CPE 462 Image Processing
Final Project Report

Thomas Wang, Mitchell Reiff, Peter Shikhman

May 12, 2022

I pledge my honor that I have abided by the Stevens Honor System
Thomas Wang, Mitchell Reiff, Peter Shikhman

Abstract

The objective of this project was originally to create a program which could detect if cars were waiting at an intersection and appropriately make decisions about changing traffic lights according to the demand. While doing research for the project, the group found many examples of using OpenCV with both C++ and Python in order to detect objects in an image, which was used as the inspiration for developing our solution. The tools used to complete this assignment were C++ with OpenCV, to input and display the images, and TensorFlow, to detect the objects in the images. While the pre-trained models were able to detect cars without issues, the group was not able to replicate such accuracy. As a result, the decision was made to pivot from detecting cars waiting at intersections to detecting if pedestrians were still in the crosswalk before changing traffic lights. When testing, a combination of images from Google Images, screenshots of live IP cameras from insecam.org, as well as our own pictures were used. The compiled program can be run with a single parameter, which is supposed to be the path to the image file used, and it will both display the image with overlaid information about any detected objects, as well as print the final verdict to the console.

Introduction

One of the biggest problems on America's roadways is the amount of traffic congestion, namely the amount of vehicle buildup at intersections. The biggest cause of the traffic congestion problem is the lack of any traffic light synchronization between intersections. For example, lanes might be busier depending on the time of day, like peak rush hour traffic compared to midnight traffic. The consequences of lack of synchronization are drivers either waiting for minutes or green lights, or green lights wasting their time on empty lanes.

To combat this problem, our group planned on creating an imaging processing system that would detect cars at the front of each red light intersection, compare the two streets, and prioritize a green light to the street with the least amount of cars. After some consideration, we decided to pivot to detect pedestrians. In dense cities, there are always a lot of cars facing both directions of traffic. To better accommodate the traffic system we wanted to shift our focus to pedestrian traffic as that is always a large factor when it comes to city traffic. We wanted to apply the same principle of having the traffic light change color after there are no more people on the crosswalk.

When our group was writing our initial proposal, we researched previous final projects that had various kinds of image compression and restoration. However, we decided to take a unique approach and extract data from the image rather than actively manipulating it. This led us to our current project. We settled on this idea because traffic congestion is a widespread problem across the United States, and image processing systems like this would help regulate traffic through a computer system. Los Angeles recently launched an automated traffic control system that synchronizes all of its nearly 4,400 traffic lights over 469 square miles. As a result, traffic times were reduced by 20%, and the reduction in idling saved 1 million metric tons of Carbon.

Project Outcome

The first steps of our project involved researching how to process information in images. Fortunately, there were many tutorials on how to do this specifically with cars, as image processing involving traffic is well documented. We found some preexisting code on those tutorials that gave us an idea on how the code structure looked, and the expected output. Figure 1 is the result of compiling the code from one of the tutorials.

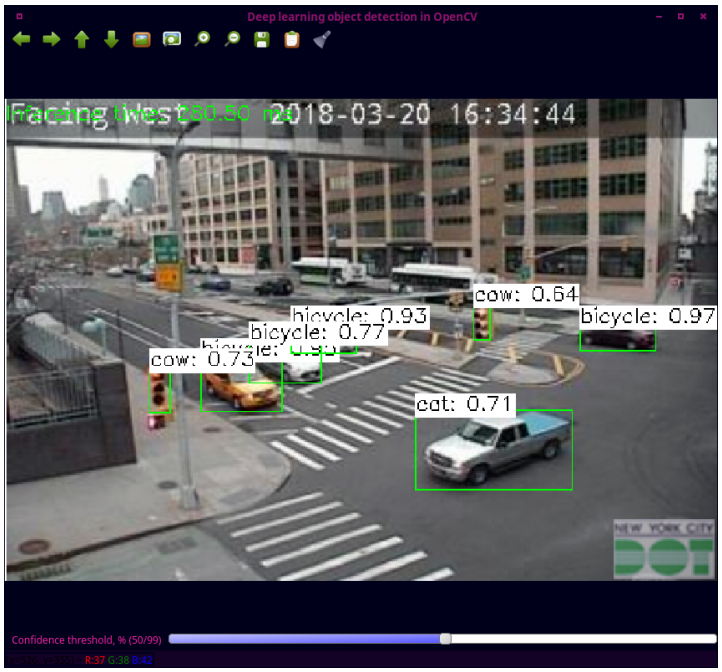

Figure 1 - First Prototype

It is clear that the code is able to recognize car shapes in an image. In addition to the cars, the code was also able to recognize the traffic lights. Even though the code was able to find these items in the image, the label on every item is wrong. Nonetheless, we had software that was able to recognize the cars clearly, and we were able to use the software on a live-feed video. We then looked to other tutorials to observe the different coding methodologies other people had while writing software.

Before we moved on to a different tutorial, the next step was debugging why Figure 1 was returning wrong labels. We fed the software clear images containing one object, so we could narrow down if there was too much noise in the image. However, when we tested a clear image of a car, the software returned that it was a bicycle. We eventually discovered a label text file containing all the objects the software could detect, and we found out that it wasn't capable of detecting cars. This wasn't a dead end though, as the tutorial that we utilized contained a wide variety of artificial intelligence datasets. After searching through the database, we found a dataset that was specifically designed for traffic recognition. Implementing the new database into our code yielded better results, returning an image that contained properly labeled cars and traffic lights. With this new dataset-specific for traffic, the code was able to recognize more items in the image like the buses and cars in the background. This is shown in Figure 2.

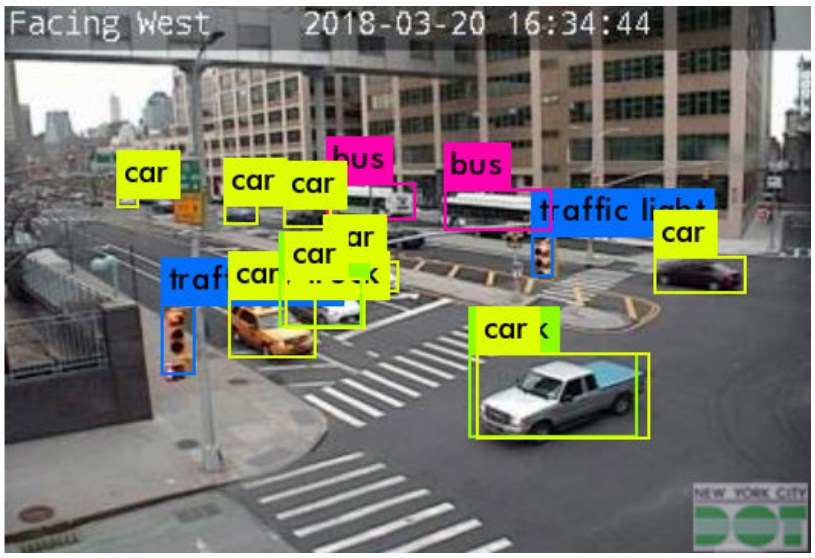

Figure 2 - Second Prototype

We then went to test a random traffic feed video and we received our desired result. The program was able to isolate and identify the specified items in the image. This process can be seen in Figures 3 and Figure 4.

Figure 3 - Raw Image

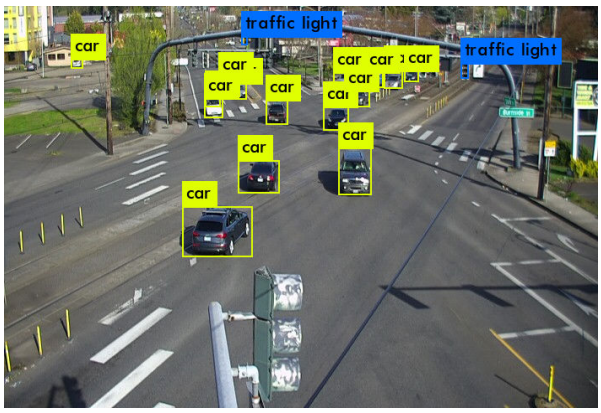

Figure 4 - Figure 3 After Processing

The next step of the development of our code was to make sure it worked with our own cameras and images, so we decided to test our webcams as a potential source for our code to run through. To make it easy for us, we didn't search for a car and instead made sure that the image processing software identified us as people. The program was able to accomplish this, and the software successfully identified Mitchell as a person as seen in Figure 5.

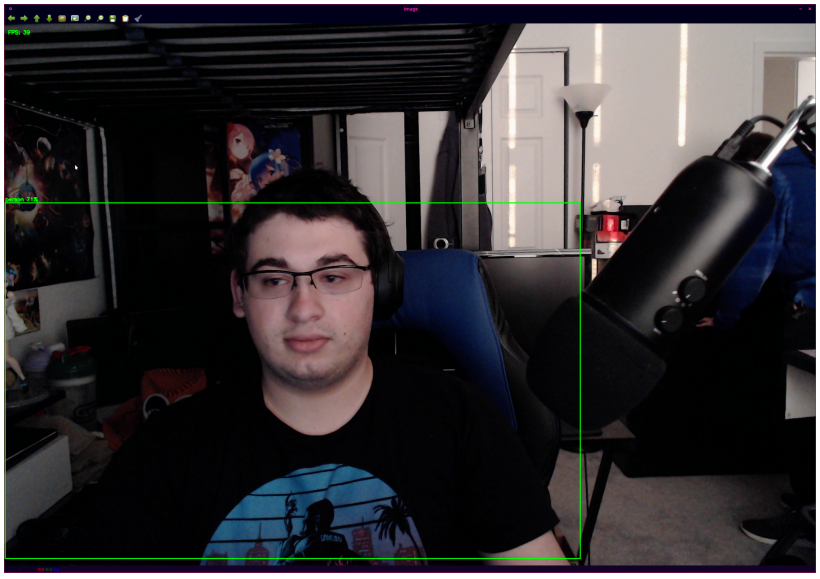

Figure 5 - Person Testing

After we verified that the code was able to detect people, we gave it images of people on crosswalks. As seen in Figure 5 and Figure 6 the code was able to detect people both facing the camera and having their backs turned to the camera.

Figure 5 - Raw Image

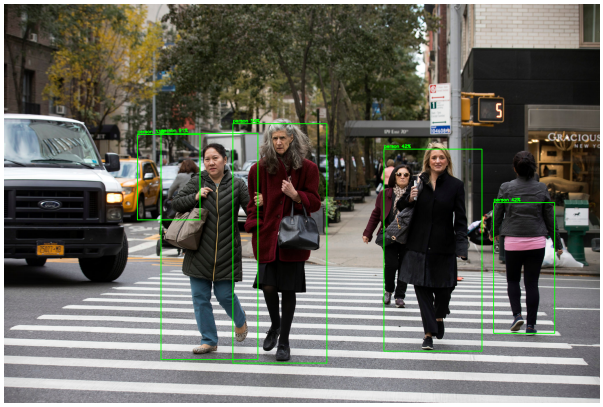

Figure 6 - Figure 5 After Processing

As a final test to the code we wanted to make sure that the code would not detect a person when there was no one in the frame. To do this we found an image of an empty crosswalk. As seen in Figure 7, the code does not place any frames meaning it does not have any recognized objects in the frame.

Figure 7 - No Objects Detected In Empty Crosswalk

To tie the project together and complete our set goal, the code sends a message to the console that says “It is not safe to turn the light green!” when people are detected or “It is safe to turn the light green!” when people are not detected. In a real world scenario, the message would be converted to a signal that would be sent to the traffic lights but for our projects, we decided to have the code send a message to verify it was working correctly.

Links:

https://github.com/MAPReiff/CPE462-PedestrianSystem-2022S - **Our code**

- Instruction on how to compile and run the code is in the README on the GitHub repository.

https://github.com/tensorflow/models - AI trained models

https://github.com/opencv/opencv_extra/tree/master/testdata/dnn - Image processing datasets and examples

Team Member Contributions:

Each team member contributed evenly to every aspect of the project. From doing research on object detection in images, writing code, getting sample photos and writing the report.

We started with research for the project. After finding some sources, we started a GitHub to share the code. Once we completed the code, we found sample pictures and some members volunteered their own photos to be used in testing. Once the project was complete, each member contributed their knowledge to the report and proofread it to make sure everything that was worked on was included.