CSC420 Final Report

Wen Xiao

November 2016

1 Introduction

Autonomous driving is one of the major research venues these years, both in industry and in academic. Recently, we can find some of the techniques have already been put into practice.
In this project, we are investigating some important fields in the topic of autonomous driving, in which image processing plays an important role. Obviously, the primary task for autonomous driving is to detect the road and cars, and then it will be better if we can know the viewpoint of cars and the tracking of cars to decide if the car should stop, change lines or accelerate. We believe that safe operation of an autonomous car can only be guaranteed if the car can interpret its environment reliably from its own sensory input just like a human driver can navigate unknown environments.
We mainly have three parts in the project, the first part is about road detection, the second part explores the car detection and viewpoints, the third part, which is the one discussed in this report is about the car tracking.
The traditional way to detect tracklets is to build a probabilistic model. In 2014, as Andreas Geiger et al. mentioned in the vehicle tracklet part in [3], they proposed a Hidden Markov model based on lanes and parking areas, and the accuracy was around 80%.
Here I mainly proposed four methods, including the Kalman Filter[1][2][3][4], the histogram of color, the overlap ratio, as well as the SURF feature matching.

2 Methods

The main idea of car tracking is that we want to match the detections of car in the adjacent frames, so that we can know the number of cars in a video, and keep track of those cars separately. So first of all, I built a list of tracklets, and each tracklet contains following fields:

- frameId: a list of frame id, the tracklet appears in those frames.
- trackletId: the id of current tracklet
- bbox: a list of bounding box of the car in the frames with corresponding frame id.

- totalActive: the number of frames in which the object is detected. If it is less than 3 for a tracklet, then the tracklet is discarded.
- consecutiveInActive: the number of consecutive frames that the object is not detected. If it is more than 3, the tracklet is regarded as nonactive anymore.

In each frame, we will have a matching 'cost' for every detections in this frame with every active tracklet, and then do the matching based on the cost by Munkres Algorithm. So the most important thing here is to estimate the cost function. I tried four kinds of cost functions, they are kalman filter, histogram of color, overlap ratio of bounding box and surf matching metric.

2.1 Kalman filter

This is the most common tool to do the tracking, it uses a series of location coordinators overtime, combined with Gaussian noises, to produce the estimation of next location coordinator. It mainly uses the Bayesian inference and estimate a joint probability distribution over the variables for each timeframe.
There are three main steps in this method. First, we need to initialize a kalman filter model for each tracklet. Then we will predict the next possible bounding box and for each active tracklet, compute the distance between the prediction and every detections in the current frame. The last step is that we need to find the best matching based on the distance we got and update the kalman filter model by the new matching bounding box.
I used the built-in matlab function 'configureKalmanFilter' to implement this method, with the 'ConstantVelocity' option. It is a simple approach for figuring a kalman filter for tracking an object, we assume it moves with the constant velocity here. The estimation equation for x_k is

$$x_k = A * x_{k-1} + w_{k-1}$$

where A is the state transition model and w is the process noise of state k. And the measurement equation is

$$z_k = H * x_k + v_k$$

where z_k is the measurement of the state, which is used for distance computing, and H is the measurement model, v_k is the measurement noise of state k.
The related code for this method is ComputeTracklets_kf.m.

2.2 Histogram of color

In this method, I computed the histogram of red, green and blue for each bounding box separately. Each histogram is a vector with length 266, representing the number of pixels with the certain number(0-255) in the certain channel(r,g,b). After that I just concatenate the three vector into one vector and divide it by

the area of the bounding box to get the persentage histogram. Only the vector of the latest bounding box will be stored in the 'colhist' field of a tracklet. Everytime we will compute the euclidean distance between the colhist of one tracklet and the vector of the new detection, and regard it as the cost of this assignment.
The related code for this method is ComputeTracklets_colorhist.m.

2.3 Overlap ratio of bounding box

In this method, the cost of assignment is the overlap ratio of bounding box in the current frame to the last bounding box of active tracklets.
The related code for this method is ComputeTracklets_overlap.m.

2.4 SURF matching

In this method, I extract the SURF features from all the bounding boxes and try to match the features between the new detection and the latest detection in all active tracklets.
The related code for this method is ComputeTracklets_surf.m.

3 Main challenges

The main challenge for the car tracking is to find a measurement between a tracklet and a car so that we can decide whether the car belongs to a current tracklet or it is a new car. So I tried to use both position information and the feature information to find the appropriate measurement.

4 Results and discussions

4.1 Important Parameters

a. Assignment Threshold
The result of tracking for all of the four methods mainly depend on the assignment threshold. It is the maximum cost that we will assign a detection to a tracklet, if the cost exceed this value, we will not match the detection and tracklet. As the threshold increase, it is more likely that a new car is assigned to an existing tracklet. Meanwhile, the loss rate of the current tracklet is also decreasing, which means the tolerance of error becomes higher.
If the threshold is very high, as shown in Figure 1, it assigns the new white car to the tracklet of the car in front of it. But if the threshold is very low, then it may find many tracklets for one car with very short length, as shown in Figure 2. Here I just take the Kalman Filter method as example, the samething happens for all the other three methods.
The reason for it is trivial, since the assignment is based on a kind of 'distance' between a detection and tracklet, when a car moves quickly, the distance

between detection and tracklet becomes long, when it is long enough, we will assign the car to a new tracklet. But when two cars are close and similar, the distance between car A and the tracklet of car B is short, if it is short enough, we will assign carA to the tracklet of car B.

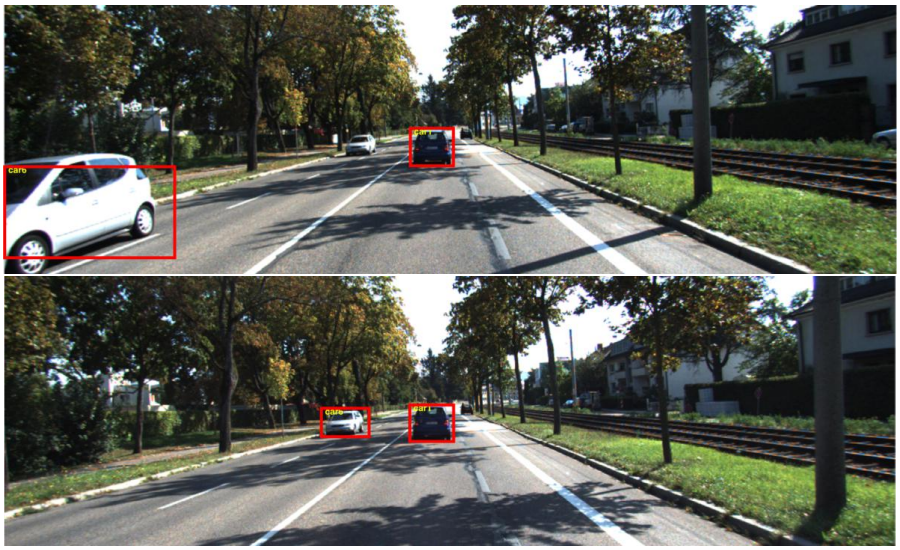

Figure 1: When the threshold is high, it will assign the new detection to an existing tracklet. The method is Kalman Filter and the threshold is 100.

b. Time lapse between frame

This is also an important factor for the tracking, and it is the interval between two adjacent frames. If the time lapse is large, then the position of the same car changes a lot between two adjacent frames. The time is totally depend on the setting of the video recording equipment, and obviously, it is better for tracking to shorten the time interval. Also, the value of assignment threshold depends on the length of time lapse, as the time lapse increase, we need a larger assignment threshold.

4.2 Comparison between four methods

The overall best method among all these methods is the Kalman Filter one, once we have a reasonable time lapse with corresponding assignment threshold, it will have a good performance.

The performance of overlap ratio is also perfect once we set a reasonable assignment threshold. This method is a simplification of the Kalman Filter one, since both of them are based on the position information of the bounding boxes. Instead of building a model and make a prediction, in this method, we just regard the last bounding box of the tracklet as the 'prediction', and then compute the overlap ratio.

Figure 2: When the threshold is low, it will assign the existing car to a new tracklet. The method is Kalman Filter and the threshold is 5.(Here, I just set the constraint of totalActive to be 1, otherwise, there is no tracklet at all, since every tracklet has length 1)

Different from the methods based on the position information, the histogram of color method and SURF method are based on the features of cars.
For the histogram of color method, it performs bad when there are two cars with the similar color, as shown in Figure 3. Although I normalized the histogram towards the area, the size of bounding box still matters, if the bounding box is much larger than the car, then it will include some pixels of road, which makes the histogram changes a lot. Besides, the shadow also makes difference, since the color of car in the shadow is very different from the one not.

The worst method I tried is the SURF one, before experiments, I used to think SURF is good for matching, that is the reason that I proposed this method. But in practice, the bounding box for each car is too small to extract features from. After decreasing the metric threshold of SURF detection algorithm, I'm able to get more features for every bounding box, but the problem is that almost all of the cars are similar, so their feature can be matched even if they are not the same car, as shown in Figure 4.

Figure 3: When there are two cars with similar color, the Histogram of Color method has a bad performance.

5 Conclusion and future work

By the result of experiments on these four methods, it seems better to use the cost related to the position, instead of the own features of cars, maybe due to the low quality of the image. Since our data is a simple road with few cars, which makes tracking easy, the next step for us is to apply this method to a complex traffic scene. It may be also a good trial to track based on the 3D model of cars, in addition, I can try to combine the position information and the feature information of the cars, which may help increase the accuracy.

References

[1] URL: `http://www.cvlibs.net/software/trackbydet/`.

[2] Andreas Geiger. "Probabilistic Models for 3D Urban Scene Understanding from Movable Platforms". PhD thesis. KIT, 2013.

[3] Andreas Geiger et al. "3D Traffic Scene Understanding from Movable Platforms". In: *Pattern Analysis and Machine Intelligence (PAMI)* (2014).

[4] Hongyi Zhang, Andreas Geiger, and Raquel Urtasun. "Understanding High-Level Semantics by Modeling Traffic Patterns". In: *International Conference on Computer Vision (ICCV)*. 2013.

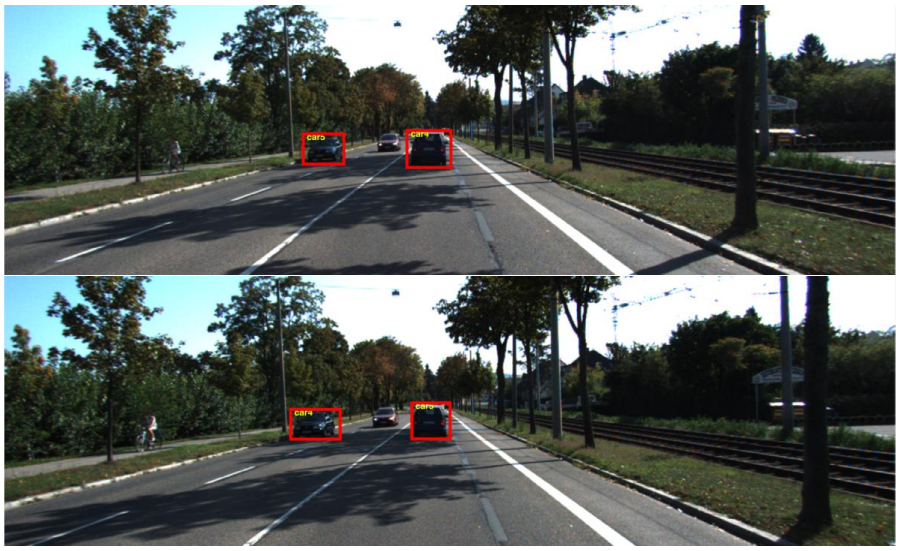

Figure 4: Wrong assignment using the SURF method