Development of a Visually-Guided Autonomous Underwater Vehicle

David Wettergreen, Chris Gaskett, and Alex Zelinsky
Robotic Systems Laboratory
Department of Systems Engineering, RSISE
The Australian National University
Canberra, ACT 0200 Australia
[dsw | cg | alex]@syseng.anu.edu.au

***Abstract* — We are developing autonomous underwater vehicles for exploration and inspection tasks. Our objectives are to enable these submersible robots to autonomously search in regular patterns, follow along fixed natural and artificial features, and swim after dynamic targets. We have developed a method of visually-guiding autonomous land vehicles using correlation-based feature tracking to follow targets of interest. We have used this method to fixate on visual features and generate motion commands for the robot. We propose to use feedback from the motion of the visual feature as reinforcement in a network that learns stable control. We are now applying these techniques to the control of our underwater vehicle, Kambara.**

I. Introduction

Australia has an extensive coastline and near-shore waters that contain vast biological and mineralogical resources. These areas are largely unexplored and unknown. Soon they must be investigated and understood so that they can be wisely developed and properly protected.

At the Australian National University we are embarking on the development of autonomous underwater vehicles (AUVs) for tasks in exploration and inspection. Our objectives are to enable these submersible robots to autonomously search in regular patterns, follow along fixed natural and artificial features, and swim after dynamic targets. These capabilities are essential to tasks like cataloging reefs, exploring geologic features, and studying marine creatures, as well as inspecting pipes and cables and assisting divers.

In collaboration with the University of Sydney, we have constructed a pair of underwater vehicles. They are mechanically identical but each is equipped with different computing and sensing components. The University of Sydney vehicle carries high-resolution sonar sensors and research focuses on using sonar for simultaneous mapping and localization.[1] Our research focuses on visually-guided navigation, so our vehicle has cameras and image processing hardware. We have also designed a battery system and removable data tether. In the future, we intend to bring the robots together for cooperative tasks and for multi-modal site exploration.

Our AUV is named Kambara, an Australian Aboriginal word for crocodile. Kambara has a frame supporting thrusters and watertight enclosures. Its thrusters enable roll, pitch, yaw, heave, and surge maneuvers. The system has a maximum depth of 30m and an anticipated dive duration of 2 hours. Kambara carries the sensors and computers it needs for autonomy. In operation, we envision Kambara will receive only occasional supervisory commands, and control its actions with its on-board resources.

We have demonstrated a method of visually-guiding

Figure 1: Kambara

autonomous land vehicles using correlation-based feature tracking to follow targets of interest and to generate motion commands for the robot.[2] Underwater, we will use stereo cameras and again apply feature-tracking techniques. We will image and track features at high rates and use the relative motion of the feature to guide the motion of the AUV. In this manner we intend to follow along a pipe, perform a grid search of the sea floor, or chase a fish.

In this paper we will describe the hardware and software for our underwater vehicle, and describe algorithms for vision-based vehicle guidance. This work has just begun so we report on design and intention rather than results. We welcome comments on our approach.

II. A Simple Underwater Vehicle

Kambara is a simple, low-cost underwater vehicle suitable as a testbed for research in underwater robot autonomy. Kambara's mechanical structure is an open frame which rigidly supports five thrusters and two watertight enclosures. Mounted in the upper enclosure are processors, video digitizers, analog signal digitizers, and communication components. A pan-tilt-zoom camera looks out through the front endcap. Also in the upper enclosure is a proprioceptive sensor package consisting of a triaxial accelerometer, heading compass, and inclination sensor.

The lower enclosure, connected to the upper by a flexible coupling, contains batteries as well as power distribution and charging circuitry. The batteries are 12V, sealed lead-acid with a total capacity of 1200W. Also mounted down below are depth, temperature and leakage sensors.

The frame has length, width, and height of 1.2m, 1.5m, and 0.9m, respectively and displaced volume of approxi-

mately 110 liters. The mass of the frame, enclosures, and thrusters is 66.5kg so the payload mass is 43.5kgs: 37.5kg for batteries and 6kg is for sensors, computers, etc.

Kambara is underactuated with thrusters to provide roll, pitch, yaw, heave, and surge but not sway (lateral) motion. It's thrusters are built from commercially available electric trolling motors. They have been modified with ducts to improve thrust and have custom power amplifiers designed to provide high current to the 24V brushed DC motors. We have put substantial effort into developing compact high-efficiency power amplifiers that mount inside the existing motor housing. A quadrature encoder affixed to the end of the motor shaft will enable closed-loop control PWM signal varying the average voltage, hence velocity.

As Kambara's primary control computing is carried on-board. This size requirement, to fit within the 25cm diameter and 46cm length of the upper enclosure, and the nature of the experiments we would like to perform led us to configure a CompactPCI backplane. A 233MHz PowerPC processor provides adequate capacity for simultaneous filtering, vision computation, and communication. The servo control of the thrusters is handled by an independent Motorola 68332 processor capable of producing PWM signals and receiving encoder feedback. Also in the backplane are video digitizers, and industry pack (IP) modules for serial communication, digital I/O and analog to digital signal conversion. The computing system runs under the VxWorks operating system. The rugged enclosure of the CompactPCI system is shock-mounted and electrically isolated.

In addition to the pan-tilt-zoom camera mounted in the upper enclosure, two color cameras are mounted at 25cm baseline in independent sealed enclosures attached to the frame. Images from these cameras are digitized on-board. The video signal from the stereo cameras is split and also sent up the data tether at full resolution.

During development we will employ a removable fiber-optic tether for command and data communication. The tether contains six fibers: 2 (paired) for 100 Mbit/second Ethernet communication, 3 for full-frame video, and 1 spare. When the system achieves reliable autonomous performance, we will not require real-time video and high-bandwidth communication and will instead record relevant video on-board and communicate at lower rates using acoustic transmission.

III. An Architecture for Supervised Autonomy

Kambara's software architecture is designed to allow autonomy of at various levels: at the signal level for adaptive thruster control, at the tactical level for competent performance of primitive behaviors and at the strategic level for complete mission autonomy.

A. Software Modules

The software modules are designed as independent computational processes that communicate over an anonymous broadcast protocol, organized as shown in Figure 2. The Vehicle Manager is the sole downstream communication module, directing commands to modules running on-board. The Feature Tracker uses visual sensing to follow targets in the environment and uses the relative motion to guide the Swim Generator. The Swim Generator contains the thruster

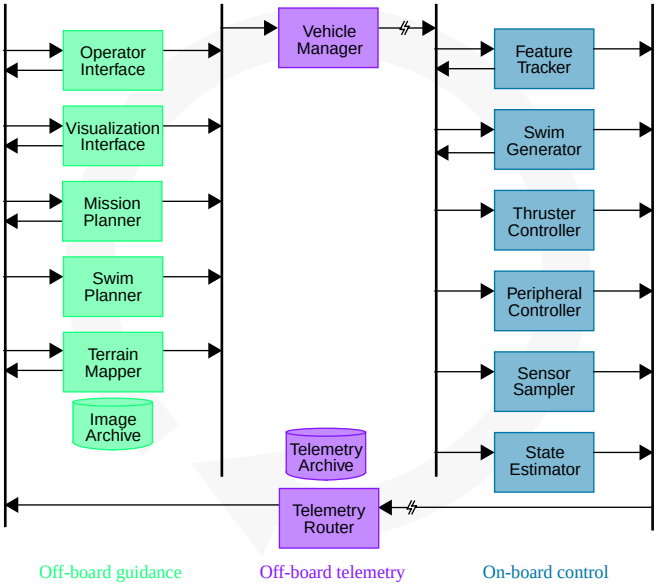

Figure 2: Architecture of on-board and off-board software modules

control mapping and interprets motion commands to produce control signals for the Thruster Controller, which runs closed-loop servo control over the thruster velocities. The Peripheral Controller drives all other devices on the vehicle, for example cameras or scientific instruments. The Sensor Sampler collects sensor information and updates the controllers and the State Estimator. The State Estimator filters sensor information to generate estimates of vehicle position, orientation and velocities. The Telemetry Router moves vehicle state and acquired image and science data off-board.

The Visualization Interface will transform telemetry into a description of vehicle state that can be rendered in a three-dimensional view. The Operator Interface interprets telemetry and presents a numerical expression of vehicle state. It provides method for generating commands to the Vehicle Interface for direct teleoperation of vehicle motion and for supervisory control of the on-board modules. The Operator Interface will operate in conjunction with the Visualization Interface to preview vehicle actions.

The Swim Planner interprets vehicle telemetry to analyze performance and adjust behavior accordingly, for example adjusting velocity profiles to better track a pattern. A Terrain Mapper would transform data (like visual and range images) into maps that can be rendered by the Visualization Interface or used by the Swim Planner to modify behavior. The Mission Planner sequences course changes to produce complex trajectories to autonomously navigate the vehicle to goal locations and carry out complete missions.

B. Operational Modes

The software architecture is designed to accommodate a spectrum of operational modes. Teleoperation of the vehicle with commands fed from the operator directly to the controllers provides the most explicit control of vehicle action. While invaluable during development, this mode is not practical for long-duration operations. Supervised autonomy, in which complex commands are sequenced off-board and then interpreted over time by the modules on-board, will be our

nominal operating mode. Under supervised autonomy, the operator's commands are infrequent and provide guidance rather than direct action commands. The operator gives the equivalent of "swim to that feature" and "remain on station". In fully autonomous operation, the operator is removed from the primary control cycle and planners monitor telemetry to obtain state information and interact to generate infrequent commands for the vehicle. The planners may guide the vehicle over a long traverse, moving from one target to another, or thoroughly exploring a site with no human intervention.

IV. Vision-based Control of a Mobile Robot

Earlier work in the vision-based control of autonomous vehicles was applied to the Marsokhod mobile robot.[2] An overview of the system is depicted in Figure 3. Input imagery comes from a stereo pair of cameras mounted on pan-tilt device. Outputs from this control system consist of angle commands to the pan-tilt device, and steering and velocity commands that drive Marsokhod to the target while keeping the cameras gaze fixed on it.

Two correlation processes track the target. A feature motion correlator tracks the target between previous and current images from one of the cameras and commands the pan-tilt head to keep the target feature centered in the camera's field-of-view. A feature range correlator correlates between left and right images to find pixel disparity, and uses estimated range in conjunction with the bearing data from the feature motion correlator to guide the vehicle to the target. The correlators use same binary correlation algorithm, which was first described in [3]. This algorithm exploits the invariance of the sign of the zero crossing in the Laplacian of the Gaussian of an image. Even in the presence of noise and image intensity shifts, this sign information is stable.[3] Binary correlation offers more efficient implementation over other schemes, such as sum-of-squared differences [4][5] and frequency domain matching [6], since it uses logical rather than arithmetic operations to match the binary sign information, and operates on several pixels at once.

Input images are subsampled, then processed using a difference of Gaussian (DOG) operator. This operator offers many of the same stability properties of the Laplacian operator, but is far simpler to implement on our hardware. We maintain sixteen bit precision in computing this Gaussian, which allows us to use up to a 16 pixel wide Gaussian in computing the DOG image. By selecting different Gaussian sizes and differences, the overall sharpness of the filter can be tuned to match to input images. The DOG image is then binarized based on its sign information. This binary image is then processed by the correlator, which matches a small window of a template image either from a previous frame or from the paired stereo frame. A logical exclusive OR (XOR) operation is used to correlate the template with the input image; matching pixels will always give a value of zero, while non-matching pixels will give a value of one. A lookup table is then used to count the number of matched pixels. By performing correlation of the template over the entire image, the correlator locates the peak where the best match occurred. The distance from center indicates pixel disparity and thus either heading or range to the target.

A. Feature Tracking

We use a sequence of images from one camera of the stereo pair to track the feature. The target is centered within the image from this camera and a target template is extracted. The target, and specifically the template, must exhibit sufficient texture to be distinctive from its surroundings[7]. We have found that even though there are no structured targets, natural terrain has sufficient features of an appropriate scale. The motion correlator matches this template with subsequent images taken as the vehicle advances.

The appearance of the target can change drastically as the vehicle approaches it. The greatest change in appearance occurs when the vehicle nears the target, within two or three meters. At longer distances, the image to image change in appearance, and pixel correlation, is slight. Simply updating the template every correlation cycle would seem to solve this aspect change problem, but small (single pixel) tracking errors integrate each time the template is updated. In the worst case, this causes the correlator to slide off the feature of interest. By using the same template for several correlation cycles, the effect of accumulated error can be reduced.

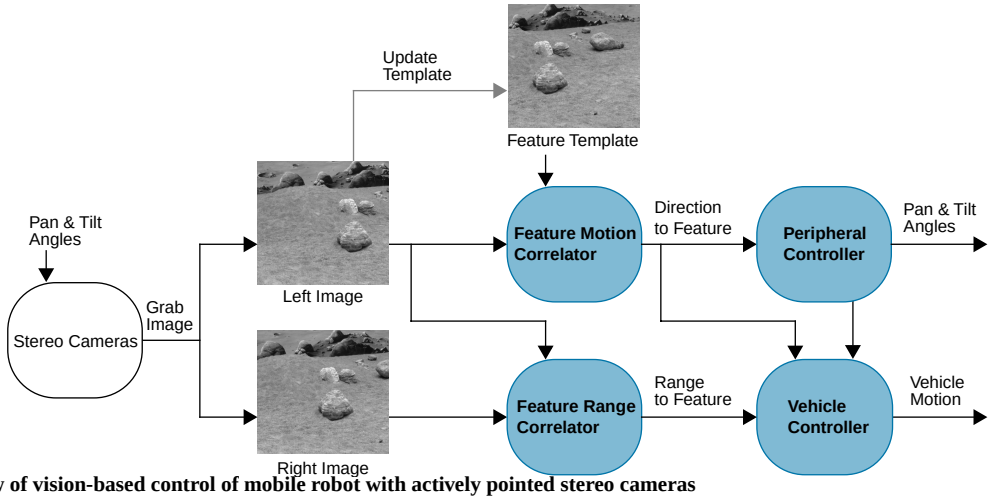

Figure 3: Overview of vision-based control of mobile robot with actively pointed stereo cameras

We found empirically that updating the template every five correlation cycles, or about every 15cm of vehicle motion, is sufficient to handle appearance change without suffering from excessive accumulated correlation error.

B. Range Estimation

Range to a feature is estimated by correlating between left and right stereo images. The range correlator extracts a template containing the feature from the left image, and correlates it against the right image to determine the disparity. This disparity is then converted to an absolute range based on a function determined in a calibration procedure.

An advantage of our approach is that a rough calibration is sufficient. The stereo cameras mounted on a beam of imprecisely known baseline (approximately 25cm) and pointed so that distant (farther than 50m) targets have zero pixel disparity. Relative roll about the optical axis is minimized by maximizing the distant target correlation. We calibrate the range estimate by measuring the pixel disparity for targets placed at various distances from the cameras. In general, disparity is linearly related to the inverse range.[5] If we fit such a function to the disparity and target distance data, we obtain a mapping between pixel disparity and range.

C. Visually Guided Motion

The correlator determines the pan and tilt offsets of the feature relative to the center of the image. These offsets are converted to absolute angles and used by the pan-tilt controller to fix the gaze of the cameras at the feature, regardless of vehicle motion. This gaze fixing is important because the Marsokhod, will pitch and roll in response to disturbances. The pan and tilt angles calculated in the pan-tilt controller are combined with the range estimates from the range correlator to command vehicle motion. The robot is steered left or right to maintain a desired pan offset angle; this offset allows the rover to closely approach a feature without self-occluding. As pan angle increases, the vehicle increases rotation and slows. At extreme pan angle, it rotates in place.

In later development to reduce the area searched for a feature match, a two-layer, feedforward neural network learned the relationship between vehicle orientation and velocities, and the position of the template correlation peak in the next time step. The network was trained using backpropagation with a Levenberg-Marquardt training rule using the correlation peak that was actually found as the correct network output. A skid-steered vehicle has a nonlinear control mapping, yet with no model of the system the neural network can predict the motion of the correlation peak with 85% accuracy. This is sufficient to reduce the area that must be searched, thereby speeding up the tracking process.

V. Learned Control with Visual Feedback

We intend to apply a visually-guided control scheme to Kambara, as diagrammed in Figure 4, based on the system previously described. We propose extending the technique to the domain of underwater vehicles. We first must identify a control mapping from desired position and orientation to thruster speed.

Control of an underwater vehicle is well-established as a nonlinear, time-varying problem.[8] Nonlinear, multivariate control solutions have been developed for specific systems, [9], and sliding mode control, for example [10], is one method for handling the dynamics and compensating for changing external disturbances. Fuzzy techniques have also been applied to compensate for changing dynamics.[11] It seems that not only is the nonlinear system difficult to model, many of the parameters required for the model are unknown either because they are unobservable or because they vary with conditions not incorporated in the model. Hence most stable systems are developed in simulation and only with considerable effort and expense are applied to a specific vehicle with restrictions on its operating regime.[12]

The difficulty in controlling underwater vehicles with a nonlinear control law (or a linearized control law) is twofold: first system identification that leads to stable control under the entire operating regime, and second, system adaptation that modifies gains to accommodate changing conditions. Kambara's open-frame structure makes is particularly susceptible to nonlinear hydrodynamic forces. Further, our intention to exercise all the available degrees-of-freedom make the control design problem difficult as obvious linearizations are not possible.

Learning with an artificial neural network is one way to develop this mapping.[13][14] To model a nonlinear system the neural-network must have multiple-layers and since the

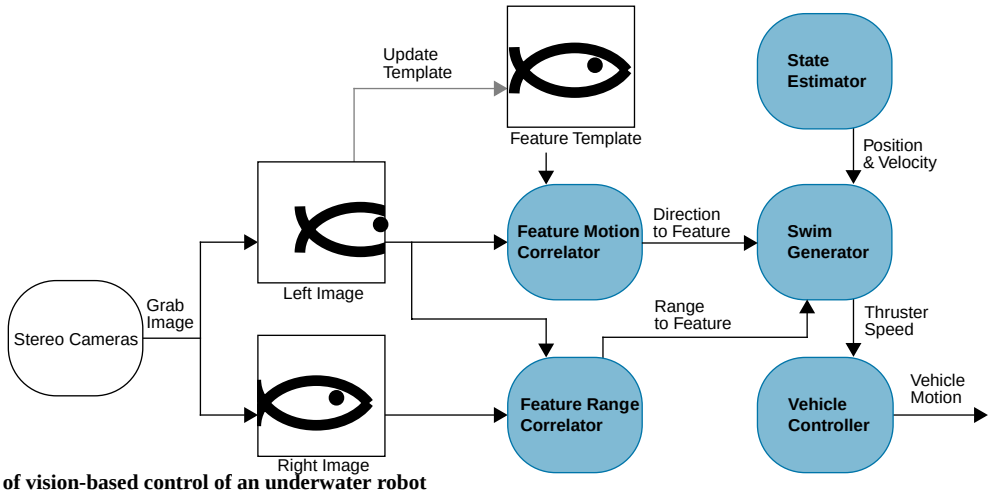

Figure 4: Overview of vision-based control of an underwater robot

effect of vehicle actions is felt over time it seems that a recurrent structure is necessary.[15] By continuing to train the network during operation, the system can adapt to changing conditions. Our approach is to instrument our AUV in its test tank and learn control parameters under actual operation with feedback of the vehicle's position, orientation and velocity. We can then construct the complete visually-guided system using the Feature Tracker to generate the desired position and orientation for the AUV. The yaw angle will be controlled as the pan angle was previously. The tilt angle is used to determine when and how to adjust depth—if the feature is below the cameras, tilt angle will decrease and Kambara should dive, and vice versa with rising tilt angle.

The system developed for Marsokhod, which is also unactuated for lateral motion, has an important property which we exploit for Kambara. As a feature moves out of view laterally, the response of the system is to yaw to keep it in view. Then as the feature moves away, the system approaches closing the distance. On a converging path it will yaw in the opposite direction as the feature moves back across the field of view. This yaw, surge, reverse yaw, accomplishes the necessary sway translation.

As we have shown previously, performance is improved if the future feature location can be predicted. Our next step is to combine the prediction of feature motion with the model vehicle control response and develop a neural network-based Swim Generator, the input layer of which would accept vehicle position, orientation, velocity as well as feature location and the output would produce commanded thruster speeds.

To accommodate the unknowns in time-varying, nonlinear system, neural-network approaches are appropriate. Learning in neural networks can be categorized into supervised, unsupervised, and reinforced.[15] In supervised learning, each input pattern is associated with a correct output pattern. The network weights are gradually adjusted so that error between the network output and the correct output is reduced. Such a strategy is not appropriate in the case of an AUV where correct output is unknown. Unsupervised learning involves distinguishing unspecified patterns in the input data in order to optimize some criterion defined on the output. The weights and outputs converge to represent statistical regularities in the input data. We have applied this method to distinguishing between small and large feature motion but it cannot produce the necessary discrimination to control thrusters. Reinforcement learning adjusts network weights in response to an evaluative signal, which unlike supervised learning need not be a correct output pattern. Reinforcement learning maximizes the probability that the network activity will result in positive external reinforcement.

Yuh [13] proposed a neural network controller to learn the dynamics of an underwater vehicle and produce the appropriate control signals for the vehicles thrusters. The network was trained with reinforcement learning. The reinforcement signal is provided by the error signal produced by the servo-loops in trying to track the commanded reference. Ishii [14] has constructed a adaptive controller trained through back-propagation of simulated system response. Guo [16] has developed an adaptive controller to maintain vehicle heading. The neural network is initially trained with supervisory learning rule with correct output coming from a linear dynamics model, then reinforcement learning is applied with tracking errors (as in [13]) as the feedback signal.

We intend to apply reinforcement learning with reinforcement provided by the position of the target in the visual field. This method differs from previous approaches in that we will couple the visual feature tracking directly to thruster control. The input layer accepts a vector consisting of vehicle position, orientation, linear and angular velocities, camera pan and tilt angles, and the image-plane location of the correlation peak, a hidden layer with recurrent links provides a memory effect and an output layer is trained to produce the vector of thruster commands that cancel the motion of the vehicle and the relative motion of the target.

VI. Summary

We are embarking on the development of autonomous underwater vehicles for exploration and inspection tasks. We have developed an architecture and a method of visually-guiding autonomous vehicles. We will extend this method to incorporate thruster control and use the relative motion of the visual feature for reinforcement learning in an adaptive control scheme.

References

[1] P. Newman, "Towards Terrain Aided Navigation of a Subsea Vehicle," International Conference on Field and Service Robotics, Canberra, Australia, pp. 244-248, December 1997.

[2] D. Wettergreen, H. Thomas, and M. Bualat, "Initial Results from Vision-based Control of the Ames Marsokhod Rover, "IEEE International Conference on Intelligent Robots and Systems, Grenoble, 1997.

[3] K. Nishihara, "Practical Real-Time Imaging Stereo Matcher", Optical Engineering, vol. 23, pp. 536-545, 1984.

[4] M. Okutomi, and T. Kanade, "A Locally Adaptive Window for Signal Matching", CMU-CS-90-178, Carnegie Mellon University, 1990.

[5] L. Matthies, "Dynamic Stereo Vision", CMU-CS-89-195, Carnegie Mellon University, 1989.

[6] D. Coombs, "Real-Time Gaze Holding in Binocular Robot Vision," Computer Science Technical Report 415, University of Rochester,1992.

[7] J. Shi, C. Tomasi, "Good features to track," IEEE Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.

[8] T. Fossen, "Underwater Vehicle Dynamics," *Underwater Robotic Vehicles: Design and Control*, J. Yuh (Editor), TSI Press, pp.15-40, 1995.

[9] D. Yoerger, J-J. Slotine, "Robust Trajectory Control of Underwater Vehicles," IEEE Journal of Oceanic Engineering, OE-10, no. 4, 1985.

[10] A. Healey, D. Lienard, "Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles," IEEE Journal of Oceanic Engineering, vol. 18, no. 3, pp. 327-338, July 1993.

[11] W. Lee, G. Kang, "A Fuzzy Model-based Controller of an Underwater Robotic Vehicle under the Influence of Thruster Dynamics," IEEE Intl. Conference on Robotics and Automation, Leuven, pp. 750-755, 1998.

[12] K. Goheen, "Techniques for URV Modeling," *Underwater Robotic Vehicles: Design and Control*, J. Yuh (Ed), TSI Press, pp.99-126, 1995.

[13] J. Yuh, "A Neural Net Controller for Underwater Robotic Vehicles," IEEE Journal of Oceanic Engineering, vol. 15, no. 3, pp. 161-166, 1990.

[14] K. Ishii, T. Fujii, T. Ura, "Neural Network System for On-line Controller Adaptation and Its Application to Underwater Robot," IEEE Intl. Conference on Robotics and Automation, Leuven, pp. 756-761, 1998.

[15] M. Hassoun, Fundamentals of Artificial Neural Networks, MIT Press, Cambridge, 1995.

[16] J. Guo, F. Chiu, C-C. Wang, "Adaptive Control of an Autonomous Underwater Vehicle Testbed Using Neural Networks," OCEANS'95, San Diego, vol. 2., pp. 1033-1392, 1995.