UniCal: Unified Neural Sensor Calibration Supplementary Material

Ze Yang[1,2⋆], George Chen[1,3⋆†], Haowei Zhang[1], Kevin Ta[1], Ioan Andrei Bârsan[1,2], Daniel Murphy[1], Sivabalan Manivasagam[1,2], and Raquel Urtasun[1,2]

[1] Waabi [2] University of Toronto [3] University of Waterloo
{zyang,gchen,hzhang,kta,abarsan,dmurphy,siva,urtasun}@waabi.ai

Abstract. In the supplementary material, we provide information about the implementation details of our method, along with details about the baselines, experimental settings, additional results and the discussion of limitations and potential negative social impact of our method. We first describe the implementation details of our approach in Appendix A1. Then in Appendix A2 we present additional details on the baseline model implementations and their adaptation to our setting. Next, we explain in more detail our experimental setting and datasets in Appendix A3. After that, we showcase additional experiments, including comaparison to additional learning-based (CalibNet [8], LCCNet [15]) and NeRF-based (MOISST [7]) calibration methods, additional results and visualizations in Appendix A4. Finally, we analyze the limitations of our model and discuss the future works and potential negative social impact in Appendix A5. Please visit https://waabi.ai/unical for an overview of our methodology and video results for multi-sensor calibration.

A1 UniCal Details

Scene Representation Model: Our scene representation model is based on a multi-resolution feature grid and `MLP` network. Following [16], we employ a spatial hash function to map each feature grid to a fixed number of features, with the hash table size set to 2^{21}. To obtain the signed distance value s and appearance feature $\mathbf{f}$ from the interpolated feature (Eq. 1 in main paper), the `MLP` network consists of two layers, with a hidden size of 64. For distant regions outside the scene volume, we adopt an inverted sphere parameterization similar to NeRF++ [27].

Camera and LiDAR Intensity Decoder: The camera RGB decoder $\mathcal{D}_{\text{cam}}$ (Eq. 3 in main paper) and the LiDAR intensity decoder $\mathcal{D}_{\text{lidar}}$ (Eq. 5 in main paper) are both three layer MLPs. They take the queried appearance feature $\mathbf{f}$ and view direction encoding $\mathbf{d}$ as input and output the RGB color and LiDAR intensity. To account for variations in exposure and color tone across different sensors, we learn a per-sensor linear mapping over the intensity channel. We found this simple model to be effective in capturing these variations in practice.

⋆ Indicates equal contribution. † Work done while an intern at Waabi.

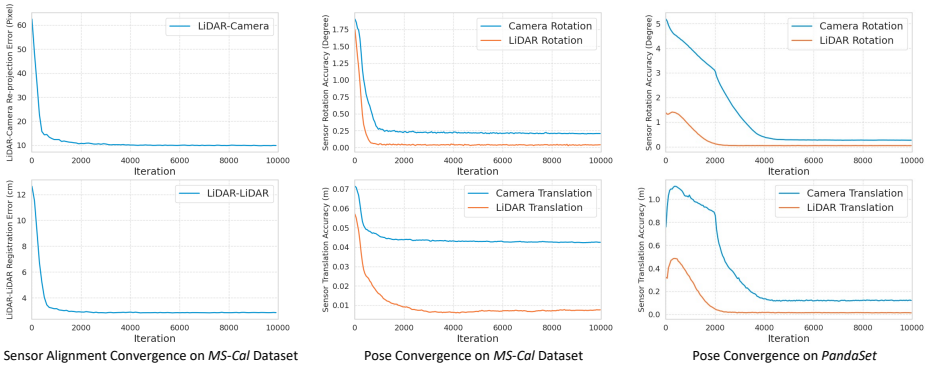

Fig. A1: Convergence of sensor alignment metrics and pose metrics. The calibration typically converges within around 10K iterations ($\approx$ 30 minutes on an A5000 GPU) for both *MS-Cal* and *PandaSet* datasets. **Left:** Convergence of LiDAR-Camera re-projection error (pixel) on *MS-Cal* checkerboard data and LiDAR-LiDAR registration error (cm) on *MS-Cal* outdoor data. **Middle:** Convergence of sensor pose error on *MS-Cal* dataset. **Right:** Convergence of sensor pose error on *PandaSet* dataset.

Rendering Details: To perform efficient volume rendering, we leverage the geometry priors from LiDAR observations to identify near-surface regions, enabling the evaluation of the radiance field exclusively within these areas. This significantly reduces the number of required samples and radiance queries. Specifically, we generate an occupancy grid for the scene volume using the aggregated LiDAR point clouds similar to [26], with a voxel size set to be 0.5 m. We sample query points with a fixed step of 10 cm for regions inside the scene volume, and sample an additional 16 points for the distant sky region during volume rendering.

Sensor Pose Representation: The choice of sensor pose parameterization plays a crucial role for pose-optimizing NeRFs. For instance, certain parameterizations of rotation, such as Euler angles, are known to lack continuity over the $\mathbb{SO}(3)$ manifold, posing challenges in the learning process. To address this, we use a continuous 6D representation [31] to parameterize the sensor rotation and 3D vector to parameterize the sensor translation.

Rolling Shutter Modelling: LiDAR sensors usually accumulate measurements over time, it takes non-negligible time to finish the scan of a full sweep. If the data collection platform is in motion during this period, the LiDAR scan may become distorted due to changes in the sensor pose throughout the scan, known as the rolling shutter effect. To accurately render the LiDAR sweep and model the rolling shutter effect, we interpolate the vehicle pose $\mathbf{P}_{\text{veh}}(t)$ for each LiDAR laser at the firing timestamp t and composite it with the sensor extrinsic to obtain the per ray sensor pose, we then generate the LiDAR ray using Eq. 4 in main paper for volume rendering.

	Run-time ↓
MLCC	1 hr
ICP + Edge and Plane + PGO	3 hr (18 sensor pairs * 10 min)
SC-NeRF	1 day
INF	1 day
Ours	30 min

Table A1: Comparison of calibration time on the *MS-Cal* dataset.

Surface Alignment Distance: To compute the surface alignment distance (Eq. 12 in main paper), we first select candidate image pairs that have an overlapping field-of-view. Then we run SuperPoint [4] and LightGlue [13] to identify the correspondences between the image pairs. During training, we randomly select a image for each camera sensor and its candidate pair image to compute the alignment loss in each iteration. We filter out correspondences with a re-projection error $\|\pi(\mathbf{p}) - \mathbf{u}\|_2 > 50$ for PandaSet [25] and $\|\pi(\mathbf{p}) - \mathbf{u}\|_2 > 20$ for our collected *MS-Cal* dataset. We also filter out correspondences with a ray termination probability $\sum_{i=1}^{N} w_i < 0.5$.

Ray Sampling Details: During the coarse-to-fine ray sampling phase, we run SuperPoint [4] to detect 2048 keypoints for each camera image and progressively apply Gaussian blur to create the blurred heat maps. The initial Gaussian blur kernel has a $\sigma = 40$ and the final gaussian blur kernel has a $\sigma = 5$.

Training Details: We employ a multi-stage training schedule. For the initial 2000 iterations, the model is trained with uniform ray sampling, and transit to coarse-to-fine ray sampling from iteration 2000 until the end of training. Throughout the training process, we utilize the Adam optimizer with a initial learning rate of 0.01 for the scene model and 0.0001 for the sensor poses. The learning rates are exponentially decayed by 0.1 for the scene model and by 0.01 for the sensor poses. During the training, we dynamically adjust the number of sample rays in each iteration to ensure a fixed sample points of 2^{19}. The allocation of rays is evenly distributed among sensors to ensure an equal number of sampled rays for each sensor. Regarding the loss weights in the learning objective, we set $\mathcal{L}_{\text{rgb}} = 1$, $\mathcal{L}_{\text{int}} = 0.1$, $\mathcal{L}_{\text{depth}} = 0.1$, $\mathcal{L}_{\text{align}} = 0.001$, and $\mathcal{L}_{\text{reg}} = 0.01$. We train the model for 30K iterations in total. Notably, we observe that calibration typically converges within around 30 minutes on an A5000 GPU. Fig. A1 shows the convergences of LiDAR-Camera re-reprojection error, LiDAR-LiDAR registration error, and sensor pose accuracy for the initial 10k iterations ($\approx$ 30 minutes). However, we opt to extend the training to enhance calibration further.

Run-time and Resources: Table A1 reports the calibration runtimes compared to other methods. Our approach is more efficient than the other NeRF-based baselines due to our efficient scene representation and rendering, as well as the surface

alignment constraints. While classical calibration methods are in principle fast, they operate on each sensor pair and the time scales linearly unless parallelized for multi-sensor setups. Additionally, existing public implementations of classical calibration such as MLCC [14] or Edge and Plane [28] can be slow, complicating direct comparisons. Besides improved performance, UniCal is more scalable as it does not require expensive infrastructure and operational overhead. This allows calibration in any location without needing to build calibration sites.

A2 Baseline Implementation Details

A2.1 Classical Calibration Baselines

Point-to-Plane ICP: Point-to-Plane ICP [21] is a commonly used algorithm for LiDAR odometry and LiDAR-LiDAR calibration that is typically more efficient and exhibits better average performance [19] than point-to-point ICP. As these algorithms are widely available, we employ the implementation present in Open3D [29]. We run the Point-to-Plane ICP to calibrate each LiDAR pair with sufficient co-visible field-of-views. The calibration is conducted on the five stationary outdoor scenes that include a variety of poles, walls, and distant objects for calibration.

MLCC: MLCC [14] is a targetless sensor extrinsic calibration method for camera and LiDAR sensors. We leverage this framework to perform LiDAR-LiDAR calibration. Specifically, MLCC first uses an adaptive voxelization technique to extract and match LiDAR feature points, subsequently formulating the multi-LiDAR extrinsic calibration problem as a LiDAR Bundle Adjustment (BA) problem. We employ the implementation from the official repo[4] to conduct calibration on our outdoor collects, yielding LiDAR-LiDAR calibration results. We empirically found the MLCC performs slightly worse than Point-to-Plane ICP [21], possibly due to the absence of distinctive structures/features in the outdoor scene data.

Edge and Plane: For LiDAR-camera calibration, we use a custom implementation of a commonly-used target-based method [28]. This approach optimizes for both plane and edge correspondences given known dimensions of a checkerboard target. Improvements over the existing utility available in the MATLAB toolbox involve better target segmentation and edge correspondence matching between identified image edges and extracted point cloud edges. We exploit more accurate initial CAD estimates for coarse bounding box segmentation and use the known laser scan lines as priors for target edge detection. This approach ensures we can reject poor matches in low co-visible regions. To improve robustness, we ensure a variety of target poses are captured and cover the full field-of-view of each camera. The calibration is performed in an indoor environment to control for lighting conditions and improve target visibility.

[4] https://github.com/hku-mars/mlcc

Mutual Information: Another approach for LiDAR-camera calibration is to leverage the correlation between passive material reflection of visible light and LiDAR reflectivity of near infrared wavelengths. A popular approach is to maximize the mutual information [18] of the grayscale image and the intensity of LiDAR returns. We employ a modified implementation of [23] across 10 stationary outdoor scenes for each co-visible LiDAR-camera pair. This approach includes image pre-processing steps including a gaussian blur with a standard deviation of 5 pixels and histogram equalization. We limit the optimization space to within 20 cm and 2 degrees of the initial guess for each translational and rotational DoF.

Pose Graph Optimization: To unify LiDAR-camera [28], LiDAR-LiDAR [2], and LiDAR-INS [1, 5] calibration results, we employ a global pose graph optimization [3,29] to align the full sensor setup. As the methods employ different sensing and registration modalities, we unify the pose graph optimization with empirical weights to ensure that traversal of the calibration graph is fully self-consistent. Pose-graph optimization allows for the averaging across multiple registrations of the same sensors, across sensors, and across registration methods. As the modalities operate in different domains, however, the process requires careful tuning of the relative information matrices of each edge, which is a tedious and time-consuming process.

A2.2 Neural Rendering Calibration Baselines

Self-Calibrating NeRF: SC-NeRF [9] jointly optimizes the neural field, camera pose, intrinsics, and distortion model. In our experiments, we modified the training algorithm to jointly optimize multiple cameras by randomly sampling a camera at each iteration, sampling rays and computing the loss for the selected camera. To ensure that the extrinsics can be properly evaluated against the ground truth, we do not optimize the intrinsics and distortion parameters when training. To match our robot sensor platform setting, we learn a fixed (sensor to vehicle) transformation for each camera from the (given) ground truth vehicle poses at each timestep rather than learning the individual camera to world transformations at each timestep. In our experiments, we use the NeRF++ [27] backbone and the same settings as their Tanks and Temples [11] evaluation with a total of 1.5 million training iterations. We only replace their scene normalization factor with 100 and 60 for Pandaset [25] and *MS-Cal* respectively to reflect the larger scene sizes.

Implicit Neural Fusion: INF [30] first jointly learns a neural field and pose for a LiDAR sensor, then learns a radiance field and transformation from LiDAR frame to camera frame. In our experiments, we again modified the training algorithm to learn on multiple LiDARs and multiple cameras by sampling a sensor at random at each training iteration, and sampling rays and computing the loss as usual. For both LiDARs and cameras, we learn the fixed (sensor to vehicle)

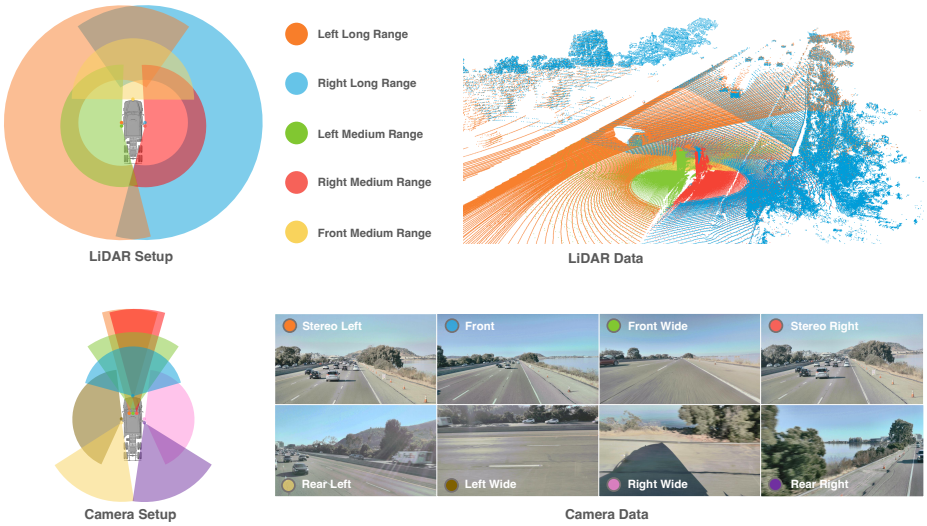

Fig. A2: *MS-Cal* dataset sensor setup and captured data. We show the LiDAR and camera data on the static outdoor collect.

transformation from the (given) ground truth vehicle pose at each time step, instead of learning the per-frame sensor to world transformation as was originally done for LiDAR. We also elected to replace their internal pose representation with 6DoF [31] as we found that the original pose representation was not well defined for the initial sensor extrinsics. We also run all experiments with a pinhole camera intrinsics model to match the cameras present in *PandaSet* and *MS-Cal* datasets. In our experiments, we base the hyperparameters on the settings for their outdoor scene. For both datasets, we set the max depth and scene normalization factor for the depth network to 150m. For Pandaset, we train the density model for 300k iterations and for *MS-Cal* we train for 750k iterations. For the color model, we set the scene normalization factor to 150m, but set the far range to 50m as we found that this helped with stability. We also lowered the learning rate of the camera poses to $5e^{-4}$. We train the color model for 1.2 million and 1.6 million iterations for Pandaset and *MS-Cal* respectively, to compensate for the additional cameras present in these datasets.

A3 Experiment Details

A3.1 Multi-Sensor Calibration Dataset

We collect a *MS-Cal* dataset to study the calibration performance of our proposed method and baselines. Additionally, we investigate the impact of driving trajectories on the calibration performance. The data collection vehicle is a Class 8 truck equipped with five mechanical spinning LiDARs. Among these, two are long-range LiDARs (with a range of up to 200m) positioned on the left and right

Camera Type	Camera Name	Paired LiDAR Names
Narrow FoV	Stereo-Left	Long-Range-Left, Long-Range-Right
	Stereo-Right	Long-Range-Left, Long-Range-Right
Medium FoV	Front	Long-Range-Left, Long-Range-Right
	Rear-Left	Long-Range-Left, Medium-Range-Left
	Rear-Right	Long-Range-Right, Medium-Range-Right
Wide FoV	Front	Long-Range-Left, Long-Range-Right, Medium-Range-Front
	Left	Medium-Range-Left
	Right	Medium-Range-Right

Table A2: List of LiDAR-camera pairs for computing re-projection error on *MS-Cal* checkerboard data.

	Narrow FoV Cameras		Medium FoV Cameras			Wide FoV Cameras		
	Stereo-Left	Stereo-Right	Front	Rear-Left	Rear-Right	Front	Left	Right
Mutual Info [17]	58.75	40.10	12.59	16.87	37.46	52.55	56.50	27.89
Edge and Plane [28]	11.15	9.63	**2.28**	**6.67**	**12.59**	11.11	14.63	11.65
Pose Graph Optim [29]	20.46	9.68	2.89	7.77	13.75	10.14	**12.86**	**11.31**
SC-NeRF [9]	115.81	114.98	13.83	36.82	45.09	28.66	86.82	34.17
INF [30]	54.66	24.84	3.43	15.56	63.21	57.20	120.69	61.59
Ours	**7.78**	**8.74**	2.82	6.89	12.94	**9.91**	15.97	12.68

Table A3: LiDAR-camera re-projection error (in pixel) on *MS-Cal* checkerboard data. We report the breakdown metric for each camera sensor. The average metric is in Tab. 1 in main paper.

sides of the truck. The remaining three are medium-range LiDARs (with a range of up to 50m) mounted at the front, left, and right sides of the truck to provide near-range sensing. The LiDAR setup ensures comprehensive 360° coverage. In addition to LiDARs, the data collection vehicle is equipped with eight cameras. These include three wide-angle cameras oriented towards the front, left, and right. Furthermore, three medium-angle cameras are placed to capture views from the front, rear left, and rear right. Lastly, two long-range stereo cameras are positioned at the front to provide far-distant observations. Please refer to Fig. A2 for a visual representation of the sensor setup and the data captured. The dataset includes indoor data and outdoor data. For the indoor data, the data collection vehicle remains stationary, while a checkerboard is positioned at various locations and orientations for camera-LiDAR pair calibration. One collect is utilized for optimizing classical calibration methods, while another is held out for evaluating LiDAR-camera sensor alignment metrics. For the outdoor parking lot data, we collected both stationary and dynamic trajectories, including eight stationary scenes and four "figure-8" ∞ loops. Two ∞ loops were selected for training neural-rendering methods, and five stationary scenes were used for classical LiDAR alignment calibration. The remaining ∞ loops and stationary scenes were reserved for evaluating the calibrations. To delve into the influence

LiDAR Name	Paired LiDAR Name
Long-Range-Left	Long-Range-Right, Medium-Range-Left, Medium-Range-Front
Long-Range-Right	Long-Range-Left, Medium-Range-Right, Medium-Range-Front
Medium-Range-Front	Long-Range-Left, Long-Range-Right, Medium-Range-Left, Medium-Range-Right
Medium-Range-Left	Long-Range-Left, Medium-Range-Front
Medium-Range-Right	Long-Range-Right, Medium-Range-Front

Table A4: List of LiDAR-LiDAR pairs for computing registration error on *MS-Cal* outdoor data.

	Long Range LiDAR		Medium Range LiDAR		
	Left	Right	Front	Left	Right
Point-to-Plane ICP [2]	2.535	**2.521**	**3.056**	3.042	**2.942**
MLCC [14]	2.842	2.829	3.215	3.318	3.194
Pose Graph Optim [29]	2.660	2.702	3.170	3.180	3.167
INF [30]	6.559	6.800	9.626	12.526	11.158
Ours	**2.516**	2.612	3.078	**3.018**	3.064

Table A5: LiDAR-LiDAR registration error (in cm) on *MS-Cal* outdoor data. We report the breakdown metric for each LiDAR sensor. The average metric is in Tab. 1 in main paper.

of driving trajectories on calibration performance, we also collect six additional outdoor dynamic trajectories. These trajectories include two flower loops, two circular loops, one S curve, and one straight path. Please refer to Fig. A8 for an illustration of different trajectories.

A3.2 Urban Driving Dataset

To evaluate our method on urban driving dataset, we choose public available real-world *PandaSet* [25], which contains 103 urban driving scenes captured in San Francisco. Each scene spans 8 seconds, equivalent to 80 frames sampled at 10Hz. The data collection platform consists of a 360° mechanical spinning LiDAR as well as a forward-facing LiDAR, along with six cameras. These cameras are facing front, front-left, left, back, front-right, and right. We calibrate all the sensors, including the two LiDARs and six cameras. Please see Fig. 1 in main paper for the sensor setup. To quantitatively evaluate our approach against baseline methods that are computationally intensive to train, we selected scenes that have few dynamic actors as the calibration logs. Our selected logs also have different driving trajectories (*e.g.* incline, turning) and feature rich geometric elements in the scene (*e.g.* parked vehicles). We selected four logs `028, 039, 040, 053` for calibration training. We chose two scenes for reconstruction evaluation: `034, 056`. This necessitated the training of eight reconstruction and rendering models for each baseline.

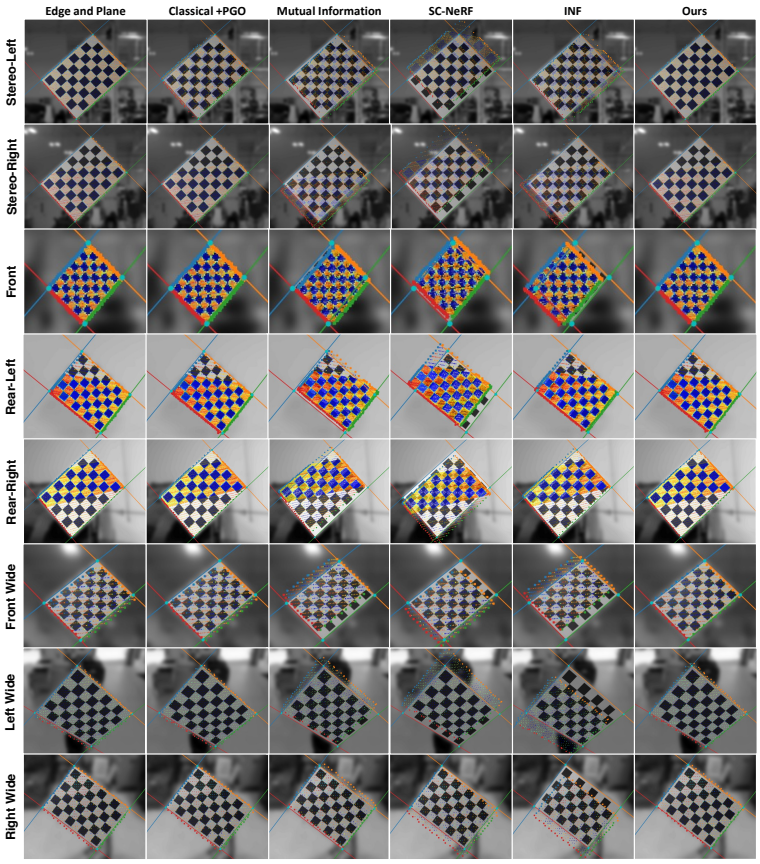

Fig. A3: Visualization of LiDAR-Camera alignment on *MS-Cal* checkerboard data for each camera sensor. We colored the detected checkerboard edge from both LiDAR point cloud and camera images. Additionally, the LiDAR points on the checkerboard plane are colored with intensity value.

A3.3 Reference Calibration for Pose Accuracy Metrics

To evaluate the pose accuracy metrics, we report the average rotation and translation error between the reference and the estimated calibrations. We now describe how we obtain the reference calibration for *PandaSet* and *MS-Cal* datasets.

PandaSet Dataset: For *PandaSet* [25], we use their provided calibration file[5] as the reference for ground truth. It is noted that this file exclusively contains relative poses between different sensors, but does not provide a reference pose

[5] https://github.com/scaleapi/pandaset-devkit/blob/master/docs/static_extrinsic_calibration.yaml

Fig. A4: More qualitative comparison on *PandaSet* dataset. We show the projection of LiDAR and camera images for each camera sensor. Please zoom-in to see the mis-calibrations.

of the sensors to the vehicle. To establish the pose between the sensors and the vehicle frame of reference, we run Iterative Closest Point (ICP) algorithm between the raw LiDAR (in sensor coordinates) and the pose-processed LiDAR (in vehicle coordinates) on the static log `004`. This process enabled us to determine the $\mathbb{SE}(3)$ transform between the 360° mechanical spinning LiDAR and the vehicle frame. In log `004`, the data collection vehicle remains stationary, eliminating rolling shutter effects. The computed rotation from the 360° mechanical spinning LiDAR to the vehicle frame (FLU convention) is represented in quaternion as: `{w:-6.9577e-01, x:5.8054e-03, y:5.2777e-03, z:-7.1823e-01}`. The translation is given by: `{x:7.8202e-01, y:1.1396e-04, z:1.8596e+00}` in meters.

MS-Cal Dataset: For our collected *MS-Cal* dataset, we utilized both classical calibration and brute-force blackbox optimization to establish the ground-truth reference. We first calibrate each LiDAR-LiDAR pair using Point-to-Plane ICP [2] on the outdoor stationary collects, and we calibrate each LiDAR-camera pair using Edge and Plane [28] correspondences on the indoor checkerboard colects, and we calibrate the long-range LiDAR to Inertial Navigation System (INS) based on LiDAR odometry [1, 5]. Subsequently, we run pose graph optimization to derive the optimal global alignment for full sensor calibration. Finally, we run blackbox optimization [20] to search the reference calibration that minimizes both

Fig. A5: More qualitative comparison on *PandaSet* dataset. We show the projection of LiDAR and camera images for each camera sensor. Please zoom-in to see the mis-calibrations.

LiDAR-camera re-projection error (evaluated on the indoor checkerboard data) and LiDAR-LiDAR registration error (evaluated on the outdoor static data) on the evaluation collects. The search space for optimization was identified by analyzing the range of pose discrepancies between the different evaluated calibration methods.

A3.4 Evaluation Metric Details

For the **LiDAR-camera alignment metric**, we report the average re-projection error measured in pixels between the corners of the checkerboard planes derived from the LiDAR points and those in the images. This assessment is conducted in a 1080×1920 resolution image. For the **LiDAR-LiDAR alignment metric**, we compute the average Point-to-Plane distance (cm) for all inlier correspondences for each LiDAR pair on the stationary evaluation scenes. To identify inlier correspondences, we set a maximum correspondence distance of 30cm. Regarding the **pose accuracy metrics**, we report the average rotation error (in degree) and translation error (in meter) between the reference calibration and the estimated calibration. Since some of the baseline methods we compare against do not perform calibration with respect to a reference point on the vehicle, we designate a root sensor and align its calibrated pose with its reference before computing the metric. Specifically, for *PandaSet*, the root sensor is set as the 360° mechanical

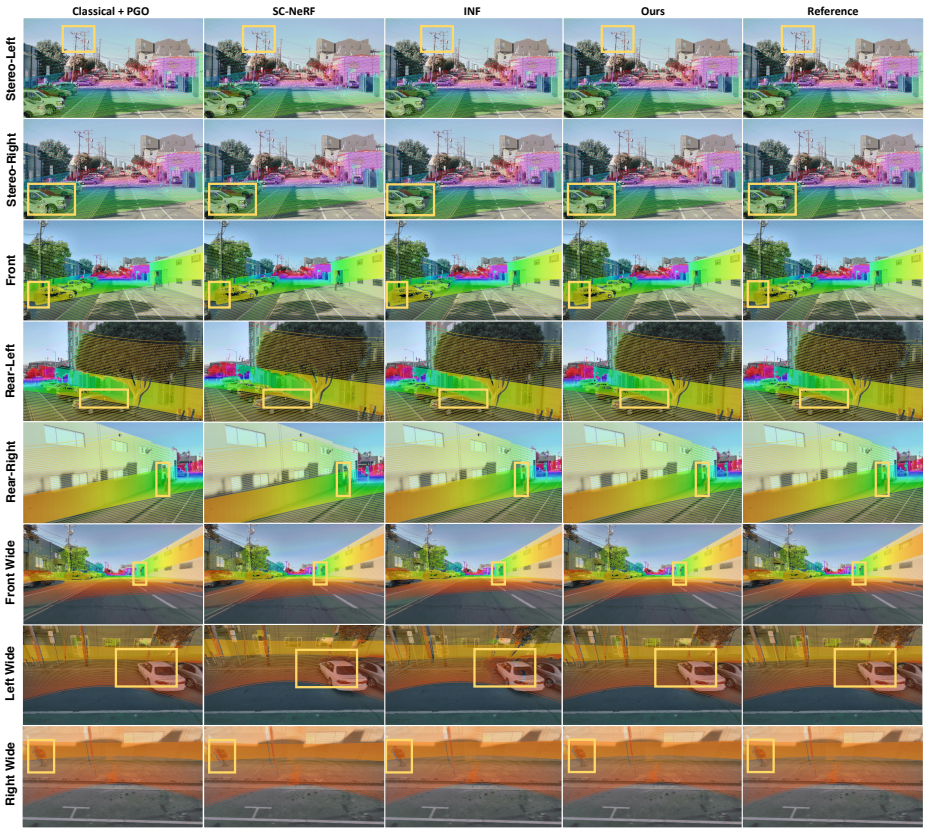

Fig. A6: More qualitative comparison on *MS-Cal* dataset. We show the projection of LiDAR and camera images for each camera sensor. Please zoom-in to see the mis-calibrations.

spinning LiDAR, while for the *MS-Cal* dataset, it is the long-range mechanical spinning LiDAR. For the **rendering metrics**, we train a neural rendering model [26] for each rendering scene, considering the calibration result from each calibration scene. This entails training a total of $N_{\text{calib}} \times N_{\text{render}}$ neural rendering models for each baseline. Each model is trained on every other frame and evaluated on the remaining frames. The reported rendering metrics represent the averages across the $N_{\text{calib}} \times N_{\text{render}}$ models for each baseline. Note that to ensure fair comparison, the neural rendering method is fixed across all methods, and only the input calibration from each evaluated method changes - We optimizing the neural rendering model given the evaluated calibration result.

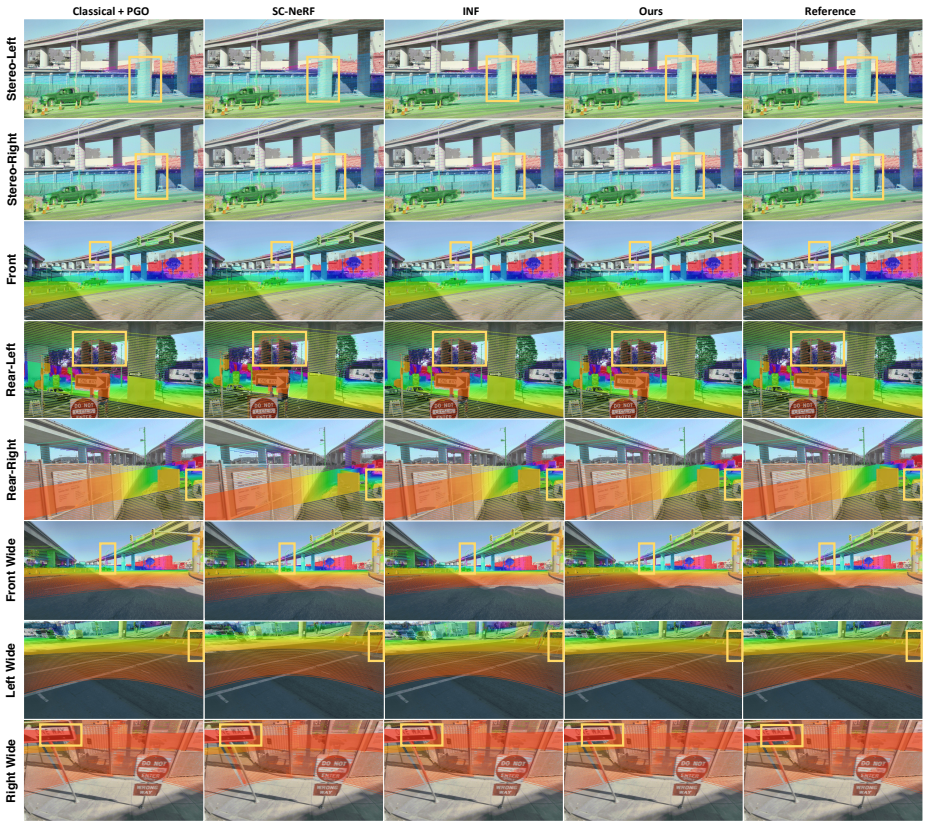

Fig. A7: More qualitative comparison on *MS-Cal* dataset. We show the projection of LiDAR and camera images for each camera sensor. Please zoom-in to see the mis-calibrations.

A4 Additional Experiments and Analysis

In this section, we provide additional quantitative and qualitative results, additional comparison to learning-based (CalibNet [8], LCCNet [15]) and NeRF-based (MOISST [7]) methods, analysis on the calibration initialization and driving trajectory, the feature-ness of the scenes, and additional ablation study. We also show that UniCal improved calibration enables more realistic reconstruction and simulation of driving scenes.

A4.1 Additional Sensor Alignment Results

Tab. A2 shows all the LiDAR-camera pairs used to compute re-projection error, we report the LiDAR-camera re-projection error for each camera in Tab. A3. Additionally, Tab. A4 shows all the LiDAR-LiDAR pairs used to compute registration error, and the corresponding LiDAR-LiDAR registration error for each

Method	Stereo-Left Camera		Stereo-Right Camera	
	Rotation↓	Translation↓	Rotation↓	Translation↓
CalibNet [8]	$5.83 \pm 2.93°$	14.36 ± 6.37 cm	$5.77 \pm 2.93°$	14.22 ± 6.37 cm
LCCNet [15]	$0.17 \pm 0.45°$	$\mathbf{1.29 \pm 1.94}$ cm	$1.52 \pm 0.71°$	52.49 ± 0.51 cm
Ours	$\mathbf{0.12 \pm 0.07}°$	2.16 ± 0.49 cm	$\mathbf{0.12 \pm 0.05}°$	$\mathbf{1.96 \pm 0.87}$ cm

Table A6: Comparison of calibration accuracy to learning-based methods on *KITTI-odometry* dataset. 10° rotation error and 20 cm translation error are added on each axis. CalibNet [8] and LCCNet [15] are both trained on stereo left camera, we also report metrics on stereo right camera.

Perturbation	Method	Front-Right Camera		LiDAR	
		Rotation↓	Translation↓	Rotation↓	Translation↓
Rotation 2°	MOISST	$0.09 \pm 0.01°$	1.6 ± 0.3 cm	$0.39 \pm 0.09°$	9.2 ± 1.9 cm
	Ours	$\mathbf{0.06 \pm 0.01}°$	$\mathbf{0.9 \pm 0.2}$ cm	$\mathbf{0.26 \pm 0.05}°$	$\mathbf{1.5 \pm 0.4}$ cm
Rotation 5°	MOISST	$0.07 \pm 0.05°$	1.7 ± 0.5 cm	$0.40 \pm 0.14°$	9.9 ± 2.3 cm
	Ours	$\mathbf{0.07 \pm 0.01}°$	$\mathbf{0.9 \pm 0.2}$ cm	$\mathbf{0.31 \pm 0.08}°$	$\mathbf{1.8 \pm 0.5}$ cm
Rotation 10°	MOISST	$15.81 \pm 0.02°$	127.2 ± 0.6 cm	$17.07 \pm 0.13°$	104.4 ± 1.7 cm
	Ours	$\mathbf{0.11 \pm 0.04}°$	$\mathbf{1.6 \pm 0.6}$ cm	$\mathbf{0.37 \pm 0.20}°$	$\mathbf{1.8 \pm 0.5}$ cm
Transl 20 cm	MOISST	$0.09 \pm 0.02°$	1.8 ± 0.4 cm	$0.46 \pm 0.1°$	8.7 ± 1.5 cm
	Ours	$\mathbf{0.06 \pm 0.01}°$	$\mathbf{1.2 \pm 0.2}$ cm	$\mathbf{0.19 \pm 0.02}°$	$\mathbf{2.1 \pm 0.5}$ cm
Transl 50 cm	MOISST	$0.09 \pm 0.02°$	1.7 ± 0.5 cm	$0.5 \pm 0.06°$	7.8 ± 1.2 cm
	Ours	$\mathbf{0.06 \pm 0.01}°$	$\mathbf{1.1 \pm 0.3}$ cm	$\mathbf{0.20 \pm 0.01}°$	$\mathbf{2.3 \pm 0.9}$ cm
Transl 100 cm	MOISST	$0.09 \pm 0.02°$	1.7 ± 0.3 cm	$0.43 \pm 0.08°$	8.8 ± 2.4 cm
	Ours	$\mathbf{0.06 \pm 0.01}°$	$\mathbf{1.2 \pm 0.2}$ cm	$\mathbf{0.21 \pm 0.02}°$	$\mathbf{2.5 \pm 0.7}$ cm

Table A7: Comparison of calibration accuracy to MOISST [7] on *KITTI-360* dataset with different calibration initialization. Our method can recover from large rotational error (10°) while MOISST failed to get a satisfactory calibration.

LiDAR is detailed in Tab. A5. Please refer to Fig. A3 for a visual comparison of the LiDAR-camera alignment on the *MS-Cal* checkerboard data for each camera sensor. It can be seen from the figure that our method consistently achieves better sensor alignment compared to baseline methods across all sensors.

A4.2 Additional Qualitative Results

For more qualitative comparisons of projections of LiDAR points and camera images, please refer to Fig. A4 and Fig. A5 for examples from the *PandaSet* dataset. Additionally, we show qualitative results from our collected data on urban driving scenes in Fig. A6 and Fig. A7.

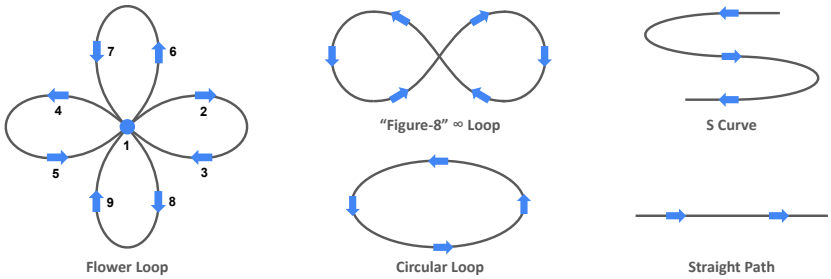

Fig. A8: Illustration of different driving trajectories in *MS-Cal* dataset. Arrows indicate the driving direction, and numbers indicate driving order.

Driving Trajectory	Camera Pose		LiDAR Pose		Rendering Quality		
	Rotation↓	Translation↓	Rotation↓	Translation↓	PSNR↑	SSIM↑	Depth↓
Straight path	0.374	0.046	0.075	0.013	29.59	0.845	0.137
Circular loop	0.271	0.098	0.065	0.030	31.75	0.898	0.040
S curve	0.210	0.050	0.054	**0.008**	31.83	0.899	0.037
∞ loop	0.186	**0.033**	**0.036**	**0.008**	31.96	0.903	**0.035**
Flower loop	**0.178**	0.041	0.039	0.009	**31.97**	**0.904**	**0.035**

Table A8: Analysis of various driving trajectories on *MS-Cal* dataset. Rotational errors are measured in degrees, while translation errors are measured in meters.

A4.3 Additional Comparison with Learning-based Methods

Learning-based approaches [8,15,22,24] formulate extrinsic prediction from camera and LiDAR observations as a supervised learning task. They are effective on the trained sensor configurations/scenes similar to those scene in training and are fast to run. We compare to LCCNet [15][6] and CalibNet [8][7] using the provided pre-trained model. We follow the same setting as in [8, 15] to use the odometry branch of the KITTI [6] dataset. Tab. A6 shows the results on sequence 00. LCCNet and CalibNet pre-trained models are trained on stereo left camera, we report the calibration accuracy results on both stereo-left camera and stereo-right camera by intializeing the calibration with rotation error perturbations of 10° and translation errors of 20 cm on each axis. We use 10 different seeds and compute the error statistics over these 10 runs. LCCNet's performance is good on trained stereo left camera, but degrades on unseen stereo right camera, indicating that learning-based methods do require re-training when the sensor configuration changes and also requires access to the GT calibration for training.

[6] https://github.com/IIPCVLAB/LCCNet
[7] https://github.com/gitouni/CalibNet_pytorch

Correspondance Loss	Camera Pose		LiDAR Pose		Rendering Quality		
	Rotation↓	Translation↓	Rotation↓	Translation↓	PSNR↑	SSIM↑	LPIPS↓
No	0.856	0.614	**0.047**	**0.015**	23.89	0.681	0.500
Projected Ray Dist	0.550	0.622	**0.047**	**0.015**	24.02	0.689	0.492
Surface Alignment Dist	**0.267**	**0.122**	0.048	**0.015**	**25.14**	**0.727**	**0.450**

Table A9: Comparison of Surface Alignment Distance and Projected Ray Distance on *PandaSet* dataset. Rotational errors are measured in degrees, while translation errors are measured in meters.

Sensors	Camera Pose		LiDAR Pose	
	Rotation↓	Translation↓	Rotation↓	Translation↓
Full	**0.267**	**0.122**	**0.048**	**0.015**
Camera-only	0.308	0.133	-	-
LiDAR-only	-	-	0.050	0.017

Table A10: Camera-only and LiDAR-only calibration results on *PandaSet*. Rotational errors are measured in degrees, while translation errors are measured in meters.

A4.4 Analysis on Calibration Initialization and Comparison to NeRF-based Method MOISST [7]

We also study the calibration initialization and compare to MOISST [7]. Specifically, we follow the same setting as in MOISST [7] and report results on KITTI-360 [12] NVS training sequence 1. We consider the front-left (stereo-left) camera as reference sensor and apply up to ± 100 cm translation and $\pm 10^\circ$ rotation offsets on all axes to simulate spatial calibration errors, respectively. For each perturbation level, we use 10 different seeds and compute the error statistics over these runs. Tab. A7 shows the calibration results and comparison to MOISST [7] with different translation perturbation and rotation perturbation initialization. Our method can recover accurate calibration from large rotational and translation errors compared to MOISST [7] due to our additional calibration-inspired enhancements, such as surface alignment constraints.

A4.5 Analysis on Driving Trajectory

To study the calibration performance on different driving patterns, we run our method on the straight path, circular loop, S curve, ∞ loop, and flower loop on our collected *MS-Cal* dataset. Tab. A8 shows the results for each driving trajectory. It can be seen from the tables that straight path and circular loop exhibit inferior performance compared to other trajectories, possibly due to under-constrained observations and incomplete sensor overlap. This implies that running ∞ or flower loops is more favorable for multi-sensor calibration.

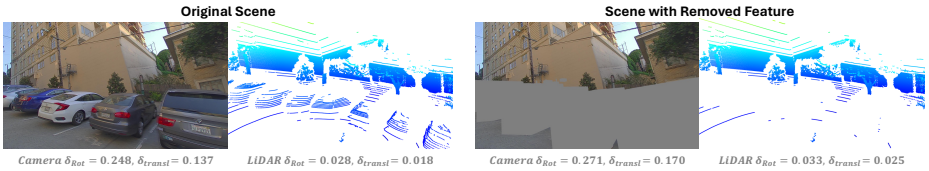

Fig. A9: We remove LiDAR points and camera pixels of objects from existing scenes to simulate the change of scene featureness. **Left:** Camera image and LiDAR point cloud of original scene. **Right:** Scene after removing actors, with corresponding camera image and LiDAR point cloud. We show the Camera and LiDAR pose error w.r.t. the reference below the figures.

Rendering Model	Calibration	PSNR↑	SSIM↑	LPIPS↓	Depth↓
Rasterization [10]	Original	20.91	0.661	0.526	-
	UniCal	**21.27**	**0.666**	**0.523**	-
Raytracing [26]	Original	24.54	0.696	0.474	**0.049**
	UniCal	**25.23**	**0.730**	**0.449**	**0.049**

Table A11: Scene reconstruction and rendering with dataset original and UniCal-refined calibration for rasterization-based 3D-GS [10] and raytracing-based UniSim [26].

A4.6 Performance on Feature-less Scenes

Our method assumes that there is interesting scene geometry with which to reconstruct, and may have challenges on empty scenes with little to no geometry features. We analyze our method's performance when reducing features in the scene by removing annotated actor observations from an existing scene (Figure A9). Specifically, we leverage annotated bounding boxes to identify actors within the scene (*e.g.* vehicles, motorcycles, pedestrians, and construction items), and subsequently remove corresponding camera pixels and LiDAR points. Figure A9 shows a comparison of UniCal's performance on the original scene and the scene with removed features, utilizing PandaSet `Log-040`. The performance drop is small even upon the removal of all annotated objects within the scene.

A4.7 Additional Ablation Study

We further study the effectiveness of the surface alignment loss (Eq. 12 in the main paper) as compared to the projected ray distance proposed in SC-NeRF [9]. The projected ray distance measures the distance between the corresponding rays from pairs of camera images but falls short in ensuring that the 3D structure inferred from these correspondences aligns accurately with the underlying scene representation. Tab. A9 presents a comparison of surface alignment distance and projected ray distance. The table reveals that optimizing the projected ray distance alone faces challenges in recovering accurate camera rotation and

translation. Additionally, we show the camera-only and LiDAR-only calibration results in Tab. A10. The results demonstrate that leveraging multiple sensor modalities leads to better performance.

A4.8 UniCal Improves Scene Reconstruction

We find that with UniCal, we can further refine the calibration from the existing reference provided by *PandaSet* to achieve better scene reconstruction and rendering. We jointly learn the calibration using the calibration logs on *PandaSet* to obtain the refined sensor calibration, and compare this refined calibration with the original calibration on evaluation logs for novel view synthesis. Table A11 shows that for both raytracing-based UniSim [26] model and rasterization-based 3D Gaussian Splatting [10] model, UniCal's refined calibration consistently leads to better scene reconstruction and novel viewpoint rendering.

A5 Limitations and Future Works

Our method focuses on calibrating the sensor extrinsics offline and currently assumes that the intrinsics and trajectory are provided. We also focus on calibrating using static scenes and do not explicitly model changes in lighting or motion. We note that we focus on calibration of LiDAR and camera sensors and exclude calibration of other sensors, such as the IMU sensor. As noted in Table A8, our method also has performance variation depending on the trajectory driven. UniCal also assumes that there is interesting scene geometry with which to reconstruct, and may have challenges on empty scenes with little to no geometry features. Future work will involve extending the method to reduce these assumptions for further robustness and scalability.

Potential Negative Social Impact: Our methods are valuable for self-driving sensor calibration. We acknowledge that there might be privacy concerns arising from data collection used for running UniCal, which can be mitigated through data anonymization techniques. While UniCal significantly reduces costs and operational overhead for calibrating large SDV fleets, we recognize that there may be situations where the calibration results deviate from the reality. Holistic and thorough evaluation of autonomy safety before deploying is critical.

References

1. Barfoot, T.D., Furgale, P.T.: Associating uncertainty with three-dimensional poses for use in estimation problems. IEEE Transactions on Robotics **30**(3), 679–693 (2014) 5, 10
2. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image and Vision Computing **10**(3), 145–155 (1992) 5, 8, 10
3. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: CVPR. pp. 5556–5565 (2015) 5

4. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 224–236 (2018) 3
5. Furgale, P., Barfoot, T.D.: Visual teach and repeat for long-range rover autonomy. Journal of field robotics **27**(5), 534–560 (2010) 5, 10
6. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: CVPR. pp. 3354–3361. IEEE (2012) 15
7. Herau, Q., Piasco, N., Bennehar, M., Roldão, L., Tsishkou, D., Migniot, C., Vasseur, P., Demonceaux, C.: MOISST: Multi-modal optimization of implicit scene for spatiotemporal calibration. In: IROS (2023) 1, 13, 14, 16
8. Iyer, G., Ram., R.K., Murthy, J.K., Krishna, K.M.: CalibNet: Geometrically supervised extrinsic calibration using 3D spatial transformer networks. In: IROS (2018) 1, 13, 14, 15
9. Jeong, Y., Ahn, S., Choy, C., Anandkumar, A., Cho, M., Park, J.: Self-calibrating neural radiance fields. In: ICCV. pp. 5846–5854 (2021) 5, 7, 17
10. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (ToG) **42**(4), 1–14 (2023) 17, 18
11. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017) 5
12. Liao, Y., Xie, J., Geiger, A.: Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(3), 3292–3310 (2022) 16
13. Lindenberger, P., Sarlin, P.E., Pollefeys, M.: LightGlue: Local Feature Matching at Light Speed. In: ICCV (2023) 3
14. Liu, X., Yuan, C., Zhang, F.: Targetless extrinsic calibration of multiple small fov lidars and cameras using adaptive voxelization. IEEE Transactions on Instrumentation and Measurement **71**, 1–12 (2022) 4, 8
15. Lv, X., Wang, B., Dou, Z., Ye, D., Wang, S.: LCCNet: LiDAR and camera self-calibration using cost volume network. In: CVPR Workshop. pp. 2894–2901 (2021) 1, 13, 14, 15
16. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding (2022) 1
17. Pandey, G., McBride, J., Savarese, S., Eustice, R.: Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information. In: AAAI. vol. 26, pp. 2053–2059 (2012) 7
18. Pandey, G., McBride, J.R., Savarese, S., Eustice, R.M.: Automatic extrinsic calibration of vision and lidar by maximizing mutual information. Journal of Field Robotics **32**(5), 696–722 (2015) 5
19. Pomerleau, F., Colas, F., Siegwart, R., Magnenat, S.: Comparing ICP variants on real-world data sets. Autonomous Robots **34**(3), 133–148 (Apr 2013). https://doi.org/10.1007/s10514-013-9327-2 4
20. Powell, M.J.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The computer journal **7**(2), 155–162 (1964) 10
21. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: 3-D Digital Imaging and Modeling. pp. 145–152. IEEE (2001) 4
22. Schneider, N., Piewak, F., Stiller, C., Franke, U.: RegNet: Multimodal sensor registration using deep neural networks. In: 2017 IEEE intelligent vehicles symposium (IV). pp. 1803–1810. IEEE (2017) 15

23. Ta, K., Bruggemann, D., Brödermann, T., Sakaridis, C., Van Gool, L.: L2e: Lasers to events for 6-dof extrinsic calibration of lidars and event cameras. In: ICRA. pp. 11425–11431 (2023) 5
24. Wu, S., Hadachi, A., Vivet, D., Prabhakar, Y.: NetCalib: A novel approach for LiDAR-camera auto-calibration based on deep learning. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 6648–6655 (Jan 2021). https://doi.org/10.1109/ICPR48806.2021.9412653 15
25. Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., et al.: Pandaset: Advanced sensor suite dataset for autonomous driving. In: ITSC (2021) 3, 5, 8, 9
26. Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R.: Unisim: A neural closed-loop sensor simulator. In: CVPR (2023) 2, 12, 17, 18
27. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020) 1, 5
28. Zhou, L., Li, Z., Kaess, M.: Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In: IROS. pp. 5562–5569. IEEE (2018) 4, 5, 7, 10
29. Zhou, Q.Y., Park, J., Koltun, V.: Open3d: A modern library for 3d data processing. arXiv preprint arXiv:1801.09847 (2018) 4, 5, 7, 8
30. Zhou, S., Xie, S., Ishikawa, R., Sakurada, K., Onishi, M., Oishi, T.: INF: Implicit neural fusion for LiDAR and camera. IROS (2023) 5, 7, 8
31. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR. pp. 5745–5753 (2019) 2, 6