Intersection detection and recognition for autonomous urban driving using a virtual cylindrical scanner

Qingquan Li¹, Long Chen², Quanwen Zhu³, Ming Li⁴, Qun Zhang⁵, Shuzhi Sam Ge⁵
¹Shenzhen Key Laboratory of Spatial Smart Sensing and Services, Shenzhen University, Shenzhen, Guangdong, People’s Republic of China
²School of Mobile Information Engineering, Sun Yat-Sen University, Zhuhai, Guangdong, People’s Republic of China
³School of Geodesy and Geomatics, Wuhan University, Wuhan, Hubei, People’s Republic of China
⁴Computer School, Wuhan University, Hubei, People’s Republic of China
⁵Electrical and Computer Engineering Department, National University of Singapore, Singapore
E-mail: lchen.whu@gmail.com

Abstract: In this study, the authors propose an effective real-time approach for intersection detection and recognition during autonomous driving in an unknown urban environment. The authors approach use point cloud data acquired by a three-dimensional laser scanner mounted on the vehicle. Intersection detection and recognition are formulated as a classification problem whereby roads are classified as segments or intersections and intersections are subclassified as T-shaped or + -shaped. They first construct a novel model called a virtual cylindrical scanner for efficient feature-level representation of the point cloud data. Then they use support vector machine classifiers to resolve the classification problem according to the features extracted. A series of experiments on real-world data sets and in a simulation environment demonstrate the effectiveness and robustness of the authors approach, even in highly dynamic urban environment. They also performed simulation experiments to investigate effects of several critical factors on their proposed approach, such as other vehicles on the road and the advance detection distance.

1 Introduction

Autonomous driving is the most fundamental requirement for unmanned ground vehicles (UGVs) in intelligent transportation systems and has attracted much research interest in the past two decades. First, an UGV must have environmental perception capability. When the vehicle is autonomously driving on the road, it needs to know the location of drivable regions and the basic road geometry and topological relations. When there is an intersection ahead, the vehicle needs to detect and locate it in advance for safety reasons and for path planning. Road and lane detection has been intensively investigated. A comprehensive literature review summarises the most popular lane detection techniques used up to 2006 [1]. Several solutions have been proposed in recent years [2, 3]. We previously developed a system for our autonomous SmartV-II vehicle that detects drivable regions and lanes via sensor fusion [4]. However, few real-time and reliable solutions have been proposed for detection of intersections during autonomous driving within the past two decades of research. The difficulties are as follows:

- It is hard to accurately represent all intersections with one versatile model because of their diverse forms and geometries (T-shapes and + -shapes, large and small, regular and irregular etc.).
- Relatively dense traffic with many vehicles and pedestrians around an intersection can cause significant interference with detection.
- Non-negligible uncertainty arises because most intersections are too large for complete coverage by a single onboard sensor, especially when attempting to detect an intersection in advance.
- The possibility that a variety of road markings might be broken, blocked or omitted makes intersection detection more challenging.

For these reasons, intersection detection is a challenging problem in UGV perception systems.

To date, most of the methods commonly used for autonomous navigation rely heavily on an accurate global positioning system (GPS) or inertial navigation system (INS) integrated with a precise digital map [5–8]. Sotelo et al. [6] used a differential GPS (DGPS) to acquire data on vehicle location and applied a vision-based method for navigation through intersections. Cue and Ge [7] combined the interacting multiple model (IMM) method with an extended Kalman filter (EKF) to estimate the position of a vehicle in intersection areas. GPS traces have also been
used to detect intersections [8]. GPS-based methods are effective in simplifying many types of perception problems. However, a major issue is that GPS is not always accurate enough because of signal blockage or multi-path effects in urban environments where tall buildings abound. In addition, digital maps for navigation are sometimes missing or not up-to-date, especially in rapidly developing countries. Autonomous driving systems that rely heavily on accurate GPS and GIS systems fail without reliable pertinent information, which represents a significant limitation. Therefore approaches for intersection detection based purely on onboard sensors are very desirable.

Vision-based methods have long been proposed in the literature. These approaches can be categorised as methods based either on road area [9–13], for unstructured roads or on lane markings [14] for structured roads. However, poor lighting conditions (overcast conditions or overexposure), weather conditions (rainy and murky) and the presence of moving vehicles and pedestrians make vision-based methods extremely difficult and unreliable even if sophisticated image-processing techniques are used at the expense of processing speed. In addition, approaches mainly based on lane markings and road shoulders are not applicable when lanes and shoulders are broken or blocked by other obstacles. A new approach addresses road shape analysis from a vision-based point of view [15]; this is excellent, but it cannot be used in autonomous vehicles because of its offline algorithm.

The application of active sensors is increasingly widespread because of their independence from illumination. Kodagoda et al. [16] used laser depth measurements for speedy extraction of two corresponding road edges or curbs and then an EKF for detection of Y-shaped, T-shaped and X-shaped intersections. Wijesoma et al. [17] combined a two-dimensional (2D) laser scanner and charge-coupled device (CCD) cameras for tracking of road curbs. However, the single-scan laser used could not guarantee a stable algorithm. Larsson et al. [18] proposed a feature detection algorithm for recognising tunnel intersections. However, intersections in tunnels are quite different from intersections on the ground, since roads on the ground potentially have various traffic participants, whereas a tunnel environment is relatively simple. The multiple light detection and ranging (LIDAR) cooperation method [19, 20] is an effective solution that compensates for the problem of a limited field of view. However, these studies focused on detecting moving objects and estimating the degree of safety with the presumption that intersections are known a priori and can be obtained from the road network. Aycard et al. [21] used multiple sensors such as LIDAR and stereo vision, but focused on safety issues and risk assessment rather than intersection detection.

Considering the limited view and the complexity of an intersection, we used a high-end three-dimensional (3D) laser scanner (Velodyne HDL-64ES2 [22]) as a sensor for perceiving intersections because of its omnidirectional representation ability for large-scale scenes. The Velodyne sensor has been used in UGVs over the past few years. Many excellent unmanned autonomous vehicles have been equipped with this sensor, including, MuCAR-3 [23], AnnieWAY [24] and Stanley [25]. The Velodyne HDL-64ES2 model is a dense 64-beam scanning LIDAR that provides 360° coverage at 10 Hz, generating over 1 million 3D points and the corresponding infrared reflectance values per second. To process such rich raw data in real-time, in addition to grid maps [26], we propose a novel model termed virtual cylindrical scanner (VCS) to represent and downsample the raw data. The concept of VCS is similar to the virtual scan used by Petrovskaya and Thrun [27]. However, there are several crucial differences: (i) the virtual scan is a planar structure, whereas VCS is a 3D structure; (ii) the number of rays in the virtual scan is fixed, but varies in VCS; and (iii) the virtual scan is mainly used for vehicle detection, but VCS focuses on intersection recognition. Chen et al. [28] proposed a similar solution to ours and used a Velodyne laser to detect intersections. The difference is that we model intersection detection as a classification problem and our algorithm can deal with intersections where other vehicles are present on the road, whereas their algorithm is quite vulnerable to interference from traffic participants. Furthermore, the advanced detection distance in our algorithm can change adaptively based on the vehicle speed. This is an in-depth study of our previous investigation [29], with more detailed illustrations of the algorithm, more substantial experimental validation and a thorough investigation of the influence of several critical factors. The key aim is to find intersections in advance and identify the type as T-shaped or +-shaped. We model the detection as a classification problem and use four features extracted via VCS for binary classification. The optimal feature is selected empirically and then combined with a support vector machine (SVM) to predict the type of current scene. Specifically, the contributions of this paper can be summarised as follows:

- We propose a reliable scheme for locating and classifying intersections using Velodyne data without GPS, GIS or INS. In our approach, intersection detection is modelled as a classification problem.
- We propose a novel 3D virtual sensor named VCS to represent and sample raw data. Four features are extracted from the represented data and are used for intersection detection.
- We propose an effective placement strategy for the VCS launch point. A vehicle detection method based on 3D point data is also presented for preprocessing.
- The detection distance of our method is not fixed and adapts to the speed of the autonomous vehicle, which guarantees that the vehicle has adequate time to prepare for navigation through an intersection.

The remainder of the paper is organised as follows. Section 2 introduces the VCS concept, including its definition and implementation. Section 3 describes the VCS-based intersection detection and recognition algorithm in detail. Experimental results for real data sets and vehicle tests are presented in Section 4, followed by an investigation of some critical factors in simulation experiments. Conclusions are given in Section 5.

2 Virtual cylindrical scanner

The Velodyne HDL-64E sensor produces 1 million points per second. Processing of such rich data in real-time is a challenge. Consequently, we propose a new technique for downsampling raw data and generating a new representation of the road environment.

Definition 1 (VCS): A VCS is a virtual 3D scanner consisting of $n$ rays $R_i$ emitted from the same point, called the launch point ($P_l$). The VCS elements are described as follows:

1. $R_i$ (ray): a 3D line with a specific direction. This is the basic detection unit of the VCS that is cut off by any ‘obstacle’ it meets.
2. $P_L$ (launch point): the centre of the VCS, from which all rays are emitted.

As shown in Fig. 1a, the launch point is also the origin of the VCS coordinate system. The $y_v$-axis coincides with the direction of the initial ray $R_1$ (ray number 1) and the $x_v$-axis is perpendicular to the $y_v$-axis. All rays are launched around the $z_v$-axis at an equal-angle interval $\theta$. The direction of the $i$th ray can be determined by the clockwise angle interval between the $i$th ray and initial ray: $d_i = (i-1)\theta$. The detectable area is a cylinder, denoted by the dashed circles of radius $D$ centred at $P_L$ with height $h$. Three ray types are defined according to their 3D structure: a cuboid ray, a cylinder ray and a fan ray (Fig. 1b).

- ‘Cuboid ray (Cu-ray)’: the ray is a cuboid with corresponding dimensions and can be represented by
  
  $$ R_i = (l_i, w_i, h_i, d_i) $$
  
  where $l_i$, $w_i$ and $h_i$ are the length, width and height, respectively, of the cuboid generated by the $i$th ray.

- ‘Cylinder ray (Cy-ray)’: the ray is a cylinder that can be represented by
  
  $$ R_i = (l_i, r_i, d_i) $$
  
  where $l_i$ is the length and $r_i$ is the radius of the cylinder.

- ‘Fan ray (F-ray)’: the ray is a quadrant with characteristic length and angle. It can be represented by
  
  $$ R_i = (l_i, \delta_i, h_i, \theta_i) $$
  
  where $l_i$ is the length, $r_i$ is the radius, $\delta_i$ is the angle and $h_i$ is the height of the fan.

Therefore $VCS$ can be represented as a simple vector:

$$ VCS = [P_L, R_1, \ldots, R_n]^T \in \mathbb{R}^n $$

where $R_i(1, 2, \ldots, n)$ is the $i$th ray and $P_L(x_L, y_L, z_L)$ are the coordinates of the launch point in the sensor frame. $VCS$ can cope with different types of sensor data, such as images and point clouds. According to requirements, the launch point is placed in a reasonable position for the application and then rays are launched from the launch point along their characteristic direction until they detect an ‘obstacle’. An ‘obstacle’ is defined as follows.

**Definition 2 (obstacle):** Data with a special attribute $O_a$ that $VCS$ cannot pass through.

Thus, ‘obstacle’ as defined here is not limited to physical obstacles in the real world and can refer to any object with the specified attribute. For instance, ‘obstacle’ could be pixels of images whose grey values or gradients exceed a
specific threshold, or some specific clusters or objects. For point clouds, ‘obstacle’ can also be points or cells in a grid map whose elevation values exceed a specific threshold and some other specific objects such as trees, pedestrians, vehicles or buildings.

We choose a 3D structure for the ray instead of a line because a ray has two main advantages. One is its strong anti-interference properties, especially for point clouds. A line could easily pass through an ‘obstacle’ because of the sparsity of the data. However, a 3D structure greatly mitigates this type of error. The other advantage is its wider detection range. A 3D structure can cope with sensing of floating or partly floating objects such as bridges, tree canopies and traffic signs, which are unlikely to be detected by a single line.

After launching all the rays, the current environment can be represented by some characteristics of the **VCS**. The basic representation vector

\[
E_i = \begin{bmatrix} l_{R_1}, \ldots, l_{R_j}, \ldots, l_{R_n} \end{bmatrix}^T
\]

is given by the distance measured for an ‘obstacle’ by each ray of the **VCS**; \(l_{R_j}\) is the distance measured by the \(j\)th ray, that is, the measurement of the \(j\)th ray to the closest obstacle. In addition, the attributes of the detected ‘obstacle’ can also be considered in representing the current environment, as follows

\[
E_a = \begin{bmatrix} a_{R_1}, \ldots, a_{R_n} \end{bmatrix}^T
\]

where \(E_a\) is the environment vector represented by the attributes of the ‘obstacles’ and \(a_{R_j}\) is the attribute of the ‘obstacle’ detected by the \(j\)th ray. In this implementation, we choose a **VCS** with a ‘Cu-ray’, which performs best empirically on a grid map constructed according to the Velodyne data. An ‘obstacle’ is defined as all cells for which the elevation variance of the constituent points exceeds a threshold in the grid map after removing detected vehicles.

### 3 VCS-based intersection detection and recognition

Our algorithm works on a grid map constructed from Velodyne laser data. Considering the real-time requirement, a 2D grid map is chosen in the proposed application because of the lower computational complexity than for a 3D grid map. Accordingly, we simplify VCS from the 3D form to a 2D form in our experiment. First, other vehicles in the scene are detected and removed via a simple vehicle detection method. Then VCS is launched in an appropriate place as determined by the speed of the vehicle and its position on the road and the road width. Four features from the VCS are used to classify the road scene. The feature with the best performance is chosen as the final feature for real road testing.

In Fig. 2, the sensor highlighted by the red rectangle is the Velodyne HDL-64ES2 model used in this study. The sensor is mounted on top of the vehicle, parallel to the ground. The coordinate system for each frame is based on the coordinate system for the vehicle body: the centre of the Velodyne sensor is the origin in the coordinate system; the vehicle direction corresponds to the \(x\)-axis, the \(z\)-axis points upwards and the \(y\)-axis is perpendicular to the \(x-y\)-plane according to the right-hand rule.

#### 3.1 Grid map construction and vehicle detection

The basic feature vector we use is the normalised distance measured for an ‘obstacle’ by each ray of the VCS. To represent the road structure accurately, the extracted feature is considered perfect when the ray is blocked by road edges instead of other objects, and its effectiveness decreases with increasing deviation of distance between the ray’s end point and the expected terminal point on the road edge. Consequently, traffic participants, especially those emerging in the centre of the road instead of close to the edges, undoubtedly impair the effectiveness of our algorithm and should be detected and removed beforehand. Therefore we apply a method for detecting vehicles according to the geometric cluster characteristics from the grid map.

The basic steps are as follows:

1. Construct a 2D occupancy grid with 400 × 400 cells, each covering a small patch of ground of 50 cm × 50 cm. Each cell stores a single floating point value expressing the elevation information of the cell, which can be used to evaluate whether the cell is an obstacle or free. The centre of the grid map is the position of the Velodyne laser.
2. Construct an obstacle grid map based on the elevation variance \(\text{Var}[C_u(v, v)]\) for all points falling in the corresponding cell, which can be calculated as

\[
\text{Var}(C_u(v, v)) = \frac{1}{n} \sum_{i=1}^{n} (e_i - \mu(u, v))^2
\]

where \(\text{Var}[C_u(v, v)]\) is the elevation variance for cell \(C_u(v, v)\) with coordinates \((u, v)\) in the grid map and \(e_i\) is the elevation of the \(i\)th point in cell \(C_u(v, v)\), \(\mu(u, v)\) is the mean elevation for the cell points \(C_u(v, v)\) and can be obtained as

\[
\mu(u, v) = \frac{1}{n} \sum_{i=1}^{n} e_i
\]

where \(n\) is the number of points that fall in the cell. If \(\text{Var}[C_u(v, v)] > \text{th}\), the corresponding cell is considered as an obstacle cell; otherwise it is considered as a free cell. Considering most of the vehicles and pedestrians on the road are over 1 m, \(\text{th}\) is chosen as 1 m for this study. Fig. 2c shows the obstacle grid map constructed from the original data shown in Fig. 2b.
3. Cluster these obstacle cells within a four-connected region.
4. All clusters whose geometric properties obey the following rules are classified as vehicles: the width of the boundary box is \(\text{width}_{bb}\) ∈ [1 m, 3 m], the length of the boundary box is \(\text{length}_{bb}\) ∈ [1 μ, 4 μ] and the length/width ratio is \(\text{width}_{bb}/\text{length}_{bb}\) ∈ [0.3, 1].
5. Modify the obstacle map by changing all the labels for cells that belong to detected vehicles from an obstacle to free. The final obstacle grid map (Fig. 2e) is used for subsequent feature extraction.

Theoretically, only failure to detect vehicles that are relatively close to the road centre will have a significant influence on our algorithm. A thorough investigation of the impact of interference by other vehicles was conducted via simulation experiments, as described in Section 4.
Fig. 2  SmartV-II and step-by-step results for preprocessing

a  SmartV-II, the intelligent vehicle developed by Wuhan University. The rectangular box shows the Velodyne scanner
b  Original data
c  Grid map
d  Vehicle detection
e  Grid map after vehicle removal. Small squares indicate obstacle cells and rectangles denote the boundary boxes for detected vehicles
3.2 Calculation of the launch point position

In addition to the presence of other vehicles, the position of the launch point is a key factor that affects VCS effectiveness. There are two desirable requirements for the position of the launch point: (i) it should be on the middle line of a road segment; and (ii) the x-axis of the VCS should coincide with the direction of the road instead of the direction of movement of the autonomous vehicle, namely, the initial ray should be perpendicular to the right-hand boundary of the road segment. The key point to fulfil these requirements is to accurately estimate the middle line of the road segment. We assume that the autonomous vehicle is always in the road segment since we want to detect the intersections in advance. Then we can calculate the best position for the launch point \( P_L(x_L, y_L) \) and the angular deviation \( \theta \) between the vehicle direction and the road according to Algorithm 1 (see Fig. 3).

To reserve enough time for appropriate maneuvers before entering the intersection, the distance \( d \) between the centre of the vehicle and launch point is speed-dependent here and is calculated as

\[
d = d_b + v_0 t + d'
\]  

where \( d_b \) is the braking distance for the vehicle from \( v_0 \) to 0 and \( v_0 \) is the initial velocity when the vehicle begins to brake. According to the law of conservation of energy, to reduce the kinetic energy to zero, we obtain

\[
d_b = \frac{v_0^2}{2 \mu g}
\]

where \( g \) is the acceleration because of gravity and \( \mu \) is the effective coefficient for friction between the tires and road. Thus, \( d_b \) is determined by \( \mu \), which is related to the surface quality of the tire and road conditions, (e.g. wet, icy, slippery and sandy). The term \( v_0 t' \) is the reaction distance, which is the distance travelled by the vehicle during reaction time \( t' \). For an autonomous vehicle, \( t' \) is the maximum computation time. In addition, we add a fixed value \( d' \) to guarantee the shortest detection distance.

3.3 VCS-based feature extraction

The basic feature we consider is the normalised length for all VCS rays, that is, normalised \( E_i \) in (5)

\[
L = \left[ I_{b_1}, \ldots, I_{b_n} \right]^T, \quad n = \frac{2\pi}{\theta}
\]

where \( I_{b_i}/D \) denotes the normalised length of the \( i \)th ray. \( D \) is the original length of the ray when it is not blocked by any 'obstacle'. Here we use \( \theta = \pi/180 \) and \( \pi = 360 \). As shown in Fig. 4, the distribution of the normalised lengths is different for T-shaped and +/shaped intersections and road segments. Intuitively, the number of continuous peaks corresponds to the number of exits in the current road scene. For a segment, there are only two, whereas for T-shaped and +/shaped intersection there are three and four exits, respectively. Moreover, the positions of the continuous peaks are also distinct.

Derived from \( L \), the gradient feature \( G_\psi \) for direction \( \psi \) can be defined as

\[
G_\psi = \begin{cases} g_{\psi,1}, \ldots, g_{\psi,n}, & n = \frac{2\pi}{\theta} \end{cases}
\]

where

\[
g_{\psi,i} = \begin{cases} I_{R_i} - I_{R_{i+\psi/(\theta/2)}}, & \text{if } i + \frac{\psi}{\theta} < n \\ I_{R_i} - I_{R_{i+\psi/(\theta/2)-n}}, & \text{else} \end{cases}
\]

In an urban environment, most intersections exhibit right-angle characteristics, so \( G_{90}, G_{180} \) and the serial fused alignment \( G_{90} = \{g_{90,1}, \ldots, g_{90,n}, g_{180,1}, \ldots, g_{180,n}\} \) is chosen for the environment features. Classification over these feature descriptor vectors is then formulated as a supervised machine learning problem [30]. Here we use an SVM-based supervised learning mechanism to verify the effectiveness of these four feature types. The most effective features are then chosen as the final selection.

4 Experimental results

In this section, we describe the data sets used in our experiments, apply the proposed algorithm to these data sets, estimate the main factors that influence the algorithm, and provide an in-depth analysis of the results.

4.1 Experiments with real data sets

To test the performance of the features \( L, G_{90}, G_{180}, G_{90} \) and \( G_{180} \), we used two sets of data. Data set 1 contains 219 frames for T-shaped intersections, 49 frames for +/shaped intersections and 1161 frames for road segments. Data set 2 contains 336 frames for T-shaped intersections, 161 frames for +/shaped intersections and 1038 frames for road segments. We used data set 1 for training and data set 2 for testing. All of the data were captured using the intelligent SmartV-II vehicle developed by the Wuhan University (Wuhan, China).

For classification of intersections and road segments, all 268 frames for intersections in the training data were labelled as positive samples and the 1161 frames for road segments were labelled as negative samples. The test data
consist of 497 frames for intersections and 1038 frames for road segments.

The balanced-ridge design in an SVM for adjusting weights between majority and minority classes was set to [1, 4]. In addition, a radial basis function was applied with the form $K(u, v) = \exp(-\gamma \|u - v\|^2)$, where the kernel parameter $\gamma$ and the soft margin $C$ are selected between $[2^{-3}, 2^{-2}, 2^{-1}, 1, 2, 2^2, 2^3]$ and $[10^{-2}, 10^{-1}, 1, 10, 10^2]$ respectively, using 3-fold cross-validation on the training set. The advance detection distance $d$ was calculated according to (9) and the vehicle velocity was randomly selected from 0 to 20 km/h.

Let TP, TN, FP, FN be the number of true positive, true negative, false positive and false negative examples, respectively. Hence, the true positive rate (TPR), defined as $TP/(TP + FN)$, is an important criterion in unbalanced problems, because we are more concerned about the minority class, that is, intersections. The true negative rate (TNR) is defined as $TN/(TN + FP)$ and the accuracy (ACC) is defined as $(TP + TN)/(TP + FP + TN + FN)$. The unweighted ACC (UACC), defined as $(TPR + TNR)/2$, is also used as an indicator of classifier performance.

Results for the classification performance for the four feature types are listed in Table 1. Except for $G_{180}$, TPR, TNR, ACC and UACC results for the other three features are all >96.7%. The $G_{90}$ and $G_{180}$ feature outperforms all the others. It is evident that TPR is greater than TNR in all cases, which means the classifier is more sensitive to intersections and road segments tend to be classified as intersections.

After detecting an intersection, we categorise it as T-shaped or +-shaped. The training data for this task were from the main training data set: 219 frames for T-shaped intersections were labelled as positive samples and 49 frames for +-shaped intersections were labelled as negative samples. There were 336 positive and 161 negative samples in the test data. The classification performance is reported in Table 2. Compared with the performance for classification of intersections and road segments, the results for classification of T-shaped and +-shaped intersections are much worst. The best performance was observed for $L$, with UACC = 91.26% and the worst for $G_{180}$ with TPR <50%. In addition, $G_{90}$ performed better than $G_{180}$ and $G_{90}$ and $G_{180}$.

![Fig. 4](https://example.com/fig4.png) *VCS works with Velodyne data and the corresponding normalised length for all rays*

Examples of (top to bottom) a road segment, a T-shaped intersection and a +-shaped intersection. Green lines represent the launched VCS.

![Table 1](https://example.com/table1.png)

<table>
<thead>
<tr>
<th>$z$</th>
<th>TPR, %</th>
<th>TNR, %</th>
<th>ACC, %</th>
<th>UACC, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>98.39</td>
<td>96.92</td>
<td>97.39</td>
<td>97.66</td>
</tr>
<tr>
<td>$G_{90}$</td>
<td>97.79</td>
<td>96.72</td>
<td>97.07</td>
<td>97.43</td>
</tr>
<tr>
<td>$G_{180}$</td>
<td>80.48</td>
<td>95.76</td>
<td>90.81</td>
<td>88.12</td>
</tr>
<tr>
<td>$G_{90}$ and $G_{180}$</td>
<td>98.60</td>
<td>97.78</td>
<td>98.04</td>
<td>98.19</td>
</tr>
</tbody>
</table>

![Table 2](https://example.com/table2.png)

<table>
<thead>
<tr>
<th>$z$</th>
<th>TPR, %</th>
<th>TNR, %</th>
<th>ACC, %</th>
<th>UACC, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>94.41</td>
<td>88.10</td>
<td>90.14</td>
<td>91.26</td>
</tr>
<tr>
<td>$G_{90}$</td>
<td>78.89</td>
<td>97.91</td>
<td>91.75</td>
<td>88.40</td>
</tr>
<tr>
<td>$G_{180}$</td>
<td>47.21</td>
<td>99.40</td>
<td>82.495</td>
<td>73.30</td>
</tr>
<tr>
<td>$G_{90}$ and $G_{180}$</td>
<td>65.84</td>
<td>98.81</td>
<td>88.13</td>
<td>82.33</td>
</tr>
</tbody>
</table>

![Fig. 4](https://example.com/fig4.png) *VCS works with Velodyne data and the corresponding normalised length for all rays*
These experiments revealed that $G_{90}$ and $G_{180}$ yields the best results for classification of road segments and intersections, whereas $L$ has better performance for classification of T-shaped and $\perp$-shaped intersections. Therefore we used $G_{90}$ and $G_{180}$ and $L$ for intersection detection and classification in the following tests.

Fig. 5 shows some examples of successful classification for road segments and intersections. The top right segment is on a bridge and the top right one is on a wider road similar to a highway. The second $\perp$-shaped intersections are on a campus road and the right intersection is larger than the left one. The T-shaped intersections in the third row are on urban roads. The $\perp$-shaped intersection in the last row is a 20-m-wide two-way pavement with six lanes. The right road segment is even wider than the left one. According to our experiments, the maximum road width for which our algorithm is effective is approximately 25 m owing to the large coverage area of the Velodyne LIDAR. These examples demonstrate that our algorithm is effective for various road environments, including urban and campus roads and highways, even the large-scale roads.

Fig. 6 Results for the detection of T-shaped intersection, $\perp$-shaped intersection and road segment

$\perp$-shaped intersection data

Road segment data
algorithm operates in real-time at approximately 5 Hz on a PC with Intel Core2 Duo central processing unit at 2.3G.

4.2 Effect of the detection distance

Detection distance, defined as the distance from the VCS launch point to the vehicle centre, is one of the main factors affecting our algorithm. To evaluate its effect, we chose some relatively clean scenarios without interference from other vehicles as the test set. This set included 120 T-shaped intersections, 31 + shaped intersections and 437 road segments. The VCS launch point for different detection distances in one scenario were manually input in the middle of the road segments or in the centre of intersections.

Here we use TPR and FPR to evaluate the results of the intersection detection algorithm. TPR means the true positive rate after intersection/road segment classification and T-shaped/+shaped intersection classification. Similarly, FPR means the false positive rate after the two rounds of classification. Fig. 6 shows the classification results for the three types of road scenes. Fig. 6a shows the results for T-shaped intersection recognition. TPR decreases with increasing distance, with results of 91.67, 87.5, 80.83, 79.17 and 71.67% corresponding to 10, 20, 30, 40 and 50 m. FPR rapidly increases with distances greater than 40 m. For + shaped intersections, similar TPR results are evident in Fig. 6b. TPR results corresponding to the five test distances are 100, 96.78, 93.55, 83.87 and 77.42%, respectively. However, in this case the increase in FPR is not as fast, which means that T-shaped intersections and road segments are less likely to be misclassified as a + shaped intersection. For road segments, the TPR results are 99.1, 98.6, 97.3 and 90.6% for 10, 20, 30 and 40 m, respectively. These results are all >90%. However, TPR decreases to 49% for the 50 m distance. The main reason is the characteristics of the sensor, that is, the laser point is sparser at greater distance. It is much easier for the VCS rays to penetrate through cells originally belonging to the road boundary. The FPR results in Fig. 6c reveal that an intersection can hardly be misclassified as a road segment.

This set of experiments reveals that our algorithm can accurately distinguish an intersection from a road section, but the ACC for road segment recognition and subclassification of T-shaped and + shaped intersections rapidly decreases for detection distances greater than 40 m. Thus, the effective distance of our proposed algorithm is no more than 40 m.

4.3 Effect of vehicles

Detection of other vehicles is another major factor that affects our method. The effect of vehicles refers to vehicles that are not detected and removed from the road scene. To explore this effect, we designed an experiment using simulation data for all the scenes we need. The width of the road was set to 12 m and the size of each vehicle was set to 5 m × 2 m. The vehicle distribution differs for road segments, T-shaped intersections and + shaped intersections, as shown in Fig. 7, where grey rectangular boxes indicate possible vehicle positions. There are 30, 42 and 54 possible such positions for a road segment, a T-shaped intersection and a + shaped intersection, respectively. The red boxes indicate road curbs and ‘obstacles’. We simulated five situations corresponding from one to five vehicles.

Classification results for all five situations are listed in Table 3, where ‘total’ is the total number of possible cases for each situation. Results under ‘S’, ‘T’ and ‘+’ are the ACC of recognition of segments, T-shaped intersections and + shaped intersection, respectively. It is evident that the more vehicles there are, the greater is the misclassification. For road segments, when there are five vehicles on the road, 19.8% of the segments are misclassified as T-shaped intersections. For T-shaped intersections, the ACC decreases from 97.6 to 82.1% as the number of vehicles increase, but remains satisfactory. Many T-shaped intersections are classified as road segments, whereas a small number are misclassified as + shaped intersections. The worst results are for + shaped intersections. As the
number of vehicles increases, the ACC for +\-shaped intersection recognition decreases from 88.9 to 48.7\%.

For five vehicles, TN is even greater than TP. It is clear that the presence of other vehicles affects our algorithm, especially for +\-shaped intersections. However, even under these difficult conditions, the algorithm works well for classification of road segments and intersections. The worst ACC for segment detection is >80\%. For T-shaped and +\-shaped intersections, the worst ACC for intersection detection is 82.6 and 98.7\%, respectively. These results illustrate the robustness of the algorithm, even when seriously affected by other vehicles.

### 4.4 Misclassification and analysis

Despite desirable results for discrimination between intersections and road segments using our approach, some misclassification is evident. The main errors arise from two issues: misdetection of vehicles and dead zones blocked by other obstacles, such as vehicles. Fig. 8a shows misclassification of road segment as an intersection that can be attributed to these issues. First, the bus was not detected as a vehicle, then a few VCS rays passed through the area behind the bus, which should be the edge of the road.

### Table 3

<table>
<thead>
<tr>
<th>Number</th>
<th>Segment (30)</th>
<th>T-shaped intersection (42)</th>
<th>+-shaped intersection (54)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total S, %</td>
<td>Total S, %</td>
<td>Total S, %</td>
</tr>
<tr>
<td>1</td>
<td>30 100</td>
<td>42 2.4</td>
<td>54 0</td>
</tr>
<tr>
<td>2</td>
<td>435 97.7</td>
<td>851 7.0</td>
<td>1431 0</td>
</tr>
<tr>
<td>3</td>
<td>4060 93.4</td>
<td>11 480 11.2</td>
<td>24 804 0.3</td>
</tr>
<tr>
<td>4</td>
<td>27 405</td>
<td>111 930 14.7</td>
<td>316 251 0.7</td>
</tr>
<tr>
<td>5</td>
<td>142 506</td>
<td>850 668 17.4</td>
<td>3 162 510 1.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>T, %</th>
<th>+, %</th>
<th>T, %</th>
<th>+, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>6.6</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>12.7</td>
<td>0.3</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>5</td>
<td>19.8</td>
<td>0.5</td>
<td>1.3</td>
<td>1.3</td>
</tr>
</tbody>
</table>

\*Fig. 8 Examples of misclassification of a road segment as an intersection because of*

\*a\* An undetected vehicle

\*b\* A dead zone blocked by other objects

\*c\* A road segment with an intersection-like shape. Left: image of the actual scene. Middle: birds-eye view of the Velodyne data. Right: birds-eye view of the Velodyne data after VCS-based intersection recognition
The results in Table 2 show that it is more difficult to classify T-shaped and +shaped intersections. The main reasons are the more complex environment with respect to road segments and their greater similarity. An example of misclassification of a T-shaped intersection as a +shaped intersection is shown in Fig. 8b. The reason is similar to that of the above. According to the different vehicle positions relative to the Velodyne centre, the point cloud data for vehicles exhibit different styles, so a fixed vehicle model for all vehicles represents a limitation. The method of Petrovskaya and Thrun [27] may be a more feasible scheme for special cases. Dealing with dead zones blocked by other objects is another issue to be resolved to improve the algorithm performance.

Another type of misclassification is because of road irregularity. Fig. 8c shows an example of this misclassification in which the road segment contains a concave section. The VCS-based features are similar to a T-shaped intersection.

5 Conclusions

We proposed an algorithm for real-time detection and recognition of intersections during autonomous driving using a VCS virtual sensor. We modelled an intersection detection and location as a road classification problem and applied a new virtual scanner to represent and downsample mass point cloud data. In contrast to the previous work, our approach is purely based on perception, with no GPS or GIS information.

In addition, the detection distance is not fixed, but relies on the vehicle speed. The SVM classifier gave a success rate of >95% for classification of intersections and road segments according to four different features and a lower rate of >90% success for more complex T-shaped and +shaped intersections. Results show that the approach is reliable, efficient and capable of handling different types of roads and traffic situations during real vehicle tests. Overall, our algorithm yields good performance for most intersections, even when other vehicles on the road are not removed correctly. There are some feasible solutions for the problems observed, such as occlusion reasoning and filling of dead zones via registration of multiple scans. In addition, even a low-precision GPS could greatly improve the ACC of the algorithm.

Our algorithm could be improved in two ways in future work. First, obstacle detection could be based on a dynamic model, especially for detection of moving vehicles. Second, registration of several laser scans could be applied to deal with dead zones, in which the laser is blocked by other objects. In addition to improving the ACC of the intersection recognition, we plan to obtain more data on intersection attributes such as geometry and width.

6 Acknowledgment

The work described in this paper was supported by the National Natural Science Foundation of China (grant nos. 91120002 and 41001306).

7 References