Freeway Car Detection
CS 223B, Stanford

Eric Park
ejpark@stanford.edu

Brian Tran
britran@stanford.edu

Joakim Arfvidsson
joakimar@stanford.edu

January 25, 2006

1 Overview

The objective of this assignment was to locate cars in images of freeway driving. The viewpoint is that of the driver. One would imagine that the output of our algorithm could be used by an autonomous agent driver trying to keep track of other cars in the vicinity.

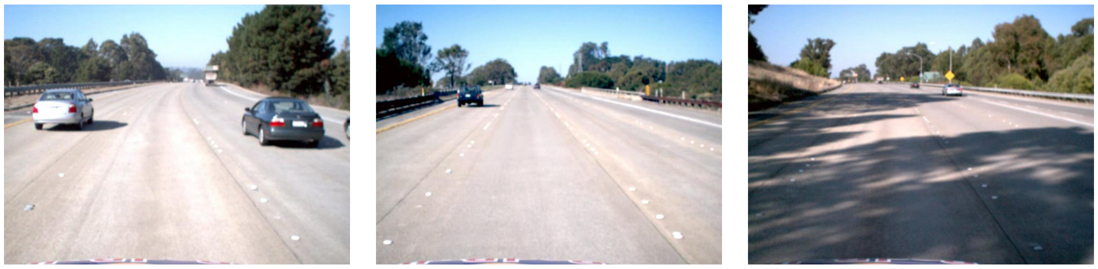

Figure 1: Typical images

2 Problem

The cars in question are all seen from the back. They can be seen from slightly different angles though, as they can be all over the road in front of the camera. Weather conditions are stable in our sample set, but there are shadows covering the road.

3 Our solution

One would hope that any autonomous driver works by positively identifying road instead of negatively identifying specific classes of obstacle. We pursue that design goal by having reliable road detection. Once we have identified most of the road, some of what is left is cars. We further use positive identification of cars through analysis of edges as well as filtering of matches of unrealistic size.

3.1 Road detection

We assume that the road just in front of our car is free (Stanley would use its lasers to confirm that), and take a few samples to capture the road color. We flood-fill the road using the sampled colors, getting most of it. We then use the RANSAC algorithm to find the edges of the road to constrain further car searching.

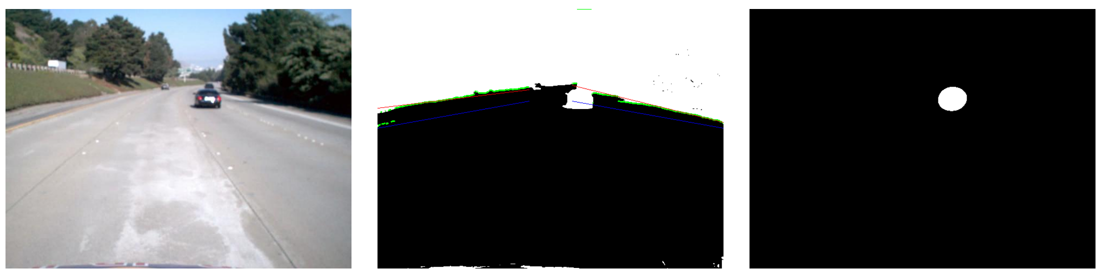

Figure 2: Starting with a source image, we flood-fill the road (black), and apply RANSAC to fit lines to the road edges (shown in red). We then fill everything above the fit lines and apply a Gaussian blur followed by thresholding.

3.1.1 Flood-filling

We sample several pixels just beyond the hood of the car, and apply a standard flood-fill to each sample point. We then use the unions of these regions as our road-classified pixels. Using multiple samples enables us to reduce the fill threshold and create less false positives for road. We get few of those, as we have balanced to prefer false negatives.

3.1.2 RANSAC

The notable feature of this algorithm is that it is robust to outliers, which there are pleny of in the road edge. This occurs specifically when the outline of a car is overlapping the edge. By correctly estimating the road edge, we can discard what's outside while keeping what's inside for car classification.

The search space of the RANSAC algorithm begins just below the estimated road edge on either side of the road; a diagonal line about 200 pixels wide. From this line, the algorithm searches upward until a transition from black pixel to white pixel occurs. Two random points are chosen along this edge and a line is fit. The error function returns the distance between the fit line and the actual road edge. We iterate until a satisfactory number of line pixels lie within a bounded vicinity of the actual road edge.

Once lines are fit on both sides of the road, we connect the two lines together and fill everything above the dividing line, thereby eliminating everything beyond the sides of the road and above the horizon.

3.1.3 Blur

Once the road, sides of the road, and everything above the horizon has been filled, we can apply a Gaussian blur. The blur amount is varied according to perspective as noted later in our report. Based on the vertical position in the image, we vary the blur radius to allow for small vehicles on the horizon to be captured while filtering out noise further down the image.

3.1.4 Threshold

Non-black pixels in the blurred image are thresholded to white producing output similar to the rightmost image in figure 2. This data is used to capture large vehicles emerging from the sides of the images.

3.2 Car edges

For positively identifying cars, we made a pattern matching system that draws on the prevalence of horizontal and vertical edges in cars. Combining those we were able to get convincing positive matches in the road, with spurious matches in the scenery that were eliminated by other means.

We tried simply using the amount of horizontal and vertical edges in an area as a measure for car likeliness. That didn't work so well. What we missed out on was spatial distribution of those edges. By taking advantage of the local edge patterns we can match much more reliably.

In fact, matching direction of edges appears to work better than matching the number of edges.

- Do edge detection with the Canny algorithm. This provides the high quality edges we require. Sobel did not fit this approach.
- Normalize edges. To reduce bias towards areas filled with edges in general, like trees, we decrease the intensity of edges proportionally to the local amount of them. This effect is visible in figure 3.
- Extract horizontal and vertical edges from the Canny result. This is done through a simple algorithm that for every three properly lined up active pixels sets the middle one to active in the output. This assigns slightly slanted lines a somewhat lower score.
- Apply separate horizontal and vertical edge pattern matching. See piece on perspective below for implementing the right amount of scale invariance.
- Multiply responses for our two patterns together. We are left with a map of where both vertical and horizontal edges are located in a way that is somewhat similar to a car.

For a typical car edge configuration, we looked at a few samples and tried out a few variants. The ones in figure 4 work quite well. Note that the slope of a line in these pattern does not have any significance, but it's illustrative that we chose to populate each pattern with edges of the same orientation that the ones it's trying to match. There is blurring, so edges does not have to match exactly.

We are satisfied with the quality of matches of this algorithm in many situations, and they are typically positioned spot on. There are however some situations where problems occur, mostly regarding false positives near the road. These are hard to separate from the real cars, and we do lose some fringe cars as a result.

3.3 Perspective

A notion of perspective or relative sizes is extremely helpful in dismissing a lot of false positives as well as searching for positive matches in a sensible way.

In general, getting training data from the test set would make the results pointless. We did however study perspective through size and position distribution of cars, as we considered this to be mostly a property of how the camera of Stanley was mounted.

With manual bounding boxes on all cars, we extracted the data in figure 5. We then used this data to determine which sizes of edge pattern matches to consider for each scanline of the output feature map.

Obviously, you are much better served in general by considering the lower edge of any bounding rectangle. The center varies with size.

3.4 Compositing

Output from the edge detection and road-fill methods are composited into a single combined image by extracting the useful components of each image.

The road-fill method does not deal well with shadows; in such cases we only look at the output from the edge detection. Similarly, the edge detection method does not deal well with large vehicles emerging from the sides of the image, and thus we only look at the output of the road-fill in these cases.

For objects between these two extremes, we multiply the output of the edge detection and road-fill and search for local maxima. Once a candidate point has been found, we create a lower bound for vehicle size based on perspective, and then increase the bound based on image intensity. Within the box, we apply corner detection to locate the edges of the vehicle. Finally, we apply convex-hull to fill the region.

4 Suggested enhancements

- The edge patterns are prime suspects for machine learning. We would have done this given time and data. You could also easily improve on them by tweaking a bit. We did not get

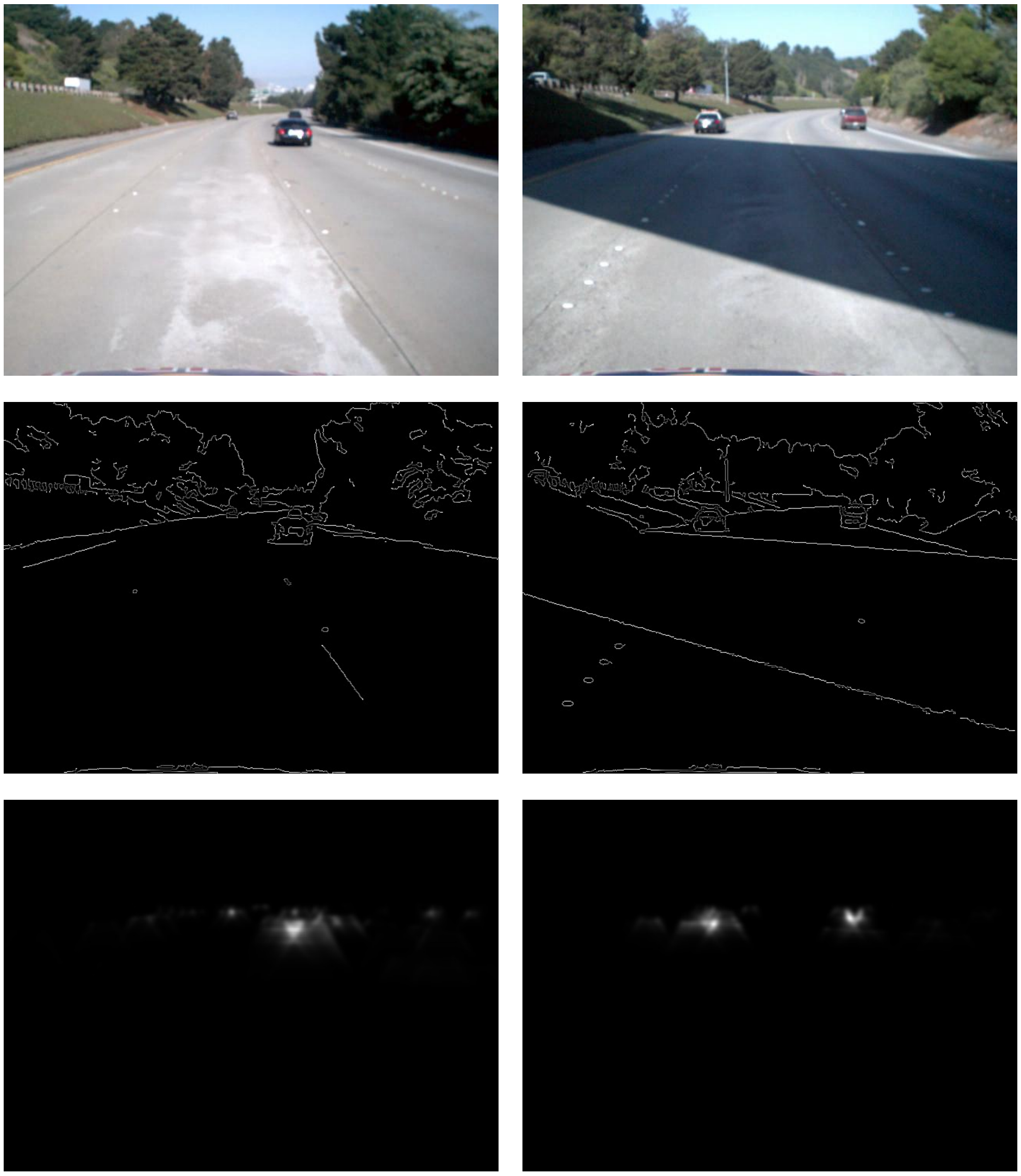

Figure 3: Response of edge pattern, with scale invariance applied through use of estimated range of car sizes for each scanline.

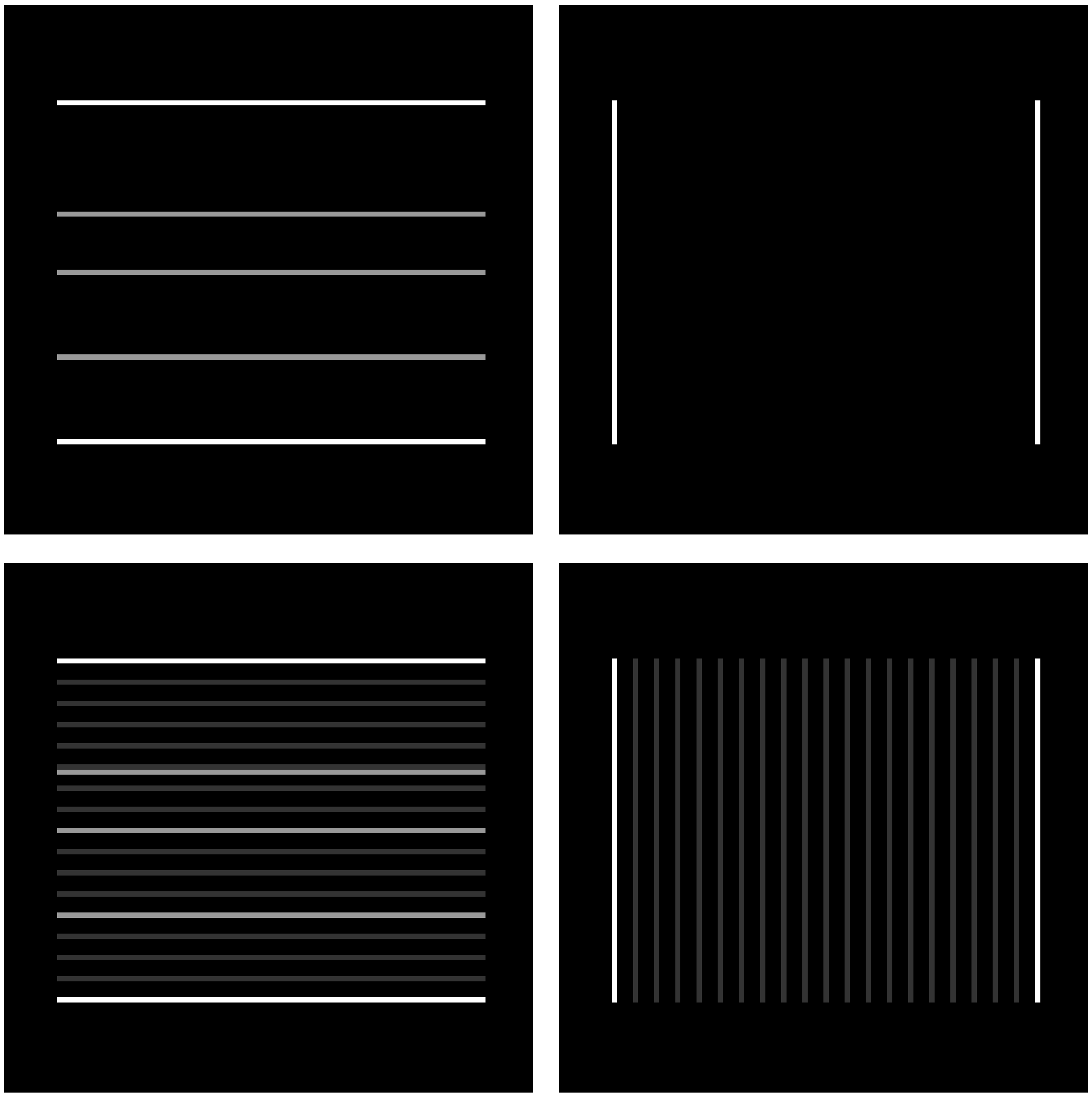

Figure 4: Car edge detection patterns employed. Only positive weights were used, with grayscale indicating weight.

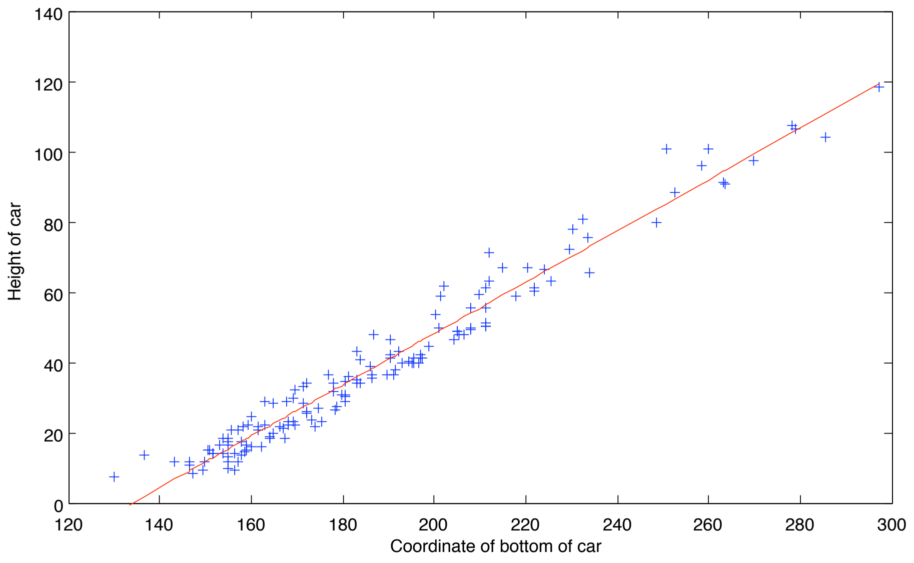

Figure 5: Information on size of back face of car from our viewpoint

good results for using diagonal edges as well. Negative weights for some edges is probably also a good thing, which machine learning would be able to use efficiently.

- There are other features that could be incorporated – car shadows and taillights for example – that might help nail a few more false negatives.

References

[1] Martin A. Fischler and Robert C. Bolles, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, SRI, (1981)

[2] M. Hu, W Yang, M. Ren, J. Yang, *A Vision Based Road Detection Algorithm*, Nanjing, China, (1993)