**Comparing the Effectiveness of Heuristic Algorithms and Support Vector Machine Learning Algorithms to Improve Upon a Haar Classifier**

Ingrid Rumbaugh, [rumbaugi@lafayette.edu](mailto:rumbaugi@lafayette.edu)

Under the Advisement of Dr. Alex Brown, [brownaa@lafayette.edu](mailto:brownaa@lafayette.edu)

GitHub Repository: https://github.com/ingridrumbaugh/ComputerVision

*Department of Mechanical Engineering*

*Lafayette College, Easton, Pennsylvania 18042*

**Abstract Here.**

1. **Introduction**

The original intent of this paper was to compare a heuristic algorithm to a more complex machine learning algorithm utilizing Haar classifiers, in order to identify and track Archerfish. This research is part of a larger project intended to investigate whether or not a robot designed to mimic an Archerfish can train other Archerfish to shoot at a target. Simplifying the process, and therefore decreasing the computing time, for identifying Haar-like features is valuable for the data collection and validation of this larger research project. This paper will highlight the path that this research project followed, in addition to discussing how the chosen algorithms compare to previous work.

1. **The Process**

The purpose of this section is to provide a roadmap of the topics investigated in this paper. Further sections will go into more detail regarding background, specific coding methods, and results. The figure below illustrates the investigative process taken throughout the Spring 2018 semester. Since the Archerfish are distinct enough against the background (the water in the tank), it was thought that using histogram comparisons would be an effective way to find the Archerfish. First, a combination of edge detection and background subtraction were used before implementing histograms. The intent of this first step was to eliminate any object that isn’t moving – effectively making the Archerfish contrast with a black background. Then, histograms of a single video frame were compared to a “ground truth” histogram taken of an Archerfish. This method is called histogram intersection and will be described in the next section.

Moving on to the third step in Figure 1, a transition was made from simple histograms to histograms of oriented gradients (HOG). This was the next logical step, given that HOG is frequently used in image processing to find edges of irregular shapes, such as the Archerfish that are trying to be identified. In the implementation used for this project, a HOG was found using the Sobel Operator, which seemed to be a better fit for the images being manipulated.

Here, HOG is used as a stepping stone into Support Vector Machines (SVMs). This is a type of linear or non-linear machine learning algorithm that can be used to differentiate objects. Although the intent of this research was to investigate replacing such machine learning algorithms, using one would enable this project to differentiate Archerfish not only from the background of the tank, but other fish as well.

|  |
| --- |
|  |

**Figure 1.** Progression of Algorithms Investigated.

1. **Background: Heuristic Algorithms**

In the interest of thoroughness, the background for the basis of this research will be explored – beginning with Haar-like features. Haar-like features are simply digital image features used in object recognition. A Haar feature considers “adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region, and calculates the difference between these sums.” [1]. While there are many types of Haar-like feature descriptors, an example of one can be seen in Figure 2.

|  |
| --- |
|  |

**Figure 2.** Example of a Haar-like feature descriptor [2].

However, a Haar classifier can take a lot of time to train and can quickly become a very complex process. Improving upon this method by finding a simpler way to identify Archerfish is very valuable.

As shown in Figure 1, the first method that was tested was background subtraction and edge detection. Python’s OpenCV2 was one of the main libraries used in this project due to it’s versatility. The *BackgroundSubtractorMOG2* method was used to help eliminate all objects in the frame that were not moving. The *BackgroundSubtractorMOG2* method is a “Gaussian Mixture-based background segmentation algorithm” [3]. An important feature of this algorithm is that it selects the appropriate number of gaussian distribution for each pixel. This provides better adaptability to varying scenes due to features like illumination changes [3]. ­

Once the background is removed, histogram intersection is implemented. This was chosen because the Archerfish do not blend into the background. The intersection algorithm was used from the GitHub repository in Reference 4. The intersection value found is the number of pixels from the model that have corresponding pixels of the same colors in the input image [4]. This intersection is then normalized between 0 and 1 by dividing it by the number of pixels in the histogram. This means that in the results section, a “perfect match” histogram would have an intersection value = 1. A visual representation of intersecting histograms can be seen in Figure 3 [4].

|  |
| --- |
|  |

**Figure 4.** Example of Histogram Intersection [4].

1. **Histogram Intersection Methods – a bit of code and the process**

Throughout all of the methods for testing this heuristic algorithm, a “ground truth” histogram was created using an image of an Archerfish taken from the current video frame. In later methods in an effort to increase reliability, three separate ground truth histograms were computed from three separate Archerfish. One of the ground truth histograms can be seen below in Figure 5.

|  |
| --- |
|  |

**Figure 5.** Ground Truth Histogram of Archerfish.

Once a histogram is taken of a section in the frame, it is overlapped with the ground truth histogram. A snippet of code in Figure 6 shows the histogram intersection method used in this algorithm. Again, this method was inspired by code from the repository in reference 4.

|  |
| --- |
| **Figure 6.** Histogram Intersection Method. |

1. The first approach to implementing this heuristic algorithm was to sweep a rectangle of size 40 x 40 pixels, iterated in steps of 5 over the frame. To test the functionality of this program with video, a webcam was used. This program was then implemented with one frame from a video of Archerfish in a tank. However, problems arose when running this program with the bag files of Archerfish swimming. The movement of the fish was not enough to yield significant results from intersecting histograms.
2. The second approach was to repeat the process from before, but without removing the background of the frame. As stated previously, a perfect match would yield an intersection value of 1. However, since this is unrealistic to expect this, intersection values were tested from 0.5 through 0.8, iterating in steps of 0.05.
3. The final approach was to use three ground truth histograms with the intent of allowing a higher rate of ‘matches’ to fish. The hope was that this would enable the algorithm to identify fish in multiple orientations, instead of simply a fish facing towards the right in a perfectly identifiable pose. This method required that the test histogram matched at least two of the three histograms, with an intersection value that was iterated the same as in method 2.
4. **Heuristic Algorithm Results**
5. Background Subtraction + 1 Ground Truth Histogram

This method did not yield significant results when implemented with the Archerfish tank videos. In addition, when this program was run through ROS (Robot Operating System), it was unexpectedly slow. Even though this method was never able to be fully tested, some intermediate results are shown below to illustrate the overall concept. These figures are from webcam testing and show the pre-processed, threshold frame, the final frame with the background subtracted, and the active histogram of the frame. These can be seen in Figures 7, 8, and 9.

|  |
| --- |
|  |

**Figure 7.** Threshold Frame.

|  |
| --- |
|  |

**Figure 8.** Final Frame with Background Subtraction.

|  |
| --- |
|  |

**Figure 9.** Active Histogram of Webcam Frame.

1. 1 Ground Truth Histogram without Background Subtraction

­This method was tested against a single video frame with increasing levels of histogram intersection. Figure 10 shows the fish (taken directly from the frame that this algorithm was tested on) that was used for the ground truth histogram.

|  |
| --- |
|  |

**Figure 10.** Fish Used for Ground Truth Histogram.

A selection of intersection values from this method are shown below. This selection shows how the algorithm performed with an increasing value of histogram intersection. When the algorithm has identified something as an Archerfish, it prints a green rectangle over the block of pixels. Ideally, the algorithm would find what it thinks is multiple fish and print multiple rectangles over a single fish. This could eventually be mitigated by replacing these overlapping rectangles with one rectangle, that would ideally be an Archerfish.

|  |
| --- |
|  |

**Figure 11.** Algorithm Run on Frame with Required Intersection = 0.6.

|  |
| --- |
|  |

**Figure 12.** Algorithm Run on Frame with Required Intersection = 0.7.

|  |
| --- |
|  |

**Figure 13.** Algorithm Run on Frame with Required Intersection = 0.8.

|  |
| --- |
|  |

**Figure 14.** Algorithm Run on Frame with Required Intersection = 0.85.

**Discuss results here.**

1. 3 Ground Truth Histograms

This final heuristic method was again tested against a single video frame with increasing levels of histogram intersection. Figure 15 shows the three fish (taken directly from the frame that this algorithm was tested on) that were used for the ground truth histograms.

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Figure 15.** Archerfish used for Ground Truth Histograms.

1. **Background: Histogram of Oriented Gradients (HOG) & Support Vector Machines (SVM)**
2. **HOG Methods**
3. **HOG and SVM Results**

* **The Process:**
  + Histogram Intersection with and without BS
  + Hist. Intersection without BS: Using 3 Ground Truth Histograms
  + Histogram of Oriented Gradients
    - Sobel Operator
    - Mag & Dir of gradient for each pixel (Goes into SVM)
  + Support Vector Machines
  + Creating a HOG descriptor for an SVM

1. **Discussion of Overall Research**
2. **Conclusions**
3. **References**

[1] <https://en.wikipedia.org/wiki/Haar-like_feature>

[2] <http://scikit-image.org/docs/dev/auto_examples/features_detection/plot_haar.html>

[3] <http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html>

[4] <https://mpatacchiola.github.io/blog/2016/11/12/the-simplest-classifier-histogram-intersection.html>