2015-04-24

Lane Departure and Front Collision Warning System Using Monocular and Stereo Vision

Bingqian Xie
Worcester Polytechnic Institute

Lane Departure and Front Collision Warning System Using Monocular and Stereo Vision

by

Bingqian Xie

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical and Computer Engineering

by

April 2013

APPROVED:

Professor Xinming Huang, Major Thesis Advisor

Professor Lifeng Lai

Professor Xiangnan Kong

Abstract

Driving Assistance Systems such as lane departure and front collision warning has caught great attention for its promising usage on road driving. This, this research focus on implementing lane departure and front collision warning at same time. In order to make the system really useful for real situation, it is critical that the whole process could be near real-time. Thus we chose Hough Transform as the main algorithm for detecting lane on the road. Hough Transform is used for that it is a very fast and robust algorithm, which makes it possible to execute as many frames as possible per frames. Hough Transform is used to get boundary information, so that we could decide if the car is doing lane departure based on the car's position in lane. Later, we move on to use front car's symmetry character to do front car detection, and combine it with Camshift tracking algorithm to fill the gap for failure of detection. Later we introduce camera calibration, stereo calibration, and how to calculate real distance from depth map.

Acknowledgments

I would like to express my gratitude to my advisor, Professor Xinming Huang, who gave the chance to dig into Computer Vision world.

Thanks to my family and all of my friends for supporting me all the time.

Contents

List of Figures

Chapter 1

Introduction

As the continuous growing of the amount automobiles, traffic accidents have been a great issue all over the world. Driver assistance system thus gains great popularity, which aims at helping drivers to be better concentrate when driving and giving proper warning when any danger might occur. A great amount of accidents are caused by tired and drowsy drivers, who did not notice the car is shifting to other lanes or coming too close to the front car. Among driver assistance features, two important feature, lane departure warning system and front collision warning system are developed to prevent drivers from ignorance of danger. In this thesis we focus on performing lane departure warning and front collision warning while driving on highways, for that lane changing and merging usually require special attention and unintentional lane departure behavior is extremely dangerous. Different approaches are applied to do lane detection. These approaches can be divided to two parts, model based and feature based. McCall et al.[1] proposed "video-based lane estimation and tracking" (VioLET) system which used steerable filters to do lane-marking detection. Others use Hough Transform to do feature based lane detection. Wei et al.[2] combined Hough Transform and Otsu Threshold method together to get bet-

ter performance of lane detection and put the design on Kintex platform[3]. Hough Transform is a classic algorithm to detect straight lines with good performance and fast speed, thus We chose Hough Transform as the algorithm for detecting lanes and use this information to decide if the car is doing lane departure or not. As for car detection methods, we limit our discussion to methods used for cameras that are mounted on moving cars. This limitation is of great importance since solution and method would be totally different if cameras are mounted still. There can be separated to two ways, using monocular vision or using stereo vision[4, 5]. Different kinds of pre-owned knowledge are used to detect cars, and the knowledge is usually achieved from observation. Color information, shadow under the car, textures, corners and symmetry property of the back of car are very useful to detect front car. Others combine features of cars and machine learning technique to detect the shape of the car, in which HOG feature is very useful. The problem with this kind of method is they are not fast enough to be real-time. Disparity map and perspective view are often used when we have stereo vision. Toulminet el al.[6] use stereo vision to get sparse disparity map which includes only corners' disparity information and combine it with car's symmetry information. Chapter 2 introduces Ostu's Method and Hough Transform algorithm, and then use results from Hough Transform to decide if the car is doing lane departure or not. Chapter 3 presents the idea of detecting car position using symmetry information, and combine it with Camshift tracking algorithm to keep tracking the car position in video stream. Chapter 4 introduces distance calculation method, calculating distance using monocular vision and apply disparity map using stereo vision. Chapter 5 presents conclusion of our achievements and possible improvement in the future work.

Chapter 2

Lane Detection and Lane Departure Warning

2.1 Introduction

The core algorithm for doing lane detection is Hough Transform, which performs on binary images and gives out the gradient and intercept of straight lines on the image. To get better result from Hough Transform, pre-process the image and get a good binary image is very important. Flow chart of processing image and send binary image to Hough Transform block is shown as below (Figure 2.1).

The input image from camera is RGB form. First step is to apply a Sobel Filter on the grayscale image to sharpen the edges after an RGB to grayscale conversion. This operation helps highlight the lane marks and front vehicle boundary while eliminating dirty noise on the road[3]. The operator of Sobel filter is

$$G = G_x^2 + G_y^2 \tag{2.1.1}$$

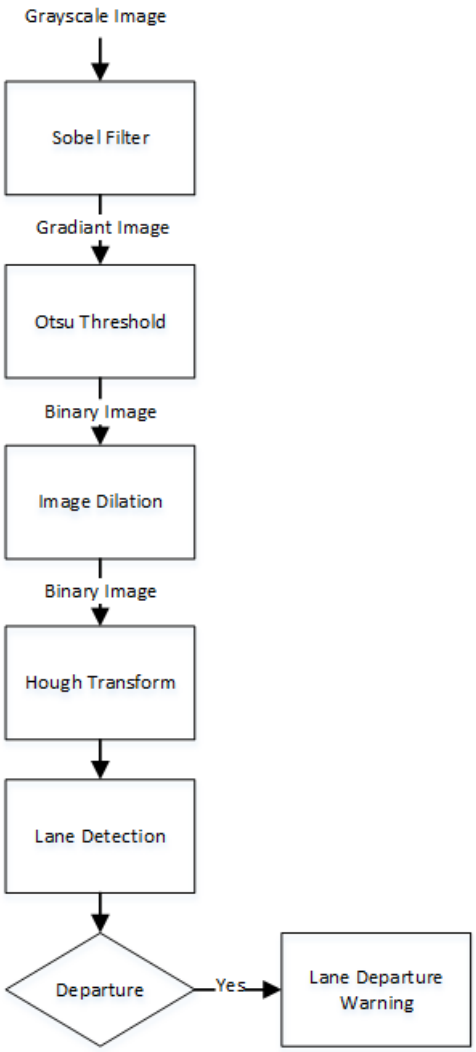


Figure 2.1: Particles in the grid with random velocities

where G_x is Sobel kernel in x direction and similarly G_y in y direction.

$$G_x = \begin{bmatrix} 0.125 & 0 & -0.125 \\ 0.25 & 0 & -0.25 \\ 0.125 & 0 & -0.125 \end{bmatrix} \tag{2.1.2}$$

Next step is image dilation, which is a basic morphology operation. It usually uses a structuring element for probing and expanding shapes in a binary image. In this

design, we apply image dilation to smooth toothed edges introduced by Sobel filter, which can assist the subsequent blocks to perform better[3]. Processed binary image is sent to Hough Transform block and the output is some formulas of straight lines. One good property of Hough Transform that we could know the amount of points that is on the same straight line. With this information, we could filter some straight lines with very few points (called votes) on it. To get better performance of every frame the system takes in, we optimize each frame's Hough Transform result using history information. It is very common that for some frames Hough Transform will give out no straight lines or wrong lines; in this situation, history information about where lines are is used for this frame in order to fill the miss lane information. After getting lane information in each frame, we would decide if the car is doing lane departure or not. If the car is doing lane departure, considering that lane position changes very quickly during that time, we would stop using history lane position to restrict the Hough Transform results. More detail will be discussed below.

2.2 Otsu Threshold

The accuracy of boundary information which is acquired from Hough Transform is highly dependent on the quality of the image it takes in. Hough Transform takes binary image as input image and produce straight line equation as output. A binary image that contains sufficient details is the prerequisite to accurate line acquisition from Hough Transform. Thus we choose Otsu's threshold method to perform the reduction of a grayscale image to a binary image.

A ' perfect ' binary image means high signal-to-noise, preserving desired information while abandoning useless data to produce more accurate results and ease computational burden. For Lane Departure Warning and Front Collision Warning

systems, the desired information is lane marks of the road and vehicles in the front. Therefore, we cut off the top part of every image and crop the area of interest to the bottom 320 rows of a 720P picture. All the subsequent calculation is conducted on this region of interest (ROI). When converting from gray scale to binary image, a fixed threshold is often not effective since the illumination variance has apparent influence on the binarization[3].

Otsu's threshold is a dynamic and adaptive method to obtain binary image that is insensitive to background changes[3]. According to Otsu's Method[7], the whole image was separated as two classes: foreground and background. Average value of the image could be expressed as:

$$u(t) = w_0(t)u_0(t) + w_1(t)u_1(t) \tag{2.2.1}$$

Where u_0 is average value inside background class and u_1 is average value inside foreground class. w_0 and w_1 are probabilities of two classes. t is the threshold that separate two classes. The basic idea of Otsu's Threshold is that traverse all t to find one that maximizes the variance between two classes, which is equal to minimize the variance inside each class.

$$g = w_0(u_0 - u)^2 + w_1(u_1 - u)^2 \tag{2.2.2}$$

The implementation of Otsu's Threshold is as follows. The first step of Otsu's method is to calculate probability distribution of each gray level as in 2.2.3.

$$p_i = n_i/N \tag{2.2.3}$$

where p_i and n_i are probability and pixel number of gray level i. N is the total pixel number of all possible levels (from 0 to 255 for gray scale) in the given grayscale

image. The second is to step through all possible level t to calculate $\omega^*(t)$ and μ^*

$$\omega^*(t) = \sum_{i=0}^{t} p_i \tag{2.2.4}$$

$$\mu^* = \sum_{i=0}^{t} ip_i \tag{2.2.5}$$

The final step is to calculate between-class variance

$$(\sigma_B^*(t))^2 = \frac{[\mu_T^*\omega^*(t) - \mu^*(t)]^2}{\omega^*(t)[1 - \omega^*(t)]} \tag{2.2.6}$$

In Figure 2.2, (a) is the origin image, after discarding upper half image and apply Sobel Filter, we get grayscale image as illustrated in (b). (c) is binary image obtained by apply Ostu's Threshold to the grayscale image.

2.3 Hough Transform

2.3.1 Hough Transform rationale

Hough Transform is widely used as a proficient way of finding lines and curves in binary image. Since most lines on the road are almost straight in pictures, we will mainly discuss the way of finding straight lines in binary image using Hough Transform. Given a binary image, on which has a straight line l:

$$y = kx + b \tag{2.3.1}$$

Where k is the slope and b is the y-intercept. Suppose point A(x1,y1) and point B(x2,y2) are on line l, they both satisfy

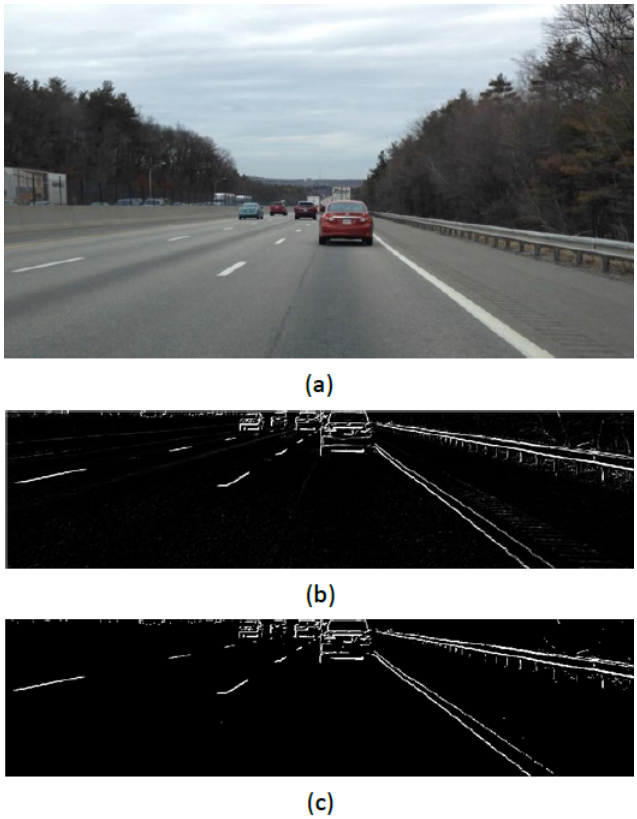


Figure 2.2: Illustration of Sobel filter and Otsu's threshold binarization. (a) is the original image captured from camera, (b) and (c) are ROI after Sobel filtering and binarization, respectively.

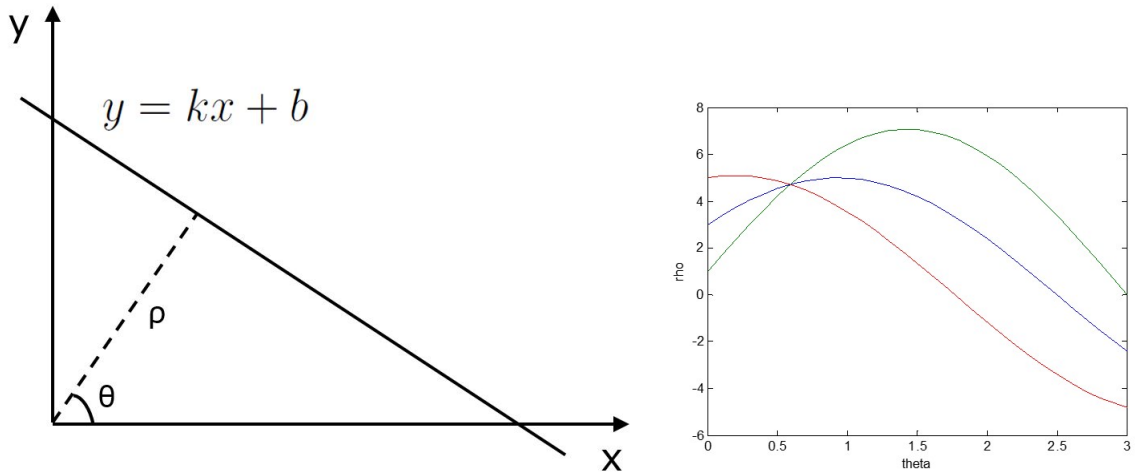
$$\begin{cases} y_1 = kx_1 + b \\ y_2 = kx_2 + b \end{cases} \tag{2.3.2}$$

Rewrite both equations to

$$\begin{cases} b = -x_1 k + y_1 \\ b = -x_2 k + y_2 \end{cases} \tag{2.3.3}$$

Letting point A(x1, y1) and B(x2, y2) and known parameters, the above two lines will have an intersection point in parameter space, thus we can calculate b and k in parameter space. Yet we cannot get vertical lines using this method, so line l could be represented in polar coordinate system as

$$\rho = x \cos\theta + y \sin\theta \tag{2.3.4}$$



(a) Straight line representation in Cartesian coordinate system

(b) family of lines for given (x,y) pairs in plane theta-rho

Figure 2.3: Line representation in Cartesian coordinate system and polar coordinate system

For point $A(x_1, y_1)$there is a curve of ρ and θ that satisfies

$$\rho = x_1 \cos\theta + y_1 \sin\theta \tag{2.3.5}$$

and same thing for point $B(x_2, y_2)$:

$$\rho = x_2 \cos\theta + y_2 \sin\theta \tag{2.3.6}$$

If point A and B are on the same straight line, there will exist a pair of ρ and θ that satisfy both equation, which means two curves have an intersection point. As more points on the same line are added to polar system, there should be one shared intersection point between these curves. What Hough Transform does is to keep track of intersection points between curves and the intersections with big voting imply that there is a straight line.

2.3.2 Implementation

To implement Hough Transform, we constrain value of $\theta \in [0, \pi]$ and $\rho \in [-R, R]$, where R is the largest distance allowed from line to the origin. After turning the image to binary image, we do following calculation for every white point (or every black point). Here we set angle resolution to one degree, so for certain ρ value in accumulator, we need 180 bins. The resolution of ρ is also set to one, which means for each ρ 2R bins are needed. Form a two dimensional matrix that has $2R$ rows and 180 columns called accumulator, each element is regarded as a bin for voting a particular pair of ρ and θ . Our goal is to find which ρ and θ pair has largest amount of points on it.

For a certain point (x, y), calculate different θ from 0 to π at interval of one degree and get corresponding ρ. Increment this bin by one. After all pixels have been processed, we get a two dimensional array accumulator that contains votes for each bin.

To further process the accumulator, we need to eliminate those lines that are very close to each other; this shows in accumulator as the bin value are very similar

to nearby bins. Getting several straight lines that are very close to each other on the image is useless and waste of computing time, so we set a 3×3 window and only choose one local maxima as the 'representative' of these similar lines. Given a threshold of number of votings, we would examine each bin in the accumulator to see (1) if it's larger than the threshold and (2) if it's the local maxima in its window. If it satisfies both conditions, then set the rest in the window to zero, if not, continue to examine next bin.

After eliminating similar straight lines, qualified (ρ, θ) pairs remain in the accumulator and the value stands for the number of points that are on this straight line. The rest of the bins in the accumulators is set to zero.

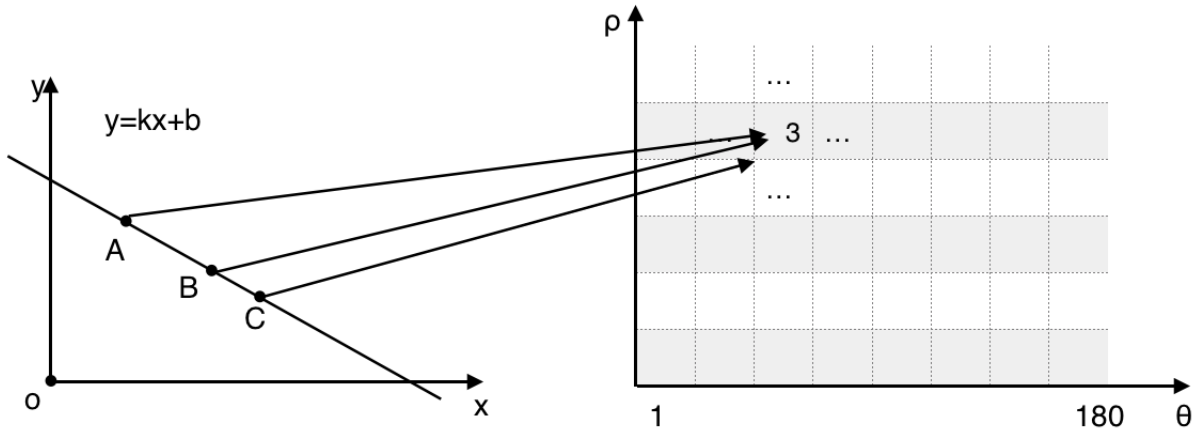


Figure 2.4: Points from same straight line go to same bin in accumulator

The algorithm is as follow:

1. Create a two dimensional array called accumulator
2. For every point in binary image, **do**
   a. **for** theta=1 to 180, compute corresponding rho
   b. increment bin (rho, theta) by one in accumulator
3. check every (rho,theta): **if** this bin is local maxima and has enough votes, save (rho,theta)

2.3.3 Using History Information to Improve HT results

Hough Transform sometimes return false positives due to various reasons: blurring of the image, distraction of passing cars and insufficient amount of dotted line shown up in the image. To fill the missing lane on the image, we use history information to help with checking and correcting Hough Transform results.

After getting two lines' equation using Hough Transform, we compare it with the average line position from past five frames. Since images are taken at 30 frames per second, lane position in less than 1/6 second shall not change dramatically. Thus we could use history lane position to decide if the results are good or not. If it is real line, it shall be used as the new lane equation. If it does not qualify, then we need to use history information to help us get a lane equation that should be very close to the true lane.

First compare it with the weighted average of the past five frames' line equation, if they are similar, we regard this frame as good and also save line information into past five frames line information for future use. If lines are missing or far away from what is expected based on history information, history line information is directly used to substitute the wrong/missing ones. This method is easy to apply and is of high accuracy and calculation speed, which ensures to keep the whole system as real time as possible.

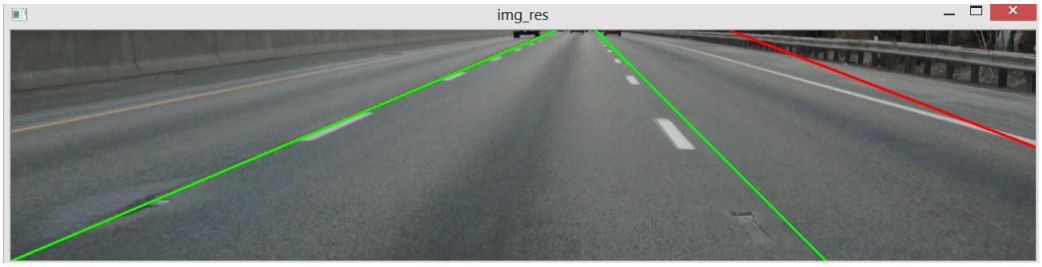


Figure 2.5: History information help detect right lane position in this image: red line is the raw data from Hough Transform, right line is the corrected one using history lane position

2.4 Detect Lane Departure

When car is driving normally on the road, two lanes on the image often appear as Figure 2.6. Left and right lanes are symmetrical to each other along a vertical line, and their angles are similar. If the car is making lane departure (Figure 2.7) we notice that one lane will appear vertical and the other will appear more horizontal. If the car is moving to left, left lane turns to vertical; if the car is moving to right, right lane shall appear vertical. We could use this observation to utilize the angle of two lanes to detect if a car is doing lane departure. Furthermore, we could decide if a car is doing left or right lane departure.

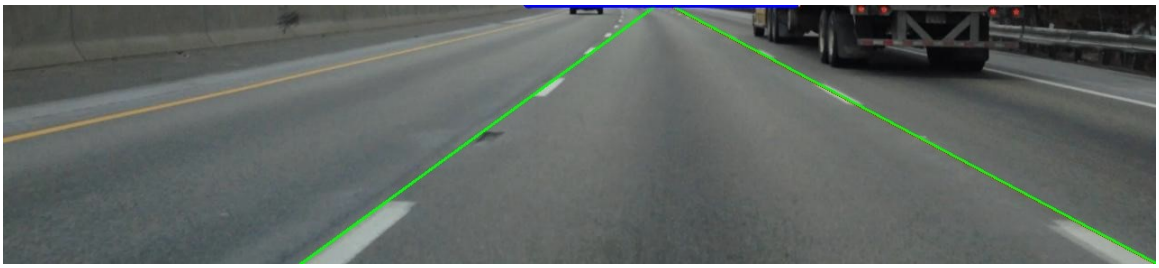


Figure 2.6: Detected lane

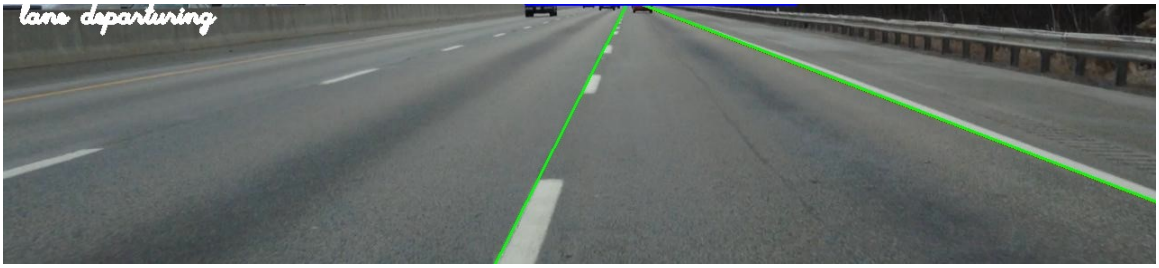


Figure 2.7: Lane departure warning

We have already used Hough transform and it gives out two lanes on the road, now we need to decide if the car is doing lane departure. Yet it will be inaccurate if we just look at one single frame and make decision for that for a single frame, even if we use history information to improve Hough Transform result, is not guaranteed to be correct. To get secure warning signal, we use consecutive 8 frames and a

flag signal to give out lane departure warning signal. If one of the two lanes keeps vertical (or near vertical) for 8 consecutive frames, set the flag to true, meaning that this is a real lane departure movement.

Here the camera is put on the middle of the car when video is taken, so two lanes appear in the middle of the image and are symmetrical to each other when the driver is driving normally. If camera is put elsewhere, nothing will change except for the angle threshold.

Figure 2.8 shows flow chart of the whole Lane departure warning system.

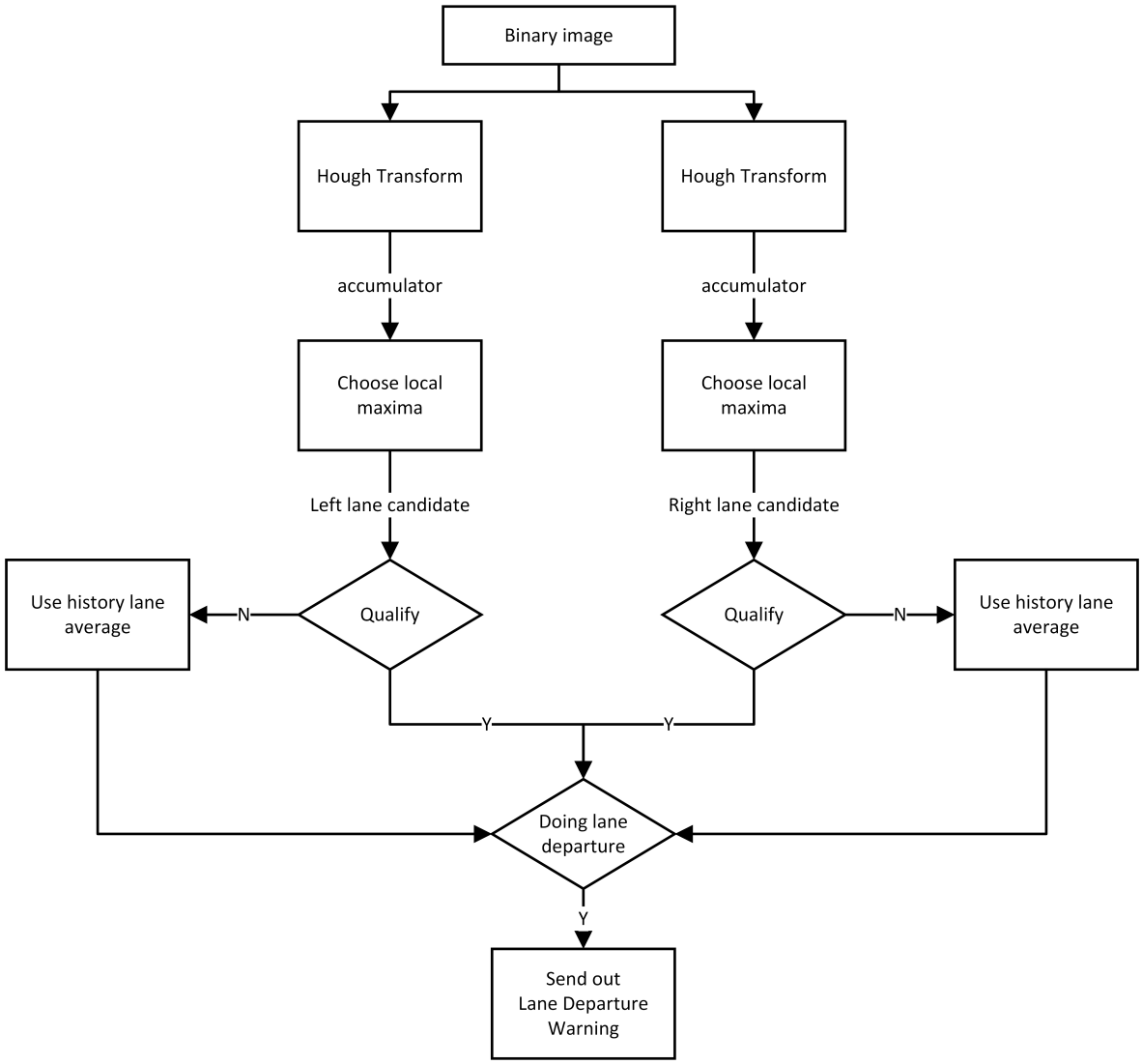


Figure 2.8: Lane departure warning system flow chart

Chapter 3

Car Detection and tracking

3.1 Using symmetry to detect car

Car detection has always been a very popular topic since it's very useful for both driving assistance equipment and some equipment that stand still for monitoring use.

Methods for detecting car differs a lot for different circumstances. For still cameras, the task of detecting objects is much easier than moving cameras. There is a background that does not move so that we could do background subtraction and any objects that are not belonged to background will stand out easily. This method could not be applied to moving camera situation since background is always changing. Here we constraint our discussion to the situation that the camera is mounted on the moving vehicle that is driving on highway. Since we are trying to do front collision warning, the only object that we care about is the front car. After we know where the front car is, we could calculate the distance between the front car and the proceeding car and give out appropriate warning signal when two cars are too close to each other. Thus we constraint our discussion on car detection to

detection of front vehicle while the camera is mounted on a moving vehicle.

3.1.1 Find bounding box

As mentioned in [4], symmetry information plays an important role in car detection. Figure 3.1 shows some images taken from the camera.

Figure 3.1: images taken on road

The reason that symmetry plays an important role in car detection is that the front vehicle that we care about, if any, is always showing its back in the image. The shape of the back of front car is always similar to rectangle, no matter what make it

is. In paper [6] symmetry map is used to find the symmetry axis of front car. The symmetry map is calculated on the area of interest. Each vertical axis is examined in this area of interest. For every vertical axis, all possible widths for that particular axis are examined and results are stored in the symmetry axis. Then examine the symmetry map and find the minimum value that represents the vertical axis and corresponding width with it. Here we extract the idea of finding symmetry axis and combine symmetry property of two lanes.

If there is a proceeding car in the front, it is between the two lanes that the tested car belongs to by definition, which means the position of the front car must be between the two lanes. So the symmetry axis of front car is similar to the symmetry axis of two lanes we detected. Thus the triangle area between two lanes is our region of interest.

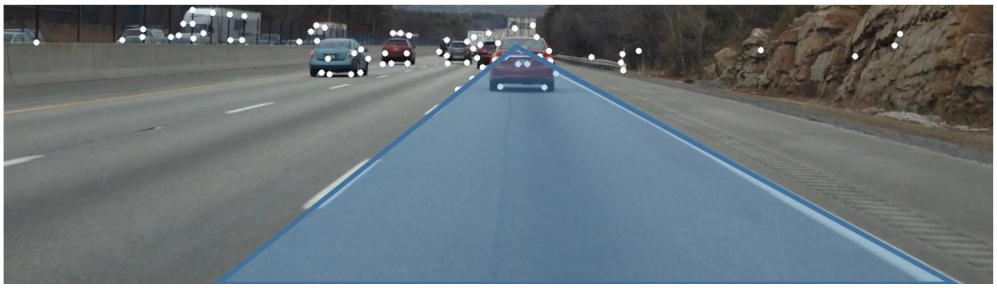


Figure 3.2: Get region of interest ROI and feature points

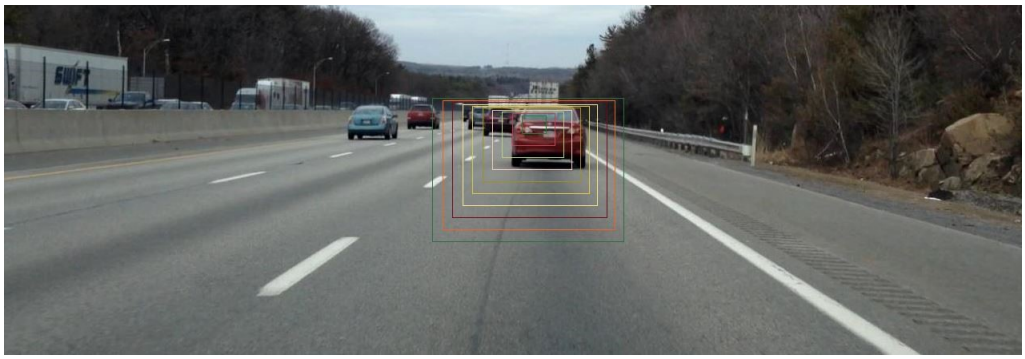


Figure 3.3: draw bounding box

3.1.2 Good Features to Track

Good features are of great importance in feature-based vision system. Here we use good features to track [8], an improved algorithm based on Harris corner, to extract corner features from the image. Our goal is to get two lower corners of the front car so that we could choose the correct bounding box to cover it. After getting the correct bounding box, we would refine the position of proposing bounding box according to the two lower corners' position. The intensity difference by a slight movement (u, v) for image I at position (x, y) can be expressed as

$$S(x, y) = \sum_{u,v} w(u, v) \left[I(x + u, y + v) - I(u, v) \right]^2 \tag{3.1.1}$$

where w(x,y) is the window function, I(x,y) is the value of image at position (x,y), (u,v) is a small movement relative to (x,y). To find corners in the image, we maximize the weighted sum of squared difference should be

$$S(x, y) \approx \sum_{u} \sum_{v} w(u, v)(I_x(u, v)x + I_y(u, v)y)^2 \tag{3.1.2}$$

Apply Taylor Expansion to above equation and we get

$$S(x, y) \approx [x\, y]\, M \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.1.3}$$

where

$$M = \sum_{u,v} w(u, v) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{3.1.4}$$

Matrix M is called Harris matrix and I_x and I_y are first order derivatives in x and y directions. In Harris Corner detection algorithm,

$$R = det(M) - k[trace(M)]^2 \tag{3.1.5}$$

where $det(M) = \lambda_1\lambda_2$ and $trace(M) = \lambda_1 + \lambda_2$, λ_1 and λ_2 are the eigenvalues of matrix M In good features to track, we use

$$R = min(\lambda_1, \lambda_2) \tag{3.1.6}$$

if R is greater than the threshold, the point is considered to be a corner; if it is smaller or equal to threshold, it is not a corner. Good Features to Track has several advantages over Harris Corner. It gives out stronger corners than Harris corner does, and the computational speed is faster. Thus here we utilize Good Features to Track to find corner points of the image (as shown in Figure 3.4).

Figure 3.4: Use Good Features To Track to obtain feature points

3.1.3 use features to find corners of front car

Good Features to Track produces feature points on the image that contains lower corners of the front car. As shown in [pic 18], each bounding box is examined to see if it contains a qualified car image. We define that if there is a car in the bounding

box, it must satisfies that (1) there is at least one feature point that falls in lower left area (denoted in Figure 3.5 as area A), (2) there is at least one feature point that falls in lower right area (as denoted in Figure 3.5 as area B) and (3) the two qualified feature points could not be the same one.

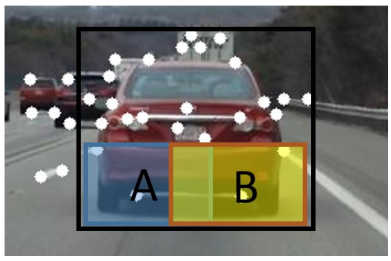


Figure 3.5: Find car corners

On each frame there are more than 25 bounding boxes that are on symmetry axis, and each bounding box needs to examine all the feature points derived from Good Features to Track. The position (x, y) of feature points are stored in a one dimensional array whose order is randomly arranged. It leads to the result that for each bounding box, we need to go through all the feature points of which most are useless. And all feature points have to be examined for every bounding box in the image.

To reduce the computational burden for each frame, we add an additional operation on the one dimensional array that stores all feature point. After Good Features to Track gives out the array of feature point of which each element is described as (x, y), we use Quick Sort to sort the array according to x axis by ascending order and leave y unchanged. As denoted in Figure 3.6, we sort the array by x using Quick Sort and let y just follow how x changes. Feature points are less than 100 points so that sorting time is very short for each frame. Doing one Quick Sort on feature points for each frame will benefit all the bounding boxes that tries to find qualifying points in them. Each bounding box used to search all 100 feature points and for

each frame there will be more than 25 bounding boxes leading to more than 2500 searches for feature points of which most are repetitive. After doing quick sort to the array, we only need to examine those points whose x axis falls in the bounding box, and then check if corresponding y value qualifies. If it qualifies, check further if it is the front car's lower corner point.

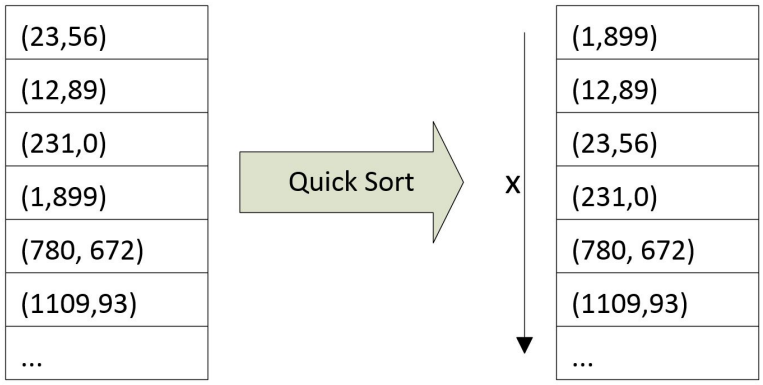
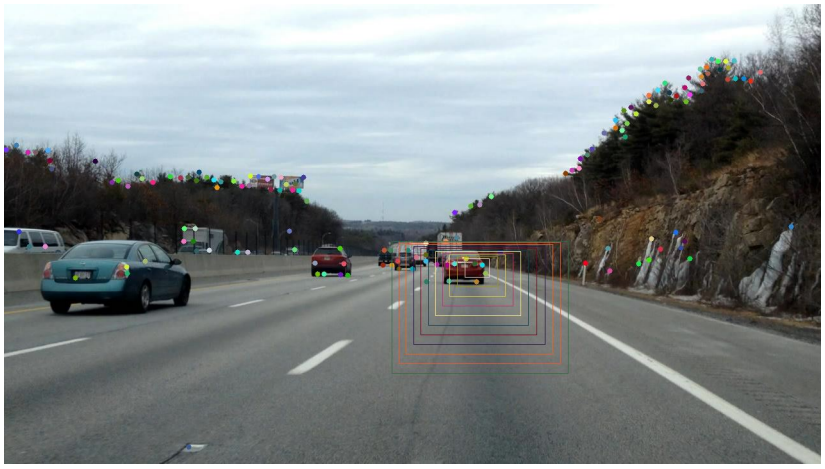


Figure 3.6: Quick Sort to sort array according to x value

3.2 Tracking: Camshift

Detecting car position by finding symmetric bounding box is not enough in order to get an accurate result since false positives are inevitable. Thus we choose to apply a tracking algorithm to reduce the rate of false positive and fill the gaps for failure of detection between frames.

Continuously Adaptive Mean-shift (CamShift) is an algorithm usually used to track a given object, and it is a variation from mean-shift algorithm, which is a widely used data analysis method. Mean-shift is used to find local maxima in a density data set and was found very useful to track moving object in 1998 [9]. Later it was greatly extended and developed. CamShift was a step further than mean-shift for that it could be used to track those objects that change sizes in video stream.



(a) draw bounding boxes

(b) find qualified bounding boxes with corner information

(c) choose the best qualified bounding box

Figure 3.7: Find front car using bounding box and feature points

Given a bounding rectangular box that covers the object, CamShift could keep track of the object even if the size of the object changes.

We have several reasons to choose CamShift instead of other tracking algorithms. In front car detection and tracking problem, line jumping happens frequently. Thus the object in the front usually change without any transition, so that we mainly rely on the results of object detection because tracking results will always stick to one object and very hard to change to another one.

The priority of detection result is higher than the priority of tracking result. Our goal of tracking is helping to fill gaps when detection fails instead of predicting object position for next frame. Some widely used tracking algorithms such as Kalman Filter is used when we need to use history information to predict future results and noise is white noise. Yet here the detection result sent to Kalman Filter is not promised to have white noise and Gaussian distribution. Kalman Filter also help to eliminate noise of past results to make values more "true to real values" which is not needed here. And yet CamShift is suitable here since it use color information to track object and gives out tracking result based on the given object position (using a rectangular box) as initialization information.

Here we would first introduce back projection calculation since it's the prerequisite information needed by both mean-shift and CamShift, then move onto introduction of how mean-shift works and how CamShift make up for the deficiency of mean-shift algorithm.

3.2.1 back projection

Back projection is an efficient method of knowing how well the pixels from an image fit into a histogram model that is derived from another image. We could use back projection to describe the feature we wanted as a histogram and search for similar

features in some test images. The result of back projection is an output image that shows how similar this image is compared to the model image. It is used in both mean-shift and CamShift algorithm. First, histogram model is calculated for targeted feature, which is usually a rectangular patch on the image that is given by user (or other object detection algorithm). Instead of using RGB information, we use HSV (hue-saturation-value) representation. RGB Histogram of object colors changes dramatically in different lighting conditions, yet HSV Histogram (especially hue) of object barely changes. Thus we use hue planes to keep those color information and eliminate lighting influences.

The procedure of turning test image to hue distribution probability image according to histogram model is call back projection.

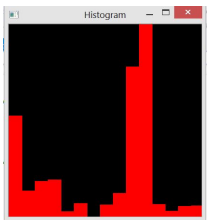
Back projection steps:

(1) Turn the test image from RGB to HSV
(2) Check every 'pixels hue value h(i,j). Search the bin position of h(i,j) in histogram model
(3) Find the value of the bin in histogram model
(4) Save value to result image. If not finished, go back to (2)
(5) Normalize result image to range 0−255 so that the image could be displayed.

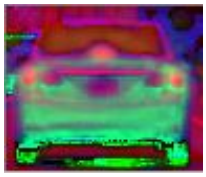
3.2.2 mean-shift

The mean-shift algorithm is widely used technique in finding local maxima of a density function. It has lots of applications and is of great importance in computer vision, such as segmentation, discontinuity preserving smoothing and tracking. What mean-shift does is keep doing the hill-climbing in a density map until it gets to the area where the density is the largest. We use Camshift, which is a deviation of mean-shift algorithm as tracking algorithm.

(a) front car image



(b) histogram of detected front car



(c) HSV image of front car

(d) back projection

Figure 3.8: Steps to get back projection of detected front car image

Mean-shift algorithm is used along with back projection to track objects in computer vision. Mean-shift algorithm takes in a probability image, an initialization window and iteration criteria, and it will output the position after it finds the local maxima. When mean-shift is used in tracking, the input image shall be a back projection image.

The procedure of back projection is as following. Take the histogram model of target object (in our case it is the histogram image of a targeted car) and a complete input image. The output back projection image is of the same size as the input image, and pixel values on output images show the how likely that this pixel is part of target object; the lighter the color is, the higher the probability is. Therefore we could use mean-shift to find the area with highest density in the back projection map, which is the most possible target object position.

In a space whose dimension is d (d is larger than 2) there are lots of points; our goal is move to a place where most points are. We now restrict our discussion to two dimensional space since it is the same situation as how images works. Draw a circle

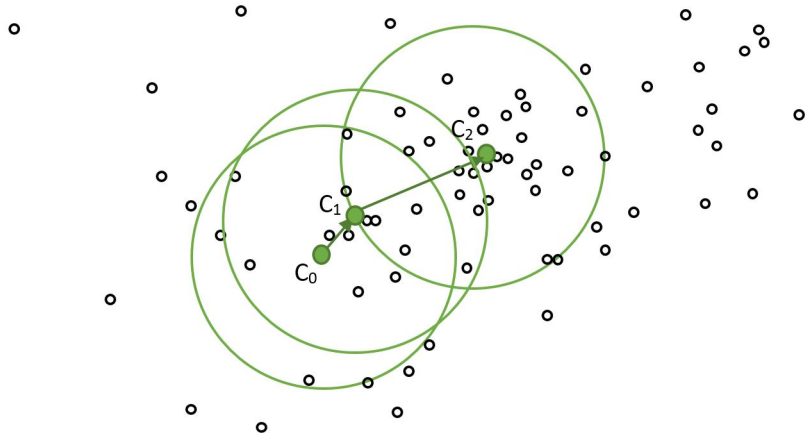


Figure 3.9: meanshift algorithm illustration

in this space and all the points that falls in the circle has a vector. Sum up all the vectors in this circle leads to a new vector that has a direction and value, and this vector is mean-shift. Move towards the vector and we are now one step closer to the highest density area. Keep doing hill-climbing and finally when the circle stops moving, we are at a stable peak of the density map.

Zero-moment mean-shift:

```
for(int i=0;i<I.height;i++)
        for(int j=0;j<I.width;j++)
                M00+=I(i,j);
First moment:
for(int i=0;i<I.height;i++)
        for(int j=0;j<I.width;j++)
        {
                M10+=i*I(i,j);
                M01+=j*I(i,j);
        }
xc=M10/M00;
yc=M01/M00;
```

3.2.3 Camshift

Meanshift is a robust algorithm to find maximum value of a probability distribution, but it requires that user predefine the object size (window size) ahead of process, which means the window size is always the same no matter if the object has come closer and become bigger in the image, which is not very good.

Camshift (Continuously Adaptive Mean Shift) was first introduced by Gary Bradski [9], it fixes this problem by bringing adaptive window size and rotation of the target. First, apply Meanshift to image. When it converges, center window on (x_c, y_c) and update size of the window:

$$width = 2\sqrt{\frac{M_{00}}{256}} \tag{3.2.1}$$

Height is set to $0.7 * width$ in car detection case. Then continue using Meanshift to converge and update window size until it finally converges.

Chapter 4

Distance Estimation

4.1 Introduction

Distance calculation with single camera is sometimes not accurate due to different reason. Minor shaking of camera, camera position not properly aligned with the ground, road situation all influence the result of distance calculation. In order to increase distance calculation accuracy, we proceeds with the approach using two cameras, which is usually called Stereo Vision method. Human eyes are the best example of stereo vision. Stereo vision algorithm tries to work as eyes do: connecting two images acquired and combine them together to get more information, the most important one of which is depth information. For stereo cameras, we try to find correspondent points in left and right images, and with the knowledge of baseline, which is the distance between two cameras, we could get 3D position of each points. Section 4.2 introduce the prerequisite of all vision related algorithm: camera calibration. Camera calibration is critical to help remove tangential and radial distortion, and stereo vision algorithms can only work properly with undistorted images. We will introduce how to do stereo calibration and getting disparity map in Section 4.3.

4.2 Camera Calibration

We begin to look at how simplest camera is working. Suppose f is focal length, Z is the real distance between object and camera, X is the actual size of an object, then the relationship of the size of the object on the image and X is:

$$-x = f\frac{X}{Z} \tag{4.2.1}$$

Take displacement of chip when manufacturing into account, then the center of the image is not on optical axis. As shown in Figure 4.1, (c_x, c_y) is not the center of image plane in most cases. Thus we bring two more parameters c_x and c_y:

$$\begin{cases} x = c_x + f_x\frac{X}{Z} \\ y = c_y + f_y\frac{Y}{Z} \end{cases} \tag{4.2.2}$$

In real world, in order to get the relationship between object point (X, Y, Z) and object position in the image (x, y), we expand (x, y) to (x, y, w) where w is proportional value and (x, y, z) is called homogeneous coordinates. [10]

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.2.3}$$

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2.4}$$

M was called intrinsics matrix, because it contains the information about the inner parameters that camera sensor has.

Extrinsics matrix, which represents the position and pose of a particular object, includes two parts- rotation matrix and translation vector. When an object is taken a picture, it is the procedure of moving object from real world coordinate to camera coordinate, thus we use rotation matrix and translation vector to represent this procedure. First rotate object separately in x, y, z direction by ψ, φ, θ degree:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix} \tag{4.2.5}$$

$$R_y = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} \tag{4.2.6}$$

$$R_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2.7}$$

$$R = R_x R_y R_z \tag{4.2.8}$$

$$P_c = (P_w - T)R \tag{4.2.9}$$

In order to get intrinsics matrix and distortion coefficients, we use chessboard to do camera calibration. After calibration, undistorted images could be acquired, which is of great importance for following stereo calibration. The relationship between the point on chessboard and on image plane is connected by perspective transform:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = M \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2.10}$$

where M is intrinsics matrix and $\begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix}$ is the 3 by 4 matrix that is the combination of rotation and translation matrix. $[x, y]^T$is the point position in image plane coordinate, $[X, Y, Z]$ is the point position in real world. Since chessboard is flat so the point position in chessboard coordinate $[X', Y', Z']$ is actually $[X', Y', 0]$, and transform is

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = M \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} \tag{4.2.11}$$

We use two C310 cameras from Logitech for recording and testing images. After camera calibration, we get intrinsics and extrinsics for both cameras:

```
M1:
rows: 3     cols: 3
data: [ 1.4157914383711604e+003, 0., 6.6802976893357720e+002, 0.,
          1.4162766911131785e+003, 3.4914784448315351e+002, 0., 0., 1.
    ]
D1:
rows: 8     cols: 1
data: [ -1.4776484941109638e-001, 7.6499050522731760e-001, 0., 0.,
          -9.6916887332979833e+000, 0., 0., -8.2795353403607788e+000 ]
M2:
rows: 3     cols: 3
data: [ 1.4157914383711604e+003, 0., 6.5260788018733592e+002, 0.,
```

```
            1.4162766911131785e+003, 3.5555599289921815e+002, 0., 0., 1.
    ]
D2:
rows: 8    cols: 1
data: [ 6.6749687756802317e-002, -2.8720351084190282e-001, 0., 0.,
            -1.3154781737158865e+000, 0., 0., -2.0339647716111684e+000 ]
```

4.3 Stereo Image

Suppose we have two perfect identical cameras that do not have lens distortions and are aligned perfectly side by side, the real distance between two cameras is T, the real distance from object to camera plane is Z

$$\frac{T-d}{Z-f} = \frac{T}{Z} \tag{4.3.1}$$

$$Z = \frac{fT}{d} \tag{4.3.2}$$

where disparity d is:

$$d = x_l - x_r \tag{4.3.3}$$

An important note is that d is inversely proportional to Z, which means when Z is extremely large, d is too small to be noticed and the change of d cannot be asserted either. On the contrary, when d is extremely small, the same of amount of $\triangle d$ would not change real distance by much. This tells us that the distance acquired from disparity map is not accurate when distance is too big or too small, and luckily neither do we care about these two situation: when front car is too far away, we do not care about it; if the front car is too close, we do not care because the front car

must be detected before distance gets too close.

To make computers emulate what human eyes are doing, we need to go through following processes.

First, align two camera side by side and make them as parallel as possible. The distance T between two cameras shall be set properly corresponding to the distance of target objects. If target objects are rather close to cameras (e.g. for indoor detection) T could be set within 200mm˜300mm; if target distance is far away, then T shall be set to larger value. Calibrate each camera and get intrinsics parameters and undistorted images to eliminate most of lens distortions. One of the reasons of aligning cameras as frontal parallel ([10])is because when doing stereo correspondence, only the points that could be seen from both images can be used to find its correspondence from one image to the other one.

For perfectly aligned cameras, there is no rotation between two cameras, and translation is just the distance between them, yet in real world, due to manufacture limitations and camera placement limitation, two cameras can never be perfectly aligned, which will leads to the need for calibration. The purpose of stereo calibration is to calculate relative rotation matrix R and relative translation T. This is the position of one camera relative to the other one, here we set the rule that R and T are right camera relative to left one. We are still using chessboard image to do stereo calibration, so during stereo calibration process, it will take 20 pair of image pairs (the number of image pairs is decided by users, but the more the better.) Insufficient calibration images will lead to inaccurate calculation of rotation matrix and translation matrix. Also, intrinsics of left and right cameras are fed to stereo calibration. Also stereo calibration itself could be used to do camera calibration and stereo calibration at the same time, it is better to get good verified intrinsics data and feed them as input when doing stereo calibration.

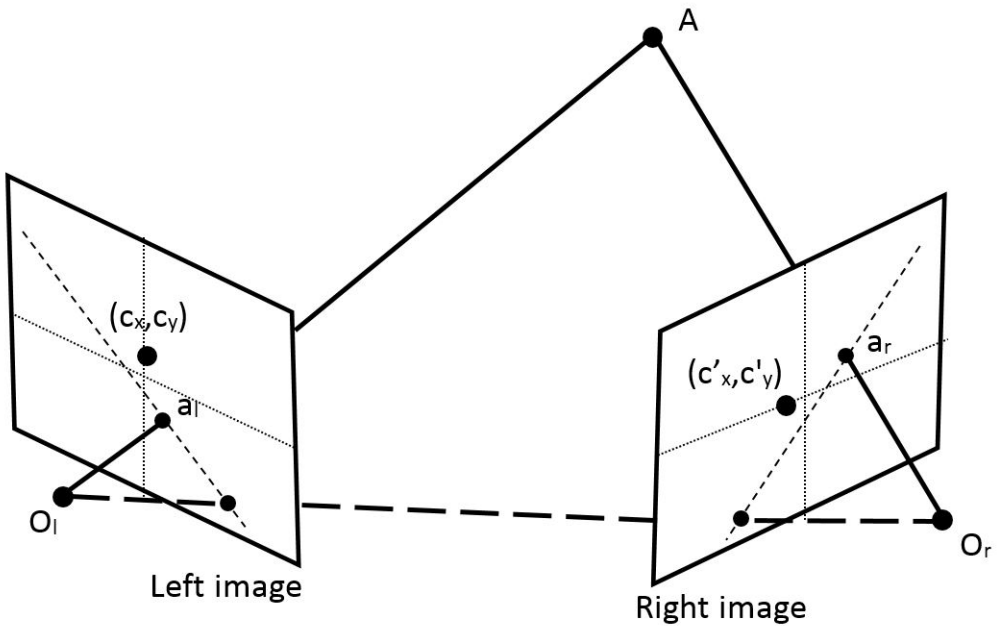


Figure 4.1: camera position and epipolar lines when not calibrated

In Figure 4.1, A is object point in real world and $a_l$and $a_r$are projection to left and right image planes. $O_l$and $O_r$are center of projection for left and right camera, and point O_l, O_r, A define epipolar plane. The two intersection lines between epipolar plane and left and right images, are call epipolar lines.

In Stereo rectification is processed after calibration. Stereo calibration is getting relative position between two cameras, and rectification is turning one of the camera and make it be in the same plane as the other one, which is the ideal situation we discussed at the beginning of the section. Rectification process uses R and T from stereo calibration and use different algorithm to rectify images so left and right images look like they are taken by two parallel cameras sitting side by side. Reprojection matrix Q turn 2D point on image plane to 3D point in world coordinate, and point (x, y) in image is mapped to point $(X/W, Y/W, Z/W)$ in real world.

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \tag{4.3.4}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c'_x)/T_x \end{bmatrix} \tag{4.3.5}$$

where d is disparity calculated from stereo correspondence, (c_x,c_y) are principal point in left image, c'_x is principle point's x coordinate value in right image. T_x is translation between right and left cameras in x direction.

After rectification, left and right images are row aligned, which means right camera is virtually rotated and moved so that its epipolar line is the same as the one in left camera. This leads to the fact that for each point in left image, if it's also shown in the right one, then the correspondent point must be in the same row. Rectification greatly reduce the computational burden for correspondence and now we only need to search one row for each point. The purpose of correspondence is to find disparity $d = x_l - x_r$ for the object points in two images, so that reprojection matrix Q is acquired from rectification process and disparity d together will enable user to calculate real position for any point in image plane, as long as the point is seen from both left and right images after rectification.

In Figure 4.2 the object A's physical position in left and right image is (x_0, y_0) and (x_1, y_1). Since images have already been rectified, left and right camera's epipolar line is the same, so that $y_0 = y_1$. Also, notice that $x_0 < x_1$because A appears on the right in left image and appears on the left in right image. To search for

the corresponding point for (x_1, y_1), start from the pixel at (x_1, y_1) in right image and move left while keep $y_1$unchanged until a matching is found for pixel at $(x_{0,}y_0)$ in left image. the distance that the search travels in -x direction is disparity d. After correspondence, we have d for every point in the image. We could refer to reprojection matrix Q if we want to calculate the real position, including distance information, for any point in the image.

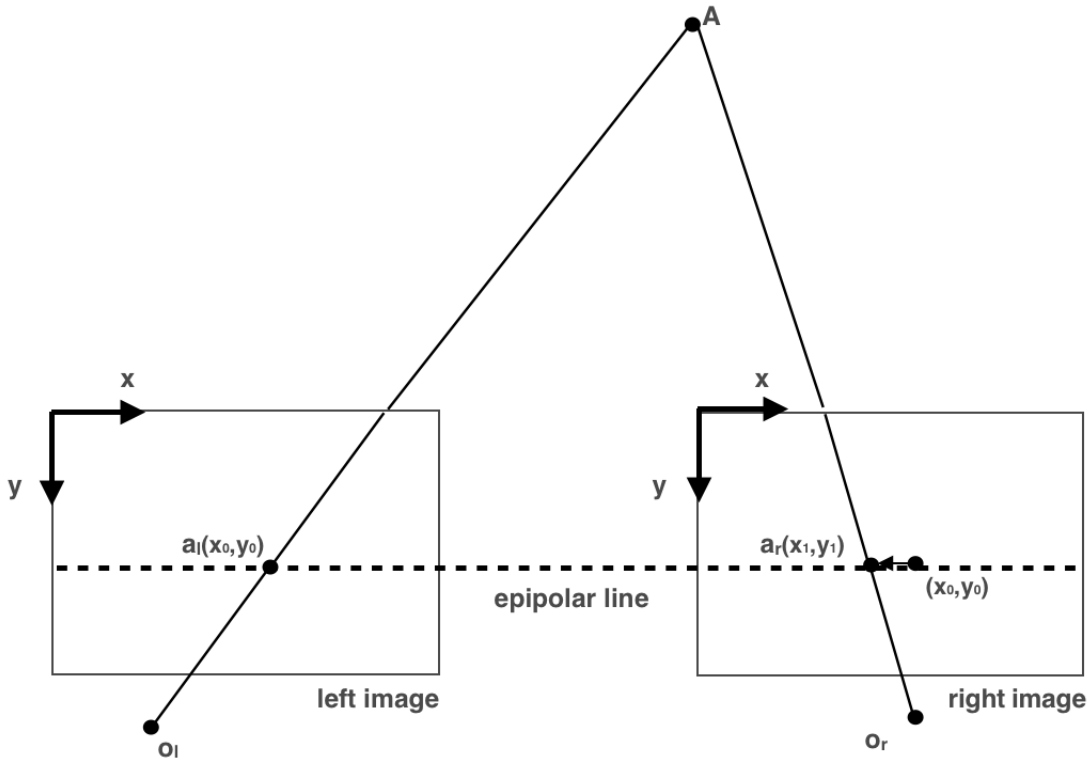


Figure 4.2: stereo rectification result

After stereo calibration and rectification, we get relative translation matrix R and relative translation T.

```
R:
rows: 3    cols: 3    dt: d
data: [ 9.9790309865240179e-001, -6.3715996994504490e-002,
    -1.1387599700156383e-002, 6.3826544576960839e-002,
    9.9791461114552471e-001, 9.6229449567846129e-003,
```

```
    1.0750716594717668e-002, -1.0329597730423501e-002,
    9.9988885457506105e-001 ]
T:
rows: 3    cols: 1    dt: d
data: [ 1.8625930899805514e+001, 5.8167566764790146e-001,
    6.3793034328336717e-001 ]
```

We could also verify the matrix by checking if the baseline between two cameras matches $T.x$:

$$T.x \times chessboardBlockSize = baseline \tag{4.3.6}$$

Chapter 5

Conclusions

In this thesis, we describe a way to implement lane departure warning and front collision warning system. We first use Ostu Threshold method to get binary image, then apply the image with Hough Transform, which is a well-known algorithm for line detection. Later we use the boundary information from current frame and combine it with history information to decide if the car is doing lane departure. Then we utilize symmetry character of front car appearance to do car detection, and combine it with Camshift tracking algorithm. Also, we move on to use stereo vision method to do stereo calibration and later calculate depth map, which includes distance information for detected car.

However, the method may be further improved in two ways. First, stereo image processing is very time consuming and cannot be processed as fast as single image, especially when we want to calculate disparity map. So we may need to find a faster way to get essential disparity map for detected car but not the whole map for the image.

We also notice that detecting and tracking front car process is currently combined in a rather intuitive way. Front car detection is different from other object detection

because on one hand front car's position barely move while driving, on the other hand it will change dramatically once the front car or user's car left current lane. So we may want to find a better algorithm to handle front car detection part to make the algorithm more robust.

In conclusion, lane departure warning and front collision warning system is a promising system that might be essential for road driving in the future.

Bibliography

[1] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 20–37, 2006.

[2] W. Wang and X. Huang, "An fpga co-processor for adaptive lane departure warning system," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 1380–1383.

[3] J. Zhao, B. Xie, and X. Huang, "Real-time lane departure and front collision warning system on an fpga," *ipi*, vol. 1, p. 0.

[4] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 694–711, 2006.

[5] S. Sivaraman and M. M. Trivedi, "A review of recent developments in vision-based vehicle detection." in *Intelligent Vehicles Symposium*, 2013, pp. 310–315.

[6] G. Toulminet, M. Bertozzi, S. Mousset, A. Bensrhair, and A. Broggi, "Vehicle detection by means of stereo vision-based obstacles features extraction and monocular pattern analysis," *Image Processing, IEEE Transactions on*, vol. 15, no. 8, pp. 2364–2375, 2006.

[7] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[8] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[9] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," 1998.

[10] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[11] R. Marzotto, P. Zoratti, D. Bagni, A. Colombari, and V. Murino, "A real-time versatile roadway path extraction and tracking on an fpga platform," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1164–1179, 2010.

[12] H.-Y. Lin, L.-Q. Chen, Y.-H. Lin, and M.-S. Yu, "Lane departure and front collision warning using a single camera," in *Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on*. IEEE, 2012, pp. 64–69.

[13] R. Risack, N. Mohler, and W. Enkelmann, "A video-based lane keeping assistant," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 356–361.

[14] S. Lee, H. Son, and K. Min, "Implementation of lane detection system using optimized hough transform circuit," in *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*. IEEE, 2010, pp. 406–409.