SmartDashCam: Automatic Live Calibration for DashCams

Gopi Krishna Tummala, Tanmoy Das, Prasun Sinha and Rajiv Ramnath
 {tummala.10, das.167, sinha.43, ramnath.6}@osu.edu
 The Ohio State University

ABSTRACT

Dashboard camera installations are becoming increasingly common due to various Advanced Driver Assistance Systems (ADAS) based services provided by them. Though deployed primarily for crash recordings, calibrating these cameras can allow them to measure real-world distances, which can enable a broad spectrum of ADAS applications such as *lane-detection*, *safe driving distance estimation*, *collision prediction*, and *collision prevention*. Today, dashboard camera calibration is a tedious manual process that requires a trained professional who needs to use a known pattern (e.g., chessboard-like) at a calibrated distance. In this paper, we propose SmartDashCam, a system for automatic and live calibration of dashboard cameras which always ensures highly accurate calibration values. SmartDashCam leverages collecting images of a large number of vehicles appearing in front of the camera and using their coarse geometric shapes to derive the calibration parameters. In sharp contrast to the manual process we are proposing the use of a large amount of data and machine learning techniques to arrive at calibration accuracies that are comparable to the manual process. SmartDashCam implemented using commodity dashboard cameras estimates real-world distances with mean errors of 5.7 % which closely rivals the 4.1% mean error obtained from traditional manual calibration using known patterns.

CCS CONCEPTS

• **Computer systems organization** *Sensor networks*;

ACM Reference format:

Gopi Krishna Tummala, Tanmoy Das, Prasun Sinha and Rajiv Ramnath {tummala.10, das.167, sinha.43, ramnath.6}@osu.edu. 2019. SmartDashCam: Automatic Live Calibration for DashCams. In *Proceedings of The 18th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2019)*, Montreal, QC, Canada, April 16–18, 2019 (IPSN '19), 12 pages. <https://doi.org/10.1145/3302506.3310397>

1 INTRODUCTION

With reduced cost of cameras, many vehicular manufactures and drivers are deploying dashboard cameras in vehicles [4]. The dashboard camera is a key sensor in all autonomous navigation systems being designed today. DashCams can be classified into three categories. The first category of cameras are fixed to the vehicle's body

(often to the windshield) by the manufacturer (e.g., Toyota safety sense technology [16], Subaru EyeSight [13], KIA Drive Wise [6]). The second category consists of cameras which can be bought and installed by the user [8, 18]. The third category is based on smartphone apps [11] that can transform a smartphone into a DashCam. In this paper, we focus on calibrating the three categories of DashCams to support a wide range of vehicle safety applications.

Figure 1: Manual calibration patterns used today [9].

DashCam calibration is an essential step for emerging Advanced Driver Assistance Systems (ADAS) applications such as Forward Collision Warning (FCW) [8, 18], Lane Departure Warning (LDW), Pedestrian and Cyclist Detection and Collision Warning (PCW) [10]. It can also be used for parking assistance. Views from multiple calibrated cameras can also be used to synthesize a bird's eye view of the vehicle (feature already available in some vehicles such as Audi [1] and Mercedes [7]). With calibrated DashCams, different real-world distances on the road can be measured which enables geotagging of events (such as accidents), and creation and maintenance of 3D maps. [29] exploits the calibration of the camera to map free parking spaces. Essentially, DashCam calibration is a crucial step upon which several ADAS applications depend on.

DashCams must be checked for calibration errors and recalibrated continuously. Calibration errors will translate to catastrophic safety issues in different ADAS applications. New DashCam installations, windshield installations [2], collisions, blown airbags [5], installation of portable DashCams, and manual placement of smartphone based DashCams on the smartphone holder are some events that necessitate periodic recalibration of the dashboard camera. Additionally, Continuous vehicular movements may also reorient the camera and possibly change its position.

Camera calibration involves estimating two types of camera parameters, viz., the intrinsic parameters such as focal length and distortion matrix of the camera; and the extrinsic parameters which are orientation (represented by a rotation matrix R) and position of the camera in vehicle coordinates (T). In this paper, we focus on automatic estimation of the extrinsic parameters (also referred to as *pose estimation*) of dashboard cameras and assume that the intrinsic parameters, which are based on the camera's make/model, are known. R is a function of three Euler angles (α , β and γ) which signify the yaw, pitch, and roll of camera coordinates w.r.t the vehicle's coordinates. T has three unknowns which are translations along the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IPSN '19, April 16–18, 2019, Montreal, QC, Canada

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6284-9/19/04...\$15.00

<https://doi.org/10.1145/3302506.3310397>

three axes. However, the height of the camera h (translation along y -axis) and the angles (α , β and γ) are sufficient for measuring distances between any two points on the ground plane. So, a total of 4-unknowns (α , β , γ and h) need to be solved for calibrating a DashCam.

Estimation of camera's extrinsic parameters is challenging today as it requires manual effort. Physically measuring the orientation of the installed camera is challenging due to poor accuracy of commodity compasses as they are affected by electromagnetic properties of the environment. Cameras are calibrated today by positioning a chessboard-like pattern at a calibrated distance by a highly trained technician [15] (costs about 300-400 USD [3]). Different patterns used for calibration today are depicted in Fig. 1. Automatic and live calibration can substantially reduce manual effort and cost involved in the calibration process. This can enable a wide range of vehicle-safety applications on commodity portable DashCams and smartphones. Further, the changes in camera position and orientation can be detected on the fly and calibration parameters can be kept up-to-date for the smooth functioning of ADAS applications.

Prior works for automatic camera calibration have assumed the properties of the road and road markers to calibrate the DashCams. In [31], the length and width of the lane markers are exploited to calibrate the DashCam. [25, 43] exploit lane boundary detection algorithms and use the road width and speed of the vehicle to calibrate DashCams. [22, 41] assume the height of the camera and exploit the parallel property of the lane markers to derive the orientation of the camera. The major drawback using the properties of the road is that the vehicle needs to be aligned with the road and should have the knowledge of road dimensions and lane marker lengths. Additionally, the errors in calibration are exacerbated by detection errors. In contrast, SmartDashCam assumes only the speed of the vehicle which can be obtained from GPS or the OBD-II port.

We design and implement a system called SmartDashCam, that takes in a video snippet from a DashCam and computes its calibration parameters. Parallel lines in the vehicular coordinate system (VCS), when projected onto the camera frame, intersect at a point referred to as a vanishing point. Identifying the vanishing point along the length of the road (z -axis of VCS), can be used to estimate the orientation of the z -axis in the Camera Coordinate system (CCS). SmartDashCam derives two vanishing points along the length and width of the vehicle which suffice to estimate R . The vanishing point along the length is referred to as the forward vanishing point (FVP) and the vanishing point along the width is referred to as the lateral vanishing point (LVP). It exploits the observed motion of static feature points on the road or the roadside to derive the FVP. We present a simple pipeline to extract lines joining taillights of other vehicles to derive the LVP. However, in our experiments, we face several challenges in estimating it using the lines joining taillights as such lines are almost always parallel which is referred to as *ill-conditioned vanishing point* [35, 51]. We propose approximation techniques to derive Euler angles (α , β , and γ) in such cases. SmartDashCam uses filtering and aggregation algorithms that exploit map information smartly by identifying intervals when accurate calibration values can be derived. SmartDashCam derives the height of the camera w.r.t the ground by fitting the information derived from visual odometry to that of GPS based odometry. It uses observed

static points on the road along with the speed of the vehicle from the odometer to estimate the height of the camera.

We evaluate SmartDashCam using video feeds from twenty one DashCams which are placed at different orientations, and different lighting conditions. SmartDashCam is able to estimate the FVP with errors less than 4% and estimate the Euler angles (α , β , and γ) with mean error of 2.0 degrees. This is in comparison with MonoSLAM based approaches which have mean error of 9.7 degrees. Finally, we show that SmartDashCam is able to produce accurate calibration values with mean distance estimation errors of 5.7% while manual calibrations have mean error of 4.1%. A demonstration of SmartDashCam is available at [12].

In summary, we make the following contributions:

- Techniques to derive the rotation matrix by exploiting the relative motion and position of taillights of neighboring vehicles by smartly leveraging map information.
- Techniques to derive the height of the camera by comparing monocular visual odometry with GPS-based odometry.
- First robust automatic calibration system for dash cameras with mean distance estimation errors of 5.7%.

2 BACKGROUND

In this section, we describe the coordinate systems involved in the calibration process, the camera model, background on the camera calibration problem, potential approaches to solve the calibration problem, and the use of camera calibration in different vehicular applications.

2.1 Vehicular and Camera Coordinate Systems

Figure 2: Depiction of CCS, VCS and ACCS.

Vehicle coordinate system (VCS) is a fixed coordinate system in which real-world coordinates are measured. The origin of VCS is attached to the bottommost point of the left-front tire with the z -axis facing forward along the length of the vehicle, and the x -axis along the width of the vehicle, and the y -axis along the opposite direction of gravity as depicted in Figure 2. The Camera Coordinate System (CCS) is analogous to VCS with its x -axis along the length of the frame, and y -axis is along the width of the frame. Let us refer to the coordinate system which is collocated with CCS whose axes are parallel to VCS as the Aligned Camera Coordinate System (ACCS) (depicted in Figure 2).

2.2 Camera Model

The *pinhole camera model* [48] describes the geometric relationship between the 2D image-plane (i.e, pixel positions in an image captured by a camera) and the 3D Vehicle Coordinate System (VCS). Let the image plane be represented by the UV -plane and the VCS be represented by the XYZ space. The following equation relates the pixel position (u, v) with its respective VCS coordinates (x, y, z) as,

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (1)$$

where f_u and f_v are focal lengths of the camera along the u and v axes respectively, (c_u, c_v) represents the image center of the camera, and s is a scaling value which is dependent on the pixel position.

$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ and $\mathbf{T} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$ are the rotation matrix and

the translation vector of the camera respectively. Equation 1 can be written by separating translation and rotation matrices as,

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{M}(\mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{T}). \quad (2)$$

where $\mathbf{M}$ is camera intrinsic matrix.

2.3 The Calibration Problem

The *camera calibration problem* is the problem of estimating $\mathbf{M}$, $\mathbf{R}$, and $\mathbf{T}$. The camera matrix $\mathbf{M}$ depends only on the camera's make and model. Hence, $\mathbf{M}$ is referred to as an intrinsic camera parameter. $\mathbf{R}$ and $\mathbf{T}$ are referred to as the camera's extrinsic parameters. In our work, we focus only on the problem of estimating the DashCam's extrinsic parameters. We assume DashCam's intrinsic parameters are known.

2.4 Potential approaches

The techniques to calibrate the camera can be classified broadly in to two categories: (i) using known geometric fixtures, and (ii) using vanishing points.

(i) *Calibration using known geometric fixtures*: Camera calibration can be performed from different markers present on the road. These markers must be identified in the camera frame and also their coordinates in VCS need to be obtained. The problem of solving the translation and rotation by observing a set of points in camera frames and their respective coordinates in VCS is referred to as the *perspective-n-point* problem [38]. A minimum of four points is needed for solving this problem and this process is highly manual and error-prone. DashCam calibration is performed today by placing chessboard patterns at a calibrated distance and solving the *perspective-n-point* problem.

(ii) *Rotation matrix using vanishing points*: A set of parallel lines in the VCS when projected to the camera frame intersect at a unique point which is referred to as a *vanishing point*. Essentially, the direction of vanishing point along the z -axis (of VCS), gives the direction

VCS's z -axis in CCS. The locations of the vanishing points along any two axes can be used to derive the orientation of VCS w.r.t the CCS.

2.5 Calibration for vehicular applications

Once the camera parameters ($\mathbf{M}$, $\mathbf{R}$, and $\mathbf{T}$) are estimated, Equation 2 can be used to map between VCS and image coordinates. The height of the points belonging to the road is zero. By substituting this value in Equation 2, we can derive transformation matrix from image plane to ground plane, using which we can measure different real world distances.

3 RELATED WORK

Dashboard camera calibration: Most of the prior works in the literature have leveraged the road markers or the regular patterns present in environment to calibrate the DashCams. [22, 41] assume the height of the camera and exploit the parallel property of the lane markers to derive the orientation of the camera. The length and width of the lane markers are exploited to calibrate the camera in [31]. [25, 43] exploits lane boundary detection algorithms and use the road width and speed of the vehicle to calibrate the DashCams. [29] assumes the alignment of DashCam's axes with the vehicle and exploits road-side markers (such as stop-signs) for estimating the height of the camera. [30] employs MonoSLAM techniques to estimate the ground plane and uses lines in the scene to estimate the Euler angles w.r.t the road. [32] generates SFM (Structure from Motion [17]) 3D Map of calibration room instead of using calibration patterns for calibrating a DashCam. Generating such 3D maps itself is a laborious process involving a huge number of images [17]. Additionally, the vehicle needs to be driven to the calibration stations. In contrast, SmartDashCam attempts to remove the necessity of calibration stations by designing automatic calibration techniques. [20, 27] makes *Manhattan assumption* (imaged scene contains three orthogonal, dominant edge directions) to derive camera calibration parameters. [36] assumes periodic nature of pedestrian walking to derive the *horizon line* for camera calibration. In contrast, SmartDashCam presents techniques for continuous calibration requirements of DashCams without making assumptions about the nature of the environment.

FVP estimation: Lane marker detection techniques [19, 41] can be exploited by fitting lines across them and solving for the FVP as in [44, 47]. Lane markers might not be visible, or might not be present in all parts of the road. Therefore, prior works have employed computationally expensive scene analysis [37, 39, 40] to derive the vanishing point. However, the vanishing point detection techniques for DashCam must be of lightweight. Employing such techniques is challenging as the vehicle might not be aligned with the road/lane-markers, or the road might have curvature leading to incorrect vanishing point estimates. SmartDashCam exploits vehicular motion to derive the FVP without depending on the environmental conditions.

LVP estimation: Estimating the LVP is challenging due to the difficulty in finding appropriate lines which are parallel to the vehicle's width. [28] identifies the lines across the width of the vehicle by employing edge detection techniques to solve for the lateral vanishing

point. Identifying the lines which are parallel to the width of the vehicle from the lines obtained from the image analysis is challenging. This challenge is addressed by SmartDashCam by employing lines joining taillights. During our experiments we observed lines across the width of the vehicles (parallel lines in the world) appear parallel in the image. The problem of ill-conditioned vanishing points has been addressed by [35] using known length and width of road lane markings. SmartDashCam designs techniques to solve the calibration by using the slope of the lines along the width of the vehicles.

Traffic camera calibration: The closest work to ours is [28] where authors assume straight line motion of vehicles for computing the two vanishing points and use known average sizes of vehicles (width, height, and length) to automatically calibrate the traffic camera. AutoCalib [21] exploits deep neural networks (DNN's) for identifying key-points of neighboring vehicles and uploads a large number of images from an edge node (traffic camera) to a central server to derive accurate calibration values.

Drawbacks of prior work: The major drawback using the properties of the road is that the vehicle needs to be aligned with the road. Additionally, identifying lane markings or other geometric features automatically can be error-prone. Techniques such as [24, 28] which rely on vehicle's dimensions is challenging to implement as the DashCam might be observing different types of vehicles with varying dimensions. Techniques that exploit DNN based key-point detection algorithms [21] are computationally intensive to incorporate on a DashCam. In contrast to these works, SmartDashCam use simple red thresholding and lightweight vehicle detectors for deriving the lateral vanishing point.

4 DESIGN

This section presents the design challenges and the design details of SmartDashCam.

4.1 Overview

The SmartDashCam pipeline is depicted in Fig. 3 and has the following steps:

- The map information is analyzed to identify straight road segments which are ideal for performing calibration. Based on the trigger, video frames are processed by tracking different feature points to derive the FVP (§4.3).
- Taillights of neighboring vehicles are extracted and processed to estimate the LVP. Also, approximation techniques are proposed when these lines are almost parallel for estimating the Euler angles (§4.4).
- GPS odometry and feature tracking techniques are exploited to estimate the height of the camera. Errors incurred in this process are eliminated by identifying instances when GPS odometry is accurate (§4.5).

4.2 Challenges

SmartDashCam faces the following key challenges in designing techniques for automatic DashCam calibration:

- **Ill-conditioned LVP:** We observed for most of the DashCam installations, the line joining the taillight pairs (in the camera

frame) are almost parallel leading to erroneous estimates of LVP. This is exacerbated by the errors in taillight detection techniques.

- **Consolidating the estimates:** Vanishing points estimated by observing different road segments and neighboring vehicles (taillights) might be different. Additionally, the rotation matrix R derived from vanishing points might not give a stable estimate of the height. We observed that errors in R translate to time varying estimate for the height.
- **Errors in GPS based odometry:** SmartDashCam relies on GPS odometry to derive the height of the camera. However, we observed significant errors in GPS odometry which translated to errors in the estimated height.

4.3 Forward Vanishing point (FVP)

SmartDashCam exploits vehicular motion to derive the FVP therefore calibrates the vehicle w.r.t its body instead of the road. For the sake of simplicity, let us assume that the vehicle is moving on a straight line aligned with the length of the vehicle and the direction of the road. In the VCS, all the stationary points belonging to the road and surroundings move with the vehicle's velocity but in the opposite direction. The path traced by these points are parallel straight lines, which are parallel to the length of the vehicle. SmartDashCam tracks different feature points to derive the FVP. Fig. 4 depicts tracking of different feature points to derive the FVP.

SmartDashCam faces several errors in identifying the stationary feature points and misalignments of the vehicle's body w.r.t its direction of motion. Additionally, time-varying calibrations due to change in camera placement and orientation pose a big challenge in estimating a stable vanishing point. If the vehicle's motion is not on a straight line, then the path traced by the feature points when projected on the camera frame creates complex patterns, making vanishing point estimation challenging. SmartDashCam addresses these challenges by designing the *ComputeFwdVP Algorithm* (Algorithm 1) which identifies straight road segments for calibration, and presents statistical filters to remove outliers arising due to misalignments, tracking errors, and non-stationary points.

Identifying the straight road segments: First, the road segments which are straight are identified by analyzing the map information. With high probability, the vehicle will be navigating on this road segment by aligning with the road. We can assume the vehicle's motion is along its length. This method is implemented in Line 1.

Statistical filters to remove outliers: Once the straight road segments are identified, SmartDashCam analyzes the frames by employing feature tracking algorithms (line 3) to derive tracks K of feature points. Essentially, different distinguishing points on the road are tracked across the frames. These tracks are passed through a linear fitting module (line 9). Non-stationary points, and matching errors are some of the errors which can be eliminated by measuring the error incurred during linear fitting process. Due to matching errors, different feature tracks are concatenated as a single track, or past tracks gets interchanged. These tracks have different line equations, therefore employing linear fitting techniques, will result in a fitting error which are eliminated (line 7). The pairwise intersection points of the filtered lines are computed and the vanishing point is estimated by deriving its cluster center. This estimated vanishing

Figure 3: SmartDashCam pipeline for automatic calibration of DashCams.

Figure 4: An example snapshot of SmartDashCam while estimating the forward vanishing point.

point is compared with the past estimates and rejected if it is far by a value greater than threshold ϵ (5% of frame width is used as a threshold in our experiments). The outlier estimates caused by lane changes and sudden turns are eliminated by this filter. The vanishing point estimate is triggered based on the map information. The estimated vanishing point is then passed through an exponential moving average filter (line 15). This filtering process updates the vanishing point estimate by giving more weight to recent observations. Whenever the DashCam calibration changes, the change in vanishing point is detected and the camera is recalibrated with most recent estimate of the vanishing point.

Euler angles from FVP: The direction of the FVP is equated to the direction of z-axis, i.e, the third column of the rotation matrix [23] to solve the Euler angles as,

$$\begin{bmatrix} \cos(\alpha) \sin(\beta) \cos(\gamma) + \sin(\alpha) \sin(\gamma) \\ \sin(\alpha) \sin(\beta) \cos(\gamma) - \cos(\alpha) \sin(\gamma) \\ \cos(\beta) \cos(\gamma) \end{bmatrix} = \begin{bmatrix} k_1 \frac{(u_z - c_u)}{f_u} \\ k_1 \frac{(v_z - c_v)}{f_v} \\ k_1 \end{bmatrix} \quad (3)$$

where, (u_z, v_z) is the FVP and $k_1 = \frac{1}{\sqrt{1 + \frac{(u_z - c_u)^2}{f_u^2} + \frac{(v_z - c_v)^2}{f_v^2}}}$. In the

Equation 3 above, the RHS is known as it is based on the camera's intrinsic parameters and the estimated location of the FVP. The above equation gives two constraints, by equating first two rows from LHS with first two rows of RHS. The third row of the equation is dependent on the first two rows, since the length of directional vectors is one. Therefore, in 3 we effectively have two constraints

and three unknowns. In the next section, we present one more equation relating the lateral vanishing point and the Euler angles. With these three equations, SmartDashCam can solve for the three Euler angles.

Algorithm 1: ComputeFwdVP Algorithm

```

Input : Video Frames V
Output : Vanishing Point  $u_x, v_x$ 
1 WaitForStraightRoadSegment ()
2  $F \leftarrow \text{ComputeFeaturestoTrack}(V)$ 
3  $K \leftarrow \text{DeriveFeatureTracks}(F)$ 
4  $L \leftarrow \phi$ 
   // Lines along the feature tracks.
5  $C \leftarrow \phi$ 
   // Estimated vanishing points list.
6 for  $k \in K$  do
7    $m, c, err \leftarrow \text{LinearFit}(k)$ 
8   if  $err < \epsilon$  then
9      $\text{add}(m, c)$  to  $L$ 
10 if  $|L| \geq 2$  then
11    $I \leftarrow \text{FindPairwiseIntersections}(L)$ 
12    $C_x, C_y \leftarrow \text{ComputeCentroid}(I)$ 
13   if  $\text{dist}((C_x, C_y), (u_x, v_x)) < \epsilon_1$  or  $C = \phi$  then
14      $\text{add}(C_x, C_y)$  to  $C$ 
15    $u_x, v_x \leftarrow \text{ExpMovingAvg}(C)$ 

```

4.4 Lateral Vanishing point (LVP)

SmartDashCam estimates the LVP by identifying lines joining taillights of other vehicles traveling in the same direction when the road segment is straight. These lines are expected to be parallel to the width of the vehicle on a straight road. Also, due to relative motion of the vehicles, multiple snapshots of these lines at various relative positions can be captured. These lines are parallel w.r.t each other and the width of the vehicle. SmartDashCam estimates the vanishing point along the width of the vehicle by computing the intersection of these lines when projected to the camera frame.

The *Compute Lateral VP Slope* algorithm (Algorithm 2) identifies the taillights of neighboring vehicles and estimates the slope of the vanishing point (later on described in the section) for deriving the Euler angles. We detect red colored blobs in the image by

Algorithm 2: Compute LVP Slope

Input : Video Frames V
Output : Slope of lateral Vanishing Point $\bar{m}$

```

1 WaitForStraightRoadSegment ()
2 LineSet  $\leftarrow \phi$ 
3 for each frame in  $V$  do
4   RedImage  $\leftarrow$  RedThresholding (frame)
5   Contours  $\leftarrow$  FindContours (RedImage)
6   Centers  $\leftarrow$  FindContourCenters (Contours)
7   Boxes  $\leftarrow$  HarrVehicleDetector (frame)
8   RedLightPairs  $\leftarrow$  FindPairs (Boxes, Centers)
9   Lines  $\leftarrow$  ComputeLines (RedLightPairs)
10  add Lines to LineSet
11  $\bar{m} \leftarrow$  ComputeAvgSlope (LineSet)
```

employing red thresholding techniques. If a car is red in color then different blobs belonging to it are also detected. Once the red blobs are identified, contours are drawn (line 5) and analyzed to identify their respective centers (line 6). Tail-lights belonging to the same vehicle must be paired up in order to draw the lines joining them. DNN based annotation tools [21] are computationally expensive for performing this operation. SmartDashCam uses a lightweight Haar [34] based vehicle detector to group red blobs belonging to the same vehicle (line 8).

Solving for α , β , and γ : With the obtained vanishing points, SmartDashCam solves for the rotation matrix between camera coordinates and VCS. The direction of the LVP is equated to the direction of VCS's x-axis i.e., the first column of the rotation matrix [45] is given by,

$$\begin{bmatrix} \cos(\alpha) \cos(\beta) \\ \sin(\alpha) \cos(\beta) \\ -\sin(\beta) \end{bmatrix} = \begin{bmatrix} k \frac{(u_x - c_u)}{f_u} \\ k \frac{(v_x - c_v)}{f_v} \\ k \end{bmatrix} \quad (4)$$

where, $k = \frac{1}{\sqrt{1 + \frac{(u_x - c_u)^2}{f_u^2} + \frac{(v_x - c_v)^2}{f_v^2}}}$ [23] and α, β, γ , and (u_x, v_x) are

the Euler angles of R and the LVP (along x-axis) respectively. In the Equation 4 above, the RHS is known as it is based on the camera's intrinsic parameters and the estimated location of LVP. From the equation, we can derive α and β . The above equation equates the first column of R with the estimated LVP to solve for the Euler angles. The above equation gives two constraints, by equating first two rows from LHS with first two rows of RHS. The third row of the equation is dependent on the first two rows, since the length of directional vectors is one.

Algorithm 3: Compute Euler Angles Algorithm

Input : Video Frames V
Output : Euler angles α, β and γ

```

1 WaitForStraightRoadSegment ()
2  $(u_x, v_x) \leftarrow$  ComputeFwdVP ( $V$ )
3  $\bar{m} \leftarrow$  ComputeLateralVPSlope ( $V$ )
4 Compute  $\alpha$  using Equation 5.
5 Compute  $\beta, \gamma$  using first two rows in Equation 3.
```

Figure 5: Lines joining the taillights which are almost parallel to each other.

Erroneous LVP estimates: The parallel lines that join the taillights of the other vehicles when projected to the camera frame intersect at the LVP. The technique described for FVP can be used to estimate the LVP from the set of lines joining the taillights. But, during our experiments with different placements of the DashCam, we observed the projected lines are almost parallel leading to a large deviation in the vanishing point estimate. This is referred to as *ill-conditioned* vanishing point [35, 51]. Distance between such parallel lines is exploited in [35] to calibrate the traffic camera. However, the distance between taillight pairs is not known as the velocity of taillight pairs is not known and they can be from different vehicles. These vanishing points are usually outside the frame at a far distance. Errors in identifying the center of taillights translate to a significant error in the estimated vanishing point. Fig. 5 shows a sample output of the taillight detection pipeline where the lines joining taillights are almost parallel. We tried with different statistical techniques such as average of the estimated LVPs (AVLP). However, these techniques are very sensitive to the errors from the taillight detection pipeline. As the lines in the Fig. 5 are almost parallel, the vanishing points computed by the pairwise intersections are widely dispersed and their centroid is far from the ground truth. So the technique similar to FVP estimation, does not work here. But we observe that the line with average slope of the lines joining taillights of vehicles does pass through the ground truth vanishing point. We show that in fact this direction suffices to compute the Euler angles and is a more robust approach. Fig. 6 shows the estimated LVPs by observing pairs of taillights, the average slope of lines joining taillights, and the vanishing point estimated from the Ground Truth calibration. Using first two sub-equations of Equation 4,

$$\tan(\alpha) = \frac{v_x - c_v}{u_x - c_u} \frac{f_u}{f_v} \approx \frac{v_x}{u_x} \frac{f_u}{f_v} \approx \bar{m} \frac{f_u}{f_v} \quad (5)$$

where $\bar{m}$ is the average slope of the lines joining taillights (estimated in line 8). We can use the average slope of lines joining the taillights to approximate the slope of the vanishing point when the vanishing point is far from the camera frame center at a large distance compared to width and length of the frame. The ALVP $(\bar{u}_x, \bar{v}_x)$ is different from the ground truth location of the vanishing point (depicted in Fig. 6). $\text{mean}(\frac{v_x - c_v}{u_x - c_u} \frac{f_x}{f_y}) \neq \frac{\bar{v}_x - c_v}{\bar{u}_x - c_u} \frac{f_x}{f_y}$. We observed LVP and ALVP are inaccurate and far from LVP estimate using ground-truth calibration. Substituting these values in the above equations will give erroneous results. The Euler angles computed using the slope of the lines are more accurate as depicted in Fig. 16.

Putting it all together: The *ComputeEulerAngles* algorithm (Algorithm 3) for solving the Euler angles. This algorithm is triggered upon identification of straight road segments (line 1). Once the straight road segments are identified, the forward vanishing point estimation is triggered (line 2). Lateral vanishing point estimation process is triggered to derive the slope of the LVP (line 3). By observing the slope $\bar{m}$ we can solve for α using Equation 5 (line 4). By substituting the forward vanishing point and α in the first two sub-equations of Equation 3 the other two angles can be solved for (line 5.2).

Figure 6: Technique from Algorithm 1 fails to estimate the LVP. The intersections of lines joining taillights are widely dispersed far from the Ground Truth.

4.5 Estimating the Height of DashCam

The height of the camera is an important parameter for measuring distances on the ground. Images contain 2D information and by just observing the image information it is challenging to measure real-world distances such as height. Manual measurements are highly error-prone for estimating the height as it often requires measuring the dimension of the vehicle, height of tires, etc.

SmartDashCam solves for the height of the camera by using the visual odometry techniques on the odometer readings from GPS or OBD-II port. If the height of the camera is accurately known, employing visual odometry techniques [46] and by tracking the static feature points on the road, the distance traversed by the vehicle can be estimated. Due to large smoothing windows of these sensors, we observed that the odometer measurements are often inaccurate. SmartDashCam studies these sensors and identifies the instances where odometer sensors are accurate to perform height estimation.

Figure 7: Using velocity of the vehicle to estimate the height of the DashCam.

4.5.1 Estimating height from ground feature points. Intuition: SmartDashCam analyzes points belonging to the road/ground plane to derive the height of the camera. Let $P(u, v)$ be the feature point belonging to the road/ground plane. Essentially, pinhole

model represents a ray emanating from the camera center towards feature point P . Let the vehicle's velocity be V , which can be obtained from GPS based odometry. Since, P is stationary w.r.t ground, its velocity in VCS will be V but directed in the opposite direction of the vehicle's motion. Consider two instances of a feature point at time instances t_1 and t_2 . Fig. 7 depicts the motion of DashCam w.r.t to P . By solving the intersection of two rays separated by a distance of $V(t_2 - t_1)$, we can solve for the height of the camera, h .

Mathematical formulation: For the sake of simplicity, let us analyze the feature point P in ACCS (depicted in Fig. 2). The y-coordinate of P in ACCS is negative of the height of the camera i.e., $-h$. Since ACCS is collocated with CCS, the translation T between the two coordinate systems is a null vector. Rewriting Equation 2 as,

$$\begin{bmatrix} x \\ -h \\ z \end{bmatrix} = \mathbf{R}^{-1} \mathbf{M}^{-1} s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (6)$$

where $(x, -h, z)$ and (u, v) are coordinates of P in ACCS and pixel position respectively. Since we know one coordinate of P (y-coordinate: $-h$), we can solve for the other two coordinates in terms of h . The value of s is derived in terms of known parameters u, v , and h by observing the second row of the above equation. Using this value of s , z-coordinate which signifies the distance between P and the DashCam along the length of the road $z = d$ can be derived from the above equation as,

$$d = - \frac{\begin{bmatrix} 0 \\ h \\ 0 \\ 1 \end{bmatrix}^T (\mathbf{R}^{-1} \mathbf{M}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix})}{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}^T (\mathbf{R}^{-1} \mathbf{M}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix})}. \quad (7)$$

For the sake of simplicity, let us write the above equation as, $d = -hf(u, v)$. The distance between two pixel positions $P_1(u_1, v_1)$ and $P_2(u_2, v_2)$ can be estimated using equation 7 as,

$$d_{12} = -h(f(u_2, v_2) - f(u_1, v_1)). \quad (8)$$

where h is the height of the DashCam. If the height h of the camera w.r.t the ground plane is known, then the velocity of the vehicle can be estimated by observing the rate of change of d . However, we know that the velocity of the feature point is $-V$, where V is the velocity of the vehicle. The height of the camera can be estimated by observing P 's motion between time t_1 and t_2 from equation 8 as,

$$h = \frac{V(t_2 - t_1)}{f(u_2, v_2) - f(u_1, v_1)} \quad (9)$$

where (u_1, v_1) and (u_2, v_2) are pixel positions of P at times t_1 and t_2 , respectively. The next section analyzes the accuracy of $V(t_2 - t_1)$ during different times of driving period and presents an optimal scheme for height estimation.

4.5.2 Accuracy of $V(t_2 - t_1)$. We evaluate the accuracy of two different techniques for measuring the speed, namely using GPS and using OBD II port of the vehicle. The ground truth for this experiment is measured using a high-precision GPS device (SXBlue II [14]). The experiments are conducted using these three devices in a car. The data is based on 5 miles of driving on a state road and

Figure 8: Reactiveness of Mobile GPS and OBD

Figure 9: Accuracy of Mobile Phone and OBD-II considering SXBlue II as ground truth

12 miles of driving on a highway. The speed data is extracted from three devices and time aligned by correcting for their time offsets.

Fig. 8 shows how the speed measured by the three devices vary with time for a highway scenario. We observe that the speed measurement using the GPS as well as OBD-II lag the ground truth especially when accelerating and decelerating. The OBD-II computes the speed of the car based on the average distance traveled by the four tires and it uses a smoothing function over time [49]. Both devices use a smoothing function over time [49] which results in the observed lagging phenomenon. Figure 9 shows how the standard deviation of the distance measurement error (expressed as a percentage of the distance) changes with the measured distance. We observe that the error is lower for highways as compared to the state roads. This is due to lag in measuring speed when accelerating and decelerating, which is encountered more frequently in state roads.

SmartDashCam triggers the height estimation process whenever the velocity of the vehicle is uniform over a long time. This happens particularly on the highways compared to state roads. The points belonging to the road can be detected by different techniques. SmartDashCam employs lane marker identification techniques [19] and tracks the centers of lanes for estimating the height of the camera. Also, the ground points that are close to the vehicle are selected to derive the height of the experiments.

5 EVALUATION

Implementation: The complete SmartDashCam pipeline is implemented as a python code. All vector algebra operations are sped up using the NumPy library and OpenCV 3.2 is used for calibration computations. For vehicle detection, we used a custom trained

Haar based vehicle detection module. SmartDashCam is deployed on a laptop running on Intel core I7 CPU and 16 GB RAM. For a 1920x1080 video frame, this deployment runs at a rate of 30 fps for identifying FVP, 3 fps for estimating LVP and 30 fps for estimating the height.

In our evaluation, we analyze SmartDashCam's performance by measuring the accuracy of the final calibration estimates. We also present evaluation of different intermediate parameters involved in SmartDashCam.

Ground Truth and DataSet for Evaluation: To evaluate SmartDashCam, we performed a total of 21 experiments each of them 10-15 min long during different times of the day and light conditions. During each experiment, we recorded the video from DashCam and GPS readings. The DashCam is placed at different orientations w.r.t the vehicle facing in the forward direction. Fig. 10 shows the snapshot captured by SmartDashCam for different experiments (E1-E21) depicting the light conditions and installations w.r.t the road. E1 is taken in morning daylight conditions, E2 is tilted towards the right, E3-E8 are performed during afternoons on different days, E10-E14 are performed on cloudy days, E15 is rotated towards the right, E16 is tilted forward, E17-E20 are performed on different days during evening times, E21 is performed during bad light condition. For each experiment, the camera is installed at a random place on the windshield. A Galaxy S8 phone is used as a DashCam to record the video of the traffic and GPS traces. Intrinsic parameters of the DashCam are estimated by using chessboard images (only once) before the start of the experiments.

Manual Calibration: For each experiment, we performed manual calibration using fifteen chessboard images which are taken from a distance of 1m to 15m (1m separation). The chessboard images taken at different distances are also used for estimating the accuracy achieved by SmartDashCam for measuring distances on the ground. We also used manual calibration as ground truth for studying the performance of various intermediate parameters involved in SmartDashCam.

5.1 Accuracy of FVP

Fig. 11 shows the absolute percentage error of FVP estimated by SmartDashCam for different experiments. The absolute percentage error for the x-coordinate of vanishing point is defined as, $\left| \frac{VP_x^s - VP_x^g}{W} \right| \times 100$ where VP_x^s and VP_x^g are x-coordinates of FVP estimated by SmartDashCam and using manual calibration respectively and W is the width of the frame. Similarly, the percentage error for the y-coordinate is defined by normalizing with the height of the frame. We selected this metric as it is independent of the resolution of the image. SmartDashCam estimates the FVP with errors less than 4%. We observed the experiments performed on the straight roads and bright light conditions have less errors (E1-E10: avg. error of 1.1%) compared to other experiments (E11-E21: avg. error of 1.7%).

Number of frames processed: To measure the time required for estimating the FVP, we evaluated the performance of SmartDashCam for varying number of frames. Fig. 12 shows the percentage error (normalized with the frame width and height) of SmartDashCam

Figure 10: Experiments for evaluating SmartDashCam are collected at different light conditions, orientations, and at random places on windshield of the DashCam.

Figure 11: Absolute error in estimating the FVP for different experiments (mean of 1.5% along x-axis and 1.4 % along y-axis).

Figure 12: Error in estimating the forward vanishing point with the number of frames processed.

in estimating x-coordinate and y-coordinate of the forward vanishing point with the number of frames processed. The estimated vanishing point converges to an accurate and stable vanishing point with more frames. Fig. 13 shows the length of video required by SmartDashCam for estimating the stable FVP. SmartDashCam estimated the stable vanishing point with less than 4-min of video for all the experiments. The number of frames required for estimating a stable vanishing point depends on the map information and the number of lane changes. We observed that the computations performed on the highway roads (such as E14, E20) converge quickly (in about a minute) to the stable estimates as the vehicle's trajectories on highways are approximately straight lines.

Figure 13: Length of video processed for estimating the stable forward vanishing point (mean of 167 sec).

5.2 Accuracy of LVP and Rotation Matrix

For estimating the Euler angles, SmartDashCam uses the slope of the LVP which is derived by approximating it to the average slope of lines joining taillights. Fig. 15 shows the error incurred in estimating the slope of vanishing points by analyzing lines joining taillights. The slope from manual calibration is used as the ground truth. Multiple lights on each vehicle, multiple lights from nearby vehicles and miss-detections created outliers in this estimation process.

Using the LVP and FVP estimates, SmartDashCam estimates the yaw, pitch, and roll of VCS w.r.t the CCS. The errors of the estimates are compared using angles estimated from manual calibration as ground truth. Fig. 14 depicts the estimation error of SmartDashCam for computing the three Euler angles. The angles derived by SmartDashCam have errors less than 7 degrees. *Yaw* (α rotation around z-axis) is estimated based on the slope of LVP (equation 5), therefore its error is correlated with errors from slope estimation (depicted in Fig. 15). Errors in *Pitch* (β) and *Roll* (γ) are correlated with the FVP. This can be observed in E12-E20, where the errors in FVP are high, leading to the significant errors in Pitch and Roll.

Comparison with inertial sensors based calibration: The IMU sensors installed with the DashCam can also be exploited for calibration. The gravity vector's direction which is estimated in CCS gives the direction of VCS's Y-axis which can be used to calibrate the DashCam. It can be used along with the FVP or LVP (slope of the taillights) for calibrating the DashCam. The average LVP (ALVP) from the taillights can be used in place of LVP for calibration. Fig. 16 shows the comparison of estimating the orientation of CCS using different approaches. For these experiments, we have

Figure 14: Absolute errors in estimating pitch (α , mean of 2.8 degrees), yaw (β , mean of 1.5 degrees), and roll (γ , mean of 1.2 degrees) for different experiments.

Figure 15: Absolute errors in estimating slope of LVP by observing lines joining taillights (mean of 0.09).

Figure 16: Absolute errors in estimating pitch (α), yaw (β), and roll (γ) using IMU sensors and 8-point algorithm [33] based approach.

collected the accelerometer readings (from smartphone) when the vehicle is stationary for deriving the direction of the gravity vector.

Comparison with MonoSLAM based calibration: Epipolar geometry [50] can be exploited to solve the direction of VCS's Y-axis. The road plane can be assumed as a flat two dimensional plane, and the vehicle's motion can be modeled as a translatory and rotatory on the two dimensional plane. The rotation matrix and translation matrix between two camera views can be derived by employing 5-point algorithm [42] on the common feature points. We have used the algorithm presented in [42] for comparison. This technique is widely used in visual odometry and Mono-SLAM based applications. The vehicle's axis of rotation will be along the vertical direction (perpendicular to the road plane) which can be estimated in CCS gives the direction of VCS's Y-axis. We have employed this approach to derive the axis of rotation at the turnings where there is significant amount of rotation. This technique is applied on different video segments and the average vertical axis direction is derived based on 50 runs (on 50 different video segments) per experiment. The obtained vertical axis direction is used along with the FVP for estimating the Euler angles. Fig. 16 shows the average absolute errors of the Euler angles estimated by this approach. SmartDashCam outperforms the MonoSLAM based approaches as these approaches are effected by feature tracking errors, potholes and irregularities on the road. SmartDashCam employs simple taillight detection and associates the lights based on the vehicle detection bounding boxes.

Yaw (α) is estimated based on the direction of VCS's x-axis. SmartDashCam and LVP+gravity use accurate slope of the LVP for deriving the yaw and therefore outperform the other approaches. As mentioned in § 4.4, the slope of ALVP is different from slope of LVP, therefore using ALVP will result in errors in estimated Euler angles. Due to this ALVP+FVP has high error in estimating the yaw. Pitch (β) and Roll (γ) are based on VCS's y-axis (gravity direction) and z-axis (FVP direction). Errors from the commodity accelerometer translate to orientation errors of CCS leading to bad performance for Slope+Gravity and FVP+gravity approaches. Estimated yaw (α) is employed for calculating roll and pitch for ALVP+FVP. Errors in yaw translate to errors in pitch and roll for ALVP+FVP. Due to these factors, SmartDashCam outperforms the IMU sensors based approaches for DashCam calibration.

5.3 Accuracy of Estimated Height

We derive the height of the camera w.r.t the ground by fitting the information derived from visual odometry to that of GPS based odometry. Fig. 17 shows the errors of SmartDashCam for estimating the height of the camera. Essentially, SmartDashCam uses GPS speed readings, derived R , and feature tracks of ground points, for the derivation of height. Errors in each of these parameters will contribute to the errors observed in the estimated height. We observed the experiments performed during bright light conditions (E1-E9) have less errors (average of 14 cm).

Figure 17: Absolute errors in height estimation by SmartDashCam (mean of 0.24m).

5.4 Calibration Accuracy

This section presents the performance study of the following approaches in measuring real-world distances: (i) Manual calibration; (ii) SmartDashCam, and (iii) Only Rotation Matrix (OnlyR): The rotation matrix estimated from SmartDashCam is used along with the translation matrix T estimated from manual calibration. We used two points separated by 5-meters along the length of the vehicle to evaluate the accuracy achieved by SmartDashCam w.r.t the other

Figure 18: Absolute errors of SmartDashCam (mean of 5.7%), manual calibration (mean of 4.1%), OnlyR (mean of 11.4%) for measuring distances along the road.

techniques. Fig. 18 shows the accuracy of measuring distances using calibrations from manual calibration, OnlyR, and SmartDashCam.

Manual calibration: We observed Manual calibration has a mean error of 4.1% error.

SmartDashCam: The performance of SmartDashCam depends on the accuracy of the estimated parameters (α , β , γ and h). SmartDashCam measures the distances on the ground with mean error of 5.7 % that is 1.6 % more than manual calibration. We observed SmartDashCam outperforms onlyR even when there are errors in the estimated parameters. To explain this, consider the experiments E10-E18 where SmartDashCam outperforms OnlyR. This counter-intuitive behavior is due to the fact that the errors in estimated R and h get canceled. To understand this behavior, consider two feature points P_1 and P_2 on the ground plane along the length of the road. We want to measure distances between P_1 and P_2 . SmartDashCam uses GPS odometer readings to derive the distance between P_1 and P_2 and thus derives h . Essentially, if there is an error in R which contracts the distances on the ground, then using the same R , SmartDashCam estimates a greater value of h (E10-E14). Therefore, SmartDashCam is able to measure distances accurately despite slight orientation errors. We observed the calibration accuracy is better during afternoon light conditions (good light condition). This can be observed in E1-E10 (avg. of 5.1%) compared to E11-E21 (avg. of 6.3%)

OnlyR: For the cameras fixed by the manufacturer and other fixed mount cameras which do not have access to odometer information, OnlyR signifies the accuracy achieved by SmartDashCam. For these cameras, the height of the camera remains unchanged and the rotation matrix R needs to be estimated periodically. We observed the accuracy of OnlyR is correlated with the accuracy of the estimated FVP. The errors in the forward vanishing point will reorient the VCS w.r.t the CCS. Due to this, we observed high variance in performance of OnlyR. For experiments with bright light conditions (E1-E10), we observed the estimated R has significantly less errors leading to better performance (avg. error of 6.1%) compared to other experiments (E11-E21: avg. error of 16.2%).

Comparison with MonoSLAM based calibration: The technique described in the above section 5.2 is used to derive the Euler angles by employing the 5-point algorithm at the turnings. Figure 19 shows the performance of this approach in comparison with SmartDashCam and onlyR based approaches. This result can be expected as a few degree error in the Euler angles translate to significant errors in distances measurements. **Interpretation of results:** The 2-degree (0.03 radians) error in the Euler angles contribute to 3% error in estimated distance and the worst-case error of 7-degree

Figure 19: Absolute errors of SmartDashCam (mean of 5.7%), manual calibration (mean of 4.1%), OnlyR (mean of 11.4%) 5-point based approach (mean of 38.0%), for measuring distances along the road.

(0.12 radians) translates to 12% error. Similar numbers are observed in the experiments. With this accuracy, different safety applications such as FCW can be enabled.

6 FUTURE WORK

The following extensions are left for future work. **1. DashCalib for night light conditions:** For night light conditions, the FVP estimation must be improved while the other two modules perform better in such conditions. An ideal choice is to track the lights from the environment as most of the roads are equipped with a good number of lights which are stationary and can be easily tracked. Lights from oncoming traffic, lights from moving vehicles, and reflections on the ground act as noise for estimating the FVP. However, these can be eliminated by observing the bounding boxes of the vehicles and discarding the lights belonging to these vehicles. **2. Assisted calibration with the length of road markers:** By observing different road-segments, SmartDashCam produces a large number of calibration values. These calibration values can be analyzed to derive an accurate estimate and reject erroneous estimates. Information about the road markers/road size can be leveraged to identify accurate calibration values. **3. Calibrating other vehicular cameras:** SmartDashCam can be extended to side-mirror camera and parking camera calibration. For these cameras, the taillights are often not visible making LVP estimation challenging. However, the same techniques can be extended by exploiting front-lights during night light conditions. **4. Outlier detection algorithm for Tail-light detection:** SmartDashCam exploits taillights from other vehicles to derive the LVP. However, in our experiments, we observed multiple lights on each vehicle, multiple lights from nearby vehicles and miss-detections created outliers in the estimation process. Algorithms such as RANSAC [26] can be employed to remove these outliers. **5. Curvy roads:** SmartDashCam assumes straight road conditions for FVP and LVP estimation. For curvy road conditions, the FVP estimation produces erroneous results as the feature tracks are

not parallel w.r.t each other. The LVP estimation can be improved by identifying the vehicles which are close to DashCam as the vehicles which are separated closely are aligned w.r.t each other. Designing techniques for addressing this limitation is left for future work.

7 CONCLUSION

In this paper, we propose SmartDashCam, a system for scalable, automatic calibration of DashCams. SmartDashCam exploits the motion of the vehicle, tail-lights from the neighboring vehicles, and GPS based odometry to derive a large data-set of parallel lines along the length and width of the vehicle in the camera frame and uses a novel filtering and aggregation algorithm to automatically produce a robust estimate of the camera calibration parameters. SmartDashCam implemented using commodity DashCams estimates real-world distances with mean error of 5.7%. The mean error of manual calibration is 4.1%. SmartDashCam can replace manual calibration and enable a wide range of ADAS applications. Demo of SmartDashCam to measure distances on the ground for different DashCam videos can be found in [12]

8 ACKNOWLEDGMENTS

We would like to thank our shepherd and the anonymous reviewers of IPSN'19 for their comments and inputs towards improving the paper. This work was partially supported by NSF grant CNS 1618520.

REFERENCES

- [1] Audi top view camera. <http://www.audi.ca/ca/web/en/search-terms/Top-View-Camera.html>.
- [2] Camera calibration for windshield replacement. <https://www.thewindscreen.co.uk/news/what-is-windscreen-calibration-and-when-do-i-need-one/>.
- [3] Cost of dashboard camera calibration. <http://blog.gtsservices.com/adas-calibration>.
- [4] Dash cameras in russia. <https://www.wired.com/2013/02/russian-dash-cams/>.
- [5] DashCam recalibration. <http://www.repairerdrivennews.com/2017/12/20/honda-many-common-body-shop-situations-require-adas-recalibration/>.
- [6] KIA Drive Wise. <http://www.kia.com/eu/quality/kia-drive-wise/>.
- [7] Mercedes top view camera. <http://www.mersuv.com/mbread-326.html>.
- [8] Mobile eye dash cam for adas application. <http://www.mobileye.com/en-us/>.
- [9] Patterns for ADAS camera calibration. <https://www.texa.com/Upload/Depliant/pieghevole-adas-en-gb-v6.pdf>.
- [10] Pedestrian and Cyclist Detection and Collision Warning (PCW). <https://www.mobileye.com/en-us/technology/features/>.
- [11] Smart phone based dash camera app: Camonroad. <https://play.google.com/store/apps/details?id=com.camonroad.app&hl=en>.
- [12] Smartdashcam demo. <https://sites.google.com/site/smartdashcalib/>.
- [13] Subaru EyeSight. <https://www.subaru.com/engineering/eyesight.html>.
- [14] SX Blue high accurate GPS. <http://sxbluegps.com/product/sxblue-iplus-gps/>.
- [15] Texa adas camera calibration kit. <https://www.texa.com/products/video-camera-calibration-kit>.
- [16] Toyota smart sense technology. https://www.toyota.com/toyota-owners-theme/pdf/toyota_safety_sense.pdf.
- [17] Visual SFM. <http://ccwu.me/vsfm/>.
- [18] Xiaomi adas camera. <https://xiaomi-mi.com/news-and-actions/car-dvr-by-xiaomi-yi-with-adas-function/>.
- [19] ALY, M. Real time detection of lane markers in urban streets. In *Intelligent Vehicles Symposium, 2008 IEEE* (2008), IEEE, pp. 7–12.
- [20] ANTUNES, M., BARRETO, J. P., AOUADA, D., AND OTTERSTEN, B. Unsupervised Vanishing Point Detection and Camera Calibration from a Single Manhattan Image with Radial Distortion. In *Proc. of the IEEE CVPR* (2017), pp. 4288–4296.
- [21] BHARDWAJ, R., TUMMALA, G. K., RAMALINGAM, G., RAMJEE, R., AND SINHA, P. AutoCalib: Automatic Traffic Camera Calibration at Scale. In *Proc of ACM BuidSys* (2017), ACM.
- [22] CATALÁ-PRAT, Á., RATAJ, J., AND REULKE, R. Self-calibration System for the Orientation of a Vehicle Camera. In *Proc. of the ISPRS Com. V Symposium: 3D Image Engineering and Vision Metrology* (2006), pp. 68–73.
- [23] CIPOLLA, R., DRUMMOND, T., AND ROBERTSON, D. P. Camera calibration from vanishing points in image of architectural scenes. In *BMVC* (1999), vol. 99, pp. 382–391.
- [24] DAILEY, D. J., CATHEY, F. W., AND PUMRIN, S. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems* 1, 2 (2000), 98–107.
- [25] DE PAULA, M. B., JUNG, C. R., AND DA SILVEIRA JR, L. Automatic on-the-fly extrinsic camera calibration of onboard vehicular cameras. *Expert Systems with Applications* 41, 4 (2014), 1997–2007.
- [26] DERPANIS, K. G. Overview of the ransac algorithm. *Image Rochester NY* 4, 1 (2010), 2–3.
- [27] DEUTSCHER, J., ISARD, M., AND MACCORMICK, J. Automatic camera calibration from a single manhattan image. In *ECCV* (2002), Springer, pp. 175–188.
- [28] DUBSKÁ, M., HEROUT, A., AND SOCHOR, J. Automatic camera calibration for traffic understanding. In *BMVC* (2014).
- [29] GRASSI, G., JAMIESON, K., BAHL, P., AND PAU, G. Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (2017), ACM, p. 16.
- [30] GRÄTER, J., SCHWARZE, T., AND LAUER, M. Robust scale estimation for monocular visual odometry using structure from motion and vanishing points. In *Intelligent Vehicles Symposium (IV), 2015 IEEE* (2015), IEEE, pp. 475–480.
- [31] GUIDUCCI, A. Camera calibration for road applications. *Computer vision and image understanding* 79, 2 (2000), 250–266.
- [32] HANEL, A., AND STILLA, U. Calibration of a vehicle camera system with divergent fields-of-view in an urban environment. *Publikationen der Deutschen Gesellschaft für Photogrammetrie* (2017), 160.
- [33] HARTLEY, R. I. In defence of the 8-point algorithm. In *ICCV 1995* (1995), IEEE, pp. 1064–1070.
- [34] HASELHOFF, A., AND KUMMERT, A. A vehicle detection system based on haar and triangle features. In *Intelligent Vehicles Symposium, 2009 IEEE* (2009), IEEE, pp. 261–266.
- [35] HE, X., AND YUNG, N. H. C. New method for overcoming ill-conditioning in vanishing-point-based camera calibration. *Optical Engineering* 46, 3 (2007), 037202.
- [36] HUANG, S., YING, X., RONG, J., SHANG, Z., AND ZHA, H. Camera calibration from periodic motion of a pedestrian. In *Proc. of IEEE CVPR* (2016), pp. 3025–3033.
- [37] KONG, H., AUDIBERT, J.-Y., AND PONCE, J. Vanishing point detection for road detection. In *IEEE CVPR* (2009), IEEE, pp. 96–103.
- [38] LEPETIT, V., MORENO-NOGUER, F., AND FUA, P. Epnp: An accurate o(n) solution to the pnp problem. *International journal of computer vision* 81, 2 (2009), 155–166.
- [39] MIKSIK, O. Rapid vanishing point estimation for general road detection. In *Proc. of ICRA* (2012), IEEE, pp. 4844–4849.
- [40] MOGHADAM, P., AND DONG, J. F. Road direction detection based on vanishing-point tracking. In *Proc. of IROS* (2012), IEEE, pp. 1553–1560.
- [41] NIETO, M., SALGADO, L., JAUREGUIZAR, F., AND CABRERA, J. Stabilization of inverse perspective mapping images based on robust vanishing point estimation. In *Intelligent Vehicles Symposium, 2007 IEEE* (2007), IEEE, pp. 315–320.
- [42] NISTÉR, D. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence* 26, 6 (2004), 756–770.
- [43] RIBEIRO, A. A. G. A., DIHL, L. L., AND JUNG, C. R. Automatic camera calibration for driver assistance systems. In *Proc. of IWSSIP 2006* (2006), Citeseer, pp. 173–176.
- [44] SEO, Y.-W., AND RAJKUMAR, R. R. Utilizing instantaneous driving direction for enhancing lane-marking detection. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE* (2014), IEEE, pp. 170–175.
- [45] SLABAUGH, G. G. Computing euler angles from a rotation matrix. *Retrieved on August 6, 2000* (1999), 39–63.
- [46] SONG, S., CHANDRAKER, M., AND GUEST, C. C. Parallel, real-time monocular visual odometry. In *Proc. of ICRA* (2013), IEEE, pp. 4698–4705.
- [47] SUTTORP, T., AND BUCHER, T. Robust vanishing point estimation for driver assistance. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE* (2006), IEEE, pp. 1550–1555.
- [48] SZELISKI, R. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.
- [49] WANG, Y., YANG, J., LIU, H., CHEN, Y., GRUTESER, M., AND MARTIN, R. P. Sensing vehicle dynamics for determining driver phone use. In *ACM MobiSys* (New York, NY, USA, 2013), MobiSys '13, ACM, pp. 41–54.
- [50] ZHANG, Z. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision* 27, 2 (1998), 161–195.
- [51] ZHUANG, H., AND WU, W.-C. Camera calibration with a near-parallel (ill-conditioned) calibration board configuration. *IEEE Transactions on Robotics and Automation* 12, 6 (1996), 918–921.