Camera direction detection and HOV lane detection

YING GAO

Instructor: Yang Cai

1. Introduction

This project deals with live traffic videos and mainly focused on three goals here. First is to capture live video and works on that live video. Second is that since one camera has two directions, so have to find a way to match which direction the camera is facing. Third problem is to detect the HOV lane in highway. HOV lane is a lane has two directions, so it can be very dangerous if there are vehicles driving in two ways at the same time. The former work on live video can deal with vehicle tracking and speeding testing things, but all the algorithms there are not perfectly suitable for HOV lane detection. This project gives a very simple way to solve the problems mentioned above and gives the test results.

2. Algorithm review

2.1 Background subtraction

Most common background subtraction methods like Frame Difference, Approximate Median and Mixture of Gaussian, are all required the camera to be fixed, which is just the case in this project. Frame difference is very sensitive to threshold and can only works on particular speed and frame rate, so it does not work well with high-speed traffic. Mixture of Gaussian has high degree of accuracy but is memory consuming.

2.2 Car detection

Based on the background model, then, can build the mask for each frame to subtract the foreground. Here, used Gaussian filters to blur the image and suppress the back ground first, subtract the background from each frame to get a raw mask. Thus, vehicle movement can be recorded.

2.3 Road detection

There are many ways to detect road from traffic video, but mostly, a specific way would only be useful and precise under certain conditions. One way to detect road is by estimation of the vanishing point associated with the main part of the road, followed by the segmentation of the corresponding road area based on the detected vanishing point. But this approach can only be used in straight roads [1]. When it comes to curvature lanes, the road cannot be detected anymore

2.4 Camera direction identification

The main way to detect camera direction is by using sensors or combines GPS with it, which is expensive. Thus, the camera should carry some metadata which tells us the information it carries, and can decide the current camera direction base on it.

3. Algorithm description

3.1 Background subtraction

In this project, first hundreds frames are used to make background. Approximate median algorithm is used here, which is very suitable in terms of memory efficiency and accuracy. In this project, based on a relative clear frame, convert it form RGB to gray, and then use approximate median method to process the model.

3.2 Car detection

After background subtraction, we can separate foreground from the background. The moving objects (foreground) can be considered as vehicles. Gaussian filter is used here to blur the image and suppress the background. Then subtract background from each frame; we can get a raw mask for the foreground. After comparing contiguous frames, we can consider the nearest moving object to be the same car.

3.3 Road detection

Normally, road detection can be done by record the way vehicles pass by, which is called 'HeatMap'. 'HeatMap' is like a brush that can track vehicles' moving history. In this project, we continue to use this approach, but in a different way. The assumption here is that all the HOV lanes lies between two normal lanes. The system picks up several horizontal lines in the image, and the main idea is to detect the edge of 'HeatMap', which is quite precise and stable to represent HOV lane.

This project uses the same idea with two ways to detect the HOV lane. The method can be described as follows:

(a)

$$For\ each\ I_{left}(x,y)\ , x\ =\ I_{left}(x,y).x$$
$$while\ (x < x_{Max})$$
$$if\ I(x,y) \neq I(x+1,y),\ I_{HOV}(x,y) = I(x,y), break$$
$$else\ x = x+1, continue$$

$$For\ each\ I_{right}(x,y), x\ =\ I_{right}(x,y).x$$
$$while\ (\ x > x_{Min})$$
$$if\ I(x,y) \neq I(x-1,y), I_{HOV}(x,y) = I(x,y), x = x-1, break$$
$$else\ x = x-1, continue$$

(b)

$$For\ each\ I_{middle}(x,y),$$
$$while(x_{left} >\ x_{Min}\ and\ x_{right} <\ x_{Max})$$
$$if\ I\big(x_{right},y\big) \neq I\big(x_{right}+1,y\big), I_{HOV}(x,y) = I\big(x_{right},y\big), x_{right} = x_{right}+1$$
$$if\ I\big(x_{left},y\big) \neq I\big(x_{left}-1,y\big), I_{HOV}(x,y) = I\big(x_{left},y\big), x_{left} = x_{left}-1$$

This approach (b) will find the edge of two lane's 'HeatMap' which are next to the HOV lane, when these is no cars in HOV lane, and will find the edge of HOV lane's 'HeatMap' when there is car drive by. Method (a) will also find the edge of two lanes next to the HOV lane.

However, there's a camera that is very different from others. It's the entrance of HOV lane. There are mainly two reasons to make it unique. First, there are not only two directions in that camera. The cars can come both vertically and horizontally in the image. Second, there are not so

many cars driving by that can use the cars themselves as useful information. So here, we use another approach that is base on the white lines on the road. The basic idea is that, since the camera's shaking will be in a small range, the focus area can be a relatively small one. Then get the white lines in that focus area and binaryzate it. Then start with reference points to test the white line's edge. It can be described as follows:

$$For\ each\ I_{focus}\ \in Area_{focus}$$
$$if\ I(x,y) > \ threshold, I(x,y) = 255$$
$$else\ \ I(x,y) = 255$$
$$For\ each\ I_{left}(x,y),$$
$$if\ I(x,y) \neq I(x+1,y),\ I_{HOV}(x,y) = I(x,y), x = x+1$$
$$For\ each\ I_{right}(x,y),$$
$$if\ I(x,y) \neq I(x-1,y), I_{HOV}(x,y) = I(x,y), x = x-1$$

3.4 Camera direction identification

The first step here is to build a library for both directions' background, and then use the background to compare with each frame to figure out which direction it towards to.

After trying this histogram way, we decide to use another method raise by myself, which is much faster, easier, and is especially suitable for this situation. First, get the background images of camera's two directions. When deciding which image is more likely to the test frame, subtract each pixel to get a difference image; set a threshold, below which the difference value set to 0, and retain the value if above. Then sum up all the difference value of the difference image, and compare it. The smaller one means that the frame is more likely to be this direction. The algorithm can be described as follows:

$$if\ I_t(x,y)\ is\ discribed\ as\ one\ pixel\ in\ time\ t'sframe,$$
$$B(x,y)\ is\ discribed\ as\ one\ pixel\ in\ background\ image,$$
$$D_t(x,y)\ is\ discribed\ as\ one\ pixel\ in\ time\ t's\ difference\ image,$$

Step 1. Get difference image

$$Then,$$
$$for\ each\ I_t(x,y),\ D_t(x,y) = \ I_t(x,y) - B(x,y)$$
$$if\ D_t(x,y) < threshold, D_t(x,y) = 0$$
$$if\ D_t(x,y) > threshold, D_t(x,y) = D_t(x,y)$$

Step 2. Cumulate difference indicator

$$DIFF = \sum_{i=0,j=0}^{i=x_{max},j=y_{max}} D_t(x_i, y_j)$$

Step 3. Compare indicators and decide direction

$$Then,$$
$$if\ DIFF_{with\ bg\_1} < \ DIFF_{with\ bg\_2}, camera\ facing\ background\_1$$
$$else, camera\ facing\ background\_2$$

4. Experiments

4.1 Live video capture

The live video is using RTSP, but the video format is a problem. And since the speed of live video transition is not stable, working on the live frame directly have a lot of problem. In order to solve problem like this, this project use a separate thread first to capture the live video to a buffer, and then work on the video from the buffer.

4.2 Camera direction identification

4.2.1 Histograms

We tried to use each frame's histogram and compare it with background at the first time. Figure 1(a) and 2(a) are the background of two directions' of a camera; figure 3(a) is one frame of the live video that is to be compared to decide the direction. Figure 1(b), 2(b) and 3(b) are the color histogram of the images shown by Picasa 3. Our way is to use the wave characteristic of each histogram, keep the two directions' background as library, compare each frame's wave pattern with it and match the closest one to be the right direction. But the fact here is that, the background images are gray, only the frame is RGB. When making comparison and matching, it would be difficult. Figure 1(c), 2(c) and 3(c) are the two-dimension gray histograms of each frame done by Opencv. It is obvious that the two directions' backgrounds are exactly the same. So we abandon this way. An improved approach is shown in Figure 1(d), 2(d) and 3(d). Here, we use Opencv to get the gray histogram of two backgrounds and GRAY and RGB histograms of test frame. In order to decide which direction it towards to, first step to take is to filter the RGB part in GRAY histogram (upper left figure), and then make pattern comparison. This approach is much better than former two methods, but the accuracy is highly depending on which algorithm is used as pattern match, and is time consuming. Thus, it cannot be used here.

Figure 1(a)

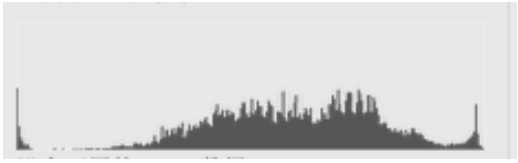


Figure 1(b)

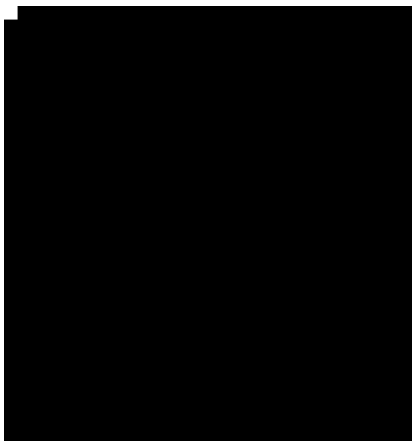


Figure 1(c)

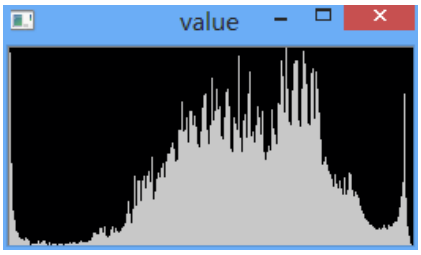


Figure 1(d)

Figure 2(a)

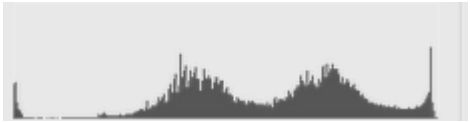


Figure 2(b)

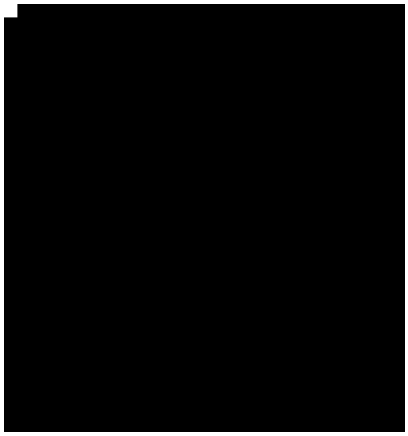


Figure 2(c)

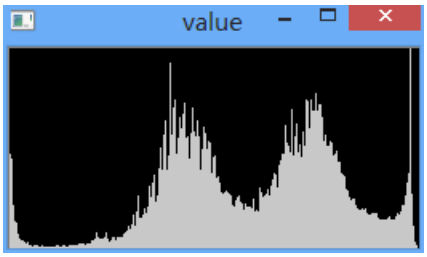


Figure 2(d)

Figure 3(a)

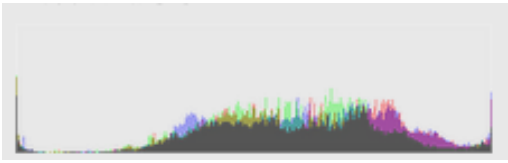


Figure 3(b)

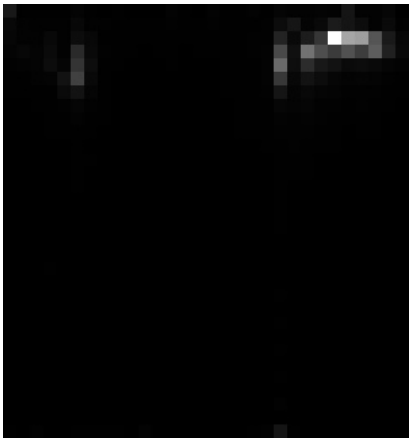


Figure 3(c)

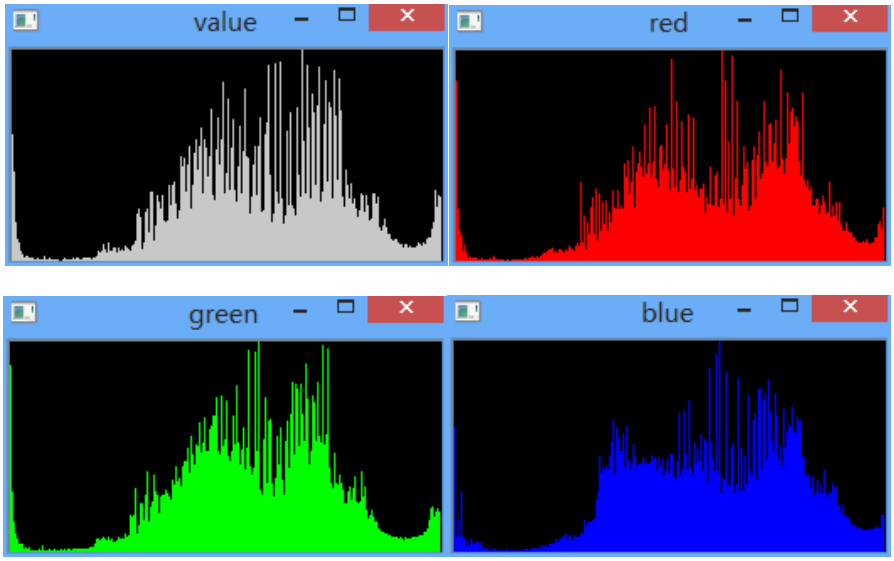


Figure 3(d)

4.2.2 Frame comparison

Here, we chose our own approach finally. When using the same frame as Figure 3(a), the results are as follows. We subtract pixel in frame by corresponding pixel in background, and repeat it pixel by pixel, and do it for each direction. The background image are in gray, which are shown in figure 1(a) and 2(a). The results are shown in Figure 4 and 5. The left ones are the different images after doing step 2 in section 3.4, and the right ones are the images after binaryzation.

Figure 4

Figure 5

Here, we print out the summation of difference value when debugging, one is 3400 and the other is 161159, as shown in Figure 9. These values are described as difference indicators in this project, which shows the difference between a frame in the live video and one direction's background. The smaller the number is, the similar they are. So for the example above, the difference indictor one is much smaller than indicator two, which means the frame is similar to former background. Thus the camera should be facing background one's direction.

```
diffCnt_1 = 3400, diffCnt_2 = 161159
1
```

Figure 6

This algorithm is much simpler than others, since the only thing should keep in sever is the two backgrounds. And it depends on difference rather than color, which is more suitable in this project. Even if there is shake in camera, the neighbors' pixel will still be much similar to the original ones, so the difference between them will still below the threshold. Even there might be some possibility that the difference is above threshold, the relatively difference indicator will still be smaller than the wrong direction's.

4.3 HOV lane detection

4.3.1 Fixed HeatMap

The situation here is that, two main direction lanes have lot of cars, but HOV lane has much less cars compared to them. So when using HeatMap Brush, there will be gap in HOV lane or in both sides of HOV lane. In that case, we can use the HeatMap brush and use the opposite image as a mask to get HOV lane. The two images are shown in figure7. After filters and smooth, the road can be drawn. But the road is fixed, like figure 8 shows.

But this approach is not always true. Because it's hard to decide which HeatMap is true, and should always store the HeatMap in server for each camera and even each direction. Thus, will add the server more load and not efficient. So I discard this method.

Figure 7

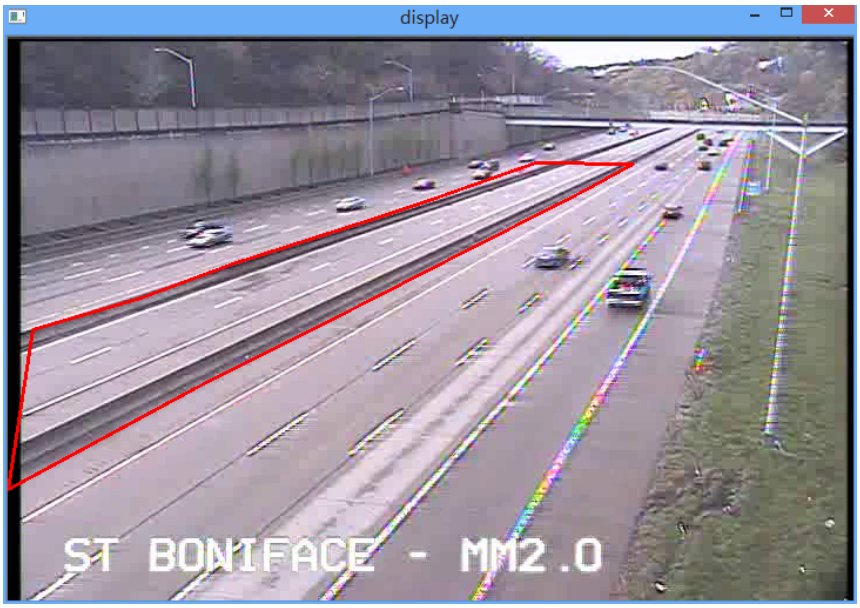


Figure 8

4.3.2 Landmark

Another approach I've tried is using landmark and get each camera's characteristic. By matching the known place with what in the frame, then one can build the position relationship between these landmark and HOV lane to decide where the lane is. This can be called alignment here.
To detect the landmark and match it, there are normally several ways, mostly they are based on colors. But color in this project is not reliable, because the color of a frame is largely based on the environment. Sunny will be difference from cloudy and rainy days. And image will be totally different in the morning and during night. Thus, makes color-based approach not doable in this way. Some of others approaches are based on edge detection. However, when using edge detection to match landmark, the accuracy is mainly base on which method of edge detection being used. Since different way of edge detection will result in different characteristic of the landmark, and totally different values such as threshold to match those landmarks.
Moreover, when using the landmark, we have to store each camera's characteristic with each direction. And there are more than 200 cameras over Pittsburgh, so get all the landmark and the features is really a hard thing. So this approach may be used in several cameras, but is impossible when there are lots of cameras, and would expense lots of time and capital when needs extension.

4.3.3 HOV lane detection by gradually HeatMap

Since in this project, the main feature of the camera is that each camera only has two directions, ad although the camera would shake sometimes, but it will remain in the same place after all. So the position and angle change will be within a small range. Based on these facts, I decide to setup several reference points, and starting from that point, to detect the edge of HOV lane horizontally. This approach do not need servers to store any other images in advance, but just using the live video itself, generating the HeatMap and detecting the point on edge, connect them, will be the HOV lane.

The process is shown in figure 9. The left ones are when HOV lane has no cars and the right ones are when HOV lane has cars. Figure 9(a) shows the HeatMap of these two situations, and figure 9(b) shows how this approach deals with it and get the HOV lane from the changing HeatMap. The read points in the middle of HOV lane are the reference points. Since the camera will only shake in a small range, so the reference points will also seated in HOV lane. After detected which direction the camera is towards to, the sever sets the points. The second step is to detect in two directions starting from those reference points. By compare each pixel with the pixel right or left next to it, seeing whether they are different or not, and set the first different point to be HOV point, we can get the edge of HOV lane. It is shown as yellow points in figure 9(b). It's like we are not blindly regard this picture as a whole, but only detect within each lines. Then connect those HOV points; we can get a HOV lane.

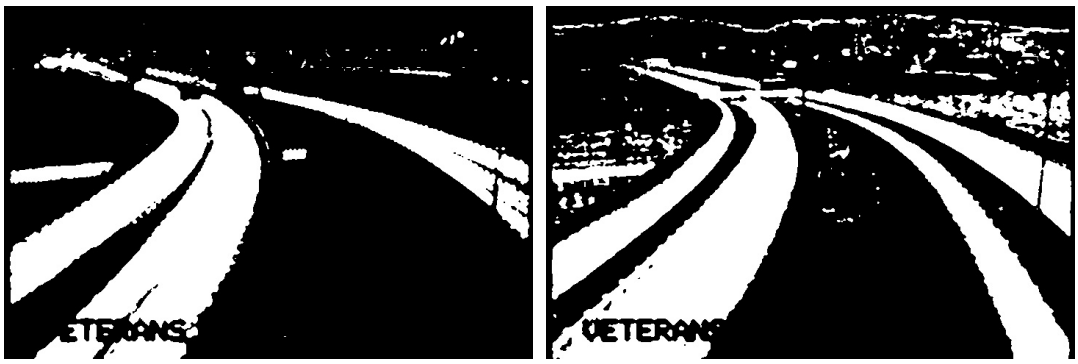


Figure 9(a)

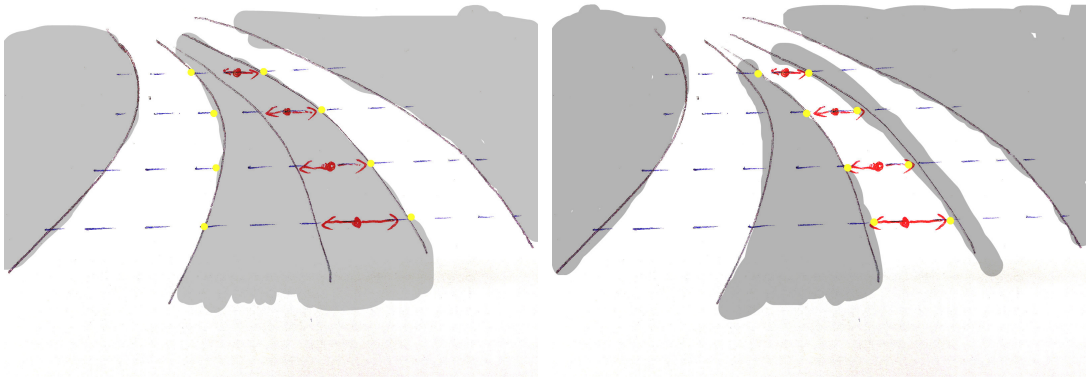


Figure 9(b)

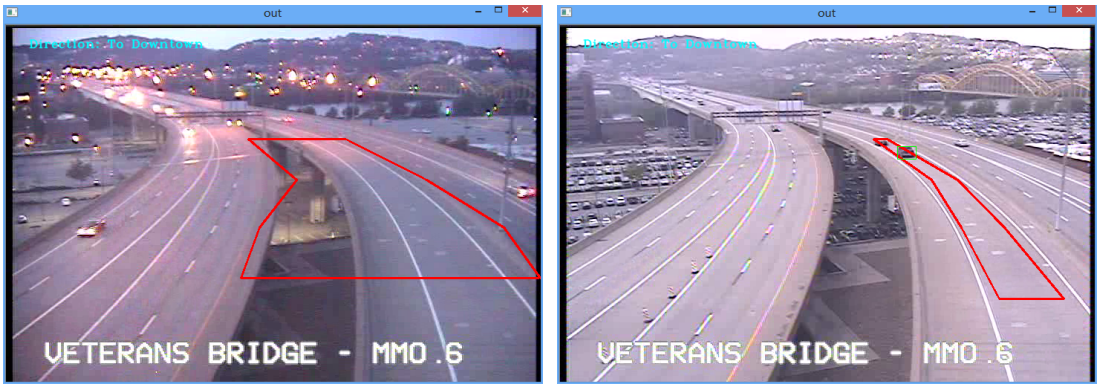


Figure 9(c)

We can see from these pictures, that this approach can also deal with the problem that since the HOV lane has few cars most of time that the HeatMap may vary. When there's no car in HOV lane, the HOV detection will get the edge of lanes next to the HOV lane. But when there are cars driving on HOC lane, the detection result will just be the HOV lane. So in both sceneries, the HOV lane detection method will work. Moreover, the HOV will be refined with more and more cars driving by, since the HeatMap will be cumulating.

5. Tests and results

5.1 Live video capture

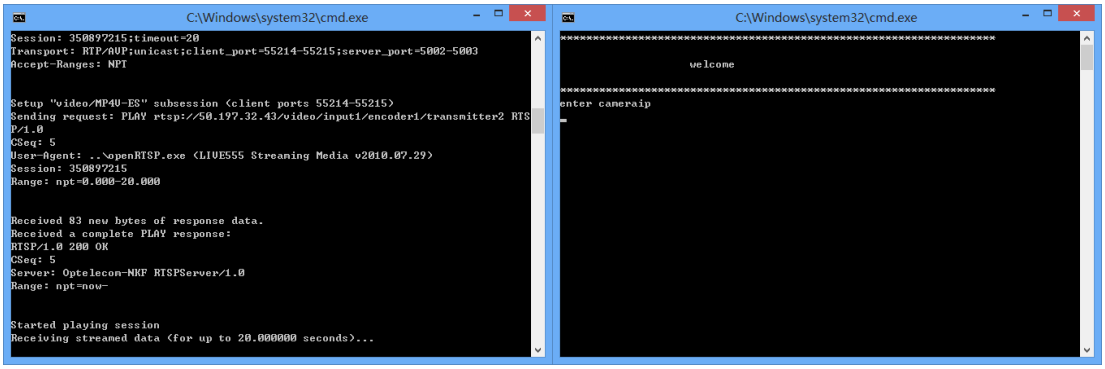


Figure 10

The left picture in figure 10 is using the thread using openRTSP to get the live video into buffer, and figure (b) is the user interface used to get the video in the buffer and display it. User should input which camera to be captured and analyzed in both user interfaces. By using two threads, the frames per second matching problem can be easily solved.

5.2 Direction match

The result of direction match shows on the left upper side of the video, it prints 'To Downtown' or 'From Downtown', and the direction match being updated every 500 frames. The figures and tables below show the test result.

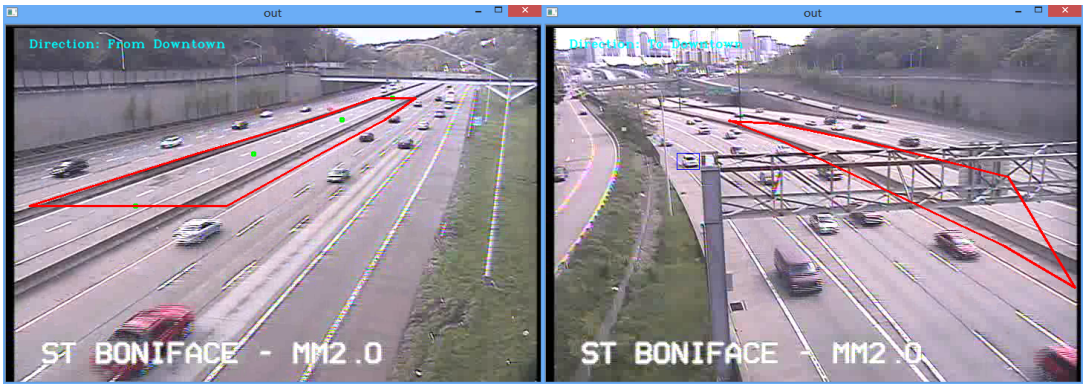


Figure 11

Figure shows the direction match results of one camera's two directions. We can see the results from upper left of the image.

5.3 HOV detection

5.3.1 Using HeatMap

The HOV lane detection has been tested, too. The results are shown in figure 12 to figure 14. As in those figures, the area with red line as boundary is the detected HOV lane. The result continually refined with more and more cars driven by. Left upper image shows the initial HOV detection result, it's not accurate. But it changing and become more and more accuracy as we see from left to right, up to down.

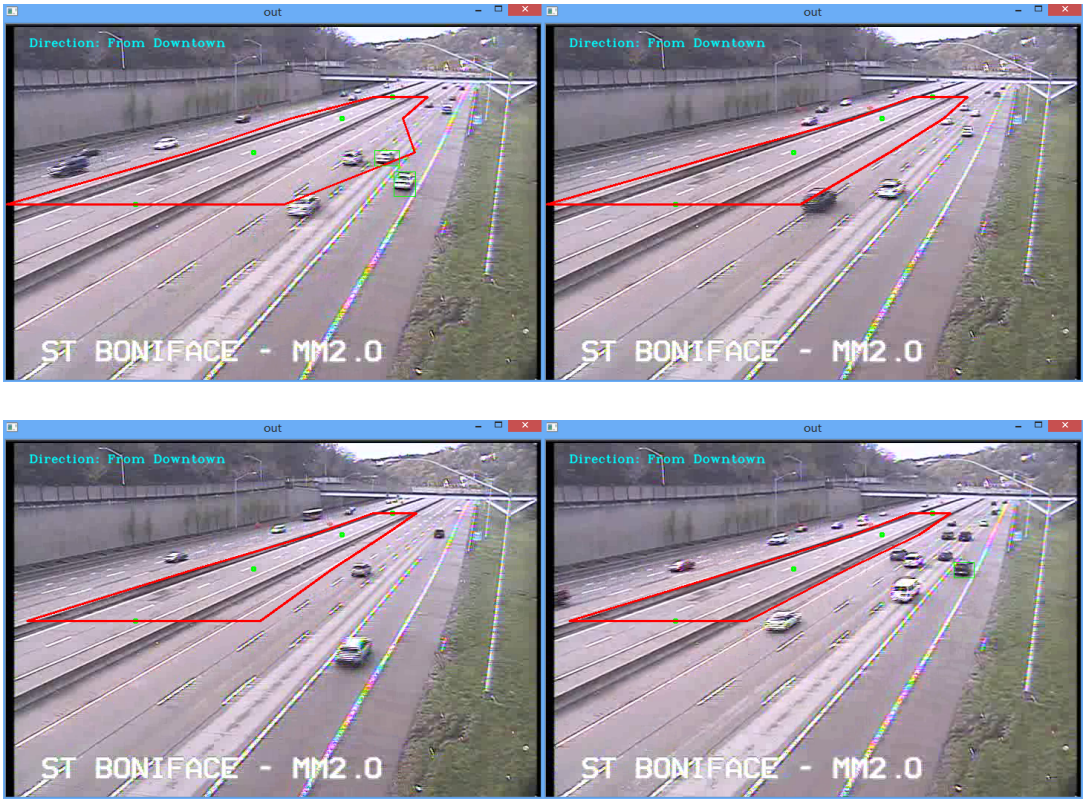


Figure 12

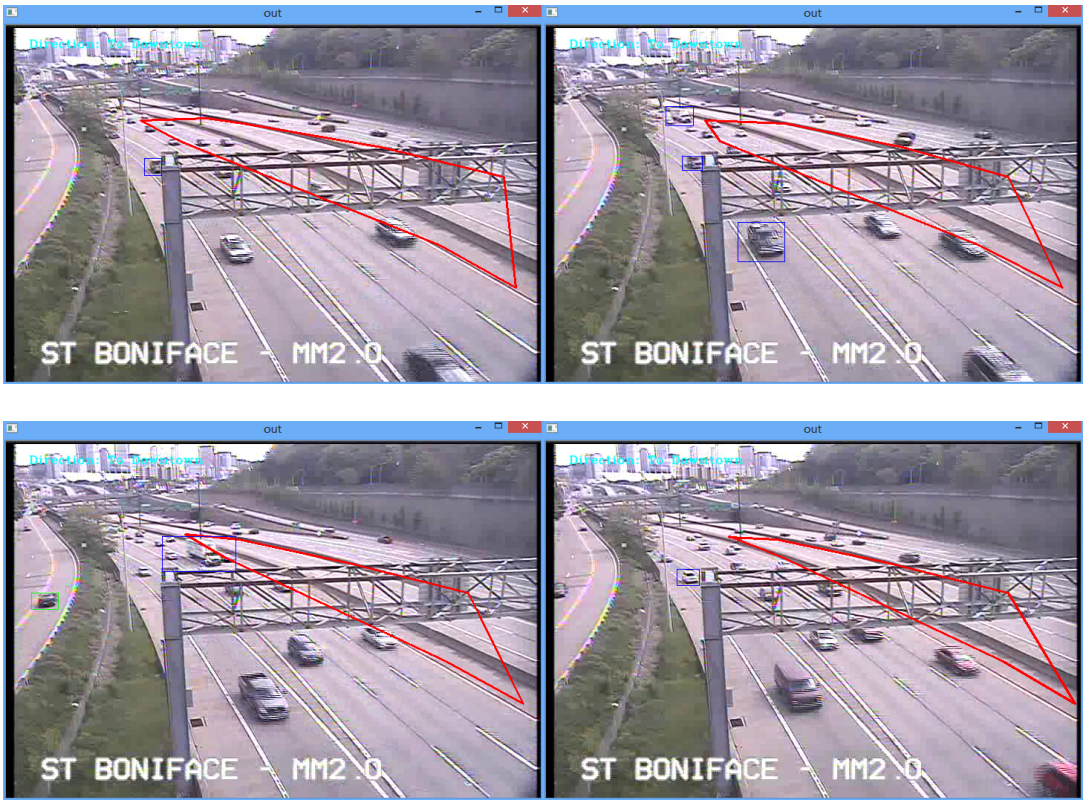


Figure 13

Figure 13 shows the same way. Even when there are some obstacles, the HOV lane detection works, too.

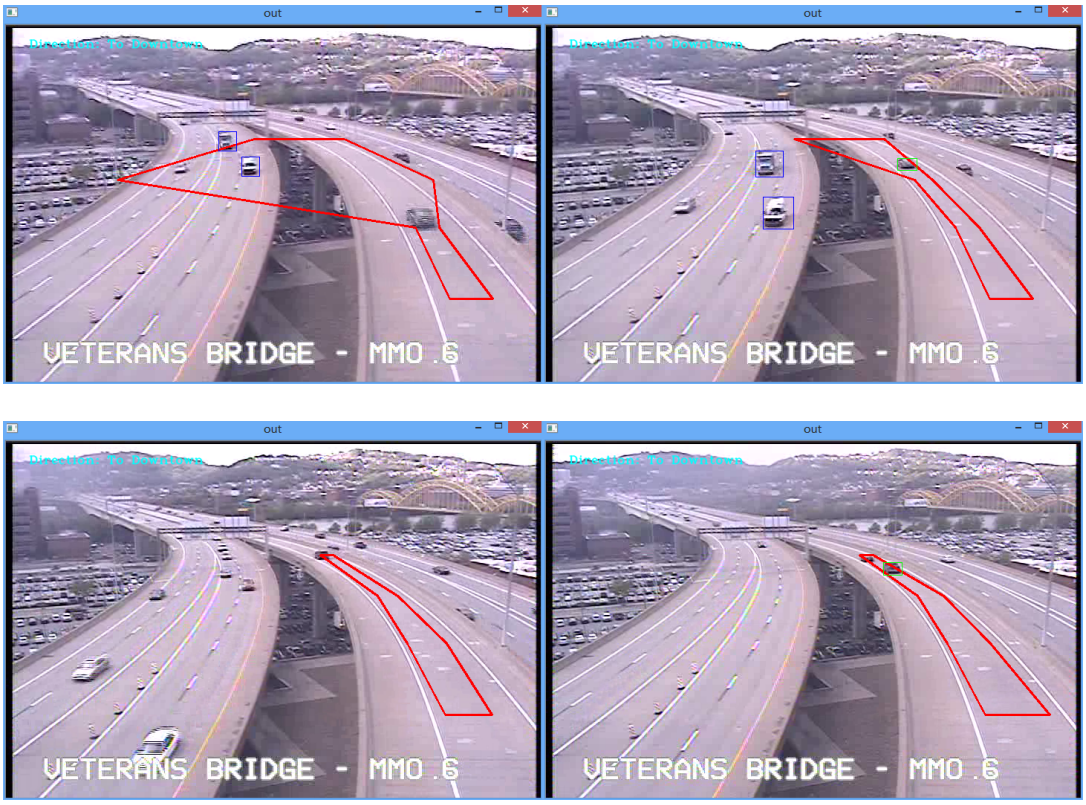


Figure 14

Figure 14 shows one of the most advantage of this approach clearly. When there's no car in HOV lane, the HOV lane detecton result is the edge of next two lanes, and it will narrow down to HOV lane with car driven by. But when there's car in HOV lane, the detection result will suddenly change to the edge of HOV lane, and it will refining by widen the area with cars driven by. Thus, make the HOV lane detection more flexible, no matter it has vehicle or not in that HOV lane, it can always work.

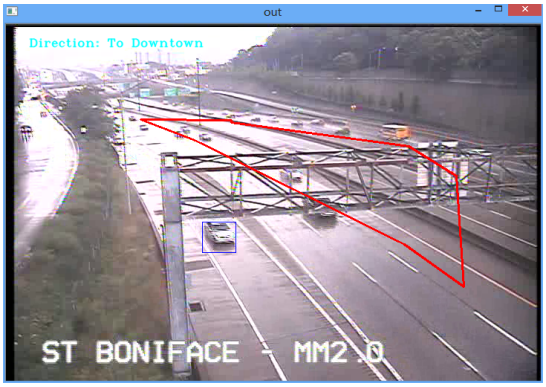
5.3.2 Using white lines

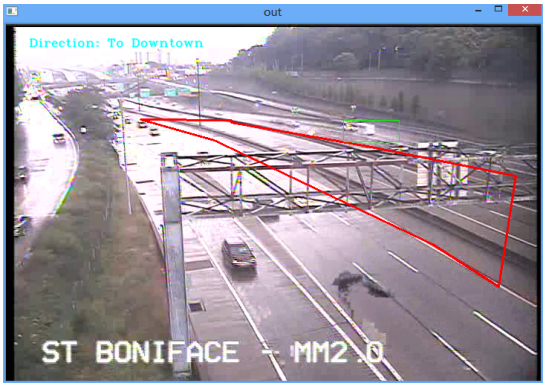
Here is a HOV lane entrance video, which is very different from others, not only because it contains more than two directions' traffic, but also it has little traffic compared to others. So here, the main idea it to have a focus area, and than get the white lines in that area and binaryzate it. Then, start with reference points, to get the edge of white lines. Thus, even if the camera change a little bit direction or angle, the result will still be right.

Figure 15

5.4 Test with different environment

5.4.1 Rainy day





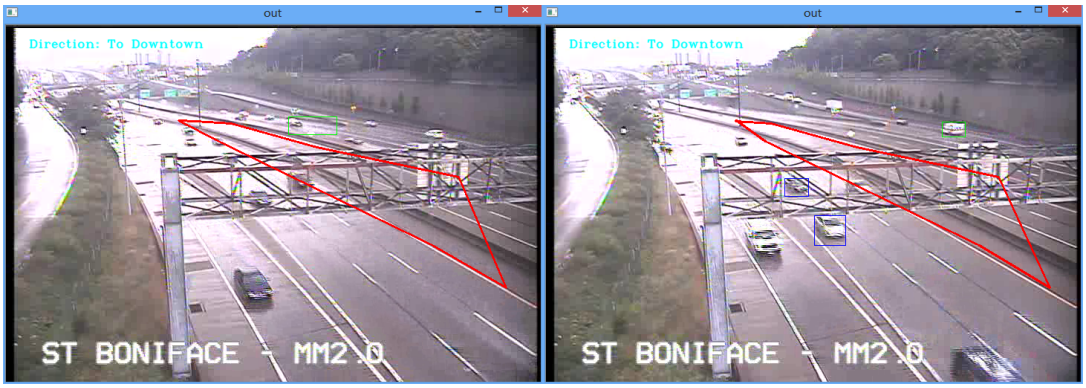


Figure 16

In rainy days, may have poor sight, and the light reflection make it even worse. From figure, we can see both the direction identificaton and the HOV lane detection work well, just as in sunny day.

5.4.2 Night

Figure 17 to figure 19 shows the same camera and how they dealing with HOV lane detection and direction matching during night.

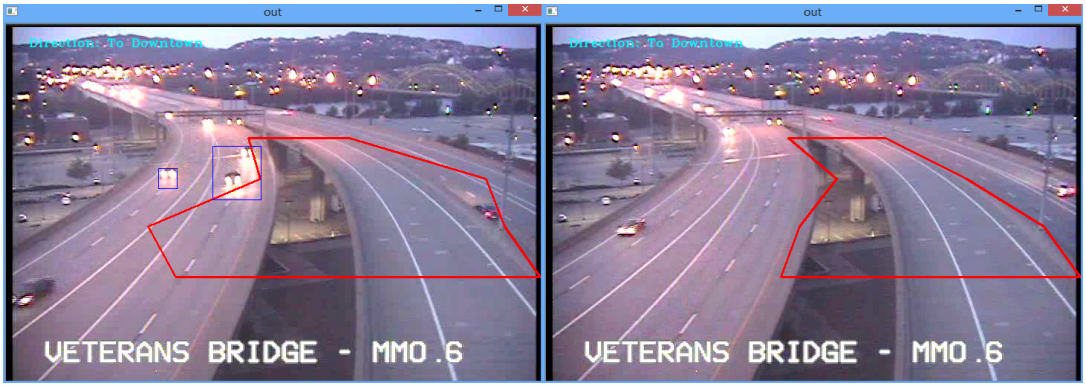


Figure 17

Figure 18

Figure 19

6. Results

5.5 Live video capture

Live video capture works really nice. And the way of using buffer solve the problem of transmission delay and format disunity.

5.6 Direction match

After testing with different environments, it shows that this way of direction match works really well; it can handle all the direction match problems both in day and night. The table below shows the results.

	Daytime	Night	Sunny	Rainy	Shake	Accuracy
Camera1	good	good	good	good	good	100%
Camera2	good	good	good	good	good	100%
Camera3	good	good	good	good	good	100%
Accuracy	100%	100%	100%	100%	100%	100%

5.7 HOV lane detection

For HOV lane detection, we have three cameras here for the project, and each camera has two directions. This project tested HOV lane detection for each direction, expect that there are two directions have no HOV lane there. The table below shows the testing results.

	Curve	Daytime	Night	Sunny	Rainy	Shake	Accuracy
Camera1 – to Downtown	HOV Entrance	good	Good but when it's really dark, hard to detect HOV lane	good	good	good	80%
Camera1 – from Downtown	No HOV lane	No HOV lane	No HOV lane	No HOV lane	No HOV lane	No HOV lane	
Camera2 –	Curve	Good,	Good, HOV	Good,	Good,	Good	95%

to Downtown	lane	HOV has cars or not influence results	has cars or not influence results	HOV has cars or not influence results	HOV has cars or not influence results		
Camera2 – from Downtown	No HOV lane	No HOV lane	No HOV lane	No HOV lane	No HOV lane	No HOV lane	No HOV lane
Camera3 – to Downtown	Straight lane	Good	Good	Good	Good	Good	100%
Camera3 – from Downtown	Straight lane	Good, seldom influent by car's distance	Good, seldom influent by car's distance	Good, seldom influent by car's distance	Good, seldom influent by car's distance	Good	90%

As we can see from the table, the HOV lane detection can work well in both straight lane and curvature lane. But there are still some situations affects the results. The distance between cars in adjacent lane may sometimes affect the results since it will make the HeatMap different. Also, sometimes the cameras may shake a little and results in the HeatMap to be totally a mess. This problem is solved by tracking the HeatMap and re-cumulates it when it detects lots of white in image.

Reference

[1] Hui Kong, Member, Jean-Yves Audibert, Jean Ponce, Willow Team and Ecole Normale, "General road detection from a single images", 2009

[2] Paolo Lombardi, Michele Zanin, "Switching Models for Vision-based On-Board Road Detection", Proceedings of the 8th International, IEEE Conference on Intelligent Transportation Systems, 2005

[3] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary Brads, "Self-supervised Monocular Road Detection in Desert Terrain"

[4] Alberto Broggi, Simona Berte, "Vision-Based Road Detection in Automotive Systems: A Real-Time Expectation-Driven Approach", Journal of Articial Intelligence Research 3, 1995