An Effective Method for Self-driving Car Navigation based on Lidar

Meng Liu[1](✉), Yu Liu[2], Jianwei Niu[1], Yu Du[3], and Yanchen Wan[2]

[1] State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China
liumeng_scse@buaa.edu.cn

[2] State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing 100191, China

[3] College of Robotics, Beijing Union University, Beijing, China

Abstract. Existing navigation methods are generally based on GPS or cameras and these methods have limitations in terms of signal strength and brightness. To overcome drawbacks of navigation methods above, we propose a Lidar-based Navigation Approach (LNA) to predict movement trajectory of self-driving vehicles through road edges information, and this approach is a fitting and real-time regression method. By combining regression model with vehicle coordinate system, navigation trajectory is accurately generated. Experiments on common road scenarios demonstrate that our approach is effective to improve navigation techniques.

Keywords: Lidar · Self-driving car · LNA · Navigation
Linear regression

1 Introduction

Navigation plays an important role in autonomous driving scenarios. Currently, navigation methods based on Global Positioning System (GPS) and images are very common. One of popular navigation methods is joint application of global positioning system (GPS) and IMU [1]. [2] shows a vision navigation method. [3,4] use vision technology to extract lane lines for navigation. Modern autopilot systems are commonly equipped with laser scanners. Lidar is very important in unmanned-vehicles areas, and it can provide very accurate environment and road profile information. Therefore, this paper aims at using lidar for navigation. We are not concerned about obstacle detection in complex traffic conditions, positioning and obstacle avoidance problems. Therefore, in this paper, a Lidar-based Navigation Approach (LNA) is proposed for unmaned-vehicles navigation.

Our approach uses linear regression methods, which can fit outline characteristics of road. We use lidar to get real-time point cloud as input. After many preprocessings, we extract clear and effective road sides information. Point cloud of two road edges are linearly fitted to get two linear models. We merge the two

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2018
I. Romdhani et al. (Eds.): CollaborateCom 2017, LNICST 252, pp. 718–727, 2018.
https://doi.org/10.1007/978-3-030-00916-8_65

linear models and translate them into vehicle coordinate system. A series of vehicle motion trajectories are generated. Then, we calculate lateral control data to control the car. Finally, LNA is evaluated in different types of lidars and different cars. Experiment results show that LNA can achieve lidar-based navigation within a general environment on different platforms. The LNA has many advantages, such as large detection range, high accuracy and strong reliability. Except that, it is also applicable to many scenarios which have obvious outline features, such as highway and urban road.

2 Related Work

Existing lidar technologies include maps making, environment detection, realtime positioning and path planning. In this paper, we briefly review previous work on Simultaneous Localization and Mapping(SLAM) and road environment detection.

Over the past years, SLAM develops very fast and there are many applications about SLAM [5]. SLAM can establish a model for spatial environment in process of movement with an absence of prior environment knowledge. [6] shows a SLAM making process. [7] shows a classic method of odometry and mapping using a lidar. [8] describes a method of using lidar for simultaneous localization and mapping. However, all of these SLAM methods mainly focus on environmental modeling and they cannot meet the requirements of realtime navigation.

[9] proposes a road boundary detection and tracking method based on 2d lidar, but this method cannot obtain enough road information. [10] describes a method of detecting road geometric features to identify road shapes. [11] proposes a road surfaces extraction method using a fuzzy cluster. However, these methods based on lidar cannot meet the requirements of realtime navigation. Some researches [12–14] use fusion methods by combing lidar with camera for environment sensing, but these work need complex sensors calibration and fusion algorithms.

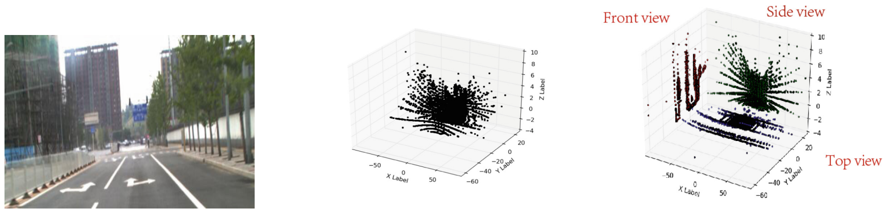
Therefore, in this paper, we propose a lidar-based realtime and concise road border modeling method for autonomous driving navigation—Lidar-based Navigation Approach (LNA).

3 The LNA Method

LNA is a lidar-based navigation approach for self-driving cars, which includes lidar data acquisition, data preprocessing, real-time fitting and computing trajectory. It takes lidar point cloud sets as input and clears noises firstly. Then, sparse process can reduce the amount of data and maintain effective data characteristics, which will help to achieve real-time fitting. And a Linear Regression (LR) or Support Vector Regression (SVR) method is followed for linear fitting. Finally, models are transformed into vehicle coordinate system to obtain a trajectory points set, which we use to calculate the lateral control value to reach the goal of navigation.

3.1 Data Preprocessing

This section includes data filtering, interception, classification and sparse processing. In filter stage, we clear noises. In interception stage, we choose point cloud which we have interests in. In classification stage, we get road shoulder and isolation belt information. For sparse processing stage, we reduce the computing load.



(a) Road image (b) Raw data (c) Projections in 3 views

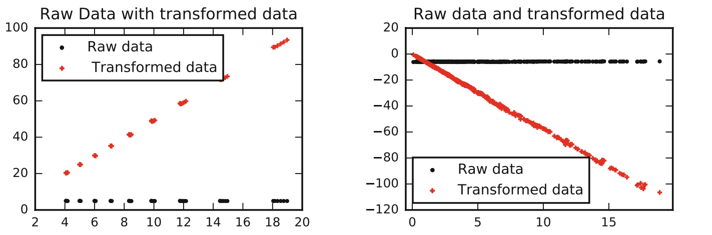
Fig. 1. Road image, point cloud raw data and projections.

Figure 1(b) shows that the collected point cloud has obvious noises. As shown in Fig. 1(a), these noises are mainly from the ground, trees, obstacles and other objects. From Fig. 1, we observe distribution of point cloud in three-dimensional space. Figure 1(c) shows projections of point cloud on top view, front view and side view. We need effectively exploit three views of point cloud distribution. To this issue, we use a meshing method to process point cloud in a three-dimensional spatial grid. Moreover, we use vehicle parameters, lidar placement parameters and some prior knowledge to filter point cloud. Taking HDL-32 lidar as an example, the vehicle height is 1.5 m, which can be defined as h_v. Height of lidar away from vehicle top is 0.3 m, written as h_{l-v}. Height of isolation belt is about 1.5 m, written as h_{ib}. Height of road shoulder is about 0.20 m, written as h_r. Vehicle width is 0.9 m, marked as w_v. Distance from vehicle front to mounting location is 2m, recorded as l_l. Taking lidar scanning distance into account, for HDL-32, we only consider point cloud in range of 20 m, recorded as l_{range}. As we control the highest speed at 20–40 km/h, which equals 5.6–11.1 m/s. And data fitting time is below 0.3 s. These make the forward reaction distance is sufficient. According to conditions above, we design a simple classifier and filter, as expressed by formula 1 and formula 2.

$$\begin{aligned}\mathrm{Filter}_R = &\{[\mathrm{z} > -(\mathrm{h}_v+\mathrm{h}_{l-v})]\&[\mathrm{z} < -(\mathrm{h}_v+\mathrm{h}_{l-v}-\mathrm{h}_r)]\}\&\{[\mathrm{y} > \mathrm{l}_l]\&[\mathrm{y} < \mathrm{l}_{range}]\}\& \\ &\{[\mathrm{x} > \mathrm{w}_v]\&[\mathrm{x} < (\mathrm{x}_{sam}+\mathrm{w}_{samp})]\}\end{aligned} \tag{1}$$

$$\begin{aligned}\mathrm{Filter}_L = &\{[\mathrm{z} > -(\mathrm{h}_v+\mathrm{h}_{l-v})]\&[\mathrm{z} < -(\mathrm{h}_v+\mathrm{h}_{l-v}-\mathrm{h}_r)]\}\&\{[\mathrm{y} > \mathrm{l}_l]\&[\mathrm{y} < \mathrm{l}_{range}]\}\& \\ &\{[\mathrm{x} < -\mathrm{w}_v]\&[\mathrm{x} > -(\mathrm{x}_{sam}+\mathrm{w}_{samp})]\}\end{aligned} \tag{2}$$

where the x_{sam} is minimum value of samples X set after filtered and classified. w_{samp} is width of samples in x direction, which also is obtained after filtered and classified. This could guarantee the robustness and universality of classifier in different road conditions.



(a) Road shoulder raw data and transformed data (b) Isolation belt raw data and transformed data

Fig. 2. Filter point cloud

The data set before fitting is projected onto the horizontal coordinate system of $x-y$. If the number of points is more than $N(N = 50)$, data set will be divided into 10 parts in the X direction, and each part randomly choose samples in proportion of $\lambda(\lambda = 90\%)$. Repeatedly sampling in value of λ^n until the number of samples is just below 50. The purpose of dividing set into 10 parts is to keep data descriptive ability for environmental information characteristics. If whole set randomly sampled in ritio of λ^n directly, it is very likely to make data distorted. Also, we make a data transformation by using formula 3 and formula 4 to reduce the fitting cost.

$$\begin{aligned} x_{rawData} &\rightarrow y', \\ y_{rawData} &\rightarrow x'. \end{aligned} \tag{3}$$

The data were originally a set of point cloud datasets approximately parallel to the X axis. This results in training was very difficult to fit, so we did the process.

$$y_i'' = y_i' \times x_i', \tag{4}$$

Eventually, we got the new data set of $D_{New} = (x_1', y_1''), (x_2', y_2''), ..., (x_n', y_n'')$, $n \approx 50$. D_{New} makes data very easy to fit, which greatly improves real-time performance of model training. Figure 2 shows the point cloud data after being preprocessed. Figure 2(a) shows road shoulder raw data after preprocessed and transformed data. Figure 2(b) shows isolation belt raw data after preprocessed and transformed data.

3.2 Real-Time Linear Fitting

In this paper, we use two linear models of LR [15,16] and SVR [17]. Using LR & SVR to fit a series of continuous points is a common sense. Linear regression is relatively simple, it is widely used and has very good effects. In industrial and commercial areas, it has extensive applications.

Linear Regression. Our model can be described as formular 5.

$$f_{LR}(x) = w^T x + b. \tag{5}$$

When choosing model, we exclude methods of multivariate linear regression and linear regression of higher degree. Multivariate linear regression function is proved not applicable for pointcloud. Higher degree model is also excluded. Firstly, it adds training costs and reduces real-time performance. Secondly, it is difficult to converge with few training samples. And final fitting effect shows higher degree model can not effectively fit road environment characteristics. For our model, formula 6 is the cost function. Our goal is to minimize the cost function, which is written as formula 7. And we use a gradient descent method to train it.

$$J_{LR}(\theta) = \frac{1}{2}\sum_{i=1}^{m}(f_\theta(x_i) - y_i)^2. \tag{6}$$

$$\min_\theta J_{LR}(\theta). \tag{7}$$

Support Vector Regression. The SVR regression function is similar to linear regression function, as shown in formula 8. We evaluate the difference between every sample ground truth value and every prediction value to determine whether it is used for cost. This can be seen from formula 9. The final cost function is designed as formula 10.

$$f_{SVR}(x) = w^T x + b \tag{8}$$

$$\max(0, |y_i - f_{SVR}(x)| - \varepsilon) \tag{9}$$

$$J_{SVR}(\theta) = \min_{w,b} \frac{1}{2}||w||^2 + \frac{1}{m}\sum_{i=1}^{m}\max(0, |y_i - f_{SVR}(x_i)| - \varepsilon) \tag{10}$$

In formula 10, if point cloud data is within region from upper fit line to lower fit line, ε makes loss function equal to 0.

3.3 Computing Movement Trajectory

The two models obtained after training need to be merged. There are two methods: predicting two movement trajectories before fusion or fusing two linear models before predicting trajectories. We choose the former way. A concise fusion way is to compute mean of two movement trajectories. We can compute steering angle in real time. As cars are equipped with GPS equipment, we can also transform trajetories lidar predicted into the GPS coordinate system for navigation. The linear fitting models need to be converted into vehicle coordinate system, that is the mounting location of VLP-16. $T(t_x, t_y, t_z)$ is a translation matrix. $R_z(\alpha)$, $R_x(\beta)$, $R_y(\gamma)$ are rotation matrices. Whole coordinate transformation process is denoted by formula 11. After simplified, formula 11 is written as formula 12 . The distance parameters used in translation transformation and angle parameters used in rotation transformation can be measured.

$$\begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & l_x \\ 0 & 1 & 0 & l_y \\ 0 & 0 & 1 & l_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\gamma & 0 & \sin\gamma & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\gamma & 0 & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{11}$$

$$Point_v = T(t_x, t_y, t_z) \cdot R_z(\alpha) \cdot R_x(\beta) \cdot R_y(\gamma) \cdot Point \tag{12}$$

The obtained trajectory points need to be processed by formula 13 to get vehicle trajectory set, which is denoted by $T = \{(x_1, f(x_1)'), (x_2, f(x_2)'), (x_3, f(x_3)'), ..., (x_n, f(x_n)')\}$.

$$f(x)' = f(x)/x. \tag{13}$$

Then, we use vehicle trajectory set to compute steering angle. [18] shows a lateral control method, and our work is inspired by it. As shown in following formulas, we makes some changes based on actual situation. Firstly, we respectively compute average Deflection and average Deviation in formula 14. As shown in formula 15, the steering angle can be computed through 4 parameters. $ADef$ is an average Deflection. $ADev$ is an average Deviation. $Kang$ and $Kpos$ are empirical control values which both could be adjusted according to feedback.

$$ADef = \arctan\left(\frac{(\sum_{i=1}^{m} x_i)/m}{(\sum_{i=1}^{m} y_i)/m}\right) \times 180/\pi, ADev = \frac{\sum_{i=1}^{m} x_i}{m} \tag{14}$$

$$steer = (Kang \times ADef + Kpos \times ADev) \times 2 \tag{15}$$

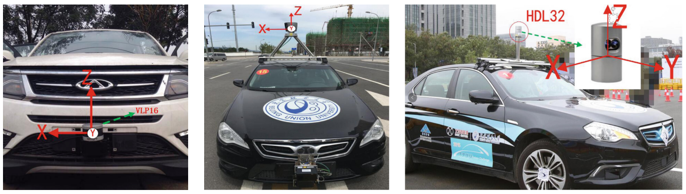
This part involves complex vehicle dynamics and control knowledge. In order to ensure safety of experiment, we add some thresholds on speed and steering to ensure that vehicles would not occur some violent driving behavior.

4 Experiments and Results

We evaluate LNA in different vehicles with different lidars. There are three lidars: VLP-16, HDL-32 and HDL-64. They provide different number of points. In this paper, for the typical feature of HDL-32, it is used to introduce the experiments. Previously obtained steering angle can be sent to control node. The control node directly driving an vehicle. Experiments are conducted in an Ubuntu 14.04 + ROS Indigo + tensorlfow environment. In this section, we show fitting effects of different regression methods. The fitting results directly determine final navigation effects.

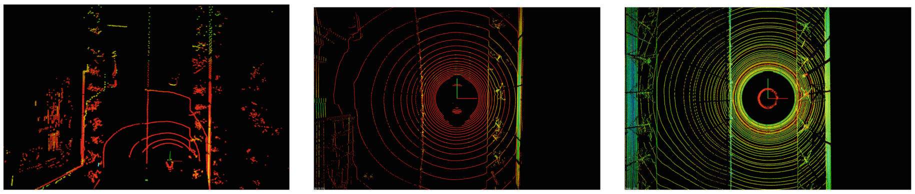
4.1 Data Acquisition and Correction

To get more reliable data, lidar mounting location should be accurated. Lidars placement structures are shown in Fig. 3. As Velodyne 16-line lidar (VLP16) laser wires are few, which result in sparse point cloud. As shown in Fig. 3(a), we choose to mount VLP16 in a front position. It has been proved that this placement can better collect road information. The HDL32 lidar and HDL64 lidar are mounted on top of cars respectively. As shown in Fig. 4, we can see point cloud collecting effects of three lidars. After preprocessing, we use Linear Regression and Support Vector Regression to fit point cloud respectively.



(a) VLP-16 (b) HDL-64 (c) HDL-32

Fig. 3. Lidar placement



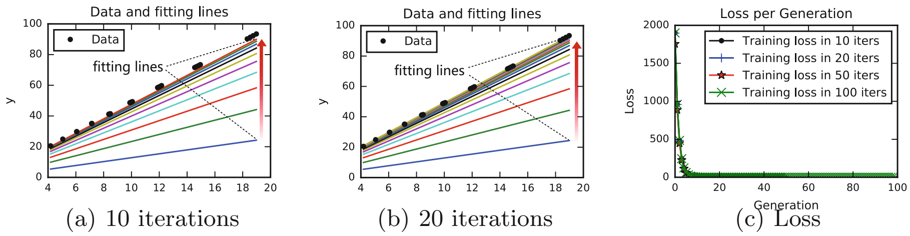
(a) VLP-16 (b) HDL-32 (c) HDL-64

Fig. 4. Point cloud in different lidars

4.2 Linear Regression Results

The parameter w is initialized in a random normal distribution ($mean = -0.2, stddev = 0.01$) and parameter b is initialized as 0. The gradient descent optimizer parameter is set to 0.001.

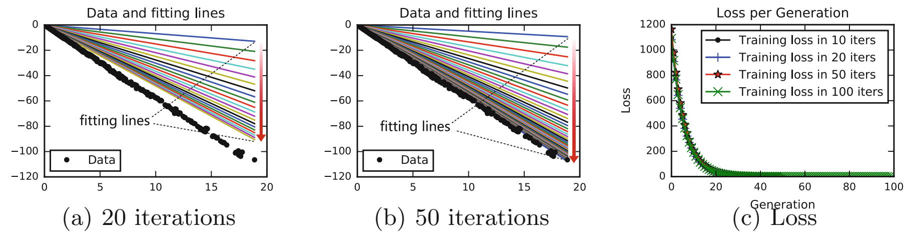
Road Shoulder Fitting Effects. Figure 5 shows different fitting effects in different iteration times. As shown in Fig. 5(b), line fits data well after 20 iterations. Also, it can be demonstrated in Fig. 5(c) that loss is approximate to zero already at 20th iter or even earlier.



(a) 10 iterations (b) 20 iterations (c) Loss

Fig. 5. Road shoulder fitting effects in different training times.

Isolation Belt Fitting Effects. As can be seen from Fig. 6, data is slower to be fitted. After 20 iterations, loss becomes very small. As the number of point cloud is large, which results in a slow convergence. But, it still could reach a good fitting result within 30 iterations. Figure 6(b) shows a fitting effect after 50 iterations. Combining Fig. 6(b) and (c), we can see that, after 30 rather than 50 iterations, model reaches a good fitting effect. All of these features prove our method has better characteristics in the aspects of validity and real-time.



(a) 20 iterations (b) 50 iterations (c) Loss

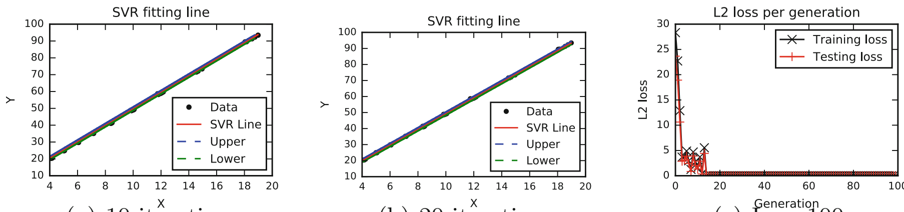
Fig. 6. Isolation belt fitting effects in different training times.

4.3 Support Vector Regression Results

In SVR experiment, we set $\epsilon = 1.0$, $batchsize = 5$ and gradient descent optimizer is set as 0.09. SVR algorithm fitting effects for road shoulders and isolation belt are following.

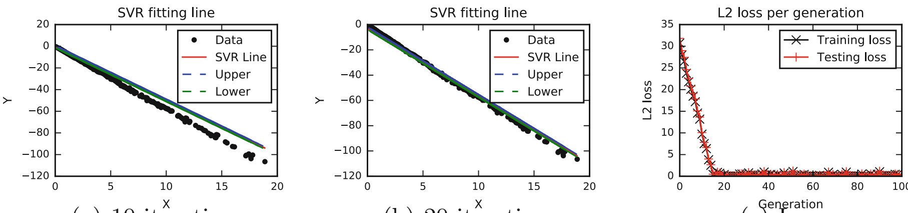
Road Shoulder Fitting Effects. As shown in Fig. 7, we can see that SVR easily fit data within 20 iterations. After 10 iterations, the model fits the data well and the loss value is very small. After 20 iterations, loss nears to 0.

Isolation Belt Fitting Effects. As shown in Fig. 8, we see that SVR is similar to Linear Regression. SVR is under fitting within 10 iterations. SVR fitting line presents a good fitting effect after 20 iterations. After 50 and 100 iterations, effects are further improved. Before that, loss is approximately reach to zero after 20 iterations.



(a) 10 iterations (b) 20 iterations (c) Loss-100

Fig. 7. Support Vector Regression Effects in different training times.



(a) 10 iterations (b) 20 iterations (c) Loss

Fig. 8. Support Vector Regression Effects in different training times.

5 Conclusion

In this paper, we propose an effective navigation approach based on lidar. In order to evaluate the method, we use linear regression and support vector regression to fit point cloud, respectively. The obtained linear fitting models are transformed into vehicle coordinate system. We predict trajectory and compute lateral control data to navigate self-driving cars. LNA can fit well of road outlines within 20 iterations. Loss functions of two linear fitting methods converge to 0 quickly. Each calculation cycle is less than 0.3 seconds. LNA does have advanteges of fast convergence and high approximation accuracy. Experiments of different cars with different lidars in general roads get good navigation effects. This real-time navigation method reaches similar effects of RTK locally. In future work, we will exploit other linear regression methods to further improve processing speed and accuracy.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (61572060, 61772060, 61472024, U1433203), CERNET Innovation Project (NGII 20151004, NGII20160316) and the 2016 Technology Project (No. 3) of Fengtai District, Beijing, China (Grant No. KSF201601).

References

1. Sukkarieh, S., Nebot, E.M., Durrant-Whyte, H.F.: A high integrity imu/gps navigation loop for autonomous land vehicle applications. IEEE Trans. Robot. Autom. **15**(3), 572–578 (1999)
2. Bojarski, M. et al.: End to end learning for self-driving cars (2016)
3. Kenue, S.K.: Lanelok: an algorithm for extending the lane sensing operating range to 100-feet. In: Fibers 1991, MA, Boston, pp. 222–233 (1990)
4. Redmill, K.A., Upadhya, S., Krishnamurthy, A., Ozguner, U.: A lane tracking system for intelligent vehicle applications. In: Intelligent Transportation Systems, Proceedings, pp. 273–279 (2001)
5. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Trans. Robot. **32**(6), 1309–1332 (2016)
6. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 834–849. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10605-2_54
7. Zhang, J., Singh, S.: LOAM: LiDAR odometry and mapping in real-time (2014)
8. Moosmann, F., Stiller, C.: Velodyne SLAM. In: Intelligent Vehicles Symposium, pp. 393–398 (2011)
9. Wijesoma, W.S., Kodagoda, K.R.S., Balasuriya, A.P.: Road-boundary detection and tracking using ladar sensing. IEEE Trans. Robot. Autom. **20**(3), 456–464 (2004)
10. Peterson, K., Ziglar, J., Rybski, P.E.: Fast feature detection and stochastic parameter estimation of road shape using multiple LiDAR. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 612–619 (2008)
11. Yuan, X., Zhao, C.X., Zhang, H.F.: Road detection and corner extraction using high definition LiDAR. Inf. Technol. J. **9**(5), 1022–1030 (2010)
12. Bourbakis, N., Andel, R.: Fusing laser and image data for 3D perceived space representations. In: IEEE International Conference on TOOLS with Artificial Intelligence, Proceedings, pp. 50–58 (1997)
13. Castellanos, J.A., Neira, J., Tardos, J.D.: Multisensor fusion for simultaneous localization and map building. IEEE Trans. Robot. Autom. **17**(6), 908–914 (2001)
14. Hong, T.H., Chang, T., Rasmussen, C., Shneier, M.: Feature detection and tracking for mobile robots using a combination of ladar and color images. In: IEEE International Conference on Robotics and Automation, Proceedings, ICRA 2002, vol. 4, pp. 4340–4345 (2002)
15. Explained, P.M.D.: Introduction to linear regression analysis. Am. Stat. **57**(1), 67–67 (2003)
16. Kutner, M.H., Nachtsheim, C.J., Neter, J.: Applied Linear Regression Models, 5th ed. Technometrics, vol. 26, no. 4 (2004)
17. Awad, M., Khanna, R.: Support vector regression. Neural Inf. Process. Lett. Rev. **11**(10), 203–224 (2007)
18. Thrun, S., et al.: The Robot That Won the DARPA Grand Challenge (2006)