Extrinsic Camera Calibration with Semantic Segmentation

Alexander Tsaregorodtsev[1], Johannes Müller[1], Jan Strohbeck[1], Martin Herrmann[1],
Michael Buchholz[1] and Vasileios Belagiannis[2]

Abstract— **Monocular camera sensors are vital to intelligent vehicle operation and automated driving assistance and are also heavily employed in traffic control infrastructure. Calibrating the monocular camera, though, is time-consuming and often requires significant manual intervention. In this work, we present an extrinsic camera calibration approach that automatizes the parameter estimation by utilizing semantic segmentation information from images and point clouds. Our approach relies on a coarse initial measurement of the camera pose and builds on lidar sensors mounted on a vehicle with high-precision localization to capture a point cloud of the camera environment. Afterward, a mapping between the camera and world coordinate spaces is obtained by performing a lidar-to-camera registration of the semantically segmented sensor data. We evaluate our method on simulated and real-world data to demonstrate low error measurements in the calibration results. Our approach is suitable for infrastructure sensors as well as vehicle sensors, while it does not require motion of the camera platform.**

I. INTRODUCTION

Monocular cameras are a keystone to cost-efficient sensor setups [1] for both intelligent vehicles (IVs) and smart infrastructure, supporting automated driving. The calibration of such cameras is vital to their operation out in the field and, therefore, of utmost importance for applications such as self-localization [2] or recognition [3]. However, while the intrinsic camera calibration can easily be accomplished once in a constrained environment, the recovery of the extrinsic calibration in the field or between vehicle sensors is still a challenging process. In particular, classical calibration methods [4] often require manually placed calibration targets in the scene. These approaches, therefore, cannot easily be applied to traffic cameras, observing densely populated junctions or to IVs out in the field. Thus, an automatic calibration solution, which works for single infrastructure cameras, is needed to reduce calibration effort. Such a solution can be used to calibrate cameras mounted on IVs as well.

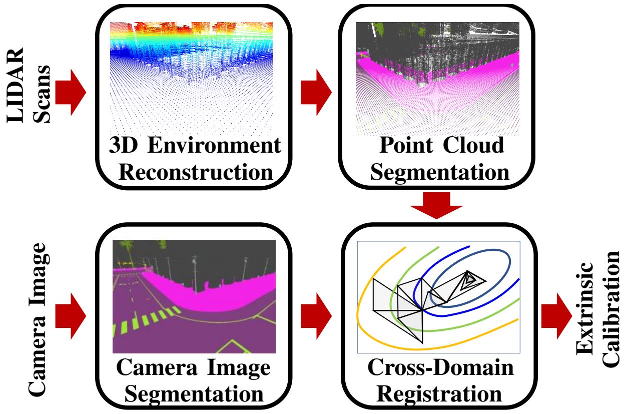

LIDAR Scans
3D Environment Reconstruction
Point Cloud Segmentation
Camera Image
Camera Image Segmentation
Cross-Domain Registration
Extrinsic Calibration

Fig. 1. Approach Overview. On the left side, data from both domains are fed into the pipeline. For lidar data, the 3D environment is reconstructed using multiple scans. Afterward, both domains are labeled semantically. In the final step, our proposed optimization is performed to register both domains to retrieve the extrinsic calibration parameters.

Recently, sensor-to-sensor registration methods, e.g. [5], [6], have been proposed to automate the calibration process. However, these methods often require broadly overlapping fields of view (FOVs), which implies redundant sensors and hence negates the cost savings achieved by a single sensor. For IVs, specific fully automated calibration schemes, such as [7], have been developed, which rely on the motion of the sensor platform, i.e., the IV, to optimize the calibration during each time step. A caveat of these specific calibration schemes is their dependency on sensor motion, which excludes static infrastructure from the range of possible applications. Deep learning approaches like [8] have recently been developed to estimate extrinsic camera parameters from camera images and lidar data. While these approaches have shown promising results, they still tend to have higher estimation errors compared to previous methods like [7], making them unsuitable for IV and infrastructure calibration. Finally, another classical method is described in [9]. It localizes a camera sensor inside of a 3D model of the environment taken from a SLAM with stereo cameras. However, this method obtains its 3D environments from a mobile stereo camera setup and thus contains texture information not present in lidar data. Furthermore, this approach is not transferable to stereo camera setups permanently mounted on IVs due to their limited field of view.

*Part of this work was financially supported by the Federal Ministry of Economic Affairs and Energy of Germany within the program ”New Vehicle and System Technologies” (project LUKAS, grant number 19A20004F). Part of this work has been conducted as part of ICT4CART project which has received funding from the European Union’s Horizon 2020 research & innovation program under grant agreement No. 768953. Content reflects only the authors’ view and European Commission is not responsible for any use that may be made of the information it contains.

[1] The authors are with the Institute of Measurement, Control and Microtechnology, Ulm University, D-89081 Ulm, Germany
{<first name>.<family name>@uni-ulm.de}

[2] Vasileios Belagiannis is with the Department of Simulation and Graphics, University Magdeburg, D-39106 Magdeburg, Germany {vasileios.belagiannis@ovgu.de}. Most of this work was done during his time at Ulm University.

DOI: 10.1109/ITSC55140.2022.9922338.

We propose a monocular camera calibration approach to perform cross-domain registration of a semantically segmented mono camera image and a semantically labeled 3D model of the environment the camera is located in. An overview of the approach is given in Fig. 1. By using a semantic representation of both the camera and the point cloud domain, we introduce semantic labels, which can be matched between the domains. Furthermore, the use of semantic labels instead of raw RGB and RGBD camera data makes the registration more robust and reduces the possibility of registration algorithm errors. These can occur with raw camera data due to color space and exposure differences between different camera types.

Our approach consists of data pre-processing steps as well as an optimization step performing cross-domain registration. In the pre-processing steps, the 3D environment model is reconstructed and then labeled semantically using neural networks. The segmentation is also performed on images of the target camera that we aim to calibrate. Then, an optimization pass is invoked to match the visual appearance and projection of a rendered view of the 3D model with the segmented camera view. This optimization then yields the extrinsic camera calibration. Finally, experiments with the CARLA simulator [10] and the KITTI dataset [11] show that our method leads to the accuracy level that is suitable for automated driving applications, both for static and dynamic camera platforms. The contribution of this paper is twofold. From a practical perspective, our approach allows for low-effort, highly automated calibration of monocular cameras in (geo-referenced) world coordinates without requiring motion of the target sensor platform. From a methodological perspective to the best of our knowledge, we are the first to use semantic segmentation as a visual feature set to guide the required parameter optimization. The source code of our approach is publicly available[1].

The related work to extrinsic camera calibration is outlined in Section II. The complete algorithm is described in Section III, while in Section IV, a theoretical and practical evaluation is provided. At last, the conclusion of this work is described in Section V.

II. RELATED WORK

Calibration of cameras has been extensively investigated in the past. A classical approach for intrinsic and extrinsic camera calibration is presented in [4]. One of the first approaches specifically for infrastructure sensors was described in [5], where a vehicle target was used to calibrate a pair of fisheye cameras with a large overlapping field of view. Another approach using planar structure from motion and textured environments was presented in [6], which is able to calibrate multi-sensor configurations. The approach of [9] circumvents the field of view (FoV) overlap requirement of the other two approaches by reconstructing a 3D environment using a mobile RGB-D camera setup. Then, a matching of feature descriptors between the target camera image and the model is performed and the closest matching keyframes are found to determine the camera pose. The automatic calibration method of [12] estimates the camera vanishing points by tracking image features of moving vehicles, while a newer method by [13] uses a neural network to find landmark positions on vehicles. By using the positional relations between landmarks obtained from real 3D models of the vehicles, a calibration is estimated from the landmarks.

Another class of algorithms performs direct 2D to 3D matching and registration. Feng et al. [14] use a neural network to directly match point cloud patches to image patches, while the algorithm in [15] performs camera localization by finding correspondences between 2D and 3D line features. Li et al. [8] train a classifier to detect if a point in the point cloud is in the camera frustum and then solve the inverse camera projection problem using the frustum labels.

Methods performing calibration through sensor motion can also be found in the literature ([7], [16], [17]). In particular, Horn et al. [7] use a visual SLAM approach to estimate the camera displacement and then represent the translation and rotation using dual quaternions instead of homogeneous transformation matrices, which allows for a more efficient implementation of the optimization. The optimization consists of finding a calibration matrix that is able to close the loop between the current and previous time step, where the calibration matrices of the previous time step are known.
A recent method using automated vehicles as calibration targets is shown in [18]. It uses lidar measurements in an empty scene to determine the ground plane and background and then regresses the position of the automated vehicle as it passes through the sensors FoV with localization data of the vehicle being available.

We improve on previous methods by exploring a new concept for matching cross-domain features by visual alignment, which does not require specific calibration targets or motion of the sensor itself. It, therefore, allows us to perform a single-shot calibration in diverse scenarios without requiring a lot of manual intervention.

III. CALIBRATION METHOD

In this section, we formulate our problem and present each step of our approach required to estimate the extrinsic camera calibration.

A. Problem Formulation and Constraints

Consider the perspective camera model [4]

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\boldsymbol{K}_{\text{int}} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}}_{:=\boldsymbol{P}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \qquad (1)$$

where $[u, v, 1]^T$ denotes a homogeneous camera image coordinate of an image $\mathcal{I}$ with pixel color value $\mathcal{I}_{u,v} \in [0; 1]^3$, and $[x, v, z, 1]^T \in \mathcal{M}^{\text{3D}}$ is a homogeneous point coordinate of a 3D model $\mathcal{M}^{\text{3D}}$ of the observed scene. $\mathcal{M}^{\text{3D}}$ is represented as a point cloud. Furthermore, it is assumed

[1] https://github.com/Tuxacker/semantic_calibration

that the scene visible in the camera image is overlapping with the scene represented in the point cloud.

Our goal is to recover the parameters of the extrinsic camera matrix $\boldsymbol{P}$, i.e., the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and the translation vector $\boldsymbol{t} \in \mathbb{R}^3$. At the same time we assume that the intrinsic parameters $\boldsymbol{K}_{\text{int}} \in \mathbb{R}^{3\times4}$, which represent the mapping from 3D coordinates in the camera coordinate frame to 2D pixel coordinates, are already known, e.g. as described in [4]. Below, we propose a new calibration approach to estimate these extrinsic camera parameters.

B. Approach Overview

We use visual matching of semantic information for the parameter estimation of the camera rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$. Our method consists of four steps, starting with the 3D environment reconstruction $\mathcal{M}^{\text{3D}}$, represented as a point cloud (see Sec. III-C). It is recorded with the lidar sensor mounted on a vehicle. Next, the point cloud segmentation step (presented in Sec. III-D) follows to extract semantic information for each 3D point. Then, the camera image segmentation (see Sec. III-E) assigns a class category to each pixel. Given the segmented 3D point cloud and image, we propose a cross-domain registration approach (explained in Sec. III-F) to estimate the extrinsic camera parameters by visually matching the camera segmentation with a rendered point cloud view using linear programming [19].

C. 3D Environment Reconstruction

We assume access to a vehicle equipped with a lidar sensor and high-precision localization to construct the 3D environment model $\mathcal{M}^{\text{3D}}$. Several lidar scans are recorded as point clouds and expressed in the world coordinate frame. The transformation from lidar sensor coordinates to world coordinates requires known lidar extrinsics. Due to road bumps and inaccurate orientation measurements, points in the sensor's far field come with increased positional errors, resulting in noisy 3D point clouds with many outliers. Thus, we filter out points that exceed the maximum distance d_{max} from the vehicle's lidar sensor. In practice, d_{max} is chosen to be between 50 m and 75 m, which are standard cutoff values for Velodyne sensors like the VLP-32C or the HDL-64E used in the KITTI [11] dataset. Next, we rely on the multiway Iterative Closest Point (ICP) registration algorithm [20], [21] between all scans, which partially compensates for localization errors that accumulate over time. To accelerate the registration, instead of registering every scan against every other scan, all scans are sorted by their capture time, and groups of three neighboring scans are recursively registered and merged. The recursion ends when all groups are combined into the single point cloud $\mathcal{M}^{\text{3D}}$. At last, the point cloud is down-sampled during each recursion step in order to relax the ICP computation requirements. The reconstruction step can be skipped if the 3D environment is obtained by another mapping algorithm or from a different source.

D. Point Cloud Segmentation

In the next step, semantic labels are assigned to the points of $\mathcal{M}^{\text{3D}}$ to obtain the semantic segmentation map $\mathcal{M}^{\text{3D}}_{\boldsymbol{c}}$, which contains a color label for each point in $\mathcal{M}^{\text{3D}}$. For this task, we rely on a pre-trained deep neural network to perform semantic segmentation, namely Cylinder3D [22], which supports segmentation of classes defined in SemanticKITTI [23]. Furthermore, the segmentation is also used to filter out dynamic object classes like cars and pedestrians. This leaves the buildings, ground points, vegetation, fences, poles, and traffic signs classes for registration. Therefore, the calibration method does not require exact time synchronicity between the lidar scans and the camera image, as points belonging to dynamic objects can easily be filtered out by their label.

E. Camera Image Segmentation

After labeling the point cloud $\mathcal{M}^{\text{3D}}$, the semantic segmentation map $\mathcal{I}^{\text{sem}}$ is extracted from the camera image $\mathcal{I}$. Similar to the point cloud segmentation, we rely on a pre-trained deep neural network to extract semantic labels available in the Cityscapes dataset [24], e.g., OCRNet [25]. We chose to use Cityscapes labels, as they share most classes available in SemanticKITTI and thus enable direct matching of class labels between domains. Again, dynamic object class categories, e.g., cars or pedestrians, are removed from the segmentation map. For the remaining object categories, we perform a category alignment between the point cloud [22] and the image [25] segmentation models by neglecting points and pixels with labels that are unique to this domain. It should be noted that the removal of dynamic objects results in holes in the resulting map, especially in crowded scenes. To minimize their impact, we introduce the normalization factor β during the registration step in Section III-F.

The image segmentation can also be done semi-automatically to improve calibration quality. In this case, the resulting segmentation is manually edited to fix incorrectly labeled regions in order to improve data quality and hence reduce the overall calibration error.

F. Cross-domain registration

Next, we aim to estimate the rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$. We formulate this estimation problem as an optimization, where a loss correlated to the error $\|\hat{\boldsymbol{R}}-\boldsymbol{R}\|$ and $\|\hat{\boldsymbol{t}}-\boldsymbol{t}\|$ is minimized using the simplex algorithm by Nelder and Mead [19]. As the Nelder-Mead method does not require gradient information, it allows a broad range of data transforms to be applied to $\mathcal{M}^{\text{3D}}$. This includes rendering transforms, which can be used to obtain a rasterized image $\hat{\mathcal{I}}$ of a specific perspective view of $\mathcal{M}^{\text{3D}}$. We, therefore, define the rendering function $\boldsymbol{f}$ as

$$\hat{\mathcal{I}} = \boldsymbol{f}\left(\mathcal{M}^{\text{3D}}, \boldsymbol{P}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}), \mathcal{M}^{\text{3D}}_{\boldsymbol{c}}\right) , \tag{2}$$

where $\mathcal{M}^{\text{3D}}_{\boldsymbol{c}}$ is the color encoding of the segmentation and $\hat{\boldsymbol{R}}$, $\hat{\boldsymbol{t}}$ are estimates of $\boldsymbol{R}$ and $\boldsymbol{t}$. This rendering function is used to transform the point cloud $\mathcal{M}^{\text{3D}}$ with color labels $\mathcal{M}^{\text{3D}}_{\boldsymbol{c}}$ into an image $\hat{\mathcal{I}}$ with the viewpoint defined by the perspective camera matrix $\boldsymbol{P}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}})$. $\hat{\mathcal{I}}$ can then be visually matched with the previously obtained image segmentation $\mathcal{I}^{\text{sem}}$ by calculating a distance metric between both images and interpreting the result as a loss value. By adjusting

$\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{t}}$ to minimize the visual difference between these images, and therefore minimizing the loss value, we actively determine the optimal camera parameters. An advantage of the Nelder-Mead method lies in its use of a simplex to define the initial search space, which then moves and shrinks towards the function minimum. By scaling the initial simplex bounds according to the expected measurement variance of the initial guess, we can easily define a sensible search space covered by the starting simplex. Note that other gradient-free approaches may be used as well.

a) Initialization: In order to start the optimization, the Nelder-Mead method requires an initial parameter set. The convergence speed of the algorithm and its ability to find an optimal solution depend on the initial parameter set and the search space. Therefore, an initial guess for the camera position and orientation is required. We furthermore constrain the search space of the pitch and yaw angle to $\pm 5°$, while we set the roll angle to zero. We expect our method to also work for non-vanishing roll angles, but in our use cases, no camera roll is desired and can be achieved by proper mounting. This is a reasonable simplification since potential camera roll is often not desired for automotive applications. The yaw angle can be measured with a compass, while the pitch can be determined with a spirit level. The maximal position offsets are constrained to $\pm 2.5\,\mathrm{m}$, i.e., the GPS accuracy of a typical smartphone GPS receiver. We argue that an initial guess with such accuracy can be easily obtained for most practical situations and is sufficient to obtain a good solution for the extrinsic calibration problem, as shown in Sections IV-B and IV-C. However, it is possible to extend the search space at a tolerable increase in calculation effort by combining the Nelder-Mead method with a coarse grid search to estimate parameters that were not measured and are therefore not known. If an IV sensor-to-sensor calibration is desired, no initial measurements are required in most cases, as the convergence range of our approach is sufficient to bridge the translation and rotation between both sensors. Therefore, they can be set to be zero vectors in such cases. After setting the initial guess, we also have to calculate the initial simplex, which will converge to the optimal solution. To this end, we generate 6 values (as a simplex of dimension n is described by $n+1$ points) by adding and subtracting the maximal constrained parameter range for each pair of simplex vertices and for each parameter.

b) Optimization Loop: The Nelder-Mead method is invoked until the loss distance between two steps falls below 10^{-4} in order to estimate $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{t}}$. The loss target we optimize is defined as

$$\mathcal{L}(\hat{\mathcal{I}}, \mathcal{I}^{\text{sem}}) = \beta \sum_{u=0}^{h-1} \sum_{v=0}^{w-1} \mathcal{V}_{u,v}\, \ell_{\text{L2}}(\hat{\mathcal{I}}_{u,v}, \mathcal{I}^{\text{sem}}_{u,v}) \tag{3}$$

with the L2 loss term ℓ

$$\ell_{\text{L2}}(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{a}-\boldsymbol{b}). \tag{4}$$

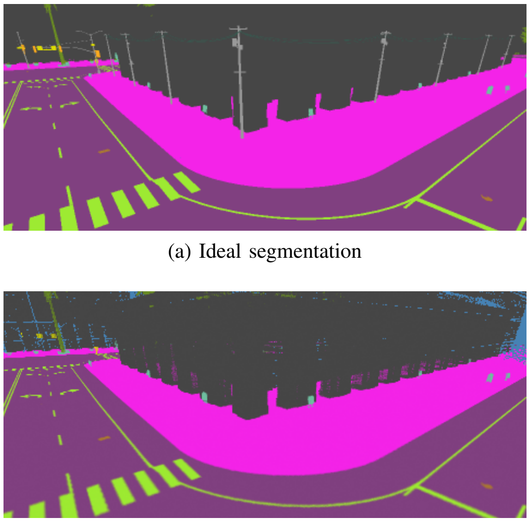
Here, β is a normalization factor, w and h denote the width and height of the images, and $\mathcal{V}_{u,v}$ is a function checking the validity of the classes. If one of the images contains an invalid class for a pixel, $\mathcal{V}_{u,v}=0$, i.e., the pixel is ignored for loss calculations. Otherwise, $\mathcal{V}_{u,v}=1$ implies a contribution to the final loss. It should be noted that, while in theory, any distance metric may be used for ℓ, we decided to use an L2 loss, as this loss is commonly used as an image similarity measure. Finally, the optimization performed by the Nelder-Mead method can be represented as

$$\hat{\boldsymbol{R}}_{\text{opt}}, \hat{\boldsymbol{t}}_{\text{opt}} = \arg\min_{\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}} \left\{ \mathcal{L}(\hat{\mathcal{I}}, \mathcal{I}^{\text{sem}}) \right\}. \tag{5}$$

c) Appearance Matching and Masking: Before optimizing the camera pose, we want to closely match the general appearance of the rendered image $\hat{\mathcal{I}}$ to the target $\mathcal{I}^{\text{sem}}$ as well as reduce rendering artifacts. For that reason, we first aim to minimize the appearance of background pixels due to the sparsity of points in the 3D model. This is achieved by calculating the distance d_i of every point in $\mathcal{M}^{\text{3D}}$ to the camera position. Then, each point in the 3D model is rendered as a circle of radius $r_i = \lambda / d_i$, where λ is a scaling factor that depends on the point cloud density and can be determined empirically by rendering a test view and increasing λ until the appearance of the rendered view roughly matches the target segmentation image. This adaptive point size approach also adds the benefit of limiting the impact of varying line counts between different lidar models, as lidars with fewer lines can still be densely rendered by increasing λ. Secondly, pixels, which still cannot be assigned with a semantic label corresponding to a static object, e.g., due to point cloud sparsity, finally are classified as invalid so that these pixels do not contribute to the loss when estimating $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{t}}$. This is achieved by storing this information in the mask elements $\mathcal{V}_{u,v}$ and masking the loss. In order to compensate for the potentially different amount of pixels contributing to the loss between two optimization steps, a normalization factor $\beta = \mathrm{numel}(\mathcal{V}) / \sum_{u,v} \mathcal{V}_{u,v}$ is introduced, where numel denotes the element count of an array. Without β, a change of the camera pose could potentially reveal more surface area of a masked dynamic object. This may result in a lower loss due to fewer non-masked pixels contributing to it. At the same time, the pose may have changed away from the true pose, which should imply a higher loss, thus creating a need for loss normalization.

d) Result Verification: After performing an optimization step, we obtain the initial calibration result. However, as the loss function is highly non-convex with respect to the calibration parameters, this result may represent a local minimum. To avoid local minima, the optimization is restarted twice using the previous result as the new start value while also lowering the convergence threshold from 10^{-4} to 10^{-6} during the last optimization step. The parameter set resulting in the lowest loss is chosen.

To check that the resulting parameters are indeed optimal, a small additive noise is added to the initial guess in order to exit a potentially found local minimum of the loss. The scale of the additive noise can be based on the measurement



(a) Ideal segmentation

(b) Rendered point cloud view

Fig. 2. CARLA simulator [10] sample result (Town10HD). (a) depicts an ideal semantic image segmentation of a camera sensor. (b) shows a rendered view of a semantically segmented point cloud.

accuracy of the initially measured camera pose. Repeating the optimization with noisy initial values can then be used to discard sub-optimal parameter sets with high final loss values, which are caused by the non-convexity of the loss function.

IV. EVALUATION

We evaluate our approach on infrastructure scenarios by observing a virtual intersection environment using the CARLA simulator [10], as well as on a real-world environment based on the KITTI Odometry benchmark [11]. Next, we describe the evaluation protocol including the scene processing used for benchmarking and finally present our results.

A. Evaluation Scenes

Similar to related work like [5] and [9], we use predefined scenes for evaluation. For both CARLA and KITTI, two distinct scenes were defined. For CARLA, we configured urban intersection views from an elevated static perspective to mimic real use cases for roadside infrastructure. Meanwhile, for KITTI, one time frame of two different urban driving sequences was chosen, as we only need one time frame at which the camera will be calibrated. In this case, however, we merged 250 lidar scans around the timestamp of the chosen frame to reconstruct the 3D environment. In the case of CARLA, the segmented point clouds, as well as the segmented camera images of the defined scenes, were generated using the built-in maps. Given completely correct segmentation maps, it is easier to assess the performance of our approach. The virtual lidar was configured to have 128 lines and a vertical field of view of 90° with the center of the FoV aligned to the ground plane. The virtual segmentation camera used a resolution of 960x600 with a horizontal FoV of 90°. For the KITTI data, the Cylinder3D [22] model, trained with Semantic-KITTI labels [23], was used to generate the point cloud segmentation. Therefore, no ground truth data available in Semantic-KITTI was used. Moreover, the image segmentation was obtained with OCRNet [25], trained with Cityscapes labels [24]. The sensor setup used for recording the raw data passed to the network is described in the KITTI paper [11]. As the image segmentation network provided sub-optimal results in some cases, we manually corrected the segmentation to additionally evaluate for a *semi-automatic* setting. In our *automatic* setting, the original image segmentation was utilized without any modification. In all KITTI scenes, road and sidewalk labels were merged in both domains to circumvent noisy borders between ground points which would hinder the optimization. At the same time, the dynamic object removal mentioned in Sec.III-D was limited to the driveable area covered by the measurement vehicle, which helped retain parked vehicles in the point cloud. This helped to create more visual comparison areas not masked out by $\mathcal{V}_{u,v}$.

Furthermore, during CARLA evaluation, the point cloud view was rendered with a sky background, as visible in Fig. 2b. In order to only match the sky region common between both images, sky regions in the rendered view not matching sky regions in the target were masked out, as these pixels are caused by point cloud sparsity. Another efficient measure consisted of only using the bottom half of the image for registration, as in the KITTI lidar data, the point cloud was cropped in the height dimension, resulting in a trimmed view of the scene which introduced high discrepancies in the upper half of the image. This can be clearly seen in Fig. 3c. Finally, the point cloud was cropped to a 75m radius around the initial location to reduce the memory footprint, in the case of KITTI evaluation. The lidar segmentation view was rendered with Pytorch3D [26].

B. Evaluation Protocol

As the ground truth calibration for both evaluations is known, a random initialization procedure was performed to assess the convergence behavior of the optimization approach. This was implemented by initializing the optimization algorithm with noisy ground truth, where the additive noise was uniformly sampled from the interval $[-2.5\,\mathrm{m}; 2.5\,\mathrm{m}]$ for positional offsets and $[-5°; 5°]$ for rotational offsets, as mentioned in Sec. III-F. By using this approach, we can also verify the usability of our approach, as such offsets could appear in field measurements. After the completion of the optimization, the remaining offsets were interpreted as the calibration quality. As all coordinate systems used are scaled to meters, the positional offsets are measured in centimeters. The algorithm is invoked 30 times on each scene and the 10 parameter sets resulting in the lowest loss $\mathcal{L}(\hat{\mathcal{I}}, \mathcal{I}^{\text{sem}})$ are considered as acceptable results to filter out results that would be considered outliers, as described in the Result Verification of Sec. III-F.
We then report the mean vector norm of the displacement error vector $\overline{\|\Delta_{\boldsymbol{t}}\|} = \overline{\|\widehat{\boldsymbol{t}_{\text{ext}}} - \boldsymbol{t}_{\text{ext}}\|}$ as well as the mean absolute angular error $\overline{|\Delta_\alpha|}$. Furthermore, to evaluate our choice of the L2 loss as a distance metric, we also repeated

the previously described evaluation protocol with the Huber loss [27]

$$\ell_{\text{Huber}}(\boldsymbol{a}, \boldsymbol{b}) = \begin{cases} \frac{1}{2}(\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{a}-\boldsymbol{b}), & \|\boldsymbol{a}-\boldsymbol{b}\| < \delta \\ \delta(\|\boldsymbol{a}-\boldsymbol{b}\| - \frac{\delta}{2}), & \text{otherwise} \end{cases} \tag{6}$$

For results using the Huber loss, we set the loss threshold δ to 0.3.

TABLE I

EVALUATION RESULTS.

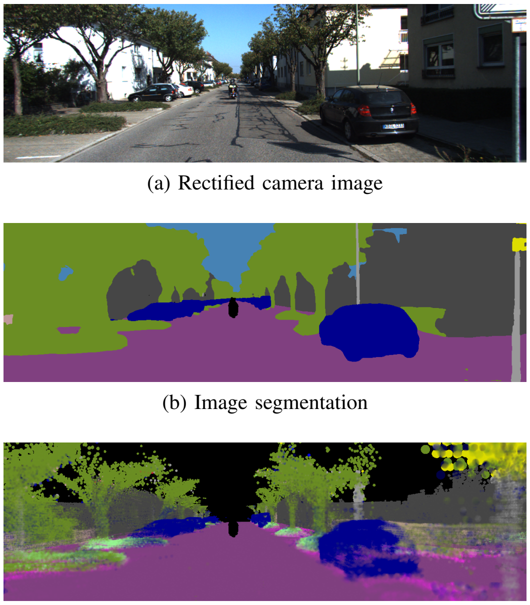
Loss	L2		Huber	
Metric	$\overline{\|\Delta_{\boldsymbol{t}}\|}$	$\overline{\|\Delta_\alpha\|}$	$\overline{\|\Delta_{\boldsymbol{t}}\|}$	$\overline{\|\Delta_\alpha\|}$
Fully automated CARLA-simulated infrastructure				
Town10HD	1.5 cm	0.03°	1.5 cm	0.02°
Town01	6.0 cm	0.11°	6.2 cm	0.10°
Semi-automated KITTI				
Sequence 00	12.7 cm	0.28°	15.3 cm	0.26°
Sequence 09	20.6 cm	0.40°	19.3 cm	0.42°
Fully automated KITTI				
Sequence 00	20.2 cm	0.34°	20.3 cm	0.33°
Sequence 09	18.6 cm	0.40°	19.6 cm	0.43°

C. Results Discussion

Table I summarizes our results for both evaluations and examined loss functions. For the CARLA evaluation, our translation error is below $7\,\mathrm{cm}$ for both scenes, while the rotation error is at most $0.11°$. We observed that the errors mostly stem from point cloud sparsity, as can be seen in Fig. 2. It should also be noted that, due to the rasterized representation of an image, we cannot distinguish image translations below a single pixel, therefore very small changes of under $1\mathrm{cm}$ in camera translation and $0.01°$ in camera rotation may not be visible in the rendered view. The real-world KITTI scenes also perform very well. While both the point cloud labels and image segmentation contain incorrect labels and, in the case of point clouds, incorrectly measured data points, the final calibration quality is still very high, especially if used for infrastructure sensor calibration. Furthermore, we tested fully automatically labeled KITTI scenes by not manually correcting the image segmentation. The performance in these scenes dropped only slightly or even stayed in the same range and is therefore comparable with the semi-automatic use case. However, it should be noted that this performance highly depends on the quality of the sensor data and the dataset used for training the segmentation network. Comparing both loss types, we can see only minor differences, which can be explained by the random initialization and testing approach. Thus, both losses are equally suitable for the task.

D. Comparison with Related Approaches

As our approach is quite different from previous works in this area and no common evaluation protocols are defined, it is difficult to directly compare performance metrics, so we only look at qualitative results. Considering Horn et al. [7], who achieve translation errors of $20.8\,\mathrm{cm}$ and rotation errors of $0.26°$ on the KITTI dataset, but require sensor movement, we get equally good results (see Table I) without sensor movement. The authors of the mobile stereo camera SLAM approach in [9] report translation errors in the range of $6\mathrm{cm} - 67\mathrm{cm}$ and rotation errors between $1.5°$ and $5.4°$ on their own data. Comparably, our positional errors are on the lower end of the spectrum, while our rotation errors are smaller by a factor of 10. We, therefore, conclude that our approach is competitive with other methods and can be used for infrastructure and IV sensor calibration.



(a) Rectified camera image

(b) Image segmentation

(c) Rendered point cloud view

Fig. 3. KITTI [11] dataset result (Sequence 00 fully automated). We can see a noticeable visual difference between the segmentation image and the rendered point cloud, which is caused by point cloud sparsity as well as incorrect labels. However, they still share enough common labelfeatures to be reliably matched, as shown in Table I.

V. CONCLUSION

We presented an extrinsic camera calibration approach for both infrastructure and intelligent vehicle cameras. In our approach, we obtained a semantic segmentation of both IV-collected lidar data as well as camera data. This segmented data was then used to find an optimal calibration by using an optimization loop to match the appearance of the image segmentation with a rendered view of the segmented lidar data, effectively performing cross-domain registration. The algorithm has been evaluated on simulated CARLA [10] scenes and real sensor data from the KITTI dataset [11] to show its feasibility in a practical application, showing equally good or better results than existing approaches, which, however, are more constrained in their application.

REFERENCES

[1] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, 2008.

[2] N. Engel, S. Hoermann, M. Horn, V. Belagiannis, and K. Dietmayer, "Deeplocalization: Landmark-based self-localization with deep neural networks," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 926–933.

[3] J. Wiederer, A. Bouazizi, U. Kressel, and V. Belagiannis, “Traffic control gesture recognition for autonomous vehicles,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 676–10 683.

[4] R. Hartley and A. Zisserman, “Computation of the camera matrix P,” in *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004, p. 178–236.

[5] S. R. E. Datondji, Y. Dupuis, P. Subirats, and P. Vasseur, “Rotation and translation estimation for a wide baseline fisheye-stereo at crossroads based on traffic flow analysis,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1534–1539.

[6] C. Zhu, Z. Zhou, Z. Xing, Y. Dong, Y. Ma, and J. Yu, “Robust plane-based calibration of multiple non-overlapping cameras,” in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 658–666.

[7] M. Horn, T. Wodtko, M. Buchholz, and K. Dietmayer, “Online extrinsic calibration based on per-sensor ego-motion using dual quaternions,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 982–989, 2021.

[8] J. Li and G. H. Lee, “DeepI2P: Image-to-point cloud registration via deep classification,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 960–15 969.

[9] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and Y. Miki, “Calibration of non-overlapping cameras using an external SLAM system,” in *2014 2nd International Conference on 3D Vision*, vol. 1, 2014, pp. 509–516.

[10] 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.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets Robotics: The KITTI Dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[12] M. Dubská, A. Herout, R. Juránek, and J. Sochor, “Fully automatic roadside camera calibration for traffic surveillance,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1162–1171, 2014.

[13] V. Bartl, J. Špaňhel, P. Dobeš, R. Juranek, and A. Herout, “Automatic camera calibration by landmarks on rigid objects,” *Machine Vision and Applications*, vol. 32, no. 1, pp. 1–13, 2021.

[14] M. Feng, S. Hu, M. H. Ang, and G. H. Lee, “2D3D-Matchnet: Learning to match keypoints across 2D image and 3D point cloud,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4790–4796.

[15] H. Yu, W. Zhen, W. Yang, J. Zhang, and S. Scherer, “Monocular camera localization in prior lidar maps with 2D-3D line correspondences,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4588–4594.

[16] L. Wei, L. Naiguang, D. Mingli, and L. Xiaoping, “Calibration-free robot-sensor calibration approach based on second-order cone programming,” in *MATEC Web of Conferences*, vol. 173. EDP Sciences, 2018, p. 02005.

[17] E. Wise, M. Giamou, S. Khoubyarian, A. Grover, and J. Kelly, “Certifiably optimal monocular hand-eye calibration,” in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2020, pp. 271–278.

[18] J. Müller, M. Herrmann, J. Strohbeck, V. Belagiannis, and M. Buchholz, “LACI: Low-effort automatic calibration of infrastructure sensors,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3928–3933.

[19] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[20] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[21] S. Choi, Q.-Y. Zhou, and V. Koltun, “Robust reconstruction of indoor scenes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5556–5565.

[22] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, “Cylindrical and asymmetrical 3D convolution networks for lidar segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9939–9948.

[23] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, “SemanticKITTI: A dataset for semantic scene understanding of lidar sequences,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[24] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes dataset for semantic urban scene understanding,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[25] Y. Yuan, X. Chen, X. Chen, and J. Wang, “Segmentation transformer: Object-contextual representations for semantic segmentation,” in *European Conference on Computer Vision (ECCV)*, vol. 1, 2021.

[26] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, “Accelerating 3D Deep Learning with PyTorch3D,” *arXiv:2007.08501*, 2020.

[27] P. J. Huber, “Robust estimation of a location parameter,” in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.