3D Trajectory Registration for Sensor Calibration

Tekla Tóth[1,*,†], Gábor Valasek[1,†] and Levente Hajder[1,†]

[1]*Department of Algorithms and Their Applications, Eötvös Loránd University, Pázmány Péter s. 1/C, 1117 Budapest, Hungary*

Abstract

Automatic sensor calibration is ubiquitous for robots and autonomous vehicles. Traditional target-based calibration methods typically require multiple registrations of an object in the field of view. However, by moving the object continuously while recording, the 3D positions of the target generate a path that may improve the calibration quality.

In this work, we propose a novel curve-based trajectory registration method to enhance calibration accuracy with spherical and checkerboard targets. This approach supplements the existing target position point set with estimated intermediate points after applying spline fitting techniques to the measurements. The curvature of the movement is aligned between the point sets resulting in reduced error between the registered point clouds and more accurate extrinsic calibration parameters. We validate these properties in indoor and outdoor real-world scenarios.

Keywords

sensor calibration, 3D trajectory, movement estimation, interpolation, pointset alignment

1. Introduction

The alignment of two or more point sequences occurs in various computer vision, computer graphics, and surface modeling problems [1]. Generally, state-of-the-art target-based sensor calibration can estimate the extrinsic parameters between *e.g.*, a LiDAR sensor, and a digital camera from a small number of input data pairs. Both the sphere-based method of Tóth et al. [2] and the checkerboard-based calibration of Zhou et al. [3] apply point set registration on some detected feature points to find the sought parameters. The methods can find the positions of a sphere center or the edges and corners of a checkerboard pattern in 3D from the sensor data with different modalities. The camera and the LiDAR take snapshots simultaneously while the target is moved around in the scene. In the final step, the original pipeline of these methods applies the point pair registration (PPR) of Arun et al. [4]: the detected pairwise positions are registered. This approach can be applied both between planar and spatial point sets.

Our goal is to improve object-based calibration processes via the directed continuous movement of the target. Instead of treating each calibration image pair independently and using point pairwise registration on them, we process the estimated trajectory points of the human-controlled movement (see in Fig. 2) as point clouds for each sensor device. Then we use point cloud registration methods [5, 6, 7] to compute the rigid body motion between the devices. This requires sufficiently large input data sets.

Our main insight is to expand the point clouds by inferred trajectory points. These are obtained by fitting an interpolating spline to the captured trajectory data and evaluating this spline at a user-adjustable frequency that is higher than the sampling rate. This assumes a temporal and spatial coherence between the calibration images and we show that it improves calibration accuracy. The proposed process is visualized in Fig. 1. We evaluate a spherical realization of the proposed method empirically for camera-LiDAR calibration in Section 3.

The motivation for our work was to improve the multimodal sensor calibration of a test vehicle (see Fig. 3) with a reduced camera data rate. In particular, our setup provided a 2-4 FPS (frames per second) video stream which resulted in a sparse trajectory sampling with about 10-20 centimeters between consecutive points. An example image sequence is presented in Fig. 2. Our idea is to approximate the trajectory with curve segments to get a more detailed path containing synthetic positions. During the measurement, synchronization errors occur due to the LiDAR turnaround time. This means that the endpoints of the paths are not necessarily aligned, but the common parts of the curves are, as much as possible. Regardless of the endpoint misalignments, the interior of the path itself may be robustly inferred; therefore, we can interpolate the curve between the measured data points and add some synthetic intermediate points. This technique can be also applied in the case of surfaces defined by sparse point set [8].

26th Computer Vision Winter Workshop, Robert Sablatnig and Florian Kleber (eds.), Krems, Lower Austria, Austria, Feb. 15-17, 2023

*Corresponding author.

†These authors contributed equally.

teklatoth@inf.elte.hu (T. Tóth); valasek@inf.elte.hu (G. Valasek); hajder@inf.elte.hu (L. Hajder)
http://cv.inf.elte.hu/ (L. Hajder)
0000-0002-7439-0473 (T. Tóth); 0000-0002-0007-8647 (G. Valasek); 0000-0001-9716-9176 (L. Hajder)

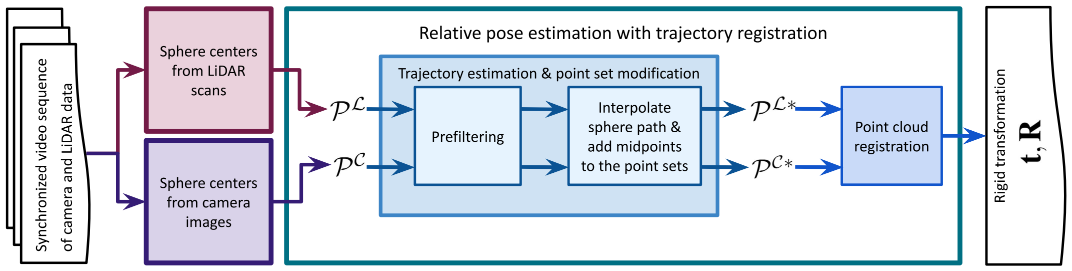
Synchronized video sequence of camera and LiDAR data
Sphere centers from LiDAR scans
Sphere centers from camera images
Relative pose estimation with trajectory registration
Trajectory estimation & point set modification
$\mathcal{P}^{\mathcal{L}}$
$\mathcal{P}^{\mathcal{C}}$
Prefiltering
Interpolate sphere path & add midpoints to the point sets
$\mathcal{P}^{\mathcal{L}*}$
$\mathcal{P}^{\mathcal{C}*}$
Point cloud registration
Rigid transformation
$\mathbf{t}, \mathbf{R}$

Figure 1: Schematic image of the proposed trajectory registration pipeline. The detailed process is discussed in Alg. 2.

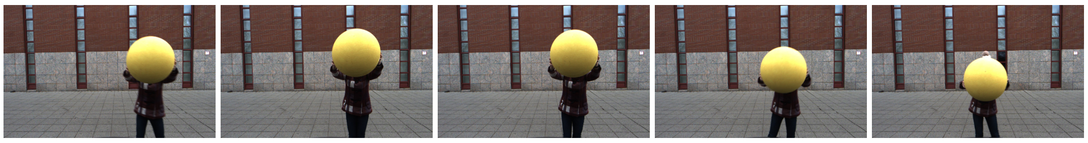

Figure 2: An input image sequence from the camera. The sphere center moves around 10-20 cm from one frame to the next regarding these images.

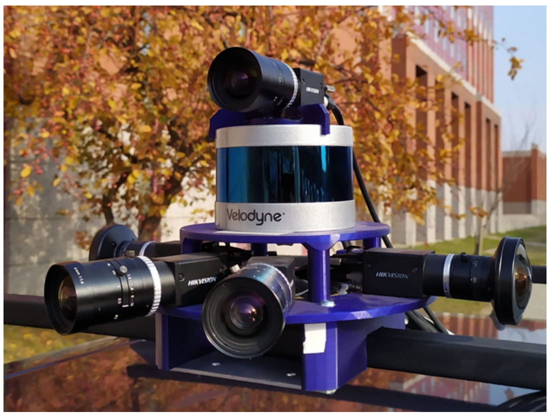

Figure 3: Camera-LiDAR setup of the test vehicle for the real-world tests. During the experiments, one camera was calibrated to the LiDAR. Expanding the proposed approach, multi-sensor calibration can be applied for all five cameras and the LiDAR.

1.1. Camera-LiDAR Synchronisation

We provide an overview on the synchronization system between the camera and the LiDAR as shown in Fig. 4. A Velodyne VLP-16 LiDAR device is spinning with a 1200 RPM (rotation per minute) while an Arduino Triggerbox triggers the camera with 4 FPS. If one trigger sign arrives, the camera mounted on the same vehicle exposes for a varying duration, at most for 0.04 seconds. Then the picture is taken by a global shutter, and the system triggers the LiDAR to save a synchronized turnaround. The LiDAR saves the data packages of every 15° rotation slice. When it gets the trigger, it finishes the current chunk of data, labels it as the end of the scan to be saved, and identifies the last 360° rotation as one complete synchronized turnaround to the current image. However, in the case of a moving object, the worst case is when the object appears at the beginning of the point cloud, which may cause an error 0.05 s at maximum. If the target object moves with a typical 0.2–0.4 m/s velocity, it causes 1–2 cm error. Therefore, the point pairwise registration may be inaccurate, while the proposed solution tries to deal with this particular problem.

2. Proposed method

This section introduces the evaluated variations of the proposed pose estimation described in Fig. 1 using interpolated points on the estimated trajectory and point cloud trajectory registration. Let $\mathcal{P}^{\mathcal{L}}$ and $\mathcal{P}^{\mathcal{C}}$ denote the point sets of estimated sphere centers from LiDAR data and camera images, respectively. The final task is to find the rigid transformation matrix $\mathbf{T} = [\mathbf{R}|\mathbf{t}]$ constructed by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$. In this paper, $\mathbf{R}^{\mathcal{LC}}$ and $\mathbf{t}^{\mathcal{LC}}$ describe the south rotation and translation from the camera frame to the LiDAR frame. In the particular case of sphere-based estimation, $n \geq 4$ point pairs are required.

First, let us consider the point pair solution as shown in Alg.1, based on the method of Arun et al. [4]. The point set registration is defined as a minimization problem:

$$\underset{\mathbf{R}^{\mathcal{LC}},\mathbf{t}}{\arg\min} \sum_{i=1}^{n} \left\| \mathbf{p}_i^{\mathcal{L}} - \mathbf{R}\mathbf{p}_i^{\mathcal{C}} - \mathbf{t}^{\mathcal{LC}} \right\|_2^2, \qquad (1)$$

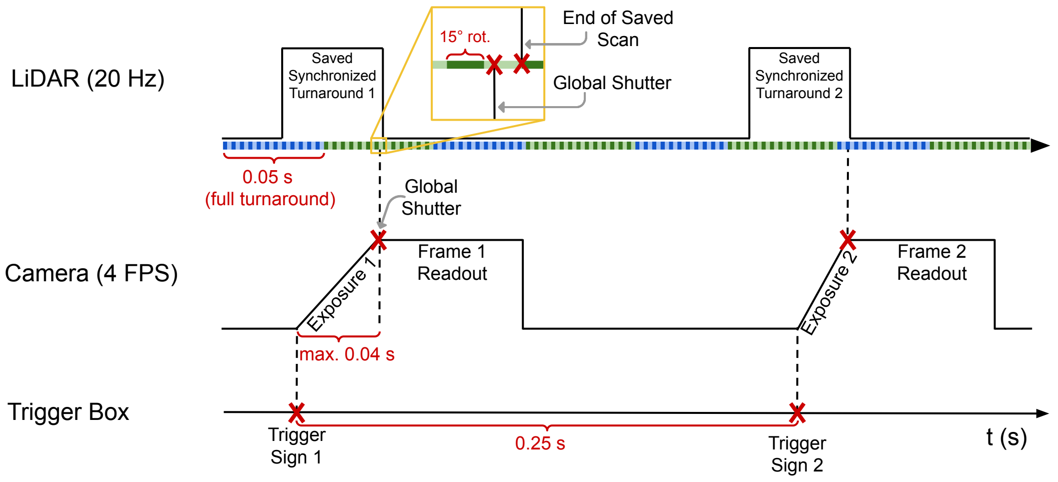

Figure 4: Hardware synchronization of the real camera-LiDAR system. The varying exposure time causes an unpredictable noise which motivates us to find a more robust solution for the point set registration during the calibration process.

where $\mathbf{p}_i^{\mathcal{L}}$ and $\mathbf{p}_i^{\mathcal{C}}$ denotes the i-th point pair of the LiDAR and the camera data. The translation is eliminated by choosing the point set barycenters as the coordinate system origins in the two images. The rotation can be derived using singular value decomposition (SVD) on the matrix $\mathbf{H}$ constructed from the point sets. If the point pairs are perfectly synchronized and noiseless, this method guarantees optimal results. However, in the presence of noise, such as due to synchronization errors as we described in Sec. 1.1, Eq. (1) no longer provides optimal solutions.

The improvements we propose are described in Alg. 2. The initial step is the pre-filtering of the point sets because noisy data may occur. This is based on scaled median absolute deviation:

$$m_i = s \cdot \text{median}\left(|x_i - \text{median}(\mathbf{x})|\right), \qquad (2)$$

where $x_i \in \mathbf{x}$ is an array of data, and s is the scale. As the approach focuses on the trajectories and not on the pairwise synchronized data, if any of the measurement data (either LiDAR or camera) is filtered out because of noise, the corresponding data may remain in the other point set, in contrast to the point pair registration. Then the algorithm has 2 main steps: (i) estimating the trajectory of the movement and (ii) registering the point clouds. The next subsections present all the realizations of these two steps that we evaluate in this paper.

2.1. Trajectory estimation

For trajectory estimation, the estimated 3D sphere positions can be interpreted as control points. To model the movement, the two main questions are the choice of the interpolation method and the sampling frequency to add new synthetic points to the point sets.

To find the best method, the following requirements must be met: the internal control points have to be interpolated while the segments tightly follow the knots without self-intersection between the points. Moreover, the final segment sequence has to be continuous and easy to evaluate at any point.

Linear interpolation (LI). A naive solution is the linear interpolation between the neighboring points of the measurement to reconstruct the path with line segments. Derivatives with a large magnitude of the curve are not expected. As rapid changes in the trajectory are not expected, the line-based approach is a good initial step; however, its precision depends on the target object movement speed and complexity. If two adjacent control points are denoted by $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ where $i \leq n-1$, n is the number of the points, and $t \in [0,1]$ is an independent parameter, the interpolation is

$$\mathbf{c}(t) = t \cdot \mathbf{p}_i + (1-t) \cdot \mathbf{p}_{i+1}. \qquad (3)$$

Catmull–Rom spline (CRS). The second approach is a curve-based solution applying cubic centripetal Catmull–Rom spline [9]. The Catmull-Rom spline parameterization produces curves that move toward the next control point and has small derivatives around control points. The advantages of the centripetal knot parametrization are that it fulfills the aforementioned requirements, *e.g.*, no self-intersections occur [10] in contrast to the uniform or the chordal parameterization. The control points are denoted by $\mathbf{p}_i$, and knots t_i, $i = 0, 1, 2, 3$ are also predefined. The derivatives are

Algorithm 1 Pose estimation with point pair registration – PPR [4]

```
Input: sphere centers from LiDAR data $\mathcal{P}^{\mathcal{L}} = \{\mathbf{p}_i^{\mathcal{L}} \in \mathbb{R}^3 \mid i = \{1,\dots,n\}, n \geq 4\}$; sphere centers from camera
    images $\mathcal{P}^{\mathcal{C}} = \{\mathbf{p}_i^{\mathcal{C}} \in \mathbb{R}^3 \mid i = \{1,\dots,n\}, n \geq 4\}$
Output: rotation matrix $\mathbf{R}^{\mathcal{LC}} \in \mathbb{R}^{3\times3}$ ; translation vector $\mathbf{t}^{\mathcal{LC}} \in \mathbb{R}^{3\times1}$
1: $\mathbf{t}^{\mathcal{LC}} := \overline{\mathcal{P}^{\mathcal{C}}} - \overline{\mathcal{P}^{\mathcal{L}}}$          ▷ The difference between the centers of gravity
2: $\mathbf{H} := \sum_{i=1}^{n} \mathbf{p}_i^{\mathcal{L}} (\mathbf{p}_i^{\mathcal{C}})^T$
3: $\mathbf{USV}^T := \mathrm{SVD}(\mathbf{H})$          ▷ Singular value decomposition of $\mathbf{H}$
4: $\mathbf{R}^{\mathcal{LC}} := \mathbf{VU}^T$
5: return $\mathbf{R}^{\mathcal{LC}}, \mathbf{t}^{\mathcal{LC}}$
```

Algorithm 2 Pose estimation with point cloud registration

```
Input: sphere centers from LiDAR data $\mathcal{P}^{\mathcal{L}} = \{\mathbf{p}_i^{\mathcal{L}} \in \mathbb{R}^3 \mid i = \{1,\dots,n\}, n \geq 4\}$; sphere centers from
    camera images $\mathcal{P}^{\mathcal{C}} = \{\mathbf{p}_i^{\mathcal{C}} \in \mathbb{R}^3 \mid i = \{1,\dots,n\}, n \geq 4\}$; $d > 0$ sampling interval on the trajectory spline
    approximation
Output: rotation matrix $\mathbf{R}^{\mathcal{LC}} \in \mathbb{R}^{3\times3}$ ; translation vector $\mathbf{t}^{\mathcal{LC}} \in \mathbb{R}^{3\times1}$
1: $\mathcal{P}^{\mathcal{L}}, \mathcal{P}^{\mathcal{C}} := prefilter(\mathcal{P}^{\mathcal{L}}, \mathcal{P}^{\mathcal{C}})$          ▷ Based on scaled median absolute deviation in Eq. 2
2: $\mathcal{P}^{\mathcal{L}*} := addInterpolatedPoints(\mathcal{P}^{\mathcal{L}}, d)$          ▷ Linear interpolation, Catmull–Rom spline [9],
3: $\mathcal{P}^{\mathcal{C}*} := addInterpolatedPoints(\mathcal{P}^{\mathcal{C}}, d)$          ▷ or Kochanek–Bartels spline [11]
4: $\mathbf{t}, \mathbf{R} := registerPointClouds(\mathcal{P}^{\mathcal{L}*}, \mathcal{P}^{\mathcal{C}*})$          ▷ ICP [5, 6] or CPD [7]
5: return $\mathbf{R}^{\mathcal{LC}}, \mathbf{t}^{\mathcal{LC}}$
```

given as

$$\mathbf{m}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{t_{i+1} - t_{i-1}}. \tag{4}$$

Based on the points $\mathbf{p}_i$ and the derivatives $\mathbf{m}_i$ calculated by (4), the Hermite curve is fitted data pair-wise. The equation of the spline is written as

$$\mathbf{c}(t) = \frac{t_2 - t}{t_2 - t_1}\mathbf{b}_1 + \frac{t - t_1}{t_2 - t_1}\mathbf{b}_2, \tag{5}$$

where

$$\begin{aligned}
\mathbf{b}_1 &= \frac{t_2 - t}{t_2 - t_0}\mathbf{a}_1 + \frac{t - t_0}{t_2 - t_0}\mathbf{a}_2,\\
\mathbf{b}_2 &= \frac{t_3 - t}{t_3 - t_1}\mathbf{a}_2 + \frac{t - t_1}{t_3 - t_1}\mathbf{a}_3,\\
\mathbf{a}_1 &= \frac{t_1 - t}{t_1 - t_0}\mathbf{p}_0 + \frac{t - t_0}{t_1 - t_0}\mathbf{p}_1,\\
\mathbf{a}_2 &= \frac{t_2 - t}{t_2 - t_1}\mathbf{p}_1 + \frac{t - t_1}{t_2 - t_1}\mathbf{p}_2,\\
\mathbf{a}_3 &= \frac{t_3 - t}{t_3 - t_2}\mathbf{p}_2 + \frac{t - t_2}{t_3 - t_2}\mathbf{p}_3.
\end{aligned}$$

The parameter $t \in [t_1, t_2]$ interpolates between $\mathbf{p}_1$ and $\mathbf{p}_2$. If the point coordinates are denoted by $\mathbf{p}_i = [x_i, y_i, z_i]$, then the knot parameterization of CRS is

$$t_{i+1} = l^\alpha + t_i, \tag{6}$$

where

$$l = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}, \tag{7}$$

and the scale $\alpha = 0.5$ describes the centripetal spline.

Kochanek–Bartels spline (KBS). The third trajectory approximation curve we evaluate is the Kochanek–Bartels spline [11] which is a generalized version of the Catmull–Rom spline extended by three parameters: the tension $\mathcal{T} \in [-1, 1]$, the bias $\mathcal{B} \in [-1, 1]$ and the continuity $\mathcal{C} \in [-1, 1]$. The derivative calculation is modified in the following way. The derivatives of the end-points of the i-th segments are as follows

$$\begin{aligned}
\mathbf{m}_i =& \frac{(1-\mathcal{T})(1+\mathcal{B})(1+\mathcal{C})}{2}(\mathbf{p}_i - \mathbf{p}_{i-1}) + \\
& \frac{(1-\mathcal{T})(1-\mathcal{B})(1-\mathcal{C})}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i), \qquad (8)\\
\mathbf{m}_{i+1} =& \frac{(1-\mathcal{T})(1+\mathcal{B})(1-\mathcal{C})}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \\
& \frac{(1-\mathcal{T})(1-\mathcal{B})(1+\mathcal{C})}{2}(\mathbf{p}_{i+2} - \mathbf{p}_{i+1}). \qquad (9)
\end{aligned}$$

The final curve segment can be defined as a cubic Hermite spline between the internal control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, for any $t \in [0, 1]$ as

$$\begin{aligned}
\boldsymbol{c}(t) =& (2t^3 - 3t^2 + 1)\mathbf{p}_i + (t^3 - 2t^2 + t)\mathbf{m}_i + \\
& (-2t^3 + 3t^2)\mathbf{p}_{i+1} + (t^3 - t^2)\mathbf{m}_{i+1} \,. \qquad (10)
\end{aligned}$$

Sampling interval. Regardless of the applied interpolation method in the second and third steps of Alg. 2, the sampling interval along the trajectory approximation spline has to be given. The Euclidean distance between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ points is denoted by d_i. The line or curve segment $\mathbf{c}_i(t)$ interpolates the end points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. If $d_i > d$, we add additional positions to the trajectory.

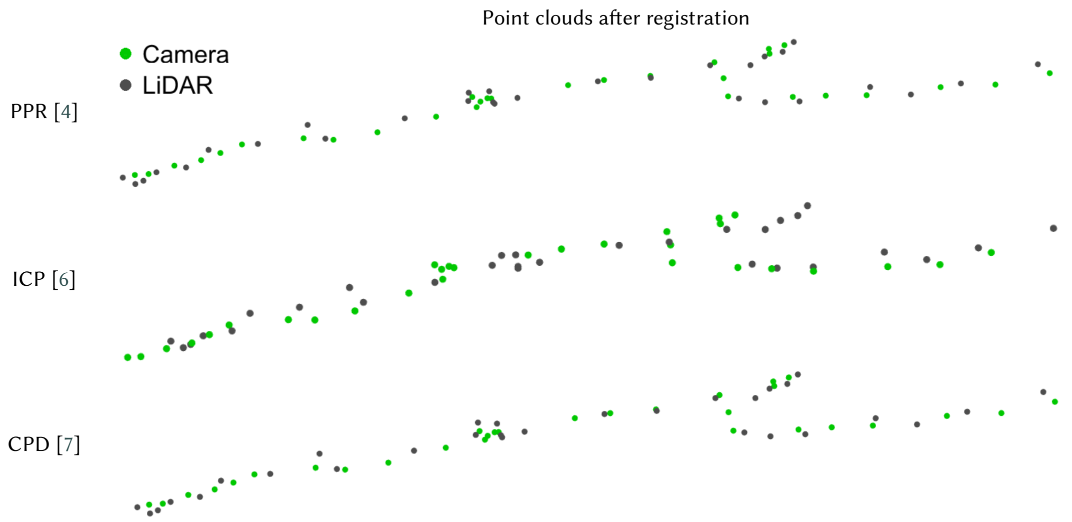

Figure 5: 3D sphere center trajectories after registration with point pair registration (PPR), iterative closest point (ICP) and coherent point drift (CPD) algorithms. In this case, no interpolated positions were added.

To this end, we evaluate the curve segment at every t step distance of d/d_i, and insert those into the trajectory path. This, however, yields trajectory point sets with different sizes; hence, point pair registration cannot be applied between them in the previous form.

2.2. Trajectory registration

After the spline fitting, the point pairs lost importance due to the quasi-continuous point set and varying sampling between the control points. Classical point cloud registration methods can be applied to dense 3D paths.

One of them is the iterative closest point (ICP) algorithm [5, 6]. The revised version improved the efficient closest point computation [6] using k-d tree. The other one is the coherent point drift (CPD) algorithm [7]. We tested the rigid version of the method.

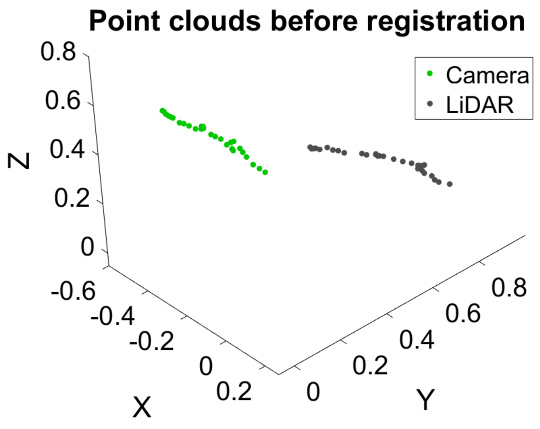

Figure 6: Estimated 3D sphere center trajectories before registration. The original calibration method provides sub-centimeter precise 3D positions.

3. Tests and results

We examined whether the registration results could be more precise using the new approach, and what method is the most suitable among the alternate versions. Furthermore, we tested the parameter setting for the sampling interval. The tests were implemented in MATLAB. To compare the results, we computed the root mean square error ($\mathcal{RMSE}$) of the Euclidean distance between the aligned point clouds. The sensors of the vehicle (see in Fig. 3 and the synchronization in Sec. 1.1) captured camera-LiDAR video sequences to test real-world scenarios.

The input of the algorithms is the 3D point sequences of sphere centers. The test input for illustration is visualized in Fig. 6 containing 30 positions per sensor which means a 7.5 sec. long video sequence in an outdoor environment in a parking lot. We processed this data feed as written in [2]. The sphere centers from images were estimated by ellipse detection, and then projecting it into 3D using the known radius of the sphere. The LiDAR-based sphere localization used a fix-point iteration method [12].

At first, we compared the trajectory registration methods discussed in Sec. 2.2 without estimating the move-

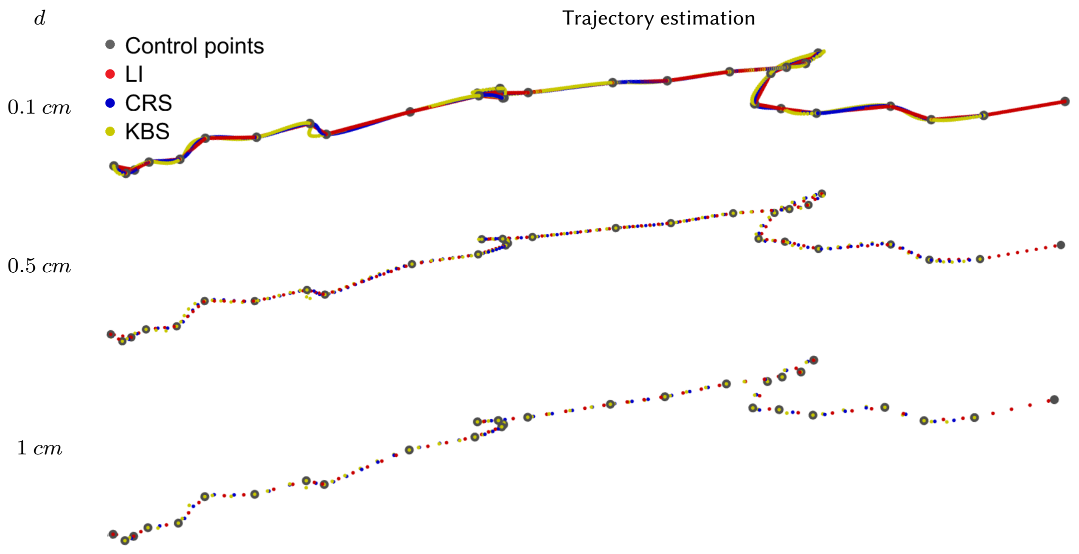

Figure 7: Estimated 3D sphere center trajectories after linear interpolation (LI), fitting Catmull–Rom spline (CRS), and Kochanek–Bartels spline (KBS) with varying sampling frequency d. The error rate is displayed in Tab. 1.

Interpolation method	$\mathcal{RMSE}$ w.r.t. sampling interval d (cm) 0.1	**0.25**	0.5	0.75	1.00
LI	0.53	**0.53**	0.55	0.58	0.61
CRS [9]	0.63	0.63	0.64	0.67	0.70
KBS [11]	0.65	0.65	0.67	0.69	0.72

Table 1
Test results of $\mathcal{RMSE}$ (in centimeters) for varying sampling interval and trajectory interpolations give subcentimeter errors for all three interpolation method. In case of sampling by $d \leq 0.25$ cm, the error rate and the point trajectory density is acceptable.

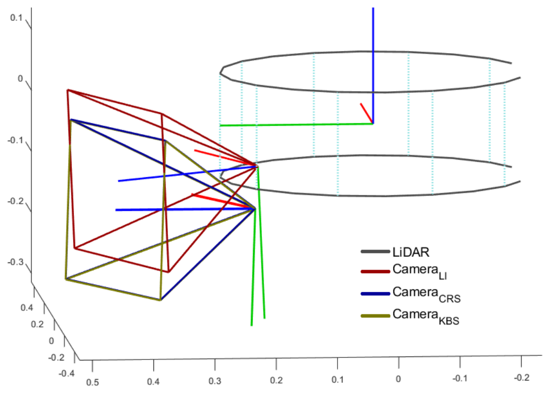

Figure 8: Test results of trajectory interpolations are applied to show the estimated rigid transformation in a schematic figure. The viewing frustum of the camera is in the LiDAR reference frame with the different interpolation methods with $d = 0.25$ cm sampling. The real-world camera is the bottom left one with perspective lenses in Fig. 3 on the left.

ment and any interpolated points. In Fig 5, the results for the test case in Fig. 6 are plotted. The $\mathcal{RMSE}$ of the ICP algorithm is 0.0155 m with 0.0114 m standard deviation while the PPR and CPD algorithms gave the same results, 0.0110 m with 0.0083 m standard deviation. Based on the experiments and the fact that applying PPR with point clouds of different sizes is not feasible, we recommend using the CPD.

We analyzed the interpolation methods and whether the d_i-based sampling gives the sought ideal density. We examined several setups in the same example in Fig. 7 and Fig. 1. In conclusion, one position per 0.25 cm is a sufficient sampling parameter. In this particular case, linear interpolation was suitable for the problem due to the stretched curvatures in the movement. Nonetheless, other scenarios require more adaptive approaches like CRS and KBS. Furthermore, the compared calibration results in Fig. 8 on the right suggest that the translation of LI is more accurate, but the rotation is expected as in the case of CRS and KBS.

Based on the test scenarios, the proposed settings of proposed Alg. 2 are the following. Tuning the sampling interval d to 0.25 cm gives sufficiently accurate results; besides, more dense sampling has no significant effect. Due to a linear-like input scenario, the linear approach performs the best in the exhibited use case, but the results are similarly acceptable with CRS and KBS. Finally, trajectory registration using CPD is the most effective choice. Applying this settings, the $\mathcal{RMSE}$ of the analyzed case was reduced from 0.0110 (PPR, no interpolated trajectory) to $0.0053\ m$ (CPD, $d = 0.25\ cm$, LI).

Synthetic tests provide the opportunity for direct comparison of the estimated calibration parameters to ground truth data. Based on early-stage results, the main open question is the best interpolation method for trajectory estimation which largely depends on the specific scenario. Paths with greater curvature give a better result for the rigid transformation with CRS and KBS. During the further refinement of the paper, we will analyze in more depth how the synthetized movement affects the precision of the proposed methods.

4. Conclusions

This paper argued how trajectory registration could improve extrinsic sensor calibration algorithms if we can extract 3D positions from the input data, *e.g.*, based on a specific target object. The novelty of the method is the controlled path of the object and the generated dense point set by spline fitting and interpolation followed by point cloud registration. The algorithm guarantees sample continuity and speeds up the measurement, as one video sequence is sufficient. The method handles the noise in the input images or the LiDAR scans independently owing to the point cloud-based approach which makes it more robust to the outliers.

There are some open questions regarding to the object path and the setup of the measurements. We would like to validate the better accuracy of the calibration via sythetic tests and analyze whether any ideal trajectory exist which makes the results more accurate. Moreover, we examine the effect of the noise in the 3D positions. Besides the spherical setup, chessboard-based approach could be also analyzed where an additional problem is the interpolation of the rotation, as not only one position but the four corner points can be detected in every frame. In the future, another application of the project can be the trajectory estimation of autonomous vehicles using multi-sensor systems.

References

[1] A. Hartey, A. Zisserman, Multiple view geometry in computer vision (2. ed.), Cambridge University Press, 2004. doi:10.1017/CBO9780511811685.

[2] T. Tóth, Z. Pusztai, L. Hajder, Automatic lidar-camera calibration of extrinsic parameters using a spherical target, in: 2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020, IEEE, 2020, pp. 8580–8586. doi:10.1109/ICRA40945.2020.9197316.

[3] L. Zhou, Z. Li, M. Kaess, Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018) 5562–5569.

[4] K. S. Arun, T. S. Huang, S. D. Blostein, Least-squares fitting of two 3-d point sets, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9 (1987) 698–700. doi:10.1109/TPAMI.1987.4767965.

[5] P. J. Besl, N. D. McKay, A method for registration of 3-d shapes., IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 239–256. URL: http://dblp.uni-trier.de/db/journals/pami/pami14.html#BeslM92.

[6] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, Int. J. Comput. Vision 13 (1994) 119–152. URL: https://doi.org/10.1007/BF01427149. doi:10.1007/BF01427149.

[7] A. Myronenko, X. Song, Point set registration: Coherent point drift, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 2262–2275. doi:10.1109/TPAMI.2010.46.

[8] R. Knothe, S. Romdhani, T. Vetter, Combining pca and lfa for surface reconstruction from a sparse set of control points, in: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition, FGR '06, IEEE Computer Society, USA, 2006, p. 637–644. URL: https://doi.org/10.1109/FGR.2006.31. doi:10.1109/FGR.2006.31.

[9] E. E. Catmull, R. Rom, A class of local interpolating splines, Computer Aided Geometric Design (1974) 317–326.

[10] C. Yuksel, S. Schaefer, J. Keyser, On the parameterization of catmull-rom curves, in: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, SPM '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 47–53. URL: https://doi.org/10.1145/1629255.1629262. doi:10.1145/1629255.1629262.

[11] D. H. U. Kochanek, R. H. Bartels, Interpolating splines with local tension, continuity, and bias control, SIGGRAPH Comput. Graph. 18 (1984) 33–41. URL: https://doi.org/10.1145/964965.808575. doi:10.1145/964965.808575.

[12] T. Tóth., L. Hajder., Robust fitting of geometric primitives on lidar data, in: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP„ INSTICC, SciTePress, 2019, pp. 622–629. doi:10.5220/0007572606220629.