BU Memo Senior Design ENG EC 463

To: Alan Pisano, Osama Alshaykh, Babak Kia

From: John McCullough, Shantanu Bobhate, Brian Tan, Bandhan Zishanuzzaman, Teng Zhang

Team: 2 aka Right Alert

Date: 2/25/2016

Subject: Second Deliverable Test Plan

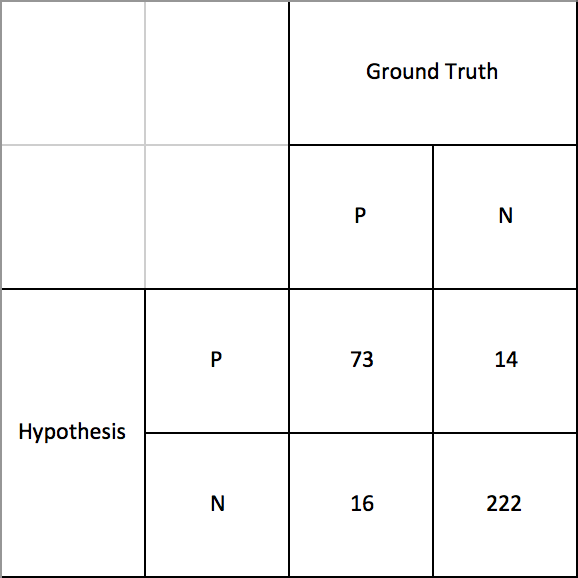
**1.0 Bike Detection System**

**1.1 Bike Detection Algorithm**

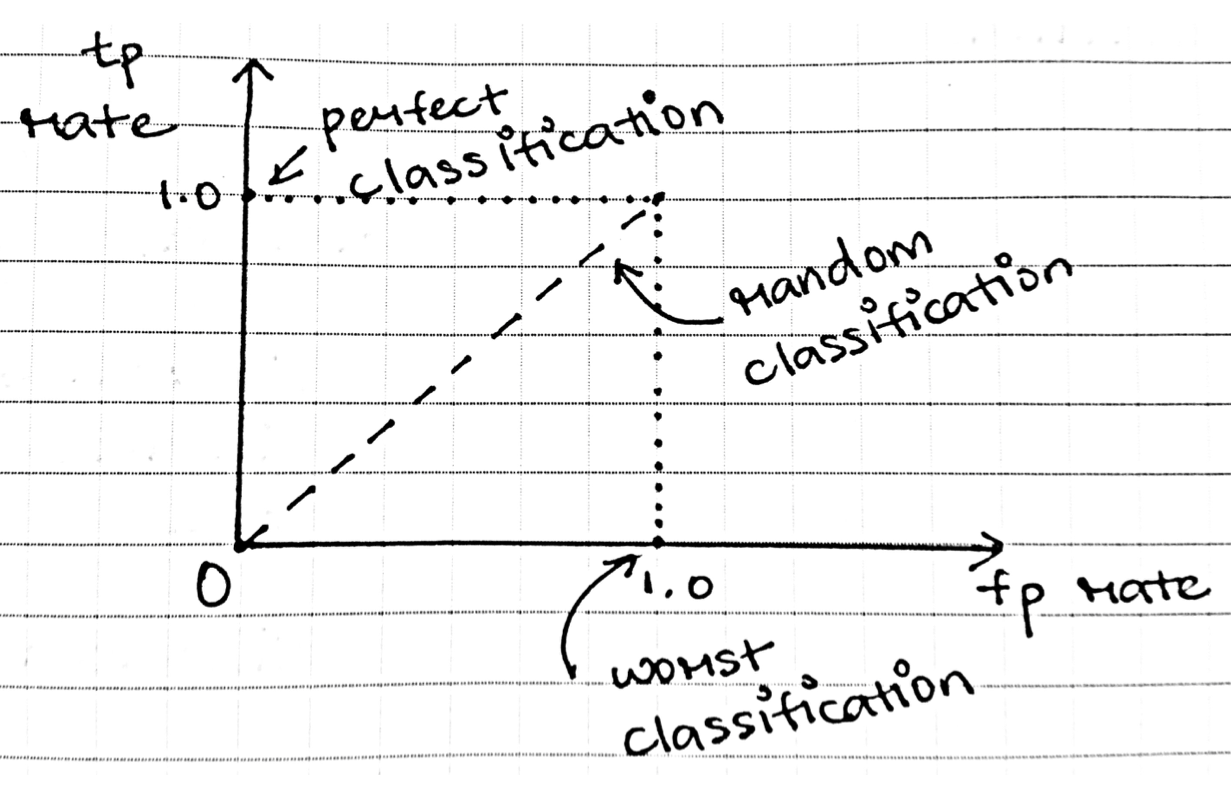
*Description and Goal*

The bike detection algorithm uses the HOG with SVM approach which is a Machine Learning Algorithm. The procedure trains an SVM classifier with positive and negative training images from the installation site (Cummington Mall). In order to provide an accurate detection rate, training images with rich differences are preferred. The negative images must focus on objects that could look very similar to bikers, eg: people, tree trunks.

Currently, the algorithm has been trained with ~70 positive and ~500 negative training images taken on Cummington Mall. In order to test the performance of this system we tested the predictions of this system for a second dataset we had collected on the same day. The confusion matrix below highlights our results.



These results yielded a true positive rate of 0.82 and a false positive rate of 0.06. This is a good result for an AI detection system.



For the demonstration, we plan to run our algorithm on a video of bikers riding down Cummington Mall and observe its accuracy. Afterwards we plan to train the algorithm with an additional set of training images and run the detection algorithm again, expecting to see improvements.

The goal of having a well trained algorithm is to minimize the rate of false negative and false positive readings of oncoming bikers. The lower these rates are, the safer an experience we can guarantee drivers and bikers using an intersection that implements RightAlert.

*Procedure*

To run the algorithm on the test video, first direct yourself to the directory *~/Desktop/Bike\ Detector\ 2.0\ for\ Jetson/2nd\ Deliverable* and run the executable *./hog*.

A GUI window guide the user through the setup.. You will be asked if you would like to train the HOG. We would like to do this in our second iteration, but for now we can skip this step by typing in “no.” Simply follow the rest of the instructions and wait until a window of the video appears.

The program will begin its detection by itself. When a green rectangle is painted around an area of the screen, the algorithm has detected something it believes is a biker. As this is happening, we will be keeping track of the algorithm’s accuracy on a spreadsheet. We will note when a biker has been successfully detected, not been detected at all, or when an object that is not a biker is detected. In addition to this data, we will use the Gnome System Monitor to keep a track of the system’s CPU and memory usage and make note of this to track its efficiency for later improvements. This will continue until the video is over.

The next part of our demonstration requires that we train the algorithm with a new set of positive and negative images. To do so, we rerun the same executable as before and this time select “yes” when asked whether to train the algorithm. The path to the training pictures are already identified in the code so we do not have to do any additional commands at the moment. After waiting for the training to complete, we will follow similar steps as before until we see the video appear in a window again. We will repeat the same procedure recording the accuracy of the detection.

*Verifiable Result*

For the initial test, we will determine a false positive and false negative rate for our bike detection. These values should provide us with assurance that our algorithm will be a safe tool for drivers to use. After the second test, with additional training, we should expect to see even lower false positive and false negative rates.

**1.2 Algorithm Interaction with Alert Sign LEDs**

*Description and Goal*

In order to simulate our regulated LED lit alert sign for the demonstration, we substituted a miniature mock-up alert sign with embedded LEDs. The LEDs are connected in parallel in a simple circuit where their positive ends are connected to pin50 of the Jetson and their negative ends are connected the GND on the Jetson. While the detection algorithm is observing the test video, the LEDs will turn on everytime it has made a detection of what it believes is a biker. This test is to ensure the Jetson’s GPIO interact harmoniously with the OpenCV code it is running. This is an incredibly important portion of the alert system because it will be the visual cue drivers will use to stop for oncoming bikers.

*Procedure*

Since the LEDs are already connected to the Jetson’s GPIO, we need only to observe them when the algorithm has made a detection. When a biker is detected, we should expect the LEDs to turn on. When a biker is no longer detected the LEDs should turn off.

*Verifiable Result*

We will get desired results if the LEDs turn on the same time a biker is detected in the video and if the turn off when there is no longer a detected biker.

**1.3 Algorithm Interaction with RightAlert Website**

*Description and Goal*

In order to provide daily traffic data on bikers to the public, the alert system will post time-stamps and the number of bikers to the RightAlert website. In order to do so, we have written code to manage this transfer of data from the Jetson to our website. The algorithm, written in C++ also contains C++ code to both turn on the LEDs and update the MySQL database. The RightAlert admin website, <http://admin.rightalert.net> displays a count of how many bikers have been detected by the system. The admin website functions are written with C# using the ASP.NET framework.

*Procedure*

Similar to the LEDs, the algorithm’s update to the RightAlert website should be automatic and rely on no input from the user. We will simply wait for the algorithm to make a detection and observe if a message was received by the website.

*Verifiable Result*

After the algorithm detects a bike, in addition to the LED lights, there is code that updates the MySQL database in the cloud. By visiting <http://rightalert.net/test> or <http://admin.rightalert.net> it can be confirmed that the database was updated with a record of a biker.

**2.0 RightAlert Website**

*Description and Goal*

Our website serves two purposes. The first is to display relevant data at every bicycle detection event. Data we are collecting will include a timestamp of when a biker passes an intersection, the bike speed, and other data obtained from the video processing. The second purpose of our website is to allow administrators to view more detailed data as well as have remote control of our system.

Taxpayers are able to visit our website at <http://rightalert.net/test>. This website displays an entry every time a biker crosses the intersection along with the time. Additionally, team and project development information may be viewed on this website. City officials or administrators of our product are able to visit <http://admin.rightalert.net>. The administration panel will show detailed visuals of collected data. The admin panel will allow administrators to manually control our product at any intersection.

*Procedure*

To test basic functionality of our website, the public may visit <http://rightalert.net/test> directly. For administration via <http://admin.rightalert.net>, the admin panel will be password protected. For testing, the admin panel has an “LED On” button which may be clicked to manually turn on the Alert Sign.

*Verifiable Result*

After clicking the “LED On” button in the administration panel, the LEDs on our mock alert sign will blink rapidly for six seconds. For this test, we have also include an LED on w/ Biker button, because in our final design, triggering of the alert sign indicates that a biker has passed. When this button is clicked, our database is updated with a new biker entry. This is also verifiable via both the public user site and the admin site as the visible data on the page is updated.