Batch Differentiable Pose Refinement for In-The-Wild Camera/LiDAR Extrinsic Calibration

Anonymous Author(s)
Affiliation
Address
email

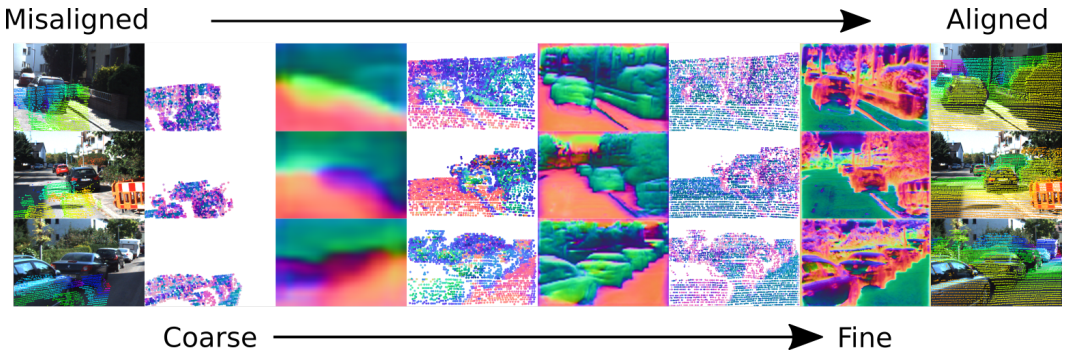


Figure 1: Coarse-to-fine refinement of the LiDAR-to-camera extrinsic parameters. The matching appearance between the LiDAR and camera features is trained using only ground-truth extrinsic parameters for self-supervision. During training, batched refinement helps retain difficult samples that individually would have been discarded. During inference, we show that batched refinement achieves state-of-the-art zero-shot transfer. The rightmost column shows the refined overlay of LiDAR points in the image.

Abstract: Accurate camera to LiDAR (Light Detection and Ranging) extrinsic calibration is important for robotic tasks carrying out tight sensor fusion — such as target tracking and odometry. Calibration is typically performed before deployment in controlled conditions using calibration targets, however, this limits scalability and subsequent recalibration. We propose a novel approach for target-free camera-LiDAR calibration using end-to-end direct alignment which doesn't need calibration targets. Our batched formulation enhances sample efficiency during training and robustness at inference time. We present experimental results, on publicly available real-world data, demonstrating 1.6cm/0.07°median accuracy when transferred to unseen sensors from held-out data sequences. We also show state-of-the-art zero-shot transfer to unseen cameras, LiDARs, and environments.

Keywords: Sensor Fusion, Extrinsic Calibration, Differentiable Optimization

1 Introduction

In many multi-sensor robotic setups, information fusion between any two sensors requires accurate knowledge of the relative transformation between the sensors — the extrinsic parameters. In the case of sensor fusion between a camera and a LiDAR, the extrinsic parameters along with the intrinsic parameters of the camera are used to determine point-to-pixel correspondence displayed in Fig. 1. This correspondence enables fusion in downstream tasks such as object detection, tracking, and ego-motion estimation. Camera/LiDAR extrinsic calibration 'in-the-wild' — meaning in uncontrolled environments without specialized targets — is difficult due to the domain gap between

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

the two sensors. Cameras register textural information but can't directly measure geometry whereas LiDARs measure geometry but can't register texture. Furthermore, cameras are passive sensors and will pick up illumination effects such as shadows, while LiDARs which emit their own light will not register shadows in the visual spectrum. Given these challenges, in-the-wild camera/LiDAR extrinsic calibration is still an active research question.

In this work, we present a framework for camera/LiDAR extrinsic calibration that:

- Can recover the accurate extrinsic parameters from a wide range of initial estimates.
- Learns relevant features for pose alignment from both camera and LiDAR input automatically.
- Generalizes to sensors and environments not encountered during training.

We formulate the camera/LiDAR extrinsic calibration problem as one of batch differentiable direct alignment – aligning a batch of learned features from the LiDAR domain to their corresponding batch of deep image features, guided by deep feature gradients in the image. The end-to-end differentiable nature of our formulation relaxes the need for manual feature tuning. We show in our experiments that not only do we achieve state-of-the-art performance on the training sensor suite (with median translation and rotation errors of 1.6cm and 0.07°), but our method also generalizes to unseen sensor models and environments — with results demonstrated using a variety of common datasets.

2 Related Works

Classical solutions to the problem of target-free camera/LiDAR extrinsic calibration form two categories: The first category exploits correlations between the image intensity value and the LiDAR reflectance value [1, 2, 3]. These methods perform well under assumptions of uniform lighting in the environment but still require an initial guess which is close to the ground truth extrinsic parameters. The other category of methods maps geometric (e.g. depth or normal) discontinuities in the LiDAR scan to image intensity discontinuities i.e. image gradients [4, 5, 6]. This use of local image gradient information can be augmented by local intensity normalization, which improves the robustness of these methods in scenes with varying illumination sources. Still, their performance in cluttered environments is limited by the abundance of local minima. Both these categories of classical methods typically require manual parameter tuning for each unique sensor suite and as such struggle to be adapted to different configurations and manufacturers.

More recent works have tackled the target-free calibration problem using deep-learned approaches. We classify these works into two categories: i) regression-based methods [7, 8] that align camera and LiDAR features using parameters regressed by a deep neural network. While these methods have demonstrated impressive accuracy on held-out datasets of their training environment, their regression-based nature makes a zero-shot transfer to other datasets difficult since the neural network is biased to output extrinsic parameters in the distribution seen during training. Another, newer category of methods employs differentiable optimization for either (ii) indirect feature point matching via non-linear perspective-n-point [9]; or (iii) calibration-flow refinement [10] to align LiDAR-extracted deep features to their corresponding pixel locations in the image. While both these methods have been shown to generalize better than regression-based methods, [10] has only shown results over a narrower region of operation and [9] achieves significantly less accurate results.

RGKCNet [9] and DXQNet [10] are of a different category of differentiable pose alignment compared to our method, in that they are forms of indirect alignment. RGKCNet learns 3-D point to 2-D pixel correspondences which incorrectly models visually salient features such as lamp posts and thin trees, which are line constraints and not point constraints. DXQNet learns a 2-dimensional weighting term which is used to weight the per-point alignment along XY-axis aligned directions. This puts the burden on the network to not just learn features but also to accurately model their orientations. Our method overcomes this challenge by explicitly computing deep image gradients, naturally dis-

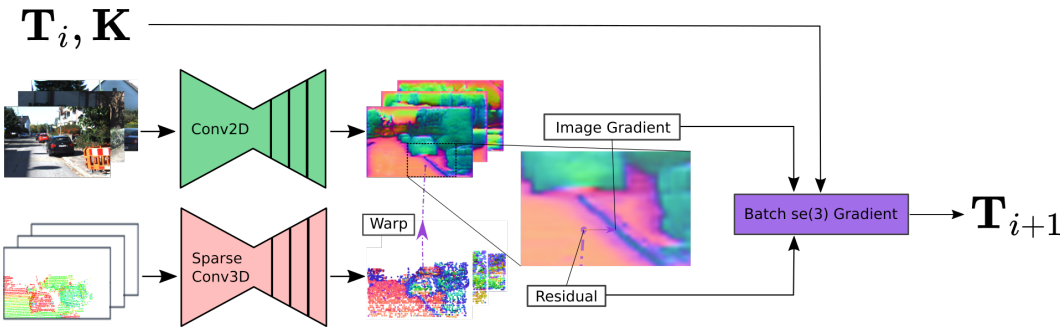


Figure 2: An overview of one iteration of the batched differentiable alignment. Batches of pairs of images and voxelized point clouds are passed through their respective U-Net feature extractors. Each sparse voxel feature is warped to the image using the latest transform parameters and the camera intrinsic parameters and registers a residual for its difference to the image feature at the corresponding pixel location. This residual along with the image gradient and the Jacobian of the projected pixel location with respect to the transform parameters forms the signal for the optimization.

tinguishing between line and point constraints — allowing the network to focus on learning useful features.

3 Method

We take inspiration from methods for scene-agnostic visual localization. Like PixLoc [11], we avoid overfitting pose estimation into a scene memorization problem by performing deep feature alignment using differentiable optimization. This allows us to decouple the effects of the geometry of pose refinement from the task of learning features. In doing so, we translate the task of achieving zero-shot transfer for camera/LiDAR extrinsic calibration, into the task of learning camera and LiDAR features that transfer to unseen environments, captured by different sensors.

3.1 Problem formulation

Given a point cloud, a set of N 3-D points, $\mathbf{P}_L \in \mathbb{R}^{3\times N}$ represented in the LIDAR reference frame (denoted by the subscript L) and a set of images, we want to determine the extrinsic parameters $\mathbf{R}_L^C \in \mathbf{SO}(3), \mathbf{t}_L^C \in \mathbb{R}^3$, such that,

$$\mathbf{P}_C = \mathbf{R}_L^C \mathbf{P}_L + \mathbf{t}_L^C. \tag{1}$$

Where $\mathbf{P}_C$ are the corresponding coordinates of $\mathbf{P}_L$, from the camera's reference frame. For brevity, we use $(\mathbf{R}_L^C, \mathbf{t}_L^C) = \mathbf{T}_L^C \in \mathbf{SE}(3)$ to denote the rigid body transform and omit the reference frame scripts i.e. $\mathbf{T} \triangleq \mathbf{T}_L^C$.

3.2 Camera/LiDAR extrinsic calibration as differentiable direct alignment

In direct alignment, the transform parameters are optimized to minimize the misalignment in appearance between signals registered by different sensors. In our case, our two sensors are the LiDAR and the camera. Instead of trying to align the raw signals from the sensors e.g. LiDAR intensity to image intensity which has been shown to fail in challenging lighting conditions [3], we first extract deep features from the raw inputs.

From the LiDAR side, we map the 3-D points using the initial guess of the extrinsic parameters to get $\hat{\mathbf{P}} = \hat{\mathbf{R}}\mathbf{P}_L + \hat{\mathbf{t}}$. We then voxelize these points and extract, using a sparse 3-D CNN, a set $(\hat{\mathbf{P}}^p, \mathbf{F}_L^p, \mathbf{w}_L^p)$ for each $p \in [1, ..., P]$ of a P-level multi-scale feature pyramid. At each level, $\hat{\mathbf{P}}^p \in \mathbb{R}^{3\times N_p}$ are the N_p voxel centroids, and $\mathbf{F}_L^p \in \mathbb{R}^{D_p \times N_p}$ are the N_p corresponding deep feature vectors at each voxel centroid of dimension D_p. Lastly, $\mathbf{w}_L^p$ is the vector of N_p learned weights with elements in the range $[0, 1]$.

From the camera side, we use a 2-D CNN to extract from the camera image $\mathbf{I} \in \mathbb{R}^{W \times H \times 3}$ a pyramid of deep features $\mathbf{F}_C^p \in \mathbb{R}^{W_p \times H_p \times D_p}$, and corresponding weights $\mathbf{W}_C^p \in \mathbb{R}^{W_p \times H_p}$. Similarly, $p \in [1, ..., P]$ stands for one of the pyramid levels, each matching in scale with their corresponding LiDAR extractor pyramid levels.

At each i-th iteration of the optimization, the misalignment between the deep features extracted from the j-th LiDAR point and camera features at its corresponding projected point is given by,

$$\mathbf{r}_j^p = (\mathbf{F}_{L_j}^p - \mathbf{F}_C^p[\Pi^p(\mathbf{R}_i \hat{\mathbf{P}}_j^p + \mathbf{t}_i)]) \in \mathbb{R}^{D_p}. \tag{2}$$

Π^p is the projection function of a 3-D point in the camera reference frame onto the image plane at level p, and $[\cdot]$ denotes sub-pixel lookup by bi-linear interpolation. We've also introduced new variables $\mathbf{R}_i$ and $\mathbf{t}_i$ which are optimization parameters that we iterate and initialize as $\mathbf{R}_0 = \mathbf{I}^{3\times3}$ and $\mathbf{t}_0 = \mathbf{0}$.

Similar to [11], we formulate a total cost function from these residuals in the form,

$$E_p(\mathbf{R}_i, \mathbf{t}_i) = \sum_j^{N_p} w_{l_j} w_{c_j} \rho(\|\mathbf{r}_j^p\|^2), \tag{3}$$

where w_{l_j} is the j-th element of $\mathbf{w}_L^p$, the LiDAR-learned weighting, and w_{c_j} is the camera-learned weighting sub-pixel interpolated at the pixel location of the j-th LiDAR point projected into the image plane at level p. Lastly, ρ is the learnable robust cost function [12]. To robustly minimize this non-linear least-squares problem we perform the alignment in a coarse-to-fine fashion, using solutions from the previous coarser level of the pyramid as the starting point in the problem of the finer level. At each level, we use the learned Levenberg-Marquardt algorithm [11], parameterizing transform updates using $\xi \in \mathfrak{se}(3)$. As such, we stack all the residual terms at level p into $\mathbf{r}^p \in \mathbb{R}^{D_p N_p}$ and formulate the Jacobian $\mathbf{J} \in \mathbb{R}^{D_p N_p \times 6}$ and Hessian $\mathbf{H} \in \mathbb{R}^{6\times6}$ with respect to ξ as,

$$\mathbf{J}_{k+(j-1)D_p} = \frac{\partial \mathbf{r}_{j_k}^p}{\partial \hat{\mathbf{p}}} \frac{\partial \hat{\mathbf{p}}}{\partial \xi}, \text{ and } \mathbf{H} = \mathbf{J}^\intercal \mathbf{W} \mathbf{J}, \tag{4}$$

where $j \in [1, N_p]$ is the j-th 3-D point, $k \in [1, D_p]$ denotes the k-th dimension of the deep-learned features at level p, and $\mathbf{W} \in \mathbb{R}^{D_p N_p}$ is a diagonal weighting matrix with N_p diagonals of size D_p each, where the j-th diagonal has the uniform weights $w_{l_j} w_{c_j}$. With these defined, we compute a gradient step in the direction of decreasing cost e.g. a Gauss-Newton step with $\xi = -\mathbf{H}^{-1}\mathbf{J}^\intercal \mathbf{W}\mathbf{r}$, and update our transform estimate using,

$$\begin{bmatrix} \mathbf{R}_{i+1} & \mathbf{t}_{i+1} \\ \mathbf{0} & 1 \end{bmatrix} = \exp(\widehat{\xi})^\intercal \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5}$$

We visualize one of these steps in Fig. 2

The training loss: During training, we perform these alignment updates for fixed M steps at each pyramid level, yielding for each level, $\mathbf{T}_M^p$. We supervise these transforms with the ground truth using the reprojection error of each 3-D point:

$$\mathcal{L} = \sum_p \sum_j \rho(\|\Pi^p(\bar{\mathbf{R}}\hat{\mathbf{P}}_j^p + \bar{\mathbf{t}}) - \Pi^p(\mathbf{R}_M^p \hat{\mathbf{P}}_j^p + \mathbf{t}_M^p)\|^2). \tag{6}$$

Note that since the points $\hat{\mathbf{P}}_j^p$ are LiDAR points projected by the initial guess of the extrinsic parameters, our supervision signal $\bar{\mathbf{T}}$ is given by, $\bar{\mathbf{T}} = \mathbf{T}\hat{\mathbf{T}}^{-1}$, where $\mathbf{T}$ is the ground truth extrinsic parameters and $\hat{\mathbf{T}}$ is the initial guess applied to the LiDAR points before voxelization.

3.3 Batch SE(3) alignment

At test time: When we make the added assumption that we are solving for the same transform parameters across a batch of image/point cloud pairs at test time, the change to our algorithm is

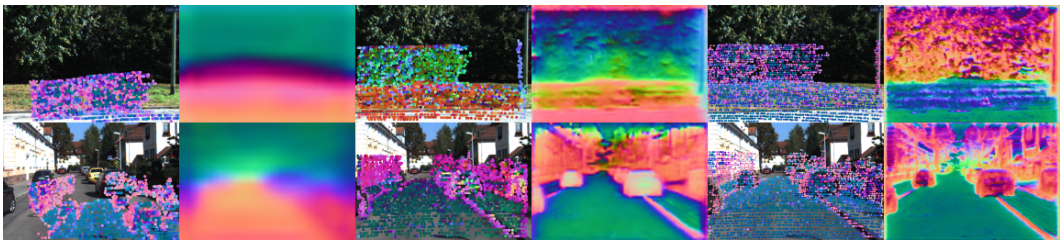


Figure 3: PCA feature visualization of a feature-deprived scene (top) and a feature-rich scene (bottom). During training, our batch formulation keeps the harder example at the top from diverging. Consequently, we can still learn the sparse amount of features it does have e.g. the ground to hedge differences in the middle columns, and the salient post on the rightmost column.

rather straightforward. Where we used to have, at each pyramid level, the residual $\mathbf{r}^p \in \mathbb{R}^{D_p N_p}$, we now stack all the residuals across the batch together and get the vector $\mathbf{r}_B^p \in \mathbb{R}^{D_p N_B}$ where $N_B = \sum_b^B N_b^p$ is the total number of 3-D points in the batch at level p. Then, we update Eq. (4) accordingly and solve the optimization steps just as we did in the single sample case. This technique has been deployed in most existing non-learning-based target-free calibration works [4, 3, 2], where it has been interpreted as making the calibration cost function Eq. (3) smoother and more convex. To our surprise, this simple yet very effective scheme has so far not been deployed in deep-learned camera-LiDAR extrinsic calibration.

At training time While at first glance it may seem impossible to perform batch pose alignment during training since we might encounter camera/LiDAR pairs with heterogeneous extrinsic parameters, note that the pose alignment is performed relative to the initial guess (Eq. (5)). So while the actual camera/LiDAR relative transform may differ across the batch, we can independently pick initial guesses for each sample in the batch such that the relative transform from the initial guess to the ground truth is identical. Mathematically, this gives for each b-th sample in the batch:

$$\hat{\mathbf{T}}_b = \Delta\mathbf{T}^{-1}\mathbf{T}_b. \tag{7}$$

Initializing the initial guess of each sample using Eq. (7) lets us jointly solve direct alignment across samples of the entire batch, allowing us even learn features from hard examples where individually the optimization would have diverged (see Fig. 3).

4 Training Setup

To test the capability of our framework to perform accurate zero-shot transfer to unseen environments, we've set up our experiments to train solely on one dataset, and subsequently evaluate performances using other datasets with different sensors and test environments.

Dataset: Due to its popularity as a benchmark for learning-based camera/LiDAR calibration, w use the KITTI Odometry dataset [13] as our sole training dataset. It consists of 22 sequences of suburban driving scenarios. During training, we only use camera "2" which is a front-facing colored perspective camera and the top-mounted Velodyne HDL-64E LiDAR. Of the 22 sequences, we use sequences "01" – "21", leaving sequence "00" out for validation and testing.

Setup for LiDAR and camera input: Different LiDARs exhibit different spatial coverage, intensity profiles, and reference coordinate frames. To make our framework robust to these variations during zero-shot transfer, we perform pose and intensity augmentations to the LiDAR point cloud and crop augmentations to the camera image. We specify these details in Appendix A.1.

Model: To facilitate robust calibration from large initial offsets, we use a coarse-to-fine alignment scheme with 3 levels. Two U-Net [14] architectures extract deep features from Lidar and camera separately for each of these levels. To aid the correspondence of similar features from different domains, we further adapt the features from each domain with a single multi-layer perceptron. We

Table 1: Results on the **same camera** of a held-out sequence of the KITTI Odometry dataset

Initial error	Method	Mean/Median Δt (cm)			Mean/Median ΔR (°)		
		x	y	z	roll	pitch	yaw
±1.5m ±20°	LCCNet	**0.24/0.26**	**0.38/0.36**	**0.46/0.35**	**0.03**/0.03	**0.01/0.00**	**0.04/0.02**
	Ours (1)	8.77/1.76	5.50/1.45	9.25/1.80	0.36/0.08	0.43/0.07	0.53/0.07
	Ours (8)	2.26/0.51	2.02/0.87	1.24/0.58	0.10/**0.02**	0.21/0.04	0.16/0.03
±0.5m ±10°	DXQNet	0.75/0.53	**0.48/0.51**	1.09/0.78	0.05/0.03	0.05/**0.03**	0.03/**0.02**
	Ours (1)	3.30/1.00	2.61/1.04	3.50/1.17	0.09/0.05	0.13/0.05	0.16/0.05
	Ours (8)	**0.40/0.30**	0.81/0.81	**0.59/0.47**	**0.02/0.02**	**0.04**/0.04	**0.02/0.02**

provide details about our model and its weight initialization in Appendix A.2. During training, use the Adam [15] optimizer with a learning rate of 10^{-5} to train for 20 epochs with a batch size of 8.

5 Results

Our objective is to evaluate the capacity of our algorithm for zero-shot transferability across various sensor setups and environments. All results showcased here are derived from models trained on a single camera/LiDAR pair as detailed in Section 4. We incrementally assess more challenging scenarios, beginning with a dataset very similar to the training set and culminating in tests on a completely different camera, LiDAR, and environment.

Metrics We assess our system's accuracy, initialized with random guesses around the true transformation, across varied sensors, and environments. We separately report calibration performance for rotation and translation. The mean/median absolute values represent translation error components, while mean/median Euler-angle components represent rotation error.

5.1 Extrinsic calibration in settings similar to training

Testing using the same camera: In this simple setting, we use images from the training camera (camera "02") but from a held-out sequence "00" of the KITTI Odometry dataset which comprises 4541 image/point cloud samples. We've found that the differences between the extrinsic parameters on this sequence and on the training set are negligible. So the key distinction from the training data is the novel environment scenes in this held-out sequence.

We compared against the regression-based LCCNet [8] and the differentiable calibration flow method DXQNet [10]. When comparing against LCCNet, we pass to our model initial extrinsic parameter guesses sampled uniformly [0, 20]°and [0, 1.5]m around the ground truth value, using the scheme presented in Appendix A.1. Note that the initial angular errors used in this experiment are larger than the values we used during training.

The upper section of Table 1 shows LCCNet outperforming our method. Our method performs poorly on single image/point cloud pairs due to scenes with insufficient data for full 6-DoF pose observability. However, our method substantially improves when run in batch optimization using 8 pairs, achieving sub-centimeter median absolute error on each translation axis and even surpassing LCCNet in median roll rotation accuracy. Remarkably, this was achieved despite the fact that our model never encountered rotation perturbations up to 20°during training.

The lower section of Table 1 shows that our method's single-sample performance is slightly worse than DXQNet in median metrics and notably worse in mean metrics since our method diverges when a single scene lacks sufficient structure. However, performing batch optimization significantly enhances our method, achieving lower error than DXQNet along all axes except for Y-axis translation and median roll-angle error.

Table 1 highlights that our method's weakest performance when running batched optimization is along the Y-axis of translation and pitch rotation. This is consistent with the sparsity of the LiDAR

Table 2: The performance change when trained on a camera "2" and tested on a **different** camera in the held-out test sequence "00" of KITTI Odometry

Initial error	Method	Mean Δt (cm)		Mean ΔR (°)		Median Δt (cm)		Median ΔR (°)	
		seen	unseen	seen	unseen	seen	unseen	seen	unseen
±1.5m ±20°	LCCNet	**1.59**	52.5	0.16	1.54	**1.01**	52.5	0.12	1.47
	Ours (1)	15.98	20.15	0.92	1.07	3.60	4.65	0.16	0.20
	Ours (8)	3.09	**3.77**	**0.15**	**0.30**	1.39	**1.69**	**0.07**	**0.07**
±0.5m ±10°	DXQNet	1.43	2.94	0.08	0.16	**0.81**	2.28	0.07	0.13
	Ours (1)	6.43	8.09	0.26	0.33	2.28	2.88	0.10	0.13
	Ours (8)	**1.19**	**1.72**	**0.06**	**0.07**	1.13	**1.58**	**0.06**	**0.07**

scans, which is sparsest along the Y-axis of the camera — making it hard to pinpoint transform parameters along this axis.

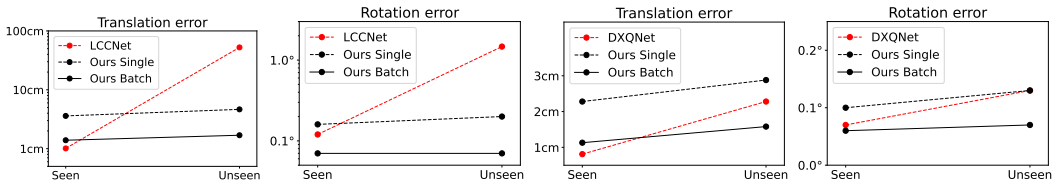
Testing using a different camera at a different vantage point: While all the methods were trained on camera "2" (seen), in this test, we perform calibration between the LiDAR and camera "3" (unseen). In the context of testing for generalization, this is significant because cameras "3" and "2" are separated roughly 50 cm apart, while the variations in the placement of camera "2" from sequence to sequence are negligible — as the sensor configuration was not changed.

Unlike the case of testing on the *seen* camera, the upper section of Table 2 shows that, when tested on the *unseen* camera, our method consistently performs better than LCCNet — whose mean and median translation error magnitudes (of more than 50cm) are more than a magnitude higher than the 1cm error achieved with the *seen* camera. While our method also experiences a drop in accuracy when tested on the *unseen* camera, our drop is significantly lower (from 3.60cm to 4.65cm median translation error), and even lower when run in batch optimization, from 1.39cm to 1.69cm median translation error magnitude – a 3mm drop.

Meanwhile, DXQNet, the learned calibration flow method, was more robust when transferred to the *unseen* camera. Seen on the lower section of Table 2, the median translation accuracy of DXQNet drops to 2.28cm on the *unseen* camera, which is still relatively accurate when compared to LCCNet.

While DXQNet performed better than our method in the single image/point cloud pair setting on the *seen* camera, once transferred to the *unseen* camera, the performance gap narrows down. In fact, shown on the lower section of Table 2, our method and DXQNet achieve the same level of median rotation accuracy.

Across both initial error settings in this experiment, we noticed that running batched optimization with our method achieved the best mean/median rotation and translation accuracy when transferred to the *unseen* camera. Additionally, the drop in accuracy (transferring from the *seen* to the *unseen* camera) is lower using our method – especially so in the batched alignment case. We highlight these facts in Fig. 4, where the slope of the graphs highlights the drop in accuracy.



(a) (b) (c) (d)

Figure 4: Slope graphs highlighting the error increase when the methods are tested on an unseen camera, having trained on the seen camera. All plots show median metrics: (a) and (b) comparing translation and rotation errors of our method versus LCCNet, and (c) and (d) comparing against DXQNet. Note that our method exhibits gentler slopes compared to the other methods, showing more robust transfer to the unseen camera.

Table 3: Zero-shot transfer performance on different datasets

Method	Δt (cm)	ΔR (°)
DXQNet	5.65/4.70	2.89/1.03
Ours Single	27.7/6.07	1.45/0.55
Ours Batch	**3.70/3.71**	**0.51/0.51**

(a) KITTI-360 dataset. Initial error in the range: $\pm 10° \pm 0.5$m. Ours Batch uses a batch size of 8

Method	Δt (cm)	ΔR (°)
LCCNet	324/318	20.8/18.1
Ours Single	102/16.8	3.60/0.64
Ours Batch	**6.97/3.87**	**0.44/0.43**

(b) Waymo dataset. Initial error in the range: $\pm 20° \pm 1.5$m. Ours Batch uses a batch size of 4

5.2 Zero-shot transfer to different environments with different sensors

Transfer to a different camera in a different environment: The KITTI-360 [16] dataset was captured in Karlruhe, just like the KITTI Odometry dataset used during training. However, the camera setup is different both in intrinsic parameters and its relative transform to the LiDAR.

We compare our calibration accuracy to DXQNet, as they have also reported the zero-shot transfer metrics. In Table 3a, we show that overall, the accuracy achieved by both methods is worse in rotation and in translation compared to their respective performances on the KITTI Odometry dataset. While our method performs better than DXQNet in rotation but worse in translation when run on a single sample; when optimized over a batch of image/point cloud pairs, our method performs significantly better.

Transfer to a different camera, LiDAR, and environment: The Waymo dataset [17], which has a higher resolution camera and a custom LiDAR, is another good candidate for testing generalizability. To match the training data image resolution, we halve the image and camera intrinsics, allowing for consistent feature extraction. This scaling feature, which updates the camera model, further distinguishes optimization-based calibration methods over regression-based ones which can't explicitly reconfigure the camera projection parameters.

In Table 3b, we see that LCCNet does not generalize, with translation errors over 3m and rotation errors at 20.8°(mean) and 18.1°(median). Our method, run on a single sample, is poor in the mean metric (just above 1m and 3.6°), but is better in the median metric (16.8cm, 0.64°), suggesting some outlier scenes impact calibration performance. Notably, batch processing image/point cloud pairs improves our performance significantly, reducing the median translation error to 3.87cm and rotation errors to 0.44°(mean) and 0.43°(median).

6 Limitations

Currently, our model assumes shared visibility between the LiDAR and the camera, enabling pose alignment from simultaneous image and point cloud pairs. In sensor settings without shared visibility, existing literature resolves this by creating a local map from several images and LiDAR scans [18]. Our model presumes simultaneous image pixel and LiDAR point registration, necessitating ego-motion compensation for rotating LiDAR models. In future work, we aim to tackle ego-motion estimation jointly with LiDAR/camera extrinsic calibration.

7 Conclusion

Our aim is to deploy robots in various environments and to have them be able to accurately perform tasks; calibration is just one of many prerequisites. Although our framework demonstrates promising results in diverse, real-world settings, we still fall short of the accuracy provided by traditional target-based calibration methods. We recognize that not every scene in the wild has sufficient information to observe all 6-DoF extrinsic parameters. With this, we are excited to build on our work in geometric priors for differentiable optimization, to tackle challenges like task-dependent calibration and mapping — where the necessary level of accuracy depends on the task required.

References

[1] N. Williams, K.-L. Low, C. Hantak, M. Pollefeys, and A. Lastra. Automatic image alignment for 3d environment modeling. pages 388– 395, 11 2004. ISBN 0-7695-2227-0. doi:10.1109/SIBGRA.2004.1352985.

[2] G. Pascoe, W. Maddern, and P. Newman. Direct visual localisation and calibration for road vehicles in changing city environments. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.

[3] G. Pandey, J. McBride, S. Savarese, and R. Eustice. Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information. *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 26, 01 2012. doi:10.1609/aaai.v26i1.8379.

[4] J. Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems*, 2013.

[5] A. Napier, P. Corke, and P. Newman. Cross-calibration of push-broom 2d lidars and cameras in natural scenes. In *2013 IEEE International Conference on Robotics and Automation*, pages 3679–3684, 2013. doi:10.1109/ICRA.2013.6631094.

[6] X. Liu, C. Yuan, and F. Zhang. Targetless extrinsic calibration of multiple small fov lidars and cameras using adaptive voxelization. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12, 2022. doi:10.1109/TIM.2022.3176889.

[7] G. Iyer, R. KarnikRam, K. M. Jatavallabhula, and K. M. Krishna. Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1110–1117, 2018.

[8] X. Lv, B. Wang, D. Ye, and S. Wang. Lidar and camera self-calibration using cost volume network. *arXiv preprint arXiv:2012.13901*, 2020.

[9] C. Ye, H. Pan, and H. Gao. Keypoint-based lidar-camera online calibration with robust geometric network. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2022. doi:10.1109/TIM.2021.3129882.

[10] X. Jing, X. Ding, R. Xiong, H. Deng, and Y. Wang. Dxq-net: Differentiable lidar-camera extrinsic calibration using quality-aware flow, 2022.

[11] P.-E. Sarlin, A. Unagar, M. Larsson, H. Germain, C. Toft, V. Larsson, M. Pollefeys, V. Lepetit, L. Hammarstrand, F. Kahl, and T. Sattler. Back to the Feature: Learning Robust Camera Localization from Pixels to Pose. In *CVPR*, 2021.

[12] J. T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019.

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[16] Y. Liao, J. Xie, and A. Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d, 2022.

[17] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[18] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman. Choosing a time and place for calibration of lidar-camera systems. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4349–4356, 2016.

[19] M. E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, apr 1959. ISSN 0001-0782. doi:10.1145/377939.377946. URL https://doi.org/10.1145/377939.377946.

[20] B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] S. Contributors. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[23] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions, 2018.

[24] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia. Spherical transformer for lidar-based 3d recognition, 2023.

[25] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.