Online Camera-LiDAR Calibration Monitoring and Rotational Drift Tracking

Jaroslav Moravec and Radim Šára

***Abstract*—The relative poses of visual perception sensors distributed over a vehicle's body may vary due to dynamic forces, thermal dilations, or minor accidents. This paper proposes two methods, OCAMO and LTO, that monitor and track the LiDAR-Camera extrinsic calibration parameters online. Calibration monitoring provides a certificate for reference-calibration parameters validity. Tracking follows the calibration parameters drift in time. OCAMO is based on an adaptive online stochastic optimization with a memory of past evolution. LTO uses a fixed-grid search for the optimal parameters per frame and without memory. Both methods use low-level point-like features, a robust kernel-based loss function, and work with a small memory footprint and computational overhead. Both include a preselection of informative data that limits their divergence.**

The statistical accuracy of both calibration monitoring methods is over 98 %, whereas OCAMO monitoring can detect small decalibrations better, and LTO monitoring reacts faster on abrupt decalibrations. The tracking variants of both methods follow random calibration drift with an accuracy of about 0.03° in the yaw angle.

***Index Terms*—Computer Vision for Transportation, LiDAR-Camera Systems, Calibration and Identification, Sensor Fusion**

I. Introduction

Like the human driver, an autonomous vehicle gathers information about the scene to interpret the current situation and correctly plan subsequent actions. For this purpose, it is equipped with several types of sensors [2]. Most researchers use a combination of LiDARs and cameras [3]. The sensing abilities of cameras and LiDARs complement each other, which allows for a more robust perception [3].

Supported by the OP VVV MEYS project 'Research Center for Informatics' [Grant CZ.02.1.01/0.0/0.0/16019/0000765], by the Czech Technical University in Prague [Grant SGS22/111/OHK3/2T/13] and in part by the Technology Agency of the Czech Republic under the National Competence Centres II Programme [Project #TN02000054 'Božek Vehicle Engineering NCC-II' (BOVENAC)]. *(Corresponding author: Jaroslav Moravec.)*

The authors are with the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic (e-mail: moravj34@fel.cvut.cz; sara@fel.cvut.cz).

Published as: J. Moravec and R. Šára, "Online Camera–LiDAR Calibration Monitoring and Rotational Drift Tracking," in IEEE Transactions on Robotics, vol. 40, pp. 1527–1545, 2024, DOI: 10.1109/TRO.2023.3347130. Available [online]: https://ieeexplore.ieee.org/document/10374278.

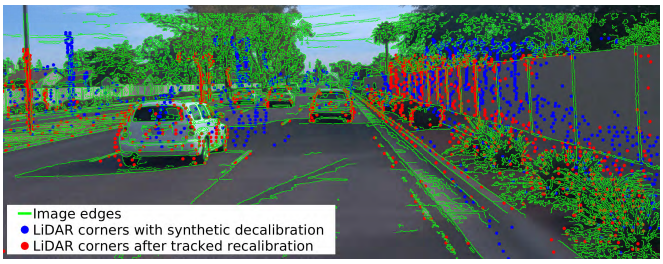


Fig. 1: Calibration parameter drift tracking between a camera and a LiDAR as a pre-requisite for calibration monitoring. The tracking result of a simulated calibration drift is shown on data from [1] after 1500 data frames. The tracker optimizes the distance of the red dots from the green curves, see the legend.

A successful sensor fusion in such multi-sensor setups requires accurately calibrating their intrinsic and extrinsic parameters. However, the extrinsic parameters (the relative translation and rotation between the sensors) are not temporally stable in a distributed sensor system due to small dynamic deformations of the vehicle's frame or body [4] and due to thermal dilations [5]. Depending on the camera type, the intrinsic calibration parameters may also vary in a non-negligible way in time and with temperature [6], [7]. We call the time-varying calibration parameters the *calibration drift*; as first conceptualized in [8].

The standard calibration procedure of such a sensor system is *offline*, performed once, in a batch optimization manner, before deploying the sensor system. Offline calibration can be performed accurately in controlled spaces using optimally designed arrangements of suitable reference targets [9] or from infrastructure using a targetless method [3].

In contrast, *online calibration* tracks the dynamically changing calibration parameters in real time during the system's normal function in the field [8]. However, we argue that an online update of the calibration may result in instability or an ultimate decalibration[1] of the sensor system under a temporary loss of the calibrating information, for instance, when the geometric structure of the scene degenerates or the sensors get temporarily blinded. Therefore, we first consider

[1] *Decalibration* is the process of altering calibration parameters due to physical damage, thermal dilation, synthetic change, etc. *Calibration* is a procedure for finding the correct calibration parameters.

a weaker variant: *calibration monitoring*. The monitor does not update the calibration; it only reports the deviation of the calibration parameters from a reference.

The monitor may (or may not) track the calibration parameters in time. If it does, it performs an online calibration only internally. We call such internal mechanism *calibration tracking*, see Figure 1. The design of the tracking algorithm should be optimized for the sequential character of data acquisition and the dynamic character of the tracking task, emphasising a fast response time. Therefore, batch optimization with its long latency should be avoided. Instead, we propose an online optimization method for the task.

The tracker may still diverge due to the lack of calibrating information in the current data frame. Therefore, we introduce the mechanism of *frame preselection*, which automatically decides if the input data frame is suitable for stable calibration monitoring/tracking.

In summary, we distinguish three intertwined mechanisms: (1) incremental *calibration tracking*, which follows the evolution of calibration parameters, (2) *calibration-frame preselection*, which limits the calibration tracker divergence, and (3) *calibration monitoring*, which provides a validity certificate for the reference calibration.

We consider two kinds of dynamic decalibrations: (a) an abrupt decalibration event, and (b) slow calibration drift, and introduce several methods, based on local low-level features that are (weakly) correlated across the modalities. We propose an information-theoretic objective function based on kernel correlation and an online gradient-descent algorithm with adaptive learning for calibration parameter tracking. The methods discussed in this paper have small computational overhead and small memory footprint.

The focus is on a single multi-modal LiDAR–camera pair. An extension to multi-LiDAR multi-camera systems is trivial.

II. Related Work

We assume that the intrinsic calibration parameters of the camera [10] and the LiDAR [11] are known, and we focus only on the extrinsic parameters for LiDAR-camera systems.

Usually, extrinsic calibration methods rely on predefined calibration targets with known dimensions and patterns [12]–[14]. Correspondences are obtained by acquiring several frames of those targets in both modalities. These methods optimize reprojection and/or point-to-plane registration errors, possibly using constraints on the transformation. They achieve a good precision of calibration parameters, although they are not very versatile due to a long setup time. Hence, to avoid these limitations, there has recently been an interest in automatic, targetless approaches [3]. We review them here as they describe image and LiDAR features and the optimization of objective functions suitable for the calibration tracking and monitoring introduced above.

As discussed in [3] and elsewhere, targetless methods can be broadly classified into the following categories:

Information Theory-Based: These methods maximize a similarity measure between image features and the projected point cloud. The most straightforward methods employ LiDAR intensity (the return strength of a laser beam) and image grayscale intensity. In [15], they maximized mutual information in terms of Shannon entropy. The kernel density estimator is used for the marginal and joint probability densities. In our experience, reliable LiDAR intensity features are typically too sparse in traffic scenes to be used alone in an online calibration mechanism, and a combination of 3D point and its intensity is needed.

Besides the LiDAR intensity, one can also extract semi-local features from LiDAR point clouds, such as surface normals. In [16], the normals are estimated using the point difference between consecutive points in the scanline and its angle relative to the x–z plane. Instead of Shannon entropy, they used normalized mutual information, which lowers the too-strong relative influence of data with a smaller overlap in the field of view (FoV).

One can also maximize the alignment between the gradient orientation and magnitude of LiDAR points and image pixels. Taylor et al. [17] extracted image gradients using the Sobel operator [18]. Obtaining the point cloud gradients from the sparse LiDAR data required a projection onto a camera-centred sphere. The mean gradient orientation and magnitude were estimated from the eight neighbors there. Although this yields good calibration results, it has an extra computational overhead since the sphere needs to be re-centred in every iteration of the procedure.

High-level features, such as object semantic labels, can also be used for automatic calibration purposes [19], [20]. In [19], the authors optimized the mutual information between the 3D point semantic label and its 2D-projection semantic label. This method highly depends on the quality of semantic labels. Obtaining them is quite time-consuming to be used in an online calibration procedure. A combination of segmentation and structure-from-motion (SfM) is proposed in [20].

Motion-Based: These methods do not look for direct similarities between modalities but rather compute the odometry from each sensor individually and then estimate the relative extrinsic calibration by solving the hand-eye calibration (HE) problem, as in [21]. They used the iterative closest point (ICP) algorithm to estimate the trajectory in LiDAR and a standard SfM procedure for the cameras. The rotation accuracy was good, but the translation accuracy was low since the observability of rotational decalibration in automotive traffic scenes is better than that of translational decalibration, which depends on the scene distance from the sensors. This approach can calibrate sensors without any overlap in their field of view. However, the trajectories required for the HE calibration have to exhaust all six degrees of freedom. It may be challenging (especially for a car) to perform complex-enough movements. These properties make the motion-based methods less suitable for calibration monitoring.

End-to-End Learning-Based: With the advent of neural networks, a new group of methods, which employ end-to-end learning of extrinsic calibration parameters, arises [22]–[24]. These methods are automatic and targetless, but rely heavily on the training data. They also require high-performance hardware for training and inference to achieve real-time performance. These requirements might make their use in the automotive domain problematic, as those processing units are usually allocated for segmentation, object detection or optical flow estimation.

Feature-Based: Last but not least, one can also extract some significant local features from both modalities and optimize their alignment based on some metric.

The correspondence-based methods of this class use a descriptor similarity. One can find features using various methods (Förstner operator [25], Sobel operator [26] or SIFT [27]). The descriptors of these features are then matched using some robust estimator (e.g., RANSAC). These methods need high-density range finders (e.g., terrestrial laser scanners) to obtain easily detectable features in both modalities. Hence, they are unsuitable for the automotive domain, where fast-spinning LiDARs with sparse point clouds are typically used.

The other sub-class of feature-based methods is not based on one-to-one correspondences. These methods also align extracted features between modalities but use ICP-like spatial geometrical relation rather than descriptor similarity. This makes these methods highly efficient, as the low-level features and the geometrical error are generally easier to work with.

One of the very first methods that used the relation between depth discontinuities in LiDAR data and changes of intensity in the camera was [8]. Their method maximizes the alignment of extracted features from the LiDAR and the camera and is efficient enough to be run in real time. They were also the first to introduce the problem of online calibration monitoring and tracking. We will use a slight modification of this method (referred to as LT) as a baseline for comparison, and we will describe this method in greater detail in Section III-D.

Lines, instead of points, are used in [28]. The method is based on the alignment between 3D line segments from the LiDAR point cloud and 2D line segments from the camera image. The 2D line segments are extracted using Hough Transform [29] on Canny edges [30]. The 3D line segments are found by intersections of fitted planes and detected 3D boundaries of objects. The method is automatic and not limited to any sensor configuration, but it relies on the existence of sufficiently many detectable lines.

Recently, a new method, based on the alignment between image and LiDAR local features, was introduced in [31]. Their model is based on the Gaussian mixture model (GMM), which is a richer model than the kernel correlation considered in this paper since the GMM weights explicitly encode feature strengths (as also implicitly done in [8]). The optimization procedure is designed for automatic offline calibration rather than online monitoring or tracking. Since this offline calibration method can be considered close to the methods proposed in this paper, we compare the estimated parameter precision in Section VI-F, and show that we achieve similar results even though we use a simpler model.

An in-depth analysis of point features in the two modalities was recently published in [32]. They investigated different types of natural edges and showed that the usage of depth-continuous edges helps the accuracy of the calibration. These edges often occur on plane intersections, so it is harder and more time-consuming to find them, as a proper plane fitting needs to be used.

Methods considered in this paper fall into the targetless feature-based category. We choose features that are present in all scenes and do not require data segmentation or interpretation, which are processes prone to errors and are computationally expensive on limited-resource hardware. In this paper, we propose a method for **O**n-line **CA**libration **MO**nitoring (**OCAMO**) of relative LiDAR-camera extrinsic calibration. Then we propose the LTO method, a blend of a modification of the LT method from [8] and OCAMO. Methods considered in this paper are sketched in Figure 2 and explained in Section III.

The main contributions of this paper are: (1) a definition of calibration tracking as an online learning algorithm, (2) robust learning loss function based on Gaussian kernel correlation, (3) adoption of frame preselection concept into the sensor calibration and demonstration of its effectiveness, (4) introduction of monitoring and calibration tracking divergences as stability criteria (besides the standard statistical accuracy and precision), and (5) a thorough experimental evaluation of the proposed methods based on extensive real and simulated data.

III. Methods

A. Problem Formulation, Goals, and Assumptions

Let a reference (initial) calibration parameters obtained, for example, from a standard factory or lab calibration procedure, be given. The goal of *calibration tracking* is to follow the temporal evolution of calibration parameters during the data acquisition needed for the normal function of the vehicle. This will be done by an online maximization of the alignment of the sensor data using low-level local features. Since tracking the calibration at the full framerate is usually unnecessary or temporarily impossible due to a lack of calibrating information in input data, the *frame preselection* mechanism will reduce the divergence of the tracked parameters due to weak or misleading information. It works by filtering the incoming data frames based on a rule learned in a supervised manner. The preselection is independent of tracking and monitoring; hence it can be used with any feature-based targetless calibration or calibration

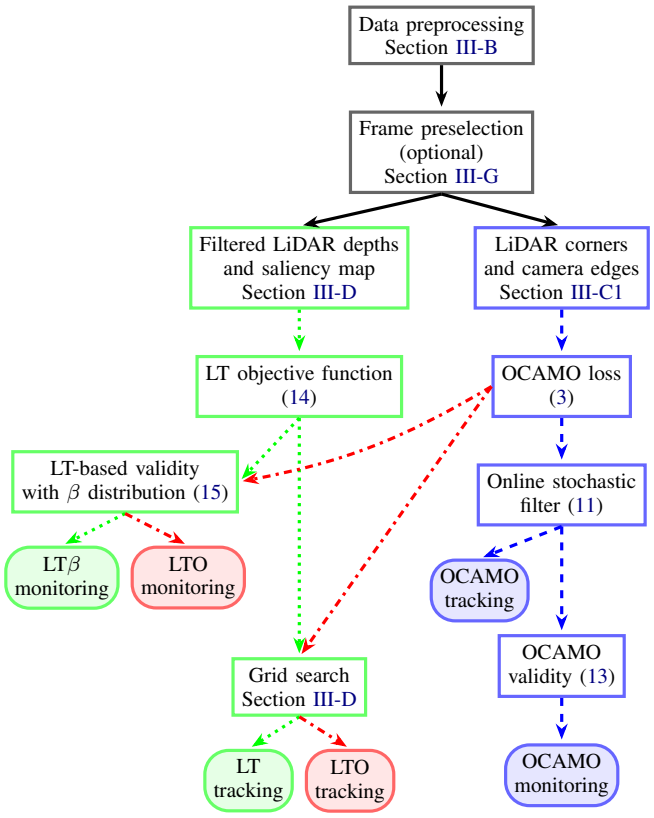


Fig. 2: The structure of three calibration monitoring and tracking methods: LT (green path), LTO (red), and OCAMO (blue). Rectangular boxes are processing blocks; ovals show outputs. OCAMO and LTO are novel methods proposed in this paper.

monitoring method. Finally, *calibration monitoring* will provide a validity certificate for the reference calibration. It is the probability that the currently tracked parameters are equal to the reference parameters, given all the observed (prefiltered) data so far and an online estimate of the data variance. Again, a learned rule will be used.

Calibration monitoring requires real-time performance and high statistical accuracy (low false positive and false negative rates). Following all the calibration parameters is unnecessary if the goal is calibration monitoring. In a real-world use case, the most important part of calibration is the relative rotation between the sensors. Accurate rotational registration is essential for multi-modal perception algorithms. We show (Section VI-A) that decalibrations in some other parameters will manifest themselves as rotational decalibrations, too.

We consider three calibration tracking and monitoring methods: (1) An original method referred to as OCAMO, (2) the reference, state-of-the-art method of Levinson and Thrun [8] referred to as LT (with minor modifications required for the monitoring task, that we refer to as LTβ), (3) and a combination of LT/LTβ and OCAMO, called LTO.

The proposed methods assume that the field of view of both sensors makes a significant overlap. We also assume that the LiDAR point clouds are composed of sequentially acquired scanlines. Our methods require synchronized sensors, and the LiDAR point cloud needs to be compensated for egomotion and the rolling-shutter effect of the cameras. We further assume that we know the intrinsic parameters of the camera and LiDAR and the reference transformation between sensors monitored by our method in the form of position and orientation in the vehicle reference frame. We also assume that data was acquired during the daytime without heavy rain or other adverse weather conditions.

We first describe data preprocessing common to all the methods, then the three methods in turn, and finally, the frame preselection method independent of both OCAMO and LT. Figure 2 shows a diagram of all three monitoring and tracking methods for an easier navigation in this section.

B. Data Preprocessing

Image Undistortion: We work with a perspective camera model with nonlinear distortion using a 3rd-order radial model with parameters κ_1, κ_2, κ_3 and a 2nd-order tangential distortion model with parameters p_1, p_2 [33], [34].

LiDAR Point Cloud Motion Compensation: If the vehicle moves and a rotating LiDAR with a finite framerate (e.g., 5, 10, 20 Hz) is used, the pose of the vehicle (sensor) at the start of the frame is different from the ending one [35]. Similarly, if a rolling-shutter camera is used, its first scanline is exposed earlier than the last scanline [36]. Motion compensation is a standard procedure that corrects the coordinates of the 3D points in the point cloud to be consistent with the image [1]. We assume we know the odometric information of the vehicle carrying the sensors. That motion is measured in the vehicle reference frame (VRF). Since we know the reference calibration between the vehicle and each sensor, we can express the sensor motion in its reference frame (sensor reference frame, SRF).

To perform the motion correction, we interpolate sensor rotation and translation independently. Let $\mathbf{x}_i(t_i)$ be i-th 3D point measured by the LiDAR at time t_i, expressed in the SRF. Let t_0, t_1 be the interval corresponding to the odometric server's framerate. It is assumed that t_0 and t_1 are close to the camera and the LiDAR frame acquisition intervals. Let the sensor translation between time t_0 and t_1 be $\mathbf{t}_{01}$, which corresponds to velocity $\mathbf{v}_{01} = \mathbf{t}_{01}/(t_1 - t_0)$. Let the sensor rotation between t_0 and t_1 be $\varphi_{01}\mathbf{o}$, where φ_{01} is the rotation angle in radians and $\mathbf{o}$ is the unit rotation axis, constituting the rotational velocity vector $\boldsymbol{\phi}_{01} = \varphi_{01}/(t_1 - t_0)\,\mathbf{o}$. Then the position of point $\mathbf{x}_i$ motion-compensated to time t is

$$\mathbf{x}_i(t) = \mathbf{x}_i(t_i) + (t_i - t)\,\mathbf{v}_{01} + \mathbf{R}\big((t_i - t)\boldsymbol{\phi}_{01}\big)\,\mathbf{x}_i(t_i)\,, \quad (1)$$

in which Rodrigues' formula [37] is used to map the the rotational vector $(t_i - t)\boldsymbol{\phi}_{01}$ to rotation matrix $\mathbf{R}$. When t

is the image's timestamp, we say the point $\mathbf{x}_i$ is motion-compensated to the image acquisition time. That would assume the entire image is acquired at the same time. This is not the case in rolling-shutter cameras: There, the timestamp of the point t_i and the timestamp of the *image scanline* l, to which the point projects, have to be equal. The difference is small but cannot be neglected because that would result in a systematic bias in the tracked calibration parameters. Since the correction depends on where the point projects to the image, an iterative method [1] is used: Let the initial value of t correspond to the middle of the image exposure interval. The point $\mathbf{x}(t)$ is projected to the image, the timestamp t_l of the closest scanline is determined, t is updated as $t \leftarrow t_l$ and $\mathbf{x}_i$ is re-compensated using (1). Typically only five iterations are required.

The current image I_j and the current motion-compensated point cloud P_j constitute a *(data-)frame* $F_j = (I_j, P_j)$. Then $\mathcal{F}_n = \{F_j\}_{j=1}^n = \{(I_j, P_j)\}_{j=1}^n$ will represent a sequence of n (possibly preselected) frames.

C. The OCAMO Monitoring and Tracking Methods

It is known from the psychology of vision that object boundaries are often co-located with image edges or texture boundaries. This observation has already been made in [38]. Therefore, we extract *edges* as image features and *corners* as LiDAR features. The tracking algorithm then maximizes the alignment of those features by changing the extrinsic calibration parameters. The alignment is measured as the entropy of the union of the image edge points and the projected LiDAR corners, expressed by kernel correlation [39]. Online stochastic optimization minimizing the correlation is used. This subsection describes the individual steps of the algorithm.

1) Data Preprocessing: Canny edges [30] are detected in every image I_j. We use only that part of the image where there is a (potential) overlap with the LiDAR FoV[2]. Thus for an image I_j, we get a so-called edge image $E_j \in \mathbf{Z}^{2\times m}$, where m is the total number of edge pixels. Examples are shown in green in Figure 6, together with the projected LiDAR corners.

3D LiDAR corners C_j are obtained as follows. Given a point cloud P_j, we process each horizontal scanline separately with 3D points ordered by the azimuth, with collapsed azimuthal gaps. Let each scanline contain only valid points uniquely defined by their azimuth φ: $p(\varphi)$. Each point has its radial distance $d(\varphi)$ and reflectance $r(\varphi)$. We extract LiDAR corners based on three criteria: (1) a large change in radial distance, (2) a large change in reflectance, and (3) wide azimuthal gaps in a scanline. A detailed description of LiDAR corner extraction is given in Appendix A, and its parameters are listed in Tab VII in Appendix B, together with all the other parameters.

[2]This is approximately known from the design of the system. In this paper, it is the bottom two-thirds of the image.

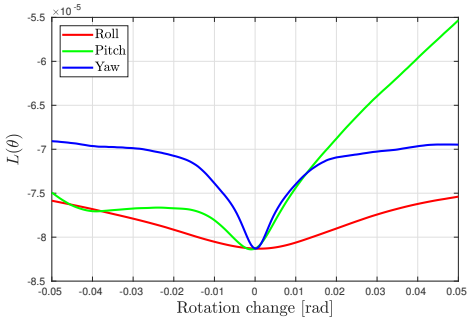


Fig. 3: Examples of the OCAMO loss function (3) for rotational displacements in the roll, pitch, and yaw angles.

2) The OCAMO Calibration Tracking: Extrinsic translation and rotation are represented as a pair $\boldsymbol{\theta} = (\mathbf{t}, \boldsymbol{\omega})$, where $\mathbf{t} \in \mathbb{R}^3$ is the translation vector and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the rotation vector such that the rotation matrix is $\mathbf{R} = \exp(\boldsymbol{\omega}_\times)$, in which $\boldsymbol{\omega}_\times$ is a skew-symmetric 3×3 matrix and exp is the matrix exponential function for which we use Rodrigues' formula again. See, e.g., [37] for the theory. Let $\mathrm{Proj}\colon \mathbb{P}^3 \to \mathbb{R}^2$ be the image projection function with all the intrinsic parameters within and let the 3D LiDAR corners $\mathbf{X}_c \in C_j$ project to the image as

$$\mathbf{x}_c(\boldsymbol{\theta}) = \mathrm{Proj}\left(\exp\left(\boldsymbol{\omega}_\times\right) \mathbf{X}_c + \mathbf{t}\right), \tag{2}$$

where $\boldsymbol{\theta} = (\mathbf{t}, \boldsymbol{\omega})$ is the extrinsic calibration and $\mathbf{x}_c(\boldsymbol{\theta}) \in \mathbb{R}^2$ is the projected point in Cartesian coordinates.

Given the reference parameters $\boldsymbol{\theta}^{\mathrm{ref}}$, calibration tracking aims to find values of the updated parameters $\boldsymbol{\theta}^{\mathrm{trk}}$ s.t. the projected LiDAR corners *align* with the image edges in frame F_j. Initially, $\boldsymbol{\theta}^{\mathrm{trk}} = \boldsymbol{\theta}^{\mathrm{ref}}$. To express a measure of the alignment, we use kernel correlation [39] between the discrete set C_j^p of projected corners and the discrete set E_j of image edge pixels in frame F_j

$$L(\boldsymbol{\theta} \mid F_j, k, \sigma) = -\sum_{\mathbf{x}_c \in C_j^p} \sum_{\substack{\mathbf{x}_e \in \\ k\mathrm{NN}(\mathbf{x}_c, E_j, k)}} \exp\left[-\frac{\|\mathbf{x}_c(\boldsymbol{\theta}) - \mathbf{x}_e\|^2}{2\sigma^2}\right], \tag{3}$$

where $k\mathrm{NN}(\mathbf{x}_c, E_j, k)$ are the k nearest neighbors of point $\mathbf{x}_c(\boldsymbol{\theta})$ in the set of edge pixels and k and σ are the parameters of the model. The k, typically large, is chosen as a trade-off between the accuracy of the kernel correlation approximation and computational efficiency. See Section V-A for parameter selection and Table VII for the values. An example of the loss function for rotational displacement is shown in Figure 3 (on a sequence of 200 frames). Note that this is a random function due to the finiteness of the data sample and the current content of the scene.

The calibration tracking is then a minimization problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_F\big[L(\boldsymbol{\theta} \mid F, k, \sigma)\big], \tag{4}$$

where $\mathbb{E}_F[L(\cdot)]$ is the expectation of $L(\cdot)$ over the set of all possible frames. We approximate the expectation in (4) with empirical expectation over the sequence $\mathcal{F}_n$ and use the adaptive stochastic gradient descent (SGD) optimization algorithm. We chose an on-line algorithm (SGD) because it has a smaller latency than batch optimization. In the case of the monitoring problem, time to detection is important.

The update rule of SGD is [40],

$$\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} - \operatorname{diag}(\boldsymbol{\nu}) \frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}^{(j)} \mid F_j, k, \sigma)\,, \qquad (5)$$

in which $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_6)$ is the learning rate vector forming a 6×6 diagonal matrix $\operatorname{diag}(\boldsymbol{\nu})$, and j is the data frame index. The learning rate adapts to the local curvature of the manifold and to the temporal changes of the calibration parameters. Let $\dot{L}_i = \frac{\partial}{\partial \theta_i} L(\boldsymbol{\theta} \mid F, k, \sigma)$ be the first partial derivative in parameter i, and similarly, let $\ddot{L}_i$ be the second derivative. We use numerical differentiation with a parameter difference of h_θ set uniformly for all[3] parameters (Table VII). Let

$$g_i = \mathbb{E}_F[\dot{L}_i], \quad v_i = \mathbb{E}_F[(\dot{L}_i)^2], \quad h_i = \mathbb{E}_F[\ddot{L}_i]. \qquad (6)$$

The per-parameter adaptive learning rate is [40]

$$\nu_i = \frac{1}{h_i} \frac{g_i^2}{v_i}\,, \qquad i = 1, \ldots, 6. \qquad (7)$$

The expectations g_i, v_i, h_i are estimated by an adaptive Robbins-Monro filter

$$g_i^{(j+1)} = \frac{m_i^{(j)} - 1}{m_i^{(j)}}\, g_i^{(j)} + \frac{1}{m_i^{(j)}} \frac{\partial}{\partial \theta_i} L(\boldsymbol{\theta}^{(j)} \mid F_j, k, \sigma), \qquad (8)$$

and similarly for $v_i^{(j+1)}$ and $h_i^{(j+1)}$. Its adaptivity is controlled by the dynamic memory size $m_i^{(j)}$ per parameter i, as in [40]:

$$m_i^{(j+1)} = 1 + \left(1 - \frac{\left(g_i^{(j+1)}\right)^2}{v_i^{(j+1)} + \varepsilon}\right) m_i^{(j)}, \qquad (9)$$

with $m_i^{(1)} = 1$ for all i. The constant $\varepsilon = 10^{-10}$ prevents a stall at the beginning of the memory adaptation process. Its exact value is not critical.

The current estimates for the learning rate are then $g_i \approx g_i^{(j+1)}$, $h_i \approx h_i^{(j+1)}$, $v_i \approx v_i^{(j+1)}$ and (7) can be used in (5).

We now describe our modifications of the SGD method.

If the memory size (9) was following the conditions for convergence [41], i.e. tending to a large value over time, the tracking might become slower to respond to calibration changes, eventually diverging. On the other hand, if the memory was too small, there could be frequent false alarms due to short sequences of non-informative frames. Therefore, we limit the maximum memory size m_i to $m^{\mathrm{bnd}} = 5$. This seems small, but note that a single frame provides hundreds to thousands of individual measurements. The m^{bnd} value is a tradeoff between convergence properties and the ability to track a nonstationary random process.

The expectations (8) need to aggregate measurements before the updates (5) can take place. Hence, we introduce a period of $b = 10$ frames, during which the estimates $\boldsymbol{\theta}$ are not updated (a burn-in period).

We also need safeguards against a large, irreversible divergence of the tracked parameters, be it false or due to a (temporary) decalibration. Therefore, we augment the learning process (5) and the memory adaptation (9) as follows. Let

$$\delta_i^{(j)} = \frac{1}{h_i^{(j+1)}} \frac{\partial}{\partial \theta_i} L(\boldsymbol{\theta}^{(j)} \mid F_j, k, \sigma)\,.$$

Consider a clipped *update proposal*

$$\hat{\theta}_i^{(j+1)} = \theta_i^{(j)} - \frac{\left(g_i^{(j+1)}\right)^2}{v_i^{(j+1)}} \operatorname{sign}(\delta_i^{(j)})\, \min\left(|\delta_i^{(j)}|,\, \theta_i^{\mathrm{upd}}\right), \qquad (10)$$

in which we use a vector of update limits $\boldsymbol{\theta}^{\mathrm{upd}}$. In addition, the final robust version of the rule (5) also clips the update. All together,

$$\theta_i^{(j+1)} = \begin{cases} 0 & \text{for } j \le b\,, \qquad (11a) \\ \operatorname{sign}(\hat{\theta}_i^{(j+1)})\, \theta_i^{\mathrm{bnd}} & \text{for } |\hat{\theta}_i^{(j+1)}| > \theta_i^{\mathrm{bnd}}\,, \qquad (11b) \\ \hat{\theta}_i^{(j+1)} & \text{otherwise,} \qquad (11c) \end{cases}$$

in which we have used another limit $\boldsymbol{\theta}^{\mathrm{bnd}}$ in (11b). The three mechanims in (11) are the burn-in period b in (11a), SGD robustification in (10) and (11c), and finally, pausing of the tracking in (11b). These techniques are combined to stabilize the calibration tracking process.

The robustification using $\boldsymbol{\theta}^{\mathrm{upd}}$ in (10) is an established approach to robust stochastic estimation [42]. A similar approach (gradient clipping) is also used in training neural networks, see, e.g. [43]. The final resort is $\boldsymbol{\theta}^{\mathrm{bnd}}$. Setting it reasonably high does not affect the decalibration detection, and, on the other hand, it helps the speed of recalibration. We use $\boldsymbol{\theta}^{\mathrm{bnd}} = 5\boldsymbol{\sigma}_d$, where $\boldsymbol{\sigma}_d$ is the vector of standard deviations of the monitoring model, see Section III-C3 for its definition. See Section V-A for the parameter selection/learning and Table VII for the values.

The complete algorithm is summarized as the blue path ending in the 'OCAMO tracking' node of the flow diagram in Fig. 2.

3) OCAMO Monitoring: We represent the reference extrinsic calibration with parameters $\boldsymbol{\theta}^{\mathrm{ref}}$. Given $\boldsymbol{\theta}^{\mathrm{ref}}$ and the data-frame sequence $\mathcal{F}_n = \{F_j\}_{j=1}^n$ up to frame n, we want to decide in each frame whether the reference parameters are valid. The validity index V is the probability that the tracked parameters $\boldsymbol{\theta}^{\mathrm{trk}} \stackrel{\mathrm{def}}{=} \boldsymbol{\theta}^{(j+1)}$ at each frame F_j are equal to the reference parameters $\boldsymbol{\theta}^{\mathrm{ref}}$.

[3]Six when the OCAMO tracks all extrinsic parameters or three when OCAMO tracks rotations only.

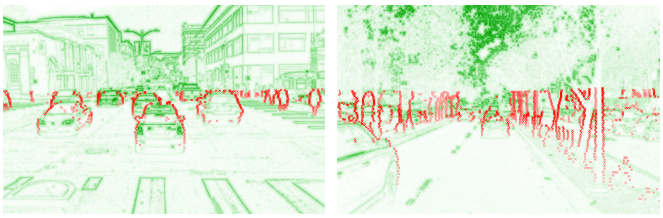


Fig. 4: Examples of saliency map D (green underlying bitmap) and saliency step S (red crosses) used in the LT method. Color saturation corresponds to saliency value.

Let θ_i^{ref} be a single reference parameter and θ_i^{trk} be the corresponding tracked value. We use a normal distribution with variance $\sigma_{d_i}^2$ for the statistical difference between θ_i^{ref} and θ_i^{trk}, assuming that the random variable is θ_i^{trk}. Let T_{θ_i} be a threshold corresponding to θ_i^{ref}. The calibration validity index for reference parameter θ_i^{ref} is then

$$V_i(\theta_i^{\text{ref}}) = P(\theta_i^{\text{ref}} - T_{\theta_i} \leq \theta_i^{\text{trk}} \leq \theta_i^{\text{ref}} + T_{\theta_i} \mid \sigma_{d_i}^2)\,, \quad (12)$$

in which $P(\cdot)$ is derived from the CDF of the univariate normal distribution with variance $\sigma_{d_i}^2$ [44]. This means that if the true decalibration is smaller than about σ_{d_i} for all i, the validity index will not indicate the decalibration.

The total OCAMO validity index for all reference parameters $\boldsymbol{\theta}^{\text{ref}}$ is then the product

$$V_{\text{OC}}(\boldsymbol{\theta}^{\text{ref}}) = \prod_{i=1}^{6} V_i(\theta_i^{\text{ref}})\,, \qquad V_{\text{OC}} \in [0,1]\,. \quad (13)$$

The selection of the $\sigma_{d_i}^2$ and T_{θ_i} hyperparameters is described in Section V-A.

The complete algorithm is summarized as the blue path ending in the 'OCAMO monitoring' node of the flow diagram in Fig. 2.

D. The LTβ Monitoring and Tracking Methods

Levinson and Thrun were the first ones to introduce the concept of calibration monitoring in autonomous vehicles and published a method to solve the problem [8]. We denote the original method as LT and the minor modification described below as LTβ. The original LT method also uses visual features extracted from LiDAR and camera and optimizes an objective function of their alignment. Since the LT approach is closest to our work, we will briefly describe it here and compare OCAMO to it in Section VI. We have reproduced the LT method from scratch.

In the image, the edge saliency map $D(\mathbf{x})$ is computed for every image pixel $\mathbf{x}$ [8]. The saliency is maximal at the image edges. In the LiDAR point cloud, step saliency $S(\mathbf{X})$ is computed in every 3D point $\mathbf{X}$ based on the local change in radial distance. High-saliency points correspond to jumps

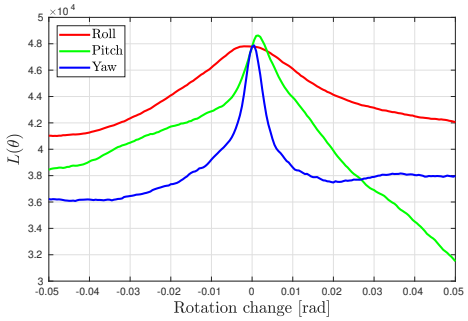


Fig. 5: Examples of the LT objective function (14) for rotational displacements in the roll, pitch, and yaw angles.

in the radial distance. Low-saliency points are excluded. Examples of edge saliency map $D(\mathbf{x})$ and LiDAR step saliency $S(\mathbf{X})$ are shown in Figure 4. Let $\mathcal{F}_{n,w} = \{F_j\}_{j=n-w}^{n}$ be the set of the last w frames, X_j be the set of salient LiDAR 3D points in frame F_j, $\text{Proj}(\cdot)$ be the image projection function, and $\boldsymbol{\theta} = (\mathbf{t}, \boldsymbol{\omega})$ be the extrinsic calibration parameters as in (2). The objective function for the LT method is joint saliency (larger is better)

$$J(\boldsymbol{\theta}) = \sum_{j \in \mathcal{F}_{n,w}} \sum_{\mathbf{X} \in X_j} S(\mathbf{X})\, D\big(\text{Proj}\left(\exp\left(\boldsymbol{\omega}_\times\right) \mathbf{X} + \mathbf{t}\right)\big), \quad (14)$$

An example of $J(\boldsymbol{\theta})$ is shown in Figure 5 (on the same data as in Figure 3). Notice that the deviation of the maxima from the reference (zero) is larger than in Figure 3. This means that the LT objective function (14) is more sensitive to noise than the OCAMO loss function (3).

Instead of maximizing $J(\boldsymbol{\theta})$, the LT monitoring method makes a statistical test if $J(\cdot)$ attains a local maximum in the given parameters $\boldsymbol{\theta}$. This is done by computing the $3^6 - 1 = 728$ directional perturbations $J(\boldsymbol{\theta} + \mathbf{e}_l)$, $l = 1, \ldots, 728$, and computing the proportion F_C of their values lower than $J(\boldsymbol{\theta})$. Higher F_C is an indication that the calibration $\boldsymbol{\theta}$ is valid. Normal distributions $p_c(F_C)$ and $p_d(F_C)$ are learned for calibrated and decalibrated sensor pairs, respectively. The calibration validity index V_{LT} is then the posterior probability of the F_C value assuming equal priors

$$V_{\text{LT}}(\boldsymbol{\theta}) = \frac{p_c\big(F_C(\boldsymbol{\theta})\big)}{p_c\big(F_C(\boldsymbol{\theta})\big) + p_d\big(F_C(\boldsymbol{\theta})\big)}\,, \quad V_{\text{LT}}(\boldsymbol{\theta}) \in [0,1]\,. \quad (15)$$

Our modification of the LT method uses beta distribution instead of normal distribution for p_c and p_d. We noticed that it captures the distribution of F_C values better, especially for the p_c. This leads to better performance, as discussed in Section VI-C. We denote this modification as LTβ, with $V_{\text{LT}\beta}$ the corresponding validity index. Unlike in OCAMO, it is important that LTβ tracks all six degrees of freedom because that results in a more concentrated distribution p_c due to the definition of F_C. That leads to a better discrimination of calibration/decalibration by LT/LTβ.

The complete LT method is shown as the green path ending in the 'LTβ monitoring' node in Figure 2.

The LT does not need to track the calibration parameters to estimate the validity index. To implement a calibration drift tracker, LT uses a different procedure: A combination of parameters is found that is better than the current ones w.r.t. the joint saliency objective function (14) using the directional perturbations. LT tries to follow these perturbations to track the parameters. The LT tracker is shown as the green path ending in the 'LT tracking' node in Figure 2.

E. A Comparison of The OCAMO and LT Methods

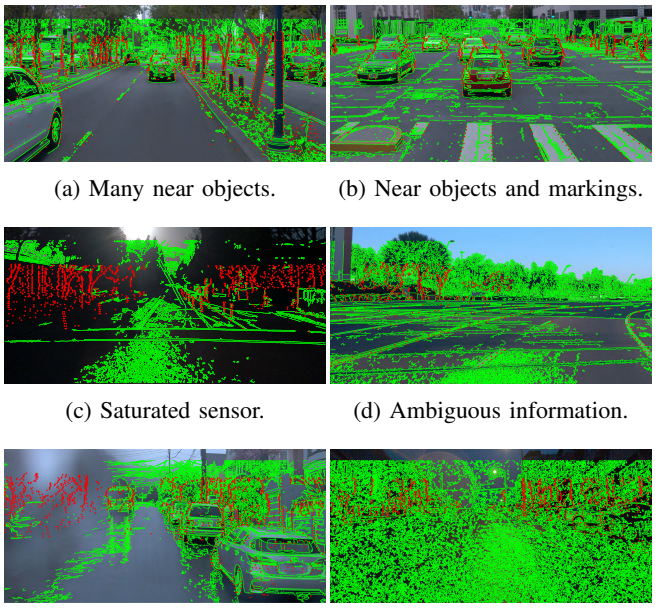
The OCAMO method differs from the LT method in several key respects. First, OCAMO uses reflectance information in addition to distance jumps in the 3D data. Therefore, valuable information from strongly reflecting objects (traffic signs, road markings, etc.) is used. Second, OCAMO uses azimuthal gaps (called skylines in [3]) in the 3D data that occur at object boundaries against distant backgrounds. In some scenes, it is the only reliable calibration monitoring information available. We found this as an important factor that stabilizes rotation parameter monitoring. Third, in place of the image edge saliency of the LT method, the OCAMO uses kernel correlation which provides similar information but can discern finer calibration changes due to aggregating support from a greater neighborhood. Kernel correlation is also implicitly robust to outliers in the statistical sense, as discussed in [39]. Lastly, the definition of the calibration validity index differs: We use the stochastic gradient descent method to track the parameters and then perform a statistical test of equivalence with the reference calibration. On the other hand, the LT method directly computes the statistical validity index. At first sight, it is not obvious which validity index is better. This question led us to introduce the LTO method below, whose performance is compared to OCAMO in Section VI.

F. The LTO method

In order to compare the monitoring efficiency of the OCAMO loss function (3) with respect to the LT objective function (14), we replaced the OCAMO validity index (13) with the LT-based validity index (15). Again, we use the beta distribution instead of the normal distribution in p_c and p_d. We call this combination the LTO monitoring method. The tracking variant of LTO is identical to the LT method in Section III-D, still with the OCAMO loss function (3), as illustrated in Figure 2.

G. Frame Preselection

An important question is what kind of data frames provide information suitable for calibration monitoring. Some scenes do not provide this information (dark, saturated, degenerate, etc.). Specifically, any accurate extrinsic calibration or calibration monitoring method highly depends on near objects visible in LiDAR point clouds and good lighting conditions for the camera.



(a) Many near objects. (b) Near objects and markings.

(c) Saturated sensor. (d) Ambiguous information.

(e) Altered optical system. (f) Low image quality.

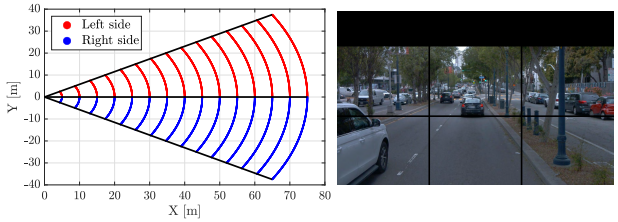
Fig. 6: Frames suitable (a, b) and unsuitable (c–f) for calibration. LiDAR corners are in red, image edges in green.

Figure 6 shows typical examples of frames of different suitability for calibration. Suitable frames in Figures 6a and 6b have many near objects (cars, curbs, poles, tree trunks), and Figure 6b also has well-detected reflective road markings. On the other hand, the image in Figure 6c is saturated by direct sunlight. In Figure 6d, all of the objects are too far, and the features in the trees are too dense to provide unambiguous registration information. In Figure 6e, raindrops on the lens alter the optical path, leading to geometric distortion. In Figure 6f, the image quality is too low at dusk.

We introduce a preselection mechanism to mitigate the influence of potentially bad frames. We make it independent of the monitoring method, although it could be considered another adaptive filter for the paused tracking discussed in Section III-C2.

We hypothesize that learning an efficient on-line predictor of frame suitability for calibration is possible. We follow a standard procedure of supervised learning of a binary classifier by specifying the feature set, creating an automatically labelled dataset, and training a classifier using k-fold cross-validation. The learning will be described in Section V-D. Here we focus on the feature extraction and the automatic labelling approach.

1) Features: As discussed above, LiDAR corners correspond to 3D object boundaries and, with the help of



(a) Thirty LiDAR semi-intervals. (b) Six image regions.

Fig. 7: Feature extraction regions that are used for frame preselection.

reflectance, to texture changes. Hence, if there are enough LiDAR corners, these will correspond to some image edges. Thus, the LiDAR features suitable for the preselection rule are based on the number of corners in specific range intervals. Since the reference calibration is known, we consider only corners in the camera's field of view. We use five-meter intervals ranging from 0 to 75 m (the maximum radial distance for the LiDAR used). This set of intervals was selected to typically contain whole objects (e.g., cars, trees, etc.) but is small enough to discretize the entire area well. Figure 7a shows the resulting 15 intervals split into left/right semi-intervals that yield a total of 30 features from the LiDAR modality, each feature being the LiDAR corner count in the respective interval. We experimented with fine azimuthal intervals, which did not yield good results.

Image edges tend to include many non-object boundaries and false negatives. We want to capture two situations: (1) Too low edge density. For example, this happens in the case of image saturation, as in Figure 6c. (2) Too high edge density. This typically happens in trees, shrubs or highly textured regions, as in Figure 6d.

Our first statistical feature counts the number of edges in an image region, providing density information. The second feature is related to high-density image regions. For each edge pixel, we count edge pixels (out of 20 neighbors) in its vertical n_v and horizontal neighborhoods n_h. Then the quotient

$$\frac{\max(n_v, n_h)}{\min(n_v, n_h)}$$

tells us how dense a given edge pixel neighborhood is. Meaningful edge pixels have a high quotient value. The average quotient over an image region is our second statistical feature.

The image is partitioned into six regions, as shown in Figure 7b. The area around the horizon is mainly in the top regions, and the top quarter of the image is discarded. Each region has two feature values, which gives a total of 12 feature values per image.

2) Automatic Labeling: We devised an automatic labelling method based on the shape of the loss function minimum. If the loss function $L(\boldsymbol{\theta} \mid F_j, k, \sigma)$ in (3) attains its minimum in the reference parameters $\boldsymbol{\theta}^{\text{ref}}$ on frame F_j, then this frame is suitable for calibration (note that no tracking is involved). We evaluate $L(\cdot)$ for each parameter θ_i independently, in the interval of $\left[\theta_i^{\text{ref}} - T^{\text{al}}, \theta^{\text{ref}} + T^{\text{al}}\right]$, using $T^{\text{al}} = 0.05$ and the step of 0.005 rad. Frame F_j is labelled unsuitable for calibration if the minimum in roll, pitch or yaw angle differs by more than 0.01 rad. This labelling was also done independently for the LT objective function by examining the maximum in the same way.

IV. Data

A. *Real Datasets* WaymoA *and* WaymoB

For our experiments on real-world data, we selected the Waymo open dataset [1] (labelled data, perception part, published in March 2020) as it is a large dataset (over 200,000 frames) with high data quality. We use data from the front, front-left, and front-right facing rolling-shutter cameras and the top mid-range LiDAR. All cameras have a horizontal field of view of 50.4° and a resolution of 1920×1280 px. The LiDAR has a 360° horizontal field of view with up to 2650 returned points in each scanline and a 20° vertical field of view with 64 scanlines.

The dataset comes in three parts: *train* (798 sequences), *test* (150 sequences) and *validation* (200 sequences), while each sequence contains around 200 frames. As the calibration procedure on arbitrary scenes may depend on the weather and lighting conditions, we use only sequences taken during the day and without rain. Since the weather information is unreliable[4], we labelled it manually. Thus, we removed 253 (102 rainy and 151 nightly) sequences from the WaymoB dataset. By merging the resulting test and validation subsets of the Waymo dataset into one, we created two datasets for the purpose of this paper:

- WaymoA: 238 sequences from the *validation* + *test* sub-sets (daylight, no rain),
- WaymoB: 545 sequences from the *train* subset (daylight, no rain).

The WaymoA dataset will be used for parameter learning (Section V), and the WaymoB will be used for experiments (calibration tracking and monitoring) in Section VI.

B. *Real* KITTI *dataset*

KITTI is one of the most well-known datasets in automotive [45], containing annotated subsets of high-quality data. The car is equipped with four global-shutter cameras and a single top-mounted LiDAR. Camera resolution is 1241×376 pixels after rectification, with around 65° horizontal field of view. All cameras look forward. The LiDAR is Velodyne HDL-64E, with 64 scanlines, a 26.9° vertical field of view, and approximately 1500 returned points per scanline. We use the raw, synchronized and rectified data recorded in a city

[4]see https://github.com/waymo-research/waymo-open-dataset/issues/210

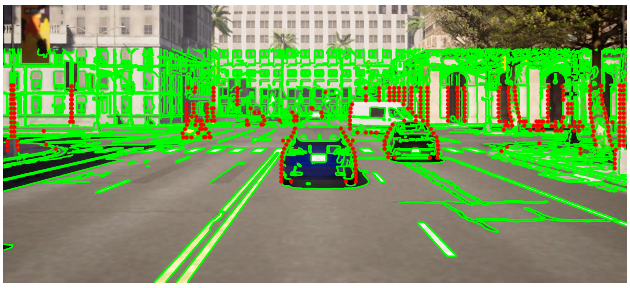
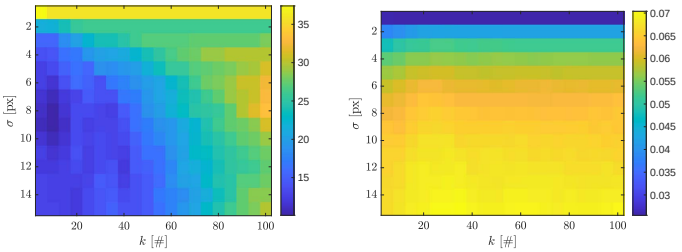


Fig. 8: A frame from the CARLA simulator. LiDAR corners are shown in red, image edges in green.



(a) Percentage of diverging frames. (b) Basin of attraction.

Fig. 9: Performance metrics for learning OCAMO parameters k and σ.

on the 26th of September 2011.[5] We will use this dataset for experimental evaluation only.

A recent study on visual stereo odometry [4] reported a rotational decalibration between stereo cameras at the beginning and end of the Sequence 01 in the Odometry Dataset.[6] We will confirm the decalibration hypothesis on the LiDAR-Camera pair in Section VI-D.

C. Synthetic Ground-Truth Dataset SYN

As there is currently no suitable public dataset of verified real decalibrations on realistic scenes, we created a synthetic decalibration dataset with simulated-world sensor data using the CARLA open-source simulator [46] based on the Unreal Engine. We simulated a 1920×1080 px global shutter camera with a $60°$ horizontal field of view. The simulated LiDAR has 64 scanlines, a $26.9°$ vertical field of view, and 4500 points per scanline.

We used a pre-created map TOWN10HD_OPT, added 100 autopilot agents, and recorded 155 sequences with 200 frames each, starting from a different spawn point on the map. One frame from this dataset is shown in Figure 8.

V. PARAMETER LEARNING

A. OCAMO Parameters

The preprocessing parameters used for feature extraction from both modalities were selected empirically. A list is given in the upper part of Table VII in Appendix B.

Parameters k and σ of the kernel correlation loss function (3) significantly impact the convergence and accuracy of tracked calibration parameters $\boldsymbol{\theta}$. We select them to achieve an accurate minimum of the loss function in the reference value $\boldsymbol{\theta}^{\mathrm{ref}}$ and a large basin of attraction around it. To this end, we performed a 2D grid search for $(k, \sigma) \in \{5, 10, \ldots, 95, 100\} \times \{1, 2, \ldots, 15\}$, using one frame from each sequence in the WAYMOA dataset (i.e. 238 frames in total). On each of these frames, we computed the loss function (3) for all three rotation parameters independently in the interval of $[-0.05, 0.05]$ rad with 0.005 steps. We evaluated two metrics: (1) The percentage of diverging frames with loss-function minima different from $\boldsymbol{\theta}^{\mathrm{ref}}$; see Figure 9a for the result (lower is better). (2) The average width of the basin of attraction over all rotation parameters computed on frames that achieved loss function minimum at $\boldsymbol{\theta}^{\mathrm{ref}}$, see Figure 9b (greater is better). The lowest divergence rate (around 15%) is achieved for $\sigma \in [7, 10]$ pixels and $k \in [5, 15]$. Increasing σ leads to a larger basin of attraction. However, narrow, distinguishable minima are more important for calibration monitoring. Thus, we selected $\sigma = 9\,\mathrm{px}$ and $k = 10$.

In this work, we will track and monitor rotational parameters (see Section VI-A). The learning of parameters related to translation could be performed similarly.

The three rotational variances $\sigma_{d_i}^2$ of the monitoring model were learned as follows. We executed the tracking method using the learned tracking parameters on WaymoA data. The mean quadratic difference between the tracked parameters $\boldsymbol{\theta}^{\mathrm{trk}}$ and the reference $\boldsymbol{\theta}^{\mathrm{ref}}$ was computed for each sequence. The $\boldsymbol{\sigma}_d^2$ vector is then obtained as the average over all the 238 sequences of these mean quadratic differences. Table VII then shows the square-root values σ_{d_i}, $i = 1, 2, 3$.

The users should define the choice of calibration monitoring thresholds T_θ based on their expected sensitivity of the monitoring system. This work sets these thresholds based on the estimated variances as $\mathbf{T}_\theta = 3\boldsymbol{\sigma}_d$. The parameters' boundary in (11b) is set larger, as $\theta^{\mathrm{bnd}} = 5\boldsymbol{\sigma}_d$.

The last parameter is the update boundary, which we set as the horizontal resolution of the LiDAR:

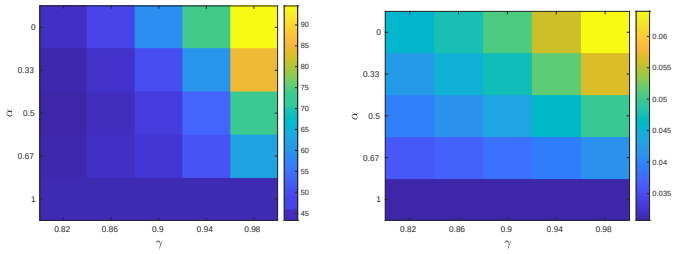
$$\theta_i^{\mathrm{upd}} = \frac{2\pi}{2650} \approx 0.0024\,\mathrm{rad}\,, \quad i = 1, 2, 3. \tag{16}$$

B. LTβ Parameters

Experiments in the original LT paper [8] were conducted with a very different camera and LiDAR; thus, we need to choose new preprocessing parameters. To estimate the edge preprocessing parameters α and γ, we use the same approach as for k and σ in Section V-A. The resulting metrics are shown in Figure 10. The γ parameter behaves similarly to the σ in OCAMO: Its increase widens the basin of attraction, but

[5] Specifically, we use sequences with at least 150 frames: 5, 9, 11, 13, 14, 18, 51, 56, 57, 59, 84, 91, 93, 95, 96, 104, 106, 117.

[6] Sequence 42 from 3rd of October 2011 in the raw dataset.



(a) Percentage of diverging frames. (b) Basin of attraction.

Fig. 10: Performance metrics for learning LTβ parameters α and γ.

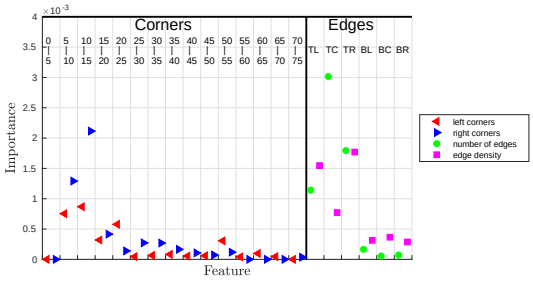


Fig. 11: Importance of LiDAR corners (left part) and image edges (right part) for frame preselection. LiDAR corners are sorted according to the z-range, and image edges are organized lexicographically from the top-left segment to the bottom-right segment.

it may also negatively affect the sharpness of the maxima. The α parameter affects the strength of the neighborhood edges. As in the original paper, we selected $\alpha = \frac{1}{3}$ but decreased γ to 0.86.

For the LiDAR corner preprocessing, we used the original exponent $\gamma = 0.5$, but we filtered out points with depth discontinuity smaller than 1 m instead of the original value of 0.3 m. We observed that this rule helped improve the results of the LTβ method. As in the original paper, the number of frames w is set to nine. The grid size for the computation of the 728 perturbations $\mathbf{e}_l$ around the reference was selected as ± 0.01 rad for the rotation parameters and ± 0.1 m for the translation parameters. These values correspond to the lower bound of the synthetic decalibration (see (17) below and Section VI-C). In the calibration tracking of rotation parameters, we used perturbations of ± 0.0005 rad, corresponding to the lowest synthetic rotational drift (see Section VI-F).

Parameters (α_c, β_c) and (α_d, β_d) of the beta distributions p_c and p_d in (15) were learned on WAYMOA calibrated data with 9-frame window averaging. This gave $\alpha_c = 8.69$ and $\beta_c = 0.367$. The parameters of p_d were learned on the WAYMOA sequences again, each synthetically decalibrated by a uniformly distributed random decalibration on the interval

$$\begin{aligned}&[-0.02, -0.01] \cup [0.01, 0.02] \text{ rad in rotations and} \\ &[-0.2, -0.1] \cup [0.1, 0.2] \text{ m in translations.}\end{aligned} \tag{17}$$

This gave $\alpha_d = 4.12$ and $\beta_d = 4.40$.

C. LTO Parameters

The preprocessing and model parameters will be the same as for OCAMO (see Section V-A). Learning (α_c, β_c) and (α_d, β_d) for LT-based monitoring with the OCAMO model (3) was done as in the previous section (see Section V-B). It gave: $\alpha_c = 40.6, \beta_c = 0.203, \alpha_d = 4.08$, and $\beta_d = 3.70$.

D. Frame Preselection Parameters

We need to train a classifier that will select input frames suitable for calibration. The training is performed on the WAYMOA dataset. We start with the extraction of 42 features described in Section III-G1. Afterwards, using the method from Section III-G2, we automatically label each frame in the dataset, using (3) for OCAMO/LTO and (14) for LT. This is done for each camera separately, as the scene views have different characteristics. This way, we get one set of features and two sets of labels per camera. We train two distinct frame preselectors, one per objective/loss function.

There are 47 202 frames in 238 sequences, with around 200 frames each. Since the individual sequences are relatively homogeneous, we used 5-fold cross-validation over sequences instead of frames. We trained the LogitBoost [47] classifier using ensembling methods on classification trees from the MATLAB® Statistics and Machine Learning Toolbox.

Figure 11 shows the importance of each predictor (feature) for our trained ensemble (OCAMO and the front camera). Each predictor is visualized as a particular color dot based on its type. These results show that the most important corners are those between 5 to 15 m distance from the vehicle. This confirms our hypothesis that OCAMO needs near objects to work well. As we also hypothesized, the most important image edge predictors were found around the horizon (the top row of the image features in Figure 7b). In the data we used, the central part of the image contained the most important information for the calibration task. Predictors for LT behave similarly.

Let $s : X \to [-\infty, \infty]$ be the classification score function specific for the classifier, where X is the feature vector of a data frame and positive values of $s(X)$ favor frames suitable for calibration. Using a threshold T_p, we define a classifier $C : X \to \{0, 1\}$

$$C(x) = \mathbb{1}[s(x) \geq T_\mathrm{p}]\,. \tag{18}$$

We call T_p the *preselection level* and choose it so that the frame preselector removes a specific percentage of frames from a given dataset, which we call *data loss*.

VI. Experiments

A. Calibration Monitoring based on Rotation Parameters Tracking

A camera has six extrinsic parameters and additional five or more intrinsic parameters, depending on the number of radial/tangential distortion coefficients. It is not necessary to track all these parameters in real-time. For instance, radial distortion changes could be partially observed as changes in focal length. Moreover, sensor fusion with sensor setups used in the automotive domain is not critically dependent on small changes in translation parameters, unlike the changes in rotation parameters, as discussed in [8]. Therefore, we first test the hypothesis that it is sufficient to monitor rotational parameters to detect decalibration in translation or focal length (or, equivalently, the horizontal field of view, HFoV). We will use OCAMO for this purpose. We empirically observed that the findings apply to other methods (i.e., LTβ and LTO), too.

We used the synthetic dataset SYN from Section IV-C. We decalibrated the sensors between frames 51 and 110, as described below.

The decalibration will be done in three sets of parameters: (1) rotation parameters δ_{rot}, (2) translation parameters δ_{tr}, and (3) horizontal field of view δ_{hfov}. We investigate three different magnitudes of decalibrations: (1) small δ^{s}, (2) medium δ^{m}, and (3) large δ^{l}. The decalibration is randomly drawn from a uniform distribution on the interval

$$\left([-2\delta, -\delta] \cup [+\delta, +2\delta]\right)^d \quad \text{rad, m, or } ^\circ,$$

where $d = 3$ for rotations and translations and $d = 1$ for δ_{hfov}. The small decalibrations are

$$\delta^{\text{s}}_{\text{rot}} = 0.00125\,\text{rad},\ \delta^{\text{s}}_{\text{tr}} = 0.05\,\text{m},\ \text{and}\ \delta^{\text{s}}_{\text{hfov}} = 0.5^\circ.$$

These are then scaled up as

$$\delta^{\text{m}} = 2\delta^{\text{s}} \text{ and } \delta^{\text{l}} = 4\delta^{\text{s}}.$$

The decalibrations are introduced by altering the camera projection matrix.

Figure 12 shows the evolution of the calibration validity index for all three parameter sets and magnitudes (nine partially overlapping curves), averaged over all 155 sequences from SYN with ten different, random decalibrations, one per sequence. The smallest-magnitude decalibrations (dotted lines) demonstrate the breaking point of the validity index. Although the index is still under 0.5 on average, smaller decalibrations would no longer be reliably detected by the index. The smallest magnitude of the translational decalibration (0.05 – 0.1 m) reflects the lower sensitivity of the reprojection error to translation when the scene is distant from the sensors, as usual in automotive traffic scenes.

Detecting the medium extrinsic decalibration (red and green dashed lines) is faster than detecting the large-magnitude one (solid lines). This is caused by the filter (8),

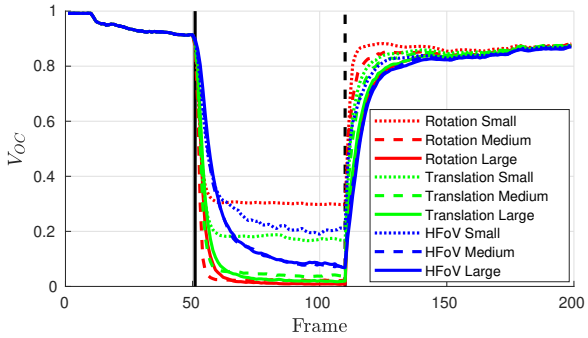


Fig. 12: OCAMO calibration validity index V_{OC} based on rotation parameters tracking on the SYN dataset when the true decalibration is not necessarily rotational. The respective decalibration starts at frame 51 (solid black line) and ends at frame 110 (dashed black line).

whose memory needs a shorter time to adapt to the medium decalibration than the large one. The variant steepness of the loss function within and outside the basin of attraction (Figure 3) further influences the behaviour via the memory update rule (9). After stabilisation, the order of V_{OC} values is inverse to the order of the decalibration magnitude, as expected.

We conclude that monitoring decalibration by tracking rotation parameters is possible. All subsequent results of OCAMO are therefore reported with rotation parameters tracking only. This decision is also partly influenced by a suspicion of a strong correlation between some rotation and translation parameters (later indicated by the experiments in Section VI-C).

B. Calibration Monitoring on Real Decalibrations

The WAYMOB dataset was recorded in several locations, mostly in San Francisco and Phoenix [1]. Even in temporally consecutive sequences (within a day) from the same location, the reference calibrations provided with the dataset were sometimes different. We found 139 such consecutive sequence pairs from Phoenix and 53 from San Francisco by examining the reference calibrations. Figure 13 shows the distribution of LiDAR-Camera decalibrations between the consecutive sequence pairs. One can see that the translations were stable, mostly changing by a couple of millimetres, but the rotation fluctuated considerably even in the short timespans considered here.

We concatenated the 192 pairs of 200-frame sequences into 600-frame sequences (first, second, and the first again). The calibration parameters of the first sub-sequence were used as the reference for the rest of the concatenated sequence. Hence, each 600-frame sequence becomes decalibrated after frame 200 and calibrated again after frame 400. The decalibrations are real and pseudo-random. The resulting dataset,

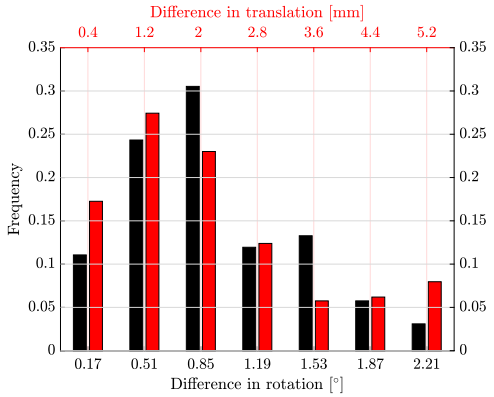


Fig. 13: Histogram of real decalibrations in translation (top axis, red) and rotation (bottom axis, black) occuring in the WAYMOBDIF dataset.

called WAYMOBDIF, tests the methods' ability to detect decalibration and recalibration.

Besides the consecutive sequences of differing calibrations in WAYMOB, there are also those preserving the calibration, specifically 17 from Phoenix and 200 from San Francisco. We call this subset of 217 sequence pairs WAYMOBEQ. Their calibrations are thus valid throughout.

The remaining pairs of consecutive sequences from WAYMOB were from different locations or temporally too distant (more than a day). These were not considered.

We executed OCAMO, LTβ and LTO on the sequences from WAYMOBEQ and WAYMOBDIF for several levels of frame preselection. If the frame preselector does not select at least 50 frames of one of the two concatenated sequences, we do not use this sequence pair for that preselection level.

Figure 14 illustrates the performance of all methods on WAYMOBDIF (w/o frame preselection). The abscissa of each plot shows the frame index in the sequence. The mean and the 15 % and 85 % quantiles[7] of the respective validity index V over all WAYMOBDIF sequences are shown at each frame. Ideally, the validity index should be close to unity on the calibrated parts (frames 1–200 and 401–600) and close to zero on the decalibrated part (frames 201–400). The further apart the means relative to the variances in the calibrated and uncalibrated parts of the sequence, the better the method performs.

On the calibrated parts of the sequences, OCAMO shows a higher variance than the other two methods. All three methods reacted on the decalibration and recalibration (solid lines). Nevertheless, the LT-based validity indices (LTβ and LTO) have a much larger variance than OCAMO on the decalibrated part. Some of the decalibrations (see Figure 13) were probably too small (hence hard to detect) for these

[7]This would correspond to $\pm\sigma$ of a Gaussian random variable.

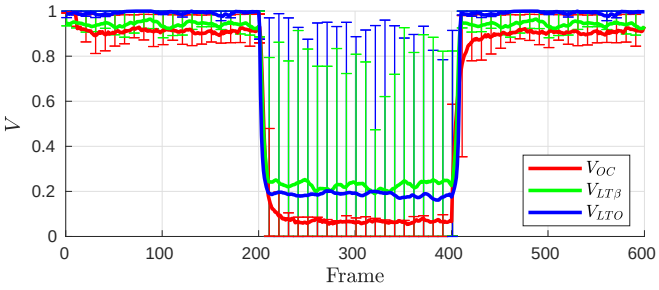


Fig. 14: Calibration validity indices on decalibrated sequences from WAYMOBDIF for all tested methods without any frame preselection.

TABLE I: The statistical accuracy of the calibration monitoring decision for all three methods based on thresholding $V_{\mathrm{OC}}, V_{\mathrm{LT}\beta}$, and V_{LTO} with 0.5, shown for four frame preselection levels influencing data loss. The best accuracy for each preselection level is emphasized.

	Data Loss	WAYMOBEQ	WAYMOBDIF	Average
OCAMO	0 %	0.9816	**0.9589**	**0.9703**
	20 %	0.9890	**0.9595**	**0.9743**
	40 %	0.9900	**0.9702**	**0.9801**
	60 %	0.9904	**0.9650**	**0.9777**
LTβ	0 %	0.9669	0.8959	0.9314
	20 %	0.9700	0.9070	0.9385
	40 %	0.9726	0.9124	0.9425
	60 %	0.9851	0.9125	0.9488
LTO	0 %	**0.9947**	0.9344	0.9646
	20 %	**0.9979**	0.9330	0.9655
	40 %	**0.9987**	0.9370	0.9679
	60 %	**1.0000**	0.9229	0.9615

two methods. Making the search grid finer in the LT-based validity could improve the results. However, a too-fine grid would also increase the probability of a false alarm in calibrated cases or make monitoring more difficult on larger decalibrations.

Validity indices in Figure 14 are a basis for the decision on camera-LiDAR system calibration status. We set the decision threshold to 0.5 (if the index V is higher, the system reports valid calibration; otherwise, it reports decalibration). For the WAYMOBEQ dataset, the validity index should be higher than 0.5 everywhere. For WAYMOBDIF, the index should be lower than 0.5 between frames 201 to 400 of each sequence and higher otherwise. Using this rule, we evaluated the statistical accuracy of each tested method, as shown in Table I (higher values are better). To allow the monitoring methods to adapt to the calibration change, we did not include the first ten frames after each change (201–210 and 401–410) in the accuracy evaluation, as well as frames 1–10 that fall in the burn-in interval of OCAMO.

Table I reveals noticeable differences between the WAYMOBEQ and WAYMOBDIF accuracies. LTO performs best when the calibration remains constant in WAYMOBEQ and

OCAMO when the calibration changes in WAYMOBDIF. Of the two, OCAMO has a more balanced performance, which is also the best on average (last column). This is consistent with our earlier observations from Figure 14 that OCAMO can detect smaller decalibrations.

Table I also shows that frame preselection does increase the accuracy of all three methods on the WAYMOBEQ dataset (second column). On the other hand, due to the removal of short sequences after preselection in WAYMOBDIF, the distribution of decalibrations changes. Hence, if the preselection removes too many easy-to-detect decalibrations, the monitoring becomes harder, resulting in lower accuracy. This happened for OCAMO and LTO (third column and 60 % data loss).

C. Calibration Monitoring on Large Synthetic Decalibrations

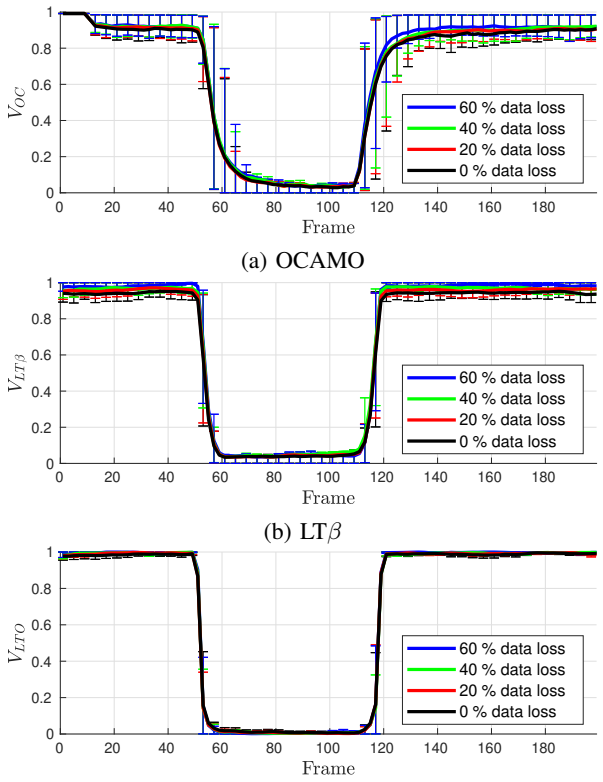
The observed decalibrations between temporally close sequences discussed in Section VI-B are often very small and variant. Therefore, we cannot see the monitoring performance and stability on larger decalibrations. Thus, in this section, we simulate a larger synthetic decalibration in a time subinterval of each WAYMOB sequence individually and compare OCAMO, LTβ and LTO methods on these. Although the levels of decalibrations considered here are not typical in real driving scenarios unless an accident occurs, we use them to demonstrate the robustness and stability of the monitoring process.

We took all 545 sequences from WAYMOB and decalibrated the pose of the LiDAR relative to the camera between frames 51 and 110. Each sequence uses a different synthetic random decalibration (17), exhausting all rotation and translation parameters.

This dataset will be denoted as WAYMOBSYNDEC, and the performance of each method will be compared with the original calibrated sequences from WAYMOB on several levels of frame preselection. As in the previous experiment with consecutive sequences, if some sequence in WAYMOB and WAYMOBSYNDEC does not have 50 frames after preselection, it is not considered for that preselection level in all subsequent evaluations.

The behaviour of a monitoring method should follow the same pattern as in the real-data experiments. Figure 15a shows the performance of OCAMO on WAYMOBSYNDEC. Let us compare these results with those on WAYMOBDIF in Figure 14. We can see that OCAMO took longer to notice and to adapt to the decalibration and the recalibration than on WAYMOBDIF. The larger decalibration magnitude is the cause. Still, OCAMO has a well-defined decision boundary (0.1 – 0.8) between the calibrated and decalibrated parts of the WAYMOBSYNDEC sequences. Hence, it should yield a good per-frame accuracy, even on larger decalibrations.

Compared to the real but smaller decalibrations (Figure 14), the larger synthetic decalibration was easier to detect



(a) OCAMO

(b) LTβ

(c) LTO

Fig. 15: Calibration validity indices on synthetic decalibrations of the WAYMOBSYNDEC sequences for several levels of frame preselection and the three tested methods.

for both LTβ (Figure 15b) and LTO (Figure 15c). This corroborates our earlier presumptions. As these two methods do not have any tracking in the monitoring mechanism, the detection of decalibration (after frame 50) and recalibration (after frame 110) is nearly instant (depending on the window size), compared to OCAMO (Figure 15a). LTO achieved the fastest and the most stable detections. The frame preselection helps to increase the lower 15 % quantile on the calibrated parts of the sequence and the recalibration speed of OCAMO.

As in the previous experiment, we evaluated the accuracy of all methods using the same validity index threshold of 0.5. On the calibrated sequences of WAYMOB, the validity index should be larger than 0.5 for all 200 frames. On the synthetically decalibrated sequences in WAYMOBSYNDEC, the validity index should be larger than 0.5 on frames 1–50 and 111–200 and smaller than 0.5 between frames 51 and 110. We again did not include the transition phases (51–60 and 111–120) or the burn-in interval (1–10).

The results are shown in Table II. Consistently with Figure 15, the LTO achieved the best results on both the calibrated (WAYMOB) and synthetically decalibrated (WAYMOBSYNDEC) datasets.

On WAYMOB, OCAMO achieved comparable results to

TABLE II: Statistical accuracy of the calibration monitoring decision for all three methods on WAYMOB and WAYMOBSYNDEC, shown for four frame preselection levels.

	Data Loss	WAYMOB	WAYMOBSYNDEC	Average
OCAMO	0 %	0.9698	0.9491	0.9595
	20 %	0.9773	0.9559	0.9666
	40 %	0.9835	0.9610	0.9723
	60 %	0.9843	0.9682	0.9762
LTβ	0 %	0.9495	0.9564	0.9529
	20 %	0.9656	0.9678	0.9667
	40 %	0.9741	0.9725	0.9733
	60 %	0.9882	0.9839	0.9860
LTO	0 %	**0.9897**	**0.9894**	**0.9895**
	20 %	**0.9929**	**0.9931**	**0.9930**
	40 %	**0.9940**	**0.9945**	**0.9943**
	60 %	**0.9963**	**0.9964**	**0.9963**

LTO but did not perform so well on WAYMOBSYNDEC. In Figure 15a, we already noticed that OCAMO needs more time to detect decalibration and recalibration than the other two methods. This is caused by the memory (9) stabilising the tracker but slowing down the detector. Larger decalibrations lead to slower transitions, making for the most significant difference between OCAMO and LTO accuracies on WAYMOBSYNDEC. When OCAMO tracks all six parameters, its monitoring accuracy drops by about 10 % due to the increased rate of false alarms. We explain this behaviour by correlations between rotation and translation parameters.

LTβ achieved good results on WAYMOBSYNDEC but did not perform well on WAYMOB. That was most likely because the proposed LTβ model did not work correctly on around 2-3 % of sequences. Nevertheless, this issue was solved by the frame preselector: The LTβ was comparable to LTO after the removal of 60 % of the data. Overall, the proposed frame preselection helped all three methods on both datasets.

Besides the statistical accuracy, we also evaluated each method's false positive rate (FPR) and true positive rate (TPR) metrics [48] and visualized the ROC curves in Figure 16. The worst-performing method is OCAMO (blue), with the lowest area under the curve (AUC). Again, the tracking mechanism's memory is the culprit.[8]

The original LT (red) method did slightly better than OCAMO (based on AUC), but its TPR could go no higher at a certain point. This was caused by a too small standard deviation σ_c of the normal distribution proposed in the original LT method [8]. This insufficiency was solved by our proposal to use beta distribution. The experiment shows that the modification LTβ of the original LT method is indeed effective (compare the red and yellow lines). Overall, the LTO did achieve the best AUC and the highest statistical accuracy over both WAYMOB and WAYMOBSYNDEC datasets.

[8]We discarded only ten frames around the change: 51–60 a 111–120.

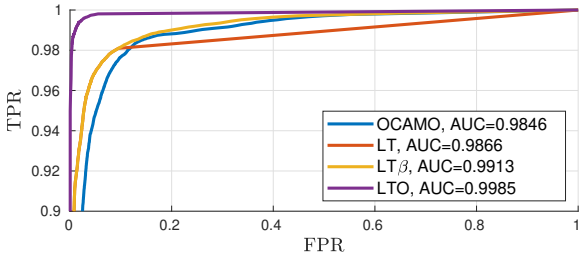


Fig. 16: ROC curves for all calibration monitoring methods on WAYMOBSYNDEC with no frame preselection.

D. Calibration Monitoring on The KITTI Dataset

Besides the Waymo dataset, we also evaluated the calibration monitoring methods on the KITTI data. In order to verify the portability of each method, we did not re-learn the parameters of the validity index function $\boldsymbol{\sigma}_d$, (α_c, β_c), and (α_d, β_d). However, we had to change some other parameters, as the KITTI sensors have different properties. First, as the horizontal field of view of the KITTI cameras is wider and the resolution lower, we decreased the σ parameter of OCAMO (and LTO) to $\sigma = 3$ and the γ parameter of LTβ image preprocessing to $\gamma = 0.4$. Second, we observed too many outliers in the LiDAR corners, so we increased $T^{\mathrm{cd}} = 0.03$ of OCAMO (and LTO) and filtered out points with depth discontinuity smaller than 3 m for LTβ.

We constructed 1000-frame series from each of the 18 KITTI sequences (those shorter than 1000 frames were re-used cyclically). We did not decalibrate the sensors for the initial 50 frames. The decalibration/recalibration then alternates every 70 frames until the last frame. In brief, random decalibration from (17) will be applied on frame i if

$$i > 50 \wedge \mathrm{mod}(i - 50, 141) < 71.$$

We executed all three calibration monitoring methods on these 1000-frame sequences and evaluated their statistical accuracy over all frames. As in the case of the WAYMOB, we did not include the transitions (ten frames) after each change in the accuracy computation. The statistical accuracies are as follows:

OCAMO	LTβ	LTO
0.9267	0.9705	0.9842

One can see that OCAMO could not generalize well on the KITTI data. Besides the slower change detection, which was already apparent with the WAYMOBSYNDEC, it is also partly caused by the $\boldsymbol{\sigma}_d$ parameters of the tracking mechanism. These should have been re-learned for the specific LiDAR-Camera pair. LTO achieved the best statistical accuracy, again showing the overall strength of the model (3). These results indicate that the fixed gridding in LT-based validity indices has better portability than the OCAMO tracking. This has to be taken with a grain of salt since when we used the original LT parameters from [8], the accuracy dropped

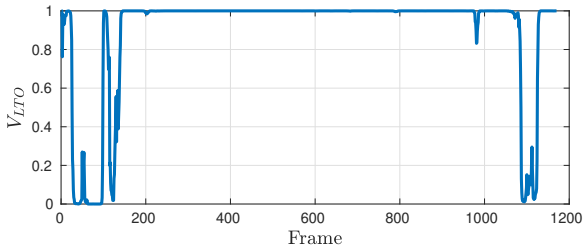


Fig. 17: LTO calibration monitoring on a highway sequence from the KITTI dataset discussed in [4].

TABLE III: Statistical accuracy of the calibration monitoring decision for the LTO on WAYMOBSYNDEC with front-left (left column) and front-right (right column) cameras, shown for four frame preselection levels.

	Data Loss	Front-left	Front-right
LTO	0 %	0.9854	0.9688
	20 %	0.9892	0.9750
	40 %	0.9889	0.9830
	60 %	0.9896	0.9853

below 0.9. We conclude that using any of these methods with parameters trained on some other sensor set limits their potential calibration monitoring possibilities. The parameters need to be re-learned for a specific sensor setup.

To confirm a LiDAR-Camera decalibration in the Sequence 01 of the Odometry Dataset reported in [4], we run the LTO monitor on the sequence. Figure 17 shows an apparent decalibration at the beginning (up to frame 150) and the end (around frame 1100). We also found Spearman's rank correlation of -0.45 between V_{LTO} and the linear acceleration of the vehicle. A further study would be needed to identify the exact cause of the decalibration.

E. Calibration Monitoring on Left- and Right-Facing Cameras

So far, we have tested forward-facing cameras in all experiments. We will now test the calibration monitoring accuracy on the front-right and front-left facing cameras from the WAYMOBSYNDEC dataset. We will use only LTO, which has exhibited superior results in the experiments. We kept the preprocessing parameters, as the sensors are the same. Besides the monitoring parameters (α_c, β_c), (α_d, β_d), we also trained new frame preselectors for each camera. We noticed an interesting difference between these two cameras; therefore, this experiment will be split into two parts.

1) Front-Left Facing Camera: Table III (left column) shows the statistical accuracy of LTO monitoring for the front-left camera on WAYMOBSYNDEC. These results are similar to those of the front-facing camera in Table II (WAYMOBSYNDEC column). The difference in statistical accuracy is $0.3-0.7\,\%$ (depending on the preselection level).

2) Front-Right Facing Camera: The results for the front-right camera in Table III are visibly worse. By examining the data, we can see that it often contains close bushes, trees or plain walls. These are naturally unsuitable for targetless, corner-based calibration because they provide ambiguous or insufficient information. The problematic frames were mostly removed by the preselection, which resulted in a 1.5% accuracy increase at the cost of a greater data loss, which could lead to a slower reaction to a sudden decalibration.

These results illustrate that monitoring other non-front-facing cameras with only a small drop in efficiency is possible. The frame preselection also removes problematic frames that could cause false decalibration alarms.

F. Calibration Drift Tracking

Besides the abrupt decalibration detection of the previous experiments, we also consider calibration parameter drift: A slow, gradual change that could be considered a random process.

Calibration drift tracking was first considered in [8], where they showed results on a simulated drift. They performed a synthetic, incremental, random decalibration at each new frame by $\pm 0.02°$. Their system followed this change with an average mean absolute error of $0.1°$ for the rotation parameters. These results were obtained on different data and would probably not be replicable on WAYMOB, where the sensors have different resolutions and fields of view.

We created the WAYMOBSYNDRIFT dataset containing 545 sequences from WAYMOB, each 1 500 frames long (shorter sequences from WAYMOB were extended cyclically, as in the KITTI data of the experiment in Section VI-D). We tested the calibration drift tracking at four preselection levels (0, 20, 40 and 60 % data loss). As before, sequences with fewer than 50 selected frames were discarded from this experiment. At each frame, we independently performed an incremental, random decalibration by ± 0.0005 rad in each rotation parameter.

1) Drift Tracking Precision: We evaluated the mean absolute rotational error (MAE) from the reference calibration for OCAMO, LT and LTO in Table IV. We also visualized the average MAE over all the rotation parameters in Figure 18.

Without any preselection, the OCAMO tracker achieves $0.13°$ overall MAE. The easiest-to-track yaw angle achieves an error of $0.0579°$, about half of the horizontal resolution of the used LiDAR. The error drops with the preselection (down to $0.0308°$ for yaw), but removing more than 20 % of frames does not help significantly: The average MAE is under $0.1°$.

LT did achieve almost two times higher average MAE ($0.25°$) than OCAMO. The frame preselection was quite effective in this method, as the average MAE decreased to $0.15°$, with the yaw angle MAE decreasing by more than half to $0.0428°$. The other angles decreased significantly as well, especially the pitch.

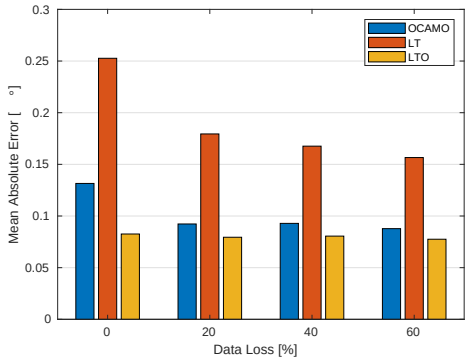


Fig. 18: Average MAE in the calibration drift tracking of all rotation parameters on the WAYMOBSYNDRIFT dataset.

TABLE IV: The MAE in the calibration drift tracking of all rotation parameters on the WAYMOBSYNDRIFT dataset.

	Data Loss	roll [°]	pitch [°]	yaw [°]
OCAMO	0 %	0.2155	0.1212	0.0579
	20 %	0.1571	0.0851	0.0347
	40 %	0.1584	0.0865	0.0337
	60 %	0.1517	0.0807	0.0308
LT	0 %	0.3365	0.3152	0.1064
	20 %	0.2687	0.2015	0.0681
	40 %	0.2690	0.1814	0.0525
	60 %	0.2522	0.1747	0.0428
LTO	0 %	0.1426	0.0788	0.0263
	20 %	0.1375	0.0755	0.0253
	40 %	0.1394	0.0779	0.0243
	60 %	0.1347	0.0762	0.0217

The LTO results show that the loss function (3) is much better for drift tracking than LT. LTO did not benefit from frame preselection, most likely because the grid step of 0.0005 rad was not large enough to walk outside the basin of attraction (as opposed to OCAMO). LTO results are slightly better (by $0.01°$) than those of OCAMO with frame preselection. However, this has to be taken with a grain of salt since LTO (and LT) benefited from the drift magnitude being similar to the grid search step. OCAMO performs better when the decalibration drift variance is unknown and not uniform, as will be the case in practice.

As calibration tracking and offline calibration are different mechanisms, we cannot directly compare their geometric precisions. Nevertheless, the tracking accuracy of the LTO method is comparable to the accuracy of a feature-based targetless offline calibration based on a similar model (GMM) published on Ford data [31]. If we take into account the different resolutions of cameras σ_{cam} (degrees per pixel), LiDARs σ_{lid} (degrees per ray), and the (independently) combined total resolution σ_{tot}, we get the results shown in Table V. The performance numbers to be compared are the relative errors $\mathrm{yaw}/\sigma_{\mathrm{tot}}$ (last column). Since the difference is small, we can say the results are comparable.

TABLE V: A comparison of precision in yaw angle between LTO tracking and the calibration method from [31] (last column).

	σ_{cam}	σ_{li}	σ_{tot}	yaw [°]	$\mathrm{yaw}/\sigma_{\mathrm{tot}}$
LTO, Waymo	0.027	0.136	0.138	0.026	**0.188**
[31], Ford	0.352	0.200	0.404	0.223	0.552

TABLE VI: Preselection reduces the drift tracking divergence (lower is better).

Method	Data Loss			
	0 %	20 %	40 %	60 %
OCAMO	0.1248	0.1069	0.0921	0.0726
LT	0.3101	0.2611	0.2292	0.2143
LTO	0.1064	0.099	0.0813	0.0726

2) Drift Tracking Divergence: We also examined the long-term stability of drift tracking. To this end, sequences in which the tracking MAE exceeded $0.25\,°$ in any parameter were considered *diverging*. An example of such a sequence is shown in Figure 6c. One can see in Table VI that the fraction of sequences from the WAYMOBSYNDRIFT dataset in which the tracking diverged was consistently decreasing with the increasing preselection level for all the methods. One example of such a diverging sequence, which was removed by the frame preselector, is visualized in Figure 6c.

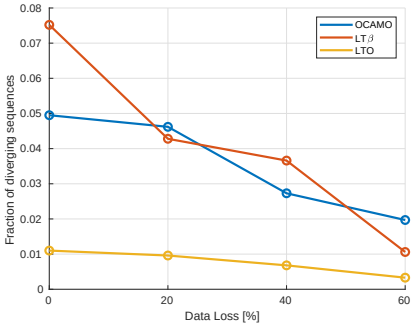
We conclude that the frame preselection increases both the statistical accuracy of each tracked parameter and the stability of the calibration tracking procedure.

G. Frame Preselection Evaluation

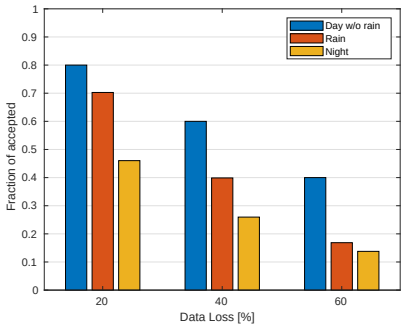
The statistical evaluations above show that frame preselection helps the accuracy and stability of all investigated monitoring methods. However, the preselection may eliminate too many frames, which may result in losing track of the calibration. Preselection may also fail by not eliminating frames or sequences that we consider unsuitable for calibration, like in severe rain or insufficient illumination. This section sheds some light on these two types of failure.

1) Recalibration Divergence: The calibration parameter tracking might not catch up after extended periods of low-quality data. We examined the frequency of such failures by comparing the data loss (fraction of frames removed by the preselection) in the WAYMOB dataset and the number of *diverging sequences*: We say a sequence from WAYMOBSYNDEC is diverging if the method's validity index reported a value lower than 0.5 on more than 25 % of the frames after the recalibration (frames 151–200). The lower the divergence rate, the higher the ability to recalibrate after a decalibration event. Thus, an efficient frame preselection minimizes the fraction of diverging sequences while removing as small a portion of data as possible.

Figure 19a shows the fraction of diverging sequences (ordinate) as a function of data loss (abscissa) induced by preselection. The fraction of diverging sequences decreases



(a) Preselection reduces the recalibration divergence



(b) Night and rain frames acceptance

Fig. 19: Recalibration divergence versus data loss for all three methods at four preselection levels (a). Acceptance of night and rainy frames by frame preselection and the acceptance on WAYMOB for comparison (day without rain) (b).

with increasing preselection level. Specifically, the fraction of diverging sequences decreases from 7.5 % to 1 % for LTβ while removing only 60 % of frames. Frame preselection also positively affects OCAMO, as the fraction decreased from 5 % to 2 %. LTO has a superior recalibration ability both with and without frame preselection.

We conclude that preselection is effective for all methods: Not only it improves calibration accuracy, as shown in the earlier experiments, but it also improves the tracking success after a decalibration/recalibration event.

2) Rainy or Night Frame Elimination: We apriori excluded sequences captured during the night and/or rain, as discussed in Section IV-A. An interesting question is whether the frame preselection mechanism can handle such exclusion automatically. We take the 151 night and 102 rainy sequences not included in the learning and evaluation data. We then run the same preselectors used for WAYMOB sequences on them.

Figure 19b shows the corresponding results (red and orange) and the sequences from WAYMOB for comparison (blue). The bars show the fraction of accepted frames in each sub-category. We can see that the night frames have around half the acceptance rate of the WAYMOB frames (for all preselection levels). These results show that most night scenes are probably unsuitable for calibration. Frames captured during rain have a higher acceptance rate than the night ones. Most of this data is not heavily corrupted by the rain, except perhaps when raindrops are stuck to the camera lens. Frame preselection could remove these cases, and the calibration might still be possible.

VII. CONCLUSIONS

In this work, we presented two novel methods (OCAMO and LTO) for online calibration monitoring and tracking for Camera-LiDAR systems, together with a modification of the state-of-the-art approach LTβ. We also introduced a frame preselection concept and have shown that it improved the accuracy and robustness of the calibration process for all the studied methods. All three methods use easily extractable low-level features and have a small computational and memory overhead, which makes them suitable for real-time implementation and easy integration into an existing vision system. The main difference between OCAMO and the LTO and LTβ is that OCAMO uses parameter learning based on a formally well-grounded online stochastic optimization with seven loss function evaluations per frame. In contrast, the LT-based methods use a simple grid search, requiring 729 objective function evaluations, cf. Figure 2.

Based on the experimental results, we conclude that the best-performing method in monitoring and tracking tasks was LTO. This is surprising since it performs no data filtering. A detailed look shows, however, that the performance of LTO depends on two critical parameters – the grid step size and the window size – that significantly affect the precision and statistical accuracy, as already discussed in the original paper [8]. We used the grid step equal to the simulated drift increment (speed), favouring the LT-based methods in the evaluations. When one does not know how fast the drift will be in real-world scenarios, the grid step might be set too small (leading to a slow response) or too large (leading to imprecisions). Automatic learning of an adaptation rule for the grid step is a topic for further research, requiring a new large dedicated dataset with real decalibrations.

In contrast, the OCAMO parameters are not so critical, except for the memory size that acts as a filter but slows down the decalibration detection. This slowdown was why OCAMO did not perform as well as the LTO in the evaluations. A fair comparison would require longer sequences that were not available to us.

Frame preselection increased the accuracy of all the studied monitoring methods on both real and synthetic decalibrations. This increase was several percentage points, depending on the preselection level and the method. The preselection was also effective for non-front-facing cameras, where it removed problematic frames, especially from the front-right-facing one. The preselection also decreased the number of diverging sequences after recalibration by almost an order of

magnitude for LTβ, and for the other methods, it also helped. We also found it effective for identifying problematic night and rain frames. However, due to the systematically different views of the traffic scenes, the preselection did not even out the differences in the average monitoring accuracy between the front and front-right-facing cameras.

Better frame preselection, the creation of large datasets dedicated to online monitoring and calibration tasks, and performance evaluation strategies are the topics for further research in calibration monitoring and tracking.

REFERENCES

[1] P. Sun, H. Kretzschmar, X. Dotiwalla *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2446–2454.

[2] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[3] X. Li, Y. Xiao, B. Wang, H. Ren, Y. Zhang, and J. Ji, "Automatic targetless lidar–camera calibration: a survey," *Artificial Intelligence Review*, pp. 1–39, 2022.

[4] I. Cvišić, I. Marković, and I. Petrović, "SOFT2: Stereo visual odometry for road vehicles based on a point-to-epipolar-line metric," *IEEE Transactions on Robotics*, vol. 39, no. 1, p. 273–288, 2023.

[5] L. Xing, W. Dai, and Y. Zhang, "Improving displacement measurement accuracy by compensating for camera motion and thermal effect on camera sensor," *Mechanical Systems and Signal Processing*, vol. 167, p. 108525, 2022.

[6] M. Elias, A. Eltner, F. Liebold, and H.-G. Maas, "Assessing the influence of temperature changes on the geometric stability of smartphone- and Raspberry Pi cameras," *Sensors*, vol. 30, no. 3, 2020, Art. no. 643.

[7] T. Läbe and W. Förstner, "Geometric stability of low-cost digital consumer cameras," in *Proceedings of the ISPRS Congress*, 2004, pp. 528–535.

[8] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers," in *Proceedings Robotics: Science and Systems Conference*, 2013, Art. no. 29.

[9] J.-K. Huang and J. W. Grizzle, "Improvements to target-based 3D LiDAR to camera calibration," *IEEE Access*, vol. 8, pp. 134 101–134 110, 2020.

[10] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[11] C. Glennie and D. D. Lichti, "Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning," *Remote sensing*, vol. 2, no. 6, pp. 1610–1624, 2010.

[12] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2301–2306.

[13] G. Li, Y. Liu, L. Dong, X. Cai, and D. Zhou, "An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 3854–3859.

[14] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09, 2005.

[15] G. Pandey, J. R. McBride, S. Savarese, and R. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015.

[16] Z. Taylor and J. Nieto, "A mutual information approach to automatic calibration of camera and lidar in natural environments," in *Australian Conference on Robotics and Automation*, 2012, pp. 3–5.

[17] Z. Taylor, J. Nieto, and D. Johnson, "Automatic calibration of multi-modal sensor systems using a gradient orientation measure," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1293–1300.

[18] K. K. Pingle, "Visual perception by a computer," in *Automatic interpretation and classification of images*, 1969, pp. 277–284.

[19] P. Jiang, P. Osteen, and S. Saripalli, "SemCal: Semantic lidar-camera calibration using neural mutual information estimator," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2021, pp. 1–7.

[20] B.-H. Yoon, H.-W. Jeong, and K.-S. Choi, "Targetless multiple camera-lidar extrinsic calibration using object pose estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 377–13 383.

[21] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor arrays," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4843–4850.

[22] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "RegNet: Multimodal sensor registration using deep neural networks," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 1803–1810.

[23] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "CalibNet: Geometrically supervised extrinsic calibration using 3D spatial transformer networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1110–1117.

[24] J. Shi, Z. Zhu, J. Zhang, R. Liu, Z. Wang, S. Chen, and H. Liu, "CalibRCNN: Calibrating camera and lidar by recurrent convolutional neural network and geometric constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 197–10 202.

[25] D. González-Aguilera, P. Rodríguez-Gonzálvez, and J. Gómez-Lahoz, "An automatic procedure for co-registration of terrestrial laser scanners and digital cameras," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 3, pp. 308–316, 2009.

[26] J. Li-Chee-Ming and C. Armenakis, "Fusion of optical and terrestrial laser scanner data," in *Canadian Geomatics Conference and Symposium of Commission I, ISPRS Convergence in Geomatics-Shaping Canada's Competitive Landscape.*, 2010, pp. 15–18.

[27] J. Böhm and S. Becker, "Automatic marker-free registration of terrestrial laser scans using reflectance," in *Proceedings of the 8th conference on optical 3D measurement techniques*, 2007, pp. 9–12.

[28] P. Moghadam, M. Bosse, and R. Zlot, "Line-based extrinsic calibration of range and image sensors," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2013, pp. 3685–3691.

[29] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[30] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[31] J. Kang and N. L. Doh, "Automatic targetless camera–LIDAR calibration by aligning edge with Gaussian mixture model," *Journal of Field Robotics*, vol. 37, no. 1, pp. 158–179, 2020.

[32] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7517–7524, 2021.

[33] D. C. Brown, "Close-range camera calibration," *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, 1971.

[34] J. Heikkila and O. Silvén, "A four-step camera calibration procedure with implicit image correction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1997, pp. 1106–1112.

[35] T. Miyasaka, Y. Ohama, and Y. Ninomiya, "Ego-motion estimation and moving object tracking using multi-layer LIDAR," in *IEEE Intelligent Vehicles Symposium*, 2009, pp. 151–156.

[36] C.-K. Liang, Y.-C. Peng, and H. Chen, "Rolling shutter distortion correction," in *Visual Communications and Image Processing*, ser. Proceedings of the SPIE, vol. 5960, 2005, pp. 1315–1322.

[37] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," 2020, arXiv:1812.01537v9.

[38] D. Marr, "Analysis of occluding contour," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 197, no. 1129, pp. 441–475, 1977.

[39] Y. Tsin and T. Kanade, "A correlation-based approach to robust point set registration," in *European Conference on Computer Vision (ECCV)*, 2004, pp. 558–569.

[40] T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," in *International Conference on Machine Learning (ICML)*, vol. 28, no. 3, 2013, pp. 343–351.

[41] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[42] R. D. Martin and C. J. Masreliez, "Robust estimation via stochastic approximation," *IEEE Transactions on Information Theory*, vol. 21, no. 3, pp. 263–271, 1975.

[43] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning (ICML)*, vol. 28, 2013, pp. 1310–1318.

[44] J. Moravec, "Automatic on-line calibration and calibration monitoring of cameras and lidars," MSc Thesis, Charles University, Faculty of Mathematics and Physics, Prague, 2020.

[45] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research*, vol. 32, no. 11, p. 1231–1237, 2013.

[46] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[47] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[48] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

Appendix A
LiDAR Corners Detection

LiDAR corners C_j are extracted from the point cloud P_j as follows:

a) For each azimuth φ_i in each LiDAR scanline, the radial distance data $d(\varphi_i)$ is whitened per window $\mathbf{d}(\varphi_i)$ of size $n_m = 11\,\text{lpx}$[9] centered on i as $d(\varphi_i)/\|\mathbf{d}(\varphi_i)\|$.

b) The whitened data is convolved with a discrete Gaussian-derivative filter m with variance $\sigma_m^2 = 1$ using a filter of n_m elements. The absolute value is taken.

c) Non-maximum suppression with window size $w^{\text{cd}} = 4\,\text{lpx}$ is performed and weak local maxima below $T^{\text{cd}} = 0.01$ are removed. Of the two neighbors forming the local maximum, the one closer to the LiDAR is taken.

d) With the exception of CARLA simulation, Steps a–c are repeated for the LiDAR intensity data, with window size $w^{\text{cr}} = 6\,\text{lpx}$ and a weak-response threshold of $T^{\text{cr}} = 0.05$.

e) Wide gaps in azimuth data are detected when the azimuth difference between consecutive, valid 3D points exceeds a gap threshold of $T^{\text{ca}} = 0.1\,\text{rad}$. These gaps typically happen on the skyline. This procedure will always choose two corners per one azimuth gap.

Appendix B
Methods Parameters

Parameters used by the studied methods are in Table VII.

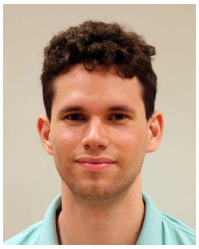
[9]We denote LiDAR pixels as lpx.

TABLE VII: Parameters of all the studied methods.

method	parameter	value	reference
OCAMO/LTO	n_m	11 lpx	Sec. A.a
	σ_m	1	Sec. A.b
	T^{cd}	0.01	Sec. A.c
	w^{cd}	4 lpx	Sec. A.c
	T^{cr}	0.05	Sec. A.d
	w^{cr}	6 lpx	Sec. A.d
	T^{ca}	0.1 rad	Sec. A.e
	h_θ	0.001 rad,m	(6)
	k	10	(3)
	σ	9 px	(3)
OCAMO	$\boldsymbol{\sigma}_d$	(0.0033, 0.0017, 0.0005) rad	(12)
	$\mathbf{T}_\theta$	$3\,\boldsymbol{\sigma}_d$	(12)
	$\boldsymbol{\theta}^{\text{bnd}}$	$5\,\boldsymbol{\sigma}_d$	(11b)
	$\boldsymbol{\theta}^{\text{upd}}$	0.0024 rad	(10), (16)
	b	10 frames	(11a)
	m^{bnd}	5 frames	Sec. III-C2
LT/LTO	grid size mon.	0.01 rad and 0.1 m	Sec. III-D
	grid size trk.	0.0005 rad	Sec. III-D
	w	9 frames	(14)
LTO	(α_c, β_c)	$(40.6, 0.203)$	(15)
	(α_d, β_d)	$(4.08, 3.70)$	(15)
LT	(α_c, β_c)	$(8.69, 0.367)$	(15)
	(α_d, β_d)	$(4.12, 4.40)$	(15)
	α	$\frac{1}{3}$	Sec. III-D
	Image γ	0.86	Sec. III-D
	LiDAR γ	0.5	Sec. III-D
	LiDAR filter	1 m	Sec. III-D

Appendix C
GitHub Repository

This paper has a GitHub repository at: https://github.com/moravecj/OCaMo.



Jaroslav Moravec received his MSc in Artificial Intelligence from Charles University, Prague, in 2020. He is pursuing a PhD at the Department of Cybernetics, Czech Technical University in Prague. His research interests include autonomous driving, data fusion and sensor calibration.

Radim Šára has been an associate professor at the Czech Technical University in Prague (CTU) since 2008. He received his PhD in 1994 from Johannes Kepler University in Linz, Austria. From 1995 to 1997, he worked at the GRASP Laboratory at The University of Pennsylvania. He has been a member of the Department of Cybernetics and the Center for Machine Perception at CTU since 1998. He heads the PhD School of Artificial Intelligence and Biocybernetics at CTU. His main research interest is 3D computer vision.