PAPER

Real-time stereo vision-based lane detection system

IOP Publishing Measurement Science and Technology
Meas. Sci. Technol. **29** (2018) 074005 (13pp) https://doi.org/10.1088/1361-6501/aac163

Real-time stereo vision-based lane detection system

Rui Fan[1,3] **and Naim Dahnoun**[2]

[1] Visual Information Laboratory, University of Bristol, Bristol, BS8 1UB, United Kingdom
[2] Department of Electrical and Electronic Engineering, Merchant Venturers Building, University of Bristol, Bristol, BS8 1UB, United Kingdom

E-mail: ranger.fan@bristol.ac.uk

Received 23 February 2018, revised 16 April 2018
Accepted for publication 30 April 2018
Published 24 May 2018

Abstract

The detection of multiple curved lane markings on a non-flat road surface is still a challenging task for vehicular systems. To make an improvement, depth information can be used to enhance the robustness of the lane detection systems. In this paper, a proposed lane detection system is developed from our previous work where the estimation of the dense vanishing point is further improved using the disparity information. However, the outliers in the least squares fitting severely affect the accuracy when estimating the vanishing point. Therefore, in this paper we use random sample consensus to update the parameters of the road model iteratively until the percentage of the inliers exceeds our pre-set threshold. This significantly helps the system to overcome some suddenly changing conditions. Furthermore, we propose a novel lane position validation approach which computes the energy of each possible solution and selects all satisfying lane positions for visualisation. The proposed system is implemented on a heterogeneous system which consists of an Intel Core i7-4720HQ CPU and an NVIDIA GTX 970M GPU. A processing speed of 143 fps has been achieved, which is over 38 times faster than our previous work. Moreover, in order to evaluate the detection precision, we tested 2495 frames including 5361 lanes. It is shown that the overall successful detection rate is increased from 98.7% to 99.5%.

Keywords: depth information, lane detection, vanishing point, GPU, real-time

(Some figures may appear in colour only in the online journal)

1. Introduction

Various prototype vehicle road tests have been conducted by Google in the US since 2012, and its subsidiary *X* is planning to commercialise their autonomous cars as from 2020 [1]. Recently, Volvo has conducted a series of self-driving experiments involving about 100 cars in China and many companies like Ford and Uber have entered the race to make driverless taxis a reality [2]. Techniques such as lane detection in advanced driver assistance systems (ADAS) are playing an increasingly crucial role in enhancing driving safety and minimising the possibilities of fatalities.

The state-of-the-art lane detection algorithms can be grouped into two main categories: feature-based and model-based [3]. The feature-based algorithms extract the local, meaningful and detectable parts of an image, such as edges, texture and colour, to segment lanes and road boundaries from the background [4]. On the other hand, the model-based algorithms try to represent the lanes with a mathematical equation based upon some common road geometry assumptions [5]. The most commonly used lane models include linear, parabolic, linear-parabolic and spline. The linear model works well for the lanes with a low curvature, as demonstrated in [2, 6]. However, a more flexible road model is inevitable when the lanes with a higher curvature exist. Therefore, some algorithms [7–10] use a parabolic model to represent the lanes with a constant curvature. For some more complex cases, Jung *et al* proposed a linear-parabolic combined lane model, where the nearby lanes are represented as linear models, whereas the far ones are modelled as parabolas [11]. In addition to

[3] Author to whom any correspondence should be addressed.

the models mentioned above, the spline model is an alternative way to interpolate the lane pixels into an arbitrary shape [5, 12]. However, the more parameters introduced into a flexible model, the higher will be the computational complexity of the algorithm. Therefore, we turn our focus on some additional important properties of 3D imaging techniques instead of being limited to only 2D information.

One of the most prevalently used methods is inverse perspective mapping (IPM). With the assumption that two lanes are parallel to each other in the world coordinate system (WCS), IPM is able to map a 3D scenery into a 2D bird's eye view [13]. Furthermore, many researchers [14–18] proposed to use the vanishing point $\boldsymbol{p}_{\mathbf{vp}} = [u_{\mathrm{vp}}, v_{\mathrm{vp}}]^{\top}$ to model lane markings and road boundaries, where u_{vp} and v_{vp} represent the vertical and horizontal coordinates of the vanishing point, respectively. However, their algorithms work well only if the road surface is assumed to be flat or the camera parameters are known. Therefore, we pay closer attention to the disparity information which can be provided by either active sensors, e.g. radar and laser, or passive sensors, e.g. stereo cameras [17]. Since Labayrade *et al* proposed the concept of '*v*-disparity' [19], disparity information has been widely used to enhance the robustness of the lane detection systems. Our previous work [17] shows a particular instance where the disparity information is successfully combined with a lane detection algorithm to estimate $\boldsymbol{p}_{\mathbf{vp}}$ for a non-flat road surface. At the same time, the obstacles contain a lot of redundant information which can be eliminated by comparing the actual and fitted disparity values. However, the estimation of $\boldsymbol{p}_{\mathbf{vp}}$ suffers from the outliers when performing the least squares fitting (LSF), and the lanes are sometimes unsuccessfully detected because the selection of plus-minus peaks is not always effective. Moreover, achieving real-time performance is still a challenging task in [17] because of the intensive computational complexity of the algorithm. Therefore, in this paper we present an improved lane detection system for the problems mentioned above.

The proposed multiple lane detection system is composed of four main components: disparity map estimation, dense v_{vp} estimation, dense u_{vp} estimation and lane position validation. The block diagram of the proposed system is illustrated in figure 1, where procedures 1 to 5 are processed on a GPU (graphics processing unit) because they are more efficient for parallel processing but the serially-efficient procedures 6 to 12 are executed on a CPU (central processing unit).

Firstly, a disparity map is estimated by comparing the difference between a pair of well-rectified left and right images. The disparity map is mainly used for

- estimation of dense v_{vp},
- road surface estimation, and
- elimination of the redundant information.

In this paper, the road surface is not assumed to be flat and the projection of the road disparities on the *v*-disparity map is modelled as a parabola. Compared with some quadratic pattern detectors, dynamic programming (DP) is a more efficient way to extract the path with the highest accumulations from the *v*-disparity map. The extracted path is then interpolated into a parabola using the LSF. However, the outliers in the LSF severely affect the accuracy of the vanishing point estimation. Therefore, we propose to update the parabola function iteratively using the random sample consensus (RANSAC) until the percentage of the inliers exceeds our pre-set threshold. This greatly improves the robustness of the proposed system. Since the bilateral filter performs better than the median filter in terms of edge preservation and noise elimination, it is utilised to reduce the unnecessary edges before estimating u_{vp}. v_{vp} and the orientation of each edge point in the road surface area are then used to estimate u_{vp}, where we employ the RANSAC to minimise the influence of outliers on the LSF. An arbitrary lane marking or road boundary can thus be extracted using the vanishing point information. Finally, we propose a novel lane position validation approach which computes the energy of each possible solution and selects all satisfying lanes for visualisation.

The remainder of this paper is structured as follows: section 2 describes the proposed lane detection system. Section 3 evaluates the experimental results. Section 4 summarises the paper and provides some recommendations for future work.

2. System description

2.1. Disparity map estimation

As compared with many other stereo matching algorithms which aim at automotive applications, the trade-off between accuracy and speed has been greatly improved in our previous work [20]. Therefore, we employ the algorithm presented in [20] to acquire the disparity information for the proposed lane detection system.

2.1.1. Memorisation. Due to the insensitivity to the intensity difference, the normalised cross-correlation (NCC) is utilised as the cost function to measure the similarity between two blocks, as shown in equation (1). Each block chosen from the left image is matched with a series of blocks on the same epipolar line in the right image. The block pair with the highest correlation cost is then selected as the best correspondence, and the shifting distance between them is defined as the disparity d:

$$c(u,v,d) = \frac{1}{n\sigma_l\sigma_r}\sum_{x=u-\rho}^{x=u+\rho}\sum_{y=v-\rho}^{y=v+\rho}(i_l(x,y)-\mu_l)(i_r(x-d,y)-\mu_r) \tag{1}$$

where c is defined as the correlation cost, and i_l and i_r represent the pixel intensities in the left and right images, respectively. The centres of the left and right blocks are (u,v) and $(u-d,v)$, respectively. The side length of the block is $2\rho+1$. $n=(2\rho+1)^2$ represents the number of pixels within each block. μ_l and μ_r denote the means of the intensities within

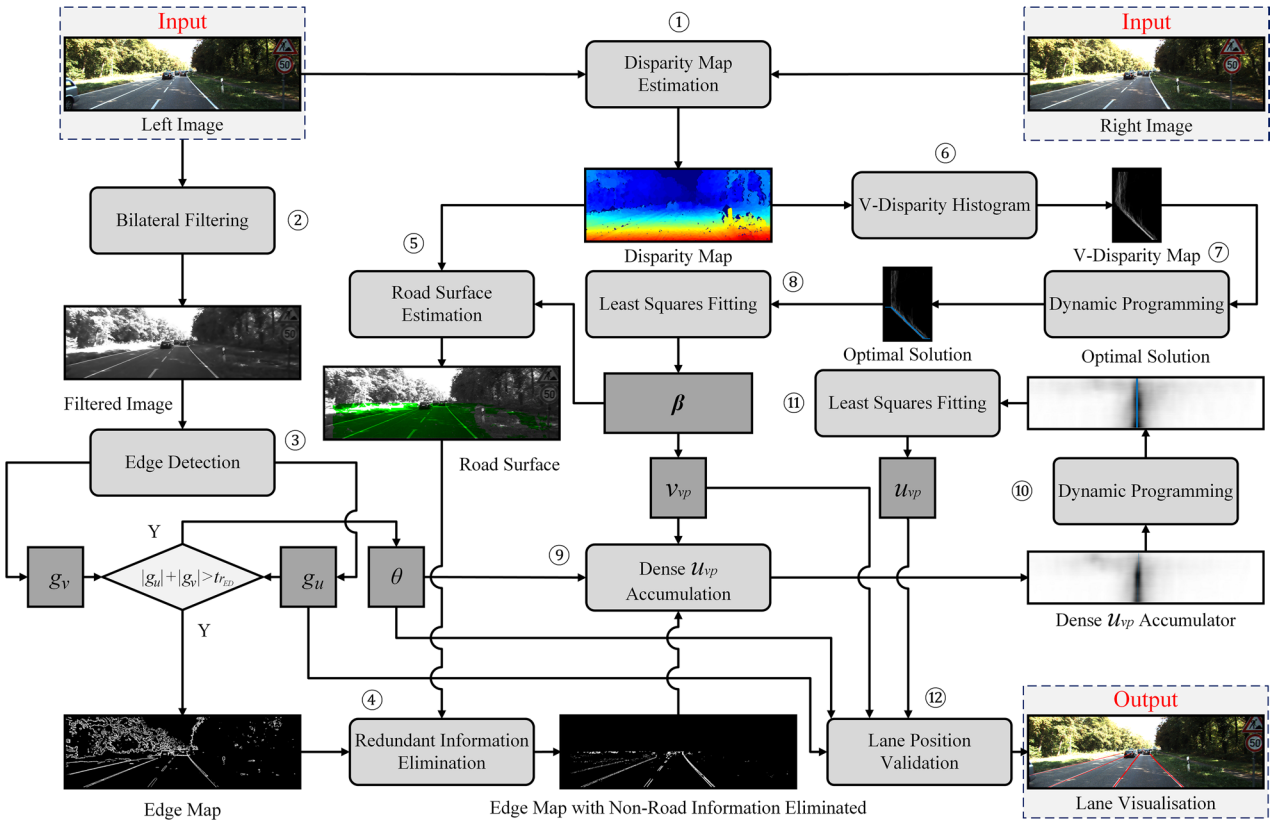

Figure 1. The block diagram of the proposed lane detection system.

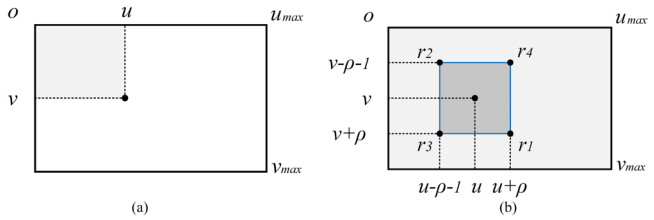

Figure 2. Integral image processing. (a) Original image. (b) integral image.

the left and right blocks, respectively. σ_l and σ_r represent their corresponding standard deviations [20]:

$$\sigma_l = \sqrt{\sum_{x=u-\rho}^{x=u+\rho}\sum_{y=v-\rho}^{y=v+\rho}(i_l(x,y)-\mu_l)^2/n} \tag{2}$$

$$\sigma_r = \sqrt{\sum_{x=u-\rho}^{x=u+\rho}\sum_{y=v-\rho}^{y=v+\rho}(i_r(x-d,y)-\mu_r)^2/n}. \tag{3}$$

When the left block is selected, the computations of μ_l and σ_l are always repeated because d is only used to select the positions of the right blocks for stereo matching. Therefore, the four independent parts μ_l, μ_r, σ_l and σ_r can be pre-calculated and stored in a static program storage for direct indexing. The integral image algorithm is used to compute μ_l and μ_r efficiently [21], which is illustrated in figure 2. The algorithm has two steps: integral image initialisation and value indexing from the initialised reference. In the first step, for a discrete image i whose pixel intensity at (u, v) is $i(u, v)$, its integral image intensity $I(u, v)$ at the position of (u, v) is defined as

$$I(u,v) = \sum_{x\leqslant u,\ y\leqslant v} i(x,y). \tag{4}$$

Algorithm 1 details the implementation of the integral image initialisation, where I is calculated serially based on its previous neighbouring results to minimise unnecessary computations.

Algorithm 1. Integral image initialisation.

Input : original image: i
Output: integral image: I
$I(u_{\min}, v_{\min}) \leftarrow i(u_{\min}, v_{\min})$;
for $u \leftarrow u_{\min} + 1$ to $u_{\max}$ **do**
  $I(u, v_{\min}) \leftarrow I(u-1, v_{\min}) + i(u, v_{\min})$;
end
for $v \leftarrow v_{\min} + 1$ to $v_{\max}$ **do**
  $I(u_{\min}, v) \leftarrow I(u_{\min}, v-1) + i(u_{\min}, v)$;
end
for $u \leftarrow u_{\min} + 1$ to $u_{\max}$ **do**
  for $v \leftarrow v_{\min} + 1$ to $v_{\max}$ **do**
    $I(u, v) \leftarrow I(u, v-1) + I(u-1, v)$
    $-I(u-1, v-1) + i(u, v)$;
  end
end

After initialising an integral image, the sum $s(u, v)$ of pixel intensities within a square block whose side length is

$2\rho+1$ and centre is (u,v) can be computed using four references $r_1 = I(u+\rho, v+\rho)$, $r_2 = I(u-\rho-1, v-\rho-1)$, $r_3 = I(u-\rho-1, v+\rho)$ and $r_4 = I(u+\rho, v-\rho-1)$ as follows:

$$s(u,v) = r_1 + r_2 - r_3 - r_4. \tag{5}$$

The mean $\mu(u,v) = s(u,v)/n$ of the intensities within the selected block is then stored in a static program storage for the computations of σ and c. To simplify the computations of σ_l and σ_r, we rearrange equations (2) and (3) as shown in equations (6) and (7), respectively:

$$\sigma_l = \sqrt{\sum_{x=u-\rho}^{x=u+\rho}\sum_{y=v-\rho}^{y=v+\rho} {i_l}^2(x,y)/n - {\mu_l}^2} \tag{6}$$

$$\sigma_r = \sqrt{\sum_{x=u-\rho}^{x=u+\rho}\sum_{y=v-\rho}^{y=v+\rho} {i_r}^2(x-d,y)/n - {\mu_r}^2} \tag{7}$$

where $\sum {i_l}^2$ and $\sum {i_r}^2$ are dot products. Similarly, the computations of $\sum {i_l}^2$ and $\sum {i_r}^2$ can be accelerated by initialising two integral images I_{l^2} and I_{r^2} as references for indexing. Therefore, the standard deviations σ_l and σ_r can also be calculated and stored in a static program storage for the efficient computation of c as follows:

$$c(u,v,d) = \frac{1}{n\sigma_l\sigma_r}\left[\sum_{x=u-\rho}^{x=u+\rho}\sum_{y=v-\rho}^{y=v+\rho} i_l(x,y)i_r(x-d,y) - n\mu_l\mu_r\right]. \tag{8}$$

According to equation (8), only $\sum i_l i_r$ needs to be calculated during the stereo matching. Hence, with the values of μ_l, μ_r, σ_l and σ_r able to be indexed directly, equation (1) is simplified as a dot product. The performance improvement achieved by factorising the NCC equation will be discussed section 3.1.

2.1.2. Search range propagation (SRP). In this paper, the disparities are estimated iteratively row by row from row $v_{\max}$ to row $v_{\min}$. In the first iteration, the stereo matching goes for a full search range $\mathrm{SR} = \{sr | sr \in [d_{\min}, d_{\max}]\}$. Then, the search range for stereo matching at the position of (u,v) is propagated from three estimated neighbouring disparities $\ell(u-1, v+1)$, $\ell(u, v+1)$ and $\ell(u+1, v+1)$ using equation (9) [22], where τ is the bound of the search range and it is set to 1 in the proposed system. The estimated left disparity map is illustrated in figure 3(c). More details on the SRP-based disparity estimation are given in algorithm 2. The performance of the SRP-based stereo will be discussed in section 3.1:

$$\mathrm{SR} = \bigcup_{k=u-1}^{u+1} \{sr | sr \in [\ell(k, v+1) - \tau, \ell(k, v+1) + \tau]\}. \tag{9}$$

Algorithm 2. SRP-based disparity map estimation.

Input : left image, right image;
left mean map, right mean map;
left standard deviation map, right standard deviation map;
Output: disparity map

estimate the disparities for row $v_{\max}$;
for $v \leftarrow v_{\max} - 1$ to $v_{\min}$ **do**
  for $u \leftarrow u_{\min}$ to $u_{\max}$ **do**
    propagate the search range from row $v+1$ using equation (9);
    estimate the disparity for (u,v);
  end
end

2.1.3. Post-processing. For various disparity map estimation algorithms, the pixels that are only visible in one disparity map are a major source of the matching errors. Due to the uniqueness constraint of the correspondence, for an arbitrary pixel (u,v) in the left disparity map ℓ^{lf}, there exists at most one correspondence in the right disparity map ℓ^{rt}, namely [20]

$$\ell^{lf}(u,v) = \ell^{rt}(u - \ell^{lf}(u,v), v). \tag{10}$$

A left-right consistency (LRC) check is performed to remove half-occluded areas from the disparity map. Although the LRC check doubles the computational complexity by re-projecting the computed disparity values from one image to the other one, most of the incorrect half-occluded pixels can be eliminated and an outlier can be found [20]. For the estimation of ℓ^{rt}, the memorisation of μ_l, μ_r, σ_l and σ_r is unnecessary because they have already been calculated when estimating ℓ^{lf}. More details on the LRC check is given in algorithm 3, where $\mathrm{tr_{LRC}}$ is the threshold and it is set to 3 in this paper. The corresponding LRC check result is shown in figure 3(d).

Algorithm 3. LRC check.

Input : left disparity map: ℓ^{lf}
right disparity map: ℓ^{rt}
Output: disparity map: ℓ

for $v \leftarrow v_{\min}$ to $v_{\max}$ **do**
  for $u \leftarrow u_{\min}$ to $u_{\max}$ **do**
    if $abs(\ell^{lf}(u,v) - \ell^{rt}(u - \ell^{lf}(u,v), v)) > \mathrm{tr_{LRC}}$ **then**
      $\ell(u,v) \leftarrow 0$;
    else
      $\ell(u,v) \leftarrow \ell^{lf}(u,v)$;
    end
  end
end

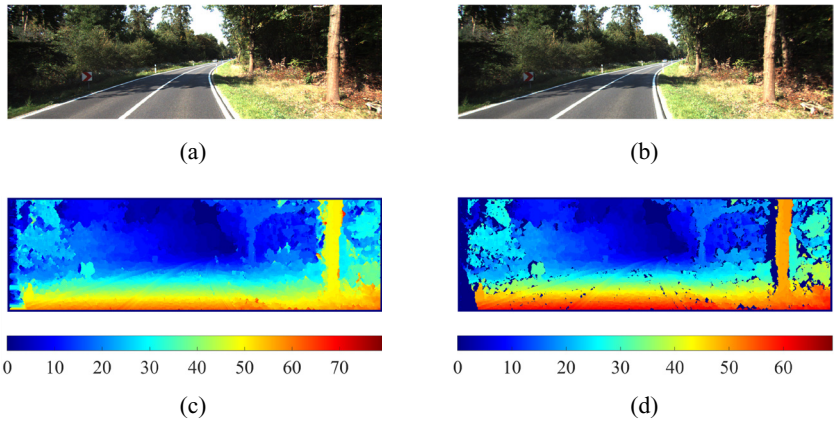

Figure 3. Input images and disparity maps. (a) Left image. (b) Right image. (c) Left disparity map. (d) Left disparity map processed with the LRC.

2.2. Dense v_{vp} estimation

Since Labayrade *et al* proposed the concept of 'v-disparity' in 2002 [19], disparity information has been widely used to improve the detection of either obstacles or lanes. The v-disparity map is created by computing the histogram of each horizontal row of the disparity map. An example of the v-disparity map is shown in figure 4(a), which has two axes: disparity d and row number v. The value $m_v(d, v)$ represents the accumulation at the position of (d, v) in the v-disparity map. In [23], Hu *et al* proved that the disparity projection of a flat road on the v-disparity map is a straight line: $d = f(v) = \alpha_0 + \alpha_1 v$. The parameter vector $\boldsymbol{\alpha} = [\alpha_0, \alpha_1]^\top$ can be obtained by using some linear pattern detectors, such as the Hough transform (HT) [2, 24]. In our previous work [17], we used a parabola model $d = f(v) = \beta_0 + \beta_1 v + \beta_2 v^2$ to represent the disparity projection of a non-flat road surface on the v-disparity map. In this case, the DP is more efficient than some quadratic pattern detectors in terms of searching for every possible solution. The path with the highest accumulations can be extracted by minimising the energy in equation (11):

$$E = E_{\text{data}} + \lambda E_{\text{smooth}}. \tag{11}$$

Equation (11) is solved iteratively starting from $d = d_{\max}$ and going to $d = 0$. In the first iteration, $E_{\text{smooth}} = 0$ and $E_{\text{data}} = -m_v(d_{\max}, v)$. Then, E is computed based upon the previous iterations:

$$E(v)_d = -m_v(d, v) + \min_{\tau_v}[E(v - \tau_v)_{d+1} - \lambda_v \tau_v], \text{ s.t. } \tau_v \in [0, 6]. \tag{12}$$

In each iteration, the index position of the minimum is saved into a buffer for the extraction of the desirable path. The buffer has the same size as the v-disparity map. The solution $\boldsymbol{M_v} = [\boldsymbol{d}, \boldsymbol{v}]^\top \in \mathbb{R}^{k\times 2}$ with the minimal energy is then selected as the optima, which is plotted in blue as shown in figure 4. The blue path includes k points. The two column vectors $\boldsymbol{v} = [v_0, v_1, \ldots, v_{k-1}]^\top$ and $\boldsymbol{d} = [d_0, d_1, \ldots, d_{k-1}]^\top$ record the row numbers and the disparity values, respectively. Therefore, the parameter vector $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2]^\top$ can be estimated by solving the least squares problem in equation (13). The parabola: $f(v) = \beta_0 + \beta_1 v + \beta_2 v^2$ is plotted in red, as shown in figures 4(b) and (c):

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \sum_{j=0}^{k-1} (d_j - (\beta_0 + \beta_1 v_j + \beta_2 v_j^2))^2. \tag{13}$$

From figure 4(b), it can be observed that the outliers severely affect the accuracy of the LSF. To improve v_{vp} estimation, we employ the RANSAC to update the inlier set $\mathcal{I}$ and the parameter vector $\boldsymbol{\beta}$ iteratively. This procedure is detailed in algorithm 4.

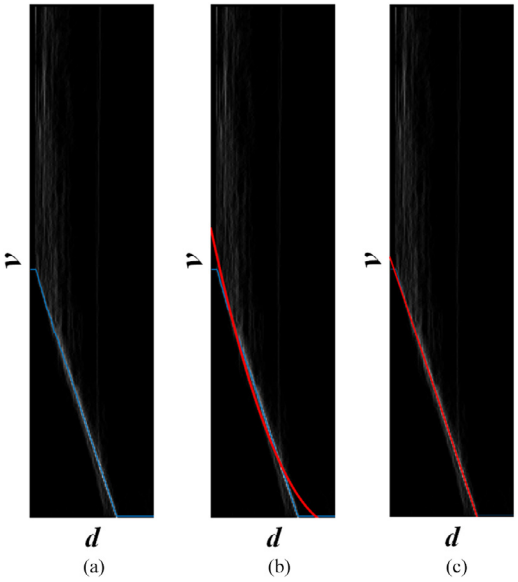

Figure 4. DP and $\boldsymbol{\beta}$ estimation. (a) v-disparity map. (b) Target solution obtained in [17]. (c) Target solution obtained in the proposed system. The blue paths are the optimal solutions obtained using the DP. $f(v) = \beta_0 + \beta_1 v + \beta_2 v^2$ is plotted in red.

Algorithm 4. $\boldsymbol{\beta}$ estimation with the assistance of the RANSAC.

Input : optimal solution: $\boldsymbol{M_v} = [\boldsymbol{d}, \boldsymbol{v}]^\top$
Output: parameter vector: $\boldsymbol{\beta}$
do
  randomly select a specified number of candidates $[d_j, v_j]^\top$;
  fit a parabola to the selected candidates and get $\boldsymbol{\beta}$;
  determine the numbers of inliers and outliers: $n_{\mathcal{I}}$ and $n_{\mathcal{O}}$, respectively;
  remove the outliers from $\boldsymbol{M_v}$;
while $n_{\mathcal{I}}/(n_{\mathcal{I}} + n_{\mathcal{O}}) < \epsilon_v$;
interpolate the candidates in the updated $\boldsymbol{M_v}$ into a parabola and get $\boldsymbol{\beta}$;

To determine whether a given candidate $[d_j, v_j]^\top$ belongs to $\mathcal{I}$, we need to compute the corresponding squared residual $r_j = (d_j - f(v_j))^2$. If r_j is smaller than our pre-set tolerance tr_v, the candidate is marked as an inlier and $\mathcal{I}$ is updated, where tr_v is set to 4 in this paper. Otherwise, it will be marked as an outlier and removed from $\boldsymbol{M_v}$. The iteration works until the percentage of the inliers exceeds our pre-set threshold ϵ_v, where ϵ_v is set to 99%. Finally, the candidates in the updated $\boldsymbol{M_v}$ are used to estimate the parameter vector $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2]^\top$. Compared with the parabola obtained in [17], the parabola estimation with the assistance of the RANSAC is more reliable and less affected by the outliers (an example is shown in figure 4(c)). v_{vp} can be computed as follows:

$$v_{\mathrm{vp}}(v) = v - \frac{\beta_0 + \beta_1 v + 2\beta_2 v^2}{\beta_1 + 2\beta_2 v}. \tag{14}$$

2.3. Dense u_{vp} estimation

2.3.1. Sparse u_{vp} estimation. Before estimating u_{vp}, we first estimate the road surface area by comparing the difference between the actual and fitted disparity values. A pixel at (u, v) in the disparity map ℓ is considered to be in the road surface area if it satisfies the conditions $|\ell(u, v) - f(v)| \leqslant \mathrm{tr}_{\mathrm{RSE}}$ and $f^{-1}(0) \leqslant v \leqslant v_{\max}$, where f^{-1} is the inverse function of $f(v) = \beta_0 + \beta_1 v + \beta_2 v^2$ and $\mathrm{tr}_{\mathrm{RSE}} = 3$ is a threshold set to remove the obstacles and potholes. The estimated road surface area is illustrated in green as shown in figure 5(a). This greatly reduces the unnecessary edge information used for dense u_{vp} estimation. The procedures in the later sections only focus on the road surface area.

Furthermore, the noise introduced in the imaging procedure makes the edge detectors such as Sobel very sensitive to the blobs [25]. Therefore, we use a bilateral filter to reduce the noise before detecting edges. Compared with the median filter which was utilised in [17], the bilateral filter is more capable of preserving edges when smoothing an image. The expression of a bilateral filter is shown as follows [26]:

$$i^{bf}(u, v) = \frac{\sum_{x=u-\varrho}^{x=u+\varrho} \sum_{y=v-\varrho}^{y=v+\varrho} \omega_s(x, y)\omega_r(x, y)i(x, y)}{\sum_{x=u-\varrho}^{x=u+\varrho} \sum_{y=v-\varrho}^{y=v+\varrho} \omega_s(x, y)\omega_r(x, y)} \tag{15}$$

where

$$\omega_s(x, y) = \exp\left\{-\frac{(x-u)^2 + (y-v)^2}{\sigma_s^2}\right\}$$
$$\omega_r(x, y) = \exp\left\{-\frac{(i(x, y) - i(u, v))^2}{\sigma_r^2}\right\}. \tag{16}$$

$i(x, y)$ is the intensity of the input image at (x, y) and $i^{bf}(u, v)$ is the intensity of the filtered image at (u, v). The block size of the filter is $(2\varrho + 1) \times (2\varrho + 1)$, and its centre is (u, v). The coefficients ω_s and ω_r are based on spatial distance and colour similarity, respectively. σ_s and σ_r are the parameters of ω_s and ω_r, respectively. In order to preserve only the edge information required for lane detection, σ_s and σ_r are set to 300 and 0.3, respectively. The output of bilateral filtering is shown in figure 5(b). The edge detection results of figures 3(a) and 5(b) are illustrated in figures 5(c) and (d), respectively. As for the edge detection result of the median filtering output, it is depicted in figure 5(e). Obviously, although the median filter has removed a lot of redundant edges, the bilateral filter still achieves a better performance in terms of noise elimination and edge preservation.

In the following procedures, we only consider the pixels in the road surface area. The edge map in the road surface area is shown in figure 5(f). The sparse u_{vp} of each edge pixel $\boldsymbol{p_e} = [u_e, v_e]^\top$ can be estimated using equation (17), where $\nabla(\boldsymbol{p_e}) = [g_u, g_v]^\top$ is the gradient of $\boldsymbol{p_e}$ that can be approximated using a Sobel operator. g_u and g_v represent the vertical and horizontal gradients of $\boldsymbol{p_e}$, respectively. $u^s_{\mathrm{vp}}(u_e, v_e)$ at the position of (u_e, v_e) is recorded in a 2D sparse u_{vp} map. It is to be noted that u^s_{vp} represents sparse u_{vp} in this paper:

$$u^s_{\mathrm{vp}}(u_e, v_e) = u_e + \frac{v_e - v_{\mathrm{vp}}(v_e)}{\nabla(\boldsymbol{p_e})}. \tag{17}$$

Next, we provide some details on the implementation. In the GPU architecture, a thread is more likely to fetch the memory from the closest addresses that its nearby threads accessed, which makes the use of cache impossible [27]. Therefore, we utilise the texture memory which is read-only and cached on-chip to optimise the caching for 2D spatial locality during the bilateral filtering. Firstly, a 2D texture object is created. Then, the texture object is bound directly to the global memory address of the left image. The value of a pixel $i(x, y)$ is then fetched from the texture object to reduce the memory requests from the global memory. Furthermore, as the constant memory is read-only and beneficial for the data that will not change over the course of a kernel execution [27], we create two lookup tables on it to store the values of ω_s and ω_r. So far, the execution of the bilateral filtering has been highly accelerated, and we move to the implementation of the edge detection. The address of $i^{bf}(u, v)$ is always accessed repeatedly when determining whether a neighbour of $i^{bf}(u, v)$ belongs to an edge or not. Thus, we load a group of data i^{bf} into the shared memory for each thread block. All threads within the same thread block will access the shared data instead of fetching them repeatedly from the global memory. In order to avoid the race conditions among different threads which run logically in parallel instead of executing physically concurrently, the threads within the same thread block need to be synchronised after they finish the data loading. Compared with the implementation on a Core-i7 4720HQ CPU processing with a single

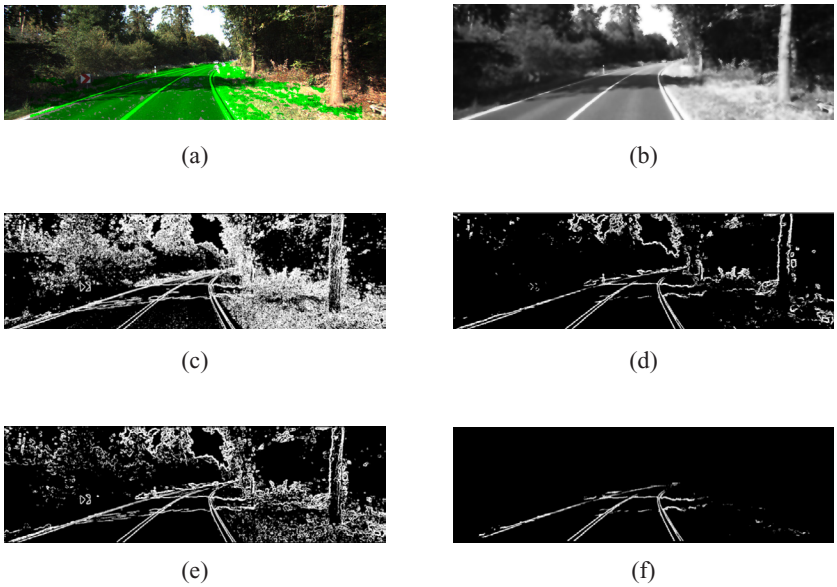

Figure 5. Sparse u_{vp} estimation. (a) Road surface estimation. (b) Bilateral filtering for figure 3(a). (c) Edge detection result of figure 3(a). (d) Edge detection result of (b). (e) Edge detection result of the median filtering output. (f) Edges in the road surface area. The green area in (a) illustrates the road surface. For the process of the bilateral filtering, σ_s and σ_r are set to 300 and 0.3, respectively. The window sizes of the bilateral filter and the median filter are 11×11. The thresholds of the Sobel edge detection in (c)–(f) are 100. In the following procedures, we only consider the edge pixels in (f).

thread, our implementation on a GTX 970M GPU speeds up the execution of sparse u^s_{vp} estimation by over 74 times.

2.3.2. Dense u_{vp} accumulation. To acquire u^d_{vp} information, we accumulate the votes of u^s_{vp} within a rectangle of width $2w + 1$, where w is set to 25 to accumulate as many votes as possible without compromising the execution speed. It is to be noted here that u^d_{vp} represents the dense u_{vp}. More details on the process of u^d_{px} accumulation are given in algorithm 5.

Algorithm 5. Dense u_{vp} accumulation.

Input : 2D u^s_{vp} map
Output: 2D u^d_{vp} accumulator

set all elements in u^d_{vp} accumulator to 0;
for $v \leftarrow v_{\max} - 2w$ to $v_{\max}$ **do**
  for $u \leftarrow u_{\min}$ to $u_{\max}$ **do**
    $u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v), v_{\max} - w) \leftarrow u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v), v_{\max} - w) - m$
  end
end
for $v \leftarrow v_{\max} - w - 1$ to $f^{-1}(0) + w$ **do**
  for $u \leftarrow u_{\min}$ to $u_{\max}$ **do**
    $u^d_{\mathrm{vp}}(u, v) \leftarrow u^d_{\mathrm{vp}}(u, v + 1)$
    $u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v - w), v) \leftarrow u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v - w), v) - m$;
    $u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v + w + 1), v) \leftarrow u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v + w + 1), v) + m$;
  end
end
for $v \leftarrow f^{-1}(0) + w - 1$ to $f^{-1}(0)$ **do**
  for $u \leftarrow u_{\min}$ to $u_{\max}$ **do**
    $u^d_{\mathrm{vp}}(u, v) \leftarrow u^d_{\mathrm{vp}}(u, v + 1)$;
    $u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v + w + 1), v) \leftarrow u^d_{\mathrm{vp}}(u^s_{\mathrm{vp}}(u, v + w + 1), v) + m$;
  end
end

The initial step is to form a 1D u^d_{vp} histogram by accumulating the votes from each edge pixel $\boldsymbol{p_e}$ on row $v_{\max} - w$. In order to ensure energy minimisation rather than energy maximisation, the parameter m has to be a positive number and it is simply set to 1 in this paper. Then, the votes of u^s_{vp} from each edge pixel $\boldsymbol{p_e}$ on row $v - 1$ are accumulated with the 1D u^d_{vp} histogram on row v to form the 1D u^d_{vp} histogram on row $v - 1$. This works until the width of the rectangle is able to reach $2w + 1$. Then, the current rectangle is shifted slightly up to create another 1D u^d_{vp} histogram. In order to improve the computational efficiency, sliding window algorithm is used to create 1D u^d_{vp} histograms. The votes that appear on row $v - w$ above from the current rectangle are subtracted and those that appear on the bottom row of the previous rectangle are added. It is to be noted here that more u^s_{vp} votes correspond to a negatively higher value in the 2D u^d_{vp} accumulator. This ensures the energy minimisation in the DP.

Furthermore, due to that the far field of the road may contain a higher lane curvature, a thinner rectangle is more desirable for the top rows of the image [17]. Therefore, only the votes on row $v + w + 1$ are added to update the 1D u^d_{vp} histogram without the subtractions from the current rectangle. The sliding window algorithm makes the 1D u^d_{vp} histogram update more efficiently by simply processing the bottom row and the top row. The result of dense u_{vp} accumulation is illustrated in figure 6, where $m_u(u, v)$ represents the votes of $u_{\mathrm{vp}} = u$ on row v.

2.3.3. u_{vp} estimation. Similarly, u^d_{vp} accumulator is optimised using the DP to extract the path with the minimal energy, as shown in equation (11). In the first iteration, $E_{\mathrm{smooth}} = 0$ and

$E_{\text{data}} = m_u(u, v_{\max})$. Then, E is computed based on the previous iterations:

$$E(u)_v = m_u(u, v) + \min_{\tau_u}[E(u_{\text{vp}} + \tau_u)_{v+1} + \lambda_u \tau_u], \text{ s.t. } \tau_u \in [-5, 5]. \tag{18}$$

The solution $\boldsymbol{M_u} = [\boldsymbol{u}, \boldsymbol{v}]^\top \in \mathbb{R}^{t\times 2}$ with the minimal energy is then selected as the optima, which is plotted in blue, as shown in figure 6. The blue path includes t points. The two column vectors $\boldsymbol{u} = [u_0, u_1, \ldots, u_{t-1}]^\top$ and $\boldsymbol{v} = [v_0, v_1, \ldots, v_{t-1}]^\top$ record the column and row numbers, respectively. The parameter vector $\boldsymbol{\gamma} = [\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4]^\top$ can be estimated by solving the least squares problem in equation (19). Here, we use the same strategy as the estimation of $\boldsymbol{\beta}$. $\mathcal{I}$ and $\boldsymbol{M_u}$ are updated using the RANSAC until the percentage of the inliers exceeds our pre-set threshold. Then, $\boldsymbol{\gamma}$ is obtained by fitting a quartic polynomial to the candidates in the updated $\boldsymbol{M_u}$. Algorithm 6 provides more details on $\boldsymbol{\gamma}$ estimation:

$$\boldsymbol{\gamma} = \arg\min_{\boldsymbol{\gamma}} \sum_{j=0}^{t-1} (u_j - (\gamma_0 + \gamma_1 v_j + \gamma_2 {v_j}^2 + \gamma_3 {v_j}^3 + \gamma_4 {v_j}^4))^2. \tag{19}$$

Algorithm 6. $\boldsymbol{\gamma}$ estimation with the assistance of the RANSAC.

Input : optimal solution: $\boldsymbol{M_u} = [\boldsymbol{u}, \boldsymbol{v}]^\top$
Output: parameter vector: $\boldsymbol{\gamma}$
do
  randomly select a specified number of candidates $[u_j, v_j]^\top$;
  fit a quartic polynomial to the selected candidates and get $\boldsymbol{\gamma}$;
  determine the numbers of inliers and outliers: $n_{\mathcal{I}}$ and $n_{\mathcal{O}}$, respectively;
  remove the outliers from $\boldsymbol{M_u}$;
while $n_{\mathcal{I}}/(n_{\mathcal{I}} + n_{\mathcal{O}}) < \epsilon_u$
fit a quartic polynomial to the candidates in the updated $\boldsymbol{M_u}$ and get $\boldsymbol{\gamma}$;

To determine whether a given candidate $[u_j, v_j]^\top$ belongs to $\mathcal{I}$, we need to compute the corresponding squared residual $r_j = (u_j - g(v_j))^2$. If r_j is smaller than our pre-set threshold tr_u, the candidate is marked as an inlier and $\mathcal{I}$ is updated, where tr_u is set to 16 in this paper. Otherwise, it will be marked as an outlier and removed from $\boldsymbol{M_u}$. The iteration works until the percentage of the inliers exceeds our pre-set threshold ϵ_u, where ϵ_u is set to 99% in this paper. Finally, $\boldsymbol{\gamma}$ can be estimated by fitting a quartic polynomial to the updated $\boldsymbol{M_u}$. Compared with the target solution obtained in [17], as shown in figure 6(b), the target solution obtained in the proposed system is less affected by the outliers (an example is shown in figure 6(c)). In our practical implementation, equation (19) is rearranged as shown in equation (20) to avoid the data overflow when fitting the quartic polynomial:

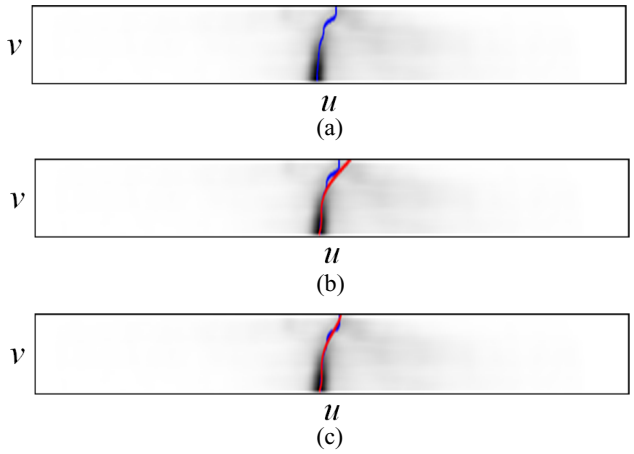

Figure 6. Dense u_{vp} accumulation and estimation. (a) dense u_{vp} accumulator. (b) Target solution obtained in [17]. (c) Target solution obtained in the proposed system. The blue paths are the optimal solutions obtained using the DP. $g(v) = \gamma_0 + \gamma_1 v + \gamma_2 v^2 + \gamma_3 v^3 + \gamma_4 v^4$ is plotted in red.

$$\boldsymbol{\gamma} = (\kappa_0 \boldsymbol{P}^\top \boldsymbol{P})^{-1} (\kappa_0 \boldsymbol{P}^\top) \boldsymbol{u} \tag{20}$$

where

$$\boldsymbol{P} = \begin{bmatrix} 1 & v_0 & \cdots & {v_0}^4 \\ 1 & v_1 & \cdots & {v_1}^4 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & v_{t-1} & \cdots & {v_{t-1}}^4 \end{bmatrix}. \tag{21}$$

$\boldsymbol{P}$ is a Vandermonde matrix. κ_0 is used to avoid the data overflow problem caused by the higher order polynomials (e.g. when $v = 375$, $v^8 \approx 4 \times 10^{20}$ which is far beyond the significand range of *long double* type in C language). After estimating $\boldsymbol{\gamma}$, u_{vp} can be computed as follows:

$$u_{\text{vp}}(v) = \gamma_0 + \gamma_1 v + \gamma_2 v^2 + \gamma_3 v^3 + \gamma_4 v^4. \tag{22}$$

2.4. Lane position validation

$\boldsymbol{p_{\text{vp}}}$ provides the tangential direction and the curvature information of lanes, which can help us to validate the lane positions. In [17], the authors formed a likelihood function $V(\boldsymbol{p_e}) = \nabla(\boldsymbol{p_e}) \cdot \cos(\theta_{\boldsymbol{p_e}} - \theta_{\boldsymbol{p_{\text{vp}}}})$ for each edge point $\boldsymbol{p_e}$ and selected the plus-minus peak pairs for visualisation, where $\theta_{\boldsymbol{p_e}}$ is the angle between the u-axis and the orientation of the edge point $\boldsymbol{p_e}$, and $\theta_{\boldsymbol{p_{\text{vp}}}}$ is the angle between the u-axis and the radial from an edge pixel $\boldsymbol{p_e}$ to $\boldsymbol{p_{\text{vp}}}(v_e)$. Although the peak pair selection algorithm presented in [17] has achieved some impressive experimental results, some inaccurate detections still occurred. In this paper, we compute the energy of each possible solution and select all satisfying lane positions for visualisation. Algorithm 7 provides more details on the proposed approach.

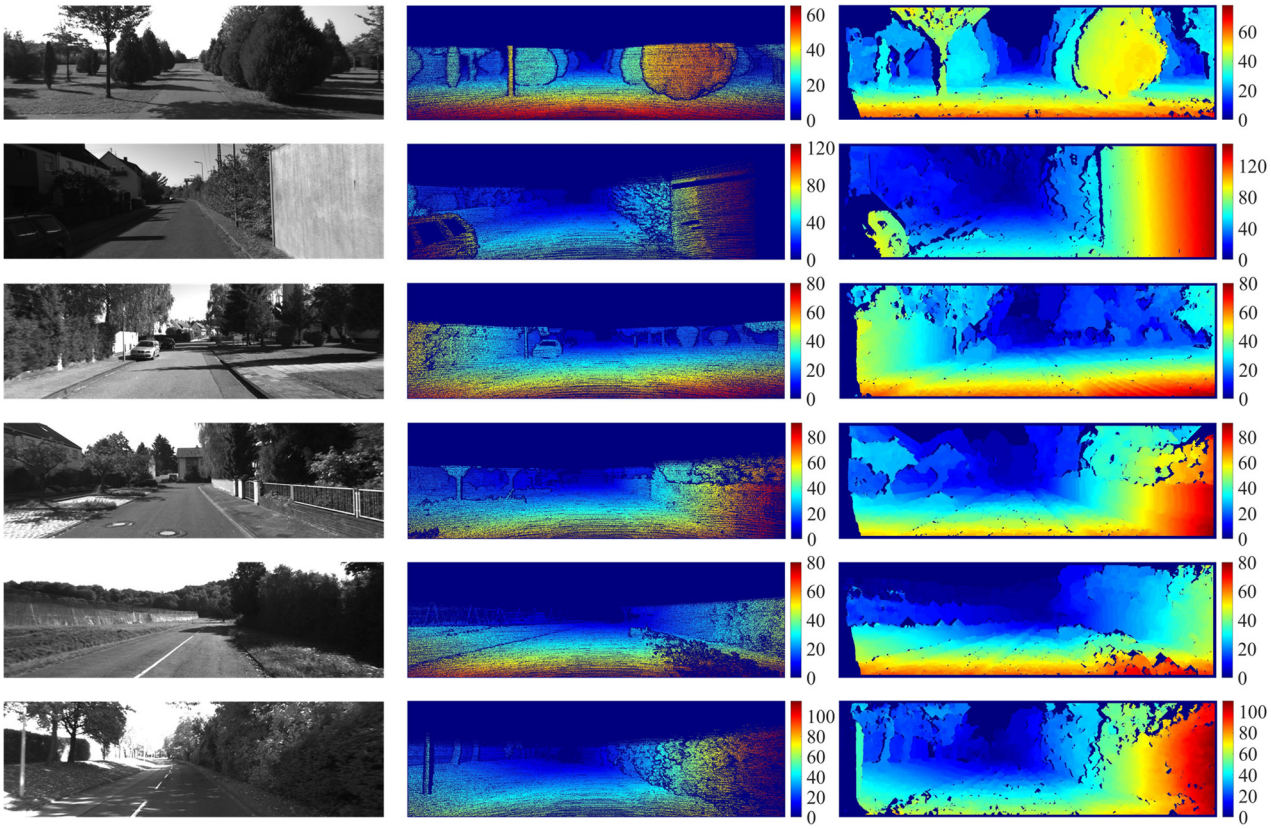

Figure 7. Experimental results of the KITTI stereo 2012 dataset [28].

Algorithm 7. Lane position validation.

Input : $\boldsymbol{p}_{\mathbf{vp}}$ and g_u
Output: lane position vector: $\boldsymbol{\delta}$
create two 2D maps $\mathcal{M}_0$, $\mathcal{M}_1$ with the same size as the input image i;
set all elements in $\mathcal{M}_0$, $\mathcal{M}_1$ to 0;
create a 1D histogram $\mathcal{H}$ of size $(2t+1)u_{\max}$;
$\forall \mathcal{M}_0(u,v) \leftarrow \sum_{x=-\kappa_1}^{x=+\kappa_1} \sum_{y=-\kappa_2}^{y=+\kappa_2} g_u(u+x, v+y)\omega_g(u+x, v+y)$;
approximate the horizontal gradient of each point in $\mathcal{M}_0$ using the Sobel operator and save the results in $\mathcal{M}_1$;
for $k \leftarrow -tu_{\max}$ to $tu_{\max}$ **do**
  aggregate $\mathcal{M}_1$ from row $v_{\max}$ to row $f^{-1}(0)$ to get the energy E;
  $\mathcal{H}(k + tu_{\max}) \leftarrow E$;
end
if $\mathcal{H}(k) < \min\{\mathcal{H}(k-1), \mathcal{H}(k+1)\}$ **then**
  put $k - tu_{\max}$ into $\boldsymbol{\delta}$;
end
remove the elements which are smaller than the threshold $\mathrm{tr}_{\mathrm{LPV}}$ from $\boldsymbol{\delta}$;
remove the nearby candidates from $\boldsymbol{\delta}$;
multiple lanes visualisation;

For the dark-light transition of a lane marking, the value of g_u is positive and higher than the ones at the non-edge positions. As for the light-dark transition of a lane marking, the value of g_u is negative but its absolute value is still higher than the ones at the non-edge positions [17]. Therefore, the task to validate lane positions now only involves the estimation of the centre position of each pair of dark-light and light-dark transitions. To reduce g_u accumulation from the non-lane edges, we propose a piecewise weighting ω_g as follows:

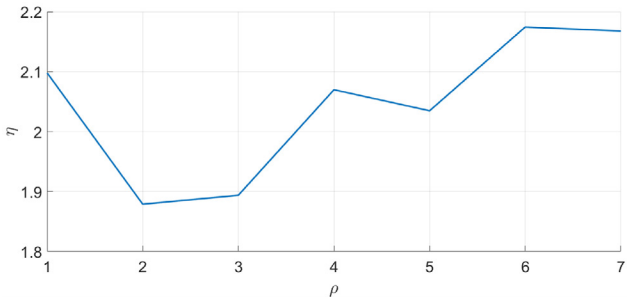

Figure 8. Evaluation of the algorithm execution speed.

Table 1. Detection results of the proposed algorithm.

Sequence	Lanes	Incorrect detection	Misde-tection
1	860	0	0
2	594	0	0
3	376	0	0
4	156	0	0
5	678	0	0
6	1060	1	2
7	644	0	0
8	993	18	7
Total	5361	19	9

Table 2. Detection results of [17].

Sequence	Lanes	Incorrect detection	Misde-tection
1	860	0	0
2	594	0	0
3	376	0	0
4	156	0	9
5	678	0	17
6	1060	14	7
Total	3724	14	33

Table 3. Detection results of [18].

Sequence	Lanes	Incorrect detection	Misde-tection
1	860	12	0
2	594	44	0
3	376	44	0
4	156	17	0
5	678	107	0
6	1060	180	0
Total	3724	404	0

$$\omega_g(u_e, v_e) = \begin{cases} \exp\left(-\frac{|\theta_{\boldsymbol{p_e}} - \theta_{\boldsymbol{V_p}}|}{{\sigma_g}^2} \cdot \frac{1}{\theta_s}\right), & |\theta_{\boldsymbol{p_e}} - \theta_{\boldsymbol{V_p}}| \leqslant \frac{\pi}{6} \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

where the step θ_s is set to $\pi/36$. The portion $|\theta_{\boldsymbol{p_e}} - \theta_{\boldsymbol{p_{VP}}}|/\theta_s$ is used to provide a Gaussian weight so as to decrease the magnitude of noise pixels, where σ_g is set to 3.5 in this paper. Then, we sum the values of $g_u\omega_g$ within a shifting box to further reduce the noise. The box size is $(2\kappa_1 + 1) \times (2\kappa_2 + 1)$, where κ_1 and κ_2 are empirically set to 1 and 3, respectively. The accumulation output is then saved in a 2D map $\mathcal{M}_0$, where the horizontal gradient from a dark-light peak to a light-dark peak is negative. To approximate the horizontal gradients, the Sobel horizontal kernel is convoluted with $\mathcal{M}_0$ and the convolution result is saved in $\mathcal{M}_1$. Then, we aggregate the values of $\mathcal{M}_1$ for each possible solution from row $v_{\max}$ to row $f^{-1}(0)$ as follows:

$$E(v)_{u_v} = \mathcal{M}_1(u_v, v) + \lambda_g E(v+1)_{u_{v+1}} \tag{24}$$

where

$$u_v = \frac{u_{\text{vp}}(v+1) + vu_{v+1} - v_{\text{vp}}(v+1)u_{v+1}}{v + 1 - v_{\text{vp}}(v+1)}. \tag{25}$$

In order to find all possible lane positions, we set t to 0.5. This implies that u starts from $-0.5u_{\max}$ and ends at $1.5u_{\max}$. In the first iteration, we select a possible position $(u_{v_{\max}}, v_{\max})$ and the total energy E is simply set to $\mathcal{M}_1(u_{v_{\max}}, v_{\max})$. Then, we use $\boldsymbol{p_{vp}}$ information to estimate the next position $(u_{v_{\max}-1}, v_{\max} - 1)$ on the selected track. The energy E is updated using equation (24). Here, λ_g has been used for test purpose and the value of 1 has been found to provide the good results during our experiments. The aggregation of $\mathcal{M}_1$ works until v reaches $f^{-1}(0)$. For each lane starting from $(u_{v_{\max}}, v_{\max})$, its total energy is saved in a 1D histogram $\mathcal{H}$. Then, we pick out the local minima which are smaller than our pre-set threshold tr_{LPV}. At the same time, if two local minima are quite close to each other, we ignore the minima with a larger energy. Finally, the lanes can be visualised by iterating equation (25). The lane detection results will be discussed in section 3.2.

3. Experimental results

3.1. Stereo vision evaluation

The proposed disparity estimation algorithm is evaluated using the KITTI stereo 2012 dataset [28]. Some examples of the experimental results are shown in figure 7, where the first column illustrates the input left gray-scale images, the second column shows the ground truth disparity maps, and the third column depicts our experimental results. The overall percentage of the error pixels is approximately 6.82% (error threshold: two pixels).

The proposed disparity estimation algorithm is implemented on an NVIDIA GTX 970M GPU for the real-time purpose. Please kindly refer to our publication [20] for more implementation details. When ρ is set to 3, the implementation achieves a speed of 37 fps. After the memorisation, the values of μ and σ can be accessed directly from a static program storage for stereo matching, which greatly boosts the algorithm execution. The performance improvement achieved using the memorisation is shown in figure 8, where η represents the runtime of the prevalent NCC-based stereo versus the runtime of NCC-based stereo optimised with memorisation. From figure 8, it can be seen that the memorisation speeds up the algorithm execution by about two times.

3.2. Lane detection evaluation

Currently, it is impossible to access a satisfying ground truth dataset for the evaluation of lane detection algorithms because accepted test protocols do not usually exist [29]. Therefore, many publications related to lane detection only focus on the quality of their experimental results [17]. For this reason, we compare the performance of the proposed system with [17] and [18] in terms of both accuracy and speed. Some successful detection examples are shown in figure 9.

Firstly, we evaluate the accuracy of the proposed algorithm. The lane detection results of the proposed algorithm and the algorithms described in [17] and [18] are detailed in tables 1–3 respectively. To evaluate the robustness of the proposed algorithm, we tested eight sequences selected from the KITTI database (including two additional sequences): 2495 frames containing 5361 lanes [30] (1637 lanes more than what were used in [17] and [18]). The image resolution is 1242×375 in sequences 1 to 6, 1241×376 in sequence 7, and 1238×374 in sequence 8. From table 1, it can be seen that the proposed algorithm presents a better detection ratio, where 99.9% lanes are successfully detected in sequences 1 to 7 (including all the sequences in tables 2 and 3), while the detection ratios of [17] and [18] are only 98.7% and 89.2%, respectively. The comparison between some failed examples in [17] and the

Figure 9. Lane detection results. The red lines illustrate the detected lanes.

corresponding results obtained using the proposed algorithm is illustrated in figure 10.

In figure 10(a), we can see that the obstacle areas occupy a larger portion than the road surface area, which severely affects the accuracy of v_{vp} estimation. When we use both inliers and outliers in the LSF, $\boldsymbol{p}_{\mathbf{vp}}$ differs too much from the ground truth. This further influences the lane position validation and leads to an imprecise detection and a misdetection. In figure 10(b), it can be seen that the LSF considering only inliers increases the precision of v_{vp} estimation significantly. Moving to the second row, an over-curved lane can be seen in figure 10(c). When we estimate $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ with the assistance of the RANSAC, the improvement can be observed in figure 10(d), where a more reasonable lane is detected. In figure 10(e), the lane near the left road boundary is misdetected because the low contrast between lane and road surface reduces its magnitude in the stage of lane position validation. In figure 10(g), an incorrect detection occurs because a road marking is more contrastive to the road surface. In section 2.4, we proposed a more effective piecewise weighting ω_g to update g_u for the edge pixels. Then, g_u of the non-lane edges reduces significantly, which therefore greatly helps the system to avoid the incorrect detections of some lane markings. Also, we sum $g_u\omega_g$ within a shifting box for each position. This increases the magnitude of the lanes which are lowly contrastive to the road surface. The misdetections in figures 10(e) and (g) are thus detectable, and the failed detection in figure 10(e) is also corrected. The corresponding results are shown in figures 10(f) and (h).

In our experiment, the failed cases consist of misdetections and incorrect detections. The misdetections are mainly caused by image over-exposure, partially occluded by the obstacles, forks on the road. The corresponding examples are illustrated in figures 11(a)–(c), respectively. In figure 11(a), due to the image over-exposure, the edges pixels on lane 1 are rare, which leads to its misdetection. In figure 11(b), we

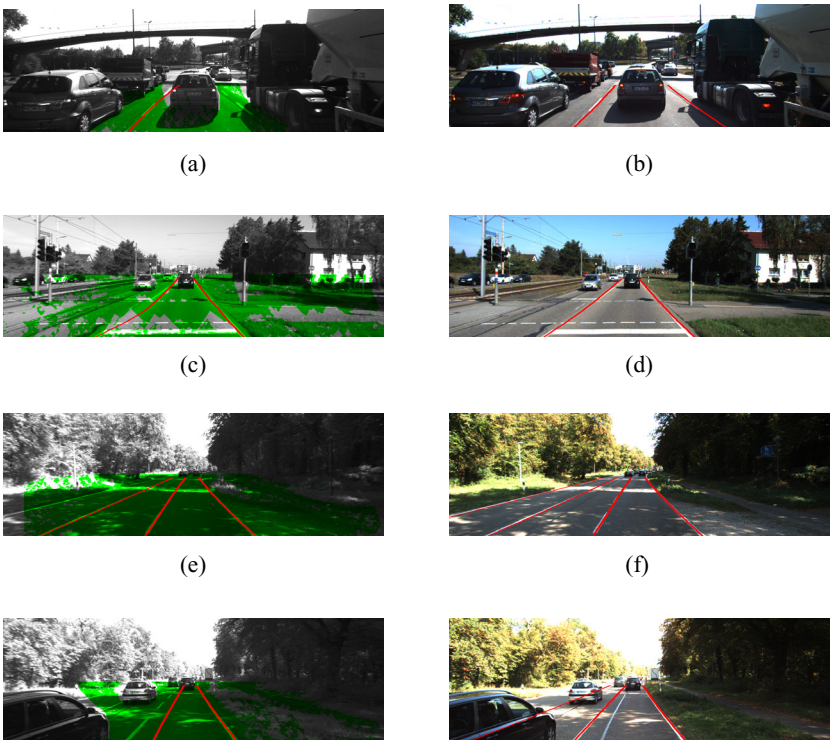

Figure 10. Comparison between some failed examples in [17] and the corresponding results in this paper. The green areas in the first column illustrate the road surface. The red lines are the detected lanes. The first column illustrates the failed examples in [17], and the second column shows the corresponding results of the proposed system.

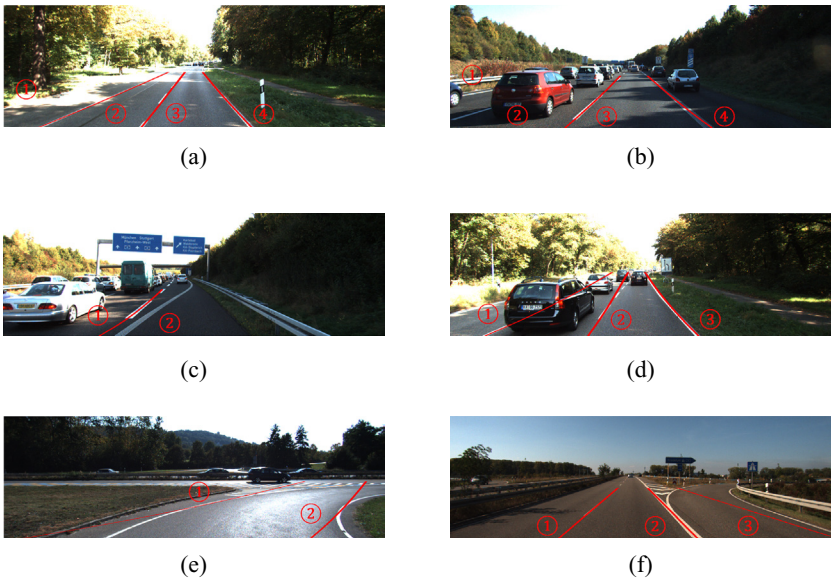

Figure 11. Examples of the failed detections in this paper.

can see that the vehicles partially occlude lane 1 and lane 2. The occlusion decreases the absolute value of $g_u\omega_g$ when we try to validate the lane positions, which makes lane 1 and lane 2 undetectable. In figure 11(c), lane 2 forks from lane 1, and thus, has a different curvature information from lane 1, which therefore causes the misdetection.

For the factors leading to incorrect detections, we group them into three main categories: ambiguous disparity projection of a road surface on the v-disparity map; $\boldsymbol{p}_{\mathbf{vp}}$ that does not exist in the image; different roadways, which are presented in figures 11(d)–(f), respectively. In figure 11(d), the obstacles, e.g. vehicles and trees, take a big portion in the image. Therefore, when d is around 0, m_v is mainly accumulated by the pixels on the obstacles and sky, which affects the accuracy of v_{vp} estimation. This further makes the detected lane markings slightly above the ground truth when they move to the boundary between the road surface and sky. In figure 11(e), $\boldsymbol{p}_{\mathbf{vp}}$ does not exist, which affects the detection results. In figure 11(f), there

are two different roadways: roadway between lane 1 and lane 2; roadway between lane 2 and lane 3. The second roadway turns right and therefore has a different $\boldsymbol{p}_{\mathbf{vp}}$ from the first roadway, which leads to an imprecise detection of lane 3.

Finally, we discuss the processing speed. The algorithm is implemented on a heterogeneous system consisting of an Intel Core i7-4720HQ CPU and an NVIDIA GTX 970M GPU. The GPU has 10 streaming multiprocessors with 128 CUDA cores on each of them. The runtime of the proposed system is around 7 ms (excluding the runtime of the disparity estimation), which is approximately 38 times faster than our previous work where 263 ms was achieved. The authors believe that the failed cases can be reduced in the future by adding a lane tracking algorithm. The demo videos are available at http://www.ruirangerfan.com

4. Conclusion and future work

A multiple lane detection system was presented in this paper. The novelties include an improved dense vanishing point estimation method, a novel lane position validation algorithm and a real-time implementation on a heterogeneous system. To evaluate the performance, 5361 lanes from eight datasets were tested. The experimental results illustrate that the proposed algorithm works more accurately and robustly than our previous work. By highly exploiting the GPU architecture and allocating different parts of the proposed algorithm on different platforms for execution, a high processing speed of 143 fps was achieved, which is approximately 38 times faster than our previous work.

As discussed in section 3.2, some actual road conditions may result in failed detections. Therefore, the authors plan to train a deep neural network for dense vanishing point estimation and lane position validation. Furthermore, the authors also plan to implement the proposed algorithm on some state-of-the-art embedded systems, such as Jetson TK2 from NVIDIA.

ORCID iDs

Rui Fan https://orcid.org/0000-0003-2593-6596

References

[1] Rosenzweig J and Bartl M 2015 A review and analysis of literature on autonomous driving *Mak. Innov.*
[2] Fan R, Prokhorov V and Dahnoun N 2016 Faster-than-real-time linear lane detection implementation using SoC DSP TMS320C6678 *IEEE Int. Conf. on Imaging Systems and Techniques* (IEEE) pp 306–11
[3] Narote S P, Bhujbal P N, Narote A S and Dhane D M 2018 A review of recent advances in lane detection and departure warning system *Pattern Recognit.* **73** 216–34
[4] Bertozzi M and Broggi A 1998 Gold: a parallel real-time stereo vision system for generic obstacle and lane detection *IEEE Trans. Image Process.* **7** 62–81
[5] Wang Y, Teoh E K and Shen D 2004 Lane detection and tracking using B-Snake *Image Vis. Comput.* **22** 269–80
[6] Ozgunalp U and Dahnoun N 2014 Robust lane detection & tracking based on novel feature extraction and lane categorization *IEEE Int. Conf. on Acoustics, Speech and Signal Processing* (IEEE) pp 8129–33
[7] Kluge K and Lakshmanan S 1995 A deformable-template approach to lane detection *Proc. Intelligent Vehicles Symp.* (IEEE) pp 54–9
[8] Wang Y, Bai L and Fairhurst M 2008 Robust road modeling and tracking using condensation *IEEE Trans. Intell. Transp. Syst.* **9** 570–9
[9] Zhou Y, Xu R, Hu X and Ye Q 2006 A robust lane detection and tracking method based on computer vision *Meas. Sci. Technol.* **17** 736
[10] Kreucher C and Lakshmanan S 1999 LANA: a lane extraction algorithm that uses frequency domain features *IEEE Trans. Robot. Autom.* **15** 343–50
[11] Jung C R and Kelber C R 2005 An improved linear-parabolic model for lane following and curve detection *18th Brazilian Symp. on Computer Graphics and Image Processing* (IEEE) pp 131–8
[12] Wang Y, Shen D and Teoh E K 2000 Lane detection using spline model *Pattern Recognit. Lett.* **21** 677–89
[13] Nieto M, Salgado L, Jaureguizar F and Cabrera J 2007 Stabilization of inverse perspective mapping images based on robust vanishing point estimation *IEEE Intelligent Vehicles Symp.* (IEEE) pp 315–20
[14] Schreiber D, Alefs B and Clabian M 2005 Single camera lane detection and tracking *Proc. IEEE Intelligent Transportation Systems* (IEEE) pp 302–7
[15] Hanwell D and Mirmehdi M 2012 Detection of lane departure on high-speed roads *ICPRAM* pp 529–36
[16] Fardi B and Wanielik G 2004 Hough transformation based approach for road border detection in infrared images *IEEE Intelligent Vehicles Symp.* (IEEE) pp 549–54
[17] Ozgunalp U, Fan R, Ai X and Dahnoun N 2017 Multiple lane detection algorithm based on novel dense vanishing point estimation *IEEE Trans. Intell. Transp. Syst.* **18** 621–32
[18] Wang Y, Dahnoun N and Achim A 2012 A novel system for robust lane detection and tracking *Signal Process.* **92** 319–34
[19] Labayrade R, Aubert D and Tarel J-P 2002 Real time obstacle detection in stereovision on non flat road geometry through *v*-disparity representation *IEEE Intelligent Vehicle Symp.* (IEEE) vol 2 pp 646–51
[20] Fan R and Dahnoun N 2017 Real-time implementation of stereo vision based on optimised normalised cross-correlation and propagated search range on a GPU *IEEE Int. Conf. on Imaging Systems and Techniques* (IEEE) pp 241–6
[21] Lewis J P 1995 Fast template matching *Vis. Interface* **95** 15–9
[22] Fan R, Ai X and Dahnoun N 2018 Road surface 3d reconstruction based on dense subpixel disparity map estimation *IEEE Trans. Image Process.* **27** 3025–35
[23] Hu Z, Lamosa F and Uchimura K 2005 A complete uv-disparity study for stereovision based 3d driving environment analysis *5th Int. Conf. on 3D Digital Imaging and Modeling* (IEEE) pp 204–11
[24] Ballard D H 1981 Generalizing the hough transform to detect arbitrary shapes *Pattern Recognit.* **13** 111–22
[25] Rafael Gonzalez C and Woods R 2002 *Digital Image Processing* (Pearson Education)
[26] He K, Sun J and Tang X 2013 Guided image filtering *IEEE Trans. Pattern Anal. Mach. Intell.* **35** 1397–409
[27] NVIDIA 2017 Cuda c programming guide
[28] Andreas A, Lenz P and Urtasun R 2012 Are we ready for autonomous driving? The KITTI vision benchmark suite *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (https://doi.org/10.1109/CVPR.2012.6248074)
[29] Hillel A B, Lerner R, Levi D and Raz G 2014 Recent progress in road and lane detection: a survey *Mach. Vis. Appl.* **25** 727–45
[30] Geiger A, Lenz P, Stiller C and Urtasun R 2013 Vision meets robotics: the KITTI dataset *Int. J. Robot. Res.* **32** 1231–7