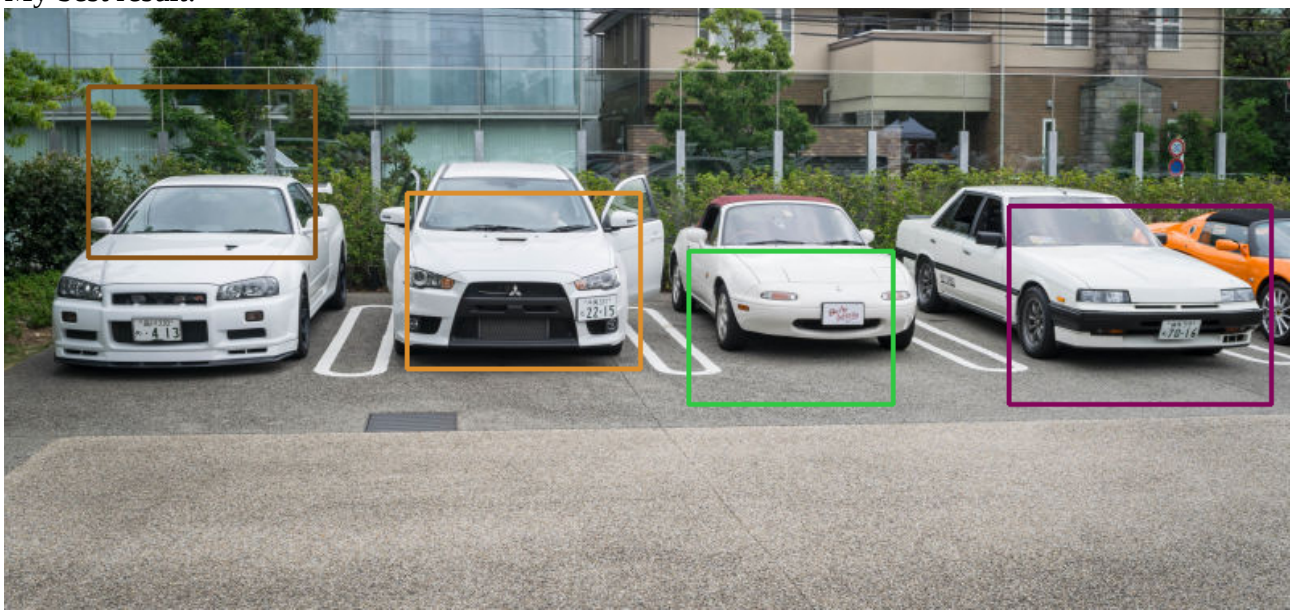
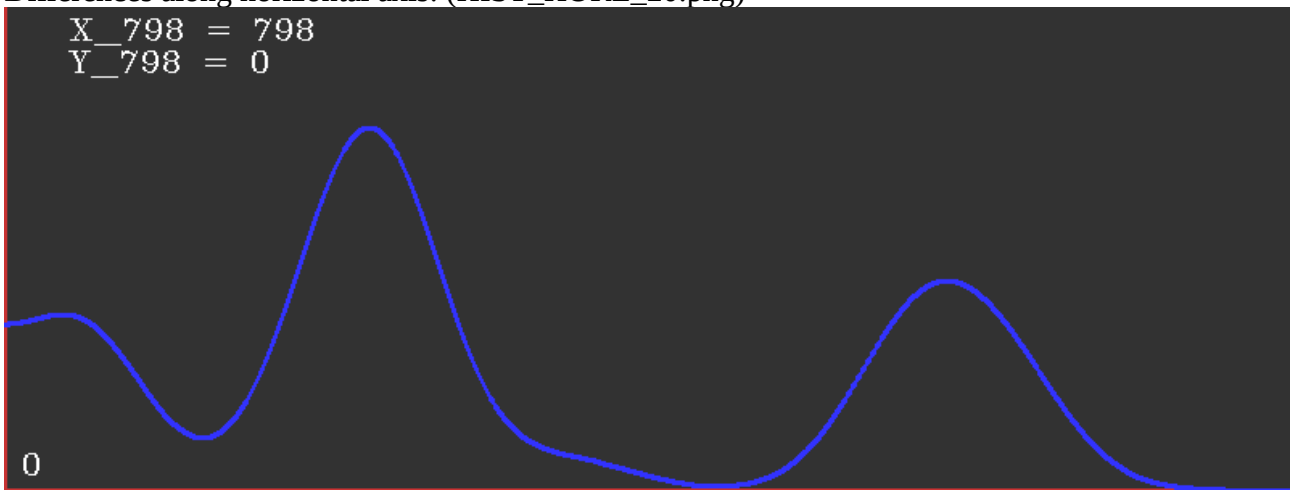
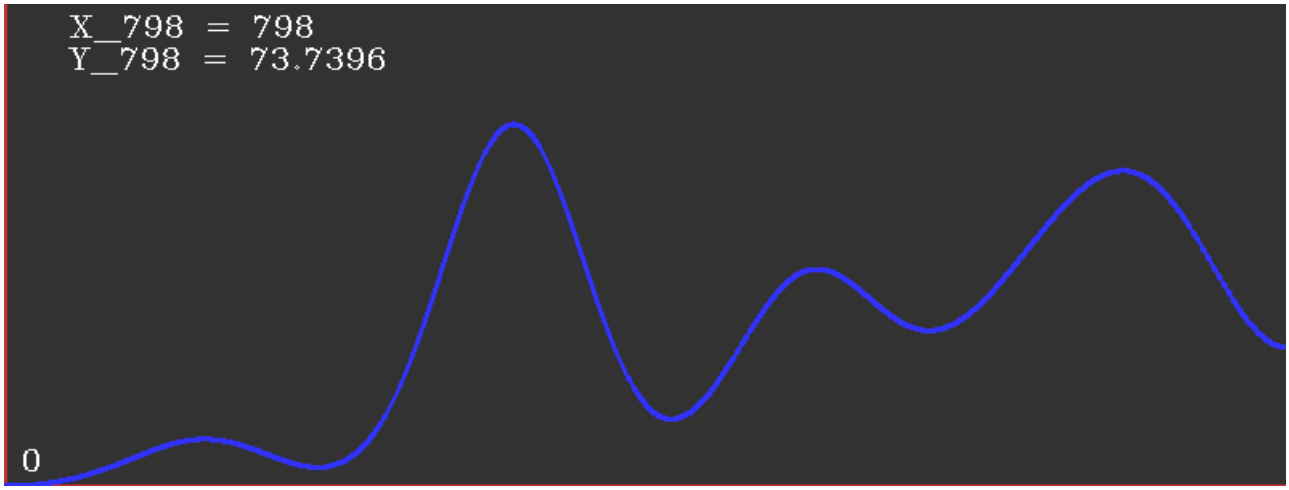
My best result:



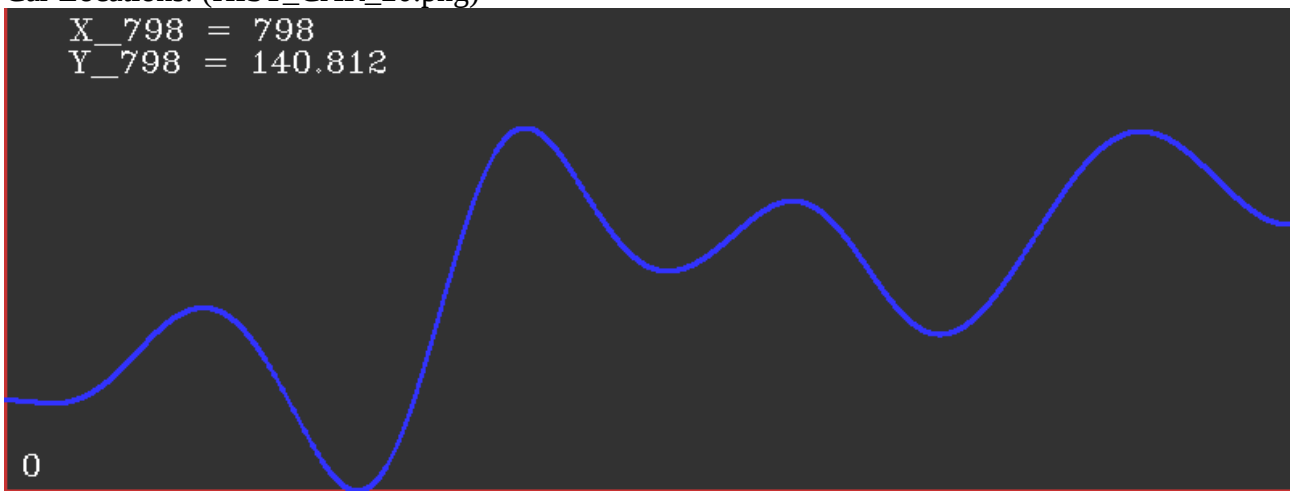
Differences along horizontal axis: (HIST_HORZ_20.png)



Differences along vertical axis: (HIST_VERT_20.png)



Car Locations: (HIST_CAR_20.png)



Good peaks in this histogram give us the x coordinate of the car. To get the y coordinate, I took the weighted average of the y coordinate while creating the vertical difference histogram. Basically, while summing the vertical differences up along one column, I created another histogram that gives the center of gravity of the vertical difference magnitudes.

```
if(abs(gy.col(i).at<double>(j)) > threshy)
{
        xData.at<double>(i) += abs(gy.col(i).at<double>(j));
        loc.at<double>(i) += abs(gy.col(i).at<double>(j)) * j;
}
```

xData is the vertical difference histogram.
j is the current y coordinate and loc is the other histogram.

To calculate the width of a car, I summed the distances from a peak to its neighboring pits.

```
width = steps[i+1] - steps[i];
```

steps is the vector of pits in the histogram. Height calculation was heuristically done as shown below:

```
height = width * 3 / 4;
```

Performance improvements:
Peak quality: if a peak has small difference in magnitude copared to neighboring pits, then that peak is not good enough and should be eliminated.

This can be further improved by merging a bad peak to a neighboring valid peak. But I did not include that in my code.

Please see other results and code for detailed understanding.