*remote sensing*

MDPI

*Article*

# Motion Estimation Using Region-Level Segmentation and Extended Kalman Filter for Autonomous Driving

**Hongjian Wei [1,2,†], Yingping Huang [1,*,†], Fuzhi Hu [1,3], Baigan Zhao [1], Zhiyang Guo [1] and Rui Zhang [1]**

[1] School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China; 181560057@st.usst.edu.cn (H.W.); hufzh@ahstu.edu.cn (F.H.); 171560051@st.usst.edu.cn (B.Z.); 181560055@st.usst.edu.cn (Z.G.); 1819660216@st.usst.edu.cn (R.Z.)
[2] School of Physics and Electronic Engineering, Fuyang Normal University, Fuyang 236037, China
[3] School of Electrical and Electronic Engineering, Anhui Science and Technology University, Bengbu 233100, China
* Correspondence: huangyingping@usst.edu.cn; Tel.: +86-21-65110651
† Co-first authors.

**Abstract:** Motion estimation is crucial to predict where other traffic participants will be at a certain period of time, and accordingly plan the route of the ego-vehicle. This paper presents a novel approach to estimate the motion state by using region-level instance segmentation and extended Kalman filter (EKF). Motion estimation involves three stages of object detection, tracking and parameter estimate. We first use a region-level segmentation to accurately locate the object region for the latter two stages. The region-level segmentation combines color, temporal (optical flow), and spatial (depth) information as the basis for segmentation by using super-pixels and Conditional Random Field. The optical flow is then employed to track the feature points within the object area. In the stage of parameter estimate, we develop a relative motion model of the ego-vehicle and the object, and accordingly establish an EKF model for point tracking and parameter estimate. The EKF model integrates the ego-motion, optical flow, and disparity to generate optimized motion parameters. During tracking and parameter estimate, we apply edge point constraint and consistency constraint to eliminate outliers of tracking points so that the feature points used for tracking are ensured within the object body and the parameter estimates are refined by inner points. Experiments have been conducted on the KITTI dataset, and the results demonstrate that our method presents excellent performance and outperforms the other state-of-the-art methods either in object segmentation and parameter estimate.

**Keywords:** motion estimation; autonomous driving; region-level segmentation; extended Kalman filter

check for updates

**Citation:** Wei, H.; Huang, Y.; Hu, F.; Zhao, B.; Guo, Z.; Zhang, R. Motion Estimation Using Region-Level Segmentation and Extended Kalman Filter for Autonomous Driving. *Remote Sens.* **2021**, *13*, 1828. https://doi.org/10.3390/rs13091828

Academic Editor: Hemanth Venkateswara

Received: 10 February 2021
Accepted: 3 May 2021
Published: 7 May 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Research on autonomous vehicles is being in the ascendant [1–3]. Autonomous Vehicles are cars or trucks that operate without human drivers, using a combination of sensors and software for navigation and control [4]. Autonomous vehicles require not only detecting and locating moving objects but also knowing their motion state relative to the ego-vehicle, i.e., motion estimation [5–7]. Motion estimation is a benefit to predict where other traffic participants will be at a certain period of time, and accordingly plan the route of the ego-vehicle. In this work, we propose a novel approach to estimate the motion state for autonomous vehicles by using region-level segmentation and Extended Kalman Filter (EKF).

Motion estimation involves three stages of object detection, tracking, and estimate of motion parameters including position, velocity, and acceleration in three directions. Accurate object detection is crucial for the high quality of motion estimation because the late two stages rely on the points within the object region; that is, only the points exactly within the object region can be used for tracking and parameter estimate. Existing

works on motion estimation such as Refs. [8–15] normally generate bounding boxes as object proposals for the late two stages. One inherent problem of these methods is that the bounding boxes contain substantial background points as shown in Figure 1. These points are noise points and will result in unreliable object tracking and incorrect parameter estimate. To address this issue, we adopt two strategies: (1) Instead of bounding boxes, we use segmented object regions as object proposals. We employ the YOLO-v4 detector [16] to generate object bounding boxes and apply a region-level segmentation on them to accurately locate object contour and determine points within the objects (Figure 1 shows the results). (2) We compose an edge-point constraint on the feature points and apply the random sample consensus (RANSAC) [17] algorithm to eliminate outliers of tracking points so that the points used for tracking are ensured within the object body and the parameter estimate are refined by inner points. By the above processing, we can obtain a high-quality point set for tracking and parameter estimate, thereby generating accurate motion estimation.

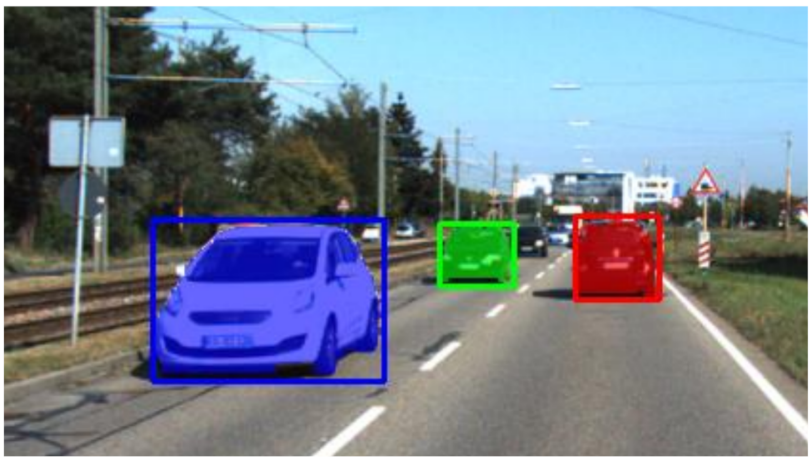


**Figure 1.** An illustration of bounding boxes including objects/background points. Three bounding boxes are detected, each of which contains background points and an object. The object regions are accurately segmented by blue, red, and green masks generated by our region-level segmentation. The pixels within the masks are used as feature points for tracking and parameter estimates.

Other aspects affecting motion estimation are how to establish the motion model for tracking and how to optimize the parameter estimate. In this work, we use optical flow to track the feature points. We propose a relative motion model of the ego-vehicle and moving objects, and accordingly establish an EKF model for point tracking and parameter estimate. The EKF model takes the ego-motion into considerations and integrates optical flow, and disparity to generate optimized object position and velocity.

In summary, we propose a novel framework for motion estimation by using region-level segmentation and Extended Kalman Filter. The main contributions of the work are:

- A region-level segmentation is proposed to accurately locate object regions. The proposed method segments object from a pre-generated candidate region, and refines it by combining color, temporal (optical flow), and spatial (depth) information using super-pixels and Conditional Random Field.
- We propose a relative motion model of the ego-vehicle and the object, and accordingly establish an EKF model for point tracking and parameter estimate. The EKF model integrates the ego-motion, optical flow, and disparity to generate optimized motion parameters.
- We apply edge-point constraint, consistency constraint, and the RANSAC algorithm to eliminate outliers of tracking points, thus ensuring that the feature points used for tracking are within the object body and the parameter estimates are refined by inner points.

- The experimental results demonstrate that our region-level segmentation presents excellent segmentation performance and outperforms the state-of-the-art segmentation methods. The motion estimation experiments confirm the superior performance of our proposed method over the state-of-the-art approaches in terms of the root mean squared error.

The remainder of this paper is organized as follows: Section 2 briefly introduces the relevant works. Section 3 describes the details of the proposed method including object detection and segmentation, and tracking and parameter estimate. The experiments and results are presented and discussed in Section 4. Section 5 concludes the paper.

## 2. Related Work

Motion estimation involves three stages of object detection, tracking, and estimate of motion parameters. The third stage that is served by the first two stages is the core of the whole pipeline. Thus, we divide the existing works on motion estimation into three categories in terms of the parameter estimates method, i.e., Kalman filter (KF)-based, camera ego-motion-based, and learning-based method.

The Kalman filter is an optimal recursive data processing algorithm that improves the accuracy of state measurement by fusion of prediction and measurement values. The KF-based method [10–12,18–20] generates optimized motion parameters by iteratively using a motion state equation for prediction and a measurement equation for updating. During the iteration, estimation error covariance is minimized. Lim, et al. [10] proposed an inverse perspective map-based EKF to estimate the relative velocity via predicting and updating the motion state recursively. The stereovision was used to detect moving objects, and the edge points within the maximum disparity region were extracted as the feature points for tracking and parameter estimate. Liu, et al. [11] combined Haar-like intensity features of the car-rear shadows with additional Haar-like edge features to detect vehicles, adopted an interacting multiple model algorithm to track the detected vehicles and utilized the KF to update the information of the vehicles including distances and velocities. Vatavu, et al. [12] proposed a stereo vision-based approach for tracking multiple objects in crowded environments. The method relied on measurement information provided by an intermediate occupancy grid and on free-form object delimiters extracted from this grid. They adopted a particle filter-based mechanism for tracking, in which each particle state is described by the object dynamic parameters and its estimated geometry. The object dynamic properties and the geometric properties are estimated by importance sampling and a Kalman Filter. Garcia, et al. [18] presented a sensor fusion approach for vehicle detection, tracking, and motion estimation. The approach employed an unscented Kalman filter for tracking and data association (fusion) between the camera and laser scanner. The system relied on the reliability of laser scanners for obstacle detection and computer vision technique for identification. Barth and Franke [19] proposed a 3-D object model by fusing stereovision and tracked image features. Starting from an initial vehicle hypothesis, tracking and estimate are performed by means of an EKF. The filter combines the knowledge about the movement of the object points with the dynamic model of a vehicle. He, et al. [20] applied an EKF for motion tracking with an iterative refinement scheme to deal with observation noise and outliers. The rotational velocity of a moving object was computed by solving a depth-independent bilinear constraint, and the translational velocity was estimated by solving a dynamics constraint that reveals the relation between scene depth and translational motion.

The camera ego-motion-based method [9,13,14,21] derives motion states of moving objects from camera ego-motion and object motion information relative to the camera. It generally consists of two steps: the first step is to obtain the camera's ego-motion, and the second step is to estimate the object's motion state by fusing the camera's ego-motion with other object's motion cues (such as relative speed, optical flow, depth, etc.). Kuramoto, et al. [9] obtained the camera ego-motion from the Global Navigation Satellite System/Inertial Measurement Unit. A framework using a 3-D camera model and EKF was

designed to estimate the object's motion. The output of the camera model was interlay utilized to calculate the measurement matrix of the EKF. The matrix was designed to map between the position measurement on the objects in the image domain and the corresponding vector state in the real world. Hayakawa, et al. [13] predicted 2D flow by PWC-Net and detected the surrounding vehicles' 3D bounding box using a multi-scale network. The ego-motion was extracted from the 2D flow using projection matrix and ground plane corrected by depth information. A similar approach was used for the estimation of the relative velocity of surrounding vehicles. The absolute velocity was derived from the combination of the ego-motion and the relative velocity. The position and orientation of surrounding vehicles were calculated by projecting the 3D bounding box into the ground plane. Min and Huang [14] proposed a method of detecting moving objects from the difference between the mixed flow (caused by both camera motion and object motion) and the ego-motion flow (evoked by the moving camera). They established the mathematical relationship between optical flow, depth, and camera ego-motion. Accordingly, a visual odometer was implemented for the estimation of ego-motion parameters by using ground points as feature points. The ego-motion flow was calculated from the estimated ego-motion parameters. The mixed flow was obtained from the correspondence matching between consecutive images. Zhang, et al. [21] presented a framework to simultaneously track the camera and multiple objects. The 6-DoF motions of the objects, as well as the camera, are optimized jointly with the optical flow in a unified formulation. The object velocity was calculated using the rotation and translation part of the motion of points in the global reference frame. The proposed framework detected moving objects via combining Mask R-CNN object segmentation [22] and scene flow, and tracked them over frames using optical flow.

Different from the first two categories of the methods, the learning-based method [8,15,23,24] does not require a specific mathematical estimation model but relies on ma-chine learning and the ability of neural network regression to estimate the motion parameters. Jain, et al. [8] used Farneback's algorithm to calculate optical flow and the DeepSort algorithm to track vehicles detected from the YOLO-v3. The optical flow and the tracking information of the vehicle were then treated as input for two different networks. The features extracted from the two networks were stacked to create a new input for a lightweight Multilayer Perceptron architecture which finally predicts positions and velocities. Cao, et al. [15] presented a network for learning motion parameters from stereo videos. The network masked object instances and predicted specific 3D scene flow maps, from which the motion direction and speed for each object can be derived. The network took the 3D geometry of the problem into account which allows it to correlate the input images. Kim, et al. [23] developed a deep neural network that exploits different levels of semantic information to perform the motion estimation. The network used a multi-context pooling layer that integrates both object and global features, and adopt the cyclic ordinal regression scheme using binary classifiers for effective motion classification. In the detection stage, they ran the YOLO-v3 detector to obtain the bounding boxes. Song, et al. [24] presented an end-to-end deep neural network for estimation of inter-vehicle distance and relative velocity. The network integrated multiple visual clues provided by two time-consecutive frames, which include deep feature clue, scene geometry clue, as well as temporal optical flow clue. It also used a vehicle-centric sampling mechanism to alleviate the effect of perspective distortion in the motion field.

Moving object detection is a prerequisite for motion estimation. Most of the existing methods use bounding boxes as object proposals which affect the accuracy of the motion estimation for the late two stages. In this study, we leverage a region-level segmentation to accurately locate object regions for tracking and parameter estimate. Therefore, we review here relevant segmentation works compared with our segmentation methods. PSPNet [25] is a pyramid scene parsing network based on the full convolution network [26], which exploits the capability of global context information by different-region-based context aggregation. PSPNet can provide a pixel-level prediction for the scene parsing task. Mask

R-CNN [22] is a classic network for object instance segmentation. It extends Faster R-CNN by adding a branch in parallel with the existing detection branch for predicting object masks. Bolya, et al. [27,28] proposed the YOLACT series, a fully convolutional model for real-time instance segmentation. YOLACT series break instance segmentation into two parallel subtasks, generating a set of prototype masks and predicting per-instance mask coefficients, to achieve compromise of segmentation quality and computation efficiency.

## 3. Method

The framework of the proposed method is shown in Figure 2. The main idea is to accurately determine feature points within the object through instance segmentation and predict the motion state by tracking the feature points through an EKF. The method includes two stages: (1) object segmentation, (2) tracking and motion estimate.

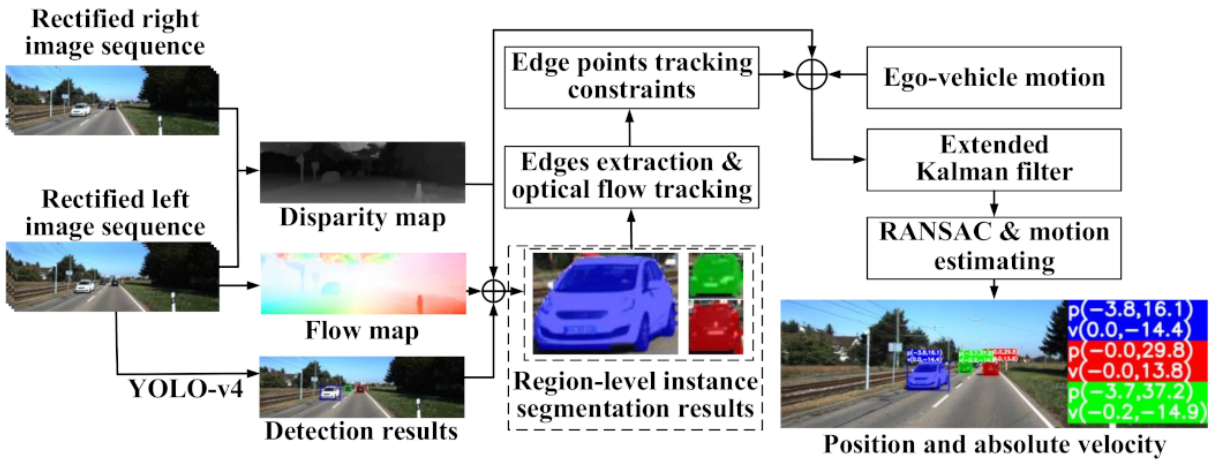



**Figure 2.** The framework of the proposed method. $P(\ ,\ )$ represents the lateral and longitudinal distances of objects in the camera coordinates, and $V(\ ,\ )$ denotes the lateral and longitudinal absolute velocities.

In the first stage, we use the YOLO-v4 detector to locate the object region in a form of a bounding box, and then extract accurate object contour through a region-level segmentation. The output is the feature points exactly within the object body.

In the second stage, we compose an edge-point constraint to further refine the feature points. We use Optical Flow to track the refined feature points. We propose a relative motion model with respect to the ego-vehicle and a moving object, and accordingly establish an EKF model for parameter estimation. We also apply the random sample consensus (RANSAC) algorithm to eliminate outliers of the tracked points. The EKF model integrates the ego-motion, optical flow, and disparity to generate optimized object position and velocity.

### *3.1. Object Detection and Region-Level Segmentation*

Object detection is to locate the object region while segmentation is to determine foreground pixels (the object body) within the region. Figure 3 shows a process of object detection and segmentation.

We employ a YOLO-v4 detector to locate the object region. The details of YOLO-v4 can be found in Reference [16]. The detection result is in a form of a bounding box contains background, as shown in Figure 3a.

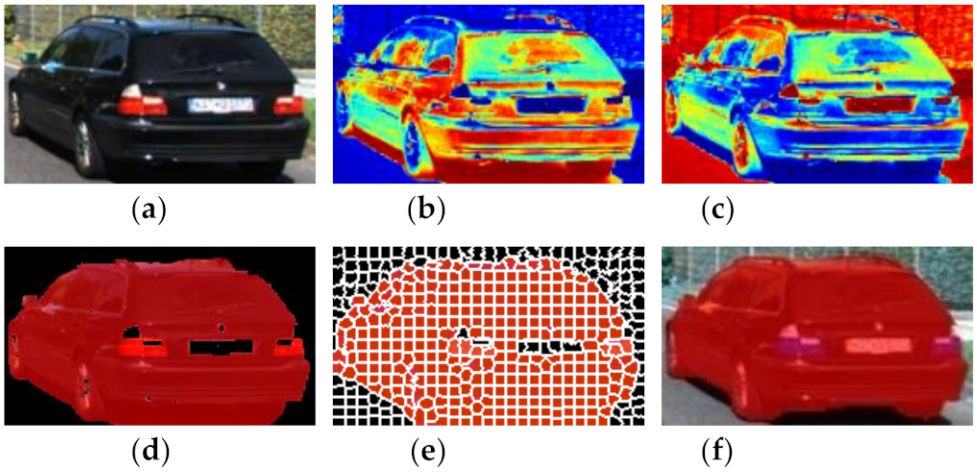



**Figure 3.** An illustration of object/background confidence maps and segmentation results. (**a**) Bounding box detected by YOLO-v4. (**b**) Foreground confidence map. Red represents higher confidence value while blue represents lower value. (**c**) Background confidence map. (**d**) Segmentation result generated by GrabCut. (**e**) Super-pixels generated by Simple Linear Iterative Clustering. (**f**) Segmentation result after applying CRF.

Region-level segmentation consists of three stages including Grabcut, Super-pixels, and Super-pixels fixed by Conditional Random Field. Starting from the bounding box (Figure 3a) detected by YOLO-v4, we apply the GrabCut algorithm to segment foreground from background. GrabCut algorithm proposed in Ref. [29] is an interactive method that segments images according to texture and boundary information. When using GrabCut, we initially define the inner of the bounding box as foreground and the external as background, and accordingly build a pixel-level Gaussian Mixture Model to estimate the texture distribution of foreground/background. By an iterative process until convergence, we can obtain the confidence maps of the foreground and background. The results are shown in Figure 3b,c.

Accordingly, GrabCut assigns a label $(\gamma_{uv})$ to pixel $(u, v)$ as follows:

$$\gamma_{uv} = \begin{cases} 1, & \text{if } (u,v) \text{ is foreground} \\ 0, & \text{if } (u,v) \text{ is background} \end{cases} \tag{1}$$

The result is shown in Figure 3d in which the background is marked as black and the foreground is marked as red. This is a pre-segmentation process with some significant errors, for example, the license plate in Figure 3d is excluded from the car body.

We refine the pre-segmentation in virtue of Super-pixels idea proposed in Reference [30]. Super-pixels are an over-segmentation formed by grouping pixels based on low-level image properties including color, brightness, etc. Super-pixels provide a perceptually meaningful tessellation of image content, and naturally preserve the boundary of objects, thereby reducing the number of image primitives for subsequent segmentation. We adopt Simple Linear Iterative Clustering (SLIC) [31] to generate $M$ super-pixels. SLIC is a simple-minded and easy-to-implement algorithm. It transforms the color image to *CIELAB* color space, constructs the distance metric based on coordinates and *L/A/B* color components, and adopts the k-means clustering approach to efficiently generate super-pixels. The label $\vartheta_{s_\beta}$ of a super-pixel $s_\beta$ is marked by Equation (2):

$$\vartheta_{s_\beta} = \begin{cases} 1, & \sum\limits_{(u,v)\in s_\beta} \gamma_{uv} \geq \frac{num}{2} \\ 0, & \text{other} \end{cases} \tag{2}$$

where $num$ is the total number of pixels within super-pixel $s_\beta$. The generated super-pixels are shown in Figure 3e where the white lines partition the super-pixels. Super-pixels can greatly reduce computation load in the late stages.

Conditional Random Field (CRF) [32] is a discriminative probability model and is often used in pixel labeling. Supposing the output random variable constitutes a Markov random field, CRF is the extension of the maximum entropy Markov model. Since the labels of super-pixels can be regarded as such a random variable, we can use CRF to model the labeling problem. We define the CRF as an undirected graph with super-pixels as nodes. It can be solved through an approximate graph inference algorithm by minimizing an energy function. The energy function generally contains a unary potential and a pairwise potential. The unary potential is only related to the node itself and determines the likelihood of the node to be labeled as a class. The pairwise potential describes the interactions between neighboring nodes, and is defined as similarity between them. In this work, we employ CRF to fix the labels of super-pixels generated in Figure 3e. Two super-pixels are considered as neighbors if they share an edge in image space. Let $s_\beta$ and $s_j$ ($\beta,\ j = 1, 2, \cdots, M$) be neighboring super-pixels, the CRF energy function is defined as

$$E_{seg}(\Theta) = \sum_{s_\beta} \varnothing_u\left(s_\beta, \vartheta_{s_\beta}\right) + \sum_{(s_\beta, s_j)\in\varepsilon} \varnothing_p\left(s_\beta, s_j\right) \tag{3}$$

where $\varepsilon$ denotes the set of all neighboring super-pixels. $\vartheta_{s_\beta}$ is the initial super-pixel label assigned in Equation (2). $\Theta$ represents the $1/0$ labeling of super-pixels. The energy function is minimized by using graph cuts algorithm. We refer readers to [33] for a detailed derivation of the minimization algorithm.

The unary potential $\varnothing_u\left(s_\beta, \vartheta_{s_\beta}\right)$ in Equation (3) measures the cost of labeling $s_\beta$ with $\vartheta_{s_\beta}$:

$$\varnothing_u\left(s_\beta, \vartheta_{s_\beta}\right) = \begin{cases} -log\left(COF_{fg}\left(s_\beta\right)\right), & if\ \vartheta_{s_\beta} = 1 \\ -log\left(COF_{bg}\left(s_\beta\right)\right), & if\ \vartheta_{s_\beta} = 0 \end{cases} \tag{4}$$

where $COF_{fg}\left(s_\beta\right)$ denotes the probability that $s_\beta$ belongs to the foreground, computed by averaging the foreground confidence scores (Figure 2b) over all pixels in $s_\beta$. $COF_{bg}\left(s_\beta\right)$ is the probability that $s_\beta$ belongs to the background.

The pairwise potential $\varnothing_p\left(s_\beta, s_j\right)$ in Equation (3) describes the interaction relationship between two neighboring super-pixels. $\varnothing_p\left(s_\beta, s_j\right)$ incorporates the pairwise constraint by combining color similarity, the mean optical flow direction similarity and the depth similarity between $s_\beta$ and $s_j$. $\varnothing_p\left(s_\beta, s_j\right)$ is defined as

$$\begin{cases} \varnothing_p\left(s_\beta, s_j\right) = \lambda \mathbf{1}\left(\vartheta_{s_\beta} \neq \vartheta_{s_j}\right) \cdot D_{lab}\left(s_\beta, s_j\right) \cdot D_{flow}\left(s_\beta, s_j\right) \cdot D_{depth}\left(s_\beta, s_j\right) \\ D_{lab}\left(s_\beta, s_j\right) = 1 / \left(1 + \left\| lab\left(s_\beta\right) - lab\left(s_j\right)\right\|_2\right) \\ D_{flow}\left(s_\beta, s_j\right) = FL_{s_\beta} FL_{s_j} / \left(\left\|FL_{s_\beta}\right\|_2 \left\|FL_{s_j}\right\|_2\right) \\ D_{depth}\left(s_\beta, s_j\right) = \sum \sqrt{hist_{s_\beta} \times hist_{s_j}} \end{cases} \tag{5}$$

where $\lambda$ is the weight used to adjust the pairwise potential function in $E_{seg}$. $\mathbf{1}(\cdot)$ is an indicator function: if the input condition is true, the output is 1; otherwise, the output is 0. $\|\cdot\|_2$ denotes the $L$ 2-norm. $D_{lab}\left(s_\beta, s_j\right)$ defines the color similarity between $s_\beta$ and $s_j$. $lab\left(s_\beta\right)$ is computed as the average $LAB$ color of $s_\beta$ in $CIELAB$ color space. $FL_{s_\beta}$ is the mean optical flow of $s_\beta$ and $D_{flow}\left(s_\beta, s_j\right)$ represents the direction similarity between the mean flows of $s_\beta$ and $s_j$. $D_{depth}\left(s_\beta, s_j\right)$ is the depth similarity between $s_\beta$ and $s_j$, measured using the Bhattacharyya distance. $hist_{s_\beta}$ is the normalized depth histogram of $s_\beta$. It can been seen that the pairwise potential integrates color, temporal (optical flow) and spatial (depth) information as criteria for segmentation purpose. The final segmentation result is shown in Figure 3f.

### *3.2. Tracking and Parameter Estimate*

We use Optical Flow to track the feature points. We establish a relative motion model between ego-vehicle and object by taking camera ego-motion into considerations, accordingly build an EKF model for point tracking and parameter estimate. The EKF model integrates the ego-motion, optical flow, and disparity to generate optimized object position and velocity. During the tracking process, we compose an edge-point constraint to refine the feature points. During the parameter estimate, we apply the RANSAC algorithm to eliminate outliers of tracked points.

#### 3.2.1. The Relative Motion Model of the Ego-Vehicle and the Object

Figure 4 shows the relative motion model between the ego-vehicle and a moving object.

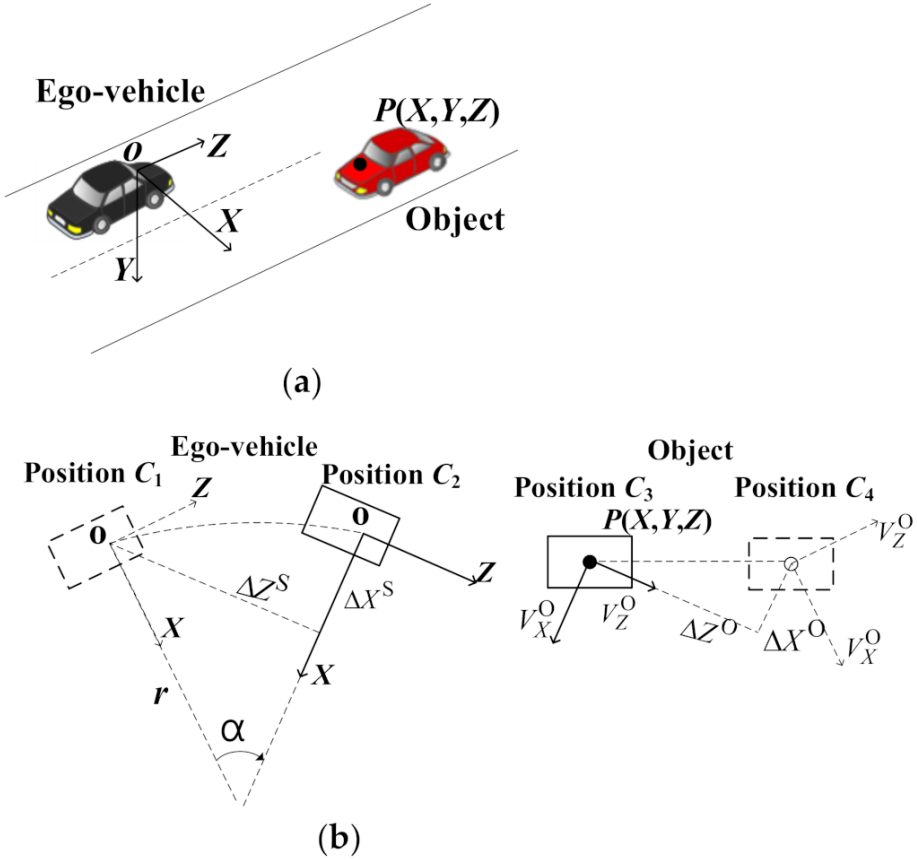



**Figure 4.** Relative motion model. (**a**) The camera coordinates. (**b**) Relative motion between ego-vehicle and moving object.

The ego-vehicle and the object move on $X$-$Z$ plane. Assuming that the ego-vehicle moves from position $C_1$ to $C_2$ within a time interval $\Delta t$ with a translational velocity $V^S = \left[V_X^S, V_Y^S, V_Z^S\right]^T$ and a rotational velocity around $Y$-axis $\omega^S$, the trajectory can be regarded as the arc $C_1C_2$ with a rotation angle $\alpha = \omega^S \times \Delta t$. The displacement $\boldsymbol{\Delta L}^S = \left[\Delta X^S, 0, \Delta Z^S\right]^T$ in the camera coordinates at position $C_2$ will be:

$$\boldsymbol{\Delta L}^S = \begin{bmatrix} \Delta X^S \\ 0 \\ \Delta Z^S \end{bmatrix} = \frac{\|\boldsymbol{V}^S\|_2}{\omega^S} \begin{bmatrix} 1 - cos\alpha \\ 0 \\ -sin\alpha \end{bmatrix} \tag{6}$$

The object $P$ is located at $C_3$ at time t, and the absolute velocities in the camera coordinates at position $C_1$ is $\boldsymbol{V}_t^O = \left[V_{Xt}^O, V_{Yt}^O, V_{Zt}^O\right]^T$. Assuming that the object moves from $C_3$ to $C_4$ with $\boldsymbol{V}_t^O$ within $\Delta t$, the absolute velocities $\boldsymbol{V}_{t+\Delta t}^O$ of $P$ at time $t + \Delta$t is related to

the change of the camera coordinates. Taking ego-vehicle motion into considerations, the displacement $\boldsymbol{\Delta L}^O$ and $\boldsymbol{V}^O_{t+\Delta t}$ of $P$ in the camera coordinates at $C_2$ are computed from:

$$\boldsymbol{\Delta L}^O = \boldsymbol{V}^O_{t+\Delta t} \times \Delta t \tag{7}$$

$$\boldsymbol{V}^O_{t+\Delta t} = \boldsymbol{R}(\alpha)\boldsymbol{V}^O_t \tag{8}$$

where $\boldsymbol{R}(\alpha)$ is the rotation matrix given by the Rodrigues rotation formula:

$$\boldsymbol{R}(\alpha) = \begin{bmatrix} cos\alpha & 0 & -sin\alpha \\ 0 & 1 & 0 \\ sin\alpha & 0 & cos\alpha \end{bmatrix} \tag{9}$$

Thus, given the coordinates of $P$ in the camera coordinates at $C_1$ at time $t$ $\boldsymbol{P}_t = [X_t, Y_t, Z_t]^T$, the coordinates of $P$ in the camera coordinates at $C_2$ at time $t + \Delta t$ $\boldsymbol{P}_{t+\Delta t}$ is calculated by:

$$\boldsymbol{P}_{t+\Delta t} = \boldsymbol{R}(\alpha) \times \boldsymbol{P}_t + \boldsymbol{\Delta L}^O + \boldsymbol{\Delta L}^S \tag{10}$$

3.2.2. Design of Kalman Filter

(1) Motion Model

The state vector for $P$ is defined as

$$\boldsymbol{SV} = \left[X, Y, Z, V^O_X, V^O_Y, V^O_Z\right]^T \tag{11}$$

where $[X, Y, Z]^T$ represents the coordinates of $P$ in the moving camera coordinates. $\left[V^O_X, V^O_Y, V^O_Z\right]^T$ is the absolute velocities of $P$ moving along the $X$-axis, $Y$-axis and $Z$-axis.

Combing Equations (6)–(8) and (10), The time-discrete motion equation for the state vector $\boldsymbol{SV}$ is given by:

$$\boldsymbol{SV}_k = \boldsymbol{A} \times \boldsymbol{SV}_{k-1} + \boldsymbol{B}_{k-1} + \delta_k \tag{12}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{R}(\alpha) & \Delta t \times \boldsymbol{R}(\alpha) \\ 0 & \boldsymbol{R}(\alpha) \end{bmatrix} \tag{13}$$

$$\boldsymbol{B}_{k-1} = \frac{\boldsymbol{V}^S_{k-1}}{\omega^S} \begin{bmatrix} 1-cos\alpha \\ 0 \\ -sin\alpha \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{14}$$

where $k$ is the time index, the process noise $\delta_k$ is considered as Gaussian white noise with a mean value of zero.

(2) Measurement Model

The measurement vector for $P$ is $\boldsymbol{MV} = [u, v, d]^T$ where $(u, v)$ is the projection, and $d$ is the disparity. The optical flow is used to track $P_k(u, v)$ at time $k$ to $P_{k+1}(u, v)$ at time $k$+1, and the corresponding disparities $d_k$ and $d_{k+1}$ can be measured from the stereovision.

According to the ideal pinhole camera model, the nonlinear measurement equation can be written as:

$$\boldsymbol{MV}_k = \mathrm{H}(\boldsymbol{SV}_k) + \epsilon_k \tag{15}$$

$$H(\boldsymbol{SV}_k) = \begin{cases} u = \frac{f_u \times X}{Z} + c_u \\ v = \frac{f_v \times Y}{Z} + c_v \\ d = \frac{b \times f_u}{Z} \end{cases} \tag{16}$$

where $\epsilon_k$ is the Gaussian measurement noise. $f_u$, $f_v$ are the camera focal lengths; $c_u$, $c_v$ are the camera centre offsets and $b$ is the camera baseline length. The Jacobian matrix of measurement equation can be expressed as

$$\boldsymbol{J} = \begin{bmatrix} \frac{f_u}{Z} & 0 & -\frac{f_u \times X}{Z^2} & 0 & 0 & 0 \\ 0 & \frac{f_v}{Z} & -\frac{f_v \times Y}{Z^2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{b \times f_u}{Z^2} & 0 & 0 & 0 \end{bmatrix} \tag{17}$$

(3) Estimation and Update

The location and absolute velocities of $P$ can be obtained by iterating the following estimation and update process. The time update equations are:

$$\boldsymbol{SV}_k^- = \boldsymbol{A} \times \boldsymbol{SV}_{k-1} + \boldsymbol{B}_{k-1} \tag{18}$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}_k \tag{19}$$

where $\boldsymbol{SV}_k^-$ is the priori estimate of the state vector $\boldsymbol{SV}$ at time $k$, $\boldsymbol{SV}_{k-1}$ is the posteriori estimate (optimal value) of the state vector $\boldsymbol{SV}$ at time $k$-1, $\boldsymbol{P}_k^-$ is the priori estimate of the variance of the estimation error, $\boldsymbol{Q}_k$ is the covariance of $\delta_k$.

The measurement update equations are

$$\boldsymbol{G}_k = \boldsymbol{P}_k^- \boldsymbol{J}_k^T \left( \boldsymbol{J}_k \boldsymbol{P}_k^- \boldsymbol{J}_k^T + \boldsymbol{W}_k \right)^{-1} \tag{20}$$

$$\boldsymbol{SV}_k = \boldsymbol{SV}_k^- + \boldsymbol{G}_k \left( \boldsymbol{MV}_k - \mathrm{H}\left( \boldsymbol{SV}_k^- \right) \right) \tag{21}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{G}_k \boldsymbol{J}_k) \boldsymbol{P}_k^- \tag{22}$$

where $\boldsymbol{G}_k$ is the Kalman gain, $\boldsymbol{W}_k$ is the covariance of $\epsilon_k$, $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{SV}_k$ is the posteriori estimate (optimal value) of the state vector $\boldsymbol{SV}$ at time $k$, and $\boldsymbol{P}_k$ is the posteriori estimate of the variance of the estimation error.

#### 3.2.3. Feature-Point Filtering

The tracking discussed in the above EKF is for a single object point. As described in Section 3.1, each segmented object consists of a cluster of points, i.e., a set of foreground pixels. For sake of tracking reliability and computation efficiency, it is essential to select reliable feature points for tracking and estimation. The motion state of an object is taken as the average of these points. Feature-point filtering is crucial for tracking and estimation.

Since the edge points have a strong textural feature and facilitate optical flow calculations, we employ the Canny operator [34] to extract the edge points as feature points. During the tracking, we compose an edge-point constraint on the tracking results. That is, the tracked points must still be edge points, otherwise, they are excluded.

Furthermore, we enhance estimation accuracy by applying the RANSAC algorithm [17] to eliminate outliers of tracking points. The RANSAC is a statistics-based hypothesis-verification method that iteratively finds the inner data from noisy data. In each iteration, a minimum number of samples is randomly selected to construct a consistency hypothesis, and other samples are verified whether they conform to the hypothesis. The samples that conform are taken as inner samples. Repeat the above steps to form a sample set with the largest number of inner samples, i.e., the maximum consensus set, for calculation of the motion parameters.

We compose a consistency constrain on the estimate results, that is, the estimate results for feature points in the same object should be consistent. In this work, the longitudinal distance and velocity, the lateral distance and velocity are used as target parameters to iteratively select the inner data set. The implementation flow for the RANSAC filtering is illustrated in Algorithm 1.

**Algorithm 1.** Implementation flow for RANSAC filtering (example of longitudinal distance).

**Input:** A set of feature points: $FR$ The maximum iterations: $I_{max}$
Consistency threshold $th$, *i.e.*, the threshold of the deviation that is the difference between longitudinal distance and its average.
**Output:** The maximum consensus set: $\prod_{max}$ The object longitudinal distance: $\zeta_{\text{final}}$

$i = 0$, $N_{max} = 0$
**while** $i < I_{max}$ **do**
**1 Hypothesis generation**
Randomly select $m$ feature points form $FR$ as minimal consensus set
Calculate the average longitudinal distance $\zeta_Z$ in the minimal consensus set
**2 Verification**
Calculate the difference between the longitudinal distance of each point in $FR$ and $\zeta_Z$, *i.e.*, deviations
Determine a set $FR_i$ whose deviations are less than $th$
Count the total number of $FR_i$ as $N$
**If** $N > N_{max}$ **then**
$\prod_{max} = FR_i$, $N_{max} = N$
**end if**
$i = i + 1$
**end while**
Calculate the average longitudinal distance in $\prod_{max}$ as $\zeta_{\text{final}}$

## 4. Experiments

Experiments have been conducted on image sequences (Road and City) of the KITTI public datasets [35]. The binocular camera settings are: baseline length 0.54 m, mounting height 1.65 m, tilt angle to the ground 0°, and rectified image resolution 375 × 1242. KITTI provides the ground truth of ego-vehicle motion, motion state of moving objects. The experiments were implemented in the workstation with an Intel Xeon Silver 4110 4 core processor, 16GB RAM, a Nvidia GeForce gtx1080ti graphic processor, and 11 GB video memory.

We use HD$^3$-flow [36] for predicting optical flow and employ PSMNet [37] to generate the disparity maps. We retrained HD$^3$-flow and PSMNet based on the original weights using the KITTI dataset.

### *4.1. Segmentation Results*

We compare our segmentation method with two state-of-the-art methods, PSPNet [25] and YOLACT++ [28]. The results of three methods compared with the ground truth are shown in Figure 5. The fourth row shows the results obtained by our method only using color information without using optical flow, and disparity, called "Our method[1]".

In the road scene, it can be seen that our method can accurately segment the Obj. 1, 2, and 3. Our method[1] fails in recognizing the license plates and lights of Obj. 2 and 3 as part of the car bodies. PSPNet wrongly mixtures Obj. 2 and 3 together while YOLACT++ wrongly mixtures distant building with Obj. 2 into one object.

In the city scene, our method also achieves the best result either in frame 4 (no-occlusion case) or frame 9 (occlusion case). Especially, our method is able to accurately distinguish Obj. 4 from the traffic light poles in frame 9. PSPNet presents significant errors either on frame 4 or frame 9 while YOLACT++ fails to segment Obj. 4 from the traffic poles. Our method[1] does not correctly segment the front windshield of Obj. 4 in frame 9, while the wheels are excluded from the car body in frame 4 and 9.

We use four metrics to quantitatively evaluate the segmentation performance.

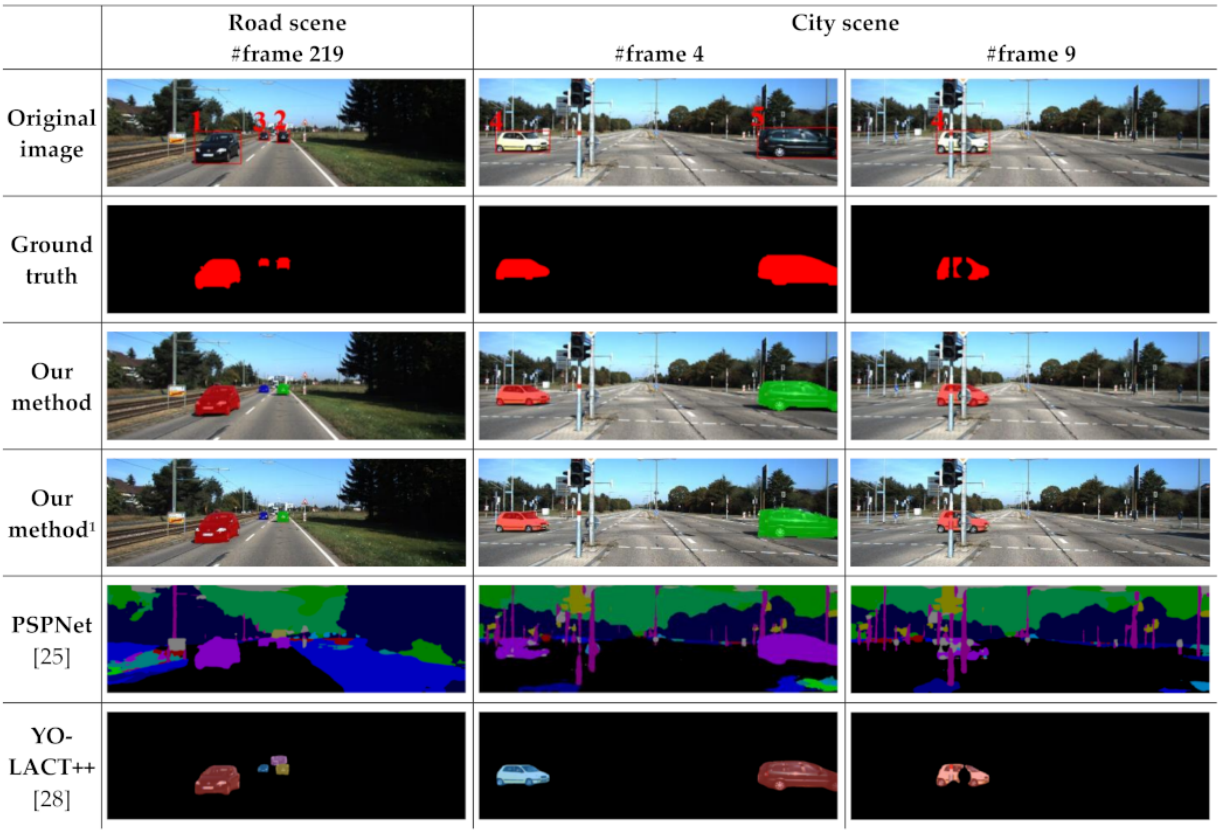



**Figure 5.** The segmentation results of three methods and ground truth in two traffic scenarios, including (1) Road scene; (2) City scene. Our method[1] indicates that our method only using color information without using optical flow, and disparity.

**Mean Intersection over Union (*MIoU*)** [38]: It computes a ratio between the intersection and the union of the ground truth and predicted segmentation.

$$\boldsymbol{MIoU} = \frac{1}{l}\sum_{l_1=0}^{1}\frac{CN_{l_1l_1}}{\sum_{l_2=0}^{1}CN_{l_1l_2}+\sum_{l_2=0}^{1}CN_{l_2l_1}-CN_{l_1l_1}} \tag{23}$$

where $l$ is the class number, in this case $l = 2$ (foreground/background). $CN_{l_1l_2}$ is the number of pixels of class $l_1$ inferred to belong to class $l_2$ and by parity of reasoning. When calculating ***MIoU***, $l_1$ and $l_2$ are regarded as foreground (1) or background (0) respectively to count the positive and negative pixels. Thus, $l_1$ and $l_2$ refers to be 1 or 0.

The False Positive Rate (***FPR***) and the False Negative Rate (***FNR***) are computed by

$$FPR = \frac{FP}{FP+TN} \tag{24}$$

$$FNR = \frac{FN}{FN+TP} \tag{25}$$

where True Positive (*TP*) and False Positive (*FP*) indicate the correctly and incorrectly segmented positive (foreground) pixels, while the True Negative (*TN*) and False Negative (*FN*) indicate the correctly and incorrectly segmented negative (background) pixels.

**Overall error (*Ov. err.*)** is the percentage of wrongly labelled pixels.

Since the KITTI doesn't provide the ground truth of the instance segmentation, we manually labeled 411 images from the Road and City sequences. We conducted experiments on those images, and the average values of the metrics are listed in Table 1. It can be seen that our method achieves the best ***MIoU*** score and the lowest ***FPR***, ***FNR***, and ***Ov.err.***, and outperforms other methods.

**Table 1.** Comparison of segmentation performance of three methods.

| | ***MIoU* (%)** | ***FPR* (%)** | ***FNR* (%)** | ***Ov.err.* (%)** |
|---|---|---|---|---|
| PSPNet [25] | 72.04 | 19.05 | 13.98 | 14.75 |
| YOLACT++ [28] | 84.37 | 11.09 | 4.57 | 7.51 |
| Our method[1] | 63.67 | 20.53 | 23.35 | 19.57 |
| Our method | 88.42 | 5.6 | 4.88 | 5.24 |

The reasons for the superior performance of our method are: (1) our method segments object from candidate region (bounding box) pre-generated by YOLO-v4 detector rather than from the whole image, which eliminates trivial information and makes segmentation easy; (2) our method combines color, temporal (optical flow), and spatial (depth) information as the basis for segmentation; and (3) super-pixels naturally preserve the boundary of objects and are computationally efficient for processing.

### *4.2. Results of Feature-Point Filtering*

As described in Section 3.2.3, the edge points within the object point cluster are used as feature points for tracking. The edge point constraint and consistency constraint are applied to filter the feature points. Taking a segmented object as an example, the filtering processing is shown in Figure 6. Figure 6a shows the point cluster obtained from our region-level segmentation, and Figure 6b shows the edge points extracted by the Canny operator, which are taken as feature points. The yellow points in Figure 6c are the feature points in Figure 6b (previous frame) that are tracked to the current frame while the white points are the edge points in the current frame. Some of the yellow points do not overlap the white points and should be eliminated. The blue points in Figure 6d are the result of excluding the non-overlapping points, i.e., satisfying the edge point constraint. The results of applying the consistency constraint on Figure 6d are shown in Figure 6e. The red points are the feature points with consistent distances and velocities that have been selected by the RANSAC, i.e., the maximum consensus set. The arrows in Figure 6f represent the optical flows of the valid feature points. It can be seen that the optical flows are identical, indicating a valid feature point selection.

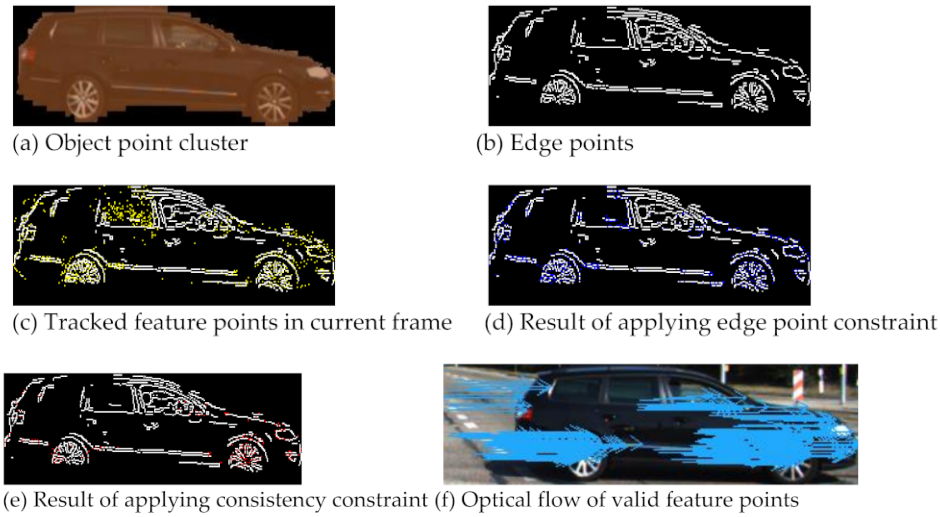



**Figure 6.** The processing of the feature-point filtering. (**a**) Object point cluster; (**b**) Edge points; (**c**) Tracked feature points in current frame; (**d**) Result of applying edge point constraint; (**e**) Result of applying consistency constraint; (**f**) Optical flow of valid feature points.

### *4.3. Motion Estimate Results and Analysis*

#### 4.3.1. Motion Estimate Results

Table 2 lists the estimates and errors of position and absolute velocity of the objects in Figure 5. It can be seen that our method presents small errors. In the Road scene, objects have little variation in their lateral positions and mainly move longitudinally. The maximum absolute errors of the objects' longitudinal position and velocity estimations are 0.4 m (Obj. 3, corresponding to the ground truth 37.5 m) and 0.6 m/s (Obj. 3, corresponding to −13.0 m/s), respectively. In the City scene, the objects are mainly moving laterally from left to right. The maximum absolute errors of the objects' lateral position and velocity estimation are −0.2 m (Obj. 4 in frame 4, corresponding to −9.9 m) and −0.3 m/s (Obj. 5 in frame 4, corresponding to 10.8 m/s), respectively.

**Table 2.** Position and absolute velocity of objects in Figure 5 and their estimate errors.

| | | **Estimates (*E*)** | | | | **Ground Truth (*GT*)** | | | | **Absolute Errors ($e = E - GT$)** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Sequence** | **Obj.** | $X_E$ **(m)** | $Z_E$ **(m)** | $V_{XE}$ **(m/s)** | $V_{ZE}$ **(m/s)** | $X_T$ **(m)** | $Z_T$ **(m)** | $V_{XT}$ **(m/s)** | $V_{ZT}$ **(m/s)** | $e_X$ **(m)** | $e_Z$ **(m)** | $e_{V_X}$ **(m/s)** | $e_{V_Z}$ **(m/s)** |
| **Road scene** (#frame 219) | 1 | −3.5 | 12.2 | −0.2 | −14.7 | −3.6 | 12.3 | −0.1 | −14.4 | 0.1 | −0.1 | −0.1 | −0.3 |
| | 2 | 0.0 | 30.5 | 0.0 | 12.7 | 0.0 | 30.8 | 0.0 | 12.9 | 0.0 | −0.3 | 0.0 | −0.2 |
| | 3 | −3.5 | 37.9 | 0.0 | −12.4 | −3.5 | 37.5 | −0.4 | −13.0 | 0.0 | 0.4 | 0.4 | 0.6 |
| **City scene** (#frame 4) | 4 | −10.1 | 15.7 | 10.7 | −2.2 | −9.9 | 15.9 | 11.0 | −2.1 | −0.2 | −0.2 | −0.3 | −0.1 |
| | 5 | 8.1 | 12.0 | 10.5 | −2.1 | 8.2 | 12.2 | 10.8 | −2.2 | −0.1 | −0.2 | −0.3 | −0.1 |
| **City scene** (#frame 9) | 4 | −4.6 | 14.7 | 11.2 | −2.2 | −4.5 | 15.0 | 11.0 | −2.1 | −0.1 | −0.3 | 0.2 | −0.1 |

Figure 7 shows the results of object segmentation and motion estimation in three frames of the Road scene sequence. $P(X, Z)$ indicates the lateral and longitudinal distances of objects in the camera coordinates, while $V(V_X, V_Z)$ denotes the lateral and longitudinal absolute velocities. Starting from frame 210 to frame 291, the red car that moves in the same direction as the ego-vehicle is tracked. At frame 210, it is 29.8 m away from the ego-vehicle with a longitudinal velocity of 13.8 m/s. At frame 210, it is getting far with a distance of 33.9 m and a velocity of 15.2 m/s. At frame 291, it is getting closer with a distance of 32.3 m and a velocity of 13.7 m/s. At the same time, other vehicles on the road (as shown in the blue, green, brown, and purple masks) are also segmented, tracked and predicated with their motion states.

#### 4.3.2. Evaluation and Comparison

We tested our method against the ground truth over a sequence of images. We evaluated our method in terms of: (1) the method with feature-point filtering (*w Ft.Pts.F.*); (2) the method without feature-point filtering (*w/o Ft.Pts.F.*). Figure 8 shows the variations of lateral distance and velocity of Obj. 4 from frame 4 to 23 in the City scene. It moves almost uniformly from left to right, the lateral distance becomes progressively larger and the lateral absolute velocity is approximately constant. Figure 9 shows the variations of longitudinal distance and velocity of Obj. 2 from frame 4 to 294 in the Road scene. It moves in the same direction as the ego-vehicle. It can also be seen that the variations of the *w Ft.Pts.F.* method are closer to the ground truth and smoother than the *w/o Ft.Pts.F.* method. This indicates that the performance of our method is improved by using feature-point filtering.

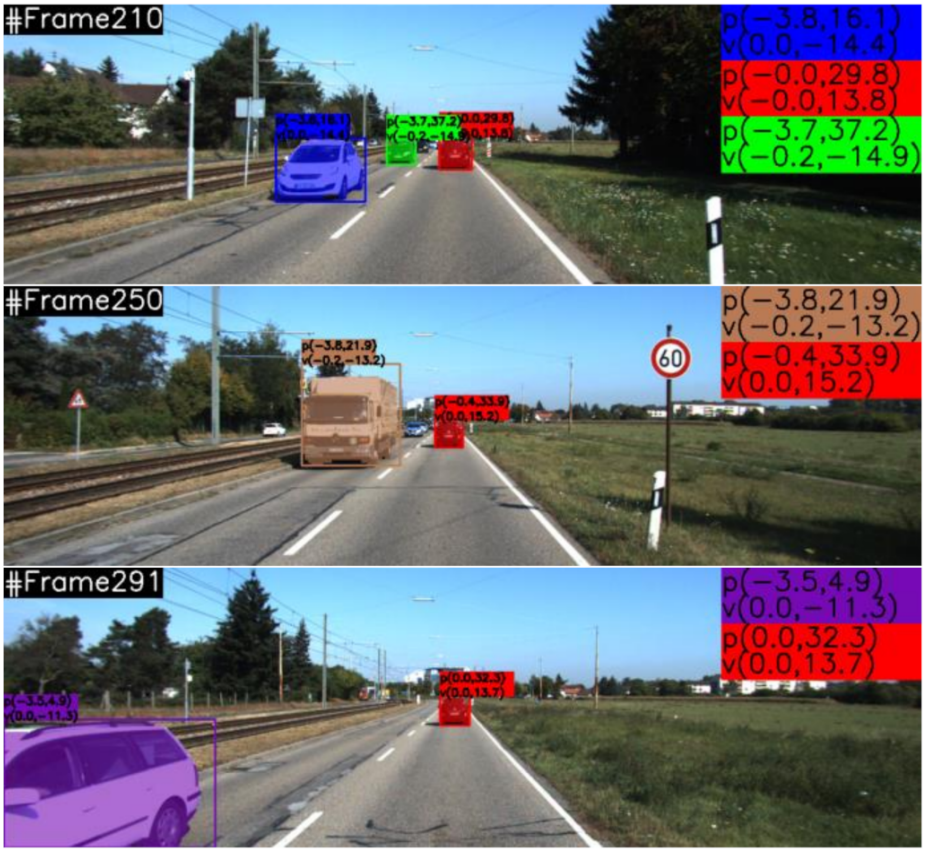



**Figure 7.** Object segmentation and motion estimate in three frames of Road scene.

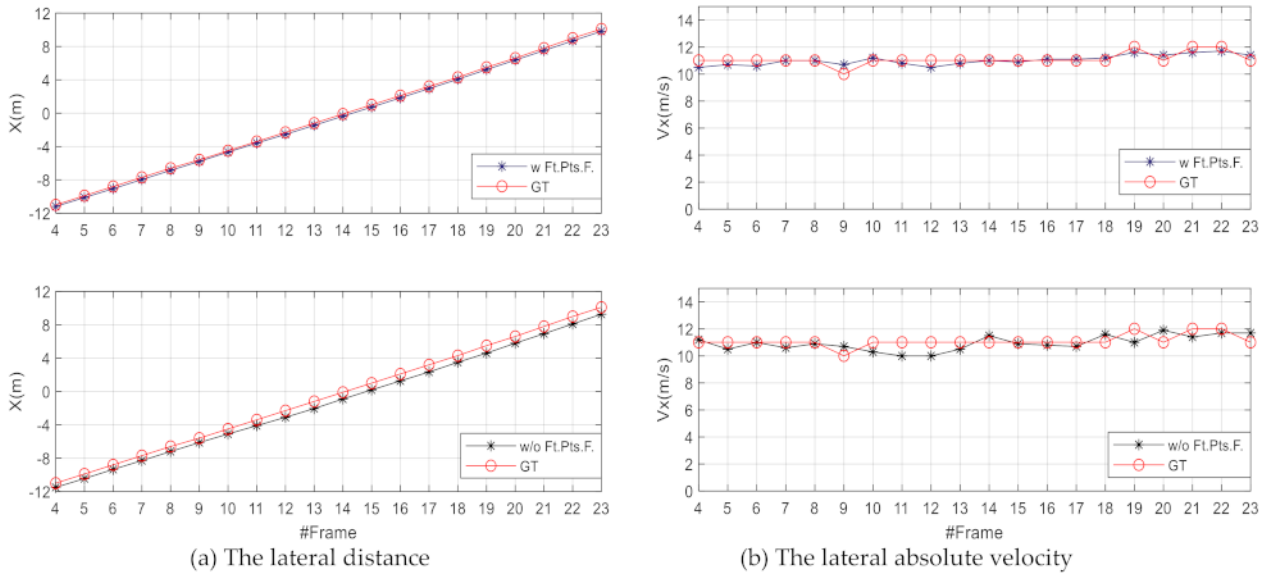
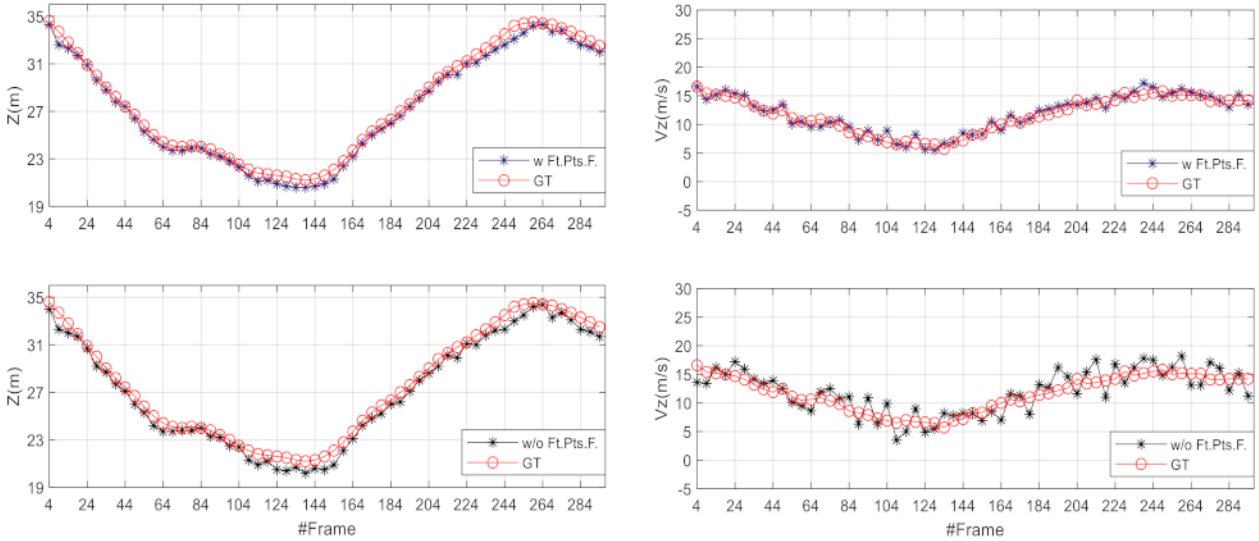



**Figure 8.** Variations of lateral distance and velocity of Obj. 4 over frame 4–23 in City scene. (**a**) The lateral distance; (**b**) The lateral absolute velocity.




(a) The longitudinal distance (b) The longitudinal lateral absolute velocity

**Figure 9.** Variations of longitudinal distance and velocity of Obj. 2 over frame 4–294 in Road scene. (**a**) The longitudinal distance; (**b**) The longitudinal lateral absolute velocity.

There is no uniform evaluation metric for object motion estimation. One of the commonly used metrics is the root mean squared error (*RMSE*) over a sequence of images. The *RMSE* is defined as

$$RMSE^{*} = \sqrt{\frac{\sum_{c=1}^{NF}(mt_{c}^{*} - gt_{c}^{*})^{2}}{NF}} \tag{26}$$

where $NF$ refers to the number of frames that at least one object is being tracked in a sequence, $mt_c$ the estimate and $gt_c$ the ground truth. $*$ represents the parameters used for evaluation including the lateral distance $X$, the lateral velocity $V_X$, the longitudinal distance $Z$ and the longitudinal velocity $V_Z$. For example, $RMSE^{X}$ is the root mean square error of the lateral distance $X$. Therefore, we compared our method with other three start-of-the-art works [13,21,24] that also used the *RMSE* as evaluation metric. Table 3 lists the comparison results.

**Table 3.** The *RMSE* comparisons of our method with the state-of-the-art methods.

| | | ***RMSE* of Distance** | | ***RMSE* of Velocity** | |
|---|---|---|---|---|---|
| | | $RMSE^{X}$ | $RMSE^{Z}$ | $RMSE^{Vx}$ | $RMSE^{Vz}$ |
| **ours** | with feature-point filtering (*w Ft.Pts.F.*) | 0.25 m | 0.51 m | 0.37 m/s | 0.91 m/s |
| | without feature-point filtering (*w/o Ft.Pts.F.*) | 0.74 m | 0.67 m | 0.6 m/s | 2.07 m/s |
| | Ref. [13] | 1.19 m | 1.7 m | 0.7 m/s | |
| | Ref. [21] | —[1] | | 1.0 m/s | |
| | Ref. [24] | 4.64 m | | 0.97 m/s | |

[1]—indicates that there is no corresponding parameter in Reference [21].

As can be seen in Table 3, our proposed method with feature-point filter brings significant improvement compared with the other methods, particularly in the *RMSE* of distance. It can also be seen the performance of our method is improved by using the feature-point filtering.

To evaluate the effect of each component in the proposed method on motion estimation, we have conducted an ablation study on different versions of the method. The results are summarized in Table 4.

**Table 4.** Quantitative comparison of different versions of our method in ablation study. *RLS* indicates our region-level segmentation method, *Ft.Pts.F.* denotes feature-point filtering method, EKF model represents the designed Kalman filter in Section 3.2.2, *w/o* means without and *w* is short for with. A tick mark indicates the corresponding term is included in the counterpart.

| ***RLS*** | | ***Ft.Pts.F.*** | | **EKF Model** | | ***RMSE* of Distance** | | ***RMSE* of Velocity** | |
|---|---|---|---|---|---|---|---|---|---|
| *w/o* | *w* | *w/o* | *w* | *w/o* | *w* | $RMSE^X$ | $RMSE^Z$ | $RMSE^{Vx}$ | $RMSE^{Vz}$ |
| | √ | | √ | | √ | 0.25 m | 0.51 m | 0.37 m/s | 0.91 m/s |
| √ | | | √ | | √ | 0.87 m | 1.64 m | 1.31 m/s | 2.27 m/s |
| | √ | √ | | | √ | 0.74 m | 0.67 m | 0.6 m/s | 2.07 m/s |
| | √ | | √ | √ | | 0.49 m | 1.13 m | 1.86 m/s | 2.72 m/s |

Comparing other rows with the first row which is the standard version, it can be seen how each component contributes to improving the *RMSE* values. Comparing the second row (using bounding box rather than region-level segmentation) with the first row demonstrates the proposed region-level segmentation method can significantly improve the results. Comparing the third row with the first row demonstrates the effect of the feature-point filtering. Comparing the fourth row with the first row demonstrates that the EKF model is effective.

## 5. Conclusions

In this work, we adopt three strategies to achieve accurate and robust motion estimation for autonomous driving. (1) Instead of bounding boxes, we use segmented object regions as object proposals for tracking and parameter estimates. We propose a region-level segmentation to accurately locate object contour and determine points within the objects. (2) We compose an edge-point constraint on the feature points and apply the random sample consensus algorithm to eliminate outliers of tracking points so that the points used for tracking are ensured within the object body and the parameter estimate are refined as inner points. 3) We develop a relative motion model of the ego-vehicle and the object, and accordingly establish an EKF model for point tracking and parameter estimate. The EKF model takes the ego-motion into considerations and integrates the ego-motion, optical flow, and disparity to generate optimized motion parameters. Substantial experiments have been conducted on the KITTI dataset, and the results demonstrate that our region-level segmentation presents excellent performance and outperforms the state-of-the-art segmentation methods. For the motion estimation, our proposed method presents a superior performance on *RMSE* compared to the other state-of-the-art methods.

**Author Contributions:** Conceptualization, H.W. and Y.H.; methodology, H.W. and Y.H.; software, H.W.; validation, H.W. and Y.H.; formal analysis, H.W.; investigation, H.W.; writing—original draft preparation, H.W. and Y.H.; writing—review and editing, H.W., Y.H., F.H., B.Z., Z.G., and R.Z.; project administration, Y.H.; funding acquisition, Y.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Shanghai Nature Science Foundation of Shanghai Science and Technology Commission, China, grant number 20ZR1437900, and National Nature Science Foundation of China, grant number 61374197.

**Data Availability Statement:** Data available in a publicly accessible repository.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Trubia, S.; Curto, S.; Severino, A.; Arena, F.; Zuccalà, Y. Autonomous vehicles effects on public transport systems. *AIP Conf. Proc.* **2021**, *2343*, 110014. [CrossRef]
2. Curto, S.; Severino, A.; Trubia, S.; Arena, F.; Puleo, L. The effects of autonomous vehicles on safety. *AIP Conf. Proc.* **2021**, *2343*, 110013. [CrossRef]

3. Arena, F.; Ticali, D. The development of autonomous driving vehicles in tomorrow's smart cities mobility. *AIP Conf. Proc.* **2018**, *2040*, 140007. [CrossRef]
4. Arena, F.; Pau, G.; Severino, A. An Overview on the Current Status and Future Perspectives of Smart Cars. *Infrastructures* **2020**, *5*, 53. [CrossRef]
5. Brummelen, J.V.; O'Brien, M.; Gruyer, D.; Najjaran, H. Autonomous vehicle perception: The technology of today and tomorrow. *Transp. Res. C Emerg. Technol.* **2018**, *89*, 384–406. [CrossRef]
6. Bersani, M.; Mentasti, S.; Dahal, P.; Arrigoni, S. An integrated algorithm for ego-vehicle and obstacles state estimation for autonomous driving. *Robot. Auton. Syst.* **2021**, *139*, 103662. [CrossRef]
7. Geng, K.; Dong, G.; Yin, G.; Hu, J. Deep Dual-Modal Traffic Objects Instance Segmentation Method Using Camera and LIDAR Data for Autonomous Driving. *Remote Sens.* **2020**, *12*, 3274. [CrossRef]
8. Jain, D.K.; Jain, R.; Cai, L.; Gupta, M.; Upadhyay, Y. Relative Vehicle Velocity Estimation Using Monocular Video Stream. In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2020), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
9. Kuramoto, A.; Aldibaja, M.A.; Yanase, R.; Kameyama, J.; Yoneda, K.; Suganuma, N. Mono-Camera based 3D Object Tracking Strategy for Autonomous Vehicles. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 459–464. [CrossRef]
10. Lim, Y.-C.; Lee, M.; Lee, C.-H.; Kwon, S.; Lee, J.-H. Improvement of stereo vision-based position and velocity estimation and tracking using a stripe-based disparity estimation and inverse perspective map-based extended Kalman filter. *Opt. Lasers Eng.* **2010**, *48*, 859–868. [CrossRef]
11. Liu, Z.; Lu, D.; Qian, W.; Ren, K.; Zhang, J.; Xu, L. Vision-based inter-vehicle distance estimation for driver alarm system. *IET Intell. Transp. Syst.* **2019**, *13*, 927–932. [CrossRef]
12. Vatavu, A.; Danescu, R.; Nedevschi, S. Stereovision-Based Multiple Object Tracking in Traffic Scenarios Using Free-Form Obstacle Delimiters and Particle Filters. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 498–511. [CrossRef]
13. Hayakawa, J.; Dariush, B. Ego-motion and Surrounding Vehicle State Estimation Using a Monocular Camera. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 2550–2556. [CrossRef]
14. Min, Q.; Huang, Y. Motion detection using binocular image flow in dynamic scenes. *EURASIP J. Adv. Signal Process.* **2016**, *2016*, 49. [CrossRef]
15. Cao, Z.; Kar, A.; Häne, C.; Malik, J. Learning Independent Object Motion From Unlabelled Stereoscopic Videos. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019), Long Beach, CA, USA, 15–20 June 2019; pp. 5587–5596. [CrossRef]
16. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. Available online: https://arxiv.org/abs/2004.10934v1 (accessed on 23 April 2020).
17. Raguram, R.; Chum, O.; Pollefeys, M.; Matas, J.; Frahm, J. USAC: A Universal Framework for Random Sample Consensus. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2022–2038. [CrossRef]
18. Garcia, F.; Martin, D.; de la Escalera, A.; Armingol, J.M. Sensor Fusion Methodology for Vehicle Detection. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 123–133. [CrossRef]
19. Barth, A.; Franke, U. Estimating the Driving State of Oncoming Vehicles From a Moving Platform Using Stereo Vision. *IEEE Trans. Intell. Transp. Syst.* **2009**, *10*, 560–571. [CrossRef]
20. He, H.; Li, Y.; Tan, J. Relative motion estimation using visual–inertial optical flow. *Auton. Rob.* **2018**, *42*, 615–629. [CrossRef]
21. Zhang, J.; Henein, M.; Mahony, R.; Ila, V. Robust Ego and Object 6-DoF Motion Estimation and Tracking. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), Las Vegas, USA, 24–29 October 2020; pp. 5017–5023.
22. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 386–397. [CrossRef] [PubMed]
23. Kim, K.; Choi, W.; Koh, Y.J.; Jeong, S.; Kim, C. Instance-Level Future Motion Estimation in a Single Image Based on Ordinal Regression. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2019), Seoul, Korea, 27 October–2 November 2019; pp. 273–282. [CrossRef]
24. Song, Z.; Lu, J.; Zhang, T.; Li, H. End-to-end Learning for Inter-Vehicle Distance and Relative Velocity Estimation in ADAS with a Monocular Camera. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2020), Paris, France, 31 May–31 August 2020; pp. 11081–11087. [CrossRef]
25. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 6230–6239. [CrossRef]
26. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [CrossRef] [PubMed]
27. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-Time Instance Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2019), Seoul, Korea, 27 October–2 November 2019; pp. 9157–9166. [CrossRef]
28. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT++: Better Real-time Instance Segmentation. *arXiv* **2019**, arXiv:1904.02689.
29. Rother, C.; Kolmogorov, V.; Blake, A. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.* **2004**, *39*, 309–314. [CrossRef]

30. Jampani, V.; Sun, D.; Liu, M.-Y.; Yang, M.-H.; Kautz, J. Superpixel Sampling Networks. In Proceedings of the European Conference on Computer Vision (ECCV 2018), Munich, Germany, 8–14 September 2018; pp. 352–368. [CrossRef]
31. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef]
32. Wei, L.; Yu, M.; Zhong, Y.; Zhao, J.; Liang, Y.; Hu, X. Spatial–Spectral Fusion Based on Conditional Random Fields for the Fine Classification of Crops in UAV-Borne Hyperspectral Remote Sensing Imagery. *Remote Sens.* **2019**, *11*, 780. [CrossRef]
33. Boykov, Y.; Veksler, O.; Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 1222–1239. [CrossRef]
34. Yuan, W.; Zhang, W.; Lai, Z.; Zhang, J. Extraction of Yardang Characteristics Using Object-Based Image Analysis and Canny Edge Detection Methods. *Remote Sens.* **2020**, *12*, 726. [CrossRef]
35. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
36. Yin, Z.; Darrell, T.; Yu, F. Hierarchical Discrete Distribution Decomposition for Match Density Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019), Long Beach, CA, USA, 15–20 June 2019; pp. 6037–6046. [CrossRef]
37. Chang, J.; Chen, Y. Pyramid Stereo Matching Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018), Salt Lake City, UT, USA, 18–23 June 2018; pp. 5410–5418. [CrossRef]
38. Alberto, G.-G.; Sergio, O.-E.; Sergiu, O.; Víctor, V.M.; Pablo, M.G.; Jose, G.-R. A survey on deep learning techniques for image and video semantic segmentation. *Appl. Soft Comput.* **2018**, *70*, 41–65. [CrossRef]