# SOAC: Spatio-Temporal Overlap-Aware Multi-Sensor Calibration using Neural Radiance Fields

Quentin Herau[1,2] Nathan Piasco[1] Moussab Bennehar[1] Luis Roldão[1] Dzmitry Tsishkou[1]
Cyrille Migniot[3] Pascal Vasseur[4] Cédric Demonceaux[2]
[1]Noah's Ark, Huawei Paris Research Center [2]ICB UMR CNRS 6303, Université de Bourgogne
[3]ImViA UR 7535, Université de Bourgogne [4]MIS UR 4290, Université de Picardie Jules Verne
{Quentin.Herau, Nathan.Piasco, Moussab.Bennehar, Luis.Roldao, Dzmitry.Tsishkou}@huawei.com
{Quentin.Herau@etu, Cyrille.Migniot@, Cedric.Demonceaux@ }@u-bourgogne.fr
Pascal.Vasseur@u-picardie.fr

## Abstract

*In rapidly-evolving domains such as autonomous driving, the use of multiple sensors with different modalities is crucial to ensure high operational precision and stability. To correctly exploit the provided information by each sensor in a single common frame, it is essential for these sensors to be accurately calibrated. In this paper, we leverage the ability of Neural Radiance Fields (NeRF) to represent different sensors modalities in a common volumetric representation to achieve robust and accurate spatio-temporal sensor calibration. By designing a partitioning approach based on the visible part of the scene for each sensor, we formulate the calibration problem using only the overlapping areas. This strategy results in a more robust and accurate calibration that is less prone to failure. We demonstrate that our approach works on outdoor urban scenes by validating it on multiple established driving datasets. Results show that our method is able to get better accuracy and robustness compared to existing methods.*

## 1. Introduction

Multi-sensor calibration plays a key role in autonomous systems as it ensures accuracy, reliability, and robustness in safety-critical tasks such as localization [6] and perception [22] in self-driving. In typical multi-sensor setups, the sensors are attached to a common rigid body where the spatial relationship between them can be obtained through a rigid transformation matrix. It is important to identify the exact values of those matrices to correctly exploit the data provided by the sensors. The process of finding these spatial transformations is called extrinsic calibration, which is a topic that has been and is still being heavily studied thanks to the increasing popularity of multi-sensor algorithms. In addition to extrinsic calibration, without an external synchronization system, it is also necessary to perform temporal calibration. With incorrect temporal calibration, performance on different tasks can be severely hindered. Even though some of the existing methods take temporal miscalibration into account [12, 29, 35], most of them assume that the sensors are correctly synchronized. Due to the importance of sensor calibration, a multitude of calibration solutions exist in the literature, as highlighted in the review from Li et al. [17] and as summarized in Table 1. These methods can be classified into two main categories: target-based methods and targetless methods.

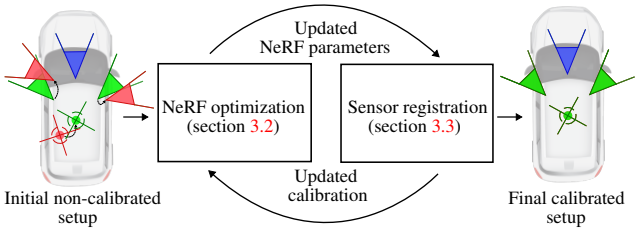



Figure 1. **Method overview –** SOAC is a novel multimodal spatio-temporal calibration method for cameras and LiDAR in the context of autonomous driving. By alternating the training of multiple implicit scenes (section 3.2) and sensors co-registration from these representations (section 3.3), SOAC achieves precise self-supervised calibration from raw data acquired in unconstrained urban environments.

Target-based calibration methods rely on one or more elements of known dimensions and features purposefully placed in the scene. The most classic target is a checkerboard [9, 47], but custom-made planar targets [11] or boxes [30] have also been proposed. These methods usually offer precise and robust calibration compared to targetless approaches. However, requiring hand-placed targets

prevents them from being deployed on a large scale and does not enable on-the-fly re-calibration if needed. Thus, a more suitable method for mass-produced autonomous driving cars would be targetless.

Targetless methods do not require manually placed targets and thus can be used on sequences captured without user intervention. This makes them more suitable for large-scale deployment. These approaches usually rely on shared information (i.e. overlap) between the different sensors, which can be of different modalities. Wang et al. [38] propose a correspondence between the reflectivity of the LiDAR scans and the grayscale intensity of the camera images. Other methods propose to find matches of specific features, like edges [45] or semantic classes [16].

Following the development of deep learning, methods relying on deep models were introduced to calibrate RGB images and LiDAR scans. These methods have the advantage of being fast and precise, enabling reliable online calibration. Deep learning techniques can leverage regression [13, 20, 32], flow [15], keypoints [43] or convolutional features [7] to supervise or regularize the training. However, as they are supervised methods, they need an accurately calibrated training dataset to be optimized and have issues with cross-domain data due to overfitting to a specific dataset or sensor layout.

Recently, with the arrival of Neural Radiance Fields (NeRF) [24] for implicit representation of 3D scenes, some works [12, 41, 48] propose to take advantage of the fully differentiable structure of the model to achieve self-supervised targetless calibration. Using a NeRF as the common frame for the sensors, these methods are able to densely correlate the captured observation from different sensors in an implicit volumetric space. Yet, by simultaneously learning the information from multiple sensors, the NeRF might overfit regions of the scene only visible from a single sensor without enforcing consistency on the overlapping regions. This causes the calibration to easily get stuck in a local minimum.

We take inspiration from the aforementioned works by exploiting the fully differentiable properties of the implicit scene representation to achieve spatial and temporal calibration. Different from existing methods [12, 41, 48], we propose to represent the scene by using multiple NeRFs akin to their corresponding sensor and advocate to alternate the optimization target between NeRF training and sensor calibration (i.e. Figure 1). Our method avoids overfitting the pose optimization to partial regions of the scene, resulting in a more robust and accurate calibration.

## 2. Related Work

With NeRF and the papers improving upon it [1, 26], the main focus was on the quality of novel view synthesis in addition to training and rendering speeds. However, since these approaches often validate their claims on carefully curated datasets, it is often assumed that the input poses corresponding to the data are already available and are accurate. However, in real-world situations, some or all the captured frames might be unposed or suffer from inaccuracies, hence, significantly impacting the quality of the final reconstruction result [19]. Therefore, many works later on attempted to tackle this issue through different formulations and adaptations of the overall optimization problem.

Table 1. Comparison of calibration methods.

| Methods | | Targetless | Cam/Cam | Cam/LiDAR | Temporal | Self-supervised |
|---|---|---|---|---|---|---|
| **Target-based** | Zhang et al. [47] | X | X | ✓ | X | - |
| | Geiger et al. [9] | X | X | ✓ | X | - |
| **Feature-based** | Wang et al. [38] | ✓ | X | ✓ | X | - |
| | Park et al. [29] | ✓ | X | ✓ | ✓ | - |
| **Deep-learning** | RegNet [32] | ✓ | X | ✓ | X | X |
| | LCCNet [20] | ✓ | X | ✓ | X | X |
| **NeRF-based** | INF [48] | ✓ | X | ✓ | X | ✓ |
| | MOISST [12] | ✓ | ✓ | ✓ | ✓ | ✓ |
| | SOAC (our method) | ✓ | ✓ | ✓ | ✓ | ✓ |

**NeRF-based Image Registration.** To register an image with incorrect or no pose, iNeRF [44] proposes to use an already trained NeRF. It finds the pose that minimizes the photometric difference between the captured image and the rendered result from the model. By focusing on regions of interest, it is able to register unseen images with high precision. Using this idea as a basis, Loc-NeRF [21] combines Monte Carlo localization method [5] with the use of a pre-trained NeRF as a map, to build a real-time global localization method. CROSSFIRE [25] takes advantage of the NeRF model's flexibility to learn not only the radiance and density information of the map, but also a descriptor field. During the localization process, by iteratively matching the descriptors from the query image and the information given by the NeRF model, this method is able to provide high precision localization. Nevertheless, all these methods require training a NeRF from precise camera poses first before being able to localize new query images.

**NeRF-based Pose Optimization.** The first method to leverage the fully differentiable nature of NeRF to optimize the input poses through backpropagation is NeRF-- [40]. It proposes to optimize both the NeRF and the input poses by representing them as embeddings and show higher novel view synthesis quality when trained from noisy poses. BARF [19] improves upon this idea by adding a coarse-to-fine component to this method. It progressively liberates the frequencies of the input positional encoding to pre-

vent the optimization from getting stuck in local minimum. SCNeRF [14] adds camera distortion estimation and uses a different 6-vector rotation formulation in the optimization, while SPARF [36] achieves pose optimization with sparse input views by relying on pixel matching and depth consistency.

While the aforementioned methods need an initial estimate of the camera poses, some recent methods completely remove the need for prior poses. NoPe-NeRF [2] uses an off-the-shelf monocular depth estimator (i.e. DPT [31]) to regularize relative poses between successive images. GNeRF [23] relies on adversarial learning to coarsely estimate the initial poses before refining them in a second phase. IR-NeRF [46] improves upon GNeRF by regularizing the implicit pose estimator with the unposed real images, increasing its robustness.

Although the NeRF-based pose optimization methods achieve reasonable scene reconstruction by recovering accurate camera poses, they are not suited for autonomous driving data as they are not handling multi-modal observations neither taking into account the rigidity constraint between multiple sensors mounted on a vehicle.

**NeRF-based Calibration.** Considering a multi-sensor rig, accurate calibration is crucial to exploit the data provided by the sensors together. NeRF-based calibration methods [12, 41, 48] take advantage of the rigid constraint between the sensors and the differentiable nature of NeRF to efficiently solve this challenging task. These methods have the advantage of being targetless and self-supervised, as they do not rely on an annotated training dataset. The idea is to use a NeRF as a common scene representation. Each sensor provides its observation (RGB images, depth measurement or point clouds), to both train the NeRF to represent the scene and to optimize its own extrinsic calibration parameters to fit the NeRF representation. In INF [48], the goal is to find the extrinsic transformation between a 360° camera to a LiDAR. First, the density network of NeRF is trained using the LiDAR depth data. Then, the whole scene's radiance is trained using images, while simultaneously calibrating the camera. This method is limited to the calibration of a single 360° camera with a LiDAR, whereas autonomous driving systems rely on multiple cameras with narrower fields of view. AsyncNeRF [41] calibrates a pair of camera and depth sensors. It takes into account the temporal miscalibration between the sensors, by building a trajectory function. Nevertheless, the time offset is provided as input and not determined through optimization, which limits its utilization for spatio-temporal calibration. MOISST [12] proposes to accomplish temporal calibration in addition to extrinsic calibration, and to do so with any number of LiDARs and cameras, by training the NeRF with all the data, while also optimizing the prior extrinsic transformations and time offsets. By using a single NeRF to fuse the information from all the sensors, we cannot prevent degenerate cases where the estimation of the extrinsic parameters of one sensor diverges and causes the NeRF to learn a wrong scene geometry without correlating multi-sensor observations. Our method, SOAC, aims to achieve better robustness and calibration performance by leveraging the use of multiple NeRFs to counterbalance such limitations.

## 3. Method

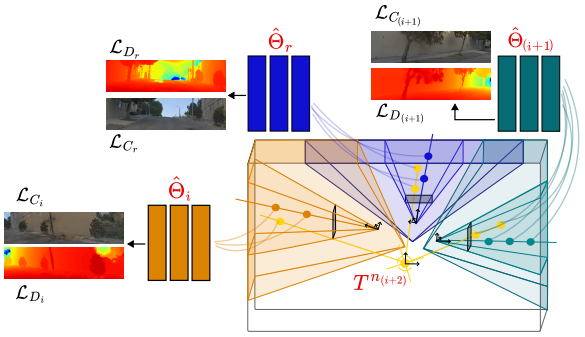
Our multi-sensor calibration problem is formulated as follows: given a vehicle trajectory and initial priors of sensor positions mounted on the vehicle, we aim to recover the exact spatio-temporal calibration of the sensors on the vehicle. Our method is composed of two optimization steps that are performed sequentially all along the training (cf. Figure 1). The first step consists of training multiple implicit scene representations (NeRFs), one by camera, using only the observations from the dedicated sensor. During the second optimization step, we refine the extrinsic and temporal parameters of each sensor using the trained NeRF of all the other sensors in a round-robin manner. The motivation behind this design is to prevent over-fitting, calibration divergence, or implicit model convergence to a poor local minimum when all the observations are fused within the same implicit representation, as in MOISST [12].

### 3.1. Notations and Background

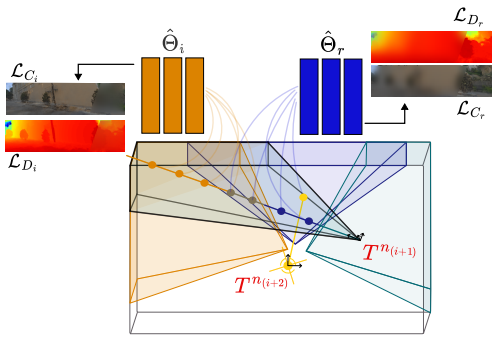
**Notations.** Without loss of generality, we consider the trajectory of camera $r$ (our reference sensor) as the known trajectory of the vehicle. We use the same notations introduced in MOISST [12] to describe our method:

- $S = \{C, L\}$: the set of sensors composed of at least one or more cameras $C$ and, optionally, one or more LiDARs $L$,
- $\{F_i\}$: the set of frames captured by the sensor $i \in S$,
- $t^{n_i} \in \mathbf{R}^+$: the timestamp of frame $n_i \in F_i$ relative to the sensor $i \in S$,
- $\delta_i \in \mathbf{R}$: the time offset between the reference camera and the sensor $i \in S$ ($\delta_r = 0$),
- ${}_wT^i(t) \in \mathbf{R}^{4\times4}$: the pose of sensor $i \in S$ at time $t$ (the time is relative to sensor $i$'s own clock) in the world reference frame,
- ${}_jT^i \in \mathbf{R}^{4\times4}$: the transformation matrix from sensor $i$ to sensor $j$.

Our goal is to find the optimal transformations ${}_r\hat{T}^i$ and time offsets $\hat{\delta}_i$ of the different sensors to calibrate with respect to the reference camera. The poses of the reference camera $r$ can be obtained by relying on an IMU, SLAM [27], or Structure-from-Motion [33]. Similar to MOISST, we build a continuous trajectory of the reference




(a) **Scene representation training step –** The parameters $\hat{\Theta}$ of each NeRF are trained with the images from their associated cameras and the LiDAR scans. The LiDAR calibration is also optimized through $T^{n_{(i+2)}}$.




(b) **Extrinsic and temporal optimization step –** The real frame from the sensor is compared to the predicted frame on the other NeRFs to calculate the losses. The calibration is then optimized with backpropagation through the poses $T^{n_{(i+1)}}$ and $T^{n_{(i+2)}}$.

Figure 2. Overview of the two training steps.

sensor $r$, $\mathcal{T}_r$, from the discrete poses of $r$ using linear interpolation for the pose translation and spherical linear interpolation (SLERP [34]) for the rotation. This trajectory is expressed as a function of time, that returns the pose of the reference camera $r$ for any given time $t$: ${}_wT^r(t) = \mathcal{T}_r(t)$. Using the extrinsic transformations and the time offsets between the other sensors and the reference camera $r$, we can compute the absolute pose of sensor $i$ at specific timestamps with the following equation:

$$ {}_wT^i(t^{n_i} + \delta_i) = \mathcal{T}_r(t^{n_i} + \delta_i)\, {}_rT^i. \tag{1} $$

In order to simplify the equations, we designate the absolute pose of sensor $i$ computed from its extrinsic as $T^{n_i} = {}_wT^i(t^{n_i} + \delta_i)$.

**NeRF model.** NeRF is a function of parameters $\Theta$ that takes as input rays obtained from a sensor's intrinsic parameters and pose, and generates for each ray color and density information via volumetric rendering. This information can be combined into a color image $\mathcal{R}_I\left(T^{n_i} \mid \Theta\right)$ and a depth scan $\mathcal{R}_D\left(T^{n_i} \mid \Theta\right)$ of frame $n_i$ for sensor $i$.

### 3.2. Scene Representation Training

For each camera sensor, a dedicated NeRF with parameters $\Theta_i$ is trained using rays that are generated exclusively from camera $i$. Each NeRF model with parameters $\Theta_i$ will only learn the part of the scene that is observed by its respective camera sensor $i$ (cf. Figure 2a). The color loss for training the scene representation is:

$$ \mathcal{L}_C = \sum_{i \in C} \sum_{n_i \in F_i} \left\| \mathcal{R}_I\left(T^{n_i} \mid \Theta_i\right) - I^{n_i} \right\|_2^2, \tag{2} $$

with $I^{n_i}$ the color image $n_i$ of camera $i$. The training objective is to estimate the optimal parameters $\hat{\Theta}_i$ for the NeRF models such as:

$$ \left\{\hat{\Theta}_i\right\}_{i \in C} = \underset{\{\Theta_i\}_{i \in C}}{\operatorname{argmin}} \left(\mathcal{L}_C\right). \tag{3} $$

### 3.3. Extrinsic and Temporal Optimization

During the calibration step, our objective is to optimize the extrinsic transformation matrix ${}_rT^i$ and temporal parameters $\delta_i$ by optimizing poses of camera $i$ using all the NeRFs, except the NeRF of parameters $\Theta_i$ associated to the current camera being calibrated (cf. Figure 2b). Using this optimization formulation, we enforce the images captured by each camera to be coherent with the NeRF trained by the other cameras. The camera calibration loss can be written as:

$$ \mathcal{L}_{Cam} = \sum_{j \in C} \sum_{\substack{i \in C \\ i \neq j}} \sum_{n_i \in F_i} \left\| \mathcal{R}_I\left(T^{n_i} \mid \Theta_j\right) - I^{n_i} \right\|_2^2, \tag{4} $$

and by considering equation 1, the optimization objective during the spatio-temporal optimization step is:

$$ \left\{{}_r\hat{T}^i, \hat{\delta}_i\right\}_{i \in C} = \underset{\{{}_rT^i, \delta_i\}_{i \in C}}{\operatorname{argmin}} \left(\mathcal{L}_{Cam}\right). \tag{5} $$

### 3.4. LiDAR Calibration

As a LiDAR only provides geometric information, we cannot register an RGB image to a NeRF which was only trained on LiDAR scans. This means that the registration step (cf. section 3.3) could not be accomplished on a LiDAR-trained NeRF. Instead of dedicating a NeRF for each LiDAR, we both train the camera NeRFs with all the LiDAR scans, and calibrate the LiDARs against all NeRFs (cf. Figure 2). Thus, we have for both the NeRF training step and calibration step:

$$ \mathcal{L}_D = \sum_{j \in C} \sum_{i \in L} \sum_{n_i \in F_i} \left| \mathcal{R}_D\left(T^{n_i} \mid \Theta_j\right) - D^{n_i} \right|, \tag{6} $$

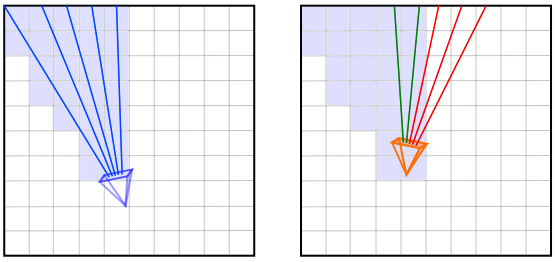
with $D^{n_i}$ the point cloud scan $n_i$ of LiDAR $i$. When adding the LiDAR loss in the objective equation 3, it becomes:

$$\left\{\hat{\Theta}_i\right\}_{i\in C}, \left\{{}_r\hat{T}^j, \hat{\delta}_i\right\}_{j\in L} = \underset{\{\Theta_i\},\{{}_rT^j,\delta_j\}}{\operatorname{argmin}} (\mathcal{L}_C + \mathcal{L}_D). \quad (7)$$

and equation 5 becomes:

$$\left\{{}_r\hat{T}^i, \hat{\delta}_i\right\}_{i\in S} = \underset{\{{}_rT^i,\delta_i\}}{\operatorname{argmin}} (\mathcal{L}_{Cam} + \mathcal{L}_D). \quad (8)$$

### 3.5. Visibility Grid



(a) **Grid filling –** Rays from camera $C_i$ fill the visibility grid linked to Nerf $\Theta_i$.

(b) **Ray filtering –** For cameras $C_{j,\forall j\neq i}$, rays are kept or filtered according to visibility from (a)

Figure 3. Illustration of visibility grid in practice.

In a multi-sensor setup, the portions of the scene observed from the different sensors might not overlap. This can lead to noisy reconstruction on the NeRF model if inference is performed at the unobserved regions (cf. Figure 4). To overcome this problem, NeRF2NeRF [10] performs pairwise registration of two NeRF models produced from different viewpoints by aligning the partially overlapping geometry of the two models. In a similar sense, we aim to consider the overlapping geometry from our different NeRFs that have been learned separately from each camera. To achieve this, a boolean visibility grid for each NeRF model is reconstructed by considering the rays belonging to its akin sensor (see Figure 3a) during the scene representation step (Section 3.2). During the calibration step (Section 3.3) we exploit this visibility grid to only consider rays that overlap with trained regions on each NeRF used for registration (cf. Figure 3b). The grids are reinitialized every few epochs to account for the new poses resulting from the calibration refinements.

### 3.6. Optimization Details

Overall, the training process can be summarized as follows: during each training step, a mini-batch of rays is first used in the scene representation training step (Section 2a). Rays of each camera train their respective NeRF and fill the respective visibility grids (Section 3.5). The LiDAR rays train all the NeRFs and are used to optimize the LiDAR calibration parameters after being filtered by the visibility grids. In a subsequent step, the same mini-batch is passed to the extrinsic and temporal optimization. Rays are filtered through the visibility grids before being fed to the NeRFs as explained in section 3.3. Calibration losses (equation 8) are computed and the gradient is backpropagated to optimize the calibration parameters. Once this is done, we continue the training with the next mini-batch.

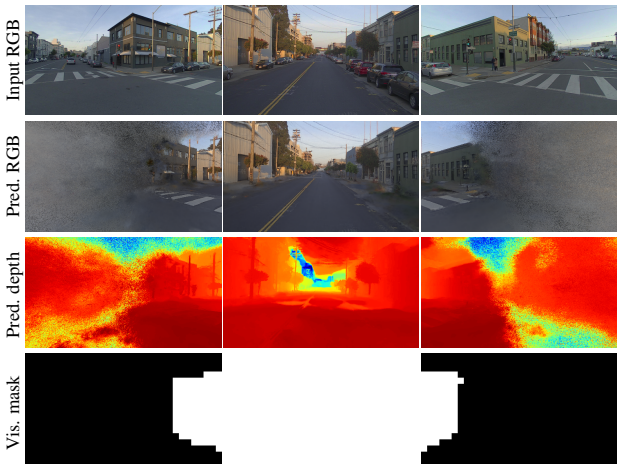



Figure 4. Results using visibility grids on a Pandaset sequence – The predictions are done with the NeRF trained by the front camera.

**NeRF delaying.** In our system, all the sensors, except the reference camera, have incorrect calibration. As such, the NeRF trained with the reference camera is the most adequate for calibration at the beginning. That is why we introduce a delaying schedule for the other NeRFs based on the overlap with the reference camera; more details about this policy are provided in the supplementary materials.

**Correction bounding.** As we consider the extrinsic and temporal calibration on a car, we can suppose that the translation error should not be off by more than the car's size. We can also consider that the sensors should not have a time offset too high, even without the help of an external synchronizing system. Thus, by using an offset and scaled sigmoid function on the output of the embeddings for the translation and temporal correction, we can confine the learned correction, avoiding divergence and increasing the stability and robustness of the calibration.

## 4. Experiments

### 4.1. Setup

**Datasets.** For our experiments, we use three public driving scene datasets: KITTI-360 [18], nuScenes [3] and Pandaset [42]. For KITTI-360, we consider the two front cam-

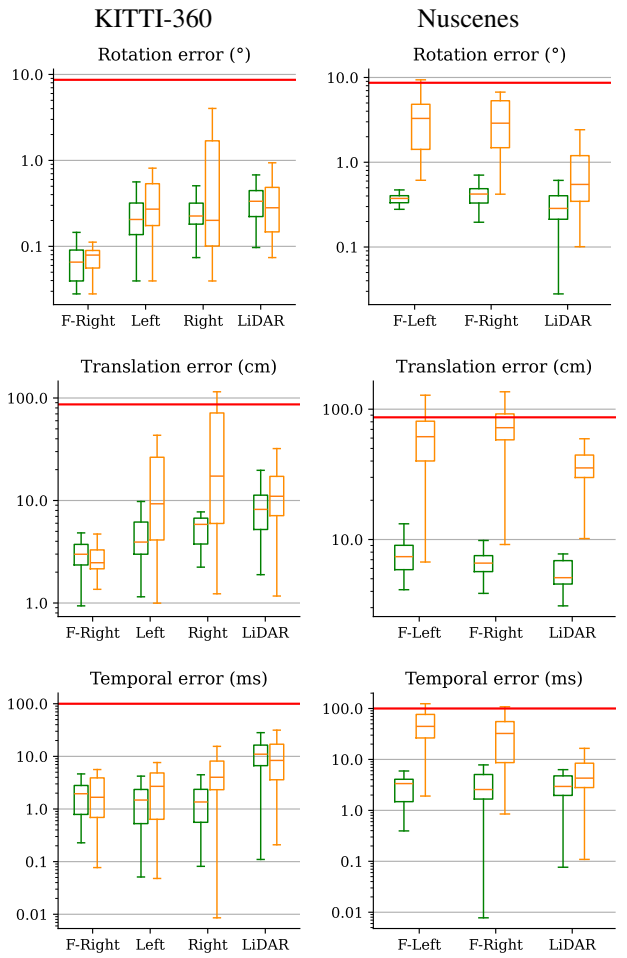



Figure 5. Results for SOAC and MOISST as box plots with log scale on KITTI-360 and Nuscenes sequences. The red lines show the initial error (Best viewed in color).

eras, the two side cameras and the Velodyne LiDAR. For nuScenes and Pandaset, we consider the front camera, the two front diagonal cameras, and the LiDAR. We assign the front-left camera of KITTI-360, and the front camera of nuScenes and Pandaset to be the reference sensor. More details on selected sequences and dataset parameters are provided in the supplementary materials.

**Baseline.** As a comparison baseline we use MOISST, as it is also a method that aims to solve the targetless, multi-modal and spatiotemporal calibration problem. We implemented the method as described by the paper [12] with slight changes (more details in supplementary materials). For SOAC, we use the same architecture as MOISST for the NeRF model and apply the same supervision and regularization losses. More details on the hyperparameters used are given in the supplementary materials. For the LiDAR/Camera calibration task, we compare with LCCNet [20] by using the code and the pretrained weights from the official repository[1].

### 4.2. Results

**Spatial and temporal calibration.** We run the benchmarks on 4 KITTI-360 sequences and 3 nuScenes sequences. For SOAC, we downscale by a factor of 4 the image dimensions of KITTI-360 and by a factor of 6 for nuScenes. For MOISST, we do not downscale the KITTI-360 images and downscale by a factor of 2 nuScenes images as this method performs better with high-resolution images. Each test is run with an initial noise of 50 cm translation error and 5° rotation error on each axis, as well as 100 ms of time offset. We use 10 different seeds to randomly add positive or negative signs for each added noise and compute the error statistics over these 10 runs. We provide the box plots[2] of the results in Figure 5. We obtain better calibration results on KITTI-360 an overall error (average over median for each sensor: `rotation/translation/time offset`) for SOAC: `0.21°/5.24 cm/3.95 ms` compared to MOISST: `0.21°/10.02 cm/4.19 ms`. On nuScenes, the difference is wider as we have for SOAC: `0.36°/6.36 cm/2.96 ms` and for MOISST: `2.24°/56.34 cm/27.07 ms`. Detailed quantitative results by sequence are given in the supplementary materials.

Table 2. LiDAR/Camera calibration results on KITTI-360.

| Method | Rotation (°) | Translation (cm) |
|---|---|---|
| SOAC | **0.33** | **7.75** |
| LCCNet | 1.92 | 95.8 |

**LiDAR/Camera calibration.** For the task of LiDAR/Camera calibration, we compare our method against LCCNet [20]. The provided weights were pretrained on the KITTI odometry dataset [8]. We predict the calibration between the Front-Left camera and the LiDAR of KITTI-360, as it is a setup very similar to KITTI odometry. We can see in Table 2 that the performance of LCCNet, especially on the translation, is very poor compared to SOAC (results per sequence are provided in the supplementary materials). As it is a supervised method, we observe that it is setup-specific, and a slight change in the LiDAR/Camera configuration greatly reduces the performance. This was also highlighted in the work by Fu et al. [7] when using the Front-Right camera for calibration on the KITTI odometry dataset.

[1] https://github.com/IIPCVLAB/LCCNet

[2] The boxes show the first quartile $Q_1$, median, third quartile $Q_3$. The whiskers use 1.5 IQR (Interquartile range) above and below the box and stop at a value within the results

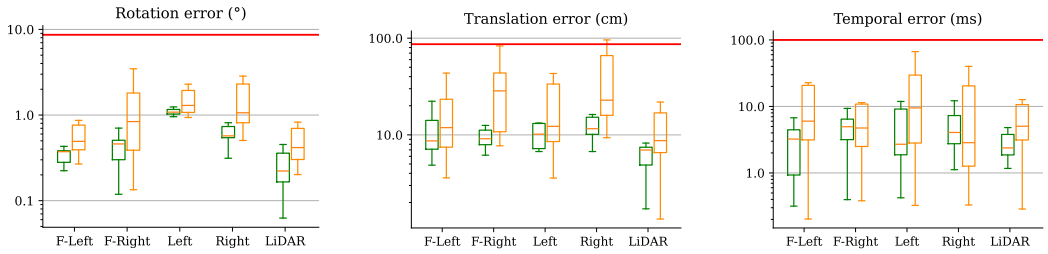



Figure 6. Results on nuScenes with 5 cameras for SOAC and SOAC w/o NeRF delaying as box plots with log scale, the red lines show the initial error (Best viewed in color).

**Calibration in dynamic environment.** For the evaluation on dynamic environment, we use 3 Pandaset sequences with the presence of dynamic elements such as cars or pedestrians. When calibrating on dynamic scenes, the moving elements are not handled by the NeRF model. However, a simple and efficient way of removing these elements is to filter the dynamic classes with semantic segmentation. We lose some useful information for calibration, for instance stationed cars, but if the rest of the scene is sufficiently informational, proper calibration can be obtained. We apply a similar setup to the one for KITTI-360 and nuScenes, except that we remove the temporal calibration and the initial time offset (cf. Section 4.3 on Time-space compensation). We downscale the image by a factor of 4 for SOAC and 2 for MOISST. We use semantic segmentation computed by Mask2Former [4] to remove all classes that can be considered dynamic. We show the results of SOAC vs. MOISST and SOAC without semantic filtering in Figure 7. By using the semantic filtering, we greatly improve the calibration performance on the LiDAR with a median error for SOAC of `0.41°/7.79 cm` compared to MOISST: `2.36°/79.17 cm`. We can also see that SOAC performs much better than MOISST on the overall calibration of all the sensors with error an of `0.42°/11.18 cm` versus `2.18°/44.73 cm`.

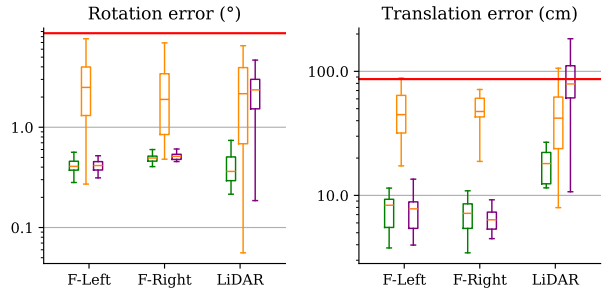



Figure 7. Results on Pandaset for SOAC, MOISST and SOAC w/o semantic filtering as box plots with log scale, the red lines show the initial error (Best viewed in color).

**Complete camera rig calibration.** To evaluate SOAC performances with a nearly complete 360° camera rig, we add two additional side cameras on the nuScenes sequences. We run both with and without the NeRFs delaying scheduling as explained in section 3.6. In Figure 6 we can see the impact of not delaying the NeRFs, as the accuracy and stability of the calibration plummet.

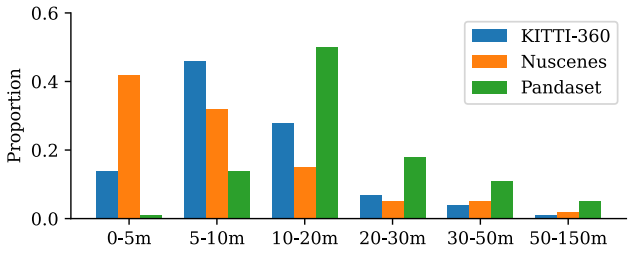



Figure 8. LiDAR ray length distribution of the sequences used in our calibration experiments.

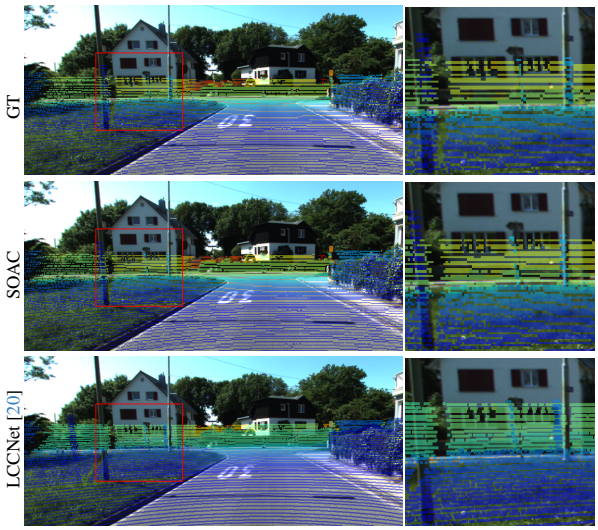



Figure 9. Qualitative LiDAR/front camera calibration results on KITTI-360 dataset.

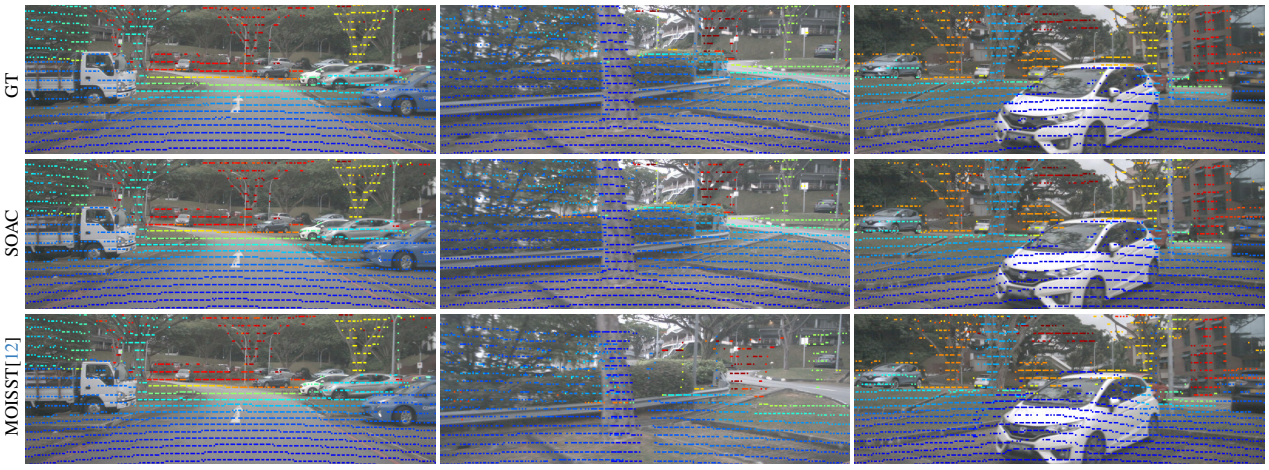



Figure 10. Qualitative LiDAR/Camera reprojection results on nuScenes dataset.

**Qualitative results.** We show the reprojection of the LiDAR on the images using the calibration obtained from different methods. On KITTI-360 (cf. Figure 9) we can see that LCCNet does not provide a satisfying result and that SOAC is able to provide a visually comparable alignment to the ground truth calibration. On nuScenes (cf. Figure **??**), the calibration from SOAC provides a better alignment than MOISST, assessing the quantitative results of Figure 5. More qualitative results are given in the supplementary materials, along with ablation studies on visibility grids (cf. Section 3.5) and correction bounding (cf. Section 3.6).

### 4.3. Limitations

Table 3. Illustration of SOAC space-time compensation on a sequence from KITTI-360 – Mean absolute poses of sensors are correct whereas the spatio-temporal calibration computed by the method is erroneous.

| Errors | | Cam Front | Cam Left | Cam Right | LiDAR |
|---|---|---|---|---|---|
| Translation (cm) | Extrinsic | 47.9 | 67.8 | 70.4 | 50.9 |
| | Poses | 3.5 | 2.6 | 26.2 | 19.1 |
| Rotation (°) | Extrinsic | 0.13 | 0.18 | 0.23 | 0.58 |
| | Poses | 1.63 | 1.12 | 1.35 | 0.60 |
| Time offset (ms) | | 39.18 | 58.16 | 40.74 | 39.38 |

**Time-space compensation.** When simultaneously calibrating spatially and temporally, there are cases where the disentanglement is impossible. In a sequence where the vehicle is driving in a straight line at a constant speed, there is an infinite number of solutions that can provide the correct poses. In Table 3, we show the calibration results on a straight line with constant speed from KITTI-360. We can see the pose error is fairly low, but the extrinsic and temporal calibration is incorrect. This means that there is a need to select scenes with speed variation in order to reduce to a single possible solution. As most Pandaset sequences are in a straight line at a constant speed, we decided to not do temporal calibration on them.

**Scene structure.** When the scenes are more open and/or larger, the projected rays will have to travel a longer distance before reaching the scene's structures. Considering LiDAR to camera calibration, the rotation error has a linearly increasing impact according to the ray distance when reprojected to the camera frame, while the translation error's impact is independent from the ray distance. Thus, we tend to lose precision on the translation as the ray gets longer. When analyzing the LiDAR rays length distribution of the datasets in Figure 8, we observe that the LiDAR rays on Pandaset are longer, meaning that the scenes are larger and open, and the structures are farther than on KITTI-360 and nuScenes. This explains most likely the decrease in calibration performances for the LiDAR extrinsic translation parameters on Pandaset (median error of `18.1 cm`) compared to KITTI-360 (median error of `8.2 cm`) or nuScenes (median error of `5.1 cm`).

**Training time.** As we train one NeRF per camera, and register all the other sensors on each NeRF, the training time increases exponentially with the number of cameras. For instance, on nuScenes one epoch takes approximately 1 minute 45 seconds for 3 cameras and 8 minutes for 5 cameras using the same GPU. This reduces the scalability of our method, but this phenomenon is mitigated by the fact that we use much smaller images than MOISST to reach better performance (see supplementary materials for details about the total training time of the methods).

## 5. Conclusion

In this paper, we presented SOAC, a targetless and self-supervised method for spatial and temporal calibration. This approach is able to simultaneously calibrate multiple sensors of different modalities, by leveraging the use of multiple camera-specific implicit scene representations, and taking into account the overlap between the sensors. Our approach is fully automatic by relying on gradient descent for the optimization process, and surpasses similar methods previously introduced. The reliance on a reference sensor with known trajectory, and the need of near structures for a precise calibration, are restrictions that could open to future research to alleviate them.

## References


[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5470–5479, 2022. 2, 11

[2] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. In *CVPR*, pages 4160–4169, 2023. 3

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11621–11631, 2020. 5, 11

[4] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, pages 1290–1299, 2022. 7

[5] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *IEEE international conference on robotics and automation (ICRA)*, pages 1322–1328, 1999. 2

[6] Jamil Fayyad, Mohammad A Jaradat, Dominique Gruyer, and Homayoun Najjaran. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*, 20(15):4220, 2020. 1

[7] Lanke Frank Tarimo Fu and Maurice Fallon. Batch Differentiable Pose Refinement for In-The-wild Camera/LiDAR Extrinsic Calibration. In *Conference on Robot Learning (CoRL)*, 2023. 2, 6

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. 6

[9] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In *IEEE international conference on robotics and automation (RA-L)*, pages 3936–3943, 2012. 1, 2

[10] Lily Goli, Daniel Rebain, Sara Sabour, Animesh Garg, and Andrea Tagliasacchi. nerf2nerf: Pairwise registration of neural radiance fields. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9354–9361, 2023. 5

[11] Carlos Guindel, Jorge Beltrán, David Martín, and Fernando García. Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *IEEE international conference on intelligent transportation systems (ITSC)*, pages 1–6, 2017. 1

[12] Quentin Herau, Nathan Piasco, Moussab Bennehar, Luis Roldão, Dzmitry Tsishkou, Cyrille Migniot, Pascal Vasseur, and Cédric Demonceaux. MOISST: Multimodal Optimization of Implicit Scene for SpatioTemporal calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023. 1, 2, 3, 6, 8, 11, 15

[13] Ganesh Iyer, R Karnik Ram, J Krishna Murthy, and K Madhava Krishna. CalibNet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1110–1117, 2018. 2

[14] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, pages 5741–5751, 2021. 3

[15] Xin Jing, Xiaqing Ding, Rong Xiong, Huanjun Deng, and Yue Wang. DXQ-Net: differentiable lidar-camera extrinsic calibration using quality-aware flow. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6235–6241, 2022. 2

[16] Akio Kodaira, Yiyang Zhou, Pengwei Zang, Wei Zhan, and Masayoshi Tomizuka. SST-Calib: Simultaneous Spatial-Temporal Parameter Calibration between LIDAR and Camera. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2896–2902, 2022. 2

[17] Xingchen Li, Yuxuan Xiao, Beibei Wang, Haojie Ren, Yanyong Zhang, and Jianmin Ji. Automatic targetless LiDAR–camera calibration: a survey. *Artificial Intelligence Review*, 56(9):9949–9987, 2023. 1

[18] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE TPAMI*, 45(3):3292–3310, 2022. 5, 11

[19] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 2

[20] Xudong Lv, Boya Wang, Ziwen Dou, Dong Ye, and Shuo Wang. LCCNet: LiDAR and camera self-calibration using cost volume network. In *CVPRW*, pages 2894–2901, 2021. 2, 6, 7

[21] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-nerf: Monte carlo localization using neural radiance fields. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4018–4025, 2023. 2

[22] Enrique Marti, Miguel Angel De Miguel, Fernando Garcia, and Joshue Perez. A review of sensor technologies for perception in automated driving. *IEEE Intelligent Transportation Systems Magazine (ITSM)*, 11(4):94–108, 2019. 1

[23] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. Gnerf: Gan-based neural radiance field without posed camera. In *ICCV*, pages 6351–6361, 2021. 3

[24] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[25] Arthur Moreau, Nathan Piasco, Moussab Bennehar, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. CROSSFIRE: Camera Relocalization On Self-Supervised Features from an Implicit Representation. *ICCV*, 2023. 2

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15, 2022. 2, 11

[27] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics (T-RO)*, 31 (5):1147–1163, 2015. 3

[28] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, pages 5480–5490, 2022. 11

[29] Chanoh Park, Peyman Moghadam, Soohwan Kim, Sridha Sridharan, and Clinton Fookes. Spatiotemporal camera-LiDAR calibration: A targetless and structureless approach. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1556–1563, 2020. 1, 2

[30] Zoltan Pusztai and Levente Hajder. Accurate calibration of LiDAR-camera systems using ordinary boxes. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 394–402, 2017. 1

[31] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, pages 12179–12188, 2021. 3

[32] Nick Schneider, Florian Piewak, Christoph Stiller, and Uwe Franke. RegNet: Multimodal sensor registration using deep neural networks. In *IEEE intelligent vehicles symposium (IV)*, pages 1803–1810, 2017. 2

[33] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, pages 4104–4113, 2016. 3

[34] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985. 4

[35] Zachary Taylor and Juan Nieto. Motion-based calibration of multimodal sensor extrinsics and timing offset estimation. *IEEE Transactions on Robotics (T-RO)*, 32(5):1215–1229, 2016. 1

[36] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *CVPR*, pages 4190–4200, 2023. 3

[37] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023. 11

[38] Ruisheng Wang, Frank P Ferrie, and Jane Macfarlane. Automatic registration of mobile LiDAR and spherical panoramas. In *CVPRW*, pages 33–40, 2012. 2

[39] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 11

[40] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 2

[41] Zirui Wu, Yuantao Chen, Runyi Yang, Zhenxin Zhu, Chao Hou, Yongliang Shi, Hao Zhao, and Guyue Zhou. Async-NeRF: Learning Large-scale Radiance Fields from Asynchronous rgb-d Sequences with time-pose function. *arXiv preprint arXiv:2211.07459*, 2022. 2, 3

[42] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101, 2021. 5, 11

[43] Chao Ye, Huihui Pan, and Huijun Gao. Keypoint-based LiDAR-camera online calibration with robust geometric network. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2021. 2

[44] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting Neural Radiance Fields for Pose Estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330, 2021. 2

[45] Chongjian Yuan, Xiyuan Liu, Xiaoping Hong, and Fu Zhang. Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments. *IEEE Robotics and Automation Letters (RA-L)*, 6(4):7517–7524, 2021. 2

[46] Jiahui Zhang, Fangneng Zhan, Yingchen Yu, Kunhao Liu, Rongliang Wu, Xiaoqin Zhang, Ling Shao, and Shijian Lu. Pose-Free Neural Radiance Fields via Implicit Pose Regularization. In *ICCV*, 2023. 3

[47] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2301–2306, 2004. 1, 2

[48] Shuyi Zhou, Shuxiang Xie, Ryoichi Ishikawa, Ken Sakurada, Masaki Onishi, and Takeshi Oishi. INF: Implicit Neural Fusion for LiDAR and Camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023. 2, 3

# Supplementary

## A. Technical Details

### A.1. Datasets

**KITTI-360** [18]: Sequences are selected and cropped by considering vehicle speed variations to remove time-space compensation issues as described in main article Section 4.3. Once sequences are cropped, one out of two frames are kept for all sensors to obtain a total of 40 frames per sequence. This decision was made to match the same length as the NVS benchmark sequences present on the dataset. The details from each sequence are summarized in Table 4.

Table 4. Selected frames for each KITTI-360 sequence.

| Sequence | KITTI-360 run | Starting frame | Ending frame |
|---|---|---|---|
| 1 | 0009 | 980 | 1058 |
| 2 | 0009 | 2854 | 2932 |
| 3 | 0010 | 3390 | 3468 |
| 4 | 0002 | 4722 | 4800 |
| Straight line | 0009 | 220 | 298 |

**nuScenes** [3]: Since nuScnes poses are provided only in $\mathrm{SE}(2)$, they cannot be used directly for our method. Instead, we use KISS-ICP [37] to get a good estimate of the LiDAR poses. Extrinsic calibration provided by the dataset is then used to obtain the poses for all cameras. We select the sequences 916, 410 and 417 for our experiments, as they are more suitable for the calibration (closer structures, more speed variation). All LiDAR scans are used during calibration as the LiDAR is sparser than the one in KITTI-360, while one out of two images is subsampled to reduce training time.

**Pandaset** [42]: Since extrinsic parameters are not provided by the dataset, they are estimated using the global poses of all sensors at several frames by calculating the transformation between the frames with the same timestamp from each sensor. Sequences 33, 40 and 53 are used for our experiments as they have more close structures. We apply the same subsampling strategy as for nuScenes.

### A.2. Architecture and Losses

For our NeRF network architecture, we use the same model as MOISST [12] which is inspired by the `nerfacto` model of Nerfstudio[3] open source project. It uses the combination of two papers. The first one is the proposal network from MipNeRF-360 [1] with two proposal networks for the coarse density estimation and a final NeRF for the radiance and the fine density, improving the geometry of the scene, the rendering quality and reducing the training time. The second one is the hash grid introduced by instant-NGP [26] to replace the deterministic positional encoding, which also accelerates the training. Following the nerfacto implementation, 128 points (instead of 256) per ray are sampled for the first proposal model, 96 points for the second one, and 48 points for the final NeRF model, which outputs our results.

On top of $\mathcal{L}_C$, $\mathcal{L}_{Cam}$ and $\mathcal{L}_D$, two losses for geometric consistency, also used by MOISST, are added: a structural dissimilarity (DSSIM) loss $\mathcal{L}_{SSIM}$ [39], and a depth smoothness loss $\mathcal{L}_{DS}$ from RegNeRF [28].

[3] https://docs.nerf.studio/en/latest/nerfology/methods/nerfacto.html

### A.3. Hyperparameters

Table 5. SOAC hyperparameters used for the training.

| Hyperparameter | Value |
|---|---|
| Number of epochs | 20 |
| Initial calibration lr | 1e-3 |
| Final calibration lr | 1e-4 |
| Visibility grid size | 20 |
| Batch size | 100 |
| Patch size | [15, 15] |
| $\mathcal{L}_C$ coef | 1 |
| $\mathcal{L}_{Cam}$ coef | 1 |
| $\mathcal{L}_{SSIM}$ coef | 0.1 |
| $\mathcal{L}_D$ coef | 1 |
| $\mathcal{L}_{DS}$ coef | 1e-4 |
| Translation bounding | 2 meters |
| Temporal bounding | 500 ms |

Table 6. Dataset specific SOAC hyperparameters used for the training (in epochs).

| Hyperparameter | KITTI-360 | nuScenes | Pandaset |
|---|---|---|---|
| Diagonal cams NeRF delay | - | 1 | 3 |
| Side cams NeRF delay | 1 | 9 | - |
| LiDAR delay | 6 | 5 | 8 |

In Table 5 are indicated the hyperparameters used for the training of SOAC, and in Table 6 the hyperparameters specific for each dataset. The dataset-specific parameters largely depend on the overlap between the sensors, and the frame number/size. Basically, larger overlaps and larger quantities of data reduce the number of delay epochs.

The number of epochs for training MOISST is reduced to 20, as improvement was not observed with more. The spatial and temporal optimization learning rate is fine-tuned to 5e-4.

## B. Additional ablations

**Correction bounding.** The addition of the sigmoid for bounding the translation and temporal corrections allows better stability and robustness as shown in Figure 11 on which a huge decrease in calibration accuracy can be noticed when removing the sigmoid.

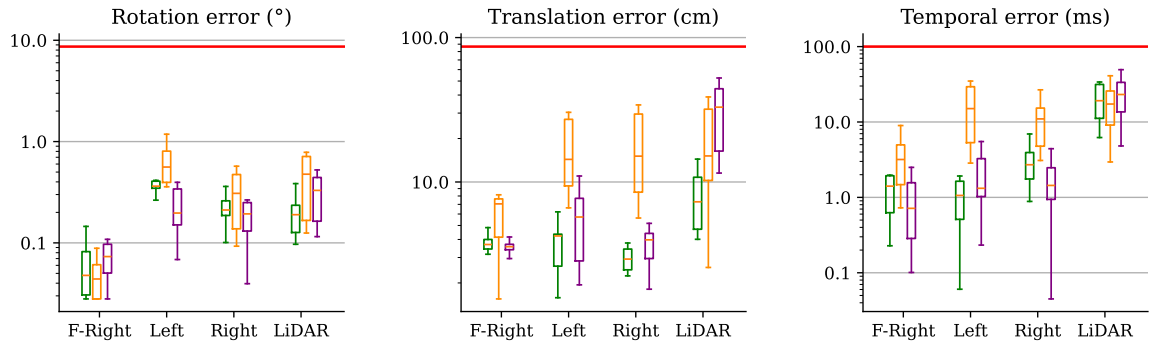



Figure 11. Ablation results on KITTI-360 sequence 4: for SOAC, SOAC w/o Sigmoid and SOAC w/o visibility grid as box plots with log scale, the red lines show the initial error (Best viewed in color).

**Visibility grid.** Removing the visibility grids deteriorates the performance of the LiDAR calibration rotation and translation as shown in Figure 11.

## C. Training time

Table 7. Training time comparison on different sequences.

| Dataset | MOISST | SOAC |
|---|---|---|
| KITTI-360 | ∼ 2 h 30 min | ∼ 1 h 30 min |
| Nuscenes 3 cams | ∼ 2 h 30 min | ∼ 1 h |
| Nuscenes 5 cams | ∼ 4 h 30 min | ∼ 2 h 30 min |
| Pandaset | ∼ 1 h 30 min | ∼ 1 h 20 min |

In Table 7 are indicated the training time for SOAC and MOISST with a high-tier GPU on the sequences. The indicated results are with the downscaled images as described in the paper. SOAC is able to provide better calibration than MOISST with shorter training time, even if multiple NeRFs are used, as it can use much smaller images.

## D. Quantitative results

Specific box plot results are provided for each sequence. The results for KITTI-360 are in Figure 12, the results for Nuscenes in Figure 13, and the results for Pandaset in Figure 14.

On KITTI-360, MOISST seems to provide results on par with SOAC on the Front-right camera and the LiDAR. However, on the side cameras, there is a significant difference in the stability of the calibration. On Nuscenes and Pandaset, SOAC is much more precise and stable than MOISST all across the board.

## E. Qualitative results

In Figure 15 and Figure 16 are shown LiDAR/Camera projection on nuScenes and Pandaset sequences. The calibration optimized by SOAC provides substantially better alignment than the one from MOISST.

Figure 17 shows the predicted images and masks from each NeRF trained with different cameras. The visibility masks are coherent with the predicted RGB images, allowing correct filtering for SOAC.

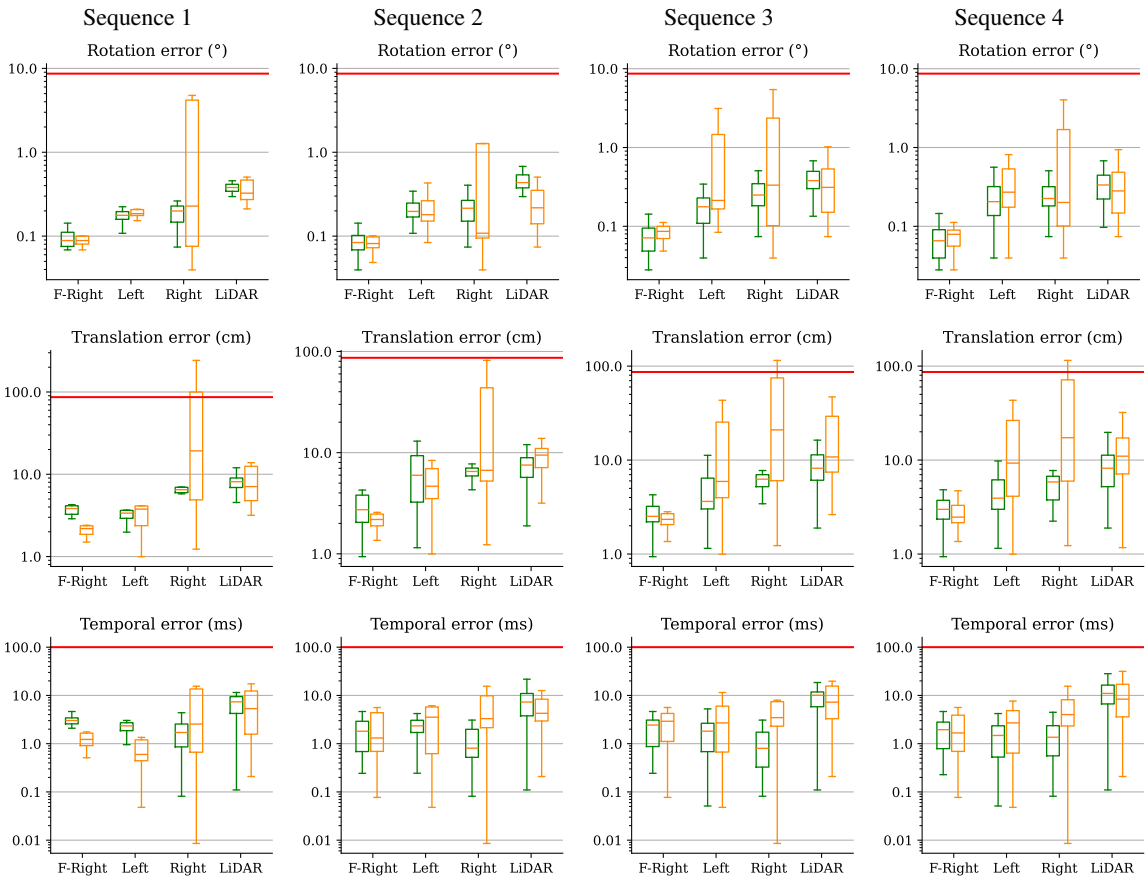



Figure 12. Results for KITTI-360 per sequence for SOAC and MOISST as box plots with log scale. The red lines show the initial error (Best viewed in color).

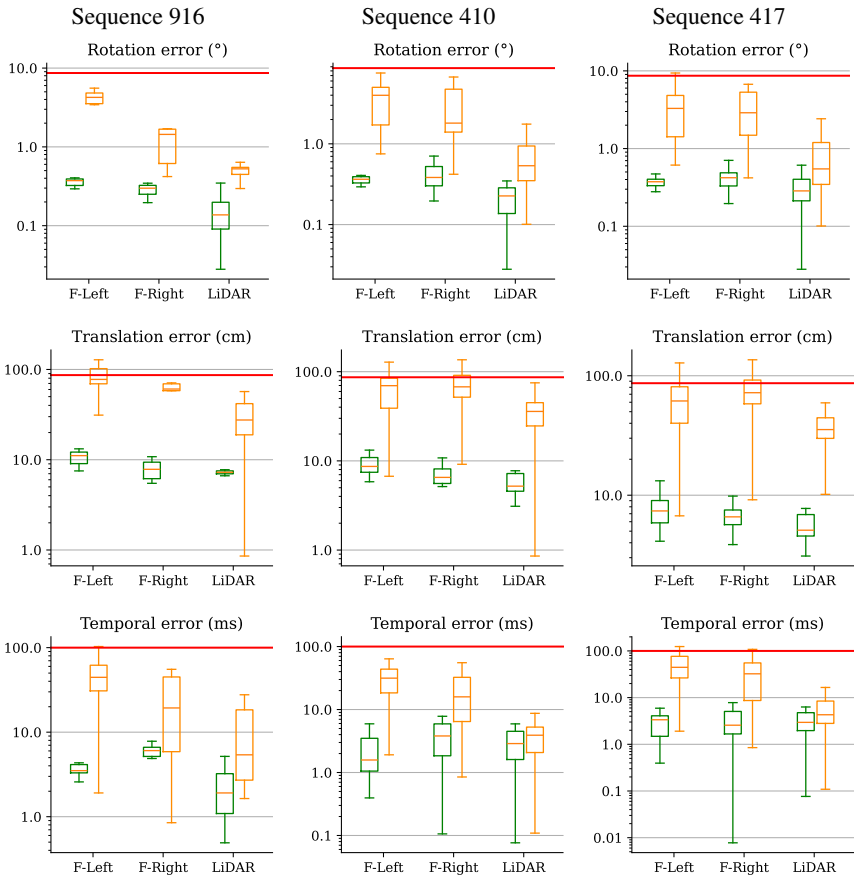



Figure 13. Results for nuScenes per sequence for SOAC and MOISST as box plots with log scale. The red lines show the initial error (Best viewed in color).

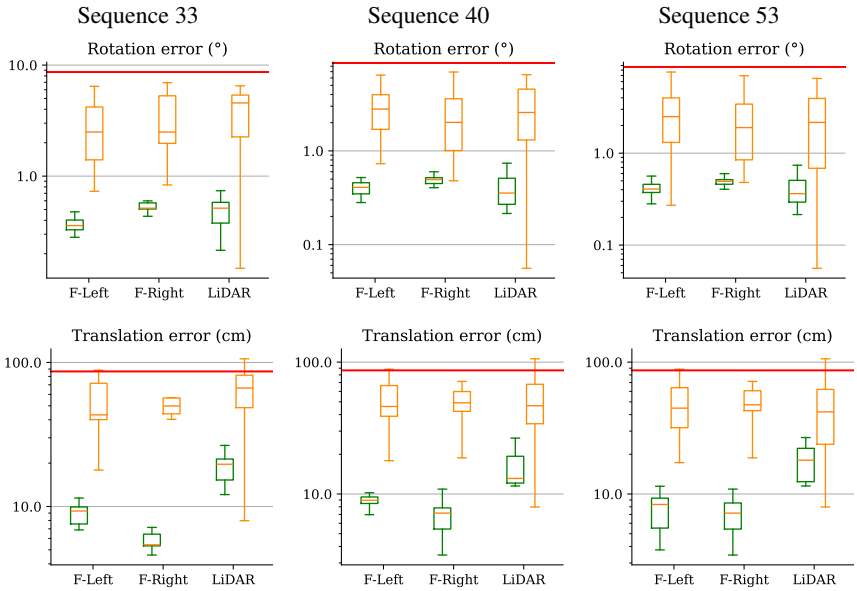



Figure 14. Results for Pandaset per sequence for SOAC and MOISST as box plots with log scale. The red lines show the initial error (Best viewed in color).


Figure 15. More qualitative LiDAR/Camera reprojection results on nuScenes dataset.

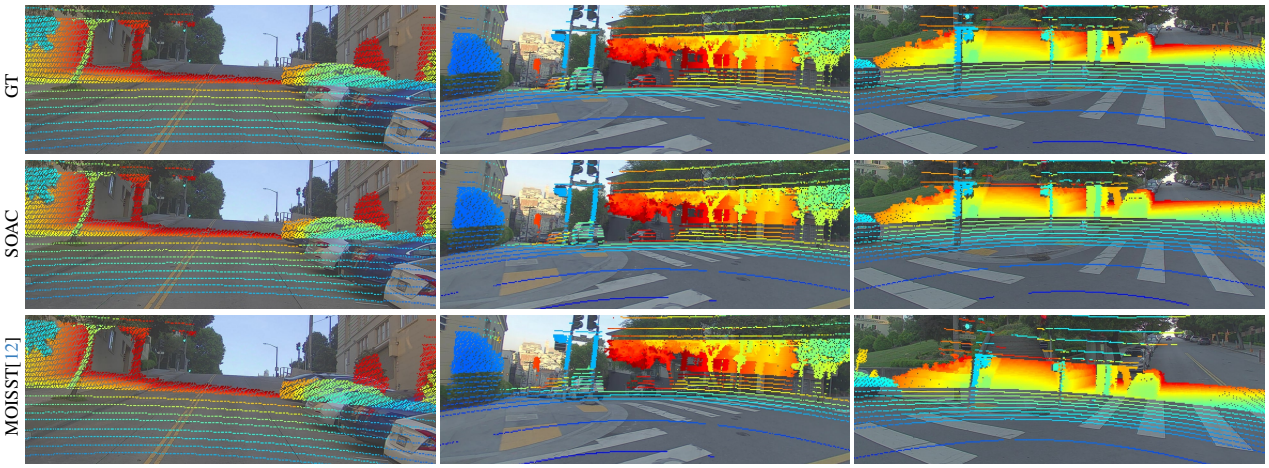



Figure 16. Qualitative LiDAR/Camera reprojection results on Pandaset dataset.


Figure 17. Results using visibility grids on a Pandaset sequence – Prediction from different NeRFs.