research project about trajectory predication technology for autonomous driving vehicle

Xi Hu

January 2020

1 Introduction

Since 2010, with the development of deep learning technology and the improvement of computing power of Central Processing Units (CPU) and Graphics Processing Units (GPU), the use of algorithms for real-time image processing has also become realistic. Therefore, the self-driving car technology based on those technologies is also gaining a high degree of development. At the same time, self-driving is also a major trend in the future: More than 1.2 million people are killed every year because of traffic accidents. These accidents often occur because of the driver's negligence. If cars could be driven by computers instead of people, these accidents would happen less. The amount of accidents will be greatly reduced, which is the necessity of self-driving car technology.

From their life's experience, human drivers are adept at both tracking other participants in traffic and intuitively estimating where they will go in the future. For autonomous driving systems, the prediction of the other objects' behavior in the next few seconds based on their current state, including position, velocity, and yaw angle, has a critical influence on ego-vehicle's decision on the behavior at the next moment. A typical case is when the ego-vehicle desires to change lanes and overtake, it needs to observe the velocity of the vehicle in front and judge whether it is possible to change lanes in the next moment. In detail, it requires us first to detect and identify the surrounding objects such as car, truck, or pedestrian. Additionally, based on the context of the observation on continuous data, like multiple frames of video or point cloud, it is possible to infer the states of these objects. At last, based on these states, the trajectory in the next seconds could be predicted. The primary purpose of this research project is to investigate and compare different state-of-the-art algorithms for trajectory estimation in videos in the driving context with respect to these criteria. For this, the following tasks are required to complete:

1. Research recent published papers in the area of object tracking and trajectory estimation.

2. Implement at least one of the according algorithms.

3. Evaluate its performance regarding trajectory estimation precision and reliability on a suitable data basis.

4. Consider and implement improvements to the limitations identified.

2 Prerequisites and Related Work

2.1 Prerequisites

This research project mainly consists of two parts: detection and prediction. The technologies used in these two parts are different, therefore the corresponding prerequisit-knowledge is required.

1. Neural network: The detection part utilizes a machine learning algorithm called neural network(NN), which is multi-layers of "neuron", which can be represented as linear transformation and non-linear activation. In this way, the neural network model can learn from the raw data, such as image data or point cloud data, to finish classification or regression tasks. A variant of neural networks commonly used for detection tasks is the convolutional neural network(CNN) which extracts features from the original image to improve classification effect.

2. RCNN: "Regions with CNN features" proposed by the paper[1], which is the first model applying CNN for object detection. In simple terms, it could be divided into 4 steps: *Region proposals:* Using a region-proposal method based on image color and texture consistency as metrics to obtain several proposed regions. *Feature extraction:* Applying CNN to perform feature extraction on the proposed region obtained in the previous step. *Classification:* Utilizing a classifier to classify different regions based on the features. *Bounding box fine-tuning:* Utilizing a regressor to fine-tune the coordinates of the Bounding box. The fast RCNN and faster RCNN are based on this framework with more optimizations and details. The idea of pointRCNN is similar, which is extended to process the point cloud data.

3. Kalman filter: The core framework of prediction is based on the Kalman filter, which is frequently used for state estimation. In simple words, it divides the determination of the state of an object into two parts, one is the prediction based on history measurements, and the other is the current measurement value, and the weighted average of the two determines the best estimate of the current state.

2.2 Detection and Tracking

For detection and tracking of objects, there are two main approaches, based on different sensors: The first method utilizes mono-camera, whereas another one works on Lidar data. Briefly, the mono-camera based method[2] uses a camera to obtain image data then uses a pre-trained faster-RCNN[3] to detect the object. As a result, the bounding box coordinates would be returned. After obtaining the bounding box of the object of each picture, the next step is to use CNN[4] to infer 3D information of the objects according to bounding boxes. Finally, the context information of multiple image frames will be used for position estimation and tracking. In this process, the detector can be replaced with other models, such as the YOLO network[5]. Compared to the first method, the Lidar-based method[6] is more intuitive. The point cloud data collected through Lidar contains the position information of the surface of objects, within a limited range. Using these point clouds, a neural network, so-called point-RCNN, can be trained and applied to infer. The output of inference is 3d bounding box coordinates. The bounding box obtained from different point cloud frames will be used for location estimation and tracking. The most commonly used method is the Kalman filter [7]. In literature [2], the LSTM neural network was used for tracking. Experiments demonstrate that the result is better than the Kalman filter. The camera-based approache and Lidar-based approache have their own advantages and disadvantages: because of the large number of image detection datasets and available models, the mono-camera-based method has several options. Another benefit is the low price of the camera as a sensor, which is considered as the most encouraging method for commercialization. However, At the same time, the limitation is apparent as well: from the image, it is difficult to measure the spatial information of the object accurately, such as position and yaw angle. Even if it is possible to infer the information through a neural network, the error generated from inference will cause significant effects on the following trajectory predication. Moreover, according to the literature [8], the estimation of location information can be improved by the stereo camera, which could be an extra option in the future. On the other hand, although the Lidar-based method can provide more stable and accurate position information, while because of the lack of a large number of labeled data sets, the literature related to this method is still scarce. Another disadvantage of this method is that Lidar is still high-costly as a sensor, especially for those commercial-grade high-resolution Lidars.

2.3 Prediction

Once the tracking results are obtained, the next step of the workflow could be processed: predication. There are two most commonly used methods for predication: data-driven approach and model-based approach. The literature [9] gives a typical example of the data-driven method: trajectory clustering. As shown in the name, this method is based on trajectory data collected from the same scenario. During the training phase, different trajectories are clustered and stored. Each cluster represents one behavior, such as going straight and turning around. Next, whenever a part of the initial trajectory of an object is obtained, it can be classified to a particular cluster. As a result, the remaining trajectory can be predicted. Compared to the former method, the kinematic-model-based method is more intuitive, which aims to infer velocity, yaw angle, and other motion information in the short future, according to the states from the current frame and history frames.

The advantage of the data-driven method is that completely using realistic data can help to perform a more robust prediction to some irrational behavior, while the disadvantage is that a large amount of data needs to be collected from different scenarios. As a comparison, the kinematic-model-based method does not require data collection. However, the disadvantage is that it can only predict "rational" behavior in typical situations.

In the trajectory prediction part of this research project, we choose the model-based method because it does not require numerous data. Among many process-models of different complexity, the non-linear point model was determined. Compared to other process-models, it keeps a balance between complexity and descriptiveness of the behavior, which means there are no excessive parameters to consider, while it can also be used to describe many common motion behaviors. The other models, such as the linear point model, assume that the object only moves at a constant velocity, which is not realistic in usual cases. In contrast to the linear point model, the kinematic bicycle model with the PID controller[10] can better describe the motion of the object vehicle, but at the same time, it is unavoidable to set the parameter of the ego-vehicle, such as the distance from the mass point to the front and rear tires. The parameters of the PID controller also need to be considered. Another more complicated option is the dynamic bicycle model with the PID controller. This model includes more parameters, such as the friction between the tire and the ground.

3 Method and Implementation

In this research project, the methods based on mono-camera and lidar were selected as object detector and object tracker because it is more accessible to get the corresponding labeled dataset and model. Concretely, according to the literature [2], we choose faster-RCNN as the camera detector and decide on the LSTM network as the tracker considering performance and convenience. Additionally, the pointRCNN[6] will work as Lidar detection model. As for the trajectory predictor, we choose the model-based method. According to the literature [11], the non-linear point model is the most balanced choice in terms of model complexity and behavior description capabilities. So the workflow of the entire project is shown as Figure 1.

3.1 Method

Because the implementation of detection and tracking model is given and can be used directly, therefore only prediction part will be introduced here. As discussed earlier, the non-linear point model will be utilized as the process model in this project. After selecting the process model, multiple possible motion behaviors need to preset. In the literature [11], nine common motion behaviors are presented as the model set, which consists of the constant velocity (CV) model, and eight maneuver models. Four of the maneuver models model a constant tangential acceleration (CTA), and each has a preset acceleration:

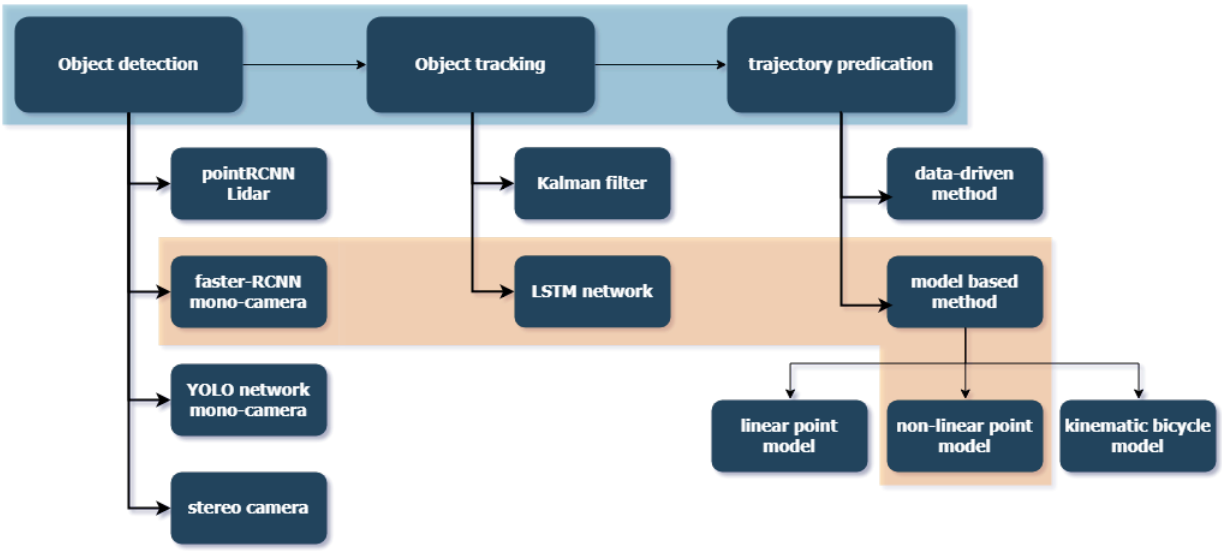

Figure 1: The project workflow

$67m/s^2, -67m/s^2, 33m/s^2, -33m/s^2$, which model constant accelerations at approximately ± 7 and ± 3.5 times gravitational acceleration. Another set of accelerations modeled are those in the direction normal to the velocity, which model constant turns (CT): 14 $deg/s, -14$ $deg/s, 7$ $deg/s, -7$ deg/s.

In order to represent these motion behaviors with a non-linear point model, and assuming the Markov hypothesis is met, i.e., the state at time k is only related to the state at time $k-1$. According to literature[11], The update process from time $k-1$ to k can be described as the following formula:

$$x_k = Fx_{k-1} + Gu_{k-1} + Gw_k, \quad w_k \sim \mathcal{N}(0, Q) \tag{1}$$

where state vector $x = (x, \dot{x}, y, \dot{y})^T$, which represent x and y coordinates, and corresponding velocity $\dot{x}$ and $\dot{y}$, respectively. F is the state transition matrix, which will vary w.r.t different behaviors; G is the control matrix; u_{k-1} is the control parameter at time $k-1$, which is related to acceleration and deceleration. w_k is a Gaussian noise with a mean of 0 and a variance of Q.

And F, G and u_{k-1} for nine different model are shown as below:

CV:

$$F_{CV} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}; G = \begin{bmatrix} T^2/2 & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T^2/2 & 0 \\ 0 & 0 & 0 & T \end{bmatrix}; \tag{2}$$

$$u_{k-1} = 0; T = 1s$$

CTA:

$$F_{CTA} = F_{CV}; u_{k-1} = \hat{u}_{k-1} = \frac{a_t}{\sqrt{\hat{\dot{x}}_{k-1} + \hat{\dot{y}}_{k-1}}} \begin{bmatrix} \hat{\dot{x}}_{k-1} \\ \hat{\dot{y}}_{k-1} \end{bmatrix} \tag{3}$$

where a_t is given as $67m/s^2, -67m/s^2, 33m/s^2, -33m/s^2$.

CT:

$$F_{CT} = \begin{bmatrix} 1 & \frac{\sin\omega T}{\omega} & 0 & \frac{1-\cos\omega T}{\omega} \\ 0 & \cos\omega T & 0 & -\sin\omega T \\ 0 & \frac{1-\cos\omega T}{\omega} & 1 & \frac{\sin\omega T}{\omega} \\ 0 & \sin\omega T & 0 & \cos\omega T \end{bmatrix}, u_{k-1} = 0 \tag{4}$$

where ω is given as 14 $deg/s, -14$ $deg/s, 7$ $deg/s, -7$ deg/s.

Once the model set is determined, the measurement can be applied to estimate the object state at different time steps. The specific method is the so-called Kalman filter[7]. In short, the estimated state at time $k-1$ is used to obtain the predicted value at time k. This predicted value and the measured value at time k are combined by a Kalman coefficient. Additionally, this coefficient depends on the difference between the predicted value and the measurement and the corresponding variance and covariance.

The mathematical description of the Kalman filter is given as formulas, for the $i-th$ model $m^{(i)}$ in model set with initial condition $(\overline{x}_{k-1|k-1}, \overline{P}_{k-1|k-1})$, the state transition equation needs replacing for different model:

$$\hat{x}_{k|k-1}^{(i)} = F_{k-1}^{(i)} \overline{x}_{k-1|k-1} + G_{k-1}^{(i)} \overline{w}_{k-1}^{(i)} \tag{5}$$

$$P_{k|k-1}^{(i)} = F_{k-1}^{(i)} \overline{P}_{k-1|k-1} {F_{k-1}^{(i)}}^T + G_{k-1}^{(i)} Q_{k-1}^{(i)} {G_{k-1}^{(i)}}^T$$

$$\tilde{z}_k^{(i)} = z_k - H_k^{(i)} \hat{x}_{k|k-1}^{(i)}, \quad S_k^{(i)} = H_k^{(i)} P_{k|k-1}^{(i)} H_k^{(i)T} + R_k^{(i)}$$

$$K_k^{(i)} = P_{k|k-1}^{(i)} {H_{k-1}^{(i)}}^T S_k^{(i)-1}$$

$$\hat{x}_{k|k}^{(i)} = \hat{x}_{k|k-1}^{(i)} + K_k^{(i)} \tilde{z}_k^{(i)}, P_{k|k}^{(i)} = P_{k|k-1}^{(i)} - K_k^{(i)} S_k^{(i)} K_k^{(i)T}$$

As explained previously, x is the state, F and G are the transition matrix and control matrix. Moreover, P represents the uncertainty covariance, z is the measurement, and H is the measurement matrix, which is used to project the measurement onto the state, and R is the measurement noise. S is the uncertainty covariance projected into the measurement space. K is the Kalman coefficient to estimate the state.

For the $i-th$ model in the model set, after obtaining the measurement at time k and the predicted value at time $k-1$, the corresponding estimated values x_i and uncertainty S_i will be provided by Kalman filter. Then different multiple-model fusion algorithms can be used to fuse nine states. According to the literature [11], there are also multiple algorithms to choose at this step: the Autonomous multiple model (AMM)algorithm, Generalized pseudo-Bayesian algorithm of first-order (GPB1), and of second-order (GPB2), interacting multiple-model (IMM) algorithm, B-best based MM algorithm and Viterbi-based MM algorithm. For simplicity, we chose the most intuitive and straightforward Autonomous MM (AMM) Algorithm.

The concrete approach is:

*Step1.*Calculating nine states of estimation by applying Kalman filter in nine models in the model set.

*Step2.*Calculating the probability of each estimation value, with the formula:

$$\mu_k^{(i)} = \frac{\mu_{k-1}^{(i)} L_k^{(i)}}{\sum_{j=1}^{M} \mu_{k-1}^{(j)} L_k^{(j)}}, \ with \ L_k^{(j)} = \mathcal{N}(\tilde{z}_k^{(i)}; 0, S_k^{(i)}) \tag{6}$$

Where z is the measurement, S is the uncertainty obtained from the Kalman filter. The model probability of the CV model was initialized to $\mu_{CV0} = 0.99$ and the remaining model probabilities were all initialized to the value $\mu_0 = (1 - \mu_{CV0})/8 = 0.00125$.

*Step3.*Fusing all estimation values and process noises w.r.t the probability:

$$\hat{x}_{k|k} = \sum_{i=1}^{M} \hat{x}_{k|k}^{(i)} \mu_k^{(i)}, \; P_{k|k} = \sum_{i=1}^{M} [P_{k|k}^{(i)} + (\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)})(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)})^T] \mu_k^{(i)} \tag{7}$$

Finally, after several times of estimations, we can get the most accurate and precise estimation of the current state and the probability of fused model. Next, the process can continue, except updating the probability for prediction. As a result, the prediction state and variance will be presented.

3.2 Implementation

3.2.1 Dataset

In this project, we decide to use the test set of the tracking part of the KITTI dataset. On the one hand, the image data collected by the camera were used to apply the mono-camera-based tracking model in order to detect the vehicle and locate it, on the other hand, they are utilized for visualization in the end as well. The point cloud data collected by Lidar is applied to the Lidar-based model, which is also used to detect and locate the position of the vehicle. From a practical point of view, Lidar, as a sensor that can accurately describe location information, is more suitable for tracking tasks because it can provide a result of higher quality and more stable location information. However, the purpose of this project is to explore the possibility of tracking tasks using a mono camera, so we use the tracking results of Lidar as the ground truth to evaluate the predicting results of the camera. Other auxiliary data such as ego-vehicle GPS information is used to transform the ego coordinate system to the world coordinate system, and for more accurate positioning and visualization, the camera parameters are used to calibrate the camera.

3.2.2 Detection and Tracking

Firstly, the J3DT algorithm is used to determine and track the location of the target vehicles. In this case, the input is sequential image frames, and the output is the pose information of vehicles in each frame and the unique ID of each vehicle. The shape of the vehicle can be inferred as well, but it is useless for predicting task in this project. Similarly, the AB3DT algorithm is used to perform similar processes on the point cloud frame, except the input is a series of point cloud frames. An example of Lidar and camera result comparison is shown as Figure 2.

J3DT algorithm[2] The detail of J3DT can be described as follows:

Step1. Objects detection in each image with a neural network called faster-RCNN. The out is coordinates of 2d bounding box of vehicles.

Step2. Shape inference using a feature exactor and CNN to get 3d information such as the shape, the orientation, and the distance.

Step3. Object tracking from different but continuous frames with another neural network LSTM because of a better result comparing the classical model Kalman filter.

In implementation, we used the code provided by the author, and because of memory issues, we were unable to train the entire neural network, so the pre-trained model was also used in the experiments.

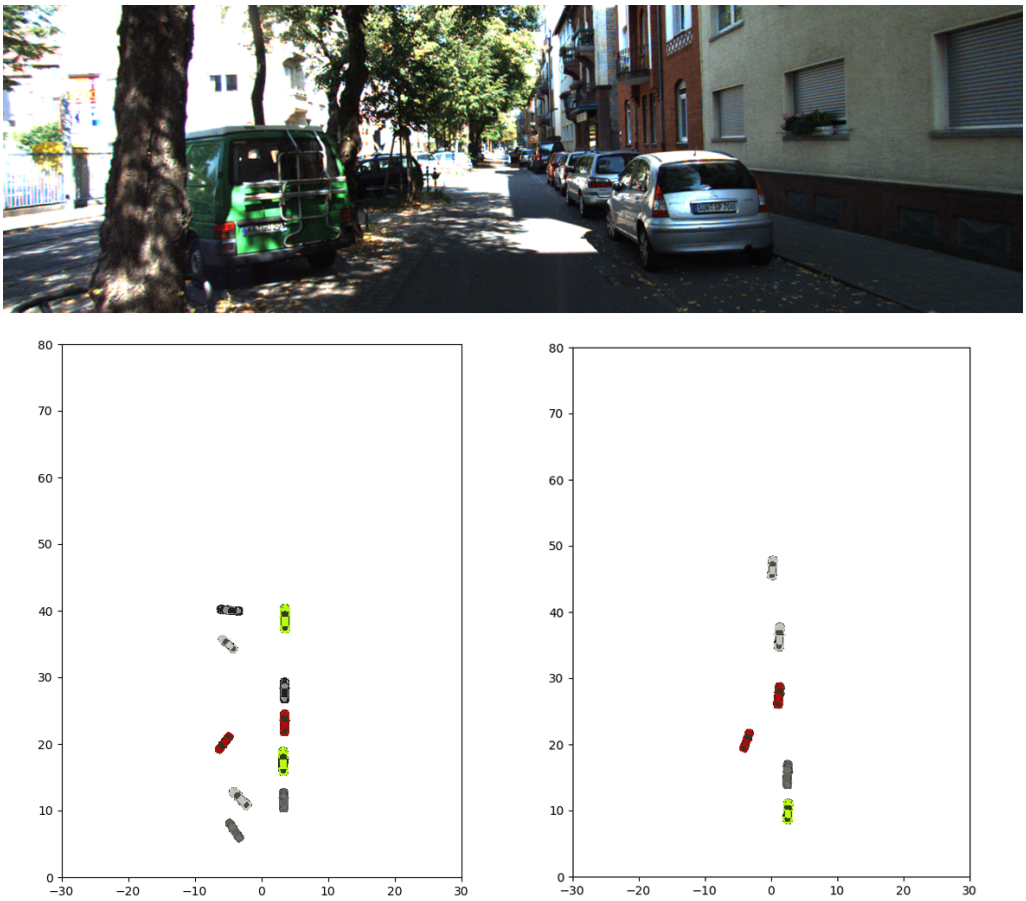

Figure 2: visualization of Lidar detection result(bottom lelf), mono-camera detection result(bottom right) and the realistic scenario(top)

AB3DT algorithm[6] The workflow of AB3DT is relatively more straightforward than J3DT model.

Step1. Object detection by a neural network specifically for the point cloud data called pointRCNN, with which we can directly get the 3D bounding box from the point cloud.

Step2. Object tracking from continuous point cloud frames, the classical algorithm Kalman filter is used.

3.2.3 Pose information conversion

In order to predict the trajectory, the pose information of each vehicle will be converted. In this project, because the non-linear point model is used to describe the vehicle, only 2D position information is required, which means we discard the shape and yaw angle information of the vehicle. Next, the position information needs to be converted from the ego coordinate system to the world coordinate system, so the motion of the target vehicle will not be influenced by the motion of the ego-vehicle.

Concretely, we use the GPS information from the KITTI dataset, which contains latitude and longitude of ego-vehicle at each frame. According to the Haversine formula, yaw angle and the distance in the x and y-direction relative to the original position can be calculated:

$$\theta = (yaw_n - yaw_{orig}) \tag{8}$$

$$\Delta_{lon} = (lon_n - lon_{orig}),\ \Delta_{lat} = (lat_n - lat_{orig})$$

$$distance_x = 2R \arcsin(\cos(lat_{orig}) \cos(lat_n) \sin(\Delta_{lon}/2)^2)^{\frac{1}{2}}$$

$$distance_y = 2R \arcsin(\sin(\Delta_{lat}/2))$$

Here R is the radius of earth-surface in meters, in our experiment we set it as 6378137, yaw and yaw_{orig} are the current and original yaw angle of ego-vehicle, lon and lat represent longitude and latitude, respectively.

After getting relative distance and yaw angle, a affine transformation could be applied to transform coordinate system as following formula:

$$\begin{bmatrix} x_{new} \\ y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & distance_x \\ 0 & 1 & distance_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} \tag{9}$$

Here x_n and y_n is the coordinates of target vehicle before transformation. θ, $distance_x$ and $distance_y$ are result of Haversine formula. The transformed coordinate is the output of this step and will be used for the next processing.

3.2.4 Trajectory interpolation

Because of the occlusion and unstable detection, the same vehicle may not appear in sequential frames sometimes, so interpolating the position information of the missing vehicle is required for the following processing.

When a target vehicle is predicted, the requirement is that it appears at least 3 times in the past n frames, so that its position of the remaining frames can be obtained by interpolation. The number of occurrences influences the prediction result: the larger it is, the less interpolation is needed, therefore the more reliable the history trajectory is, the better the prediction achieves.

At the beginning of this experiment, we used spline equal-interval interpolation in either direction of x or direction of y. This method is easy to implement, but the problem is that the trajectory cannot be

evenly interpolated: when the target vehicle moves in the vertical direction of the interpolation, or when curvilinear movements such as turning are performed, this method will cause a large interpolation error. Therefore, we tried another performing interpolation method, the equidistant spline interpolation, under the assumption of uniform movement. The comparison of an example is shown as Figure 3.

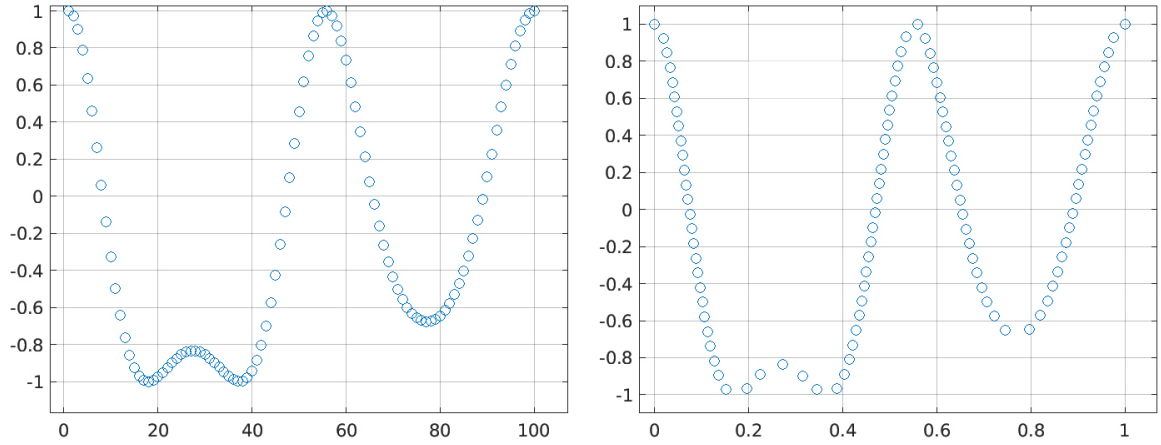

Figure 3: interpolation result comparison: x-direction evenly interpolation(left), equidistant interpolation(right)

Interpolation occurs only when the trajectory is not continuous in sequential frames. If there are already trajectory points in the frames, the original data will be used without interpolation.

3.2.5 Trajectory prediction

After the above processes, we obtained the position information of each vehicle in sequential frames over a period of time. Formally, the trajectory can be described as $\{(x_0, y_0), (x_1, y_1), ..., (x_i, y_i)\}^{(t)}$, where i is the vehicle id and t is the frame id. In the prediction part, after choosing the target vehicles, each position of its history trajectory will be observed, then the AMM algorithm will be used to produce the optimal estimate of the current position, including state $x = (x, \dot{x}, y, \dot{y})^T$ and corresponding covariance σ^2.

As explained in section 2, the process of estimating the state at each timestamp is determined from the weighted average of the current observation and the predicted value at the previous timestamp, and the weight depends on the uncertainty of the predicted value and the observed value. More concretely, in order to calculate the predicted value, different motion models are utilized to represent common 9 behavior: the constant velocity model, the constant tangential acceleration model for acceleration: $67m/s^2, -67m/s^2, 33m/s^2, -33m/s^2$ and constant turns model of 14 $deg/s, -14$ $deg/s, 7$ $deg/s, -7$ deg/s. The weighted average result of these models determines the predicted value at the next timestamp, and the weight depends on how well each model fits the history trajectory: The better the model fits, the higher the weight is, which will ensure that predication always works properly in different situations. After obtaining the best estimate of the last point of the history trajectory, i.e. the current point, the prediction part of the AMM algorithm will continue based on this estimate. Finally, the predicted position and the variance of it will be obtained. More mathematical details can be found in section 2.

Figure 4 shows an example of a comparison of predicted result and ground truth. As the figure shows that the longer the prediction performs, the farther away the predicted position from the ground truth is, while the higher the uncertainty is. Furthermore, the uncertainty grows faster in the dimension with higher velocity, which is consistent with intuition.

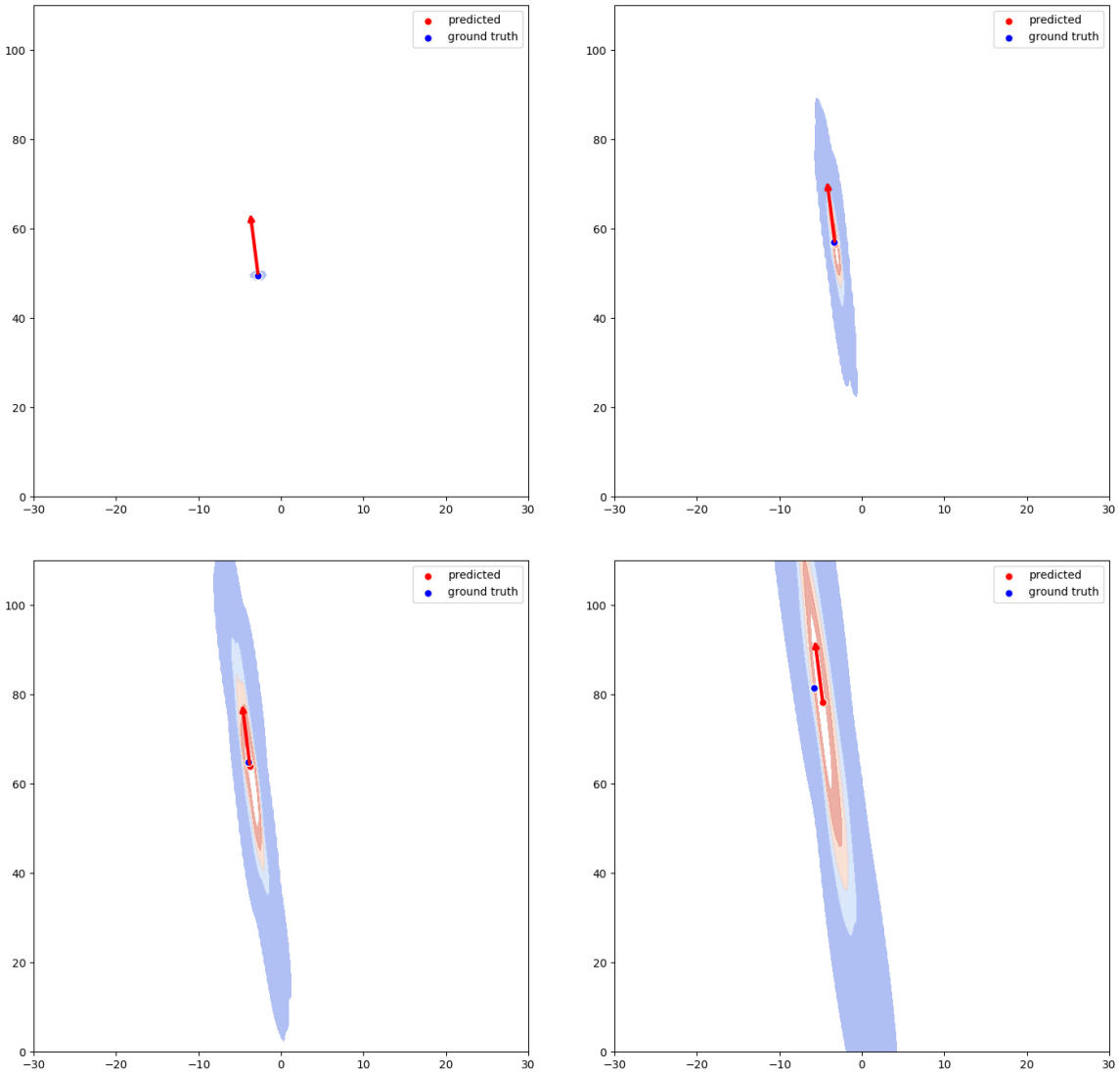

Figure 4: comparison of predicted result and ground truth at frame 0 (top left), frame 5 (top right), frame 10 (bottom left) and frame 20 (bottom right). The contour presents the probability distribution

3.3 Evaluation

For the evaluation of predication result, the most intuitive metric is the accuracy, which was measured in terms of position errors:

$$Error_k^{(p)} = ((x_k - \hat{x}_{k|k})^2 + (y_k - \hat{y}_{k|k})^2)^{\frac{1}{2}} \tag{10}$$

while x and y represent the corresponding coordinates respectively. In the original paper, the velocity error is compared as well. However, in our detection case there is no velocity information available, so only the position error will be utilized to evaluate the result.

In the experiment we chose 4 scenarios for comparison: constant moving case, still case, braking case and turning case. And we use the Lidar-based tracking result as the ground truth, and compare the prediction result of camera and Lidar, which means the partial camera-based tracking result are used to predict the future trajectory for camera-based prediction, while the partial Lidar-based tracking result are used to predict the future trajectory for Lidar-based prediction. The position error value over time of constant moving case and still case is shown in Figure 5.

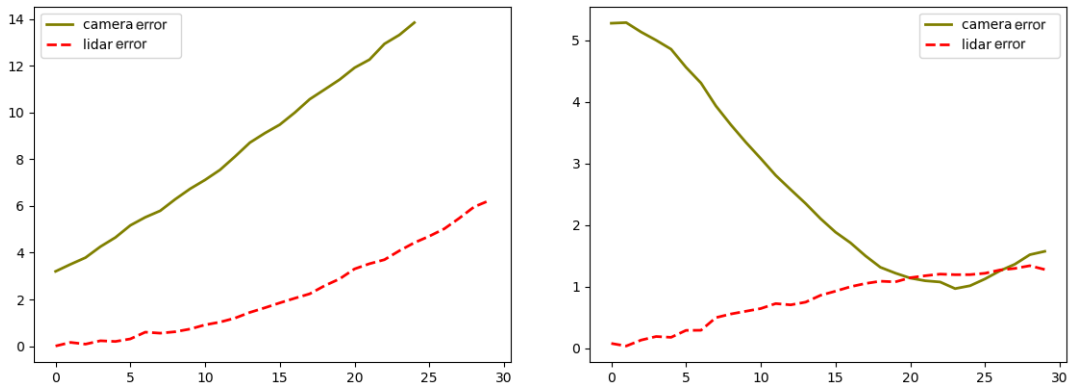

Figure 5: comparison of position error over frames: moving vehicle case (left) and still vehicle case (right), the x-axis represents the period after receiving the measurement from beginning of prediction in frames, and the interval between each frame is 0.1 second, the y-axis represents the position error value at the corresponding frame.

As shown, in the first case, the position error of Lidar-based prediction is always less than the camera-based. Furthermore, the position error is gradually increasing over time. When predicting the position after $2.5s$, the position error given by Lidar-based prediction is about 4 meters, while the velocity of the target vehicle is about $15m/s$. Considering the size of the vehicle, the result is acceptable and understandable.

However, the result of camera-based prediction is not so practical. Even in the beginning, the error has already reached 3 meters. The increase rate also exceeded the results of Lidar. This result is expected because the distance estimation from the monocular camera is not accurate enough, so even if at the initial position, which represents the estimation based on the past and current measurement results, there is an unneglected gap with ground truth. Moreover, because of errors in the history measurement, there is an impact on the prediction as well.

In the second case, something different happens: first of all, the error of Lidar-based prediction stays in a very low range and rises slowly over time, as expected. However, the error of the camera-based prediction gradually decreases from the initial high level to a lower level and then rises again. The possible reason for

this phenomenon is still the inaccuracy of the distance measurement, which makes the predictor misjudge the object is still moving at low velocity: the initial position is 5m away from the position of ground truth, and it moves towards the ground truth position. Lastly it goes far away again. It explains why error value decreases first and then increases.

Similarly, the comparison of the braking case and the turning case is shown in Figure 6. It can be seen from the figure that the braking case looks like a hybrid of the still case and the constant moving case: the camera-based prediction shows a trend of decreasing first and then increasing but relatively gently, while the overall error value is higher than Lidar-based prediction, as expected. In the turning case, the camera-based prediction differed from the ground truth from the beginning, but the growth rate of error value is similar to that of Lidar-based prediction.

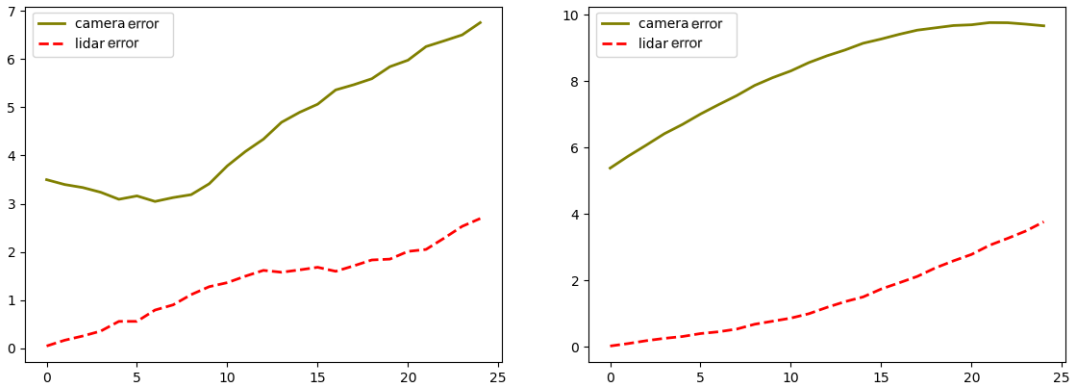

Figure 6: comparison of position error over frames: braking vehicle case (left) and turning vehicle case (right)

Through evaluation and analysis, it can be seen that the accuracy of distance measurement has a crucial effect on the quality of the prediction. Under the algorithm of state of the art, accuracy using only a monocular camera to measure the distance is not enough to apply to prediction. On the other hand, by evaluating the prediction results of Lidar-based measurement, it proves the AMM algorithm based on the motion model can predict the future trajectory of the object within a certain range.

4 Conclusion

This project discusses the trajectory prediction of vehicles. It starts with the detection of the vehicle from the raw data, including image and point cloud. And finally obtains the trajectory of the target vehicle in the next few seconds. First, we researched detection and tracking algorithms, and finally chose two methods of state of the art: one uses image frames to detect objects and infer distance. The other utilizes point cloud for object detection and distance measurement. Based on the detection results, tracking is performed to ensure that the ID of the target vehicle is consistent in frames. Then according to the tracking results, some pre-process will be performed, such as global coordinate system transformation, trajectory interpolation. Finally, an autonomous multiple model (AMM) algorithm based on the motion model is applied. On the one hand, it is used to obtain the best estimate of the state of the target vehicle at each moment. On the other hand, it is used to predict the trajectory in the future, including coordinates, velocity, and corresponding covariance. In the evaluation part, we found that the result of image based prediction is different from the result of point cloud based prediction, which proves mono-camera cannot

perform trajectory prediction alone because of inaccuracy on distance measurement.

Concretely, this project achieves following goals:

1. Researched recent published papers in the area of object tracking and trajectory estimation.

2. Implemented at least one of the according algorithms.

3. Evaluated its performance regarding trajectory estimation precision and reliability on a suitable data basis.

4. Considered and implemented improvements to the limitations identified.

5 Further Work

Based on this research project, many extensions are possible. For example, the following are several ideas that can be investigated:

1. Based on the evaluation, a intuitive idea is to improve the accuracy of distance measurement of the camera, such as using the stereo camera instead of the mono camera, which uses epipolar geometry to measure distance instead of inferring distance.

2. Another idea is to use the camera measurement result as a supplement to the Lidar measurement result, i.e. try to use the extended Kalman filter or unscented Kalman filter for sensor fusion instead of using camera alone.

3. The motion model used in this project is a relatively simple: the non-linear point model, so the state given only contains simple position and velocity. In the future, more complex models could be applied, such as kinematic bicycle model with PID controller, with which the information such as yaw angle for more accurate estimation and prediction can be used.

4. The camera as a sensor for distance measurement is not accurate enough, but its ability to identify other information is higher than Lidar. Therefore, other methods are also worth considering. For example, we can use the camera to identify lane lanes, traffic lights, or the turn signals of the target vehicle to help better trajectory prediction.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[2] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krähenbühl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. 2019.

[3] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *european conference on computer vision*, pages 354–370. Springer, 2016.

[4] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[5] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[6] Xinshuo Weng and Kris Kitani. A Baseline for 3D Multi-Object Tracking. *arXiv:1907.03961*, 2019.

[7] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[8] Peiliang Li, Tong Qin, et al. Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 646–661, 2018.

[9] Cynthia Sung, Dan Feldman, and Daniela Rus. Trajectory clustering for motion prediction. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1547–1552. IEEE, 2012.

[10] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015.

[11] Ryan R Pitre, Vesselin P Jilkov, and X Rong Li. A comparative study of multiple-model algorithms for maneuvering target tracking. In *Signal Processing, Sensor Fusion, and Target Recognition XIV*, volume 5809, pages 549–560. International Society for Optics and Photonics, 2005.