A Boosting Approach to Visual Servo-Control of an Underwater Robot

Junaed Sattar and and Gregory Dudek

Center for Intelligent Machines,
McGill University, Montréal, Canada H3A 1S1.
{junaed, dudek}@cim.mcgill.ca

Summary. We present an application of the *ensemble learning* algorithm in the area of visual tracking and servoing. In particular, we investigate an approach based on the Boosting technique for robust visual tracking of color objects in an underwater environment. To this end, we use AdaBoost, the most common variant of the Boosting algorithm, to select a number of low-complexity but moderately accurate color feature trackers and we combine their outputs. From a significantly large number of "weak" color trackers, the training process selects those which exhibit reasonably good performance (in terms of mistracking and false positives), and assigns positive weights to these trackers. The tracking process applies these trackers on the input video frames, and the final tracker output is chosen based on the weights of the final array of trackers. By using computationally inexpensive but somewhat accurate trackers as members of the ensemble, the system is able to run at quasi-real time, and thus, is deployable on-board our underwater robot. We present quantitative cross-validation results of our visual tracker, and conclude by pointing out some difficulties faced and subsequent shortcomings in the experiments we performed, along with directions of future research on the area of ensemble tracking in real-time.

1 Problem Statement

We investigate the application of machine learning algorithms, ensemble learning in particular, to visual tracking and vision-based robot control. This work looks at using machine learning to create a robust tracking system for an autonomous amphibious legged robotic vehicle. We look into the unique challenges posed by the underwater environment to machine vision systems and also at the effect of learning algorithms on tracking accuracy and performance. To this end, we use a number of low-complexity but moderately accurate trackers and we 'boost' their outputs using the AdaBoost algorithm. The training process we use obtains weights for the AdaBoost 'weak learners', which in our case are a set of computationally-inexpensive visual trackers. The final tracking decision is a weighted average of the outputs of each individual tracker that were chosen by AdaBoost to be a part of the ensemble. The goal is to have a robust visual tracker which will be able to overcome the challenges of reduced and degraded visibility, by automatically choosing a subset of the visual trackers at its disposal that performs well under the conditions.

2 Overview

An application domain of rapidly increasing significance, the underwater domain is rife with challenges of both scientific and pragmatic importance. While computer vision has matured enormously in the last few decades, the peculiarities of underwater (sub-sea) vision have been largely ignored, presumably due to the enormous logistic and pragmatic overhead in examining them (just as the actual topography and zoology of the sub-sea environment has been ignored relative to the terrestrial analogs). Vision can be as valuable a sensing medium underwater, and perhaps even more significant than on land. Simple inspection of marina fauna demonstrates the ubiquity of eyes, and other optical sensors, in the marine environment and thus

suggests its potential utility. In our particular application we are interested in tracking a diver as he swims either along the surface or under water (using scuba apparatus). In this case we need a tracking technology that imposes a very limited cognitive load on the robot driver, which operates despite variations in lighting (due to refractive effects and/or waves), which is immune to nearby wave action and which operates over a moderate range of distances.

Vision has the advantage of being a passive sensing medium, and is thus both non-intrusive and energy efficient. These are both important considerations (in contrast to sonar) in a range of applications ranging from environmental assays to security surveillance. Alternative sensing media such as sonar also suffer from several deficiencies which make them difficult to use for tracking moving targets at close range in potentially turbulent water. In addition, challenges arising from variable lighting, water salinity, suspended particles, color degradation and other similar issues often invalidates vision algorithms that are known to work well in terrestrial environments. Since the underwater environment is hazardous for humans, more so for extended periods of time, an autonomous robotic vehicle thus has certain advantages for many underwater applications. Such applications include but are not limited to marine life inspection, cable or pipeline inspection, search-and-rescue etc. Our vehicle, the Aqua amphibious legged robot [4] is being developed as a vision-based autonomous system with such applications in mind. This work looks at overcoming challenges posed by the underwater domain in the field of visual tracking and servoing by using methods from machine learning. The underwater domain poses complicated challenges in designing efficient vision systems, as quantitative analysis is difficult to perform in sub-sea environments. With the goal of circumventing these issues, we make use of video footage and ensemble learning, with minimum user input, to come up with an efficient tracking algorithm for underwater targets.

3 Background and related work

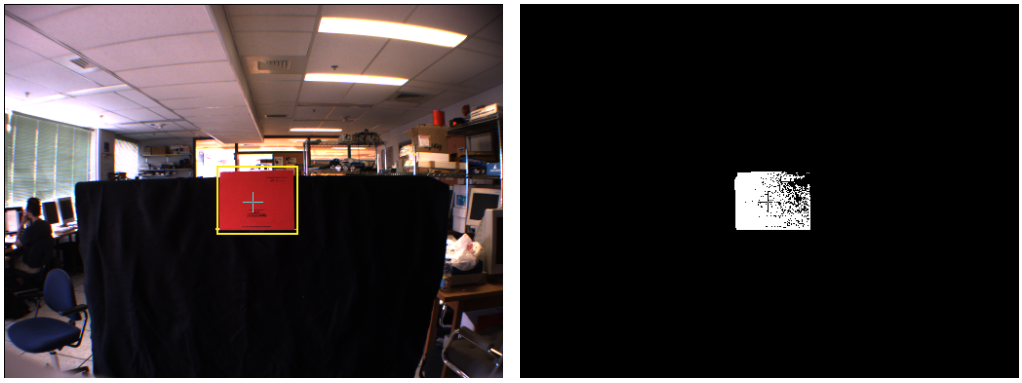
This section looks at past research on underwater vision applications, as well as machine learning in general, with a particular focus on ensemble learning. Machine learning has been extensively used in the recent past in applications in machine vision, and we briefly discuss few such applications in the following sections.

3.1 Visual Tracking and Servoing

In general, visual servoing [8] refers to the task of controlling the pose of the robot or a robotic end-effector mounted on the robot by using visual information. This commonly closed-loop system is used with "eye-on-hand" camera configuration in manufacturing industries, but there have been a quite few applications of vehicle control with a fixed camera mounted on the robot for car steering and aircraft landing, among others [9]. Although a number of different approaches to visual servoing exist, the process fundamentally works by tracking an object or set of objects in successive image frames and controlling the robot manipulator or motors using the information provided by the tracker [1]. In *image-based visual servoing*, target pose estimation is not necessary, as opposed to *position-based visual servoing*, which saves significant computational costs. For a real-time system like Aqua, reducing this overhead improves the overall system performance.

3.2 Machine Learning and Ensemble Classifiers

Ensemble methods [5] such as Boosting or Bagging combine several hypotheses into one prediction. They work better than the best individual hypothesis from the same class because they reduce bias or variance (or both) [10]. Boosting is a method of finding a highly accurate hypothesis (classification rule) by combining many "weak" hypotheses, each of which is only moderately accurate. By moderately accurate, it is implied that the 'weak' learning algorithms perform better than random guessing; *i.e.* with a error rate of less than 50 *per cent.* Typically, each weak hypothesis is a simple rule which can be used to generate a predicted classification for any instance. Theoretical analysis has shown [11] that Boosting does not suffer from overfitting issues, unlike other classification methods. The AdaBoost algorithm [6], first introduced by Freund



(a) A red target. (b) Segmented output showing the output blob.

Fig. 1. A color blob tracker tracking a red-colored object.

and Schapire, is a fast and robust Boosting algorithm. AdaBoost works by maintaining a set of weights over the training set. Initial weights are set uniformly, but as training progresses, the weights of the wrongly classified training examples are increased to force the subsequent weak learners to focus more on the 'hard' problems and boost overall accuracy. In the past, this approach has been used in people detection in the seminal work by Viola and Jones [13], and visual tracking [2] in computer vision applications. Mobile robot localization [12], gait selection [3] and environmental estimation problems [7] have also seen applications of various other machine learning techniques.

Many other variations of the AdaBoost and other Boosting algorithms exist, for multiclass problems (AdaBoost M2, as an example) and regression, although in this work we use the original AdaBoost algorithm for classification.

4 Technical Approach

The following subsections describe the core of our approach to robust visual tracking using AdaBoost learning. Section 4.1 describes briefly the color threshold tracking algorithm we use as weak learners. Technical details of the learning process is described in Sec. 4.2, and tracker selection is explained in Sec. 4.3.

4.1 Visual Tracking by Color Thresholding

In a thresholding-based tracking approach, a segmentation algorithm outputs (possibly disconnected) regions in a binary image that match the color properties of the tracked object. These regions are commonly referred to as 'blobs', and hence the approach is often known as color-blob tracking. We attempt to form these blobs through a *thresholding* process. By thresholding, we refer to the operation where pixels are turned 'on' if and only if their color values fall within a certain range and turned 'off' otherwise.

The tracker is initialized with the target's color properties; for example, in the RGB space, the tracker is initialized with the red, green and blue color values of the tracked object, within some degree of tolerance. Next, a pixel-by-pixel sequential scanning is performed on the image. The pixels falling within the threshold of the color values of the target are switched on in the output image, and other pixels are turned off. Figure 1 below shows the segmentation output of tracking a red object. The target is framed by a yellow rectangle for clarity. The tracker was tuned beforehand to the red rectangular target in Fig. 1(a). The segmentation process produced the image in Fig. 1(b). The tracking algorithm detects this blob in the binary image in every frame, and calculates its centroid. This centroid is taken as the new target location. This process iterates over every frame to achieve target localization in the image space.

One rather obvious downside to using a naive color blob tracker as explained above, is the presence of duplicate targets. For example, in Fig. 1(a) above, if any other red colored object appears in the scene, the segmentation process will generate another blob for that object. This second blob will effect the calculation of the center of mass for the tracker; the effect will be more prominent if the two generated blobs are disconnected; *i.e.* further away in the image frame. Therefore, the tracker works only accurately when there are no similarly-colored object in the camera's field-of-view. Coupling the simple blob tracker with a prior for the position of the target in the previous frames can eliminate this problem by a large degree. A mean-shift tracker has been proven useful [15] in achieving exactly this goal.

4.2 Ensemble Tracking

In this approach to ensemble tracking, we apply the Boosting algorithm to visual tracking and address tracking as a binary classification problem. An ensemble of classifiers (subsequently referred to as *trackers*) are trained to distinguish between the background and the target object. These weak trackers are then combined and their outputs are strengthened using AdaBoost. Figure 2 shows an outline of the boosted tracking process that uses color channel trackers as weak learners.

For visual tracking in color, we choose the normalized-RGB color space, as it has been shown to be more robust against lighting variations, and such variations are predominant in underwater vision systems. Our weak trackers are a collection of individual channel trackers, working on the red, green and blue channels independently, or together. The training process randomly picks one from a bank of trackers, and sets the associated thresholds for that tracker in the normalized-RGB space, attempting to fit the training set to the tracker. If for any instance in the training set, the tracker outputs the target location within a small margin of error, we label it as a correct output for that instance. Otherwise, the label is set as incorrect. Any tracker that has more than 50 *per cent* error rate is discarded by the Boosting process. At each iteration, any training instance that has been misclassified by the current tracker, is given a higher weight according to the AdaBoost algorithm. This ensures that the subsequent trackers focus more on this instance that was wrongly classified by the current tracker. The cycle iterates for a certain number of Boosting rounds, or until the improvement in the training error falls below a certain threshold. At the end of the training process, an array of weak trackers and their associated weights are available for use in the tracking process. The final chosen trackers are usually a mixture of different tracker types, both in terms of threshold and design. We discuss more about the trackers used in the training stage in Sec. 4.3.

Currently, we perform training as an offline process, using video sequences of target objects to train our trackers, as well as synthetic data. Weights for the weak trackers are generated by AdaBoost, which are

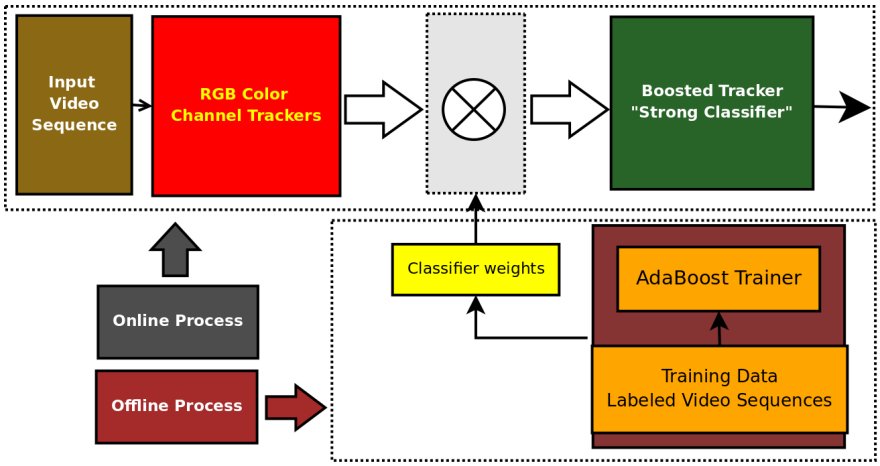


Fig. 2. Outline of the ensemble tracker using AdaBoost

passed on to the on-line stage, where they are multiplied by the individual tracker outputs and eventually summed up to create the final tracking decision.

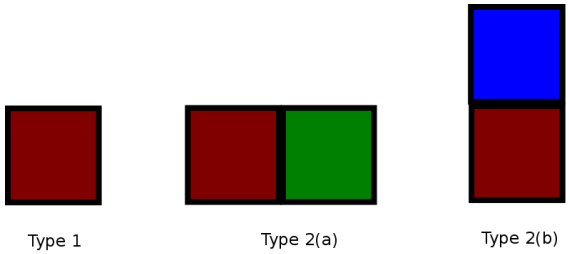


Fig. 3. Examples of the three tracker types used during Boosting.

4.3 Choice of trackers

For weak trackers in our approach, RGB channel trackers are chosen to track the target objects and output the locations individually, or in combination with each other. We use trackers for each of the red, blue and green channels, and combine them in three different ways. *Type 1* trackers consist of one single threshold of either R,G or B channel, and segments the image based on this threshold and color channel to track. *Type 2* trackers are a combination of two channels, and their associated two thresholds, and are useful for detection borders of target objects, in both horizontal (*Type 2(a)*) and vertical (*Type 2(b)*) directions. These trackers are able to detect both the target and the background, and in combination with the type 1 trackers, are able to localize the target outline as well as the centroid. Figure 3 shows these three different types of weak trackers.

In practice, the individual blocks in the trackers calculate the average color intensity in a square neighborhood in the image, instead of just using the value of one pixel. Various combinations of these trackers make up the bank of weak classifiers we want to boost. We perform tracking on colored objects in different video sequences, using a large number of image frames. By using a combination of tracker types and threshold values (which lie in the real number space, between 0 and 1), a large number of weak trackers can be generated, creating a large space of learners for the Boosting process to choose from. Trackers that perform worse than random chance are rejected by AdaBoost, but the large number of available trackers ensures that a significant number of trackers still passes the weak learning criterion.

5 Experimental Setup

We describe the experimental setup used to validate our boosted tracking algorithm in this section. The training and the validation process is explained in detail, and we also describe in brief the hardware and software capabilities of our experimental platform, the Aqua amphibious legged robot.

5.1 The Aqua Robot

The Aqua family of robots are vision-guided semi-autonomous legged amphibious vehicles, capable of terrestrial and underwater operation. The robots are equipped with stereo cameras for visual sensing and navigation. They have six degrees of freedom, and is power autonomous. The legs enable the robots to operate amphibiously, so the vision systems can be used in terrestrial as well as underwater environments. The robots can be operated remotely, via a fiber optic tether connected to the robot, or without the tether, by visual cues using specially engineered markers [14]. The robots are equipped with two computers; one for

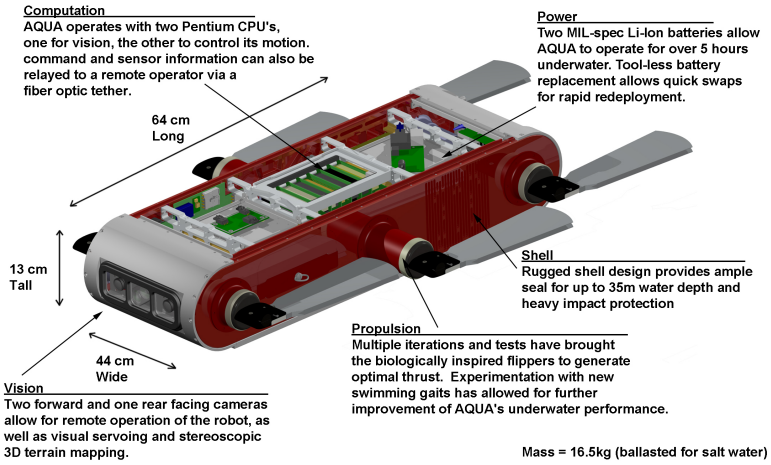


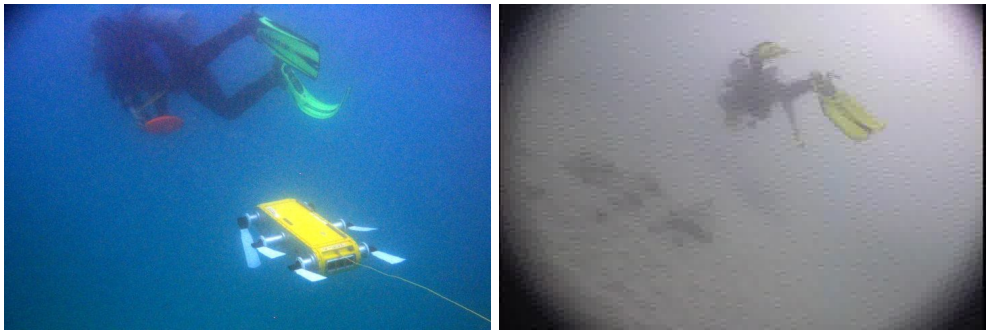
Fig. 4. A cutaway diagram of the Aqua amphibious robot.

real-time control and the other for visual and other sensory processing. Control stabilization is maintained by using an inertial navigation system (IMU). Figure 4 shows a cutaway section of the robot, with key components labeled.

The control computer runs on a real-time operating system, enforcing the hard real-time constraints on the control loop. The vision computer runs a custom version of the Linux operating system, created specifically for the purpose of on-board visual processing for the Aqua robots. The visual processing code is written in C++ and runs soft real-time on-board the robot.

5.2 Training

For training our system, we use video sequences containing different target objects in an aquatic environments, approximately 2000 frames for each target. The color channel trackers are trained on these images using the AdaBoost algorithm and the weights are obtained. We use our own implementation of AdaBoost written in C++, with the VXL vision library for image processing support. The training process is designed with parallel execution support, reducing the overall training time on supported hardware. Ground truth for the training data is supplied as an uncompressed plain text file. The ground truth data contains target boundaries in (x, y) coordinates of the top left and bottom right corners as input data (*i.e.* X) and as output $Y \in \{-1, +1\}$, where -1 indicates the target is not in the frame. In such cases, the target rectangle coordinates are set to negative values. The ground truth data is manually obtained, by grabbing frames from video files, inspecting them and setting the bounds by hand. The training method works as follows: each tracker outputs a target center (x, y) coordinate. If the target location falls within the boundary of the target region in the training data for a given instance, or is within $\pm 10\%$ of the original boundary size, we assume an output of $+1$. Any other output location is treated as a misclassification. Based on this training logic, the AdaBoost algorithm trains the system for either 2000 rounds, or until the difference in improvement in the training error is less than $\epsilon = 0.01$.



(a) External view: Robot tracking flippers. (b) Robot camera view.

Fig. 5. The Aqua robot tracking scuba diver's flippers.

5.3 Tracking Experiments

We have tested our boosted tracker on video sequences containing a target object in an aquatic environment (Fig. 5). Validation data is supplied using the same format as the training data, containing limits of the target object in a plain text file. The tracking algorithm is run on the validation set and the output is compared to the ground truth. For real-time servo control, we define the error measures for both the X and Y axes as the difference between the tracker output and center of the image frame (the set-point). The goal for visual servo control is to minimize these errors. The error values are fed into two Proportional-Integral-Derivative (PID) controllers, one for each axis of motion, which generate pitch and yaw commands for the robot controller. The robot controller accepts pitch and yaw commands from the PID controller and adjusts the leg positions accordingly to achieve the desired robot pose.

5.4 Experimental Snapshots

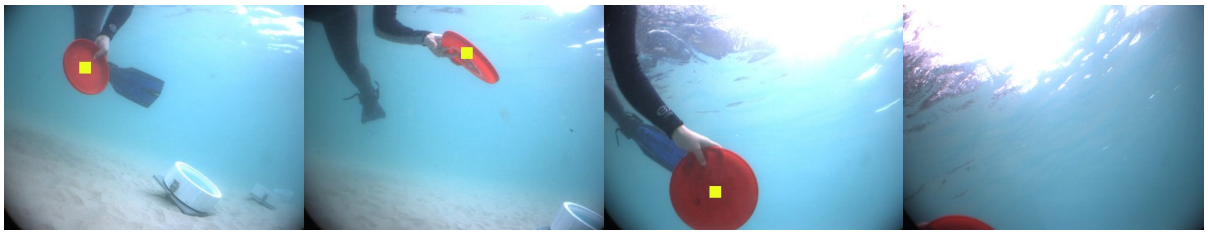
We have evaluated the boosted tracker using several different data sets with different targets, both synthetic and natural images. The performance of the boosted tracker is compared against the non-boosted color segmentation ("blob") tracker, in terms of tracking accuracy. The natural video sequences have been acquired from field trials of the Aqua robot in the Caribbean sea, near Barbados. To demonstrate the performance improvement achieved by using the Boosting process, we present two short sequences of underwater tracking data. In the example sequence shown in Fig. 6, the non-boosted color blob tracker fails to lock on to the target, as seen in the fourth image of the sequence, possibly due to change in illumination. Figure 7 shows the output of the boosted tracker on the same sequence, and it has successfully found the target in the fourth frame.

Figures 9 and 10 show examples of the robot tracking the flippers of the scuba diver. While the non-boosted blob tracker performs reasonably well to track the flippers (Fig. 9), the boosted tracker has a better accuracy as demonstrated in Fig. 10.

We record the accuracy rate for both the boosted tracker and the color segmentation tracker, and also record the number of false positives detected by the trackers. The results are shown in Fig. 8(a) and Fig. 8(b), respectively.

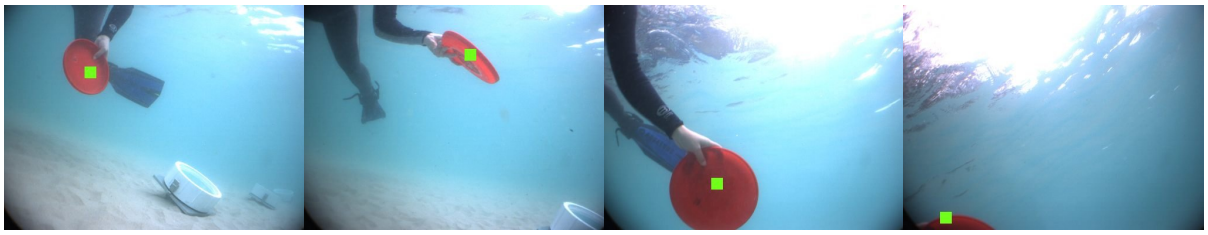
5.5 Tracking results

We use three targets of different colors to quantitatively evaluate the boosted tracker. As shown in Fig. 8(a), the boosted tracker performs better than the standard segmentation-based algorithm in tracking each type of colored target. In Figure 8(b), we see that the boosted tracker is detecting smaller number of false positives for the red and green target (data sets 1 and 2, respectively, in the plots), but more for the yellow. One



(a) Frame 1. (b) Frame 2. (c) Frame 3. (d) Frame 4.

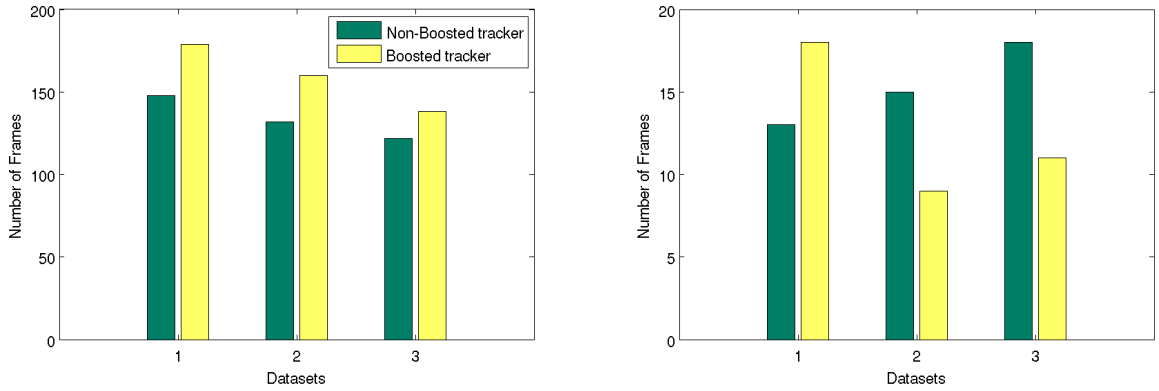
Fig. 6. Output of the non-boosted color segmentation tracker on an image sequence, shown as the yellow square. The target in the last frame is missed by the non-boosted tracker.



(a) Frame 1. (b) Frame 2. (c) Frame 3. (d) Frame 4.

Fig. 7. Output of boosted color segmentation tracker on an image sequence, shown as the green square. The target in the last frame is picked up by the boosted tracker.

possible explanation for this error can be the similarity in color of the surface reflection to the color of the target being tracked. In a significant number of frames, the robot actually is directed upwards, therefore catching the surface and the light falling on it. The surface lighting effect exhibits similar characteristics as a yellow target and is classified as a false positive output.



(a) Frames with correct tracking. (b) Frames tracking false positives.

Fig. 8. Tracking results of the boosted tracker showing accuracy(left) and false positives(right) compared to the non-boosted tracker. The legends have been omitted from the right plot to avoid data occlusion.

(a) Frame 1. (b) Frame 2. (c) Frame 3. (d) Frame 4.

Fig. 9. Output of unboosted color segmentation tracker on a video sequence of a diver swimming, shown by the yellow squares.

(a) Frame 1. (b) Frame 2. (c) Frame 3. (d) Frame 4.

Fig. 10. Output of boosted color segmentation tracker of the diver tracking sequence, shown by the white squares.

6 Conclusion

We have presented a visual tracking algorithm based on AdaBoost, for robust visual servoing of an underwater robot. Results achieved till date has shown promising improvement in tracking performance, and the computational complexity is within bounds of real-time performance in the underwater domain. Currently work is underway to integrate the boosted tracker in the robot and perform open-water validation tests of the visual tracking system.

As further enhancements to the current work, we are investigating different trackers to integrate in the collection of weak trackers for the Boosting process. Specific tracker types may perform better in particular environments, and we are looking at ways to automate this process of choosing the right tracker for the task. We are also investigating on-line Boosting for adaptively retraining the tracker to dynamically improve robustness. Finally, while using AdaBoost for classification gives us good results, it remains to be seen whether using it in a regression environment provides any real benefits, at the expense of additional computation cost incurred at the training stage.

References

1. K. Arbter, J. Langwald, G. Hirzinger, G. Wei, and P. Wunsch. Proven techniques for robust visual servo control. In *IEEE International Conference on Robotics and Automation, Workshop WS2, Robust Vision for Vision-Based Control of Motion*, pages 1–13, 1998.
2. Shai Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(2):261–271, February 2007.
3. S. Chernova and M. Veloso. An evolutionary approach to gait learning for four-legged robots. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 3:2562–2567 vol.3, 28 Sept.-2 Oct. 2004.
4. Gregory Dudek, Michael Jenkin, Chris Prahacs, Andrew Hogue, Junaed Sattar, Philippe Giguère, Andrew German, Hui Liu, Shane Saunderson, Arlene Ripsman, Saul Simhon, Luiz Abril Torres-Mendez, Evangelos Milios,

Pifu Zhang, and Ioannis Rekleitis. A visually guided swimming robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, Canada, August 2005.

5. Yoav Freund. Boosting a weak learning algorithm by majority. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1990.
6. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.
7. Philippe Giguere and Gregory Dudek. Clustering sensor data for terrain identification using a windowless algorithm. In *Robotics: Science and Systems*. The MIT Press, 2008. In press.
8. S. A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.
9. Yi Ma, Jana Kosecka, and Shankar Sastry. Vision guided navigation for a nonholonomic mobile robot. In *IEEE Conference on Decision and Control*, 1997.
10. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
11. Robert E. Schapire. A brief introduction to boosting. In *International Joint Conference on Artificial Intelligence*, 1999.
12. S. Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33(1):41–76, 1998.
13. Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
14. Anqi Xu, Gregory Dudek, and Junaed Sattar. A natural gesture interface for operating robotic systems. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, Pasadena, California, May 2008.
15. Richard Y. D. Xu, John G. Allen, and Jesse S. Jin. Robust mean-shift tracking with extended fast colour thresholding. In *International Symposium on Intelligent Multimedia, Video and Speech Processing (ISIMP2004)*, pages 542–545, Hong Kong, October 2004.