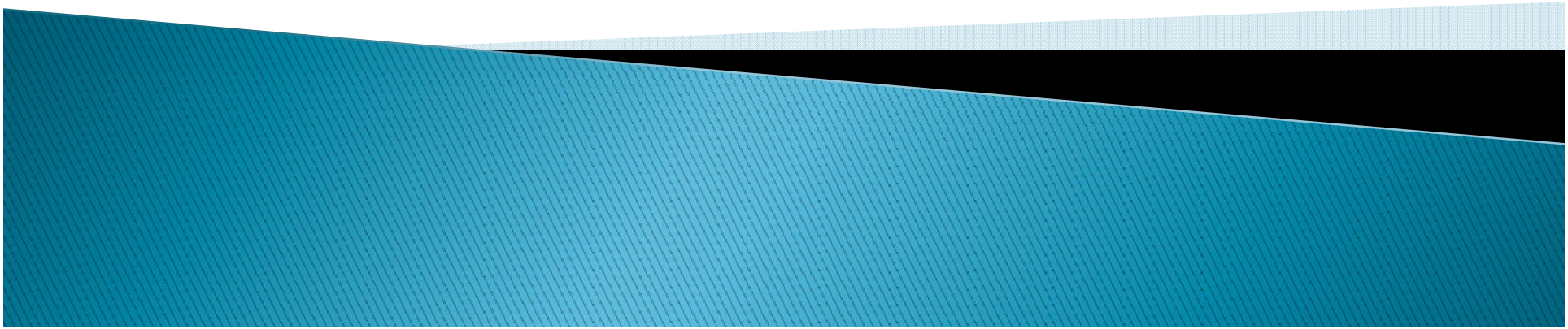
License Plate Recognition

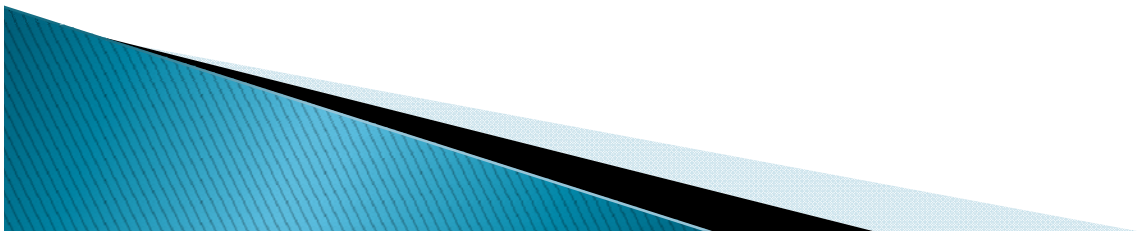
Brian Forbis



Commercial and Government Uses

A license plate locator can be used for:

- CCTV cameras at intersections and parking lots
- Toll roads to bill users of the road
- Dashboard mounted cameras on police cars to identify drivers
- Plate blurring on database images to preserve privacy



The Algorithm

- Image Pre-Processing
- Horizontal Line Scanning
- Platerun Extending/Merging
- Conditional Classifiers
- Extraction of Features
- Neural Network Classifiers

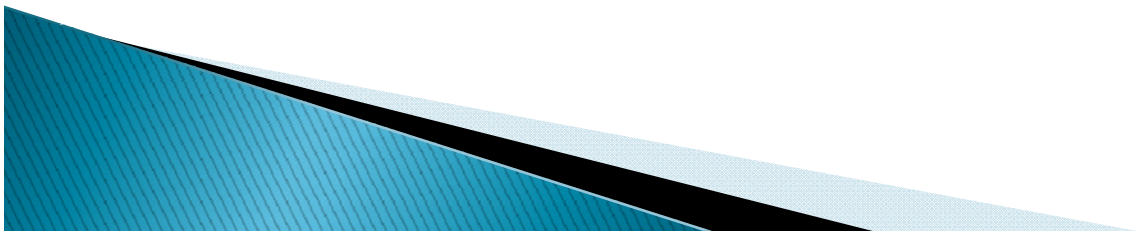


Image Pre-Processing

- Use a horizontal Sobel filter to filter out horizontal lines.

$$\begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix}$$

- Use Otsu's method to get a binary thresholding of the image.

- Use a morphological opening with a disc shape of radius 1 structuring element

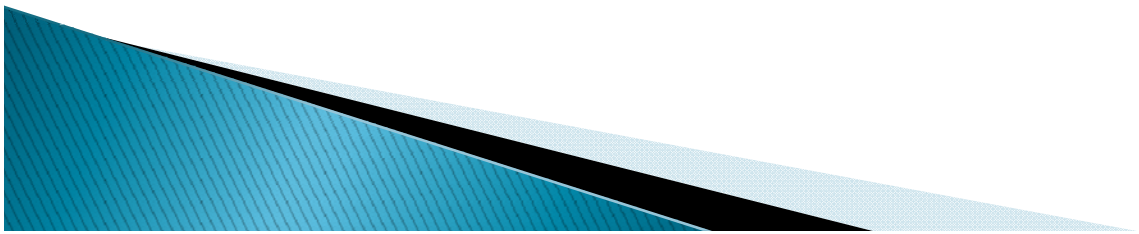


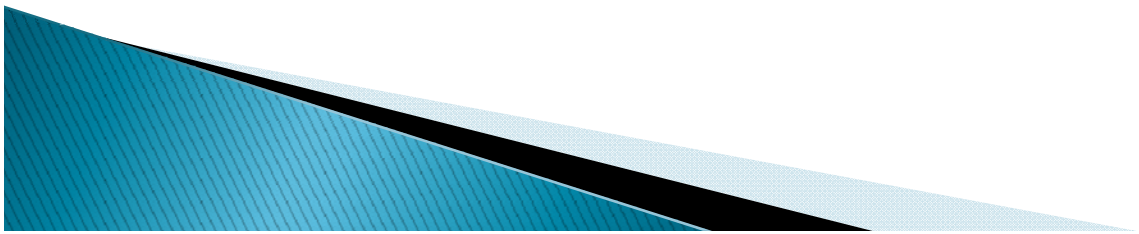
Image Pre-Processing cont.

Original Image

Binary Image after: filter, threshold, imopen

Horizontal Line Scanning

- **Algorithm 1**: Horizontal Line Scan for Columns
- **Data**: row_y of *BW*
- **Result**: row of numbers depicting size of plateruns
- Initialize a vector Acc [width of *Bw*]
- Initialize a matrix [size of *Bw*]
- **foreach** row_{bw}
- **Acc=row**$_{bw}$
- **foreach** col_{row}
- **if** Acc[col]==1, *then*
- Acc[col]=Accs[row-1, col]+1;
- **end**
- Accs[row, :]=Acc;
- **end**



Horizontal Line Scanning, cont.

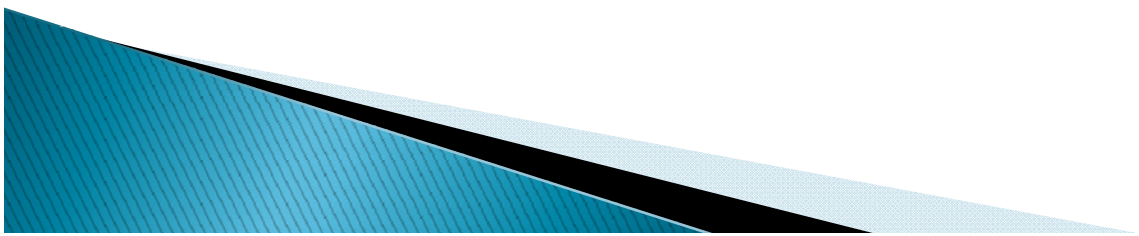
- The creation of Accs should look like:

Binary Image				
1	0	0	1	0
0	0	0	1	1
1	1	1	1	1
1	0	1	0	1
1	0	1	1	0
(a)				

Accs				
1	0	0	1	0
0	0	0	2	1
1	1	1	3	2
2	0	2	0	3
3	0	3	1	0
(b)				

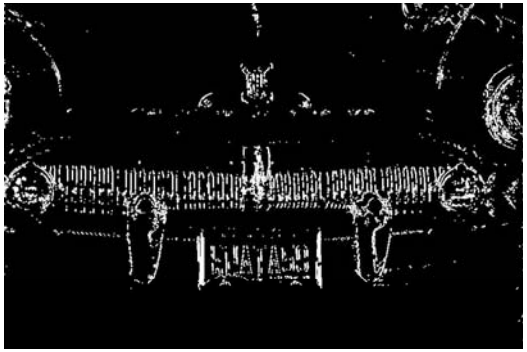
Platerun Extending/Merging

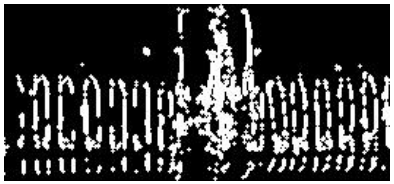
- **Algorithm 2**: Extend Plateruns
- Scan Accs for Plateruns and add to matrix *plateruns* [r_1, c_1, r_2, c_2]
- **foreach** $pr_{\text{plateruns}}$
  while exists a platerun within λ_x
  -Scan horizontal and merge plateruns within λ_x
- **end**
- **end**



Example of Failed Platerun

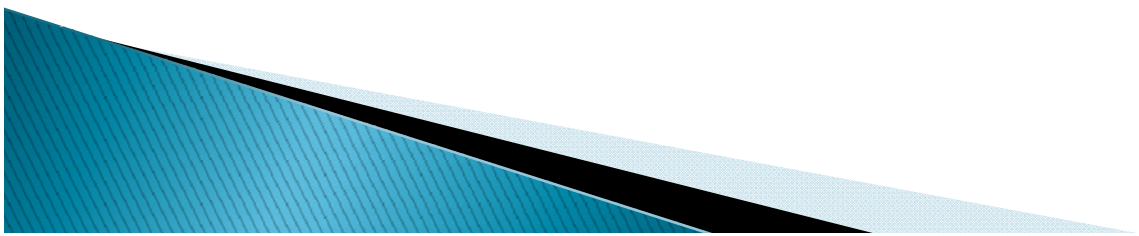
The front grill in this image is picked up as a platerun. This is due to the fact that it consists of many vertical lines in that are close together.





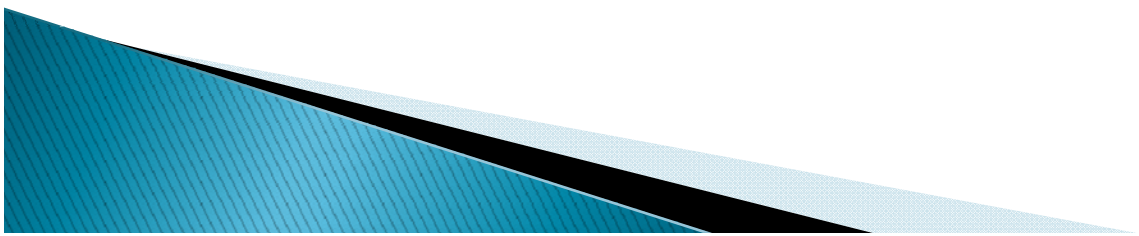
Conditional Classifiers

- Plateruns are thown out if they pass any of the following classifiers:
- $w < 40$
  $h < 20$
  $w > 2/5$ width of full image
  $h > 1/4$ height of full image
  $h/w > 0.55$
  $h/w < 0.22$



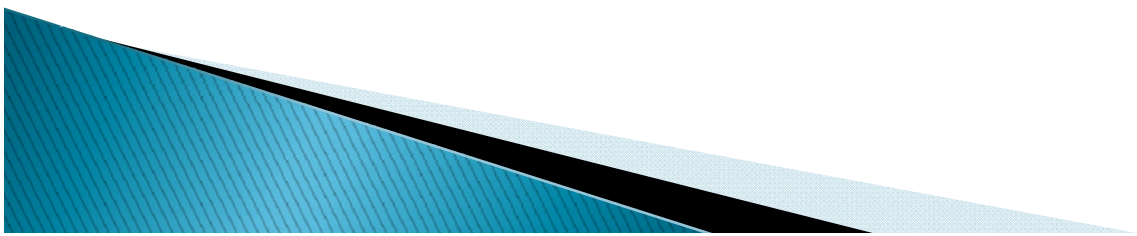
Feature Extraction

- Features are extracted from plateruns to create 80x1 feature vectors. The extracted features are:
- Mean
- Standard deviation
- Skewness
- 18-bin edge histogram
- 9-D texture feature



Neural Network

- MATLAB's build in neural network is used to classify the plate candidates.
- The neural network was trained with 232 extracted plates and 232 extracted nonplates using the normalized 80x1 feature vectors.



Results

- The results from running the program on 66 testing images are as follows:

True Positive	31
False Positive	19
True Negatives	44099
False Negatives	35
Accuracy	0.99
Precision	0.62

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{numbers of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

$$\text{precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false positives}}$$

Conclusion

- The high accuracy of this algorithm shows that with some modification, it could be used commercially. Speed can me improved by converting the code to a different language base such as C++.