

A Hybrid Control Architecture for Autonomous Robotic Fish

Jindong Liu, Huosheng Hu, Dongbing Gu

Department of Computer Science, University of Essex, Colchester CO4 3SQ, U.K.

Email: jliua@essex.ac.uk, hhu@essex.ac.uk, dgu@essex.ac.uk

Abstract—This paper presents a hybrid control architecture for autonomous robotic fishes which are able to swim and navigate in unknown or dynamically changing environments. It has a three-layer configuration: cognitive layer, behaviour layer and swim pattern layer. The state-based planning in the cognitive layer provides a good foundation for potential adaptation through machine learning methods such as reinforcement learning(RL). The behaviour layer and the swim pattern layer are specially designed to match the needs for the real-time control of our robotic fish. To test the feasibility and performance of the proposed architecture, the experiment of “tank border exploration” is conducted with Q-learning.

Index Terms—Robotic fish, Robot behaviors, Control Architecture

I. INTRODUCTION

The astonishing swimming abilities of fish has inspired many researchers to work on a new kind of aquatic man-made robotic systems, namely *Robotic Fish*. Up to now, majority of research work has been focused on the fish-like propulsion mechanism [1][2], the fin material [3], remote operation[4] and multi-agent cooperation[5] and the mechanical structures [6][7][8]. Little research work has been done on building intelligent control architectures for robotic fish that can swim and navigate in unknown and dynamic environments.

It is well known that control architectures used in mobile robots can be classified to three categories: Deliberative Control (DC), Reactive Control (RC) and Hybrid Control (HC). A mobile robot with DC architecture has advantages of high-level intelligence and predictive capabilities, but suffers a low speed of response and requires complete knowledge about the world before its operation [9]. On the other hand, RC architecture has no needs for building a complete model about the world and no needs for reasoning, which is also called behaviour-based architecture. Each behaviour only focus on a fast connection from perception to action, and several behaviours are coordinated to generate emergent behaviours to realize the global task. Several kinds of RC architectures were developed including subsumption architecture [10] and schema model [11]. However, RC architecture can not guarantee an optimal solution since it lacks of the planning capability. It is also unable to use of the external world models when they are available [12]. Although the emergent behaviours can be argued for the robotics ability to generate complex behaviours, RC architecture is unable to make this a reality since it lacks mechanisms for managing complexity [13].

To take advantage of both DC architecture and RC architecture, many researchers[14][15] proposed HC architecture

which possesses the planning ability of DC architecture to implement goal-oriented tasks and the fast response ability of RC architecture to deal with the uncertainty and dynamics in the real world. They focus on the integration of the low-level behaviours and high-level planning.

In this paper, a hybrid control architecture is proposed for our robotic fish, which consists of three layers: a cognitive layer, a behaviour layer and a swim pattern layer. Within this hybrid architecture, parameter optimization can be conducted separately at each layer. For instance, in the swim pattern layer, optimization is conducted to achieve best performance of each swim pattern. In the behaviour layer, behaviours are optimized to output proper swim patterns responding to sensor data. In the cognitive layer, optimization also could be designed for the parameters of reasoning and planning tasks.

The rest of this paper is organized as follows. Section II presents the structure of the proposed hybrid control architecture, the content of each layer and the relationship between them. Section III addresses an example of optimization on one of layers: cognitive layer. In Section IV, experimental results are presented to show the feasibility and performance of optimization under the proposed hybrid control architecture. Finally, conclusions and future work are given in Section V.

II. CONTROL ARCHITECTURE DESIGN

The proposed hybrid control architecture is presented in Fig. 1, which consists of three layers: *Cognitive Layer*, *Behaviour Layer* and *Swim Pattern Layer*. The centre of the architecture is the behaviour layer where most instant decision is made. The swim pattern layer is particularly designed for the robotic fish because of its special propulsion feature. In the behaviour layer, states and actions are directly related to the robot physical sensors and swim patterns. In the cognitive layer, states related with high-level activities are abstracted from the behaviour layer and actions are correspond to behaviour coordination which directs the configuration of several behaviours. Learning or optimization can be applied separately in each layer from bottom to top, i.e. once the optimization is finished in the swim pattern layer the behaviour layer can start to be optimized according to the optimal swim patterns. Similarly, optimization in the cognitive layer will employ the optimized behaviour in the behaviour layer.

A. Swim Pattern Layer

swim patterns are the basic motion descriptions of our robotic fish. Each of them is a series of joint motion functions(Equation (1)) which control the joints on the fish tail

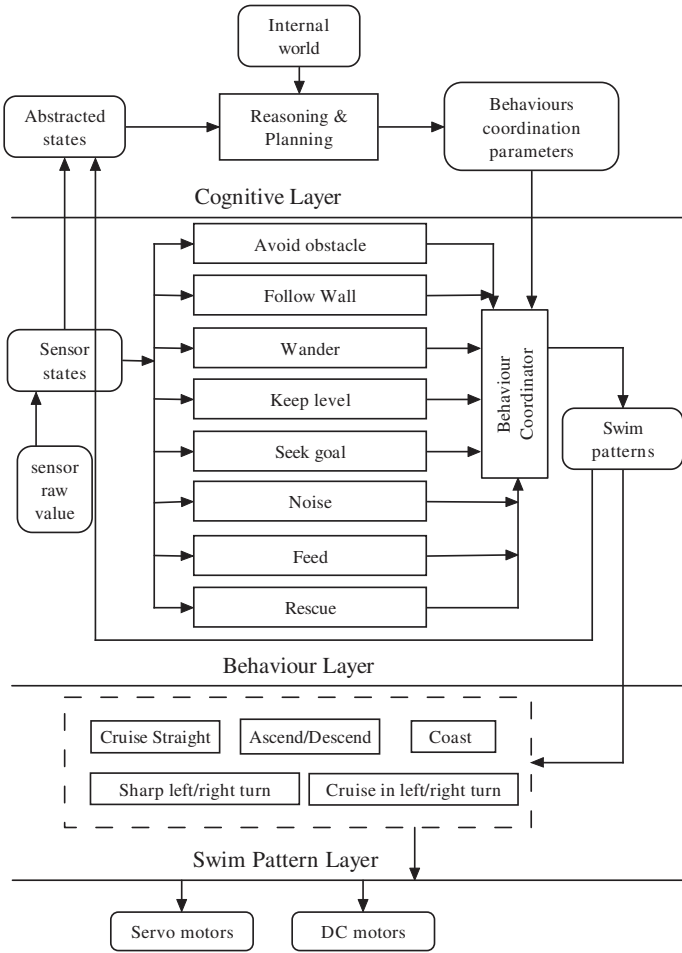


Fig. 1. The hybrid control architecture for our robotic fish

to realize an associative movement in order to propel or maneuver the robotic fish. The swim pattern layer is at the bottom of our three-layer architecture, which consists of all swim patterns for basic control purpose. Five swim patterns are designed here, namely *Cruise-Straight*, *Ascend/Descend*, *Coast*, *Sharp-Left/Right-Turn* and *Cruise-in-Left/Right-Turn*. Each swim pattern corresponds to a series of “time-position” functions of joint motions:

$$p_i = f_i(t), i = 1 \dots k, t \in [0, T_i] \quad (1)$$

where k is the number of joints and T_i is the duration time of the swim pattern. The position p_i means the turning angle of the i th joint relative to its centre position.

When the behaviour layer has decided a swim pattern, the robotic fish will implement it by setting $t = 0$ and then actuate the joint in functions (1) until $t = T_i$ or a new swim pattern is received. Except for swim patterns of ascend/descend, all swim patterns are mutually exclusive, i.e. only one swim pattern can be executed at the same time. Those swim patterns are called “*level-swim patterns*” because they are limited in a 2D plane, while the ascend/descend pattern is called “*vertical-swim patterns*”. A rule for swim pattern selection is that only

one swim pattern can be selected from level swim patterns and so does from vertical swim patterns.

B. Behaviour Layer

The behaviour layer locates in the middle of the proposed hybrid control architecture. It is responsible for the reactive control of our robotic fish. It firstly converts the sensor raw data into sensory states by fuzzy membership functions and then input these states into each behaviour. Fuzzy logic controllers are designed for individual behaviours to process these states. The responses of all behaviours are coordinated together by the behaviour coordination function which is defined by the parameters from the output of the cognitive layer. The subsumption architecture is adopted for this layer because of its flexibility. Eight behaviours are designed for the generic purpose in this layer, namely *Avoid-Obstacle*, *Follow-Wall*, *Wander*, *Keep-Level*, *Seek-Goal*, *Feed*, *Rescue* and *Noise*. Stimulus-response (SR)[11] diagrams are used for graphic representations of specific behavioural configurations. Here, the stimulus is the sensor states and the response is the action.

A brief description of each behaviour is listed as follows.

- *Avoid-Obstacles*: This behaviour is to avoid both stationary and moving objects based on IR sensors. The possible action is selected from all swim patterns.
- *Follow-wall*: This behaviour is to keep the robotic fish at a certain distance from the wall based on the information gathered by IR sensors. The possible actions are *cruise-straight* and *cruise-in-left/right-turn*.
- *Wander*: This behaviour is to explore the tank randomly. No sensor states is necessary. Possible actions include *cruise-straight*, *cruise-in-left/right-turn* and *ascend/descend*.
- *Keep-level*: This behaviour is to keep the robotic fish swimming at a desired depth level specified by the cognitive layer. It monitors the pressure sensor data and control the robotic fish to ascent or descent.
- *Seek-goal*: This behaviour is to direct the robotic fish to swim toward the goal, and the possible actions are *cruise-straight* and *cruise-in-right-turn*.
- *Noise*: This behaviour uses a random swim pattern in order to avoid the local minima, and go out from a trap situation by trying all kinds of swim patterns.
- *Feed*: This behaviour is for a robotic fish to return to the charging station when its battery voltage is low.
- *Rescue*: This behaviour is to do emergent self-protection action when robotic fishes face uncontrollable situations such as servo motors malfunction, the extreme high internal temperature, etc.

Assume that a behaviour can be denoted as a triple (s, a, β) where s denotes the domain of all sensor states, a denotes the range of possible responses, and β denotes the mapping $\beta : s \rightarrow a$. The behavioural coordination function C for multiple behaviours is defined as:

$$\rho = C(G * B(S)) = C(G * A) \quad (2)$$

where:

$\mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_n]'$, $\mathbf{S} = [s_1 \ s_2 \ \cdots \ s_n]'$,
 $\mathbf{G} = [g_1 \ g_2 \ \cdots \ g_n]'$ and $\mathbf{B} = [\beta_1 \ \beta_2 \ \cdots \ \beta_n]'$
and “*” denotes the special scaling operation for multiplication of each scalar component (g_i) by the corresponding magnitude of the component vector \mathbf{a}_i , resulting in a column vector \mathbf{a}'_i of the same dimension as \mathbf{a}_i [11].

The behaviour layer not only decides which swim pattern will be next selected but also adjust some key parameters for the selected swim pattern. Each swim pattern has tens of parameters in function (1) to specify its performance. However, to simplify the control loop, the behaviours layer only adjusts two parameters: (i) speedcode s for level-swim patterns and (ii) pitchTarget P_t for vertical-swim patterns. Note that s has different meanings in different level-swim patterns, e.g. s is proportional to the linear speed of *cruise-straight* swim pattern, while it indicates the turning speed in the *cruise-in-turn*. For the vertical swim pattern, P_t means the position of the internal pitch weight relative to its centre position. The pitch weight affects *ascend/descend* by changing the centre of gravity(COG) of the robotic fish. It is controlled by a DC motor.

C. Cognitive Layer

The top layer is the cognitive layer. It extracts robotic fish status from the sensor states in the behaviour layer, and then makes task-oriented reasoning and planning. Finally it makes decisions on how to coordinate behaviours in the behaviour layer to achieve a given task. Different from the traditional DC architecture, the cognitive layer does not provide the detailed objective for the behaviour layer, e.g. a particular trajectory for the robotic fish to follow. It only focused on the global performance of emergent behaviours. By changing the coordination parameters, the cognitive layer adjusts the contribution weight of each behaviour for emergent behaviours. Besides, the cognitive layer will also change the parameter for some individual behaviours, such as the depth level for *keep-level* behaviours.

In the cognitive layer, planning can be realized in several ways. In a known environment with available models, planning can be done offline. Solutions can be found and evaluated prior to execution. However, in dynamically changing environments, models and policies need to be adaptively revised online. “Trial and Error” processes are normally carried out. See Section III for details.

The control loop in this hybrid control architecture can be summarized as following:

- 1) The raw sensor data is converted into sensor states in the behaviour layer. Individual behaviours take sensor states as stimulus and calculates responses concurrently.
- 2) The cognitive layer extracts the abstracted states from the sensor states. The parameters for behaviours coordination are transferred into the behaviour layer after reasoning and planning.
- 3) The responses of all behaviours are combined by behaviours coordination. It determines the swim patterns

to be executed and the parameter s and P_t for them.

- 4) The selected swim pattern is configured in the swim pattern layer by using s , P_t and parameters stored in this layer.
- 5) Servo motors are driven according to function (1) which is corresponding to the selected swim pattern to implement the *level-swim pattern* and the *vertical-swim pattern* is realized through a DC motor.

III. STATE-BASED PLANNING IN THE COGNITIVE LAYER

A typical reinforcement learning (RL) based planner for mobile robots is normally built up by three parts: *Action Space*- a set of possible actions, *State Space*- the discrete possible situations of a robot on the way from initial time to a goal, and *Markov Decision Process(MDP) Model*- a relationship description between the action space and the state space. The planner produces a sequence of actions that lead a robot from the initial state to a state meeting the goal[16][17].

A. Action Space and State Space

The cognitive layer aims at organizing actions to accomplish a task. The action could be turning on a software switch to execute a behaviour or just setting a configuration parameter of a behaviour. Actions in the cognitive layer are denoted as ca . To simplify the computation, the action space is divided into two independent subspaces: level-plane actions and depth-control actions. The domain of level-plane actions (la_i) is related with all the behaviours which affect the movement in all 2D planes parallelling water surface such as *follow-wall*, while depth-control actions va_i will change the parameters of the behaviours which control the swimming depth such as *keep-level*. Note that, *avoid-obstacle* behaviour is not listed as one of actions in the cognitive layer because it is a low-level behaviour in the behaviour layer. To guarantee the fast response when approaching obstacles, the cognitive layer will not change its property.

States in the cognitive layer are extracted from the sensor states and the swim patterns previously executed in the behaviour layer. They are the high-level condition or mode of our robotic fish. States come from the sensor data but they don't represent the quantity of individual sensors. However, states reflect the significant physical event which is sensed or recognised by a temporal and spatial combination of several sensors. For example, when the robotic fish reaches the bottom of the tank, its down-facing infrared detector outputs a higher value to indicate that the fish is near the bottom of the tank. A set of these kinds of events constitutes the state space finally. Furthermore, the event can also be the previous swim pattern generated in the behaviour layer.

Formally, each event is denoted by a binary variant bv . Once an event occurs, the relative bv is set to 1, otherwise it is clear to 0. Grouping n events (bv_1, \dots, bv_n) in an order generates a state cs for the cognitive layer, i.e. $cs \leftarrow bv_1, \dots, bv_n$. The state space consists of a 2^n combination of n events. To decrease the size of the state space, these events are divided into two independent subspaces: level-states cs_l which only

have relationship with the level actions, and vertical-states cs_v , which are connected to the depth control actions.

B. Markov Decision Processes Model

For the robotic fish application, the RL based planner in the cognitive layer can be described by two finite MDPs: level-MDP Γ_l and vertical-MDP Γ_v . The former is a model for level-states and level-actions while the latter is for vertical-states and depth-control actions. Assume that Γ_l is defined by level-states cs_l , level-actions la and the one-step dynamics of the environment. The state transitions are described by transition probabilities:

$$P_l = Pr\{cs_{l(t+1)}|cs_{l(t)}, la_{(t)}\} \quad (3)$$

The expected value of the next reward given current state and action, $cs_{l(t)}$ and $la_{(t)}$, together with next state, $cs_{l(t+1)}$ is expressed as:

$$R_l = E\{r_{t+1}|cs_{l(t)}, la_{(t)}, cs_{l(t+1)}\} \quad (4)$$

A policy π is a mapping from states to actions. The optimal policy π^* maximizes the probability of reaching the goal. The *value* of a state s under a policy π , denoted $V^\pi(s)$, is the expected return when starting in s and following π thereafter. Function V^π is called *state-value function for policy π* . So, for Γ_l , we define $V_l^\pi(cs_l)$ formally as:

$$V_l^\pi(cs_l) = E_\pi(R_t | cs_l) = E_\pi\left(\frac{1}{T} \sum_{k=0}^{T-1} r_{t+k+1} | cs_l\right) \quad (5)$$

where T is the number of samples, $E_\pi\{\}$ represents the expected value given that the robotic fish follows policy π . At the same time, the value of taking level action la in level-state cs_l is defined under policy π as $Q_l^\pi(cs_l, la)$:

$$Q_l^\pi(cs_l, la) = E_\pi(R_t | cs_l, la) = E_\pi\left(\frac{1}{T} \sum_{k=0}^{T-1} r_{t+k+1} | cs_l, la\right) \quad (6)$$

For Γ_v , there are similar definitions of P_v , R_v , $V_v^\pi(cs_v)$ and $Q_v^\pi(cs_v, va)$. The optimal policy π^* is defined to the one to maximize $\lim_{T \rightarrow \infty} V_i^\pi$. Given the P_l and R_l for the all level states cs_l and level actions la , a full description of Γ_l is drawn up. It can be found analytically by using *Dynamic Programming* which recursively calculates V_l^π and Q_l^π first. If P_l is unknown, modelling techniques can be used to find it by model-based RL. Alternatively, π^* can be found directly based on R_l through model-free RL, such as *Monte Carlo method* or *Temporal-Difference Learning (TD Learning)*. In Section IV, *Q-learning*, an off-policy TD learning, is designed in our experiment to prove the learning ability of our systems.

IV. IMPLEMENTATION AND EXPERIMENT RESULTS

To test the feasibility and performance of the proposed architecture, a tank boarder exploration task is implemented by our robotic fish. The task is to make the robotic fish autonomously explore the tank border. It should follow tank walls in an appointed distance, and be able to avoid the corner

and other fishes, and keep itself in a desired depth level. Besides, it must keep the wall at its right side. The robotic fish is supposed to know nothing about its environment. It will learn to explore the tank border from scratch by Q-learning.

A. Robotic Fish and Its Environment

The fish tank is 5.5 meters long, 1.7 meters wide and 1.6 meters deep. The robotic fish is about 50 cms in length, 20 cms in height and 12 cms in width. Two coordinate systems are defined in Fig. 2. The global coordinate system (xyz) origins at the front-bottom-left corner of the tank and takes the tank bottom as ($z = 0$) plane. The local coordinate system ($x_1y_1z_1$) is origins at the connection point between the head and the tail of the robotic fish. If the robotic fish is viewed as a rigid body, it has 6 degree of freedoms (DOFs) including 3 translation DOFs and 3 rotation DOFs on x_1 , y_1 and z_1 . The controllable DOFs include the translation movement on x_1 , the pitch angle, the yaw angle, and the translation movement on z .

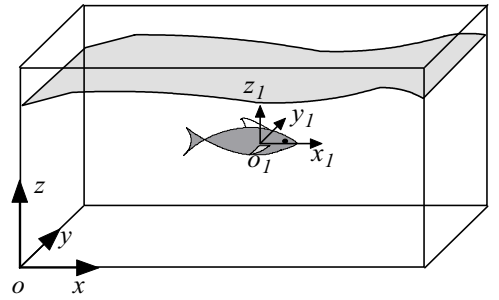


Fig. 2. The global and the local coordinate systems

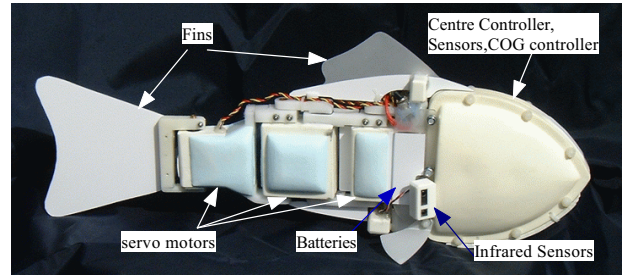


Fig. 3. G9 robotic fish profile

The robotic fish is equipped with several kinds of sensors to response to the dynamical changes in its environment, its position in the tank, the robot attitude and the internal status (e.g. the battery voltage). A standard configuration of the robotic fish includes four infrared sensors, one dual-axis accelerometer/inclinometer, one piezoelectric vibrating gyroscope, one water pressure sensor, three electric current sensors and three servo turning angle sensors. It is able to sense obstacles around it within a range of 40cm and its absolute z position in the xyz frame. It also can perceive the pitch/roll angle, the 1-order derivative of the yaw angle, the turning angle of the tail joints and the power consumption on them. However, the robotic fish has no ability to localize itself in the

xy plane because it has no sensor to measure its linear speed on x_1 . So it can not localize itself by the way of odometer used in the common mobile robots. Also, our robotic fish swims in 3D and can't rely on an overhead vision location system because wireless signals can not pass through a 20cm depth of water. Fig. 3 presents our robotic fish used in this research.

B. Action/States Space and Q-learning

The action space is structured according to the objective of the task. Four behaviours are chosen and customized from the generic behaviours layer in Section II for the level plane action subspace.

- *Wander* ($la_1 = WD$): For the specific task, this behaviour is limited on a 2D plane rather than in 3D space. The robotic fish randomly selects one of swim patterns from *cruise-straight* and *cruise-in-left/right-turning* to execute.
- *Find-wall* ($la_2 = FDW$): It is customized based on the *seek-goal* behaviour. The tank wall is defined as the goal. Once this behaviour is executed, the robotic fish will *cruise-in-right-turn* or *cruise-straight* until it finds the wall again.
- *Follow-wall* ($la_3 = FLW$): This behaviour inherits from the *follow-wall* behaviour in the generic behaviour layer. The wall is appointed on the right of the robotic fish.
- *Noise* ($la_4 = NS$): Same as the definition of *noise* behaviour in the generic behaviour layer except that it is limited in a 2D plane for the task.

Note that the *avoid-obstacle* behaviour has priority in the behaviour layer and it is limited in a 2D plane for the task. Three events are defined to create the level states(cs_l):

- Is the wall on the right side of the robotic fish? (bv_1): It is decided by the right, front and left infrared sensors. $bv_1 = 1$ if recent history of these sensors satisfied some conditions.
- Is the robotic fish in a reasonable range from wall? (bv_2): This event is similar to bv_1 but it has more strict conditions.
- Is the wall on the left side of robotic fish? (bv_3): Like bv_1 , it recognizes the situation that the nearest wall is on left side of fish. In other words, the swimming direction is reversed to the desired direction.

Because $bv_1 = 1$ and $bv_3 = 1$ are mutually exclusive, and so are $bv_1 = 0$ and $bv_2 = 1$, the total number of states is decreased from $8(2^3)$ to 4 as shown in Table I

state cs_l	bv_1	bv_2	bv_3
0	0	N/A	0
1	0	N/A	1
2	1	0	N/A
3	1	1	N/A

TABLE I
THE STATE MADE BY EVENTS

There is no vertical action subspace in the cognitive layer for this task because the *keep-level* behaviour is enough for the

task. So it has not vertical states too. The *one-step Q-learning* updated function is as follows[18]:

$$Q(s, a) = Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \quad (7)$$

where Q represents Q_l^π , s donates cs_l and a donates la . α and γ are learning ratio. r is the observed reward when take action a . In this task, $r = 1$ when *find-wall* behaviour observed a wall, $r = -1$ when *follow-wall* behaviour lost the wall to follow, $r = -3$ when there is a bumping between the wall and the robotic fish and $r = 0$ for other situations. s' is the succeeded state after taking action a . ϵ -greedy method is taken for choosing a from s using policy derived from Q .

C. Experimental Result and Analysis

According to (7), policy π^* of the task is learned in a 3D robotic fish simulator [19] and then applied to the real robotic fish. Fig. 4 presents fish swimming trajectories during learning. It is clear that the robotic fish is able to keep itself in a proper distance away from the wall after learning. Fig. 5 gives the history of root mean square (RMS) errors between the learning trajectory and the desired path over 300 trips. RMS is decreased and converged along the learning time.

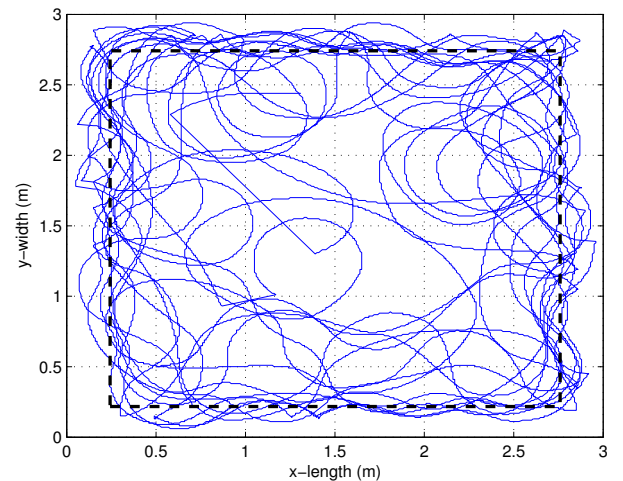


Fig. 4. The fish trajectory during learning. Note that the dashed line is the desired path, $\alpha = 0.5$ $\gamma = 0.3$, $\epsilon = 0.01$

Fig. 6 shows the trajectory of our robotic fish operated in the real world, which is recorded from an overhead camera. The time step between two record points is 3 seconds. After our G9 robotic fish is put into water from the point S , it selects the *find-wall* behaviour and executes it until reaching the left wall. Then the *follow-wall* behaviour is triggered. During its swimming, our robotic fish encounters an obstacle. It implements a *sharp-left-turning* toward the point B_1 to avoid it. Then it finds the wall again by the *cruise-in-turning-right* swim pattern.

The arrows in Fig. 6 show the heading directions at these points and the circle points indicate the COG position of robotic fish. Note that the left, right and back walls of our

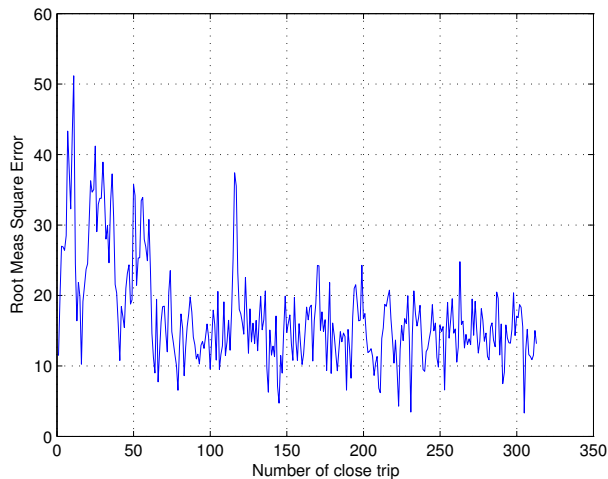


Fig. 5. The root mean square error between the learning trajectory and the desired path against the number of learning trips. The error is the sum of two parts: position error and swimming heading error. Each trip is 50 steps in Q-learning

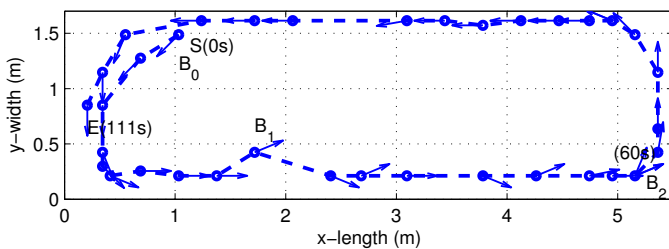


Fig. 6. A circular trajectory in $x - y$ plane

fish tank are concrete and its front wall is transparent glass located at $y = 0$. When G9 fish operated under the control of the *follow-wall* behaviour, it bumped into the front glass wall more often than it did to other walls because the infrared light can not be reflected well from the glass wall. In other words, the infrared light got through the glass wall rather than came back, G9 fish thought that it lost the wall. Therefore, it decided to select *cruise-in-right-turning* swim pattern to find the wall, which of cause caused the collision problem.

Finally, G9 robotic fish spent 105 seconds to swim around the tank one circle and finally reached the point E .

V. CONCLUSION AND FUTURE WORK

The hybrid control architecture proposed in this paper aims to realize fully autonomous control of robotic fish in the real world. The behaviour layer is located at the centre of the whole architecture. Its subsumption structure makes the robotic fish have quick response to the dynamic environment. The cognitive layer helps a robotic fish to coordinate its behaviours under the goal-oriented way. The state-based planning gives a good foundation for task planning and potential future investigation of RL strategies. Considering the characteristics of robotic fish motion, the swim pattern layer is introduced in order to free the behaviour layer from the expensive calculation of

low-level motion control parameters. The experimental results show the feasibility and good performance of the proposed hybrid control architecture. Our robotic fish has been in the public show in London Aquarium since its successful launch on 6 October 2005, and attracted thousands of worldwide visitors.

Our future work will be focused on behaviours learning and parameter optimization. The layer configuration of the proposed control architecture guarantees the independent learning and optimization can be operated asynchronously.

ACKNOWLEDGMENT

This research is funded by London Aquarium. Our thanks also go to Ian Dukes and George Francis at Essex for their contribution toward the project.

REFERENCES

- [1] J. M. Anderson and P. A. Kerrebrock. The Vorticity Control Unmanned Undersea Vehicle (VCUV): An Autonomous Robot Tuna. In *10th International Symposium on Unmanned Untethered Submersible Technology*. Durham, NH, 1997, pp. 63–70.
- [2] K. Naom. Control performance in the horizontal plane of a fish robot with mechanical pectoral fins. *IEEE Journal of Oceanic Engineering*, vol. 25(1), 2000:pp. 121–129.
- [3] J. Witting, K. Safak, and G. Adams. SMA Actuators Applied To Biomimetic Underwater Robots. In J. Ayers, J. Davis, and A. Rudolph, eds., *Neurotechnology for Biomimetic Robots*. MIT Press, 2001, pp. 117–136.
- [4] J. Liang. Researchful Development of Underwater Robofish II- Development of a Small Experimental Robofish. *Robot*, vol. 24(3), 2002.
- [5] J. Yu, S. Wang, and M. Tan. A simplified Propulsive Model of Biomimetic Robot Fish and Its Realization. *J. of Robotica*, vol. 23, 2005:pp. 101–107.
- [6] S. Guo, T. Fukuda, N. KATO, and K. OGURO. Development of Underwater Microprobe Using IMPF Actuator. In *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, Leuven, Belgium, May 1998, pp. 1829–1834.
- [7] <http://www.nmri.go.jp/eng/khirata/fish/>, Sep 2000.
- [8] http://www.mhi.co.jp/enews/e_0898.html, Sep 2000.
- [9] J. Albus, H. McCain, and R. Lumia. NASA/NBS Standard Reference Model for Telerobot Control System Architecture(NASREM). Tech. rep., NBS Technical Note 1235, Robot Systems Division, National Bureau of Standards, 1987.
- [10] R. Brooks. A robust Layered Control System for a Mobile Robot. *IEEE J. Robotics and Automation*, vol. 2(1), 1986:pp. 14–23.
- [11] R. C. Arkin. *Behavior-based robotics*. MIT Press, 1998.
- [12] D. Kirsh. Today the Earwig, Tomorrow man. *Artificial Intelligence*, vol. 47, 1991:pp. 161–184.
- [13] R. H. F. Pipitone. Experiments with the Subsumption Architecture. In *Proceedings of IEEE International Conference on Robotics and Automation*. Sacramento, CA, USA, April 1991, pp. 1652–1658.
- [14] E. Gat. *On Three-layer Architecture*. MIT/AAAI Press, 1997.
- [15] R. J. Firby. *Adaptive Execution on Complex Dynamic Worlds*. Ph.D. thesis, Yale University, 1989.
- [16] M. Huber. *A hybrid Architecture for Adaptive Robot Control*. Ph.D. thesis, University of Massachusetts, 2000.
- [17] D. Gu. *Behaviour-based Learning and Fuzzy Control of Autonomous Quadruped Robots*. Ph.D. thesis, University of Essex, Aug 2003.
- [18] C. J. C. H. Watkins. *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University, 1989.
- [19] J. Liu and H. Hu. Building a 3D Simulator for Autonomous Navigation of Robotic Fishes. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai, Japan, Oct 2004, pp. 613–618.