Targetless Lidar-camera registration using patch-wise mutual information

Matthias Hermann, Dennis Grießer, Bernhard Gundel,
Daniel Dold, Georg Umlauf, Matthias O. Franz
Institute for Optical Systems
HTWG Konstanz

***Abstract*—Targetless Lidar-camera registration is a repeating task in many computer vision and robotics applications and requires computing the extrinsic pose of a point cloud with respect to a camera or vice-versa. Existing methods based on learning or optimization lack either generalization capabilities or accuracy. Here, we propose a combination of pre-training and optimization using a neural network-based mutual information (MI) estimation technique (MINE [1]). This construction allows back-propagating the gradient to the calibration parameters and enables stochastic gradient descent. To ensure orthogonality constraints with respect to the rotation matrix we incorporate Lie-group techniques. Furthermore, instead of optimizing on entire images, we operate on local patches that are extracted from the temporally synchronized projected Lidar points and camera frames. Our experiments show that this technique not only improves over existing techniques in terms of accuracy, but also shows considerable generalization capabilities towards new Lidar-camera configurations.**

***Index Terms*—Lidar-camera registration, mutual information**

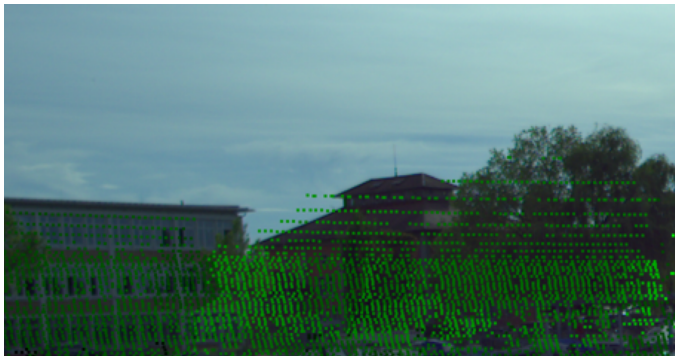
ACKNOWLEDGMENT

This research has been financed by BMBF (01IS19083A) and Baden-Württemberg Stiftung gGmbH.

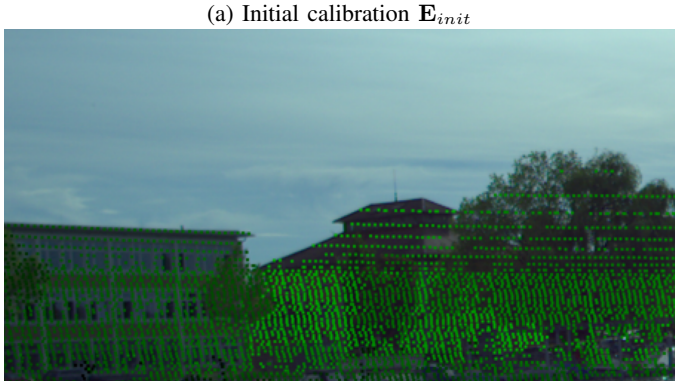
I. INTRODUCTION

For sensor fusion tasks, such as point cloud coloring, or acquiring depth information for downstream tasks (e.g., reconstruction or object recognition), it is essential to estimate the registration transformation between camera and Lidar coordinate systems. Such a transformation is denoted by $\mathbf{E}$ and consists of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$. Together, they describe the relative position of one sensor with respect to another. Compared to a manual process, automatic registration algorithms try to estimate the parameters $\mathbf{E}$ based on data automatically. In this work, we are addressing the targetless case [3], [4] which does not require a known pattern [5] such as a checkerboard. In the literature, often mutual information (MI) (e.g., [6]) has been applied [7], [8], as well as learning-based schemes [9]–[11], that learn the calibration, given a labeled training dataset. In that line of research, we propose a hybrid algorithm that uses a patch-based MI maximization scheme.

Our method is hybrid in the sense that it uses registered training data for pre-training, but involves a optimization scheme for unregistered test data. The method is based on projecting the Lidar point cloud onto the common image plane, and extracting local patches by differentiable sampling [12].



(a) Initial calibration $\mathbf{E}_{init}$



(b) LMI-optimized calibration $\mathbf{E}_{mi}$

Fig. 1: Registration results on the unseen maritime scenario. The model has been pre-trained on KITTI [2] indicating good generalization capabilities.

As a prerequisite for the common image plane, the intrinsic camera parameters, i.e., the camera matrix $\mathbf{P}$, needs to be available beforehand — extrinsic parameters can be chosen freely. The major advantage of such a local scheme is that general local shape patterns are more transferable to unseen scenarios. This allows the method to produce both accurate solutions and improved generalization compared to existing approaches, which makes the method suitable for situations where calibration is required (e.g., caused by weather conditions or mechanical changes), but no labeled training data are available. This is especially important for online or re-calibration scenarios.

We show the effectiveness of our method on the KITTI

dataset and test its generalization performance within a real world maritime scenario.[1]

II. Related work

There are two major approaches to targetless registration of camera and Lidar streams. The first approach is based on optimization and works by optimizing two unregistered streams with respect to some performance metrics, such as the alignment of detected and projected 3d edges with the corresponding 2d image edges [13]. Alternatively, measuring and optimizing the mutual information between the intensities of projected Lidar points and the corresponding image pixel intensities directly has been proposed [14]. Beside, techniques from robotics incorporate a visual-odometry pipeline for estimating the extrinsic parameters by matching odometry trajectories obtained from both sensors signals [15]. Recent methods based on deep learning architectures, e.g., RegNet [9], learn to regress the 6 parameters of the extrinsic calibration directly based on a labeled dataset consisting of examples with known calibration parameters. Here, CMRNet [16] extends RegNet with an additional refinement procedure that consists of multiple models trained on different scales in a coarse-to-fine procedure. LCCNet [11] is similar in architecture, but adds an additional correlation layer for feature matching and introduces a cost volume over matched features between Lidar and camera. The combination of feature matching and hierarchical aggregation improves overall accuracy. Recently proposed hybrid methods for the calibration task use available pixel-wise labels such as object boundaries and object affinity to register the camera and point cloud data [17]. However, even though the registration quality is considerable, pixel-wise annotations are expensive to obtain. Our approach is similar in that we also combine deep learning with an optimization procedure. However, our approach differs as we are using small patches from raw images instead of relying on labeling or pixel-wise features that are often not available in practical situations.

III. Method

The goal of our method is to compute the extrinsic calibration parameters

$$\mathbf{E} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{1}$$

of two non-registered but temporally synchronized Lidar and camera datasets with N frames. The raw RGB camera images are stored as $\mathbf{C}_i \in [0,1]^{3\times H\times W}$. The Lidar point clouds are matrices $\mathbf{L}_i \in \mathbb{R}^{4\times K_i}$ with K_i 3d points in homogeneous coordinates. Throughout the paper, we use i to reference a single frame ($\mathbf{L}_i$, $\mathbf{C}_i$) and use j to reference a single Lidar point. Our goal is to optimize mutual information $\mathcal{I}$ between patch ensembles extracted from Lidar and camera frames with respect to the calibration parameters $\mathbf{E}$. Optimizing globally is problematic, as the projected Lidar signal is very sparse. Moreover, transferable local image and shape features are underrepresented compared to the empty parts of the projections (c.f. Fig. 5), i.e., most of the pixels are empty. Therefore, we use a local approach and first project the Lidar points with the known projection matrix $\mathbf{P}$ onto the common image plane and then sample local patches from both sensors. Note, that for a common image plane at least the intrinsic camera parameters need to be available or obtained by camera calibration. The full method is a two-step procedure with an estimation, possibly offline, and an optimization step. In the estimation step (c.f. Sec. III-F) a neural network is pre-trained to estimate the mutual information between patch ensembles based on registered training data $\mathbf{X}_{train} = (\mathbf{L}_{train}, \mathbf{C}_{train}, \mathbf{E}_{true})$. In the optimization step (c.f. Sec. III-G) the mutual information between patch ensembles extracted from miscalibrated sensors is maximized with respect to the unknown calibration parameters $\mathbf{E}$ based on a unregistered test set $\mathbf{X}_{test} = (\mathbf{L}_{test}, \mathbf{C}_{test}, \mathbf{E}_{init})$. Fig. 2 illustrates the overall architecture.

A. Mutual information estimation

The mutual information $\mathcal{I}$ is a non-linear dependency measure based on shared information between two variables. In the given calibration scenario we use the measure to maximize the registration between projected Lidar points $\mathbf{B}$ and camera images $\mathbf{C}$ in the common image plane. However, instead of optimizing entire images and point clouds, we follow a local scheme and optimize the mutual information of extracted local patches $\mathbf{b}$ and $\mathbf{c}$ instead:

$$\mathcal{I}(\mathbf{b}, \mathbf{c}) = \mathcal{H}[\mathbf{b}] - \mathcal{H}[\mathbf{b}|\mathbf{c}]. \tag{2}$$

Existing MI estimation techniques rely either on binning [6] or k-nearest neighbors statistics [18]. Both techniques are unreliable in high dimensions [1]. Also, gradient optimization using these estimators is difficult, as the derivatives generally do not exist [14]. However, recently a new class of neural mutual information estimators (MINE) was proposed [1]. These techniques transform the estimation problem into a binary classification problem, which can be solved efficiently using deep neural networks. The network is trained to separate registered $(\mathbf{b}, \mathbf{c})$ from non-registered $(\mathbf{b}, \mathbf{c}^*)$ tuples of patches, and hereby implicitly estimates the mutual information between the two modalities. As neural networks are differentiable by design, gradient propagation with respect to the calibration $\mathbf{E}$ parameters is straightforward using backpropagation.

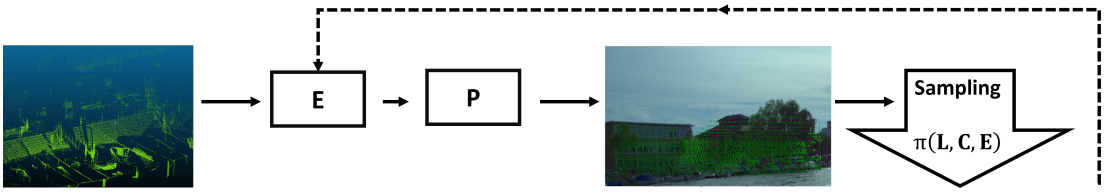
The needed training data $(\tilde{\mathbf{X}}, \mathbf{y})$ for the binary classification problem consists of two different M-sized sets of patch tuples

$$\tilde{\mathbf{X}} = \{\mathbb{J}, \mathbb{M}\}, \quad \mathbf{y} = \{\mathbf{0}, \mathbf{1}\}. \tag{3}$$

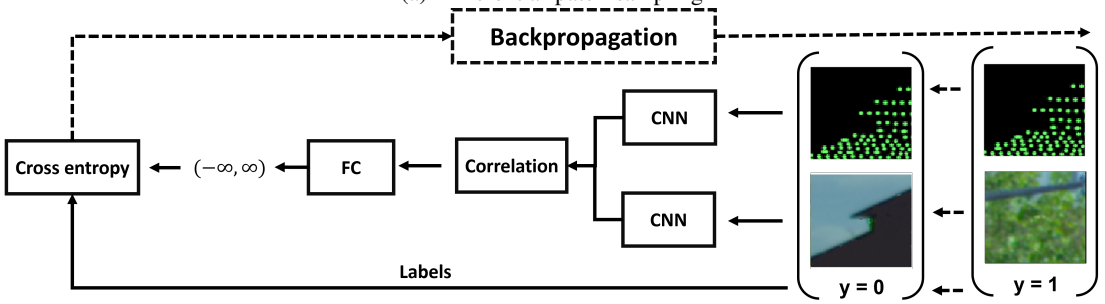
In our case the tuples $(\mathbf{b}, \mathbf{c}) \in \mathbb{J}$ are registered Lidar-camera patches, whereas the tuples $(\mathbf{b}, \mathbf{c}^*) \in \mathbb{M}$ are random and therefore non-registered Lidar-camera patches. The class indicator function

$$p(\tilde{\mathbf{x}} = 0) = f_\theta(\tilde{\mathbf{x}}) \tag{4}$$

[1]Our Python implementation: https://github.com/matherm/Patch-MI-registration (available upon acceptance).



(a) Differential patch sampling



(b) Mutual information maximization

Fig. 2: The Lidar point cloud gets transformed by $\mathbf{E}$ and projected by the known and fixed camera matrix $\mathbf{P}$. Afterwards image patches $\mathbf{c}_j$ are sampled centered around the Lidar projections $\mathbf{u}_j$. For estimating mutual information (MI), the patch tuples $(\mathbf{b}_i^j, \mathbf{c}_i^j)$ are organized as registered set $\mathbb{J}$ with label $y = 0$, and contrast set $\mathbb{M}$ with label $y = 1$ by randomly picking camera patches $(\mathbf{b}_i^j, \mathbf{c}_i^*)$. The gradient is back-propagated from the cross entropy loss to the extrinsic calibration matrix $\mathbf{E}$.

is parameterized by a neural network f_θ with trainable parameters θ. As optimization criterion, the well-known cross entropy

$$\arg\min_\theta CE(f_\theta(\tilde{\mathbf{X}}), \mathbf{y}) = \frac{1}{2M} \sum_i^{2M} \mathbf{y}_i \log \sigma(f_\theta(\tilde{\mathbf{x}}_i)) \quad (5)$$

is used, where $\sigma(\cdot)$ is the sigmoid function.

Formally, the relationship between the mutual information and the surrogate classification problem is given in terms of the Donsker-Varadhan representation (DV) [19] of the KL-divergence

$$\mathcal{I}(\mathbf{b}, \mathbf{c}) = KL(\mathbb{J}|\mathbb{M}) \geq \sup_f \mathbb{E}_\mathbb{J}[f] - \log(\mathbb{E}_\mathbb{M}[e^f]), \quad (6)$$

where f is the class of functions for which the expectation exists. MINE-like estimators [1] restrict the function class f to the class of neural network functions f_θ using the universal approximation theorem. Experiments showed that the numerically more stable cross entropy can be used as a loose lower-bound instead (e.g., [20]):

$$\mathcal{I}(\mathbf{b}, \mathbf{c}) \geq \mathcal{I}_{f_\theta}(\mathbf{b}, \mathbf{c}) = -CE(f_\theta(\tilde{\mathbf{X}}), \mathbf{y}) + \text{const.} \quad (7)$$

We describe the generation of the registered and non-registered sets of patches in Sec. III-C. The architecture of the neural network f_θ for classification is introduced in Sec. III-E.

B. Projection

We follow [2] and transform the point cloud $\mathbf{L}_i$ by the extrinsic transformation $\mathbf{E} \in \text{SE}(3)$. Here, $\mathbf{E}$ represents an element from the special euclidean group and is composed of a rotation matrix $\mathbf{R} \in \text{SO}(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$. Afterwards, the transformed point cloud is projected by the camera matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$. The projected coordinates are given by

$$\mathbf{U}_i \sim \hat{\mathbf{U}}_i = \mathbf{PEL}_i, \quad (8)$$

where $\mathbf{L}_i \in \mathbb{R}^{4\times K_i}$ is the 3d input point cloud in homogeneous coordinates, $\hat{\mathbf{U}}_i \in \mathbb{R}^{3\times K_i}$ is the projected 2d point cloud in homogeneous coordinates, $\mathbf{U}_i \in \mathbb{R}^{2\times K_i}$ are image coordinates normalized by dividing by the z-coordinate, and K_i is the total number of points in the point cloud. Next, we compute a depth image $\mathbf{B}_i \in \mathbb{R}^{1\times H\times W}$ of the point cloud with the same spatial size as the RGB camera image $\mathbf{C}_i \in \mathbb{R}^{3\times H\times W}$. $\mathbf{B}_i$ is rendered by setting the euclidean distance of the transformed Lidar point to the camera origin $d(\mathbf{EL}_i^j, \mathbf{0})$ as pixel intensity, given by

$$\mathbf{B}_i(x, y) = \begin{cases} d(\mathbf{EL}_i^j)/s & (x, y) \in \tilde{\mathbf{U}}_i \\ 0 & \text{otherwise} \end{cases}, \quad (9)$$

where $\tilde{\mathbf{U}}_\mathbf{i} \in \mathbb{Z}^{2\times K_i}$ is computed by rounding all values of $\mathbf{U}_i$ to the next integer pixel position, and $s = \text{std}(\{d(\mathbf{EL}_i^j)\}_j^{K_i})$ is the standard deviation of all euclidean distances of the transformed Lidar points in the i-th point cloud. Here, we use the euclidean distance instead of Lidar intensity values, because the needed shape features such as sharp edges are better retained and the distances are less noisy [21]. The scaling factor is needed, because we expect close and very distant Lidar points across different frames.

C. Differentiable patch sampling

One major difficulty in our approach is computing the gradient w.r.t. $\mathbf{E}$ as the patch sampling procedure is usally not differentiable. Hence, a differentiable patch sampling function $\pi(\cdot)$ is required for generating the $2M$ patch tuples

$$\tilde{\mathbf{X}} = \pi(\mathbf{L}, \mathbf{C}; \mathbf{E}) \tag{10}$$

with corresponding labels

$$\mathbf{y} = \{0^M, 1^M\} \tag{11}$$

for the surrogate classification problem. For compatibility with Eq. (7), we define the patch sample function $\pi(\mathbf{L}, \mathbf{C}; \mathbf{E})$ on the entire dataset. Next, we crop square patches with size $S \times S$ from the two images $\mathbf{B}_i$ and $\mathbf{C}_i$ (see green dots on black background in Fig. 5). To back-propagate the gradient of the neural network f_θ to the calibration parameters $\mathbf{E}$, the cropping procedure needs to be differentiable. To this end, we use a localized separable interpolation kernel [12]

$$\Phi(x) = \exp^{-(x-\frac{S}{2})^2}, \tag{12}$$

with offset S for centering the patches, and choose the projected Lidar points $\mathbf{U}_i = \{\mathbf{u}_i^j\}_{j=1}^{K_i}$ as patch centers (c.f. Eq. (12)). Note, that interpolation is mandatory at this stage as the projected Lidar points fall in-between pixels and hence the center-pixel and the pixels of the extracted patches are interpolated subpixels. The patch tuples $\{(\mathbf{b}_i^j, \mathbf{c}_i^j)\}_{j=1}^{K_i}$ corresponding to $\mathbf{U}_i$ are given by

$$\mathbf{b}_i^j(x,y) = \sum_{h=1}^{H}\sum_{w=1}^{W} \mathbf{B}_i(h,w)\ \Phi(\mathbf{u}_i^j(y)-h)\ \Phi(\mathbf{u}_i^j(x)-w) \tag{13}$$

and

$$\mathbf{c}_i^j(x,y) = \sum_{h=1}^{H}\sum_{w=1}^{W} \mathbf{C}_i(h,w)\ \Phi(\mathbf{u}_i^j(y)-h)\ \Phi(\mathbf{u}_i^j(x)-w). \tag{14}$$

Because the interpolation kernel is differentiable, the gradient can be back-propagated through the patch cropping procedure by applying the chain rule. Note, that the patch ensemble depends on the projected Lidar points $\mathbf{U}_i$ and hence on the current calibration configuration during optimization. However, as the ensemble is kept constant during the gradient update, differentiability of the optimization with respect to $\mathbf{E}$ is retained.

D. Sub-sampling

As the Lidar point clouds can contain thousands of points, there are also thousands of patch tuples which can be computationally prohibitive. To mitigate, we sub-sample a M-sized subset of the available K_i patch tuples. Therefore, we define a set of equidistant 2d grid points $\mathcal{G}$ with distance M_g in the $H \times W$ plane (see Fig. 3) and then select patch tuples that are nearest neighbors of the grid points. As a helpful side-effect, the 2d grid equalizes the sampling density of the projected points, as there are many spots that are hugely oversampled, such as walls or the floor. Unfortunately, patch pairs may be

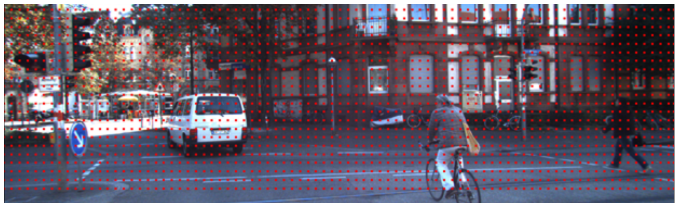


Fig. 3: Visualization of the uniform sampling grid $\mathcal{G}$ for sub-sampling the projected Lidar points. The distance M_g between the grid points is given in pixels.

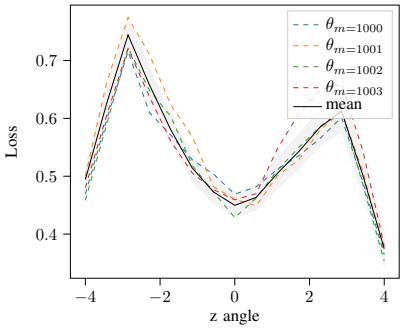


Fig. 4: Visualizing the loss function $-\mathcal{I}_{f_\theta}(\pi(\mathbf{L}, \mathbf{C}; \mathbf{E}_{dist}))$ for small Euler angle distortions of $\mathbf{E}_{dist}(0, 0, \epsilon, \mathbf{0})$, with $\epsilon \in [-4, 4]$. The dotted lines show the loss function at different iterations θ_m of gradient descent. Notice the wrong local minimum (yellow).

the nearest neighbor of multiple grid points, which causes interfering imbalances. To eliminate such duplicate patch pairs, we iterate over the K_i patch tuples and test for each tuple $(\mathbf{b}_i^j, \mathbf{c}_i^j)$, whether its corresponding projected Lidar point $\mathbf{u}_i^j$ is the nearest neighbor of a grid point $\mathcal{G}_m$. Formally, the set of registered patches is given by

$$\mathbb{J} \sim \widehat{\mathbb{J}} = \{(\mathbf{b}_i^j, \mathbf{c}_i^j) \mid \exists \mathcal{G}_m :\ j = \arg\min_k ||\mathbf{u}_i^k - \mathcal{G}_m||_{L_2}\}_{j=1}^{K_i}. \tag{15}$$

The equally sized set of randomized patch pairs

$$\mathbb{M} \sim \widehat{\mathbb{M}} = \{(\mathbf{b}_i^j, \mathbf{c}_i^*)\}_{j=1}^{K_i} \tag{16}$$

is generated by picking the camera patches $\mathbf{c}^*$ randomly. Finally, the differentiable patch sampling is given by

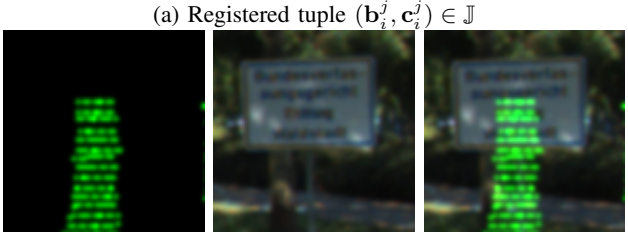
$$\pi(\mathbf{L}, \mathbf{C}; \mathbf{E}) = \{\mathbb{J}, \mathbb{M}\}. \tag{17}$$

Again, as the sampled set $\pi(\cdot)$ is kept constant during the gradient update, differentiability with respect to $\mathbf{E}$ is retained.

E. MINE architecture

The used modified MINE estimator requires a parametric function f_θ for classification and the subsequent mutual information estimation (c.f. Eq. (7)). Therefore, we define a convolutional neural network (CNN) f_θ with two input heads. One input head is for the Lidar patches $\mathbf{b}_j$, and a second for the camera patches $\mathbf{c}_j$. The schematic architecture of the neural network is shown in Fig. 2b. The *CNN* blocks consist

(a) Registered tuple $(\mathbf{b}_i^j, \mathbf{c}_i^j) \in \mathbb{J}$



(b) Random non-registered tuple $(\mathbf{b}_i^j, \mathbf{c}_i^*) \in \mathbb{M}$

Fig. 5: Example of registered and non-registered patch tuples. The left and middle columns show a patch from $\mathbf{B}$ and $\mathbf{C}$, respectively. For illustration purposes we also show the combined patches in the right column.

of a standard batch normalization layer, followed by six CNN layers with average pooling and ReLU-activation functions. We use 32 features in every CNN layer and a kernel size of five. The input layers differ for Lidar and image patches in the number of channels. The *Correlation* layer does not have parameters and simply correlates the inputs by flattening and computing the dot product between the 32 feature maps of the two input paths by computing

$$\text{Correlation}_z(\mathbf{f}_z, \mathbf{g}_z) = \langle \mathbf{f}_z \cdot \mathbf{g}_z \rangle \tag{18}$$

per feature map, where $\mathbf{f}_z \in \mathbb{R}^{uv}$ is the z^{th} feature map of the Lidar path, $\mathbf{g}_z \in \mathbb{R}^{uv}$ is the z^{th} feature map of the camera path, and uv is the dimension of the flattened feature map f_z. The *FC* block consists of two fully connected layers with ReLU-activation and also 32 features. The entire network has $259\,205$ parameters.

F. Estimating $\mathcal{I}(Lidar, Camera)$

The pre-training step is required to maximize the lower bound of the mutual information, such that it can be used as optimization criterion in the registration phase. The pre-training is done by using the registered training examples and optimizing the trainable parameters θ of the CNN f_θ as defined in Eq. 7. Plugging it all together results in the optimization problem

$$\theta^* = \arg\max_\theta \mathcal{I}_{f_\theta}(\pi(\mathbf{L}_{train}, \mathbf{C}_{train}; \mathbf{E}_{true}), \mathbf{y}), \tag{19}$$

where $\pi(\cdot)$ is the patch sampling function that transforms an input pair into tuples of aligned and non-aligned local patches (c.f. Sec. III-A). The resulting θ^* is used for initializing the second step.

G. Optimizing $\mathbf{E}$

The second step is the actual registration of an unregistered test set $\mathbf{X}_{test} = (\mathbf{L}_{test}, \mathbf{C}_{test}, \mathbf{E}_{init})$ with N input frames with respect to the extrinsic calibration parameters $\mathbf{E}_{Mi}$. The initialization is given by $\mathbf{E} = \mathbf{E}_{init}$. Because of the pre-trained $\mathcal{I}_{f_\theta}$ estimator, the propagated gradients are stable and can be directly used for improving the initial estimate of $\mathbf{E}$

$$\mathbf{E}_{mi} = \arg\max_{\mathbf{E}} \mathcal{I}_{f_\theta^*}(\pi(\mathbf{L}_{test}, \mathbf{C}_{test}; \mathbf{E})). \tag{20}$$

During experiments we noticed, that the optimization sometimes gets stuck in a local minimum near the global optimum. Therefore, we add the CNN parameters θ to the optimization

$$\mathbf{E}_{mi}, \theta' = \arg\max_{\mathbf{E},\theta} \mathcal{I}_{f_\theta}(\pi(\mathbf{L}_{test}, \mathbf{C}_{test}; \mathbf{E})) \tag{21}$$

once it reaches its first plateau. We show the loss function for a single rotation angle at different iteration steps θ_m in (c.f. Fig. 4).

H. Gradient ascent implementation

We do not optimize in full-batch, but use a single frame at a time, i.e., stochastic gradient descent (SGD) [22]. This means we compute the back-propagated gradients

$$\nabla_{\mathbf{E}} \mathcal{I} = \nabla_{\mathbf{E}} \mathcal{I}_{f_{\theta^*_{\hat{w}}}}(\pi(\mathbf{L}_{test}^i, \mathbf{C}_{test}^i; \mathbf{E})), \tag{22}$$

with

$$i \sim \mathcal{U}(0, N) \tag{23}$$

based on a single randomly sampled pair $(\mathbf{L}_i, \mathbf{C}_i)$ instead of the full data set. Next we split the parametrization of $\mathbf{E}$ into their affine components $\mathbf{R}$ and $\mathbf{t}$, and treat the components separately. While optimizing $\mathbf{t}$ is trivial, optimizing the rotation matrix $\mathbf{R}$ is non-trivial as it is constrained to be orthonormal. We optimize inside the Lie group $\mathrm{SO}(3)$ and use the *gradient flow* method as proposed in [23] for computing the derivatives. Note, that the back-propagated gradient of the loss function $\nabla_{\mathbf{R}} \mathcal{I}$ is a skew-symmetric matrix $\mathbf{\Theta}$, that represents an infinitesimal rotation and hence an element of the corresponding Lie algebra $\mathfrak{so}(3)$. Every skew-symmetric matrix $\mathbf{\Theta}$ can be uniquely parameterized by a 3d vector $\mathbf{r}$ giving rise to a vector space. Further, every skew-symmetric matrix $\mathbf{\Theta}$ can be related to an orthogonal matrix $\mathbf{R}$ by

$$\mathbf{R} = \exp(\mathbf{\Theta}), \tag{24}$$

where $\exp(\cdot)$ is the matrix exponential. In order to compute a valid gradient step beyond the neighborhood of $\mathbf{R}$, the gradient direction needs to be expressed by the Lie bracket [23]. The relation between the two gradient expressions is given by the commutator

$$\nabla_{\mathbf{\Theta}} \mathcal{I} = (\nabla_{\mathbf{R}} \mathcal{I})^T \mathbf{R} - \mathbf{R}^T (\nabla_{\mathbf{R}} \mathcal{I}). \tag{25}$$

From there, we compute the corresponding parameter vector $\mathbf{r}$ by taking the upper triangular matrix of $\nabla_{\mathbf{\Theta}} \mathcal{I}$. The *gradient flow* update rule is then given by

$$\mathbf{R}_{m+1}^T = \exp(\eta\, \mathbf{\Theta}_{\mathbf{r}_m})\, \mathbf{R}_m^T, \tag{26}$$

with step size η and the skew-symmetric matrix $\mathbf{\Theta}_{\mathbf{r}_m}$ parameterized by $\mathbf{r}_m$ at the m-th iteration step. Unfortunately, rotation matrices above two dimensions are not commutative. Hence we cannot make additive steps of ascent in $\mathfrak{so}(3)$ and need to map between the Lie algebra and the manifold in every iteration by computing the matrix exponential in Eq. (24).

IV. Experiments

For the experiments, we use the following techniques as baselines: LCCNet [11], RegNet [9], and CMRNet [16], which are neural network methods.[2] We also compare classic methods based on normalized mutual information and gradient information (gNMI [14])[3], respectively particle swarm optimization (pNMI [21]). For measuring performance, we follow [16] and also measure the translation error

$$\mathbf{t}_\Delta(\mathbf{t}_{true}, \mathbf{t}_{est}) = ||\mathbf{t}_{true} - \mathbf{t}_{est}||_{L2} \tag{27}$$

and the rotation error by using the quaternion angle

$$\mathbf{R}_\Delta(\mathbf{Q}_{true}, \mathbf{Q}_{est}) = 2 * \text{atan2}(|| \mathbf{Q}_\Delta^{Im} ||_{L2}, \mathbf{Q}_\Delta^{Re}), \tag{28}$$

with

$$\mathbf{Q}_\Delta = \mathbf{Q}_{\mathbf{R}_{true}} * \mathbf{Q}_{\mathbf{R}_{est}}^{-1}, \tag{29}$$

during the optimization procedure. The last formula first computes the difference rotation $\mathbf{Q}_\Delta$, and then measures the shortest angle to identity. atan2 is the standard 2-argument arctangent function.

A. Pre-training

For pre-training we use the KITTI dataset [2], which consists of 20 sequences with varying length. In total there are 7125 frames available. The images are stored as 375×1242 RGB. The Lidar scans are 360° Velodyne-64 scans. We take the first 10 sequences as training data and pre-train the mutual information estimator $\mathcal{I}_\theta(\mathbf{A}, \mathbf{B})$ until convergence by using left-out 10% validation data and ADAM optimizer with $\eta = 10^{-3}$ for all trainable parameters. We use $M_g = 5$ for sampling patches and a mini batch size $M = 24$ for training the estimator CNN network. The cropped patch size is fixed to 96×96 pixels, which we found to be a good trade-off between runtime, generalization and accuracy.

B. Registration performance

For measuring performance, we take the left-out 10 sequences and test the registration performance by adding small random distortions to the available ground truth calibration parameters $\mathbf{R}\mathbf{t}_{true}$. This is implemented by uniformly sampling random translations and rotations from the specified interval $\mathcal{U}(\cdot, \cdot)$. Because of the relatively large translation miscalibrations in the experiment, we found it helpful to increase the learning rate for parameter $\mathbf{t}$ to $\eta_\mathbf{t} = 10^{-2}$. Tab. I and Tab. II summarize the results. Figure 6 show the registration results for a representative test set example.

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

[3]https://github.com/xmba15/automatic_lidar_camera_calibration

Error R Euler [deg]	**Error t** xyz [m]	**Performance** $\mathbf{R}_\Delta$ [deg]	$\mathbf{t}_\Delta$ [m]
$\mathcal{U}(-2, 2)$	$\mathcal{U}(-0.30, 0.30)$	$\mathbf{0.11 \pm 0.05}$	$\mathbf{0.02 \pm 0.01}$
$\mathcal{U}(-2, 2)$	$\mathcal{U}(-0.60, 0.60)$	0.13 ± 0.07	0.03 ± 0.01
$\mathcal{U}(-5, 5)$	$\mathcal{U}(-0.30, 0.30)$	0.13 ± 0.06	0.03 ± 0.01
$\mathcal{U}(-5, 5)$	$\mathcal{U}(-0.60, 0.60)$	0.14 ± 0.06	0.03 ± 0.01
$\mathcal{U}(-7, 7)$	$\mathcal{U}(-0.30, 0.30)$	0.12 ± 0.07	0.02 ± 0.01
$\mathcal{U}(-7, 7)$	$\mathcal{U}(-0.60, 0.60)$	0.89 ± 1.13	0.41 ± 0.58
$\mathcal{U}(-8, 8)$	$\mathcal{U}(-0.30, 0.30)$	0.13 ± 0.07	0.02 ± 0.01
$\mathcal{U}(-8, 8)$	$\mathcal{U}(-0.60, 0.60)$	3.41 ± 4.14	0.31 ± 0.42
$\mathcal{U}(-9, 9)$	$\mathcal{U}(-0.30, 0.30)$	1.12 ± 2.64	0.07 ± 0.13

TABLE I: Registration results on the KITTI [2] challenge averaged over five uniform random sampled distortions in the specified interval, i.e., $r_x, r_y, r_z \sim \mathcal{U}(\cdot, \cdot)$ and $x, y, z \sim \mathcal{U}(\cdot, \cdot)$.

Method	**Performance** $\mathbf{R}_\Delta$ [deg]	$\mathbf{t}_\Delta$ [m]
gNMI [14]	8.17 ± 5.69	0.08 ± 0.05
pNMI [21]	1.08 ± 0.51	0.03 ± 0.01
CMRNet [16]	1.07 ± 0.77	0.33 ± 0.11
RegNet [9]	0.28 ± 0.20	0.06 ± 0.10
LCCNet [11]	0.16 ± 0.47	0.02 ± 0.02
LMI($N = 533$)	$\mathbf{0.14 \pm 0.02}$	$\mathbf{0.02 \pm 0.01}$
LMI($N = 235$)	0.23 ± 0.12	0.01 ± 0.01
LMI($N = 147$)	2.50 ± 2.01	0.39 ± 0.39
LMI($N = 58$)	3.05 ± 2.46	0.13 ± 0.03

TABLE II: Registration results on the KITTI [2] challenge with a random angle distortion $r_x, r_y, r_z \sim \mathcal{U}(-2, 2)$ and random translation offset $x, y, z \sim \mathcal{U}(-0.6, 0.6)$. We used the first sequences of the test dataset corresponding to sequences 10 to 13 in the original KITTI dataset. Results are averaged over five runs. To evaluate the dependence on the number of test examples, we also report results for our Local Mutual Information method (LMI) for different numbers of available frames N.

While the gNMI method fails on the KITTI dataset, mainly because of the very unreliable Lidar intensity values, LCCNet achieved a remarkable rotation error of $0.16°$ and a translation error of 0.02m. Note, that the large variance in Tab. II is mainly due to the per-frame calibration estimation. With at least 200 frames available, the LMI method is on par with the baselines. When more than 500 frames are available, the baselines could be surpassed.

C. Generalization performance

For testing the generalization performance, we use an unseen scenario of a docking maneuver of a motor boat consisting of 472 1919×1199 sized RGB images and Velodyne-128 Lidar scans. The mutual information estimator is again pre-trained on KITTI. The initial rotation estimate is $\mathbf{R} = \mathbf{I}$ and the known true solution is roughly 2° off (see Tab. III). We distorted the true translation by a small ϵ, as translation offset can be measured quite accurately in practice. In the given real world scenario, the translation miscalibration is relatively small compared to the angular offset. Therefore, we found it helpful to decrease the learning rate for parameter $\mathbf{t}$ to $\eta_\mathbf{t} = 10^{-4}$. In Fig. 1 we show an example demonstrating the generalization capabilities. Surprisingly most methods perform

Fig. 6: Visual registration results on the KITTI dataset. Note the accuracy at the bollard on the left-hand side of the image.

Method	Error $\mathbf{t}$ xyz [m]	Performance $\mathbf{R}_\Delta$ [deg]	$\mathbf{t}_\Delta$ [m]
LCCNet	$\mathcal{U}(-0.01, 0.01)$	1.80 ± 0.52 (1.84)	0.96 ± 0.27 (0.99)
	$\mathcal{U}(-0.02, 0.02)$	1.79 ± 0.52 (1.84)	4.80 ± 1.38 (4.91)
gNMI	$\mathcal{U}(-0.01, 0.01)$	1.17 ± 0.73 (0.67)	0.07 ± 0.03 (0.08)
	$\mathcal{U}(-0.02, 0.02)$	1.16 ± 0.39 (1.32)	0.06 ± 0.01 (0.05)
pNMI	$\mathcal{U}(-0.01, 0.01)$	0.96 ± 0.60 (0.74)	0.02 ± 0.01 (0.02)
	$\mathcal{U}(-0.02, 0.02)$	0.77 ± 0.48 (0.73)	0.02 ± 0.01 (0.02)
LMI	$\mathcal{U}(-0.01, 0.01)$	$\mathbf{0.62 \pm 0.60}$ (**0.28**)	$\mathbf{0.01 \pm 0.00}$ (**0.01**)
	$\mathcal{U}(-0.02, 0.02)$	0.72 ± 0.56 (0.67)	0.01 ± 0.01 (0.02)

TABLE III: Measuring generalization performance (mean±std (median)) for the maritime scenario with $R_{init} = \mathbf{I}$. Results are averaged over five runs with random measurement distortions. The manually found true solution is $r_x, r_y, r_z = (0.49, 1.09, -0.10)$, and $x, y, z = (0.77, 0.07, -0.11)$.

better than the learning approach LCCNet in the generalization scenario, indicating that the LCCNet learnt KITTI specific features that do not generalize well to the docking scene.

The LMI optimization of the unregistered test set took 3min on standard hardware using an Intel i7-7600 and a NVIDIA 1080Ti.

V. Conclusion

We presented a hybrid algorithm for targetless registration of a Lidar-camera system using mutual information maximization on local patches. The algorithm is suitable for situations where calibration is required (e.g., caused by weather conditions or mechanical changes), but no labelled data is available. Technically, our method optimizes the alignment of local shape patterns between camera and projected Lidar points over multiple frames. By using local patterns, the algorithm is able to generalize to completely new Lidar-camera configurations.

The main limitation of the algorithm comes from the chosen receptive field, i.e., the patch size, which constrains the maximum angular miscalibration to about $\pm 7^\circ$ and the translation miscalibration to approximately 0.5m in our experiments. As soon as the projected Lidar points lie outside of the corresponding RGB image patches, no valid error signal can be propagated. Hence, the real world region of convergence heavily depends on the distance of surrounding objects and the patch size needs to be increased for large distance problems. However, too large patch sizes would lead to decreased generalization performance, because large patches contain dataset-specific patterns that do not generalize to new configurations. A second limitation is the number of available frames and more important, prominent objects in the given sequence. When there are too few shape patterns (e.g., edges or occlusions) available for alignment, the method becomes unstable, similar to the existing methods.

By using a hybrid optimization scheme for calibrating temporal synchronized sensors, several extensions are possible for future work. One direction is optimizing other parameters like the lens distortion, another is integrating other sensor modalities like radar or infrared.

References

[1] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "Mutual information neural estimation," in *35th International Conference on Machine Learning*, ICML 2018, vol. 2, 2018, pp. 864–873.

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

[3] K. Yuan, Z. Guo, and Z. J. Wang, "Regnet: Tolerance aware lidar-camera online calibration with geometric deep learning and generative model," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6956–6963, 2020.

[4] H. J. Chien, R. Klette, N. Schneider, and U. Franke, "Visual odometry driven online calibration for monocular lidar-camera systems," in *Proceedings - International Conference on Pattern Recognition*. jan: vol. 0. Institute of Electrical and Electronics Engineers Inc, 2016, pp. 2848–2853.

[5] S. Mishra, G. Pandey, and S. Saripalli, "Extrinsic calibration of a 3d-lidar and a camera," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1765–1770.

[6] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

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

[8] P. Jiang, P. Osteen, and S. Saripalli, "Calibrating lidar and camera using semantic mutual information," *arXiv preprint arXiv:2104.12023*, 2021.

[9] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," *in IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 7995, pp. 1803–1810, 2017.

[10] G. Zhao, J. Hu, S. You, and C. C. J. Kuo, *CalibDNN: Multimodal Sensor Calibration for Perception Using Deep Neural Networks*, 2021. [Online]. Available: http://arxiv.org/abs/2103.14793

[11] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "Lccnet: Lidar and camera self-calibration using cost volume network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2894–2901.

[12] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *Advances in Neural Information Processing Systems*, vol. 2015, pp. 2017–2025, January 2015.

[13] J. Castorena, U. S. Kamilov, and P. T. Boufounos, "Autocalibration of lidar and optical cameras via edge alignment," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2862–2866.

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

[15] B. Nagy, L. Kovacs, and C. Benedek, "Sfm and semantic information based online targetless camera-lidar self-calibration," in *Proceedings - International Conference on Image Processing*. vol. 2019-Septe. IEEE Computer Society, sep: ICIP, 2019, pp. 1317–1321.

[16] D. Cattaneo, M. Vaghi, A. L. Ballardini, S. Fontana, D. G. Sorrenti, and W. Burgard, "Cmrnet: Camera to lidar-map registration," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1283–1289.

[17] W. Wang, S. Nobuhara, R. Nakamura, and K. Sakurada, *Soic: Semantic online initialization and calibration for lidar and camera*, 2020. [Online]. Available: http://arxiv.org/abs/2003.04260

[18] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[19] M. D. Donsker and S. S. Varadhan, "Asymptotic evaluation of certain markov process expectations for large time. iv," *Communications on Pure and Applied Mathematics*, vol. 36, no. 2, pp. 183–212, 1983.

[20] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2018.

[21] Z. Taylor and J. Nieto, "Automatic calibration of lidar and camera images using normalized mutual information," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. Citeseer, 2013.

[22] L. Bottou, "Online algorithms and stochastic approxima-p tions," *Online learning and neural networks*, 1998.

[23] M. D. Plumbley, "Geometrical methods for non-negative ica: Manifolds, lie groups and toral subalgebras," *Neurocomputing*, vol. 67, pp. 161–197, 2005.