Sensor-Based Behavior Control for an Autonomous Underwater Vehicle

Gregory Dudek and Philippe Giguere and Junaed Sattar

Centre for Intelligent Machines
McGill University
{dudek, philg, junaed}@cim.mgill.ca
http://www.cim.mgill.ca/~mrl

Abstract. In this paper, we present behaviors and interaction modes for a small underwater robot. In particular, we address some challenging issues arising from the underwater environment: visual processing, interactive communication with an underwater crew, and finally orientation and motion of the vehicle through a hovering mode. The visual processing consist of target tracking using various technique (color blob, color histogram and mean shift). The underwater communication is achieved through printed cards with virtual markers (ARTag). Finally, the hovering "gait" developed for this vehicle relies on the planned motion of six flippers to generate the appropriate forces.

1 Motivation and Problem Statement

This paper considers non-contact guidance for an amphibious robot, based on visual cues. In particular, this is motivated by the challenges of communication underwater. Teleoperation for underwater vehicles is complicated by several factors: wireless communication is problematic since conventional radio communications are infeasible, the use of a tether is awkward on land and even worse underwater in the face of buoyancy issues and 6 DOF motion, and other communication mechanisms have their own deficiencies. Notably, even human scuba divers commonly resort to simple sign language and similar short range visual communications for underwater task coordination. In a similar manner, our underwater vehicle is being developed to combine behaviors and operating modes based on visual cues.

Our target application is the visual surveillance of reef environments to assess the health and behavior of the marine life. This task, like many related inspection tasks, can be decomposed into two canonical behaviors: transition between way points, and station keeping at a fixed location. In our application, each of these is modulated by visual cues from a diver or in response to environmental stimuli (such as recognized visual landmarks). These cues take two forms: symbolic tokens used to select specific behavior classes or tune the behaviors, and target motions used for visual servoing with respect to either a diver or an environmental stimulus. Motion between the way points is performed by executing one of several swimming gaits. Station keeping, however, entails the use of a hovering gait which is both unique and challenging.

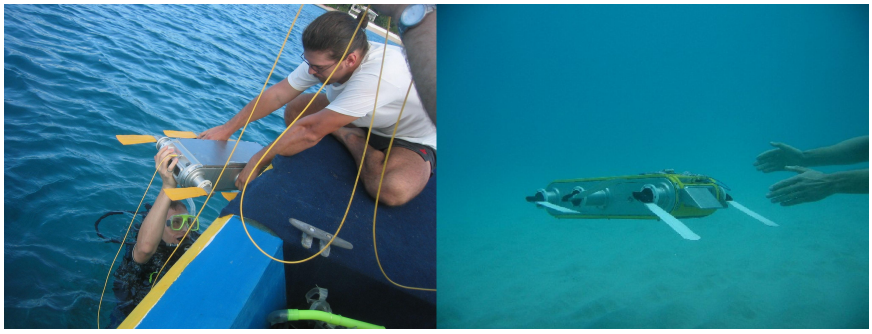


Fig. 1. The AQUA robot being deployed during open-sea experimentations (on the left) and operated in untethered mode (on the right).

Our vehicle, a descent of the RHex hexapod robot [1], has well-developed "kicking" gaits for forward locomotion that permit limited amounts of pitch, roll and yaw. These gaits are based in simple oscillatory motions of the flippers with various phase and amplitude offsets, akin to the standard up-and-down kick of a human swimmer. In this standard mode of motion, however, rotational motion is coupled to forward motion; the robot can only turn if it is moving forward. Furthermore, thrust can only be "instantaneously" applied in the forward direction (or backward if the flipper orientation is reversed).

The hovering gaits were conceived with several requirements in mind. First those gaits had to be able to move the robot in 5 degrees of freedom: pitch, roll, yaw, heave and surge. They also had to be able to combine several commands at the same time, for example a pitch and heave command. Furthermore to ensure good controllability, the reaction time of the robot had be kept to a minimum, particularly in the case of command reversal. Finally, the cross-talk between the degrees of freedom was minimized: in order to hover in place effectively, the robot needs to be able to apply rotational moments in any direction with very limited application of net forward thrust. The intrinsically non-holonomic behavior of the flippers presented a significant challenge in the design of hovering gaits.

2 Technical Approach

In order to adapt to natural environments and compensate for unforeseen forces in the environment, a key aspect of our work is the sensing of environmental conditions and the recognition of the current context. To do this we use a combination of internal sensors akin to biological proprioception as well as computer vision. This adaptation process therefore falls into two distinct categories: visual processing and interpretation, and gait synthesis, selection and control. The visual processing is further subdivided into learning-based visual servoing, and symbol detection and interpretation (i.e. a visual language akin to a simplified sign language).

2.1 Visual Processing and Interpretation

A primary interaction mechanism for controlling the robot is via *visual servoing* using a chromatic target. By detecting a target object of a pre-specified color, the robot is able to track the target in image-space up to a practical operational maximum distance of approximately two meters [6]. We currently use color blob, color histogram [7] and mean-shift [2] based tracking algorithms for target tracking. A *proportional-integral-derivative* controller takes the tracker outputs and generates yaw and pitch (but not roll) commands which determine the robot trajectory and makes the target following behavior possible. For the surveillance of a motionless target, the hovering gait of the robot is used.

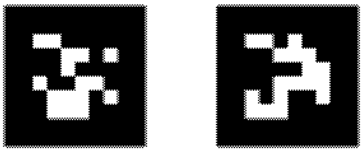


Fig. 2. ARTag Markers.

The servoing mechanism is configurable through a number of parameters that include target color properties, robot speed, gait selection and yaw/pitch gains. We use symbolic markers provided by the ARTag toolkit [3] to visually communicate with the robot and affect changes in robot behavior. An example of an ARTag marker is shown in figure 2. These markers include both symbolic and geometric content, and are constructed using an error-correcting code to enhance robustness. Switching in and out of the hovering gait, for example, is performed by detecting a particular ARTag marker.

2.2 Gait Control Overview

The gait design and control issues we consider are for a swimming robot that uses paddles (i.e. legs) for locomotion underwater. By using legs for locomotion our vehicle is able to swim underwater and walk on land. Many underwater tasks entail holding a fixed position while some task is accomplished, either surveillance or manipulation. Our robot is able to use its legs to land on the sea bottom with limited disturbance and perform certain types of surveillance task. A large class of activities is facilitated, however, by being able to hold a fixed position in middle depths, for example to monitor sea life on a coral reef, a key application of our robot.

3 Methodology and Results

Our experimental methodology is comprised of three sequential phases: numerical simulation and validation on recorded video data, pool trials in a controlled environment, and open water tests on a live coral reef. The latter scenario is the real test of

performance, but tends to provide qualitative results and video data. The deployment of the robot at sea can take place from shore or from a boat.

3.1 Visual Servoing and Tag-based Control

The integrated monocular vision system in the Aqua robot currently serves two purposes; namely visual servoing and tag-based robot control. The visual servoing behavior is used to track and follow an object of interest (*i.e.* fish, a diver etc) underwater. The tag-based navigation mode is based on the ARTag toolkit [3] , and is used to send basic motion control commands to the robot. Both these modes run in parallel; visual servoing mode can be preempted by motion control commands sent by the tag-based motion control subsystem. We discuss both these systems briefly in the two subsections that follow.

Visual Servoing The visual servoing subsystem is made up of two functional components – the visual tracker and the proportional-integral-derivative (PID) controller. The visual tracker tracks objects of interest, or targets, based on the color features of the targets. The tracking system localizes a target in image space, in Cartesian coordinates, with location (0,0) being the center of the image frame. To track an object, we use its color properties. Both single and multicolor objects can be tracked. Currently, our system is comprised of two different approaches to visual tracking. One type of approach is a naive, brute-force approach to target localization. An example of this is the color segmentation tracker, that uses trivial threshold-based color segmentation to detect the target in the image frame. We utilize statistical approaches to visual tracking as well. The histogram tracker and mean-shit tracker uses statistical similarity measures between color histograms to detect probable target location between successive image frames. These trackers differ in their methods of locating the target in consecutive frames; the histogram tracker does a global search in the entire image to locate the target, whereas the mean-shift tracker uses the mean-shift vector to detect the shift in target location and thereby reacquire the target in the next frames.

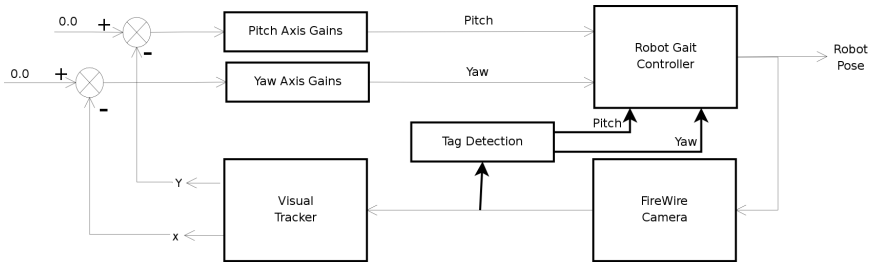


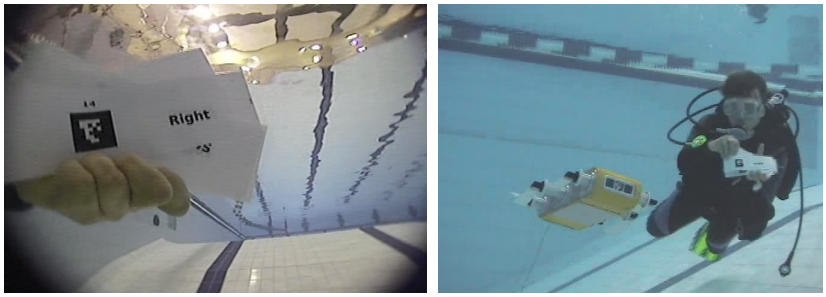
Fig. 3. Visual servoing architecture in AQUA.

We use a PID controller that takes as input the image coordinates (and size) of the target in image space, and emits yaw and pitch commands (and optionally speed) as outputs that are sent to a auxiliary gait-control computer to modify the robot's behavior and thus pose. The controller also embodies a low-pass filter, smoothing out random changes in yaw and pitch commands that are sent from the visual tracker. Since the robots swimming gaits produce a low-frequency oscillatory body motion, these motions need to be accounted for in the tracking stage and gait tuning need to be made to avoid disruption the ongoing gaits. An overview of the visual servoing architecture can be seen in Fig. 3.

Fig. 4. Visual servoing off a yellow target, open ocean trials.

Tag-based Navigation Control of the swimming robot typically entails manipulation of various parameters spanning power control (on/off), gait selection, gait tuning and behavior control. As noted above we have developed an interface mechanism to allow a diver control the robot without recourse to a tether or cumbersome or costly consoles. The technique is based on the use of robust visual targets (called "ARTags") than can be manipulated and used to iconically express commands to the robot. ARTag detector system operates in parallel to the servoing system and either can be used or suppressed as needed. In practice, ARTag (icon) detection is performed every 2 to 3 seconds at present. This latency makes the overhead in tag image processing minimal, and is sufficient given the relatively slow motions of the divers. When a tag that corresponds to a motion command is detected, for example, the vision system interpolated the command into any ongoing servo-control stream (accounting for dynamics and gait changes) and sends the appropriate yaw and pitch commands to the gait controller to achieve the desired behavior. Currently, the primitive motion commands include are pitch up and down, yaw left and right, and starting and stopping. The mapping from a tag number to motion commands is preset beforehand and cannot be modified once the system is in operation. The icon interpreter also includes the ability to express complex compound control sequences, but the effectiveness of this remains to be determined.

The current system includes a minimalistic way to modify the duration of the motion command sent through the ARTags. We perform simple addition on two



(a) Robot's eye view, showing ARTag icon and human-readable translation

(b) External view with diver holding tag dictionary

Fig. 5. Tag-based navigation in action.

numbers ranging from 1 to 10, identified by their tag numbers. A sum of the two digits is taken when a third digit is detected. This new value is the duration of the motion commands, until another arithmetic operation is performed to change the command length. By default, the robot is programmed to perform a maneuver as directed by a tag for 3 seconds. While this method of showing numbers and performing basic addition to change the duration is inherently simple, it demonstrates the programmability of the system by showing tags to modify internal parameters of the robot controller. Figure 5 shows a diver commanding the robot to turn right by showing the appropriate ARTag.

3.2 Hovering Gait Implementation

A key function for our system is video acquisition, both while the robot is moving as well as at a fixed location. For in-place surveillance the robot can land, be it has proven valuable to implement a novel hovering gait implemented by two component modules. The first module (Vectored-Thrust Computation Module) computes the required thrust at each flipper, based on the 5 possible commands (pitch C_p, roll C_r, yaw C_y, heave C_h and surge C_s). The second module (Flipper Thrust Module) computes the individual flipper motion needed to track the desired thrust vector.

Vectored-Thrust Computation Module Let $\mathbf{C} = [C_p\ C_r\ C_y\ C_h\ C_s]^T$ be the input command column vector. Let $\mathbf{F_x}$ and $\mathbf{F_z}$ be the column vectors representing the desired thrust at each flipper location in the x and z direction (they cannot

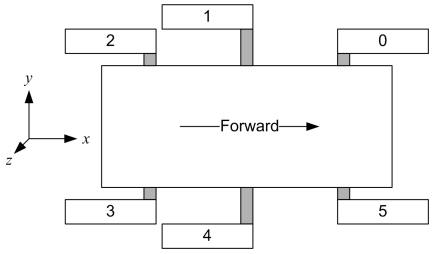


Fig. 6. Flipper placement, orientation and numbering. Note that flippers 0 and 5 are facing forward to provide an extended moment arm and symmetric pitching moments with flippers 2 and 3, as well as providing quick reverse surge.

generate thrust in the y direction). The Vectored-Thrust Computation computation can be expressed as:

$$\mathbf{F_x} = \begin{pmatrix} 0 & 0 & -k_y & 0 & k_s \\ 0 & 0 & -k_y & 0 & k_s \\ 0 & 0 & -k_y & 0 & k_s \\ 0 & 0 & k_y & 0 & k_s \\ 0 & 0 & k_y & 0 & k_s \\ 0 & 0 & k_y & 0 & k_s \end{pmatrix} \times \mathbf{C}, \qquad \mathbf{F_z} = \begin{pmatrix} k_p & k_r & 0 & k_h & 0 \\ 0 & k_r & 0 & k_h & 0 \\ -k_p & k_r & 0 & k_h & 0 \\ -k_p & -k_r & 0 & k_h & 0 \\ 0 & -k_r & 0 & k_h & 0 \\ k_p & -k_r & 0 & k_h & 0 \end{pmatrix} \times \mathbf{C}$$

The constants k_p, k_r, k_y, k_h and k_s are used to scale the input so that the absolute maximum value of the output is less than or equal to 1. The size of the column vectors $\mathbf{F_x}$ and $\mathbf{F_z}$ is equal to the number of flippers on the robot (6). The index representing the flippers is shown in Fig. 3.2.

The selected thrust angle and magnitude for flipper n thus:

$$\theta_n = arctan(\frac{F_{zn}}{F_{xn}}), \qquad T_n = \sqrt{{F_{xn}}^2 + {F_{zn}}^2} \tag{1}$$

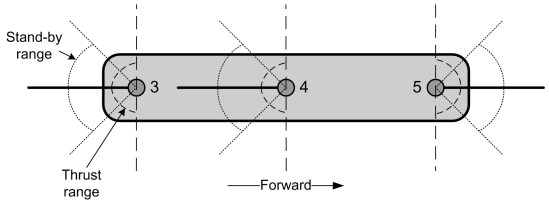
Flipper Thrust Module The most efficient way to generate thrust in direction θ_r with our flat flipper is to make it oscillate rapidly (period of 250-500 ms) around that angle θ_r. The magnitude of the thrust is approximately proportional to the amplitude of the oscillation. The flipper angle θ_f over time is thus:

$$\theta_f(t) = T_n sin(\omega_f t + \phi_f) + \theta_r \tag{2}$$

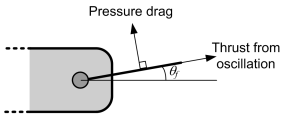
Where ω_f is the fixed frequency of oscillation and a phase shift ϕ_f is used between flippers.

One significant issue for generating the thrust this way is the holonomic constraints on the flippers with respect to thrust direction: the flippers must be first oriented at the new desired thrust angle. However, flipper re-orientation produces a force normal to its surface through pressure drag. For example, if a flipper is pointing forward but forward thrust is needed, then the flipper must first be rotated 180^o. This

rotation generates parasitic forces with orientation depending on the direction of rotation. Moreover, this also implies delays in the execution of commands. To mitigate these problems, we first limit the range of thrust angles for each flipper to a region of 180^o (see Fig. 7(a).), as is done similarly in [4]. This reduces the average reorientation angle responsible for the parasitic forces, at the cost of reduced maximum vehicle thrust. For example, the front flippers are not used when a forward thrust is commanded, thereby reducing the maximum possible forward thrust for the vehicle. To further improve the reaction time of the robot, we make use of the pressure drag



(a) Flipper angle limitations.



(b) Forces generated by a flipper.

Fig. 7. (a) Flipper angle limitations for flippers 3, 4 and 5. Flipper 5 will be generating backward thrust, while 3 and 4 be generating forward thrust. (b) Schematic force diagram for one flipper.

forces generated when the flippers are re-oriented. When the difference between the desired thrust angle and the current flipper angle is greater than 45^o, the flipper is rotated at a rate that generates a pressure drag consistent with the desired thrust T_c via the constant K_{PD}. As the flipper surface passes the 45^o region, the oscillation amplitude is increased until it reaches its selected amplitude as given by Eq. 2. Using discrete-time equations and letting θ_r be the ramped value of the computed thrust angle θ_c and magnitude T_c, Eq. 3 and Eq. 4 show our complete Flipper Thrust Module. The $ramp(rate, a, b)$ function will ramp value b toward a at a constant rate $rate$.

$$\theta_r[t+1] = \begin{cases} \theta_r[t] & \text{if } \theta_c[t] \text{ outside thrust range} \\ ramp(K_{PD}T_c, \theta_r[t], \theta_c[t]) & \text{otherwise} \end{cases} \quad (3)$$

$$\theta_f[t+1] = \begin{cases} \theta_r[r] & \text{if } |\theta_c[t] - \theta_f[t]| > 45^o \\ T_c \frac{|\theta_{r[t]} - \theta_f[t]|}{45^o} sin(\omega_f t + \phi_f) + \theta_r[t] & \text{otherwise} \end{cases} \quad (4)$$

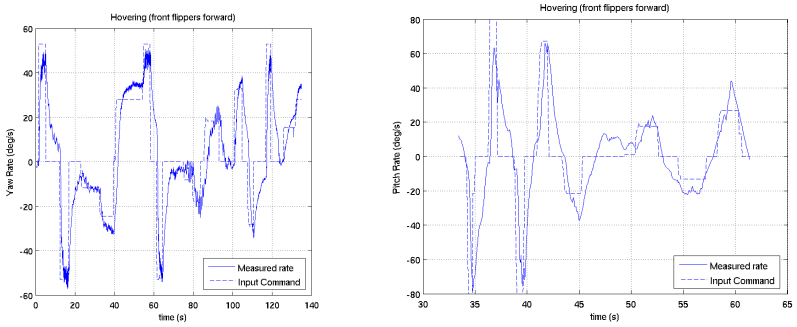
Finally, to improve slow-changing commands, when the demanded thrust T_c reaches zero, the flippers are gradually moved back to the stand-by range of 90^o (see Fig. 7(a)). This guarantees that the flippers are always able to generate the proper thrust rapidly, by making the flipper surface or its normal no more than 45^o away from any desired thrust within the 180^o window.

4 Results and Experiments

We have evaluated the visual signaling and servoing subsystem both in the lab, in simulated environments, and also in closed and open-water trials, in the pool and in the open ocean. The open ocean servo trials were held in the sea off Barbados. The lab tests were performed in a glass tank filled with water and other particles and sediment to simulate the underwater environment in the open seas. We tracked single and multi-colored objects and evaluated the performance of the three trackers by manually comparing the number of mis-tracked frames against those where successful tracking was performed [5]. In the pool and sea trials, a yellow ball of approximately 15cm in diameter was used as a target, which the robot followed over a maximum distance of 27 meters (See Fig. 4); the distance was limited by the length of the fiber-optic tether. Quantitative performance of the tracking algorithms were obtained from the lab trials, whereas the open ocean trials provided information about the robots behavior under the visual guidance mode.

The tag-based navigation system was tested in a pool environment to date. We performed both standalone tag command navigation as well as servo-and-navigate experiments. In the first case, the diver was successfully and comfortably able to guide the robot around the pool using tags expressing 6 distinct behaviors and various numerical parameters. (see Fig. 5. The vision system picked up 2-inch square-sized tags at a distance of 2 meters, although that number will probably reduce in open ocean where visibility is not as good as in a pool.

The usability of the system appeared to be excellent and, in particular, was far more convenient that prior methods based on visual communication with an observing human (who modified parameters via a tether). We are moving towards definitive usability studies.



(a) Yaw rate, actual and desired, hovering mode.

(b) Pitch rate compared to desired rate, in hovering mode.

Fig. 8. Yaw and pitch accuracy.

The hovering gaits were evaluated in both closed pools as well as open water, both separately and in conjunction with visual servoing. This allowed us to assess its

performance and correct any major model failures. In testing hovering in isolation the 5 commands were manually input in real-time to the system via a joystick connected by fiber-optic cable. The human operator would then qualitatively evaluate the motion response of the robot, using the divers as visual landmarks (and fixed fiducials in a pool setting). Quantitative data was then collected in the from of orientation and angular velocities coming from an inertial measurement unit (calibrated previously). Further tests included moving the robot within a triangular formation of divers, roughly separated by 5 m. Fig. 8(a) shows the result of testing single yawing commands and Fig. 8(b) for simple pitching commands. For clarity, the input signals C_y and C_p have been scaled to roughly match the output angular rate of the vehicle. The general performance of the robot is good, although the yawing motions entail greater delays than pitching motions. The rapid reversal of direction effective, satisfying one of the design criteria spelled earlier.

4.1 Conclusions

We have developed and validated an approach to vision-based control of an aquatic robot. Hovering and servoing in-place are relatively novel abilities which we have deployed using judicious flipper positioning. Hovering, in particular, is an exotic yet useful ability.

The effectiveness of these our behaviors and subsystems were evaluated in various combinations both in a controlled pool setting (Fig. 5) as well as in open water (Caribbean Sea and coastal Nova Scotia). Results have been very successful, but each case, the acquisition of accurate quantitative performance data was a major challenge. Inertial measurement data has been effective, but due to its potential for drift the results are not equivocal. Overall, purely visual guidance and control seems much more effective than originally anticipated.

References

1. R. Altendorfer. Rhex: A biologically inspired hexapod runner. *Autonomous Robots*, 11:207–213, 2001.
2. D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(5):564–575, 2003.
3. M. Fiala. ARTag Revision 1, a fiducial marker system using digital techniques. Technical Report NRC 47419, National Research Council, Canada, November 2004.
4. M. Kemp, B. Hobson, and J. J.H. Long. Madeleine: an agile auv propelled by flexible fins. In *14th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Lee, New Hampshire, May 2005.
5. J. Sattar and G. Dudek. On the performance of color tracking algorithms for underwater robots under varying lighting and visibility. In *International Conference on Robotics and Automation, ICRA 2006*, Orlando, Florida, May 2006.
6. J. Sattar, P. Giguère, G. Dudek, and C. Prahacs. A visual servoing system for an aquatic swimming robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, August 2005.
7. M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.