23^{rd} *Computer Vision Winter Workshop*
Zuzana Kúkelová and Júlia Škovierová (eds.)
Český Krumlov, Czech Republic, February 5–7, 2018

Robust Maximum-likelihood On-line LiDAR-to-Camera Calibration Monitoring and Refinement

Jaroslav Moravec
Faculty of Mathematics and Physics
Charles University
Prague, Czech Republic
jaroslav.moravec@centrum.cz

Radim Šára
Department of Cybernetics
Czech Technical University
Prague, Czech Republic
https://orcid.org/0000-0002-2032-5764

Abstract. *In this paper, we present a novel method for online LiDAR–Camera system calibration tracking and refinement. The method is correspondence-free, formulated as a maximum-likelihood learning task. It is based on a consistency of projected LiDAR point cloud corners and optical image edges. The likelihood function is robustified using a model in which the inlier/outlier label for the image edge pixel is marginalized out. The learning is performed by a stochastic on-line algorithm that includes a delayed learning mechanism improving its stability.*

Ground-truth experimental results are shown on KITTI sequences with known reference calibration. Assuming motion-compensated LiDAR data the method is able to track synthetic rotation calibration drift with about 0.06 degree accuracy in yaw and roll angles and 0.1 degree accuracy in the pitch angle. The basin of attraction of the optimization is about $\pm$1.2 degree. The method is able to track rotation calibration parameter drift of 0.02 degree per measurement mini-batch. Full convergence occurs after about 50 mini-batches. We conclude the method is suitable for real-scene driving scenarios.

1. INTRODUCTION

With an increasing number of sensors and other measuring devices in today's robots and autonomous vehicles, efficient calibration, its on-line monitoring and refinement are of utmost interest. Among the areas, where LiDAR–Camera systems and thus their calibrations are widely used it belongs scene reconstruction [24], object detection [5], and also autonomous vehicles navigation systems [18].

In this paper, we focus on the LiDAR–Camera pair because these two are complementary in the information they provide. The LiDAR (Light Detection And Ranging) technology does not face the problem with low external light, as in optical cameras, thus it can be used in almost any lighting environment. Automotive LiDARs measure ranges up to 25 m with an accuracy of about 3 cm. On the other hand, the angular resolution of current rotating LiDARs is non-isotropic (about 0.1° – 0.2° horizontally and about 0.4° – 2° vertically), hence the resulting point cloud is rather sparse at larger distances. Cameras, on the other hand, have good and isotropic angular resolution, but they require good lighting conditions. This good complementation is the reason, why the LiDAR–Camera system is often used in robotics.

We propose a novel approach for an on-line extrinsic calibration refinement of such sensor pair. The method is easily adapted to a multi-LiDAR, multi-camera system. In this work, we assume that the system has already been intrinsically calibrated and extrinsically precalibrated using an off-line procedure. A good off-line calibration is time-expensive. Typically, it cannot be triggered when the system is in use. We present a low-overhead method that is capable of automatic and real-time calibration parameters refinement between a LiDAR and a camera in all the six degrees of freedom of the relative Euclidean (rigid) displacement (three for rotation and three for translation).

2. RELATED WORK

The early work on the topic of LiDAR–Camera systems calibration focused on 2D LiDARs. One of the very first approaches was proposed by Zhang et al. [30] in 2004. This method is based on observing a checkerboard-like marker set from multiple views

and using a point-on-plane constraint. Subsequent methods [17], [28] and [4] used points-on-line constraints (point should lie on edges of a frame) using a black triangular board and a V-shaped calibration target.

Methods of calibration of a 3D LiDAR relative to an optical camera can be divided into several groups. The first one [11, 22, 19] use a checkerboard-like planar target that is the same or similar to the one used in precalibration. Geiger et al. in [11] used several checkerboards placed around the room for multiple cameras calibration to LiDAR in only a single shot. Checkerboard planes are found in the point cloud and then their registration into the images defines the calibration parameters. In another work with checkerboards, Pusztai et al. [22] used seven corners (from intersections of plane triples) of a known calibration box and the projections of these corners in the image. With this, the calibration problem is reduced to a PnP problem, which was proposed to be solved by using one of already known algorithms e.g., [15] or [12]. A novel method in this group was introduced by Park et al. [21], where a board of polygonal shape was used. They found classic 2D–3D point correspondences and estimated the calibration parameters with these corresponding pairs. Other approaches use a different form of planar surfaces [27, 21]. Velas et al. in [27] used a two-step calibration. During the first phase, corners in the LiDAR point cloud and edges in the optical image are found. Then the markers with rectangular, circular or triangular shapes are detected (using Hough transform for the image and a RANSAC-based algorithm for the point cloud) and with them the calibration is estimated. The second phase then refines the calibration using a local search using dense features.

A different and relatively new group of approaches [2, 26, 14] use a sequence of captured frames and compute the calibration of parameters with the help of motions obtained from an Inertial Motion Unit (IMU) device by solving the hand-eye calibration problem. One of the latest methods by Taylor and Nieto [25] can be used with any camera, 3D LiDAR and navigational system and provides calibrations of the entire multi-sensor system from marker-less natural scenes. Achieving a good calibration accuracy with a method based on the hand-eye calibration task is difficult because the method is based on the presence of significant non-commutative rotation factors.

The approaches discussed so far needed customized calibration objects or additional devices providing reference information. The last group of accurate methods [1], [16], [20] and [3], where also our approach belongs, also use marker-less natural scenes. These methods are more accurate but they cannot provide a global solution, their capabilities are mostly limited to locally-optimized calibration refinement only. To the best of our knowledge, the approach of Williams et al. [29] gave the first practical method in this area. The method uses data from relatively unconstrained natural outdoor scenes obtained during normal driving. The method is based on minimizing the χ^2 statistic between LiDAR reflectivity and optical image intensity. In a similar work of Pandey et al. [20] the calibration algorithm maximizes the mutual information between LiDAR response intensity and optical image intensity. These methods may have problems in scenes with little reflectivity variations, although this has not been studied experimentally. Bileshi [1] directly correlates depth discontinuities in LiDAR data with image edges. Edges in 3D tend to co-occur with image edges (but not vice versa). A very similar approach was used by Levinson and Thrun in [16].

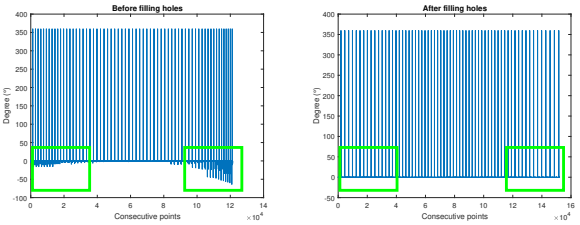
The last two mentioned methods solve a problem similar to ours. Hence, in this paper we do not propose a completely new calibration problem. Instead, we focus on algorithmic generality and simplicity, and on statistical efficiency. We found it critical for the estimation stability that a robust model for error computation between the LiDAR corners and the corresponding image edges be used. This is detailed in Sec. 3.2. With respect to this model, we were able to learn parameters of the calibration refinement using a stochastic gradient descent method, as described in Sec. 3.3. To improve stability, we use a delayed-learning mechanism. Sec. 4 gives results of several experiments that show efficiency, accuracy, learning speed, and the ability to track calibration parameter drift.

3. METHODS

3.1. Preprocessing

We assume we have a sequence of optical images $I_1, \ldots, I_n$ and the corresponding motion-compensated point clouds $P_1, \ldots P_n$. We call the pairs (I_f, P_f) frames.

A rotating LiDAR works by sending laser pulses into a set of angles whose resolution in the azimuth



(a) Before filling holes (b) After filling holes

Figure 1: Filling holes

angle is defined by the rotation speed and whose resolution in the elevation angle is given by the number of laser diodes. As it rotates, each diode provides one set of measurements per revolution. We call it a layer. A typical laser has 16 to 64 layers, depending on the model. For instance, the KITTI dataset [10] was created with a single 64-layer LiDAR.

The LiDAR point clouds P_f may contain holes when the laser beam from some azimuth and elevation angles does not return back. This creates gaps in data, which is not desired in fast convolution-based edge detection procedures. We estimate the azimuth difference between every two consecutive points in each layer and, as we know the angular resolution of our LiDAR device, we fill in the missing points between points with greater-than-expected angular difference using linear interpolation. The effect of the repair is illustrated in Fig. 1, which shows the difference in azimuth of two adjacent 3D points, before the repair and after it, respectively. The repaired point clouds will be denoted as $P'_1, \ldots, P'_n$.

With these preprocessed point clouds, we are able to find corner-like features. We use a simple corner detection filter. The corner points in the layer can have several shapes. They can be recognized using convolution with a matched filter. Point sequence with a similar local shape will have a high local value after the convolution and the local maxima of the filter response define corner point detections. We used only a step-like filter, which takes into account 50 points on both sides, as shown in Fig. 2. All points from the point cloud P'_f whose response is higher than a threshold are assigned as corner points in f-th frame C_f.

The proposed method also requires image edges. No sophisticated edge detection is needed. To obtain edge images E_f we used convolution with Sobel kernel on gray-scaled images I_f, again followed by non-maximum suppression and thresholding.

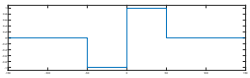


Figure 2: Convolution mask for LiDAR corner detection

3.2. Robust Model

Let θ be the parameters of the six degrees of freedom in the camera-to-LiDAR extrinsic calibration. Let $x_i \in E_f$ be a point from the set of all image edge points E_f in the current frame f. Given θ, let $y_j(\theta) \in C_f$ be the projection of j-th LiDAR corner from the set of all LiDAR corners C_f in the same frame (we consider only those points that are in front of the camera and fall in the image domain). We then define the robust negative log-likelihood function for frame f as

$$L_f(\theta) = -\frac{1}{c_f} \sum_{j=1}^{c_f} \log \left[k\,\tau + \sum_{i \in K_f(y_j(\theta))} e^{-\frac{\|x_i - y_j(\theta)\|^2}{2\sigma^2}} \right], \tag{1}$$

where c_f is the number of corners in C_f and $K_f(y_j(\theta))$ is the index set of k nearest neighbors of $y_j(\theta)$ in E_f, and k, τ, and σ are the parameters of the model.

We further consider mini-batches B_t that consist of b frames each. The likelihood for mini-batch t is just the average of the likelihoods over the frames in B_t

$$L_t(\theta) = \frac{1}{b} \sum_{f \in B_t} L_f(\theta). \tag{2}$$

The b is an additional parameter in this model.

The likelihood function $L_f(\theta)$ is based on a semi-parametric model that assumes that image edges are unoriented point-like local features and that there is an image edginess probability density $p(x)$ defined over the image domain. The $p(x)$ is a kernel density, with one non-Gaussian kernel per image edge pixel. The probability (density) value for a LiDAR corner projected to image position y_j is just $p(y_j)$. See Appendix A for a derivation leading to (1). Robustness of the likelihood function against outliers increases with increasing $0 \leq \tau \ll 1$.

Given mini-batch B_t the calibration parameters θ are obtained by minimization over the six-dimensional domain of θ. The likelihood function is non-convex due to its robustness. In this paper, we consider a local optimization method only.

We employ the exponential map for a minimal representation for the calibration parameters θ. Let

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \operatorname{expm} \mathbf{B}, \qquad \mathbf{B} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \mathbf{u} \\ \mathbf{0}^\top & 0 \end{bmatrix}$$

where $\mathbf{T}$ is the 4×4 Euclidean motion matrix and $\mathbf{B}$ is its principal logarithm [8] in which $[\boldsymbol{\omega}]_\times$ is a skew-symmetric matrix that represents cross-product as follows $[\boldsymbol{\omega}]_\times \mathbf{x} = \boldsymbol{\omega} \times \mathbf{x}$. We assumed that $0 \leq \|\boldsymbol{\omega}\| < \pi$ (the logarithm is a periodic function). Then $\theta = (\boldsymbol{\omega}, \mathbf{u}) \in \mathbb{R}^6$. In our coordinate system the elements $(\omega_0, \omega_1, \omega_2)$ of $\boldsymbol{\omega}$ are, approximately, the roll, pitch, and yaw angles.

Given a precalibration $\mathbf{T}_0$, a local optimization finds an update θ by first mapping LiDAR points by $\mathbf{T}_0$ and then finding an update θ such that

$$\mathbf{T}(\theta \mid \theta_0) = \operatorname{expm}\big(\mathbf{B}(\theta)\big)\, \mathbf{T}_0(\theta_0). \tag{3}$$

The $\mathbf{T}(\theta \mid \theta_0)$ is used in computing $y_j(\theta)$ needed in (1).

The mini-batch likelihood $L_t(\theta)$ is quite a noisy function. Some examples of $L_t(\theta)$ are shown in Figs. 5 and 6. One would need a very large mini-batch size to make the function smooth. That would mean collecting and storing large amounts of data prior to optimization. Since we need an on-line method, and data is plentiful, we decided to keep the mini-batch size small and use a stochastic gradient descent method. As long as the precalibration is reasonably accurate (see Sec. 4.3) and the calibration drifts away slowly (see Sec. 4.4) this is an adequate choice. The simplicity of stochastic gradient optimization also supports real-time performance.

3.3. AdaGrad

The stochastic gradient descent (SGD) methods are an iterative approach for a stochastic approximation of the gradient descent optimization.

After a few experiments with several SGDs,[1] we found AdaGrad [6] to be the most efficient among them. This approach needs to know the gradient of $L_t(\theta)$ with respect to all parameters, which can be easily approximated using numerical differentiation.

The AdaGrad method uses the gradient also to estimate an approximated Hessian matrix H for adapting the learning rate. After processing each batch t the matrix H is updated using a Robbins-Monro filter

$$H_t = (1 - \gamma_t)\, H_{t-1} + \gamma_t\, \nabla L_t(\theta_t) \nabla L_t(\theta_t)^\top, \tag{4}$$

[1] We have tried to use Momentum [7] (it was oscillating), Adam [13] (learning was too slow), and AdaGrad.

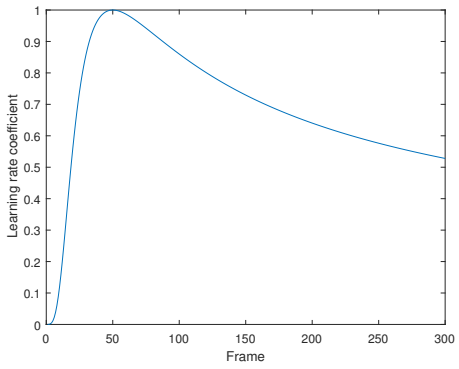


Figure 3: Delayed learning coefficient $\hat{\delta}_t$

where $\gamma_t = t^{-1}$ and $H_0 = \lambda \mathbf{I}$.

The parameter update step is then performed as

$$\theta_t = \theta_{t-1} + \nu\, \hat{\delta}_t\, \operatorname{diag}(H_t^{-\frac{1}{2}})\, \nabla L_t(\theta_t)\,, \tag{5}$$

where ν is the learning rate, $\hat{\delta}_t$ is the damping coefficient of the learning rate (a standard implementation uses $\hat{\delta}_t = t^{-1/2}$) and $\operatorname{diag}(H)$ is a matrix formed from the main diagonal of H.

We found it important to delay the start of the learning process in order not to escape from precalibration too far before the process starts converging. We therefore recognize a 'burn-in' period in the learning process. During the burn-in the method converges slowly, we even start with $\hat{\delta}_0 \to 0$. It is only important that the function $\hat{\delta}_t$ approaches $t^{-1/2}$ for $t \to \infty$. Hence, we use a redescendent function

$$\hat{\delta}_t = \left(\frac{t}{w}\right)^{a\,p} \left(\frac{p+q}{p\,\left(\frac{t}{w}\right)^{a} + q}\right)^{p+q}, \tag{6}$$

where w is the burn-in period end mini-batch, and, assuming the general convergence conditions [23],

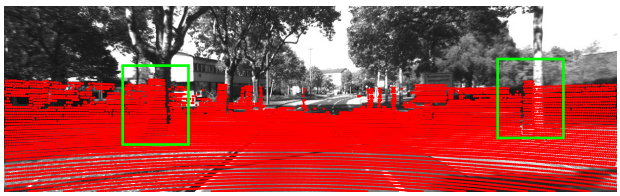
$$a,\, p,\, q > 0, \quad \frac{1}{2} \leq a\, q \leq 1\,,$$

we select

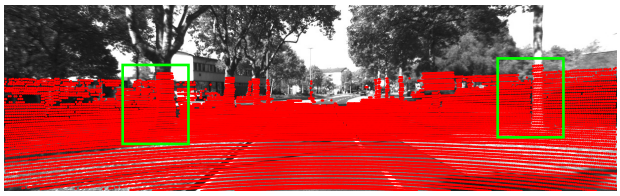
$$p = 2, \quad q = \frac{1}{4}, \quad a = 2, \quad w = 50\,. \tag{7}$$

For $t/w \to \infty$ this choice leads to $\hat{\delta}_t$ converging to $t^{-1/2}$ used in standard AdaGrad. Fig. 3 shows this $\hat{\delta}_t$ for $0 < t \leq 300$. We can see that the function climbs to unity by the end of the burn-in period and then starts descending towards zero.

Importantly, since angles are measured in different units than translations, the learning rate ν has to be scaled appropriately. We decided to use five times higher learning rate for the translation as it gave us the best results in our experiments.



(a) Projection of decalibrated 3D points



(b) After calibration correction

Figure 4: Validation. Camera-to-LiDAR pose is synthetically decalibrated and then the algorithm is run to correct the calibration.

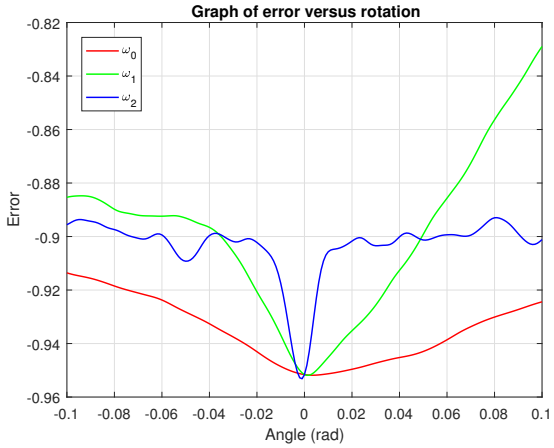


Figure 5: Likelihoods $L_t(\omega_j)$ as functions of the three elements of $\boldsymbol{\omega}$

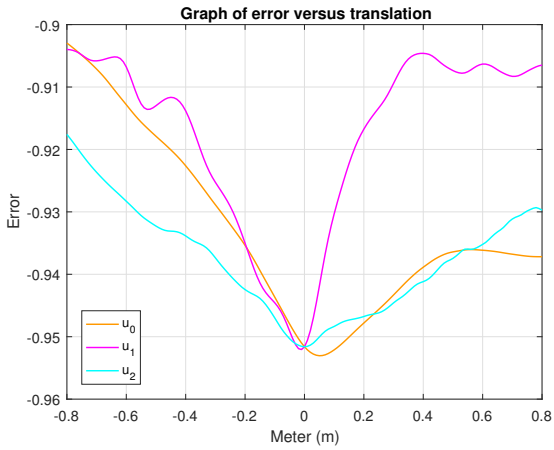


Figure 6: Likelihoods $L_t(u_j)$ as functions of the three elements of $\mathbf{u}$

4. EXPERIMENTS

In all experiments, we have used Velodyne data and their corresponding images from the KITTI dataset [10]. Drives 1, 2, 3, 5, 9, 11, 13, 14, 17, 18, 48, 51, 56, 57, 59 and 60 from 26$^{\text{th}}$ September 2011 were used.

We used $\sigma = 2\,\text{px}$, $\tau = 0.1$, $\nu = 0.002$, $\lambda = 10^{-4}$, $b = 10$, $k = 20$ and (7) in all experiments. The parameters were chosen manually. Based on our experiments, the σ, τ and ν need to be tuned (they are problem specific). The λ and b are not critical parameters, but they can influence the likelihood function smoothness. The value of k bigger than about 10 does not bring better results, it just increases the computation time.

4.1. Validation

We want to be sure that the proposed calibration refinement actually does what it should. Thus, when we run the algorithm with synthetically decalibrated LiDAR-Camera relative pose, we would like it to correct the calibration back.

Each 3D point of the frame in the mini-batch was synthetically rotated by δ_ω and the algorithm used 343 mini-batches to find the refinement. Fig. 4a shows the effect of one such decalibration with $\delta_\omega = (0.02, 0.02, 0.02)$ after the 3D points are projected to the image. Fig. 4b then shows LiDAR point cloud projection using the learned calibration correction. We can see that the method was able to correct the LiDAR–Camera calibration very well. Quantitative results of this experiment are discussed in Sec. 4.3.

4.2. Likelihood Function Shape

The likelihood function $L_t(\theta)$ should have a narrow and smooth global minimum, which could be easily found by the proposed method.

We have computed L_t from one thousand randomly selected frames from the union of all drives mentioned above as a function for each degree of freedom around zero, for rotation (Fig. 5) and for translation (Fig. 6).

When looking at the rotation, we can see that all three degrees of freedom have their global minimum near zero, which is expected because the estimated correction to the reference global calibration should vanish. Most of the scene objects have vertical edges and thus the rotation ω_2 around the z axis (the yaw angle) has the best global minimum, but also some

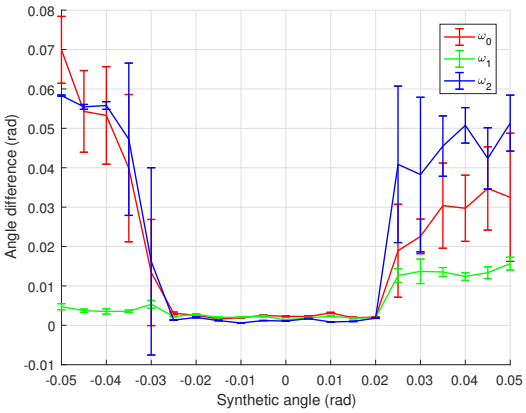


Figure 7: Basin of attraction for the elements of $\boldsymbol{\omega}$

other local minima, where a deterministic local optimization algorithm could get stuck. Rotations ω_0 and ω_1 about the other two axes (pitch, roll angles) exhibit minima of lower curvature indicating that the estimates for these angles will be less accurate. Rotation about the x axis has the widest minimum of the three, so it is the most difficult component of rotation to calibrate accurately. This is determined by the typical frame structure in the drives used in the experiment. For accurate calibration, one needs a wider variety in the scenes (horizontal and skew edges) and possibly a multi-LiDAR system. It is clear that any deployable on-line recalibration must preselect suitable frames for calibration parameter learning.

The translation estimates (see Fig. 6) are much harder to find accurately than rotations, especially in the case of the z translation. Again, this is because of a lack of horizontal edges in the scenes and a single LiDAR. This might also be the reason we saw a bias in the z translation shown in Fig. 4b from the previous experiment. The other two translations are much better, the y translation being the best one for the same reason as in the case of rotation around the z axis.

4.3. Locality of the Algorithm

The goal of this experiment is to show the ability of the proposed algorithm to find the reference calibration as a function of the amount of decalibration, i.e. the basin of attraction of the global minimum. We have used the proposed algorithm for calibration on synthetically decalibrated frames by rotation $\omega_i \in \langle -0.05, 0.05 \rangle$ with a step size equal to 0.005 radians and such that

$$\omega_0 = \omega_1 = \omega_2.$$

We repeated the calibration refinement estimation ten times for each set of rotations. Again, randomly selected frames from the dataset were used. As the experiment was very computationally demanding (210 runs), the algorithm was given only one hundred mini-batches in each run. The results are shown in Fig. 7. The flat valley in the middle is the basin of attraction. We can see that the method converges back to the reference calibration as long as the initial value falls within about 1.2 degree difference in all rotations. Then it starts to have problems with rotation about the x and the z axes. That is probably because the rotation about the z axis has a very narrow global minimum, according to Fig. 5.

4.4. Calibration Drift Tracking

This experiment was proposed by Levinson et al. in [16]. A calibration drift is generated as a sequence of angular deviations from the reference in which the initial deviation is zero and then, after each mini-batch, the deviation randomly increases by ± 0.02 degrees in each rotation parameter ω_i. We then tried to track this miscalibration by the proposed algorithm. The results are visualized in Fig. 8–10 that show the synthetic and the tracked calibration error of the three angles, during 686 mini-batches in two epochs. We follow the way results are presented in [16]. Ideally, the curves should coincide. In this experiment, the mean angle difference in pitch ω_0 is $0.052°$, for roll ω_1 it is $0.102°$ and for yaw ω_2 it is $0.047°$. We can see that in the case of the ω_1 and little bit also in the ω_2, there is a systematic error. We still need to identify the source of this bias. One of the reasons could be a bias in the reference calibration.

In [16] the experiment was done on a different dataset using 1,500 frames, thus the results cannot be directly compared. In their case, yaw had a mean average error of $0.06°$, and roll and pitch a mean average of $0.12°$ each. Therefore, we accomplished much better result in the roll, which is probably the hardest rotation in our model (according to the experiment in Sec. 4.2). And despite the unresolved systematic errors in pitch and little bit in yaw, our results are still slightly better in these two.

4.5. Efficiency

Our last experiment was testing the efficiency of the algorithm.

First, we were interested in the learning speed. We have executed the algorithm for twenty times using one hundred mini-batches. Frames were randomly

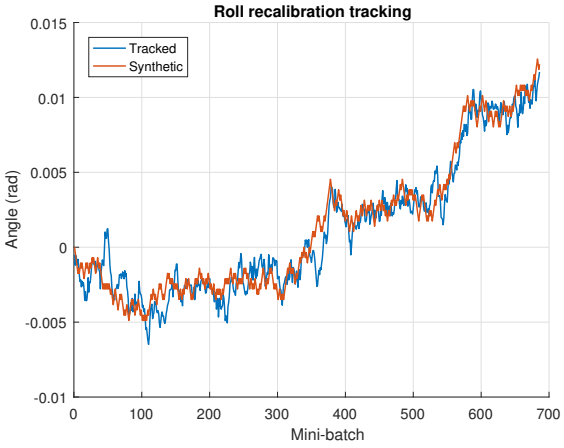


Figure 8: Tracking decalibration in ω_0

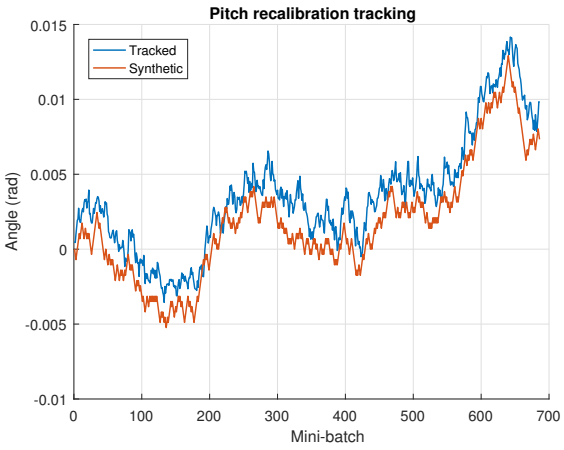


Figure 9: Tracking decalibration in ω_1

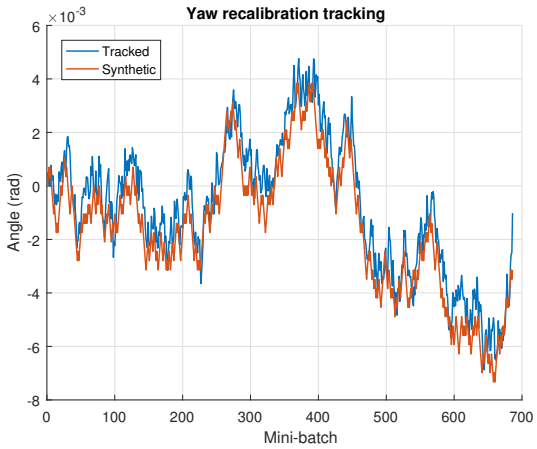


Figure 10: Tracking decalibration in ω_2

selected from the dataset. Each run had a constant miscalibration in the yaw angle ω_2 from the interval of $\langle -1, 1 \rangle$ degrees. Fig. 11 shows results of this

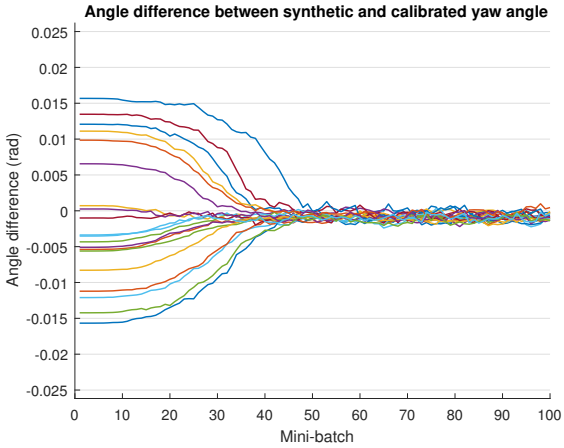


Figure 11: Efficiency (learning speed) of the proposed method

experiment. Each curve shows the convergence averaged over the 20 trials. We can see that after only about 50 mini-batches the reference calibration refinement (i.e. zero) was found in each run of the recalibration.

From the computational point of view, the bottleneck of the method is probably the k-Nearest Neighbors (kNN) algorithm. We used a kD-tree implementation from the Statistics toolbox of Matlab, based on [9]. Nevertheless, the proposed approach does not need the optimal solution to the kNN problem and thus one can use some fast approximation. The remaining parts of the algorithm are computationally very simple.

Our unoptimized experimental implementation in Matlab takes about half a second of processing per frame in a mini-batch, which is about $5\times$ slower than real-time performance.[2] Therefore we needed less than five minutes to find the reference calibration refinement.

5. CONCLUSIONS

The method described in this paper is based on a rigorous formulation of the calibration problem. We have verified that it is possible to improve a rather coarse initial estimate of the camera-to-LiDAR relative pose. This is especially true for two critical parameters: The relative yaw angle and the lateral baseline between the sensors. These can be refined very well. Even relatively fast calibration drift can be tracked without a noticeable delay. We conclude

[2]Executed on a personal laptop with Intel Core i7 6700HQ and 16 GB RAM.

the method is suitable for real-scene driving scenarios.

The proposed method relies on motion-compensated LiDAR data. In our future work we plan to address this shortcoming. It would also be interesting to see how global optimization methods cope with the likelihood function described in this paper. One could consider a MCMC method for this problem.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 688652.

References

[1] S. Bileschi. Fully automatic calibration of LiDAR and video streams from a vehicle. In *Proceedings IEEE International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1457–1464, Sept. 2009. 2

[2] Y. Bok, D. G. Choi, and I. S. Kweon. Generalized laser three-point algorithm for motion estimation of camera-laser fusion system. In *IEEE International Conference on Robotics and Automation*, pages 2880–2887, May 2013. 2

[3] J. Castorena, U. S. Kamilov, and P. T. Boufounos. Autocalibration of LiDAR and optical cameras via edge alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2862–2866, Mar. 2016. 2

[4] Z. Chen, L. Zhuo, K. Sun, and C. Zhang. Extrinsic calibration of a camera and a laser range finder using point to line constraint. *Procedia Engineering*, 29:4348–4352, 2012. 2

[5] H. Cho, Y. W. Seo, B. V. K. V. Kumar, and R. R. Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843, May 2014. 1

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. 4

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back propagating errors. *Nature*, 323:533–536, 10 1986. 4

[8] E. Eade. Lie groups for 2D and 3D transformations. Technical report, May 2017. 4

[9] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, Sept. 1977. 7

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 3, 5

[11] A. Geiger, F. Moosmann, . Car, and B. Schuster. Automatic camera and range sensor calibration using a single shot. In *IEEE International Conference on Robotics and Automation*, pages 3936–3943, May 2012. 2

[12] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (DLS) method for PnP. In *Proceedings International Conference on Computer Vision*, pages 383–390, Nov. 2011. 2

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[14] Q. V. Le and A. Y. Ng. Joint calibration of multiple sensors. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 3651 – 3658, Nov. 2009. 2

[15] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate $O(n)$ solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155, July 2008. 2

[16] J. Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems*, pages 24–28, June 2013. 2, 6

[17] G. Li, Y. Liu, L. Dong, X. Cai, and D. Zhou. An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3854–3859, Oct. 2007. 2

[18] P. Moghadam, W. S. Wijesoma, and D. J. Feng. Improving path planning and mapping based on stereo vision and LiDAR. In *10th International Conference on Control, Automation, Robotics and Vision*, pages 384–389, Dec. 2008. 1

[19] G. Pandey, J. McBride, S. Savarese, and R. Eustice. Extrinsic calibration of a 3D laser scanner and an omnidirectional camera. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, volume 43, pages 336 – 341, 2010. 2

[20] 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*, 32(5):696–722, 2015. 2

[21] Y. Park, S. Yun, C. S. Won, K. Cho, K. Um, and S. Sim. Calibration between color camera and 3D LIDAR instruments with a polygonal planar board. *Sensors*, 14(3):5333–5353, 2014. 2

[22] Z. Pusztai and L. Hajder. Accurate calibration of LiDAR-camera systems using ordinary boxes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Oct. 2017. 2

[23] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951. 4

[24] L. Smadja, J. Ninot, and T. Gavrilovic. Road extraction and environment interpretation from LiDAR sensors. In *International Archives of Photogrammetry and Remote Sensing (IAPRS)*, volume 38, pages 281–286, 2010. 1

[25] Z. Taylor and J. Nieto. Motion-based calibration of multimodal sensor arrays. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4843–4850, May 2015. 2

[26] P. N. Trujillo, P. Drews, R. P. Rocha, and J. Dias. Data fusion calibration for a 3D laser range finder and a camera using inertial data. In *Proceedings European Conference on Mobile Robots (ECMR)*, 2009. 2

[27] M. Veĺas, M. Španěl, Z. Materna, and A. Herout. Calibration of RGB camera with Velodyne LiDAR. In *WSCG 2014 Communication Papers Proceedings*, volume 2014, pages 135–144, 2014. 2

[28] S. Wasielewski and O. Strauss. Calibration of a multi-sensor system laser rangefinder/camera. In *Proceedings of the Intelligent Vehicles Symposium*, pages 472–477, Sept. 1995. 2

[29] N. Williams, K.-L. Low, C. Hantak, M. Pollefeys, and A. Lastra. Automatic image alignment for 3D environment modeling. In *Proceedings 17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 388–395, Oct 2004. 2

[30] Q. Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2301–2306, Sept. 2004. 1

A. Robust Kernel Density

We model the fact that a LiDAR corner may correspond to no image edge. Therefore we assume a hidden binary variable v_i per image edge point. The value of $v_i = 1$ means that a correspondence exists, and $v_i = 0$ means otherwise. We assume the edge image is a sample from a continuous probability density $p(x)$ over the image domain. This is represented as a kernel density approximated from k nearest (edge) neighbors of image point x as

$$p(x) \approx \frac{1}{k} \sum_{i \in \mathrm{knn}(x)} p(x \mid x_i)\,,$$

in which $p(x \mid x_i)$ is a marginal density derived from $p(x, v_i \mid x_i)$. As $k \to \infty$ this estimate converges to standard kernel density. The $p(x \mid x_i)$ is obtained as follows

$$\begin{aligned} p(x \mid x_i) &= p_1\, p(x \mid x_i, v_i = 1) \\ &\quad + (1 - p_1)\, p(x \mid x_i, v_i = 0) \\ &= \frac{p_1}{2\pi\sigma^2} \exp\left[-\frac{\|x - x_i\|^2}{2\sigma^2}\right] \\ &\quad + \frac{1 - p_1}{2\pi\sigma_o^2} \exp\left[-\frac{x^2}{2\sigma_o^2}\right] \\ &= \frac{p_1}{2\pi\sigma^2} \left(\exp\left[-\frac{\|x - x_i\|^2}{2\sigma^2}\right] \right. \\ &\quad \left. + \frac{1 - p_1}{p_1} \frac{\sigma^2}{\sigma_o^2} \exp\left[-\frac{x^2}{2\sigma_o^2}\right]\right) \end{aligned} \tag{8}$$

with $\sigma_o \gg \sigma$ and $p_1 \ll 1$, where $p_1 = p(v_i = 1)$ is the prior probability of observability. We assume the second term in the bracket is approximately constant. Let this constant be τ. Then

$$p(x) \approx \frac{p_1}{2k\pi\sigma^2} \sum_{i \in \mathrm{knn}(x)} \left(\tau + \exp\left[-\frac{\|x - x_i\|^2}{2\sigma^2}\right]\right). \tag{9}$$

Interestingly, from (8) we also have a corresponding inlier/outlier decision as a threshold T on $\|x - x_i\|$ such that

$$\frac{T}{\sigma} = \sqrt{-\log \tau^2}.$$