# Python/OpenCV Speed Detector #

This is a program that uses OpenCV and Kalman Filters to detect and track cars from a traffic cam livestream. Once a car passes a certain distance, its speed is calculated and checked against the speed limit. If the speed exceeds the limit, an image is saved, showing both the speeding car and its speed.

Here is a screenshot of the program in action:

![Screenshot](./images/example.png)

While this does work decently well, there are some pending issues, namely:

\* Objects that start in the middle of the frame will have inaccurately high speeds, due to being on the screen for less time

\* Loading the livestream noticeably lags when fetching a new chunklist, resulting in slower speeds due to lag time

\* Some noise is still present and detection, and it can often mess up tracking and speeds as a result

\* It can be inconsistent with cars coming from either side of the road

### Credits ###

The object tracking and Kalman Filter implementation (`tracker.py` and `kalman\_filter.py`, and a bit of the drawing/detection methods in `main.py`) are from [Srini Ananthakrishnan's multiple object tracking](https://github.com/srianant/kalman\_filter\_multi\_object\_tracking)

The [livestream](http://dot.ca.gov/d3/cameras.html) (Highway 80 at Whitmore Grade) is provided by the California Department of Transportation

### Side Note ###

Before realizing that OpenCV's `VideoCapture` was based on FFMPEG, and could therefore capture hlsvariant livestreams, I manually fetched new chunks and chunklists. I found this method to be somewhat more reliable than directly using VideoCapture, and definitely more reliable then piping FFMPEG using `subprocess`, as those methods can often result in 'jumps' in the livestream. The upside to these methods is that they are considerably faster than manual chunk/list loading. Also, manually updating the stream meant that I could easily account for load lag. In the end, I decided to go with direct `VideoCapture` because of its speed and because it allowed me to remove asynchrony completely.