \leqslant \lim 3 \\ 0, \text{ otherwise} \end{cases} \quad (18)$$

K_P, and K_D are control gains.

The values for lim 2, lim 3, and the control gains are function of the required minimum altitude "z_0", and the minimum preview distance that assures the system stability. The relation with "z_0" is due to the rule that both beams should be considered only when the vehicle is near to the required altitude, for avoiding unnecessary maneuvers. This implies in projections of the beams on the "z" axis whose values are not distant from the minimum altitude "z_0". On the other hand, the ratio between the projections on the "x" axis and the control gains can not bee too small , otherwise the system looses its stability, as in the case of too small values of "λ".

In the case of $R_4 <$ limit, $\Delta\theta$ is calculated by (5).

The pitch angle is controlled through deflections of the vehicle's elevators. The discrete control law adopted is given by

$$\delta_e(t) = k_1 * (\theta_{\text{ref}}(t) - \theta(t)) - k_2 * \omega(t) + k_3 * \delta(t-1) \quad (19)$$

, where:

δ_e is the elevator angle

ω is the pitch ratio

k_1, k_2, and k_3 are the controller gains.

In Ishitani et al.[12], the gains are changed in order to prevent high attack angles when following the terrain, and to provide fast responses for collision avoidance. In our simulations, only the "damping" k_2 is changed (i. e. increased) when the absolute value of a negative pitch rate "ω" is above an arbitrary limit. The other gains are kept constant.

4. 3 Cooperative Guidance

The performance of the latter algorithm in smooth terrain is satisfactory[1]. Problems appear in the case of steep terrain or in the presence of abrupt changes in the terrain profile (Fig. 5). One of the reasons for such drawback is the small number of transducers that imply in the non-detection of obstacles between the narrow ultrasonic beams. Considering such limitation, a significant drawback is due to the guidance algorithm itself: only present detection of obstacles are taken into account by the guidance algorithm. When a forward obstacle is detected, the vehicle nose turns up as expected. As soon as no other forward obstacle detection occurs, the vehicle is commanded to follow the terrain bellow it, and turns its nose down, neglecting the last command for avoiding the obstacle. Despite the limitation on the number of beams, probably there was a number of times that an obstacle was detected. However, such information is not taken into account in future occasions. As a result, the vehicle may approach an obstacle too closely in a kind of fluttering motion. This can be observed in the graphs of the pitch angle in Fig. 5.

The solution for this problem was considered in the framework of the cooperative control system. Storing the obstacle detections is a function of the *mapper* module. It exchanges information with the *pilot*, which is responsible for guidance and motion control.

The representation used in the *mapper* is a bi-dimensional grid map which is a simplification of the "occupancy grid framework"[15]. The range data received at each sampling time is interpreted according to a probabilistic sonar model, and incorporated into the

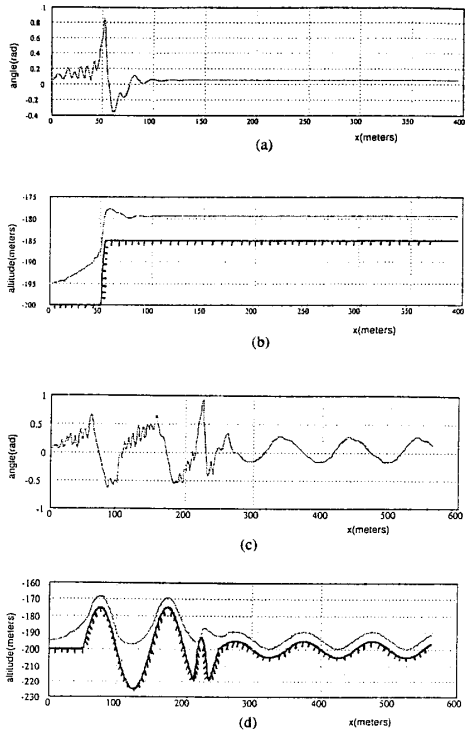

Fig. 5 Pure Terrain Following Algorithm. (a) Test 1-pitch angle; (b) Test 1-trajectory; (c) Test 2-pitch angle; (d) Test 2-trajectory.

NII-Electronic Library Service

The Society of Naval Architects of Japan

map. Instead of the gaussian model adopted by Elfes[15], we have assumed in our first tests an ideal sensor concerning the radial distances. For each measurement, angular sectors are defined by the sonar beam width. Each cell inside the sector which has a probability density function value greater than zero is considered "occupied", otherwise it is "non-occupied". Other cells not scanned yet are considered to be in the "unknown" state. Thus, the probability density function is given by:

$$p(r/m, \varphi)=\delta(r-m)\left(\frac{1}{2\sigma_\varphi}(u(\varphi+\sigma_\varphi)-u(\varphi-\sigma_\varphi))\right) \tag{20}$$

where:

r is the random variable representing the real range.

m is the measured range.

φ is the random variable representing the target angle related to the transducer longitudinal axis.

σ_φ is half of the beam width.

δ is the impulse function.

u is the unit-step function

It was emphasized the dynamic interaction between *pilot* and *mapper* rather than the use of better models in the process of map building. Such interaction is intended to cope with the uncertainty or lack of information that affects these modules during their correspondent tasks. The *pilot* consults the information in the map through requests to the *mapper* and decide on maneuvers. On the other hand, the incomplete information in the map may activate survey type maneuvers by the *pilot* that help the *mapper*.

As the vehicle is moving over the sea bed, new range and position data are received, and used in order to update a local map of the terrain. Concerning position, the vehicle coordinates are defined in relation to an earth fixed reference frame. It is adopted a fixed orthogonal system with relation to the local area considered. As new local maps are built, the origin of the reference system is displaced.

A search for the highest obstacle representation in a small area, and free spaces over it is done in the *mapper* module when requested by the *pilot*. The search process is performed over a small part of the grid map, using a window whose origin is at the reference coordinates received from the *pilot* (Fig. 6). It is implicit in such procedure that the vehicle moves in a fixed sense in the longitudinal direction. Thus, only obstacles after the reference point are taken into account. The window size should be large enough for making effective the map utilization. However, if the size is too large, the search becomes more time consuming, and obstacles more dangerous due to their short distance to the vehicle may lose importance in face of higher obstacles at further positions. Another drawback of excessive large windows is the possibility of increasing the importance of isolated obstacle detections that are due to large beam width of the sonar system, and not to the real existence of obstacles. After finding the highest obstacle detection inside the window, the search for free spaces over the correspondent coordinate starts. A fixed number of cells at the column over the obstacle detection are checked. If the number of free space cells exceeds a given limit, the obstacle coordinates are transmitted to the *pilot* as a local maximum. Otherwise, the same coordinates together with an information on the unknown status of such point are returned. Using such approach, a small number of cells with obstacle detections found inside free space areas is not taken into consideration. Considering the terrain continuity, isolated cells corresponding to obstacle detections are generally due to the beam width of the sonar system, and should not influence collision avoidance maneuvers.

In the *pilot* module, different maneuver states can be activated depending on the situation. This refers basically to a forward obstacle detection and the information received from the *mapper*. Maneuver states and the interaction with the mapper can be represented through a finite state automaton defined by the 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, where $Q=\{q_0, q_1, q_2, q_3\}$ is the set of maneuver states, $\Sigma=\{1, 2, 3, 4, 5\}$ is the input alphabet, q_0="terrain following" in Q is the initial state, and $F=\{q_0\}$ is the set of final states, and δ is the transition function represented in Fig. 7 which maps $Q\times\Sigma$ to Q. It is assumed that the terrain following is the initial and final state in the transition diagram. The maneuver states and the corresponding transitions among them are described as follows:

$q0$: "*Terrain Following*". In this state, if there is no forward obstacle detection, the vehicle is exclusively

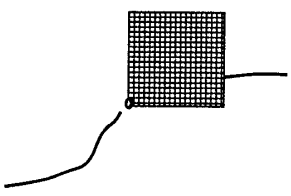

Fig. 6 Window used for the search process. The origin "O", at the corner of the window, has its coordinates given by the *pilot*, when it requests the search.

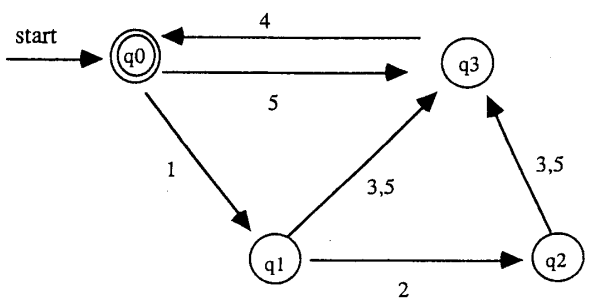

Fig. 7 Transition diagram.

NII-Electronic Library Service

The Society of Naval Architects of Japan

guided by the terrain following algorithm, and no communication with the *mapper* occurs. In the presence of a forward obstacle, two possible transitions to other states are permitted, depending whether or not the measured range is within a defined limit. If the forward obstacle is far enough (input "1") a communication with the mapper is triggered. The mapper is asked whether or not free space is found over the coordinates of the forward obstacle. The obstacle at the highest altitude near the received coordinates is searched by the mapper, and also the detected free spaces over it. The new coordinates and the information on the presence of clearance or not, is returned to the pilot. In the affirmative case, the pilot continues to guide the vehicle to follow the terrain bellow it. Otherwise, the vehicle is commanded to turn up, by the maximum value between the terrain following algorithm and the line of sight angle , "LOS" (i. e. σ_{abs}, as in (9)), using the coordinates received from the mapper as the target. In both cases, the maneuver state used at the next sampling time is changed to "survey", and the coordinates received from the mapper are stored as references values for future decisions. If the forward range is bellow a given limit (input "5") the vehicle is commanded to turn up by the "LOS" law, using the "X" coordinate of the forward obstacle and its depth subtracted by "z_0" as the target coordinates. No communication with the mapper is triggered, and the next maneuver state will change to "collision avoidance".

$q1$: "*Survey*". At this state, there is a process of investigation on the obstacle at the highest altitude in a small area that includes the reference coordinates. As described in the former paragraph, depending on the information available at the moment by the *mapper*, the *pilot* decides to proceed following the terrain, or to search for free spaces turning up the vehicle's nose. While new obstacles are detected by the forward beam at distances greater than the limit, this maneuver state is maintained. The original reference point coordinates are updated when either free spaces are reported by the *mapper* or no more obstacles are detected by the forward beam (input "2"). In both cases, the maneuver state is changed to "attention". Otherwise, the control system continues in the "survey" state until a limit distance to a forward obstacle (input "5") or to the reference point occurs (input "3"), and then it changes to the "collision avoidance" state.

$q2$: "*Attention*". This is the next stage following the "survey" state. The same course of actions followed at the survey state are used. The difference in this case is the knowledge already acquired on the nearby terrain which allows the vehicle to just follow the terrain profile bellow it most of the time.

$q3$: "*Collision Avoidance*". At this state, the vehicle is in a turning up maneuver that can be started by inputs "3" and "5". The ordered pitch angle at each instant is determined by comparing the "LOS" angle to the reference point with the output given by the terrain following algorithm, and taking the maximum value. New forward obstacles detected at this state are taken into account if the corresponding "LOS" pitch angles are greater than the value related to the reference point. Even if no forward obstacle is detected, if the reference point has not been overcome, the pilot proceeds the maneuver to the reference point which has its coordinates updated through consults to the mapper. Having just passed over the reference point (input "4"), the pilot returns to the "terrain following" state.

In all the states, calculations of the "LOS" angle assume that "Ztarget" is subtracted by the margin "z_0". The maximum measured range assumed was 50 m, and the limit distance used in "input 3" and "input 5" was 25 m.

The performance of this guidance system when facing smooth terrains is equivalent to the former terrain following algorithm[1]. In the case of steep terrains, the vehicle could usually dive deeper than in the correspondent test with the terrain following algorithm. The fluttering phenomenon was reduced, and a safe distance from abrupt changes in the terrain was achieved (Fig. 8).

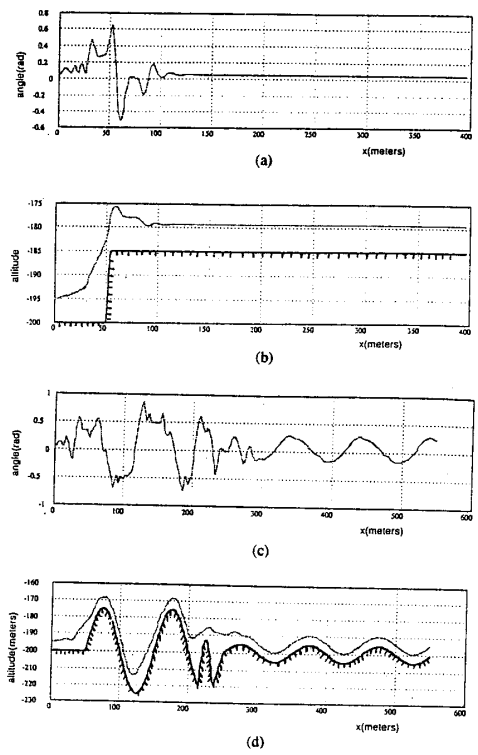

Fig. 8 "Cooperative Guidance" (a) Test 1-pitch angle; (b) Test 1-trajectory; (c) Test 2-pitch angle; (d) Test 2-trajectory.

NII-Electronic Library Service

The Society of Naval Architects of Japan

5. Interpretation of Cooperative Guidance System through the ABCM/1 and AOP Frameworks

The cooperative guidance discussed so far was based on the communication mechanisms included in the ABCM/1 , and CSP techniques implemented in Occam-2. The Agent Oriented Programming , AOP[16], framework can be used for representing the same guidance system. The AOP model can provide a more sophisticated representation, which we believe is more appropriate for future developments on higher level tasks inside an AUV control architecture. In fact, the AOP is proposed as the "artificial intelligence framework" in our control architecture.

Part of the AOP framework and its version of the cooperative guidance system is introduced in this section. Further details and implementation issues involving Occam-2 are given by de Barros[1].

The AOP can be considered as an specialization of the object-oriented model (in the original "Actors" formalism). Objects are renamed as "agents", and the states of each of them are constrained to special types called "mental states". The parameters defining mental states are: beliefs (on facts), commitments or obligations (to other agents or to oneself) and capabilities (the relationship between one agent mental state and its environment). Such special kind of states can be used not only to define commands to actuators but also for determining in real-time the relationship among control units. Messages are also constrained to three types: "INFORM" (a fact), "REQUEST" (an action) and "UNREQUEST".

Corresponding to each parameter aforementioned, there is a data base inside each agent implementation considered. For example, in the pilot module, the capability data base expresses the ability for any maneuver which is believed to be safe:

($\forall$ maneuver, B (maneuver, safe))

In the commitment data base, it is supposed that maneuvering near the terrain is an obligation from the *pilot* to the *mapper*. Thus, the complex set of maneuvering states is seem as a result from a request by the *mapper*. It does not need to be so. However, in such way, it is exemplified the heterarquical character of the control system. The location of decisions is not fixed: in this example, if safety conditions are not achieved for proceeding in a maneuver near the terrain, the *pilot* can decide to emerge the vehicle or to cruise in a safe depth.

The obligations included in the commitment data base are expressed by:

OBL (pilot, keep. maneuvering), and OBL (mapper, maneuver. near. terrain)

The first sentence expresses the obligation of the pilot which itself in order to maneuver the vehicle. The macros "keep. maneuvering", and "maneuver. near. terrain" are a combination of actions expressed by:

macro keep. maneuvering:
DO emerge OR (cruise. at. constant. depth OR maneuver. near. terrain)
macro maneuver. near. terrain:
DO follow. terrain
  ELSE
    DO careful. terrain. following
      ELSE
        DO investigate. terrain
          ELSE
            DO collision. avoidance. near. terrain

The terms "careful. terrain. following", "investigate. terrain", and "collision. avoidance. near. terrain" are used instead of the former terms "attention", "survey", and "collision. avoidance", respectively. The correspondent maneuvers are the same, however.

The activation of each of the instances in the action "maneuver. near. terrain" is a result of updating beliefs on the safety of each of them inside the *pilot's* belief data base. According to the rule in the capability data base, maneuvers considered unsafe are not activated . The choice among safe maneuvers is guided by the priorities expressed in the macro "maneuver. near. terrain". These priorities, together with belief updating procedures work equivalently to the rules for changing between maneuver states in the cooperative guidance system presented in the former section. The mentioned procedure for belief updating on safe maneuvers, and the maneuver execution are included in the macro "check. data&go", which is a parameter of one of the commitment rules inside the *pilot* module:

OBL (sonar, INFORM, data, pilot, check. data&go)

In this rule, the *pilot* decides (or has a commitment with itself) to execute "check. data&go", after having received a "INFORM" type message from the *sonar operator*.

An agent is capable of an action if and only if it believes itself to be, and only commits to actions of which he is capable. Thus, belief updating may imply changes in the capabilities data base, which may imply changes in the commitments data base. After completing all updating procedures, and before the *pilot* receives new data from the *sonar operator*, a maneuver is executed by the "actions" module. The execution of actions to which the *pilot* is committed runs concurrently with the updating of the three data bases, "commitments" , "beliefs", and "capabilities".

The *mapper* module includes commitment rules regarding the construction of the grid map and the search over it when requested by the *pilot*:

OBL (sonar, INFORM, data, mapper, represent&update)
OBL (pilot, REQUEST, check. reference. coordinates,

NII-Electronic Library Service

The Society of Naval Architects of Japan

pilot, search. free. spaces&inform)

In both versions of the map based guidance system, cooperation between the *pilot* and the *mapper* was exemplified. In the AOP version, all the system of maneuvers expressed by the action "maneuver. near. terrain" was the result of a commitment from the *pilot* to the *mapper*. The *mapper*, as in the first implementation of the guidance system, assisted the *pilot* providing information about the terrain. The assistance from the pilot to the mapper is expressed by the "survey" maneuver state. In this case, the vehicle's nose is turned up in order to provide to the *mapper* more information about a particular site of the terrain. An action with the same purpose was implemented through a steerable acoustic beam system, which is part of the experimental tests described in the section 7.

6. A novel Test Bed for Experiments in Underwater Robotics

In de Barros et al.[17], the use of the carriage system of a seakeeping basin guided by a parallel distributed control system is proposed for investigating a number of aspects of an underwater robot operation. Most of the necessary facilities are already available, requiring only the adaptation of the carriage motion control system, and the installation of the sensors related to the perception system considered. Without the usual restrictions in space of an actual vehicle, this test bed can carry sensors, computers, and researchers, for a number of experimental tests.

The carriages are mounted over the basin of the University of Tokyo (Fig. 9). An important facility related to motion control is a precision model tracking system. Through such system, the difference between the model position and the sub carriage is transformed in an electrical signal by means of a potentiometer. Such signal drives the automatic control system of the X-Y carriages, changing their velocities in order to track the model . When the control system operates at this state, that is denominated "auto-tracking mode", a non-zero fixed value for the forward velocity is set initially. Such value is supposed to be the main forward velocity of the model. Starting from the constant forward velocity condition, the motion of the main carriage is changed in order to driven the deviations in relation to the model position, "Δx", together with the accumulated tracking error to zero. The same scheme is applied to the auto-tracking mode of the sub-carriage, except for the non-necessity of setting the sub-carriage initial velocity.

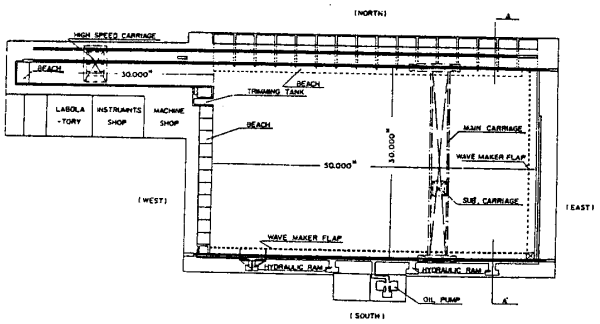

Fig. 9 Arrangement of the Basin.

Another important feature available at the sub-carriage, is the possibility for obtaining positions and velocities of both carriages in real-time. The position variables are defined according to a Cartesian system of coordinates whose origin is at the center of the tank. Through the data from wheel shaft encoders, the carriage velocity and displacement are calculated, and sent from the mini-computer, responsible for the tracking control, to the sub-carriage terminals.

The transformation of the carriage system described above in a test bed for experiments on underwater robotics was performed in two steps. The first step is a connection of the original control system with a separate processing unit, where decisions and guidance commands to the control system of each carriage are created. Such connection provides input signals to the mini-computer from the new processing unit, that will have the effect of guidance commands. For this purpose, the previously described precision tracking system was used. Instead of real deviations between the motions of a ship model and the carriages, the equivalent analog electrical signal , that would be provided by the potentiometer, is sent directly from the separate processing unit through a digital/analog converter.

The second step concerns the connection of a sensory system with the processing unit for the environmental perception, and the access to the vehicle's motion. An underwater sonar system was installed into the sub-carriage, and used for measuring distances to objects immersed into the tank. The slant-ranges provided by the sonar system as well as the carriages' speed measurements are fed into the processing unit through an analog/digital converter. A simple proportional law for the velocity control of the carriages was implemented. A schematic representation of the tracking system connected with the automatic guidance system is shown in Fig. 10.

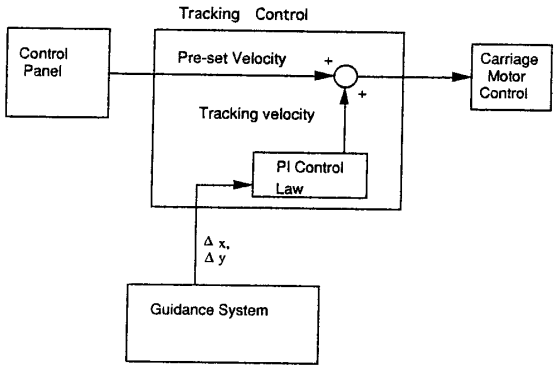

Fig. 10 Automatic guidance through the adaptation of the tracking system.

NII-Electronic Library Service

The Society of Naval Architects of Japan

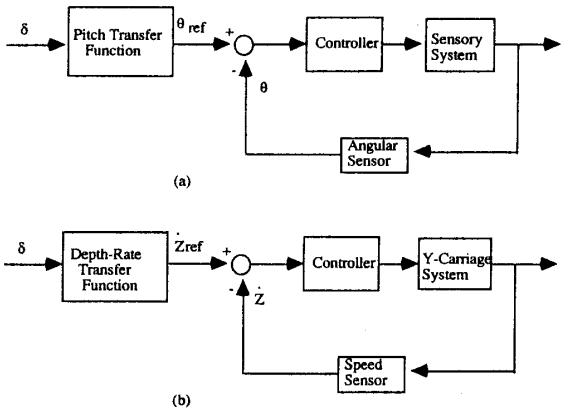

Fig. 11 Block diagrams of the motion simulators: a) Pitch motion; b) Depth rate.

A transputer network connected to the bus of a personal computer through an interface board was chosen as the additional processing unit. The system implementation is done by a "Transputer Compact System", or "TCS"[18], that includes an user friendly interface for network configuration. A TCS can be connected with another one, and so successively , providing scalability to the network. One TCS connected to a personal computer was used in the first experiments with the new test bed, concerning the sonar based navigation of an underwater vehicle.

In the first tests using such platform, it was investigated the relationship between real sensing and maneuver decisions in real-time. In order to include a simulation of an AUV dynamics, future improvements are intended. Linear models based on the estimation of hydrodynamic derivatives or transfer functions from system identification tests have been investigated by the authors[19),20]. Having estimated a model for the AUV, the inclusion of its dynamics in the experiments with our test bed can be achieved by attaching the perception system to electrical motors settled in the carriage system . Neglecting the elevator effects on the surge motion, from the AUV modeling, the pitch angle transfer function and the depth rate transfer function are obtained. These motions can be implemented by controlling the Y carriage and an electrical motor to which the sensors can be attached. This idea is illustrated by the schemes in Fig. 11.

The input "δ_e" to the transfer functions is the elevator angle calculated by the AUV model control system. A similar reasoning can be applied to the yaw and sway motions in the lateral plane.

7. Experiments

In the experiments, a scenario for representing the sea bed was installed at the south side of the tank . Curved shapes for representing hills and valleys were made by water proof sheets. More steep parts of the terrain were represented through the use of wood plates covered with sinusoidal profiles made of synthetic fiber. The sinusoidal profiles were introduced for simulating the terrain roughness, and for decreasing the frequency of specular reflection of the sonar beams. Two fixed configurations of echo sounders were used in the first trials, attached to the sub-carriage. The first configuration is described by: a) forward looking echo sounder (along the west-east direction): b) downward looking echo sounder (along the north-south direction) c) upward looking echo sounder (along the south-north direction). The second configuration includes the forward and downward echo sounders as before, and the third one amidst them, at 45 degrees, aiming the southeast direction.

Unlike the computer simulations with the model of a cruising type vehicle, the experiment does not simulate the pitch motion. Thus, the angular positions of the acoustic beams are constant. A possible application of these first trials is the investigation of missions using bluff type vehicles, where the pitch angle is usually kept constant. One of the objectives of the guidance system would be to keep a continuos motion of the vehicle, avoiding frequent stops in front of obstacles.

During experiments, the test bed was running in the forward direction (west-east) with constant velocity, and the transversal velocity was changed in order to follow the terrain and avoid collision with obstacles. The reference forward velocity was set to a constant value of 0.2 m/sec. The transversal velocity has its maximum absolute value limited to 0.3 m/sec. The reference altitude from the terrain was 3 meters. The limit range for starting the free space investigation process was 5 meters, and the limit for collision avoidance was set to 3 meters.

The grid map manipulated in the *mapper* module consists of a lattice of 500 * 500 cells. Each cell has a size of 0.1 * 0.1 meters^2. Using the same cell size, the window used in the search process has 50 * 50 cells. The guidance algorithm, in the *pilot* module, runs at a sampling interval of 0.4 seconds. The motion of the carriage system is controlled at a sampling interval of 0.1 seconds.

An experiment on a kind of active perception was tried. In the third configuration, an acoustic beam with variable direction was used. The forward looking echo sounder was attached to an electric motor, that has been used for yaw maneuvers of ship models. The resulting configuration is shown in Photo. 1. The forward looking echo sounder returns to the original south-north direction, after enough free space is reported by the *mapper*, or the limit distance (3 meters) to the original forward obstacle is achieved. The limit range for starting the investigation on free spaces, in this case, was increased to 13 meters.

The first test refers to the pure terrain following guidance algorithm. The test was interrupted at an imminent collision with one of the walls of the scenario.

NII-Electronic Library Service

The Society of Naval Architects of Japan

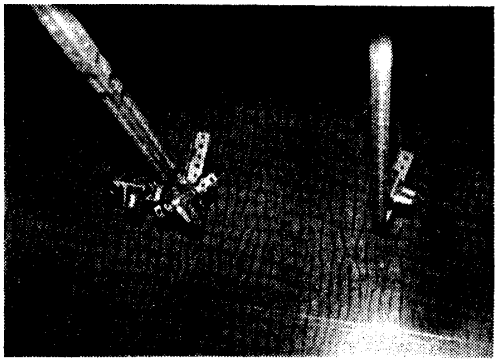

Photo 1 Third configuration of echo sounders: the forward echo sounder is attached to a rotating shaft.

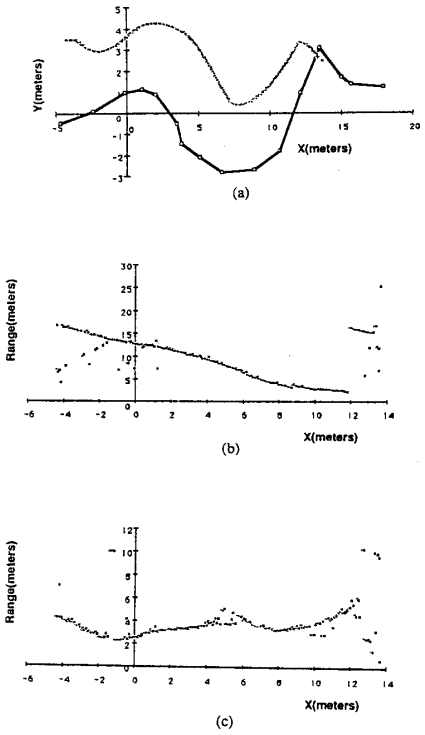

Fig. 12 Test 1. Experiment with the first configuration of sonars. Pure terrain following algorithm. (a) trajectory. (b) forward slant range variation. (c) downward slant range variation.

The trajectory estimation is represented in Fig.12 a, and the correspondent variations of the measured ranges at forward and downward directions are shown in Figs. 12 b, and 12 c respectively. The forward range graph shows the decrease on the values measured as the carriage approximates the highest hill in the scenario. Just after such hill is no more perceived by the forward echo sounder, as can be seen from the large discontinuity in the graph at 12 meters (Fig. 12 b), the guidance system starts a terrain following maneuver based on the data from the downward-looking echo sounder (Fig. 12 c), and approaches the top of the hill from above. However, at this time, the relative position between the forward echo sounder and the hill does not permit its detection.

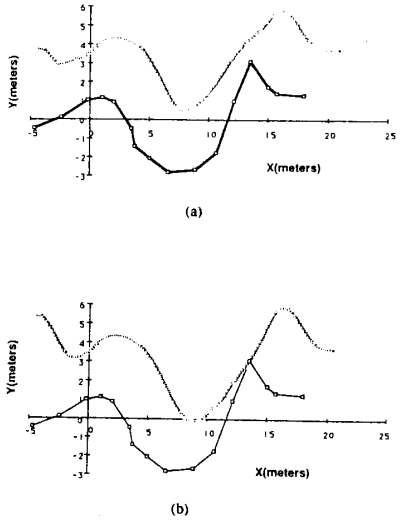

Fig. 13 Experiments using the second configuration of sonars. Pure terrain following algorithm. (a) test 2-trajectory. (b) test 3-trajectory.

When the second configuration of transponders is tried, the measured range at 45 degrees from the forward direction helps in improving the performance of the pure terrain following algorithm. The change from collision avoidance to the terrain following state was not so abrupt as in the former test. However, such change had occurred also before the carriage overcame the hill, and the carriage trajectory was close to it (Fig. 13 a). A critical situation occurs when the limit for starting the collision avoidance maneuver is decreased to 3 meters, as shown in Fig. 13 b.

In the test 4 (Fig. 14 a), the cooperative control system was tested using the first configuration of echo sounders. After reaching the deepest point inside the valley, the carriage was guided directly to a position over the hill, using information provided by the *mapper*. A safe distance from the obstacles was kept. The proposed control system was also tested using the second configuration of transponders. In the test 5 (Fig. 14 b), the minimum distance for starting the collision avoidance maneuver is 5 meters. In test 6 (Fig. 14 c), such distance was decreased to 3 meters. Both results show a better performance concerning the collision avoidance than the correspondent tests using the pure terrain following algorithm.

The decreasing values of the forward distance indicated the perception of the highest hill since long distances. Particularly, when the carriage is entering into the valley, the hill is in a frontal position to the forward looking sensor. In more general terms, despite the phenomena of specular reflection and/or the low intensity of reflected beams for small distances at large

NII-Electronic Library Service

The Society of Naval Architects of Japan

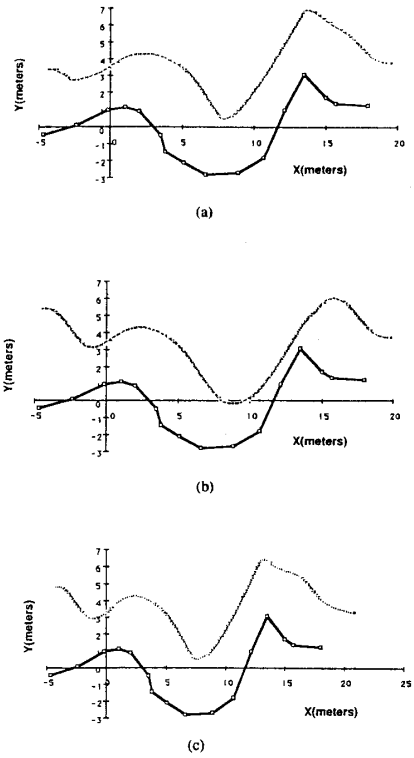

Fig. 14 "Cooperative Guidance". (a) test 4-trajectory. (b) test 5-trajectory. (c) test 6-trajectory.

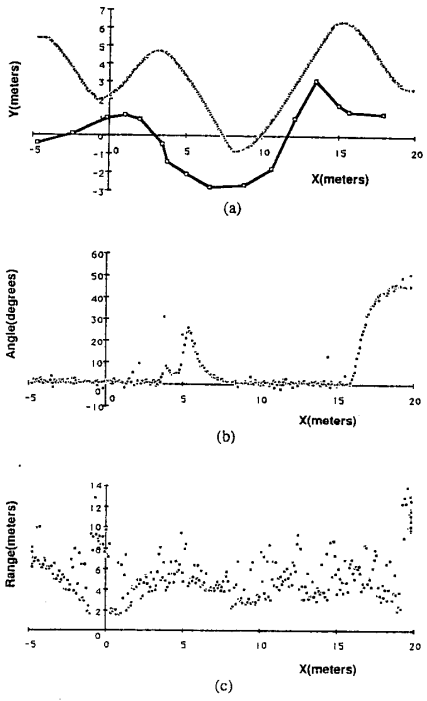

Fig. 15 Experiment using the third configuration of sonars. "Cooperative Guidance". Test 7: (a) trajectory; (b) sonar heading variation; (c) downward slant range variation.

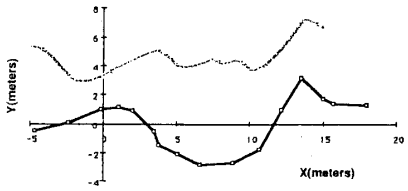

Fig. 16 Experiment using the second configuration of sonars. "Cooperative Guidance". Test 8-trajectory. The limit slant range for starting the "survey" maneuver was increased to 13 meters.

incidence angles, there is a great possibility for the obstacle detection during its approach, since different distances and incidence angles occur. However, such information is not taken into account by the pure terrain following algorithm. Map building and obstacle avoidance are not independent process in the cooperative guidance system. Taking into account forward obstacles by the guidance commands at the survey state, helps to build a more accurate representation of the terrain in the tests.

Test 7 (Fig. 15) refers to the guidance system with variable beam direction. The quite discontinuous perception of the terrain profile is due to the high noise coming from the motor driver. This phenomenon caused large variations in the readings from the sonar system, specially in the case of the downward direction, as can be observed in Fig.15 c. It is possible to observe the variation of the beam direction and the actual sonar headings during the carriage displacement. As the carriage enters into the valley, the search for free spaces is activated and finishes as soon as enough free spaces are found (Figs. 15 a and 15 b).

Test 8 was carried out for a comparison with test 7 (Fig. 16). The original system , with fixed beam directions relative to the vehicle, was used. It was imposed the same distance as test 7 for activating the free space searching process. With a larger limit distance used than before, the search for free spaces does not allow the vehicle to dive into the valley.

8. Conclusions

A guidance system for real-time terrain following and collision avoidance applied to an AUV was presented. This control system is based on discrete events related to the detection of forward obstacles by a sonar system. A dynamic interaction with a representation of the external environment around the vehicle is employed, since these discrete events may trigger a kind of "dialog" between a guidance module, the "pilot", and a module responsible for building a grid map, the "mapper".

As a background for the guidance system, a software control architecture using parallel distributed processing was proposed. Control units are represented by a

NII-Electronic Library Service

The Society of Naval Architects of Japan

group of entities, the "agents", which are based on a computational framework formed by a combination of the *Actors* model and the *ABCM/1*. A specialization of such models concerning the internal states of agents and communication among them forms the artificial intelligence framework, the *Agent Oriented Programming* model. Agent orientation provides means for expressing more complex relationships among control units than commom behavioral based architectures.

Preliminary tests with the collision avoidance system were based on simulations using a mathematical model of the AUV motion in the vertical plane. This model was determined through system identification experiments with the real scale vehicle. Futher tests were performed using a novel test bed, which is based on the adaptation of model basin facilities. The interpretaion of the real world from the sensory system, and the correspondent action in real-time could be investigated. Therefore, we claim that an important experimental phase for developing underwater robots can be executed, avoiding the costs related to the implementation of actual vehicles.

Future work with the cooperative control system includes more case studies investigating the relationship among the control units. For the collision avoidance system, an extension to the lateral plane is intended. Finally, futher improvements of the proposed test bed related to the representation of the vehicle dynamics and the external environment are necessary.

9. Acknowledgements

The first author is grateful to Dr. H. Yamato, who first proposed the use of a model reference controller for simulating ROV motions using the carriage system, Mr. F. Suzuki, for his help in the experiments, and Miss S. Suzuki, for helping in the preparation of this text.

References

1) de Barros, E. A.: A cooperative control system and its application to the collision avoidance guidance of autonomous underwater vehicles. Doctor thesis. University of Tokyo, 1994.
2) Malcolm, C. and Smithers, T.: Symbol grounding via a hybrid architecture in an autonomous assembly system. In Designing Autonomous Agents, edited by Pattie Maes, MIT/Elsevier, p. 123-144, North-Holland, 1990.
3) Brooks, R. A.: Intelligence without representation. Artificial Intelligence, 47, pp. 139-159, 1991.
4) Hall, W. D., Adams, M. B.: Autonomous vehicle software taxonomy. Proc. of the IEEE Symposium on Autonomous Underwater Vehicle Technlogy, pp. 49-64, 1992.
5) Hewitt, C.: Viewing Control Structures as Patterns of Passing Messages, Artificial Intelligence Journal, vol. 8, pp. 323-364, 1977.
6) Agha, G.: Actors: a model of concurrent computation in distributed systems, M. I. T. Press, Cambridge, MA., 1986.
7) Yonezawa, A., Briot, J. P., Shibayama, E.: Object-oriented concurrent programming in ABCL/1. In Proc. of ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), pp. 258-268, 1986.
8) Yonezawa, A. and Tokoro, M. edited.: Object-oriented concurrent programming, MIT Press, Cambridge, 1989.
9) Hoare, C. A. R.: Communicating Sequential Processes, Prentice Hall, 1985.
10) Ura, T.: Development of AUV PTEROA. The 1 st Workshop on Mobile Robots for Subsea Environments. International Advanced Robotics Programme, pp. 195-200, 1990.
11) Kato, N., Y. Ito, J. Kojima, S. Takagi, K. Asakawa, Y. Shirasaki: Control performance of autonomous underwater vehicle "Aqua Explorer 1000" for inspection of underwater cables. Proc. of "Oceans 94". vol. I, pp. 135-140.
12) Ishitani, H., Baba, Y. and Ura, T.: Altitude control system of a streamlined cruising type autonomous submersible, Proc. of the 6 th Int. Symposium on Unmanned Untethered Submersible Technology, pp. 452-457, 1989.
13) de Barros, E. A., Maeda, H., Yamato, H. and Miyajima, S.: Local path planning for and AUV using a cooperative control system, OCEANS '93, Proc. Vol. 1, 175-180, 1993.
14) Papoulias, F. H.: Dynamics and bifurcations of pursuit guidance for vehicle path keeping in the dive plane. Journal of Ship Research, Vol. 37, No. 2, p. 148-165, June 1992.
15) Elfes, A.: Occupancy grids: a probabilistic framework for robot perception and navigation., Ph. D. thesis. Carnegie- Mellon University, U. S. A., 1989.
16) Shoham, Y.: Agent Oriented Programming. Technical Report STAN-CS-90-1335, Computer Science Department, Stanford University, 1990.
17) de Barros, E. A., Suzuki, F., Miyajima, S., Yamato, H., Maeda, H.: A novel testbed for experiments on underwater robotics. Proc. of the Spring Conference of the Kansai Society of Naval Architecture, 87-92, 1994.
18) Concurrent Systems. TCS version for PC-9801, Ver. 3.13. User's Guide, 1992.
19) Maeda, H., Tatsuta, S.: Prediction Method of Hydrodynamic Stability Derivatives of an Autonomous Non-Tethered Submerged Vehicle. Proc. of the Eighth Int. Conf. on Offshore Mechanics and Artic Engineering, vol. VI, 1989.
20) de Barros, E. A., Maeda, H. and Miyajima, S.: A system identification experiment for an underwater vehicle, Proc. of the 11 th Ocean Engineering Symposium. Society of Naval Architects of Japan, pp. 255-262, 1992.

NII-Electronic Library Service