

Field Programmable Gate Array (FPGA) Based Fish Detection Using Haar Classifiers

Bridget Benson, Junguk Cho, Deborah Goshorn, Ryan Kastner

Computer Science and Engineering Department, University of California San Diego, 9500 Gilman Drive, La Jolla CA 92092
b1benson@cs.ucsd.edu

Abstract

The quantification of abundance, size, and distribution of fish is critical to properly manage and protect marine ecosystems and regulate marine fisheries. Currently, fish surveys are conducted using fish tagging, scientific diving, and/or capture and release methods (i.e., net trawls), methods that are both costly and time consuming. Therefore, providing an automated way to conduct fish surveys could provide a real benefit to marine managers. In order to provide automated fish counts and classification we propose an automated fish species classification system using computer vision. This computer vision system can count and classify fish found in underwater video images using a classification method known as Haar classification. We have partnered with the Birch Aquarium to obtain underwater images of a variety of fish species, and present in this paper the implementation of our vision system and its detection results for our first test species, the Scythe Butterfly fish, subject of the Birch Aquarium logo.

Keywords: automated fish classification, Haar classifiers, FPGAs

Introduction

The quantification of abundance, size, and distribution of fish is critical to properly manage and protect marine ecosystems and regulate marine fisheries. Currently, fish surveys are conducted using fish tagging, scientific diving, and/or capture and release methods (i.e., net trawls). All of these methods require many man hours and ship time which is costly and time consuming. Therefore, providing an automated way to conduct fish surveys could provide a real benefit to marine managers.

Automated fish surveys could be conducted using computer vision, where a computer can process underwater video images, counting and classifying fish species of interest. Some work on automatic fish classification using computer vision has already been done. These fish classification methods are based on shape (Cadieux et al., 2000; Tillett et al., 2000; Lee et al., 2003; Lee et al., 2004,) texture (Rova et al., 2007), or color (Strachan, 1993; Semani et al., 2002; Chambah et al., 2004) and are used for various applications such as guiding fisheries management, evaluating the ecological impact of dams, managing commercial fish farms, improving fish migration monitoring or enabling educational interactive displays for aquarium visitors. More recent methods focus on object detection to enhance the productivity of human annotators (Cline et al., 2008). However, none of these methods provide a framework for rapidly classifying 'any' fish species of interest.

Therefore, to provide a framework that can classify 'any' fish species of interest rapidly in parallel, we have designed an automatic fish detection system based on the Viola and Jones Haar-like feature object detection method on a field programmable gate array (FPGA). Haar-like features can be

applied to any fish species of interest, and FPGAs are well known for their ability to process computations rapidly in parallel.

In this paper we describe our method for generating fish classifiers as input to our FPGA framework and describe how the FPGA framework uses these classifiers to perform real-time fish detection. We present our detection results for our first test species, the Scythe Butterfly fish, and report the potential performance capabilities of our framework based on face detection experiments. We conclude with a discussion on future work.

Methods

This section describes the method we use to generate Haar classifiers for our FPGA framework and describes how the FPGA framework uses these classifiers to perform real-time fish detection.

Generating Haar Classifiers

Our method to generate Haar classifiers for different fish species makes use of OpenCV's (an open source computer vision library) Haar Training Code based on the Viola-Jones detection method (Viola and Jones, 2004). This code allows a user to generate a Haar classifier for any object that is consistently textured and mostly rigid (Bradsy and Kaehler, 2008). A good classifier only needs to be generated once and then can be loaded into a classification system whenever that object of interest needs detection. OpenCV's Haar Training technique has been successful for face (Viola and Jones, 2004; Cho et al., 2008), car (Hakan, 2005), and pedestrian (Montiero et al., 2006) detection and we now extend this method to fish detection.

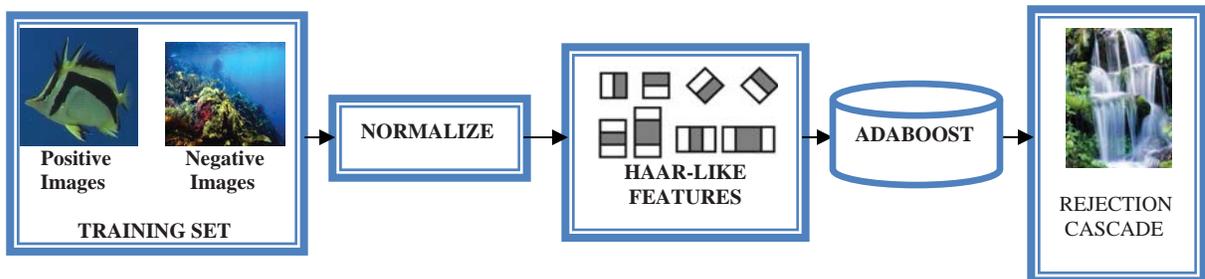


Figure 1. Haar Training Procedure

Figure 1 shows OpenCV's Haar Training procedure. Haar Training takes in a set of positive and negative images as inputs – the positive images containing cropped images of the fish species of interest and the negative images containing images of the fish's environment and images of fish of other species. These images are then converted to gray-scale and normalized to a user-specified window size (in our case, 20 x 20 pixels). The training code calculates Haar-like features (sums and differences of 2-3 rectangular image regions denoted by the (x,y) coordinator of their upper left corner and their width and height) within the normalized images to find the Haar-like features that best distinguish between the positive and negative samples. Figure 2 shows an example of a Haar-like feature used for detection of the Scythe Butterfly fish. It consists of 2 rectangles (the white rectangle interpreted as "add that area" and the dark rectangle interpreted as "subtract that area").



Figure 2. Example of a 2-rectangle Haar-like feature used for detection of the Scythe Butterfly fish

Once the Haar-like features have been calculated, the Haar Training procedure uses a form of AdaBoost (Freund and Schapire, 1997; Viola and Jones, 2004) to organize these features into a 'rejection cascade' consisting of n stages as shown in Figure 3. The stages are ordered from least to most complex so that computations will be minimized (simple stages are tried first) when rejecting non-fish regions of a test image. A fish is only detected if the Fish candidate passes through all stages of the rejection cascade.

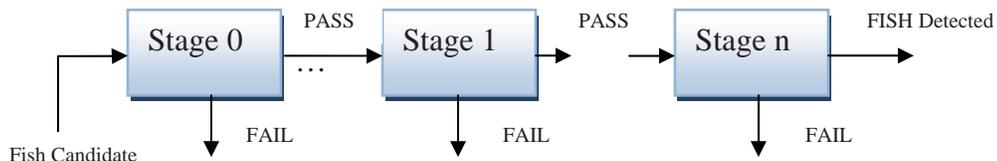


Figure 3. Rejection Cascade

Each stage consists of an Alternating Decision Tree (ADTree) (Freund and Mason, 1999) that represents an AdaBoosted set of m features. Figure 4 shows an example of an ADTree for one stage.

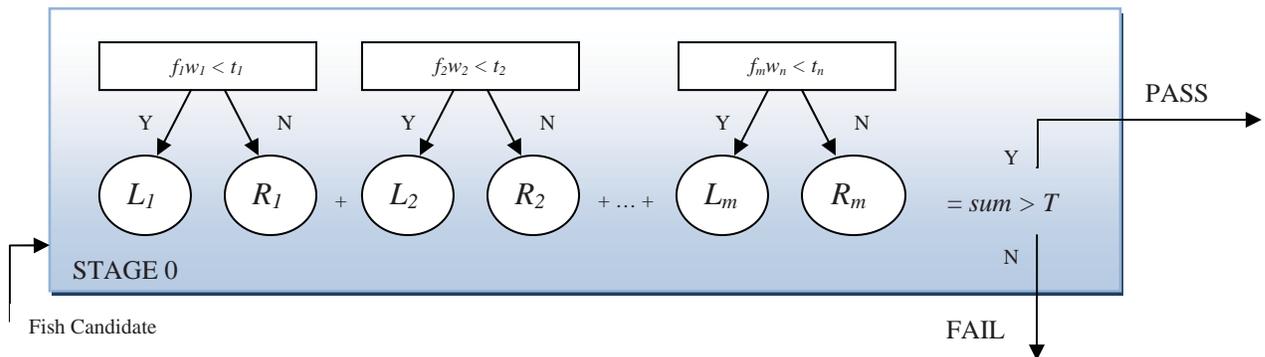


Figure 4. An Alternating Decision Tree (ADTree) representation of one stage of the Rejection Cascade

For each stage, AdaBoost selects m features f_i , feature weights, w_i , feature thresholds, t_i , left values, L_i , right values, R_i , and the stage threshold T that achieve a user specified hit rate of 99% and false positive rate of 50% when all training images are applied to the tree. A training or test image passes the stage if the sum of its selected left or right values, (depending on the outcome of the feature threshold inequality) sum , is greater than the stage threshold T .

OpenCV represents the rejection cascade, the resulting Haar Classifier, as an xml file that contains the features, feature weights, feature thresholds, left values, right values, and a stage threshold for each

stage in the cascade. This Haar Classifier achieves an excellent overall hit rate of 0.99ⁿ and false alarm rate of 0.5ⁿ on the training data. We use the generated xml file as input to our FPGA framework to provide real-time fish detection (described in the next subsection).

FPGA Framework

An FPGA is a semiconductor device that can be configured by the designer after manufacturing – hence the name "field programmable." FPGAs can be programmed to perform an application specific task (such as fish classification) through using a hardware description language to specify how the chip will work. Because the designer can create dedicated hardware to perform a specific task and can create duplicate sets of the same hardware to perform operations in parallel, FPGAs often offer higher performance than a software solution.

We implemented an FPGA framework for fish detection based on the Viola and Jones object detection method on a Xilinx Virtex-5 FPGA. Figure 5 shows the overview of this framework.

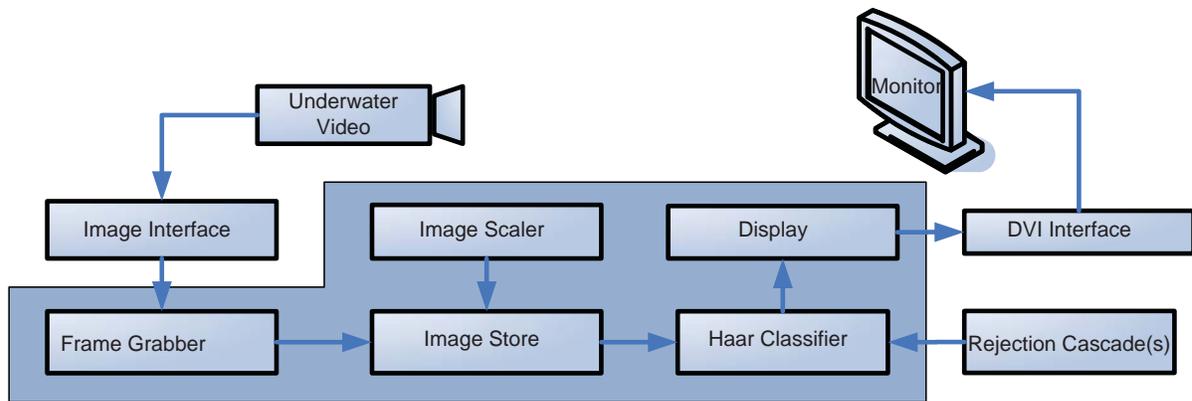


Figure 5. FPGA Framework

The underwater video is fed to the FPGA framework through an image interface which digitizes the analog image. The frame grabber module then grabs frames from the video to be processed individually. The image store module stores the image data arriving from the frame grabber module and transfers the image data to the Haar Classifier module based on the scale information from the image scaler module. The image scaler module scales the frame to various sizes to provide the capability for the framework to detect fish of various sizes within the video image. The Haar Classifier module performs the classification for the fish detection on each scaled frame based on the Haar Classifier Rejection Cascade generated as described in the previous subsection. The Haar Classifier module is the critical module of the whole fish detection system and thus will be described in more detail below. The display module stores the information of the detected fishes and displays white squares on the detected fish in the image sequence. The processed image is displayed on a monitor through the DVI interface.

The Haar Classifier module begins by computing the integral image of the input scaled frame. The integral image is the summation of the pixel values of the original image. The value at any location (x,y) of the integral image is the sum of the original image's pixels above and to the left of location (x,y) . Figure 6 illustrates the integral image generation.

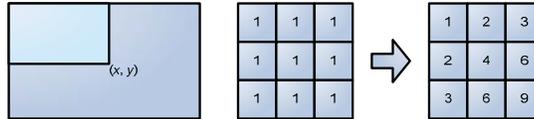


Figure 6. Integral Image Generation

The integral image allows for rapid computation (constant time) of the Haar-like features. By using each corner for a Haar-like feature rectangle, the area of the rectangle can be computed quickly as shown in Figure 7. The area of the rectangle R can be computed using the integral image values in the positions L_1 , L_2 , L_3 , and L_4 as $L_4 - L_3 - L_2 + L_1$. Since the area of L_1 is subtracted off twice by areas L_2 and L_4 , it is added back on to get the correct area of the rectangle.

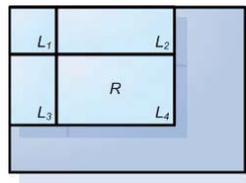


Figure 7. Calculating the area of a rectangle R using the integral image values as $L_4 - L_3 - L_2 + L_1$

Once the integral image has been computed, the Haar Classifier module scans the integral image by a 20x20 pixel window (a fish candidate), searching for a fish. Each fish candidate goes through the rejection cascade. Thus the Haar-like features in Stage0 are computed (using the integral image calculation), weighted and compared to the feature threshold (shown in Figure 4). If the sum of the selected left and right values, sum , is less than the stage threshold, T , the fish candidate is rejected and the Haar classifier module moves the scanning window over one pixel to process the next 20x20 fish candidate. If the sum is greater than the stage threshold, T , the fish candidate passes Stage0 and continues to Stage1, 2, ..., n until the fish candidate either fails at a stage or passes all stages. A fish candidate is only marked as containing a fish if the fish candidate passes all stages in the rejection cascade. To provide the ability to locate multiple fish in one frame, the Haar Classifier module processes the next scaled frame only after the entire integral image has been scanned. By processing frames of various scales, fish of various scales can be detected.

Results

Using the methods described in the previous section, we generated a 16 stage Haar classifier with 83 features for the Scythe Butterfly fish using 1077 positive images and 2470 negative images. All positive images were obtained by grabbing frames of a video of one Scythe Butterfly fish with a blank blue background (as shown in Figure 8). All positive images contained the fish's side profile while swimming left as earlier work has shown a separate classifier is needed for different profiles of the same object (Horton et al., personal communication). The negative images consisted of various gray scale underwater scenes of the Scythe Butterfly's natural environment.



Figure 8. Frame of Scythe Butterfly video used to generated one positive image

Using the OpenCV performance metric and 100 test images (also obtained from the Scythe Butterfly video), we obtained 92 hits, eight misses and three false positives with the generated 16 stage classifier. We were unable to obtain hit/miss results from the FPGA Framework at the time of this publication as OpenCV includes tilted Haar features in its rejection cascade that are currently not handled by our FPGA framework.

Thus to obtain performance measurements of our FPGA framework, we used the OpenCV frontal face rejection cascade that consists of 22 stages and 2135 features (and no tilted features) (Bradski and Kaehler, 2008; OpenCV, 2008) and tested the performance of our framework for face detection on five images containing faces. Since the system performance of face detection depends on the number of faces in the images and the size of the image, we scaled the test images to two different sizes (320 x 240 and 640 x480) with the five test images of each size containing 1, 3, 6, 9, and 12 faces respectively. Table 1 shows the average performance of the face detection system with varying levels of parallelism. The level of parallelism denotes how many Haar-features can be computed simultaneously within the framework. For a detailed description of the parallel implementations (Cho, personal communication). The table shows the performance improvement of using the FPGA framework over an equivalent software solution (written in C++) on a PC; in this case using an Intel Core 2 Quad CPU (2.4 GHz), 8 GB DDR2 SDRAM (800MHz), Microsoft Windows Vista Business (64-bit), and Microsoft Visual Studio. The table shows the face detection system on the FPGA framework has the performance improvement of up to 84.5 times the software solution with the 320x240 resolution images and up to 37.39 times the software solution with the 640x480 resolution images. All faces in the five images of each image size were accurately detected with zero false positives.

Table 1. Performance of FPGA Framework for Frontal Face Detection

Level of Parallelism	320×240 images	Improvement	640×480 images	Improvement
S/W 1	1,373ms (0.72 fps)	1.00	2,319 ms (0.43 fps)	1.00
FPGA 1	54.735 ms (18.26 fps)	25.36	190.541 ms (5.24 fps)	12.18
FPGA 2	38.997 ms (25.64 fps)	35.61	146.033 ms (6.84 fps)	15.90
FPGA 4	24.405 ms (40.97 fps)	56.90	81.499 ms (12.27 fps)	25.20
FPGA 6	21.053 ms (47.49 fps)	65.95	62.154 ms (16.08 fps)	28.53
FPGA 8	16.387 ms (61.02 fps)	84.75	62.154 ms (16.08 fps)	37.39

Discussion

The hit/miss rate results of the Scythe Butterfly fish Haar Classifier and the performance results of the FPGA framework on face detection are encouraging. The results provide evidence that the FPGA

based fish detection system we described may provide a good way to perform rapid fish classification in underwater images. However, a large amount of future work must be performed to evaluate the effectiveness of this method on fish detection and the improvement it may provide over other existing fish classification methods mentioned in the introduction.

Future work includes modifying the OpenCV Haar Training code to generate Haar classifiers without titled features so that the classifier can be tested on our FPGA framework, obtaining images of the Scythe Butterfly fish in a natural background to see if the framework can accurately pick out the fish from the natural background, obtaining images of the Scythe Butterfly fish with other species of fish to see if the framework can accurately distinguish the Scythe Butterfly in the presence of other species, and generating classifiers for other profiles of the fish so that it can be detected in more than one position. If all of these future tests prove successful, we intend to develop a method or tool to allow scientists to generate their own classifiers of different fish species that can be added to a large database of classifiers and used in the FPGA framework for rapid detection of the scientist's species of interest.

Acknowledgments

The authors acknowledge the support of Birch Aquarium for providing training data for the classifier and to Boris Babenko for his assistance with the training process. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

Ardo H. Learning Based System for Detection and Tracking of Vehicles. In: Image Analysis. Berlin, Germany: Springer-Verlag, 2005: 449-58.

Bradski G, Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. Cambridge, MA: O'Reilly Media, Inc.; 2008.

Cadieux S, Lalonde F, Michaud F. Intelligent system for automated fish sorting and counting. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems; 2000 Oct 30 – Nov 5; Takamatsu, Japan.

Chambah M, Semani D, Renouf A, Courtellemont P, Rizzi A. Underwater color constancy : enhancement of automatic live fish recognition. Proceedings of SPIE / IS&T Electronic Imaging; 2004 Jan 18-22; San Jose, California, USA.

Cho J, Kastner R, Mirzaei S, Oberg J. FPGA-based face detection system using haar classifiers. Proceedings of International Symposium on Field Programmable Gate Arrays (FPGA); 2009 Feb 22-24; Monterey, California, USA.

Cline DE, Edgington DR, Mariette J. An automated visual event detection system for cabled observatory video. Proceedings of the 3rd International Conference on Computer Vision Theory and Applications; 2008 Jan 22-25; Funchal, Madeira, Portugal.

Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 1997;55:119-39.

Freund Y, Mason L. The alternating decision tree algorithm. Proceedings of the 16th International Conference on Machine Learning; 1999 June 27-30; Bled, Slovenia.

Lee DJ, Redd S, Schoenberger R, Xu X, Zhan P. An automated fish species classification and migration monitoring system. Proceedings of the 29th Annual Conference of the IEEE Industrial Electronics Society; 2003 Nov 2-6; Roanoke, Virginia, USA.

Lee DJ, Schoenberger R, Shiozawa D, Xu X, Zhan P. Contour matching for a fish recognition and migration monitoring system. Proceedings of the SPIE optics east, Two and Three-Dimensional Vision Systems for Inspection, Control, and Metrology II; 2004 Oct 25-28; Philadelphia, PA, USA.

Monteiro G, Peixoto P, Nunes U. Vision-based pedestrian detection using Haar-like features. Proceedings of the 6th National Festival of Robotics; 2006 April 29 – May 1; Guimarães, Portugal.

Morais ER, Campos M, Padua FL, Carceroni, R. Particle filter-based predictive tracking for robust fish counting. Proceedings of the 18th Brazilian Symposium on Computer Graphics and Image Processing; 2005 Oct 9-12; Natal, RN, Brazil.

Rova A, Mori G, Dill LM. One fish, two fish, butterfly, trumpeter: Recognizing fish in underwater video. Proceedings of the IAPR Conference on Machine Vision Applications; 2007 May 16-18; Tokyo, Japan.

Semani D, Bouwmans T, Frélicot C, Courtellemont P. Automatic fish recognition in interactive live video. Proceedings of the International Workshop on IVRCIA in the 6th World Multi-Conference on Systemics, Cybernetics, and Informatics; 2002 July 10-12; Orlando, FL, USA.

Strachan NJC. Recognition of fish species by color and shape. Image Vision Computing 1993;11(1):2-10.

Tillett R, McFarlane N, Lines, J. Estimating dimensions of free-swimming fish using 3D point distribution models. Computer Vision and Image Understanding 2000;79(1):123-41.

Viola P, Jones M. Robust real-time object Detection. International Journal of Computer Vision 2004; 57(2):137-154.