CS6640 A1

Jake Bergquist

9/5/18

1) General Architecture for developing an image processing system for analyzing video sequences for moving vehicles.

In order to analyze a video to identify moving vehicles within the video an architecture would generally have to do the following: Take in the series of still frames that make up the video, distinguish what in the video constitutes a moving vehicles, and return an output that describes the vehicles that were identified.

The only file that should be needed would be the video as an input. The video consists of a series of still images taken at a regular interval.

These frames should be loaded into the function in an easy to access way. For matlab, given that these are rgb frames, a cell array with each cell containing one frame should do well. These frames should be treated as read only in order to preserve the video, and any operations to be done on these frames should be done on dedicated copies within the algorithm.

Once the frames have been imported and stored a next logical step would be to isolate a background from the video and perform any preprocessing. All moving elements in the video can be isolated by comparing the frames to the background. The moving elements will be seen as different from the background. With the background eliminated each of the frames can be compared to it to determine which ones have movement.

After the background has been isolated the frames should be preprocessed. Most of the filters and techniques we would likely use work on the gray scale images so the frames should be converted to gray (the RGB ones should still be saved in case they are also needed). Additionally steps such as smoothing could be beneficial for later analysis.

Next all of the major components of each frame should be identified, in particular cars should be segmented. This way once we figure out what is moving and not moving we can use the segmentations to classify if those moving things are cars or not.

From here thresholding, image correlation, edge detection, and other such processing techniques can be used to detect what is moving within the frame, and compare it across the other frames. With these metrics the algorithm can build up a consensus of what is moving, if fits the criteria to be a car, as well as where to goes and how many other cars move in the image. The algorithm should return an analysis that includes details about how many cars were detected, what they looked like, and where they went.

2) Acquiring a 5 second video from the ip camera and turning in the files for the frames and the video.

Files turned in with report. Video was verified to have moving vehicles present as early as frame 1. Frame 1 was also seen to be brighter than the other frames in the video (see **Figure 4**) so frame 2 (**Figure 2**) was used instead for most analysis.

3) Developing code to attain the background of the video from the IP cam.

In order to isolate the background of the image the first step is to load in the frames from the video object. By setting the video time to 0 we ensure that we are starting at the beginning of the video. We can then calculate the number of frames by multiplying the frames per second by the duration of the video. This results in a total number of 75 frames for my specific video (5 second duration with 15 fps). Next we iteratively use the readFrame function to read each of these frames and save them out into a cell array, where each cell contains one frame. Now that we have the frames extracted we can work on attaining the background image for all the frames. Given that this is a video where most of the things are not moving and the things that do move, a few cars, go across the video rather quickly, we can say that most of the pixels in the video spend most of their time as background. As such if we take the mode of all of the RGB values for each pixel that will give us a good background image. To do so we first have to extract all the red, green, and blue values for each frame and group them. We can then use the mode function to find the mode for each pixel. We then use this mode to reconstruct an image which is our background for the video. As a last step we make sure to convert the double output of the mode function back to an int8 so that it can be displayed properly as an image.

The resulting background image (**Figure 1**) can be compared to the second frame (**Figure 2**) and the fiftieth frame (**Figure 3**)

Figure 1: Video Background

Figure 2: Video Frame 2

Figure 2.1: Grayscale of Frame 2

Figure 3: Video Frame 50

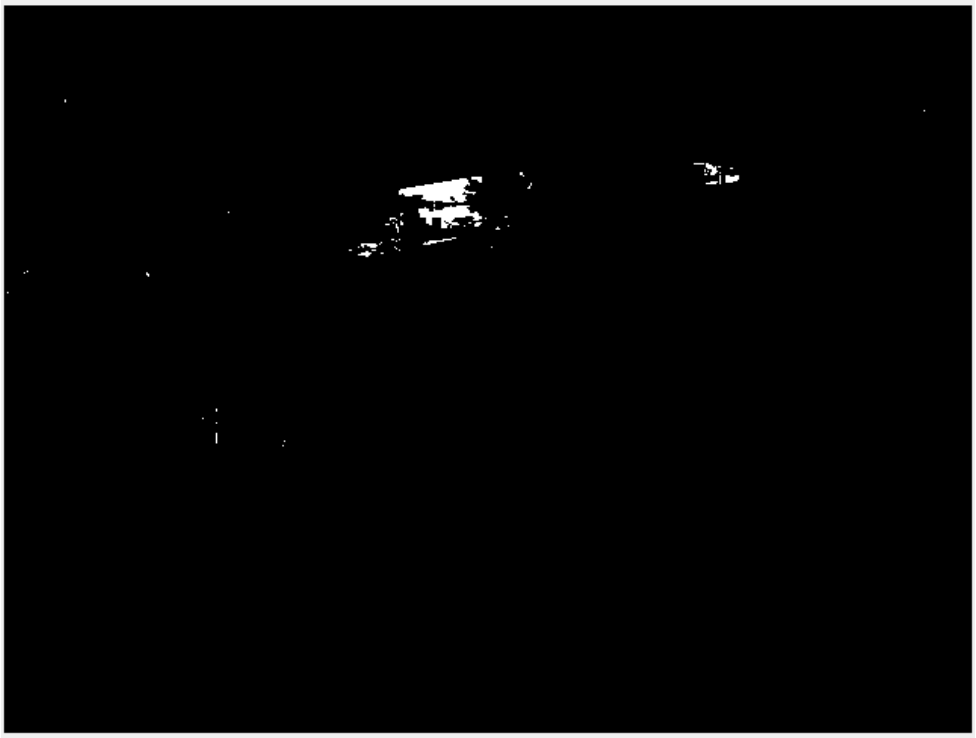
In frames 1 and 50 we see cars (a truck in frame 2, and a bus in frame 50) that are driving across the road. Ideally we would like to remove these to form our background which is composed of all of the stationary things from the video. We see that this is accomplished in our background frame. Of note, the first frame is much brighter than the others (**Figure 4**) but this does not show up on the background

frame. This should be considered for future analysis of the video. It could have been caused by an artifact of the IP camera first transmitting, or perhaps there was an excessive glare during that frame. This is not seen in the subsequent frames.

Figure 4: Video Frame 1

As a sanity check we can subtract the background image from frame 2 and run a threshold. This should show us anything from that frame that was moving relative to the background. This can be seen as the car from frame 2 is highlighted in **Figure 5**.

Figure 5



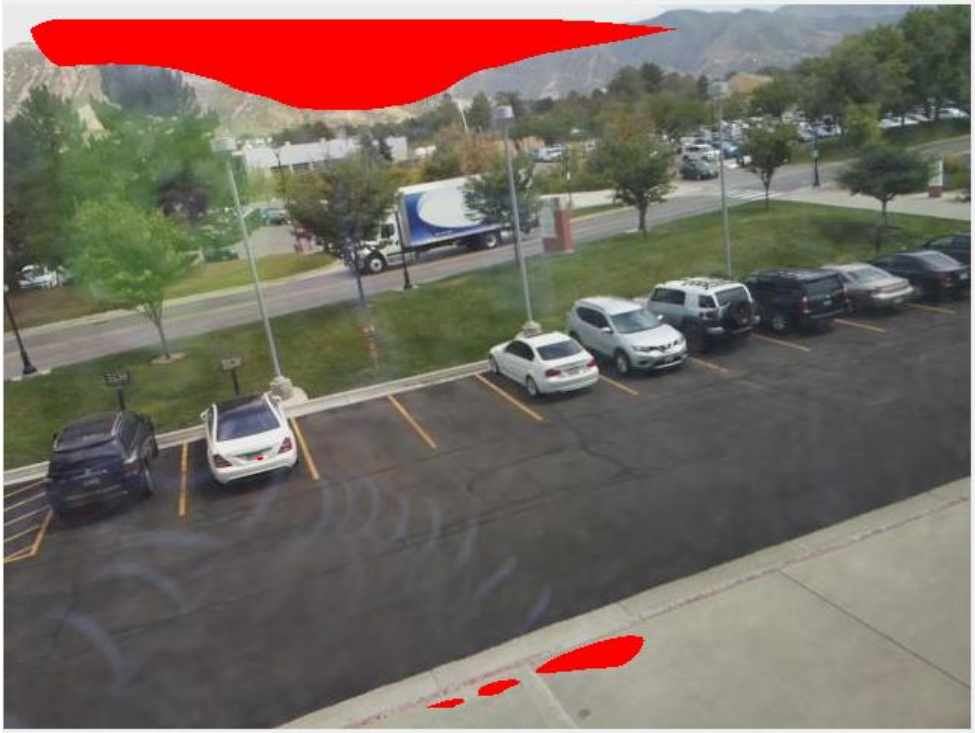
We can see the moving car from frame 2 is mostly captured, except for the trees blocking its path. Additionally it would seem that a few other elements were identified such as the slightly moving trees. Overall this demonstrates that our background image processing function is successful.

4) Investigating the use of correlation of a small section of the image with the rest to segment sections.

I started by using a sub section of the image containing a bumper of one of the parked cars to see if I could find other car bumpers in the image. For all of these correlations I used the rgb2gray() versions of the image and the subimage unless otherwise stated. Using the section in **Figure 6** taken from frame 2 (**Figure 2**)

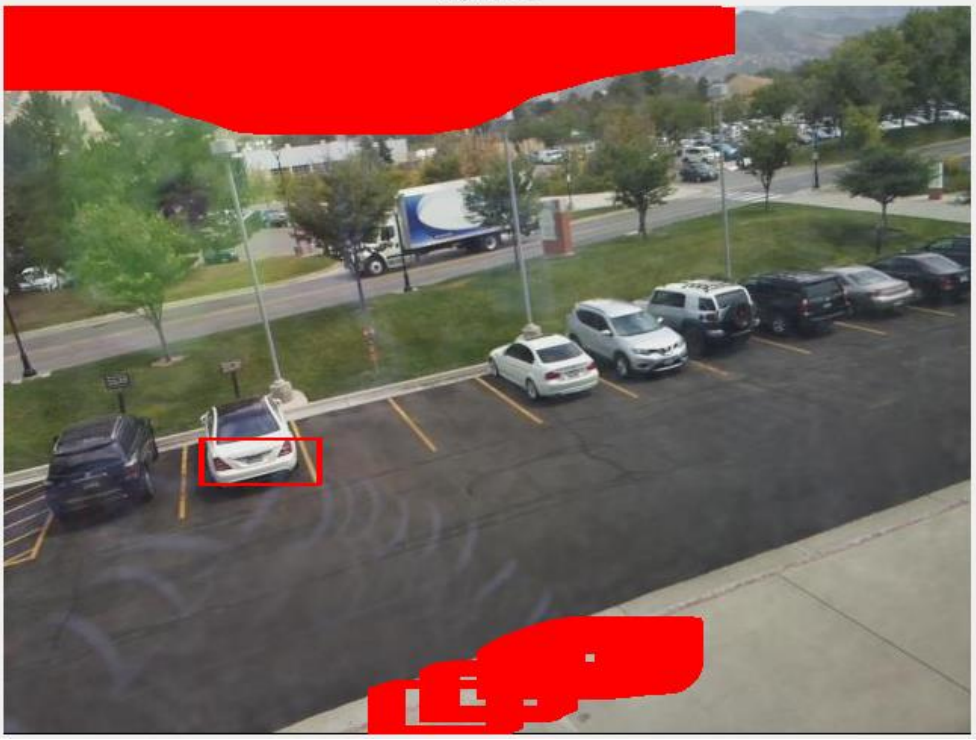
Figure 6

Figure 7



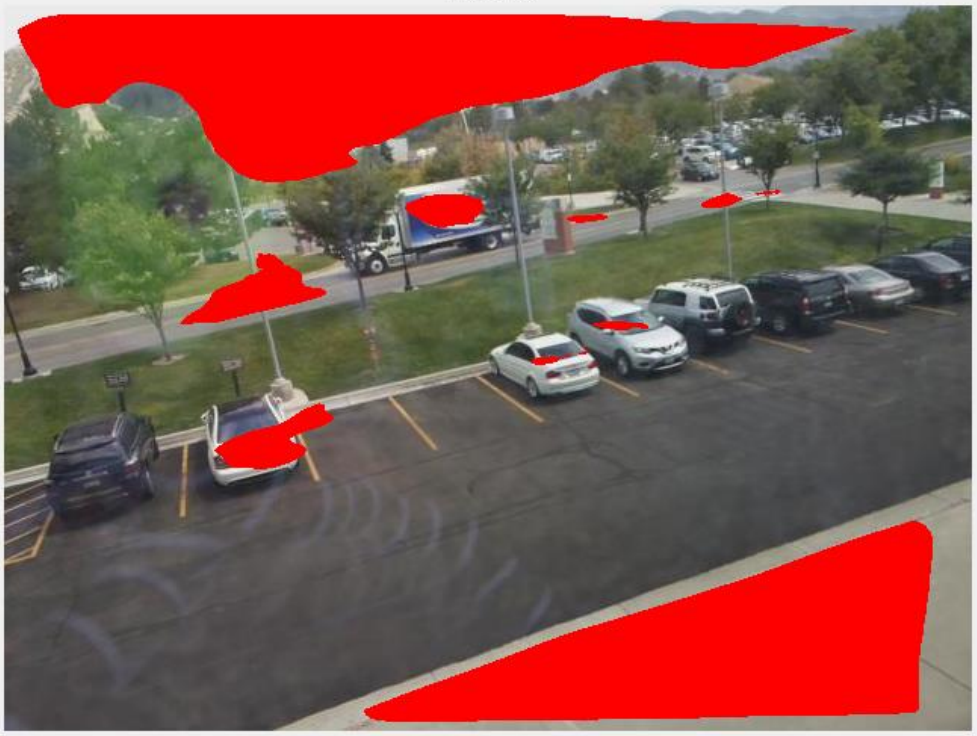
I ran a correlation across the entirety of frame 2 using xcorr2(image, filter). I then scaled the values of the correlation from 0 to 255, and performed a threshold at corrOutput > 185. All pixels that passed this threshold were colored red (by setting their R value to 255 and their G B values to 0). As we can see the white car bumper correlated most strongly with itself and the very white sky (**Figure 7**).

Figure 8



To elucidate the correlation areas I wrote a quick function to outline the areas that pass the threshold with a red box the size of the input filter for the correlation (**Figure 8**). To do this I went to each point in the image that passed the threshold, then turned the pixels around that point at the borders of the size of the input filter to red. As we can see in **Figure 8** the car that we used as our input image, as well as the sky and a bit of the sidewalk had a high enough correlation to pass our threshold. The sky in particular has so many points that passed threshold that all of the rectangles blend together.

Figur 9



Figur 10

By decreasing the threshold to 150 we get **Figure 9** (the highlight plot of locations passing threshold) and **Figure 10** (the rectangles around the areas that pass the threshold). As can be expected a larger area of the sky and sidewalk are captured. However, we also see that a large amount of the original car as well as a couple other white cars in the parking lot are also captured. Unfortunately there is a large amount of unwanted results such as the parts of the road and the semi-truck. This is likely due to the generic nature of the filter (**Figure 6**). There is a large number of white pixles and relatively generic pattern make it easy for this filter to correspond with any large white area on the image well.

Figure 11

Figure 12

I then checked using the xcorr2 function (cross correlation in 2d which I used for the previous examples) against using corr2. In order to use this function however I had to manually walk the correlation filter (the sub image) through the larger image. As such I had to dictate what happened at the edge boundaries. I chose to have the correlation only occur where the sub image could fit. The resulting values were normalized to 0 to 255 as before. When doing so and performing the same thresholding at 185 we get **Figure 11** (the highlighting map) and **Figure 12** (the rectangles around the areas that pass thresholds). When we compare these figures to those of using xcorr2 we find that using corr2 actually is able to locate the bumpers of two of the other cars. Additionally, the erroneous sky and sidewalk correlations are reduced. However, we do get some errant correlation with the road and one of the light posts that pass threshold. Overall it would seem that using corr2 and manually walking it over the image produced a better result. This is likely due to the fact that because I was explicitly providing edge conditions and controlling the normalization more directly I was able to produce a result that better match what I expected given that I had a better idea of what was occurring.

Figure 13

Figure 14

I then played with the threshold and found that at a threshold of 188 and the corr2 method I was able to elimintate the sky and sidewalk being identified while still identifying the three cars seen at the lower thresholds (**Figure 13**). At a threshold of 189 (**Figure 14**) one of the cars was lost (the one where it saw the front bumper instead of the back) but there was a lesser match with the road. Overall I feel that by using this correlation along with a sort of edge detection and flood fill I could segment the cars.

Figure 15

Switching gears lets try using a patch of grass to segment the rest of the grass via correlating with the green channel. We will try using the section in **Figure 15** from frame 2 (**Figure 2**) using the same techniques. Using the corr2 method and a threshold of 189 I get **Figure 16** and **17** showing that while it was successfully able to pick out some of the grass I also got a large amount of extra hits.

Figure 16

Figure 17

5) Using Differential Operators to understand extracting features from images.

Figure 18: Magnitude

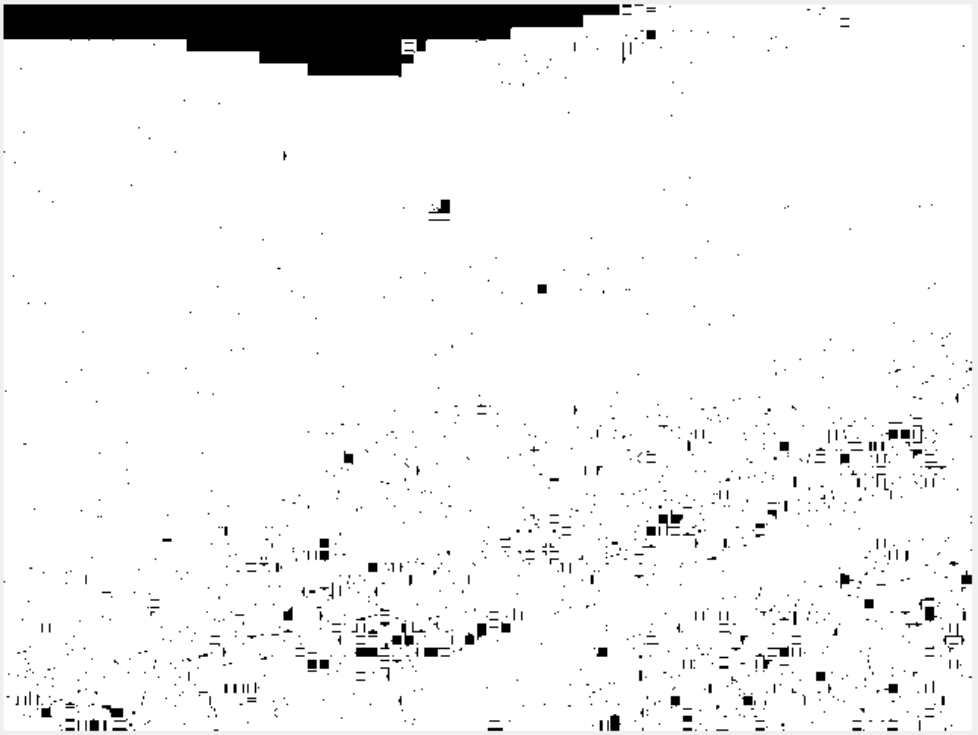
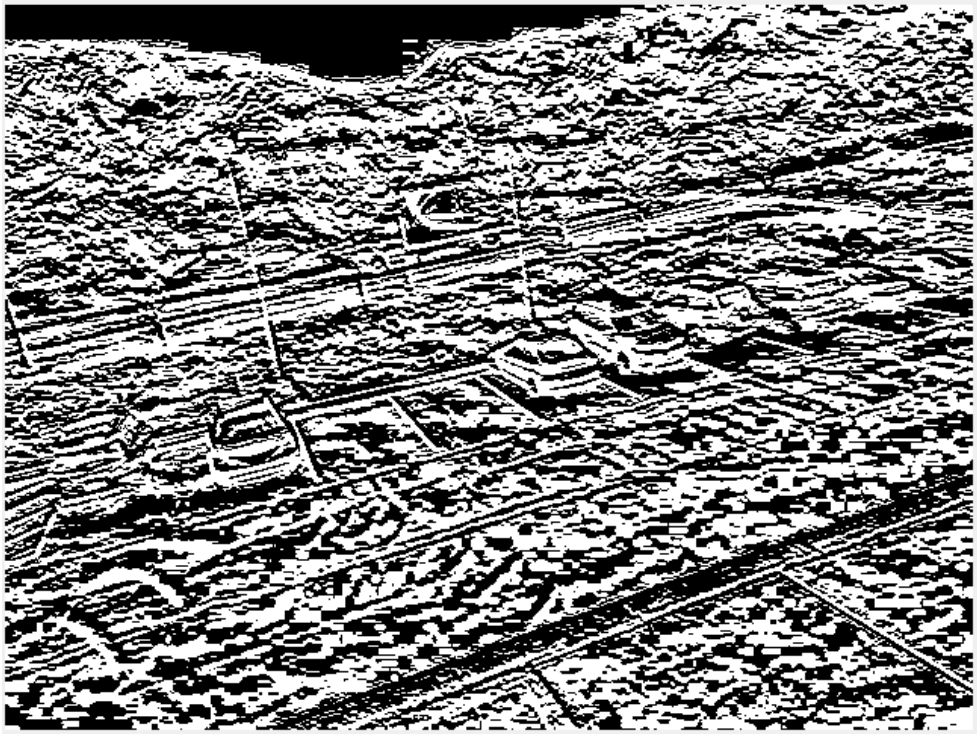


Figure 19: Direction



As a first pass I ran the standard imgradient() function on the grayscale of frame 2 (**Figure 2.1**). **Figure 18** and **Figure 19** show the magnitude and direction of that gradient filter respectively. As we can see in both the magnitude and the direction of the gradient the sky is most prominently segregated. This makes sense as it seems that the sky is saturated and thus there is little to no gradient in that section. However it would seem that the rest of the gradiant both direction and magnitude are fairly difficult to interpret. There are several steep gradients such as the white cars on the black asphalt however the cars themselves also have gradients within them that make distinguishing them difficult. To assit with this several filters can be applied to the grayscale image such as a gaussian blur (**Figure 20**) which results in the gradient direction and magnitude seen in **Figure 21** and **Figure 22** respectively.

Figure 20: Gaussian Blur Sigma = 5

Figure 21: Magnitude

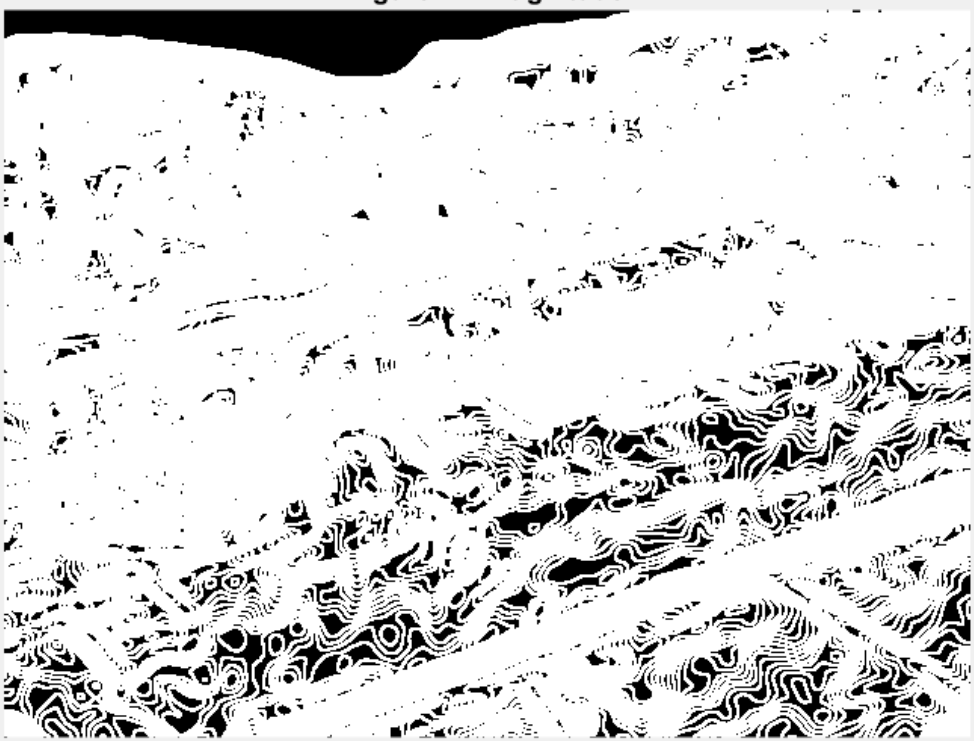


Figure 22: Direction

As can be seen in these smoothed gradients there is a little less of a busy picture and in a way it can be easier to interpret. Along the bottom it is clearer in both the magnitude and direction plots where the edge of the lower right hand sidewalk is. Using a threshold and flood fill connected component on these could potentially find this edge and allow for the segmentation of this section. One oddity to notice is that there s a sort of waving pattern seen in the bottom half of the image. This is likely due to the reflection from the window that the camera look out from. In **figure 2** and **Figure 2.1** we can see the partial reflection of the tripod that the camera sits in the bottom half of the image. The gradient magnitude and direction plots of our gaussian filtered image show this superimposed reflection in those curvy gradients.

Exploring further I used a Laplacian filter (using imfilter() and fspecial('laplacian')) to produce **Figure 23** which when binarized using imbinarize(). The Laplacian operator produces a divergence of the gradient which has a profound edge detection affect on the image. With this technique I was easily able to isolate the various boundaries between cars, the road, trees, etc. With these outlines combined with the detection methods described in question 4 it would be possible to combine cross correlation of a vehicle image to find the image, then the outline found by the Laplacian gradient to isolate a vehicle. When also combined with techniques described in question 3 regarding using the difference between the background and any frame to find movement one could use these three techniques with a flood fill and connected component to isolate moving vehicles.

Figure 23

6) Exploring the use of Fourier Transform to identify features of the image.

To begin with it would be useful to view the Fourier Transform of the entire frame 2 (**Figure 2.2**). **Figure 24** shows the FFT in 2d across the image domain and **Figure 25** shows the fft stretched into 1D and plotted. With these in mind we can now look at sub images.

Figure 24

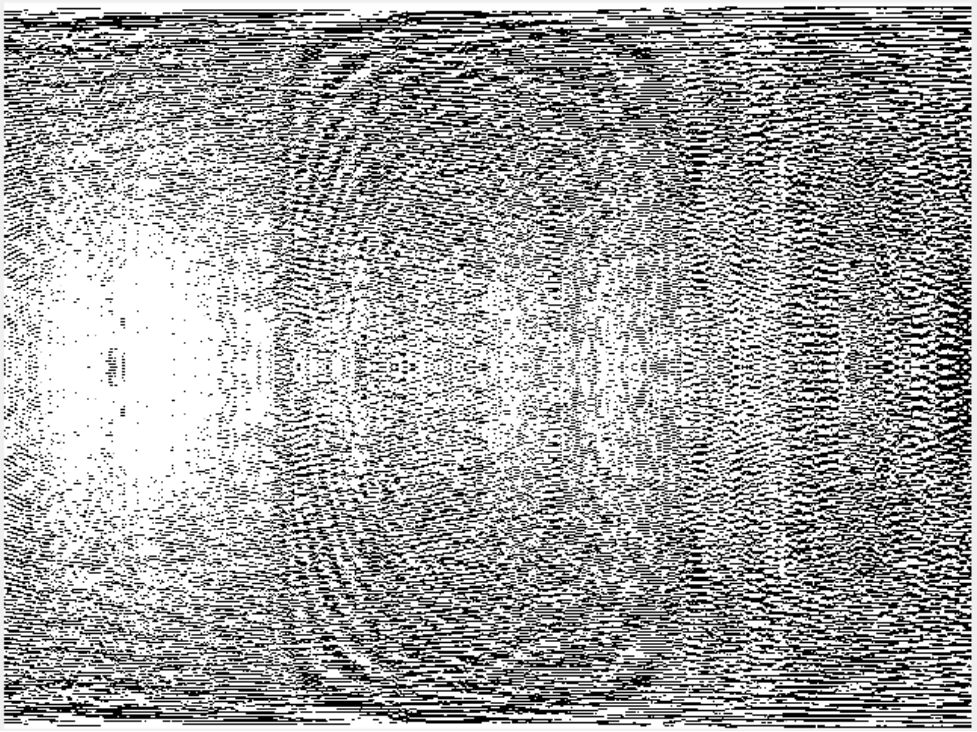


Figure 25

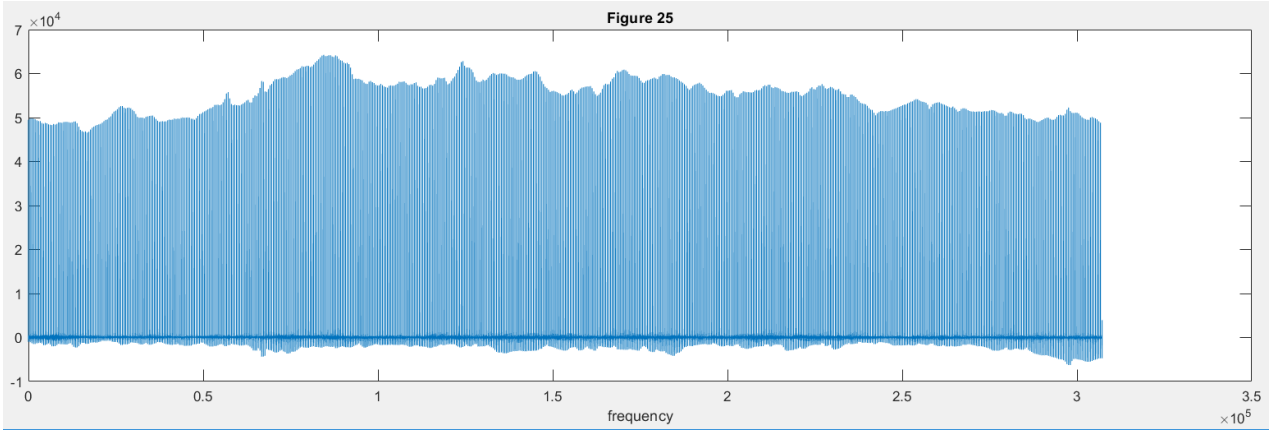
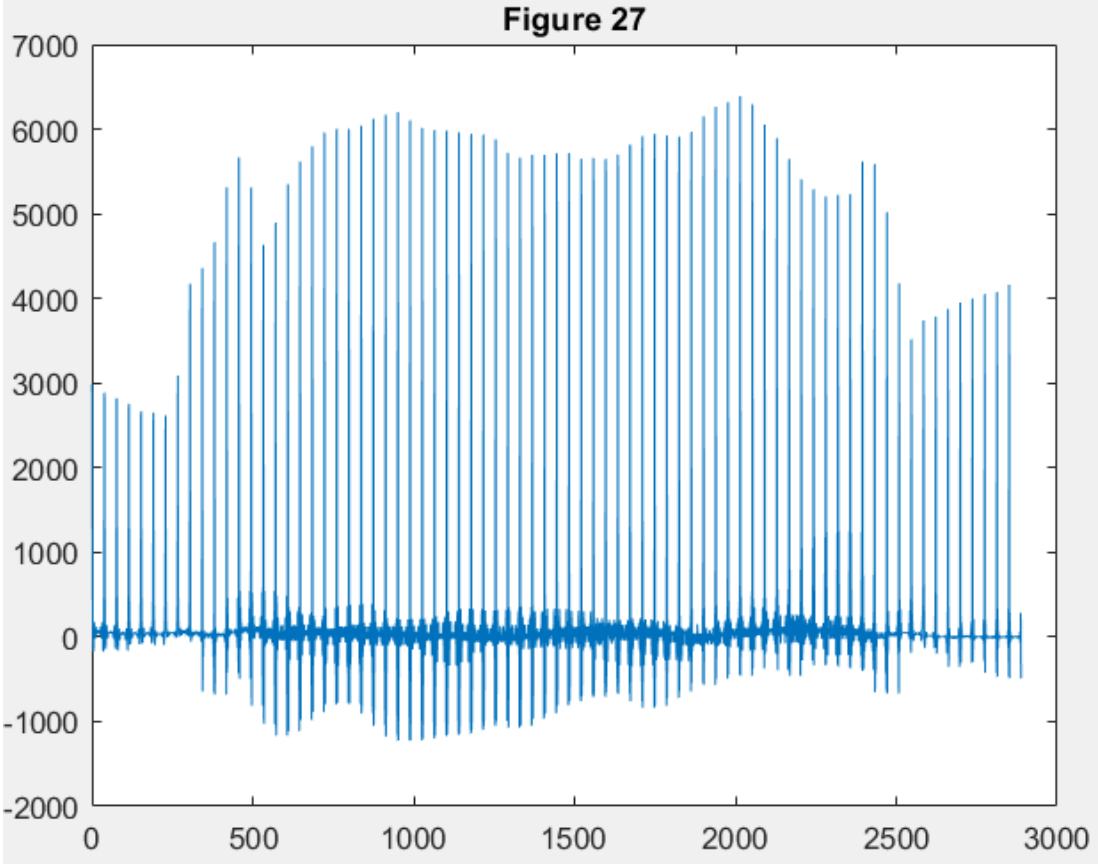
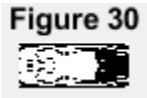
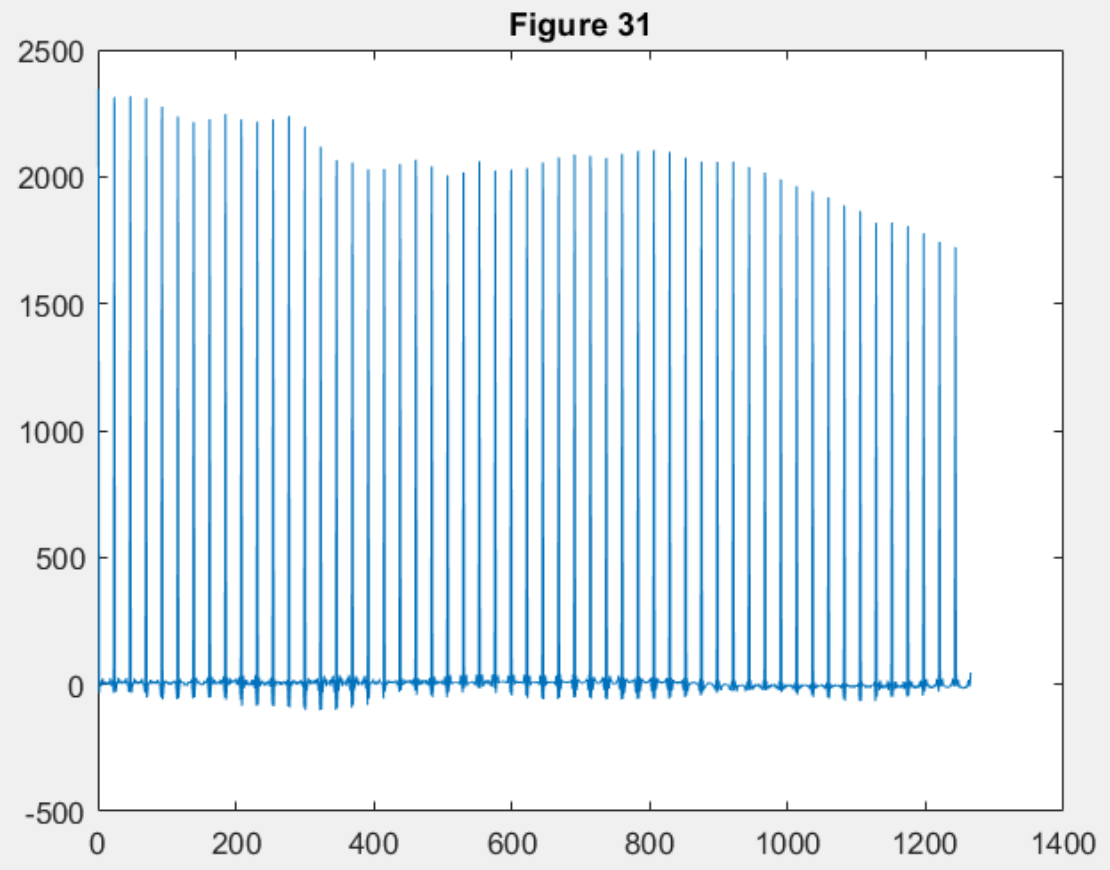


Figure 26

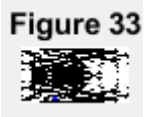


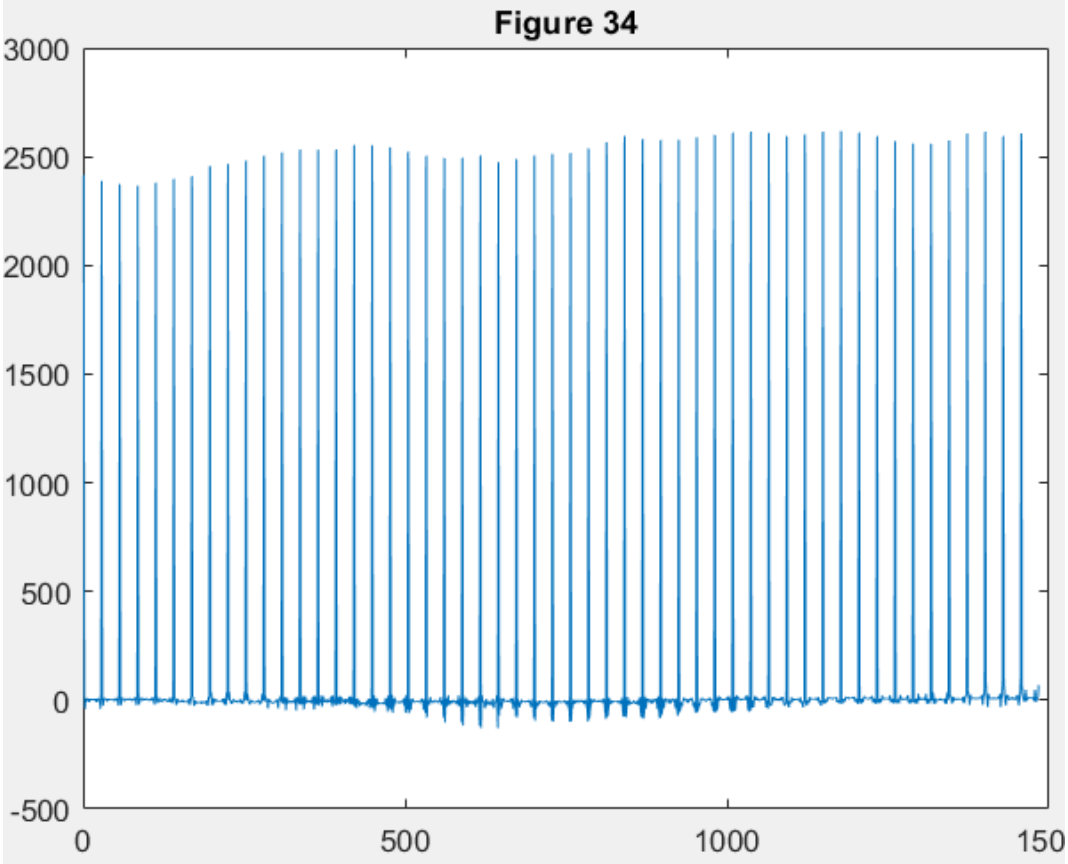
Using **Figure 28** we get the FFP spectrums shown in **Figure 26** and **Figure 27**. Comparing these to those of the entire image we can see that they are distinct and seemingly unique in shape. Lets also look at a few sections of grass to see how they compare.





Using the section shown in **Figure 29** we generate **Figure 30** and **Figure 31** which show distinct shapes and spectrum which are markedly different from those of the car. At the least we can use these to differentiate grass from car, but what about comparing grass to other grass. **Figure 32** shows a second path of grass we will investigate.





While **Figure 33** does look distinct from **Figure 30** for both grass patches, **Figure 34** and **Figure 31** look quite similar. This similarity could be exploited to help determine like surfaces within an image such as the surfaces that make up the hoods of cars or the road. Finding the road could help us find where the moving cars would be.

7) Using HSV to extract features from the image.

First off I used the rgb2hsv on frame 2 (**Figure 2**) to see an initial impression of what the HSV image would look like. It is shown in **Figure 35**. Of interest all of the cars (and a bit of the pavement) turned pink. Right off the bat that is an easy way to distinguish the cars from the rest of the image.

Figure 35

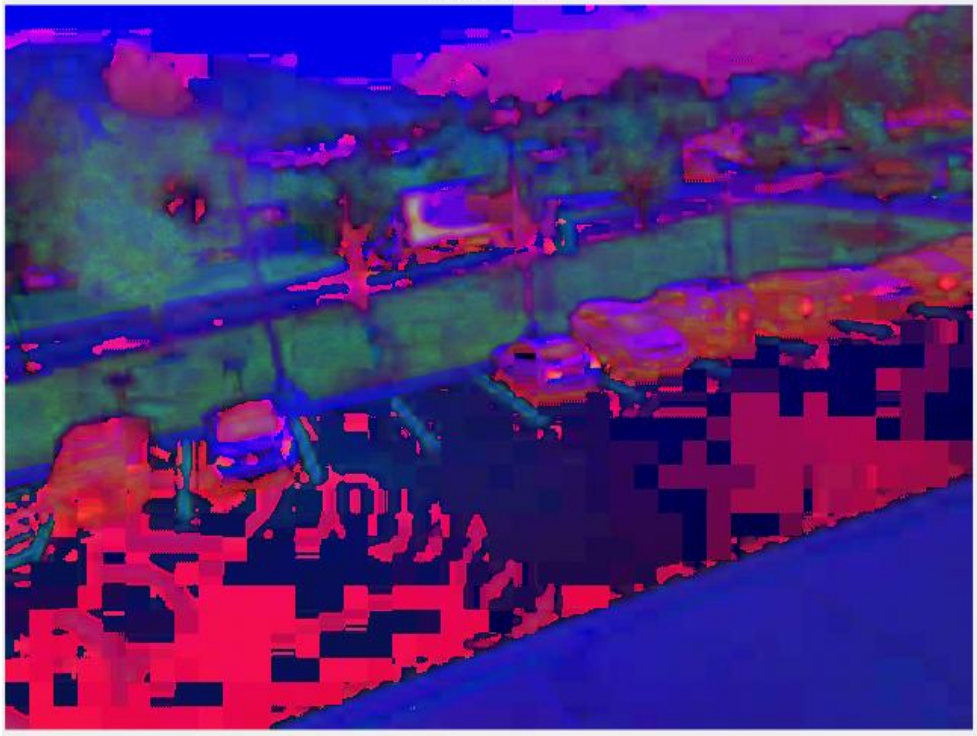


Figure 36

Figure 36 shows the H channel from frame 2. Here we see that the cars are highlighted by an intermediat value. Using a sort of band pass filter where H values between a certain two levels pass we could identify the cars in the image as seen in **Figure 37**. With some adjustment and combination with other techniques this could be an effective vehicle isolation technique.

Figure 37