Perception of Low-Cost Autonomous Driving

Tae Eun Choe, Guang Chen, Weide Zhang, Yuliang Guo, Ka Wai Tsoi

{choetaeeun, chenguang09, zhangweide, yuliangguo, kawaitsoi}@baidu.com

Baidu USA

Abstract

The paper introduces perception algorithms for low-cost autonomous driving in Apollo, the largest open autonomous driving platform with a full stack of H/W and S/W developed by the autonomous driving community. We review pros and cons of each sensor and discuss what functionality and level of autonomy can be achieved with such sensors. We will also discuss specific perception modules such as dynamic object detection and stationary object detection. We further explain sensor fusion using Dempster-Shafer theory. We will also introduce virtual lane line and camera calibration in autonomous driving.

1. Introduction

The development of the longer range and higher resolution lidar enabled Level-4 autonomous driving with more accurate perception and localization in a certain area. However, the lidar is a less reliable sensor under an extreme weather condition such as heavy rain or snow. Furthermore, the expensive cost of the lidar prevents automakers' production of a consumer autonomous car with the lidar. On the other hand, a camera is more cost-effective and more robust to weather and

it is a key sensor for traffic light recognition and lane line detection. We will present algorithms to achieve autonomous driving using economic sensors such as a camera and a radar. There are four main pillars in camera and radar based perception: pre-processing, deep network, post-processing, and fusion. The whole diagram is shown in Figure 1.

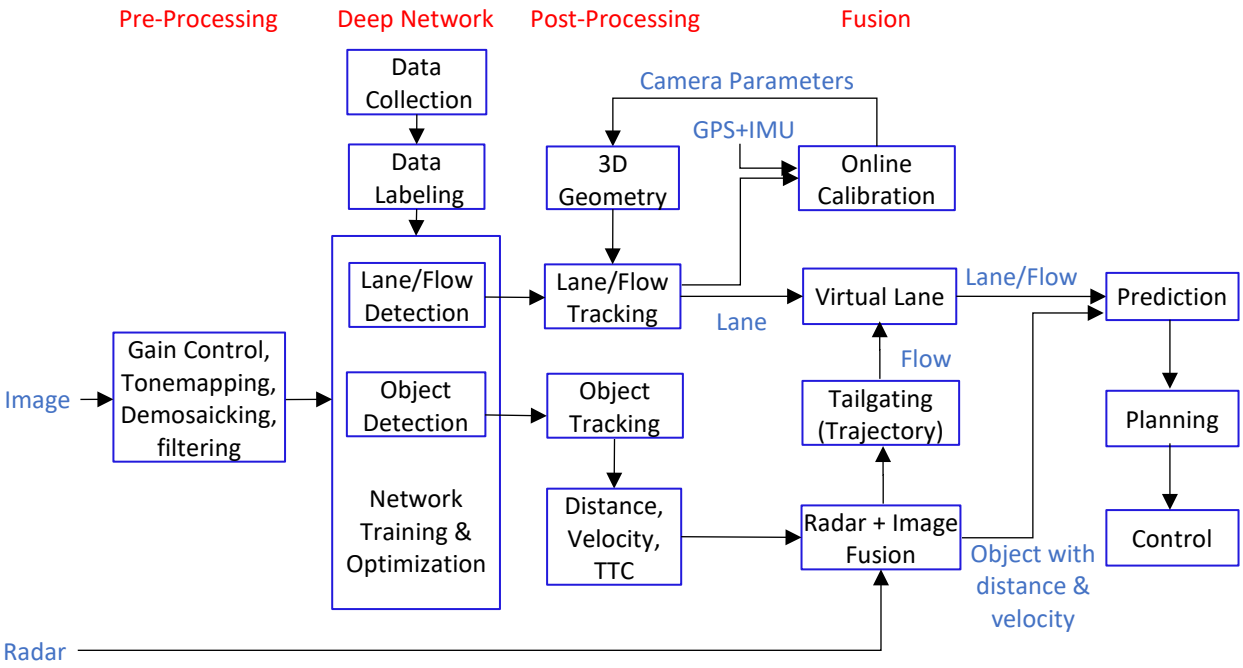

Figure 1: Flow diagram of vision-based autonomous driving

Codes and documents are available in github (Apollo2017). Apollo also provides ApolloScape (ApolloScape2018), the largest open autonomous driving data for training and test. We believe that developing autonomous driving algorithms together in an open platform and sharing training and test data are the best way to achieve the safe and agile self-driving for all. Figure 2 shows an image of an autonomously driving vehicle using Apollo platform.

Figure 2: Autonomous driving vehicle using Apollo platform

2. Data Collection and Labeling

2.1. Balanced Data Collection

In the deep learning era, data collection and labeling become more important tasks. The labeled data should be well balanced over time, weather, and road conditions. The data should cover night, dawn, sunrise, strong shadow, sunset in one axis. Another axis is weather with sunny, rainy, snowy, foggy weather. The third axis will be varying road conditions such as straight, curved, fork, merged, or intersection. All those scenes should be evenly distributed on each axis. Figure 3-(a) illustrates the actual data distribution for different environments and 3-(b) shows the distribution of data after collecting them in a balanced way.

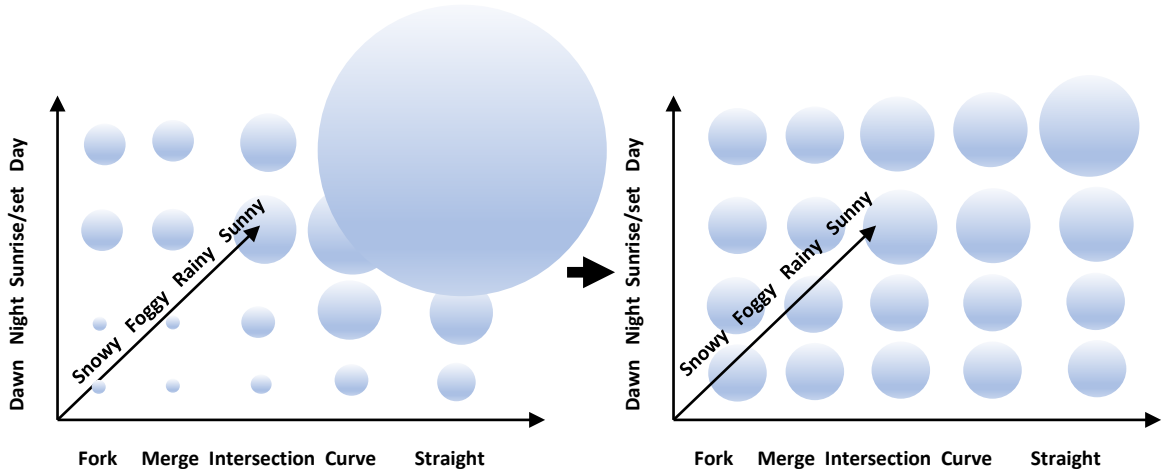

Figure 3: Data distribution (a) Unbalanced real data (b) Balanced data

2.2. Data Labeling

Before driving, the routine to log vehicle information (yaw rate, speed), GPS, wiper(rain, snow), low/high beam, timestamp, location and all sensor data should be implemented. The more such data was saved, the easier labeling process will be. During driving, a car should be driven in the center of a road as much as possible to imitate autonomous driving. In addition, there should be a simple button to save last 30 seconds of data whenever the car experience specific or rare events. After driving, any duplicated or similar scenes should be removed especially when a car stops. The face of a pedestrian and the license plate of a vehicle should be also removed to protect the privacy. After such processes are completed, data can be labeled.

2.3. Auto-labeling

Since manual labeling is costly and prune to make an human error, automatically labeled data should be added in training dataset. The good candidate of auto-labeling is a stationary object such as a lane, a traffic light, a traffic sign, or any road landmark. First, near-view objects were detected

by an existing detector while driving. After driving 200 meters, the 200-meter previous scene is reviewed. Assuming near-view object detection and motion estimation are accurate, the accumulated set of detected near-view objects, we can label objects far-away. Stationary objects such as lane, traffic lights, traffic sign, or any landmark can be auto-labeled in this manner.

For smart recording, we designed multiple events such as deceleration, curves, cut-in, cut-out, bump, etc. When such event occurs, the data before and after the event will be saved automatically.

For auto-labeling, speed and yawrate from Controller Area Network (CAN) bus or inertial measurement unit (IMU) data should be recorded for accurate motion estimation. IMU is a useful sensor to measure a vehicle pose. After we accumulate the pose of an ego-vehicle using IMU over time, we can easily estimate any motion from time *t* to time *t-n.*

When the pose at time t is $M_{t\rightarrow t-1}$, where M is 4x4 matrix with rotation and translation elements, the motion from time t to t-n will be

$$M_{t\rightarrow t-n} = M_{t\rightarrow t-1} M_{t-2\rightarrow t-3} M_{t-3\rightarrow t-4} \dots M_{t-n+1\rightarrow t-n}$$

This simple motion matrix $M_{t\rightarrow t-n}$ will convert the current vehicle pose to *n* times previous pose directly. We search such motion until the ego-car go back to 200m back and project the trajectory in the image. Then we can label the near-view stationary objects automatically without manual labeling. Contrary to the manual labeling, this auto-labeling method can label 3D depth of each object. In addition, the road surface is also reconstructed in 3D to label hill crest, bump, or clover leaves. The example of the automated labeling is illustrated in Figure 4.

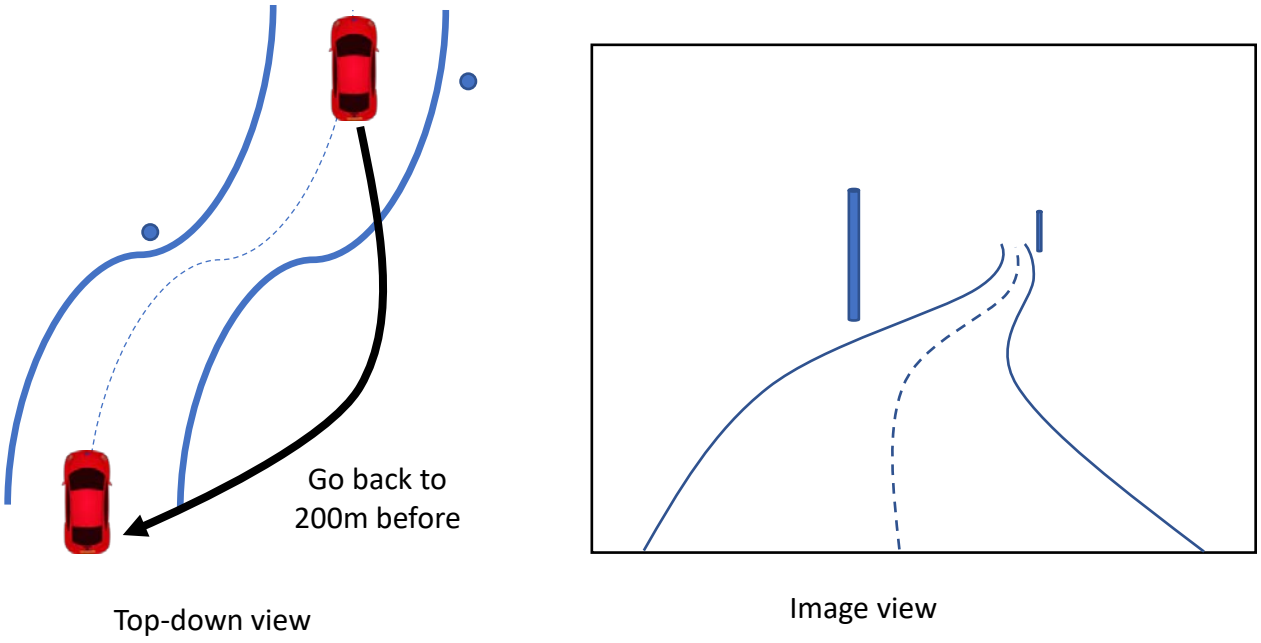

Figure 4: Automated labeling of stationary objects using vehicle motion

As shown in Figure 5, the auto-labeling can label invisible lane lines (Figure 5-(a)) and 3D lane lines (Figure 5-(b))

(a) **(b)**

Figure 5: Example of automated labeling of lane lines (a) Hidden lane lines are labeled (b) 3D lane lines are labeled.

3. Network training

Pre-processed image is transferred to a deep neural network (DNN) for object detection, object tracking, lane line detection, landmark detection and other computer vision problems. For real-time processing of high framerate and high resolution imagery data, network compression is required. In literature, there are two main network compression approaches:

Lower-bit approximation: Rather than using the conventional 32bit float as a weight representation, Float32 is quantized into INT8 to achieve real-time implementation (Dettmeers2016).

Network layer reorganization: Another approach is to re-organize layers of a network. When there are multiple tasks, the network structure can be re-organized by sharing common layers and removing unnecessary layers.

4. Object detection

In a traffic scene, there are two kinds of objects, stationary object and dynamic object. Stationary objects include a lane, a traffic light, a streetlamp, a barrier, a bridge above the road, and skyline. Among dynamic objects, we care pedestrian, car, truck, bicycle, motorcycle, animal and more. For the object detection, YOLO V3 (darknet) is used as a base network (Redmon2018). In YOLO V3, more attributes of an object was added such as 3D size, 3D position, orientation and type. Detected multiple objects were tracked across multiple frames using a cascade-based multiple hypothesis object tracker.

5. Lane detection

Among stationary objects, a lane is a key stationary object for both longitudinal and lateral control. An ego-lane guides lateral control and any dynamic object in the ego-lane determines longitudinal control. We use the same YOLO (darknet) as a base network and add extra lane tasks at the later part of the network to detect lane numbers (left, right, next left, next right, curb lines) and lane types (white/yellow, solid/broken, fork/split).

6. Sensor fusion

Installing multiple sensors around a car facilitates full coverage of an environment and redundancy for safety. Each sensor has different characteristics: the range of a lidar is short but its 3D measurement is accurate; a radar also provides longitudinally accurate distance and velocity measurements but laterally inaccurate; A camera is accurate for lateral measurement but worse for longitudinal measurement. We learn a prior and belief function of each sensor and fuse all sensor output using Dempster-Shafer theory (Wu2002).

7. CIPV detection and Tailgating

The trajectories of all vehicles were captured with respect to the self-driving vehicle (SDV). Among them closest in-path vehicle (CIPV) is chosen for tailgating that SDV follows a front car. When there is no lane such as at intersection but CIPV, SDV can still drive smoothly by following CIPV since CIPV’s past trajectory provides SDV’s future path.

8. Camera calibration

Camera calibration is challenging but the most important procedure. There are three categories of camera calibration:

Factory (initial) calibration

In the factory, we can estimate intrinsic and extrinsic camera parameters using fixed targets. However, the pose of a camera changes over time. Therefore, the pose of cameras needs to be updated frequently.

On-line calibration

We need to estimate long term pose of the camera with respect to a car body. On-line camera calibration module calibrates camera pose every single frame. Even 0.3 degrees change in pitch angle can impact totally incorrect control. For calibration, any object on the road can be used, such as parallel lane lines, vertical landmarks, or any known size of cars or optical flow.

Instant pose estimation

The pose of a car body changes every frame. Especially passing through the bump, the pose changes a lot which is not the camera pose issue but the car body pose issue. The pose can be estimated by IMU but they are too noisy to use directly. The pose can be estimated by visual features. By tracking the stationary objects, we can estimate motion, the history of pose, over time. The estimated instant pose will provide much more accurate 3D perception of the scene.

9. Virtual lane

All lane detection results and tailgating flow will be combined spatially and temporarily to induce the virtual lane. The virtual lane provides a path to drive even there is no lane line. The virtual lane output is fed to planning and control modules for the actuation of the self-driving vehicle.

10. Conclusion

In this paper, we showed the overall perception algorithms for low-cost autonomous driving using a camera and a radar. As deep neural network is the key tool to solve perception issues, data collection and labeling became more important tasks. For the sustainable data labeling, auto-labeling is introduced. For autonomous driving, dynamic object detection/tracking and stationary object detection algorithms are discussed. To handle multiple sensors fusion, Dempster-Shafer based sensor fusion algorithm is used. Additionally CIPV, tailgating, and camera calibration algorithms are introduced.

References

(Apollo2017) Apollo: Open Autonomous Driving Platform, https://github.com/ApolloAuto/apollo

(ApolloScape2018) Apollo Data: Open Data for Autonomous Driving, http://data.apollo.auto/?locale=en-us&lang=en

(Dettmeers2016) T. Dettmers, "8-Bit Approximations for Parallelism in Deep Learning," ICLR 2016

(Pan2017) Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang: "Spatial As Deep: Spatial CNN for Traffic Scene Understanding," AAAI 2018

(Redmon2018) J. Redmon, A. Farhadi, "YOLOv3: An Incremental Improvement," https://arxiv.org/abs/1804.02767

(Wu2002) H. Wu, M. Siegel, R. Stiefelhagen, J. Yang, "Sensor fusion using Dempster-Shafer theory II: static weighting and Kalman filter-like dynamic weighting," IEEE Instrumentation and Measurement Technology Conference, 2002