METROPOLITAN STATE UNIVERSITY[SM] OF DENVER

Digital Image Processing
Project 3: Crash Detection and Pedestrian Detection

Steven Jennings

09 May 2018

Overview

The goal of this project was to develop a means of detecting a crash in a given set of data (which includes a video file and a text file with GPS data and gyroscopic data) and detect pedestrians around the time at which the crash happened. These goals were to be achieved through the following steps:

- Fetch the dataset
    - Video data (which is just a set of images)
    - Text data (GPS and gyroscope data)
- Analyze the data
    - Get the first derivative of the velocity data (which yields acceleration data)
    - Determine the maximum acceleration value
    - Determine the timestamp at which the maximum acceleration value occured
- Sample frames from the video starting at the detected timestamp
- Run each sample frame through pedestrian detection

Procedure

1. Fetch video and text data
2. Analyze the text data to detect the timestamp of a potential crash

I opted to use only GPS data because the velocity is given in the GPS sensor data. By taking the first derivative of the velocity data, the acceleration data may be obtained.

A problem with this approach is that the GPS data only gives real-time timestamps, as in, it only gives the exact date and time at which the crash happened, but does not give timestamps for the video. To work around this problem, I used the very first timestamp in the GPS data as the “reference” time, and simply calculated the difference between the detected time, and the reference time, which gives me approximately the video’s timestamp at which a potential crash occurred.

3. Use the detected timestamp to sample frames at around the point in the video where the crash happened

Since I know the framerate of the video is 30 fps, I used this value to calculate near the exact frame at which a potential crash happened. Using the detected timestamp from step 2, I was able to convert the entire timestamp into one unit of just [seconds], then multiply that value by 30 [fps] to yield the frame at which a potential crash happened.

4. Run all sample frames through pedestrian detection (This is where OpenCV happens)

Analysis

Considering the vastly different approach I took, my results turned out very well. OpenCV has easily proven to be a very reliable image processing library. It’s a bit strange how OpenCV detected the woman’s leg instead of the whole pedestrian, but the problem most likely exists with the parameters I used for the detection function.

The program detected 2:37 as the timestamp in the video (the video’s timestamp, not real time) where a crash most likely occurred.

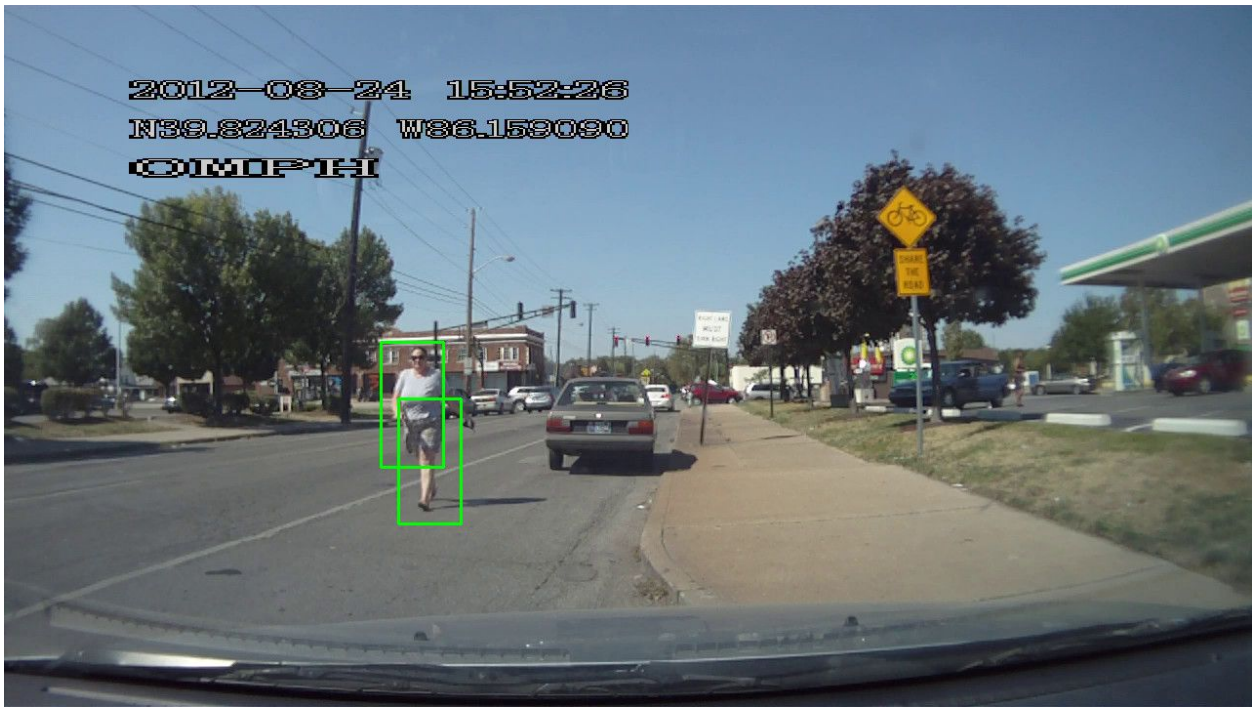
```
Starting Pedestrian Detection...
This automatically detects a crash from provided data
(from sensors, for instance), and detects pedestrians from around the
point in the video where the crash happened.

Detected potential crash at 0:2:37
Potential crash is at frame 4710.0
```

This is the frame that the program detects. The actual crash happens roughly 1 second later in the video.

These are sample frames where the program detects the pedestrian.


2012-08-24 15:52:25
N39.824308 W86.159093
0MPH


2012-08-24 15:52:26
N39.824306 W86.159090
0MPH

You can see in many of the detection images that OpenCV was detecting her leg instead of her whole person. That was probably poor implementation on my part, but for my first attempt at pedestrian detection, this wasn't too bad.

Conclusion

This was a very good project. It encouraged me to do my own research and I ended up implementing OpenCV for an actual application. Even though OpenCV had a couple of false positives, it was a very stable system that, given a good team of engineers, could probably be used very well in a real-world application.

As I was working on the project, I thought about what I would do if I was leading a team to develop a real-time pedestrian detection system. I thought the best approach would be to use dual cameras to simulate depth perception. With depth perception, it would be much easier to determine whether or not an object existed in a dataset. I haven't done too much research, but I believe Intel®'s RealSense technology implements this idea that the world can be modeled in three dimensions through depth perception. Many of Intel®'s existing RealSense hardware already implements dual cameras for depth perception. This was just a thought I wanted to share as I was completing this pedestrian detection project.

The class as a whole was also very good. I wish I wasn't graduating so I could take your machine learning course next year, Dr. Jiang, but I think I've learned enough to make a significant impact wherever I go. Thank you for a fantastic final semester!