2015 IEEE Conference on Control Applications (CCA)
Part of 2015 IEEE Multi-Conference on Systems and Control
September 21-23, 2015. Sydney, Australia

A Real-time Curb Detection and Tracking Method for UGVs by Using a 3D-LIDAR Sensor

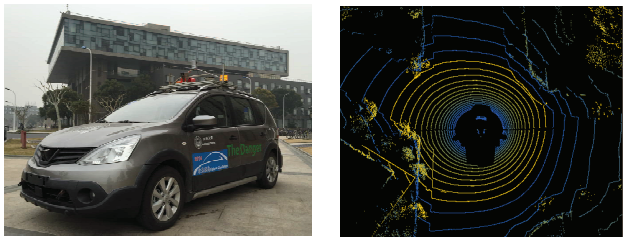
Yihuan Zhang[1], Jun Wang[1*], Xiaonian Wang[2], Chaocheng Li[3] and Liang Wang[4]

Abstract— **Environment perception is essential for autonomous driving technology. The curb is a prominent feature of urban roads and therefore is a significant part of environment perception. In this paper, a real-time curb detection and tracking method is proposed for Unmanned Ground Vehicles (UGVs). The proposed curb detection algorithm uses the the surrounding environment data provided by a 3D-LIDAR sensor to extract the curb position based on its spatial features. The curb tracking algorithm is proposed to predict and update the curb position with respect to the current vehicle states in real time. The performance of the proposed method is verified through extensive experiments with a UGV driving on campus roads. The experimental results demonstrate the accuracy and robustness of the proposed method.**

I. Introduction

Autonomous driving technology is rapidly growing to meet the needs of road safety and efficiency of transportation. Unmanned Ground Vehicles (UGVs) can be used for many applications where it may be inconvenient, dangerous, or even impossible to have a human driver on site. Environment perception is fundamental for vehicle autonomy, and the problem of curb detection and tracking is certainly one of the most important issue of the perception system.

In the past few years, researchers have utilized many different types of sensors to detect curbs of urban roads including video sensors, and laser scanner sensors. In Reference [1], a stereo vision camera with the Canny algorithm based on a multi-frame persistence map was employed to detect curbs. In Reference [2], the curbs were also detected by a stereo vision camera, and a conditional random field algorithm was proposed to propagate the curb detection results. In Reference [3], a millimeter wave radar (MMWR) was employed for road boundary detection. It considered the difference of road and off-road scatter components and derived the fisher information matrix to evaluate the achievable accuracy. In Reference [4], an MMWR and a camera were used to detect the lane and pavement boundaries. In Reference [5], a scanning two-dimensional laser measurement system was used to detect and track the road boundary and an extended Kalman filter was used to enhance its efficiency and accuracy. The drawback of the aforementioned methods is that many false positive detections could be produced due to the limited amount of sensing data. In order to obtain the overall information of the surrounding environment, a high accuracy 3D-LIDAR was used to percept the environment in References [6], [7] and [8]. In Reference [6], the 3D-LIDAR sensor was used to obtain large amount of data of surrounding environment, the result showed that the proposed method is accurate and robust in most scenarios. Nonetheless, the method was only tested on the dataset and the processing period was not mentioned, which is essential for a real-time implement in UGVs.



(a) UGV (b) 3-D point cloud

Fig. 1. Experimental platform

In this paper, a Velodyne HDL-32E 3D-LIDAR sensor was used, which can produce accurate and sparse point cloud. Fig. 1 shows the experimental platform of the UGV and the one-frame raw data generated from the sensor. The method proposed in this paper focuses on the curb detection and tracking in *real* time. The raw data from the sensor is first preprocessed to remove signal noise and to reduce data volume, the positions of curbs are then extracted according to the spatial feature of curbs and fitted by using parabola model, and finally a Kalman filter is used to track and smooth the curb position by using the information of vehicle positions and velocities.

The remainder of the paper is structured as follows. In Section II, the curb detection method is detailed along with the data preprocessing procedures. In Section III, the curb prediction model is built to track the curb position using a Kalman filter. The experiment results are presented in Section IV followed by the conclusion in Section V.

II. Curb detection

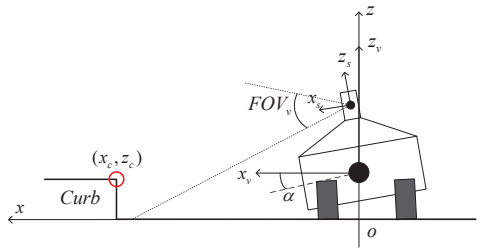
To detect the curb points, the raw data from the sensor is first preprocessed. Based on the spatial feature of curbs, the dynamic sliding window method is then used to extract the curb points. Finally, the curb points are fitted using parabola model to represent the roads.

A. Data preprocessing

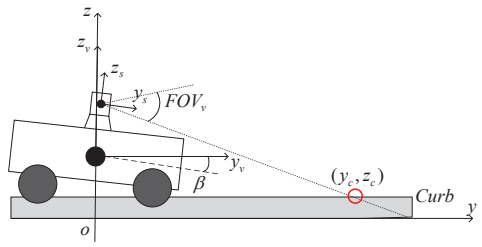
The HDL-32E LIDAR sensor is a small, lightweight, ruggedly built device and features up to 32 lasers across a 40°

Yihuan Zhang, Jun Wang, Xiaonian Wang, Chaocheng Li and Liang Wang are with the Department of Control Science and Engineering, Tongji University, Shanghai 201804, P. R. China.
*Corresponding author junwang@tongji.edu.cn

vertical field of view. The 32 lasers of the sensor are aligned from $+10°$ to $-30°$ to provide an unmatched vertical field of view, and its rotating head delivers a $360°$ horizontal field of view. The HDL-32E generates a point cloud of 700,000 points per second with a range of 70 meters and typical accuracy of 2 cm [9].



(a) Front view



(b) Side view

Fig. 2. Coordinate system and curb description

In this paper, the Velodyne sensor is mounted on the top of the UGV. The raw data is in a standard 3-D polar coordinate $(\rho_i, \theta_i, \gamma_i)$, where ρ_i represents the spatial distance from the sensor to a point, θ_i and γ_i are the horizontal and vertical angle with respect to the sensor coordinate system. The cartesian coordinate system we define is shown in Fig. 2. The raw data in the cartesian coordinate is represented by a set of points $(x_i, y_i, z_i) \in \mathbf{R}$ and can be calculated by the following equations:

$$\begin{cases} x_{i,s} = \rho_i \cos(\gamma_i) \sin(\theta_i) \\ y_{i,s} = \rho_i \cos(\gamma_i) \cos(\theta_i) \\ z_{i,s} = \rho_i \sin(\gamma_i) \end{cases} \tag{1}$$

$$[x_i, y_i, z_i]^T = \boldsymbol{T_r} \cdot [x_{i,s}, y_{i,s}, z_{i,s} - z_0]^T \tag{2}$$

where $x_{i,s}$, $y_{i,s}$ and $z_{i,s}$ represents the point i in the sensor coordinate system (x_s, y_s, z_s).

$$\boldsymbol{T_r} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ -\sin\alpha \cdot \sin\beta & \cos\beta & \cos\alpha \cdot \sin\beta \\ -\sin\alpha \cdot \cos\beta & -\sin\beta & \cos\alpha \cdot \cos\beta \end{bmatrix}$$

and z_0 is the height from the sensor's centre of gravity to the ground, α and β are the roll and pitch angle of the vehicle provided by the inertial measurement unit. Algorithm 1 is proposed to filer the raw data and prepare for the curb detection algorithm.

The results of Algorithm 1 are shown in Fig. 3 and Fig. 4. As we are focusing on the curbs in front of the vehicle, the points which have a negative value in y-axis

Algorithm 1 Raw data filtering algorithm

Require:
The raw data set, $\mathbf{R}$;
The euclidean distance threshold of x-y plane, δ_{xy};
The vertical distance threshold, δ_z;
The quantitative threshold, N;

Ensure:
The filtered data set, $\mathbf{P}$;

Separate the raw data set $\mathbf{R}$ into each laser $\mathbf{R}_k$, where $k = 1, 2, \cdots, 32$ and initialize $k = 0$;
$k = k + 1$;
adjust the δ_{xy}, δ_z, and N according to k and previous parameters;
For each point $\mathbf{R}_{k,j}$ in $\mathbf{R}_k$, calculate the euclidean distance of x-y plane p_{dist} and the vertical distance Δz between two adjacent points;
if $\Delta z \leq \delta_z$ & $p_{dist} \leq \delta_{xy}$ **then**
  if $Jump\_flag$ is set **then**
    Put $\mathbf{R}_{k,j}$ into the list;
    if the length of the list $\geq N$ **then**
      Label the points in the list;
      Reset the $Jump\_flag$;
      Clear the list;
    end if
  else
    Label the recent point $\mathbf{R}_{k,j}$
  end if
else
  Set the $Jump\_flag$;
  Clear the list;
  Put $\mathbf{R}_{k,j}$ into the list;
end if
Put the labeled points into $\mathbf{P}_k$;
While $k \leq 32$, repeat 2 - 21
return $\mathbf{P}$;

are eliminated to reduce the computational expense. After the filter algorithm the relatively smooth data sequences of each laser are obtained, which have more noticeable features between the road surface and the curbs.

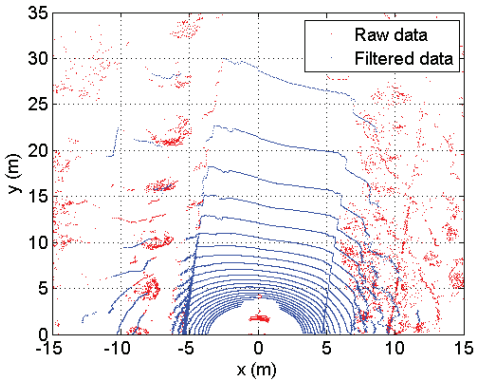
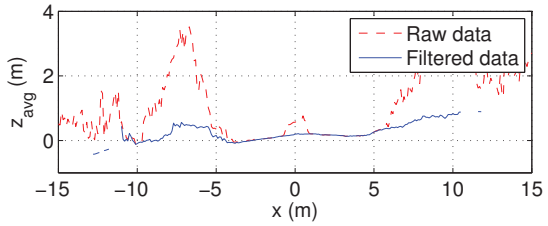
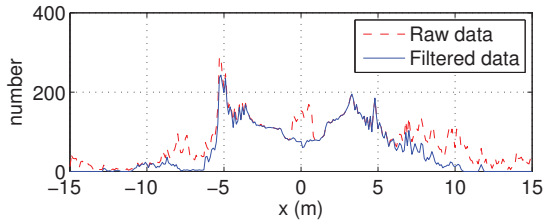


Fig. 3. Distribution of point cloud on the x-y plane;



(a) Average height along the x axis



(b) Number of points along the x axis

Fig. 4. Raw data filtering results

B. Feature extraction

There are many spatial features of curb points that differ from the road surface points. First, the curb height is uniform in most urban areas and it is often 10 to 15 cm higher than the road surface. Second, the elevation changes sharply in the z direction of the cartesian coordinate. Third, the road surface points are smooth and continuous, and the curb always appears at the two sides of the road in most cases. Based on these features, we propose the following algorithm to detect the curb points.

As shown in Algorithm 2, the curbs are represented by two sets of points $\mathbf{Q}_l$ and $\mathbf{Q}_r$. In this algorithm, we use a dynamic sliding window to make the algorithm more time-efficient and accurate. Due to the limited laser number and field of view of the Velodyne sensor, we only concern about the laser plane which is intact to cover the curb area. In order to represent the roads, we fit the curbs using the parabola model. The fitted curb points are represented by $\mathbf{D}_l$ and $\mathbf{D}_r$. The result of the curb detection algorithm is presented in Fig. 5, the sparse curb points set is shown in square and the dense curb points set is shown in green. The root mean square errors (RMSEs) are calculated to evaluate the parabola model for each side of the curb. In Fig. 5(a), the RMSEs are respectively 0.0151 m and 0.0486 m for the left and right side. In Fig. 5(b), the RMSEs are respectively 0.0345 m and 0.0241 m for the left and right side. In real-time experiments, the average RMSE of each side is 0.0285 m.

III. Curb tracking

Once the sets of dense curb point are obtained in one frame, the kalman filtering method is used to track curbs. In Reference [10], the particle filter method is used to track the curbs due to the unreliability of the predicted vehicle motion. We build a curb prediction model and propose a tracking algorithm based on the accurate vehicle state data from the GPS/IMU system.

Algorithm 2 Curb detection algorithm

Require:
The filtered data set, $\mathbf{P}$;
The euclidean distance threshold of x-y plane, ξ_{xy};
The vertical distance threshold, ξ_z;
The width of a sliding window, w_d;
The offset of a sliding window, w_o;

Ensure:
The sparse sets of curb-points, $\mathbf{Q}_l$, $\mathbf{Q}_r$;

Separate the filtered data set $\mathbf{P}$ into left and right part by each laser $\mathbf{P}_{l,k}$ and $\mathbf{P}_{r,k}$, where $k = 1, 2, \cdots, 32$;
for $k = 1$ to 32 **do**
  Adjust the ξ_{xy}, ξ_z, w_d and w_o according to k and previous parameters;
  for each i in $\mathbf{P}_{l,k}$ **do**
    Put from $\mathbf{P}_{l,k,i}$ to $\mathbf{P}_{l,k,i+w_d}$ into sliding window W_i;
    if $(\max_z(W_{i,z}) - \min_z(W_{i,z})) \geq \xi_z$ **then**
      for j in W_i **do**
        Calculate the p_{dist};
        if $p_{dist} \geq \xi_{xy}$ **then**
          Label $\mathbf{P}_{l,k,i}$ and jump to 2;
        else
          Calculate the Δz between the adjacent points and label $\mathbf{P}_{l,k,i}$ with the maximum Δz and jump to 2;
        end if
      end for
    else
      $i = i + w_o$ and jump to 5;
    end if
  end for
  Put the labeled points into $\mathbf{Q}_l$;
  Do the same process to find the curb points on the right side $\mathbf{Q}_r$;
end for
return $\mathbf{Q}_l$ and $\mathbf{Q}_r$;

A. Curb prediction model

There are three coordinate frames of interest as shown in Fig. 6. A fixed ground coordinate frame (x, y) is set as a reference. Two coordinate frames $(x_{v,k}, y_{v,k})$ and $(x_{v,k+1}, y_{v,k+1})$ represent the vehicle frame at the time instances k and $k+1$ respectively for derivation of the curb tracking algorithm. The curb prediction model is defined as follows:

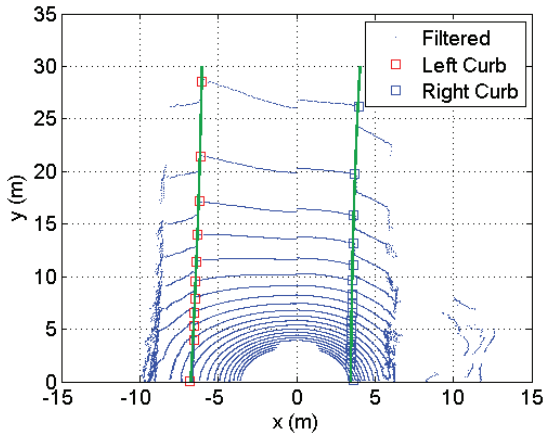
$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k) \tag{3}$$

where

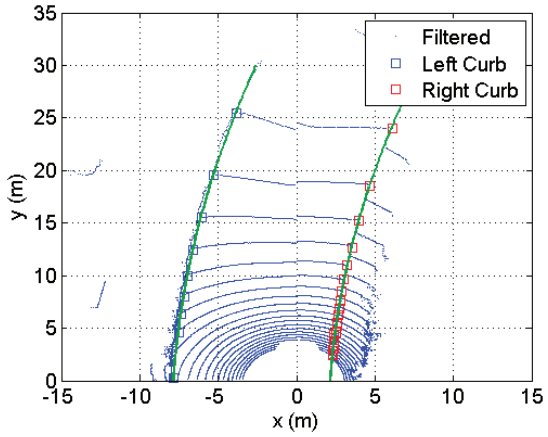
$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ y(k) \end{bmatrix}, \ \mathbf{u}(k) = \begin{bmatrix} \Delta x_v(k) \\ \Delta y_v(k) \end{bmatrix}$$

$$\mathbf{A}(k) = \begin{bmatrix} \cos\psi(k) & \sin\psi(k) \\ -\sin\psi(k) & \cos\psi(k) \end{bmatrix}, \ \mathbf{B}(k) = \begin{bmatrix} -\cos\psi(k) & -\sin\psi(k) \\ \sin\psi(k) & -\cos\psi(k) \end{bmatrix}$$

and $x(k)$ and $y(k)$ are in the set of dense curb points $\mathbf{D}_l$ and $\mathbf{D}_r$, $\Delta x_v(k)$ and $\Delta y_v(k)$ are the moving distance between



(a) straight road case



(b) curve road case

Fig. 5. Fitting curb point sets and sparse curb point sets on the x-y plane

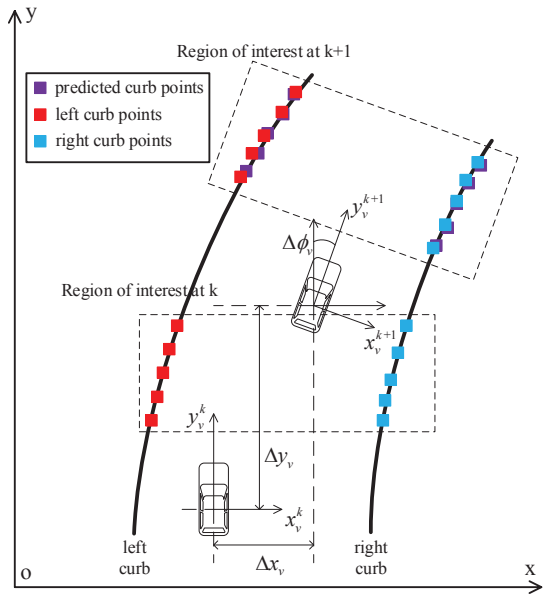


Fig. 6. Coordinate frames of curb tracking algorithm

two frames of x-axis and y-axis, $\psi(k) = \Delta\phi_v(k)$ is rotation of the vehicle. Then the predicted curb position can be obtained by Equation (3), which relies on the precision of the GPS/IMU system.

B. Tracking algorithm

Based on the prediction model mentioned above, a Kalman filter is used to track the curb position to improve the smoothness of the curbs. The tracking algorithm is carried out by the following steps:

1) Curb measurement model: As we have detected the dense set of the curb points, we assume that the measurement noise is white gaussian noise. Hence, the measurement model is represented by

$$\mathbf{z}(k) = \mathbf{x}(k) + \boldsymbol{\nu}(k) \tag{4}$$

where $\mathbf{z}(k) = [z_x(k), z_y(k)]^T$ represents the measurement vector and $\boldsymbol{\nu}(k)$ is the measurement noise with a covariance matrix $\mathbf{C}_r$.

2) Predict state and update error covariance matrix:

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k) + \boldsymbol{\omega}(k) \tag{5}$$

$$\hat{\mathbf{P}}(k+1) = \mathbf{A}(k)\mathbf{P}(k)\mathbf{A}^T(k) + \mathbf{C}_q \tag{6}$$

In Equations (5) and (6), $\hat{\mathbf{x}}(k+1)$ is the prediction of the next state vector and $\mathbf{x}(k)$ is the estimation of the current state vector. $\hat{\mathbf{P}}(k+1)$ is the error covariance prediction of the next state and $\mathbf{P}(k)$ is the error covariance estimation of the current state. $\boldsymbol{\omega}(k)$ is the process noise which we assume to be white gaussian noise with covariance matrix $\mathbf{C}_q$.

3) Calculate the Kalman filter gain:

$$\mathbf{G}_K(k+1) = \hat{\mathbf{P}}(k+1) \cdot \left[\hat{\mathbf{P}}(k+1) + \mathbf{C}_r\right]^{-1} \tag{7}$$

4) Update the state and the error covariance matrix: The estimation of the system state vector and error covariance matrix are represented by

$$\mathbf{x}(k+1) = \hat{\mathbf{x}}(k+1) + \mathbf{G}_K(k+1)(\mathbf{z}(k+1) - \hat{\mathbf{x}}(k+1)) \tag{8}$$

$$\mathbf{P}(k+1) = \hat{\mathbf{P}}(k+1) - \mathbf{G}_K(k+1)\hat{\mathbf{P}}(k+1) \tag{9}$$

where $\mathbf{x}(k+1)$ is the final output of the tracking algorithm which represents one curb point, after all the points among $\mathbf{D}_l$, $\mathbf{D}_r$ are tracked using Equations (4)-(9), the curbs of two sides are obtained.

IV. Experiment Results

To evaluate the proposed method, extensive experiments are carried out both online and offline. Furthermore, we compare the proposed method with two state-of-the-art methods, i.e. the Haar wavelet transformation [11] and the Hough transformation [12]. Both methods have their own advantages and disadvantages which as illustrated in the following experiments.

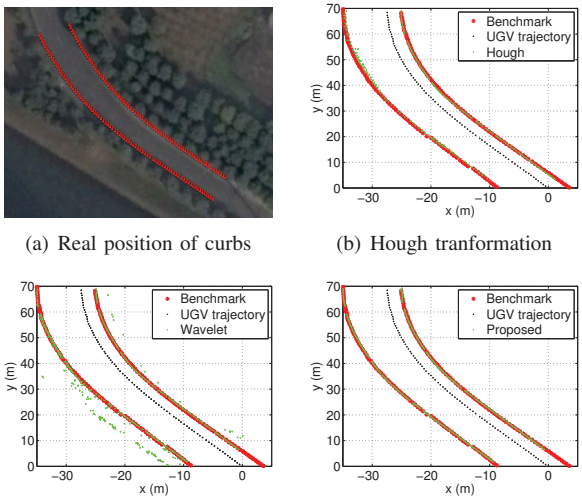
A. Dataset experiment

In order to compare with the two different methods, a dataset recorded by our UGV is used to test these methods in MATLAB. The dataset consists of the vehicle speed, heading orientation, pitch and roll angle, along with the raw data from Velodyne sensor of each frame. The three methods are evaluated based on a sequence of 100 frames in the dataset. Parameters of the curb detection and tracking algorithm are defined in Table I.

TABLE I
ESSENTIAL PARAMETER INITIALIZATION

Parameters	Initialization	Tolerance(±)
δ_{xy}	0.3 m	0.1 m
δ_z	0.08 m	0.03 m
N	20	10
ξ_{xy}	0.2 m	0.08 m
ξ_z	0.05 m	0.02 m
w_d	30	10
w_o	10	5

As shown in Fig. 7(a), the curb points in Google Earth are labeled in red and the curbs are modified using the method of Open Street Map (OSM) program in Reference [13]. In Reference [14], the bertha benz self-driving car uses the OSM data as a static curb reference to plan the global trajectory. The dataset experiment results of three different detection methods are shown in Fig. 7. The result in Fig. 7(b) shows that the Hough transformation method has difficulty in handling high curvature roads, the curb points scatters at the curve part of the road. In Fig. 7(c), there are many false positive detections. Although the method can provide relatively accurate curb points, it can not be used in our UGV without noise filtering. As shown in Fig. 7(d), with the pre-processing of the raw data and a dynamic threshold adjustment, the sparse curb points are relatively accurate. In addition, a parabola model is used to fit the sparse curb points to obtain the dense curb points which is shown in green.



(a) Real position of curbs (b) Hough tranformation

(c) Haar wavelet tranformation (d) Proposed method

Fig. 7. Dataset experiment results

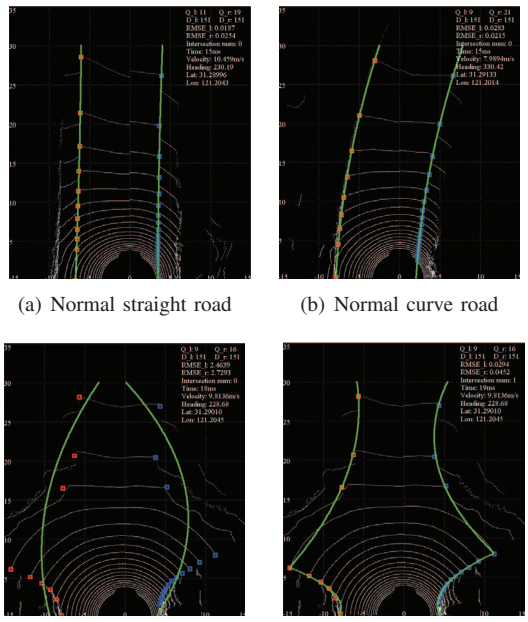
The performance indexes are shown in Table II, ε is set to 0.1m. Though the short average period of Haar wavelet transformation method, its precision rate is lower than the proposed method. Hough transformation method has a relatively high precision but it is too time-consuming for real-time implementation. The proposed method has the highest precision rate and acceptable processing period among the three methods in the dataset experiment.

TABLE II
PERFORMANCE INDEX OF THREE METHODS

Methods	Average period	Precision rate (ε)	Precision rate (3ε)
Proposed	25.3 ms	96.88%	100.00%
Hough	73.2 ms	83.22%	98.76%
Wavelet	16.4 ms	78.91%	86.79%

B. Real-time experiment

After the comparison of the three methods, the proposed algorithm is tested on the UGV platform which uses a PCI extensions for Instrumentation (PXI) as the core controller. The NI PXI-8109 is a high-performance intel core i7-620M processor-based embedded controller for PXI systems. With the 2.66 GHz base frequency, PXI is able to process the Velodyne sensor data in real time. The proposed method is implemented in Visual Studio based on Windows-7 Operating System. The real-time operating window is shown in Fig. 8.



(a) Normal straight road (b) Normal curve road

(c) Intersection (d) Intersection (fixed)

Fig. 8. Real-time operation results

Most of the experimental roads are straight or curve as shown in Fig. 8(a), 8(b), the average period is less than 20 ms which is qualified in the whole architecture of the UGV. UGV states are also shown in the operating window such as speed, latitude, longitude and heading orientation. Besides, in order to modify some important parameters, the number of

sparse curb point sets $\mathbf{Q}_l$ and $\mathbf{Q}_r$ and dense curb point sets $\mathbf{D}_l$ and $\mathbf{D}_r$ are monitored in real time. The results show that the method is accurate for most normal roads. However, the performance at intersection is unacceptable because there are no curbs. In Fig. 8(c), there is a intersection about 10 m in front of the vehicle so there are no curb points from 8 m to 15 m. If we continue to use the proposed parabola model to fit the curbs, the result is shown in green line which is obviously incorrect. In order to prevent the wrong curb positions, the fitting algorithm is modified into a segmented parabola model based on detection of the intersection. In Fig. 8(d), the number of intersection detected in one frame are labeled and two parabola model are used to fit the curb points of each side. The modified method can effectively reduce the error of curb positions.

The real-time experiment is carried out on our campus. As shown in Fig. 9, left and right curb points are marked in blue and red in a local map consists of 5363 frames and a road of about 5 km. The average speed is about 35 km/h. Curbs are matched on the digital map to evaluate the method proposed. The real-time performance indexes are shown in Table III. The rough part of the curb position is an intersection and a segmented parabola model is used to fit the curbs. The results show that the curb detection and tracking method is time-efficient and accurate for most normal roads. As for the intersections or other scenarios without curbs, the proposed method shows a relatively poor performance and should be modified to percept the road without curbs.

TABLE III

PERFORMANCE INDEXES IN REAL-TIME EXPERIMENTS

Performance Indexes	Value	Units
Average Period	16.32	ms
Average Velocity	33.56	km/h
Average RMSE (left)	0.0431	m
Average RMSE (right)	0.0376	m
Intersection Detection	2034	–

Fig. 9. Campus road testing results

V. CONCLUSION AND FUTURE WORK

This paper develops a real-time curb detection and tracking method for UGVs by using a Velodyne 3D-LIDAR sensor. It employs several prominent features of road surface and curbs to filter the raw data which contains lots of useless information in curb detection. Then a dynamic sliding window method is used to detect the curb position and the curb points are fitted based on parabola model. Also, a Kalman filter is used to track these curbs considering the vehicle motion. Experiments on dataset demonstrate the accuracy and robustness of the proposed method. The real-time experiment shows the proposed method is time-efficient and accurate for curb detection and tracking.

It is still a challenging problem to detect an intersection without curbs. The detection method should be further modified to enhance its accuracy and robustness. Furthermore, a high-accuracy digital map should be properly incorporated if possible.

ACKNOWLEDGMENT

This work would not have been possible without the dedicated efforts of the Tongji UGV team in Department of Control Science and Engineering. This work was also supported in part by the National Science Foundation of China under Grant No. 61473209.

REFERENCES

[1] F. Oniga, S. Nedevschi, and M. M. Meinecke, "Curb detection based on a multi-frame persistence map for urban driving scenarios," in *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems*, 2008, pp. 67–72.

[2] J. Siegemund, D. Pfeiffer, U. Franke, and W. Forstner, "Curb reconstruction using conditional random fields," in *Proceedings of the 2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 203–210.

[3] M. Nikolova and A. Hero, "Segmentation of a road from a vehicle-mounted radar and accuracy of the estimation," in *Proceedings of the 2000 IEEE Intelligent Vehicles Symposium*, 2000, pp. 284–289.

[4] B. Ma, S. Lakshmanan, and A. O. Hero, "Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 3, pp. 135–147, 2000.

[5] W. S. Wijesoma, K. S. Kodagoda, and A. P. Balasuriya, "Road-boundary detection and tracking using Ladar sensing," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 456–464, 2004.

[6] G. Zhao and J. Yuan, "Curb detection and tracking using 3D-LIDAR scanner," in *Proceedings of the 19th IEEE International Conference on Image Processing*, 2012, pp. 437–440.

[7] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[8] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little ben: the ben franklin racing team's entry in the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

[9] "Velodyne HDL 32-E LIDAR," http://www.velodynelidar.com/lidar/hdlproducts/hdl32e.aspx.

[10] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, 2008.

[11] K. Peterson, J. Ziglar, and P. E. Rybski, "Fast feature detection and stochastic parameter estimation of road shape using multiple LIDAR," in *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 612–619.

[12] Y. Kang, C. Roh, S. Suh, and B. Song, "A lidar-based decision-making method for road boundary detection using multiple Kalman filters," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4360–4368, 2012.

[13] F. Ramm, J. Topf, and S. Chilton, *OpenStreetMap: using and enhancing the free map of the world*. UIT Cambridge, 2011.

[14] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. Keller, *et al.*, "Making bertha drive? an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.