Online Targetless End-to-End Camera-LIDAR Self-calibration

anonymous

CONFIDENTIAL. For review only.

Abstract

In this paper we propose an end-to-end, automatic, online camera-LIDAR calibration approach, for application in self driving vehicle navigation. The main idea is to connect the image domain and the 3D space by generating point clouds from camera data while driving, using a structure from motion (SfM) pipeline, and use it as the basis for registration. As a core step of the algorithm we introduce an object level alignment to transform the generated and captured point clouds into a common coordinate system. Finally, we calculate the correspondences between the 2D image domain and the 3D LIDAR point clouds, to produce the registration. We evaluated the method in various different real life traffic scenarios.

1 Introduction

State-of-the-art autonomous driving systems [3, 15], equipped with 3D LIDAR sensors and electro-optical cameras can achieve accurate and comprehensive environment perception. While real time LIDARs, such as Velodye's rotating multi-beam (RMB) sensors measure directly 3D geometric information with a relatively low spatial resolution, optical cameras provide us high resolution visual color and texture information enabling to also investigate small details in the scene.

Accurate LIDAR and camera calibration is essential for robust data fusion, which issues are therefore deeply studied in the literature. The existing calibration techniques can be grouped based on various aspects [10]: depending on the necessity of user interaction, prescribed specific environmental conditions, and operational requirements, one can distinguish semi- of fully automatic, target-based or targetless, and offline or online approaches (see Sec. 2). In this paper, we propose a novel targetless and fully automatic extrinsic calibration method between a camera and an RMB LIDAR mounted on a moving car. Our technique requires only to place and fix the sensors onto the vehicle top and to start driving in a typical dense urban environment, meanwhile the method calculates all the necessary registration parameters online. Considering that the RMB LIDAR point clouds are notably sparse and their density rapidly decreases as a function of the distance from the sensor, we only rely on robustly observable landmark objects, which are matched with blob components extracted from a synthesized 3D environment model obtained from the consecutive camera images by a Structure from Motion (SfM) approach.

The outline of the paper is as follows: A detailed literature overview of camera-LIDAR calibration is given in Sec. 2, while the steps of the proposed method are described in Sec. 3. In Sec. 4, we evaluate the new *targetless* and *fully automatic* method in real urban environment, and quantitatively demonstrate that its performance is close to a state-of-the-art *target-based* and *semi-automatic* calibration technique [12].

2 Related works

Target based methods use special calibration targets with a preliminary known size, such as polygonal planar boards [11], boxes [12], checkerboard patterns [4] or a simple printed circle [5]. Some of these techniques are semi-automatic, i.e. they require to take several images while the calibration pattern is manually moved [12], which approach may yield a notable accurate registration, but the calibration process is time consuming, and the quality depends on the operator's skills. Other methods attempt to automatically detect and match features on static targets: the approaches of [5, 14] are based on an automatic ellipse detector, [18] detects holes on planar object, [4] uses one LIDAR-image pair with multiple checkerboards adopting an improved corner detector, and [11] uses white homogeneous target objects to calibrate LIDAR and camera frames. While these approaches avoid user intervention during the calibration process, they can only be adopted in scenes where the calibration targets are available, and in some cases they need a complex installation of several carefully placed targets in a garage (e.g. [4] uses 12 checkerboards).

On the other hand, *targetless* approaches extract features for correspondence calculation directly from the observed *natural* environment without any calibration object. [16] transform the range sensor's 3D measurement into a new image, called Bearing Angle image (BA), and identifies point correspondences between the BA and the camera image via conventional image processing operations. Alternatively, mutual Information was introduced in [17] and [19] to calibrate different range sensors with cameras. However, experiments show that the above techniques require a critical point density of the point cloud for reliable operation, which

is not ensured at the single RMB LIDAR frames provided by a car during self-driving operation [19]. The correspondences in [7] are detected based on automatically extracted sets of lines both in the 2D images and in the 3D point clouds. However, according to [7] the method is preferably adopted in indoor environment, where the required number of line correspondences can be often observed, which condition cannot be guaranteed in the sparse RMB LIDAR frames recorded in outdoor urban environment.

Even a well calibrated system needs some re-calibration due to vibration on the roads and some sensor artifacts, so to avoid complex re-calibration offline process we propose an automatic, targetless, online registration method which is able to precisely calibrate LIDAR and camera sensors on the fly.

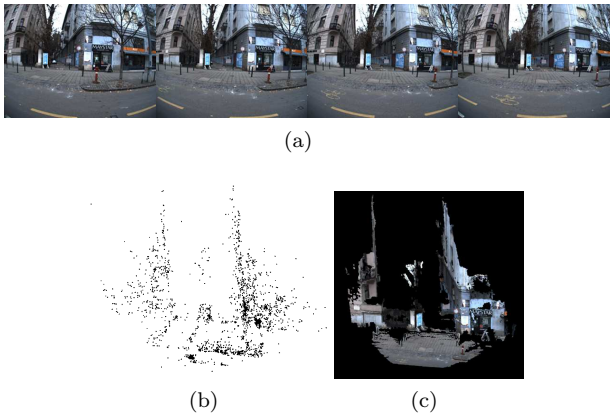
3 The proposed approach

Since state-of-the-art autonomous driving LIDAR sensors (e.g., Velodyne HDL64, VLP32 and VLP16) provide inhomogeneous, colorless, sparse point cloud streams with typical patterns, the main bottleneck of online, targetless calibration approaches is the extraction of meaningful feature correspondences from the 3D point cloud and the 2D image domain.

To avoid feature (2/3D interest points, line and planar segments) detection from very different domains we turn to a structure from motion (SfM) based approach [8] to generate point clouds from the image sequences recorded by the moving vehicle, and we perform an object level alignment between the LIDAR and the generated point clouds. During the SfM process we calculate the transformation T_1 which projects the points of the generated point cloud onto the corresponding image pixels. Then we align the point clouds on the object level [9] and we estimate a transformation T_2 which transforms the LIDAR point cloud to the coordinate system of the generated one. At this stage we can project the points of the LIDAR point cloud directly onto the image domain using transformation T_1 and we save the mapping between the corresponding 3D points and the 2D pixel coordinates. Finally, using T_2 inverse we transform back the LIDAR point cloud to the original position and using the saved 2D-3D mapping information we calculate transformation T_3 which is able to project the original LIDAR point cloud directly onto the 2D image domain.

3.1 Point cloud and transformation calculation from images

The first step of the proposed method is to generate a point cloud from a finite, continuous window of camera images that can be used for alignment and registration. First, we genarate a sparse point cloud by taking consecutive camera images. As a basis for these calculations we use the OpenMVG [1] library [8], with some modifications, as described in the following.



(a) (b) (c)

Figure 1. (a) 4 from a set of 8 images to process. (b) Generated sparse point cloud (2041 points). (c) Densified point cloud (257796 points).

To obtain a set of images for producing a sparse point cloud, we start capturing image frames. We select a set of N ($N \geq 3$) images - $N = 8$ is used for the rest of the paper. The resolution of the camera we used was 1288×964 pixels. When selecting the frames, we specify a global motion threshold of $th_{move} \geq 10$ pixels between consecutive frames, and feed these images into our structure from motion calculation pipeline, which has the following steps:

1) Image rectification: After selecting the N frames (Fig. 1(a)), perform rectification and store the selected frames.

2) Extraction and matching (L_2 fast cascade matching) of SIFT feature points for the selected images.

3) Structure from motion calculation: Perform the SfM, generate sparse point cloud (Fig. 1(b)) and store those image feature points that contribute to the sparse point cloud calculation with unique IDs. In this step, for each generated sparse 3D point we store all the 2D image points from all images that contributed to the estimation of the current 3D point of the sparse point cloud. During the point cloud generation process we assign unique IDs to all 3D points and save their associated image points from all selected images.

4) Using the stored 3D-2D point associations for all processed frames (Fig. 2(a-b)), we select M points from each processed frame based on point density (we used a constant $M = 45$) and from these 2D-3D associations we calculate the transformation using *solvePnP* from OpenCV [2]. Then, to check that the transformation is usable, we reproject all the points to all images using the obtained transformation, and Fig. 2 (c) shows an example. In the case of this example the re-

[1] Url: `https://github.com/openMVG/openMVG`

[2] Url: `https://opencv.org/`

(a) (b)

(c)

Figure 2. (a) Sparse cloud with each point assigned a unique color. (b) One frame showing color coded 2D points that contribute to the 3D point with the same color in (a). (c) Point cloud points reprojected to one of the respective 2D frames using the calculated transformation.

projection resulted in a very low average of 2.81 pixel reprojection error (averaged over all $N = 8$ processed images).

5) Densification of the generated point cloud (Fig. 1(c)): For densification we use the OpenMVS [3] library without modifications. The densified point cloud (Fig: 3(a)) and the calculated transformation will be used in the next steps for the cloud alignment and registration.

The above calculations can be performed either continuously, by selecting the next N frames in a moving time window, and updating the obtained transformations, or from time to time (e.g., every 5-10 minutes) since the vehicle's movements can cause slight (or not so slight) sensor displacements which can require regular updates. One benefit of the proposed approach is that there would be no need for returning to base to perform sensor calibrations, since the above calculations can be performed *in situ*, during the data capturing process.

3.2 Object based point cloud alignment

Several point cloud registration approaches can be found in the literature, however they are often just an extension of the ICP [20] or the NDT [6] process. These iterative methods usually fail if the initial translation is high or the density characteristic is quite different between the target and the source point cloud. Furthermore the typical ring pattern of the Velodyne LIDAR data on the ground region may mislead the registration process by matching the ground region improperly instead of finding correspondence between the foreground regions. So our proposed method, similarly to [2, 9] registers the point clouds on the object level to avoid slow and less robust point level transformation estimation.

We divide the P_z horizontal plane into 0.2 m size 2D grid cells and we assign each point of the point cloud to the corresponding cell. First we mark the ground region based on local neighborhood properties using an adaptive flood fill method starting from the sensor position in the 2D grid domain [1]. In the next step we extract connected components (objects) by merging the neighboring cells into object candidates (Fig. 3(a), 3(b)).

Since Sec. 3.1 operates with $N = 8$ long image sequences, the size of the generated point cloud is much smaller than the ranging distance of the Velodyne sensor, so we only consider an environment with 15 meter radius around the sensor's center.

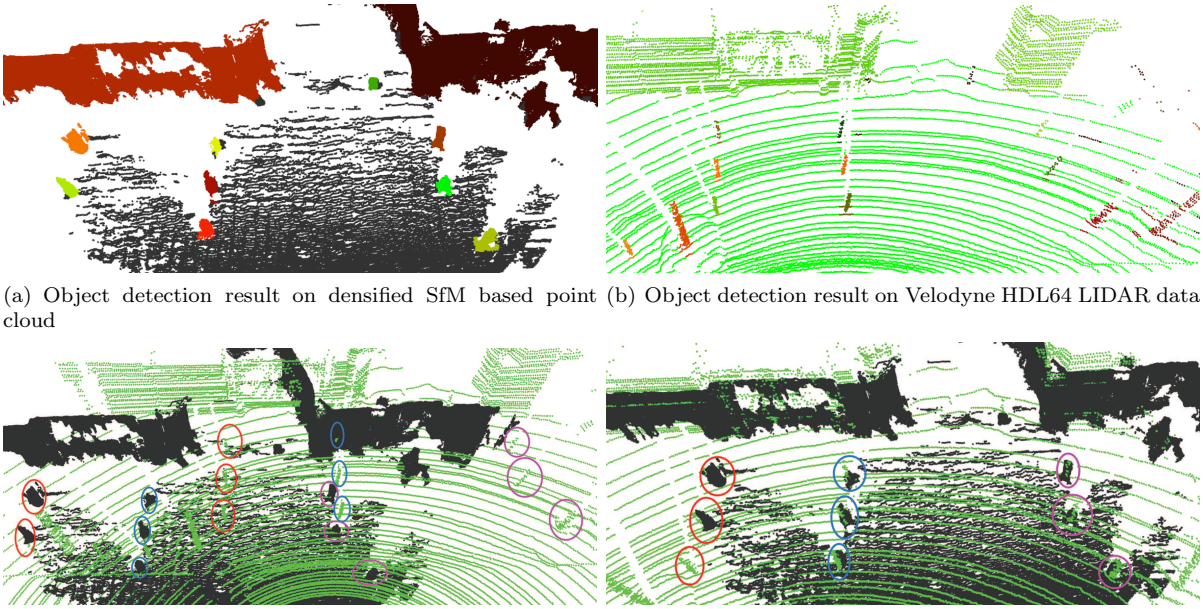
As a result of the object detection step we extract two sets of object centers O_1 and O_2 from the SfM-generated and the LIDAR-captured point clouds. Our aim is to find the global optimal matching between the two object sets by an iterative voting process [13] in the Hough space. During the transformation estimation we calculate the Euclidean distances between the objects, so finding valid object centers is crucial. Because of the several occlusion and scanning artifacts, larger objects such as *facade segments* and *larger vehicles* often fall apart during the object detection. These invalid object centers may mislead the transformation estimation, so first we eliminate larger objects and we only rely on street furniture such as *poles*, *columns*, *traffic posts* and *signs*, and smaller street objects such as *trashes* and *billboards*.

During the transformation estimation we take into account the translation and the rotation component around the *upwards* vector between the point clouds, because our measurements showed that within the 15 m search radius the effects of the other two rotation components were negligible. Thus, we define the problem as a 3D rigid body transformation which can be formulated as a rotation around the *upwards* vector with the proper α value and a 3D translation among the three coordinate axes. Mathematically we can formulate this transformation as follows:

$$T_{dx,dy,dz,\alpha} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & dx \\ -\sin\alpha & \cos\alpha & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Next, we define the transformation estimation as a discrete and finite problem, so we divide the possible transformation space into equal bins. First we allocate a 4D voting array $V[\alpha, d_x, d_y, d_z]$ which we can ad-

[3] Url: `http://cdcseacave.github.io/openMVS`



(a) Object detection result on densified SfM based point cloud

(b) Object detection result on Velodyne HDL64 LIDAR data

(c) Initial translation

(d) Proposed approach

Figure 3. Results of the proposed object based alignment method. (c-d) Velodyne LIDAR data is displayed with green, while the generated point cloud is shown with dark grey. Identical color circles show the corresponding parts of the point clouds.

dress by the given α rotation value and with the calculated translation components. The algorithm iterates through all the possible O_1 and O_2 object center pairs and it rotates O_2 with all the possible α values. Thereafter, it calculates the Euclidean distance between the rotated and the other center point:

$$\begin{bmatrix} dx^* \\ dy^* \\ dz^* \end{bmatrix} = o_1 - \begin{bmatrix} \cos\alpha^* & \sin\alpha^* & 0 \\ -\sin\alpha^* & \cos\alpha^* & 0 \\ 0 & 0 & 1 \end{bmatrix} o_2$$

In each iteration the method increases the evidence of each possible transformation candidate calculated by considering each possible object pair. Finally, we find the maximum value in the voting array which determines the best transformation by the corresponding rotation and translation components and accordingly, we transform the LIDAR point cloud into the coordinate system of the SfM-generated point cloud (Fig. 3(c) and 3(d)).

3.3 Projecting 3D point cloud onto the image domain

Since we have moved from the LIDAR's space to the SfM-generated point cloud coordinate system using transformation T_2, we can directly project the points of the LIDAR point cloud onto the 2D image domain using transformation T_1. As a result of the projection we can assign the 2D pixel coordinates to the corresponding 3D points. Since the absolute position of the LIDAR and the camera sensors is fixed, we transform back the LIDAR point cloud to the original position and based on the 2D-3D mapping we calculate transformation T_3 which is able to project the 3D points directly to the image domain without the intermediate transformation T_2. Knowing the internal parameters of the given camera we can determine transformation T_3 using the OpenCV *solvePnP* function which calculates the transformation matrix between a set of 3D and 2D points using the Levenberg-Marquardt optimization method.

Fig. 4(a) shows the result of the projection of the LIDAR data to the image domain using transformation matrix T_3.

4 Evaluation

We evaluated our proposed self-calibration method on a new manually annotated dataset (ground truth) and we compared it with a state-of-the-art target based offline calibration [12] method. The database contains two measurement sequences from the streets of an European capital city. While the second sequence was captured in strong sunshine, and the images are suffering from burn-out effect with varying contrast, the first measurement was recorded under cloudy weather conditions resulting in darker images with nearly uniform contrast. We manually annotated $10-10$ different images from each sequence, such as *heavy main roads*, *narrow streets* and *crossroads*.

(a) Prop. fully aut. targetless aproach. (seq. 1) (b) Semi aut. target based [12] approach. (seq. 1) (c) Prop. fully aut. targetless aproach. (seq. 2) (d) Semi aut. target based [12] approach. (seq. 2)

Figure 4. Qualitative comparison of the proposed online LIDAR-camera self-calibration approach and the [12] state-of-the-art offline calibration algorithm.

Pixel level projection errors and the standard deviation of the projection errors are shown in Table 1, furthermore, qualitative comparison versus [12] can be seen in Fig. 4. Although the results show that the offline target-based calibration method can produce higher accuracy registration, calibrating the camera and the LIDAR sensor with [12] method is a lengthy process, taking more than 1 hour, and when parameters change during measurements (e.g., because of sensor displacement) one needs to stop driving and the whole calibration process needs to be repeated.

Another artifact of conventional offline calibration [12] is coming from scanning platform movement: because of the nature of the RMB scanning process, as the speed of the scanning platform is increasing the shape of the point cloud is distorted and it is getting more and more stretched. Since offline calibration can only be performed on a parking vehicle, its accuracy may be decreased as the car moves with higher speed. The effect of this phenomenon is also demonstrated in Fig. 4 the first image pair (Fig. 4(a) and 4(b)) was captured from a static position, and here we can see that the target based reference method slightly outperforms our proposed targetless one. However, by the second pair (Fig. 4(c) and 4(d)), the shot was taken from a quicky moving car and it can be seen that in the current scene the proposed approach overcome the reference one.

The proposed self-calibration method calculates the correspondences between camera and LIDAR online during the operation of the vehicle and calculations can be repeated online periodically, thus, the average $6-7$ pixel error can be considered acceptable considering we process camera images with relatively large resolution (1288×964). At this resolution with $6-7$ pixel error we are able to robustly assign the 3D objects to the corresponding image regions using the calculated projection matrix, and this data fusion enables the autonomous vehicles to extract more visual features from the surroundings.

There can be situations when the sparse point cloud reconstruction process cannot produce a robust cloud, which might increase registration errors. However, the intended use case of the proposed approach is to periodically repeat the online alignment calculations, and only update the calibration when the current estimation improves the previously used one. Currently we perform such updates at fixed time intervals, however, in the future we also plan to include an automatic step to intelligently find suitable locations based on the current camera images.

Since the proposed self-calibration approach based on an object level alignment method, the quality of the registration is greatly depend on the amount and the type of the detected objects. Our experiments show that the proposed method performs better if the scenes contain column shaped objects such as *traffic signs*, *tree trunks* and *poles*, so after the object detection we count the number of the column shaped objects based on simple geometric constraints and we only calculate the calibration between the camera and the LIDAR if the given scene is appropriate. Typically in the case of main roads and larger crossroads containing several column shaped landmark objects the proposed self-calibration algorithm works more robust and accurate.

5 Conclusion

This paper proposed a targetless camera-LIDAR sensor self-calibration approach using 2D-3D data fusion, that can be performed on the fly, and updated periodically during the data capturing process, thus eliminating the need of lengthy offline sensor calibrations. The method uses a series of camera frames from a short (but continuous) time-window and the captured LIDAR sensor data to perform automatic 2D-3D registration and alignment. We evaluated the proposed method in real life scenarios using real sensors and data. In the future, we are working to make the method even more robust, lightweight, further decrease the average registration error, and incorporate it into autonomous vehicle processing and navigation pipelines.

Error	Average x translation error		Average y translation error	
	Average	Deviation	Average	Deviation
Semi automatic target based [12]	2.71	0.43	3.43	0.79
Proposed fully automatic targetless approach	6.86	1.26	7.57	0.98

Table 1. Performance analysis of the proposed self-calibration approach. Error values are in pixels.

6 Acknowledgment

Anonymous - acknowledgments will be included in the camera ready version.

References

[1] A. Börcs, B. Nagy, and C. Benedek. Fast 3-D urban object detection on streaming point clouds. In *Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving at ECCV'14*, volume 8926 of *LNCS*, pages 628–639. Springer, Zürich, Switzerland, 2015.

[2] B. Gálai, B. Nagy, and C. Benedek. Crossmodal point cloud registration in the Hough space for mobile laser scanning data. In *International Conference on Pattern Recognition (ICPR)*, pages 3374–3379, Cancun, Mexico, 2016. IEEE.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

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

[5] A. H., B. L.D., and B. B. Automatic calibration of a range sensor and camera system. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, page pp. 286292, October 2012.

[6] M. Magnusson. *The Three Dimensional Normal Distributions Transform an Efficient Representation for Registration, Surface Analysis, and Loop Detection.* PhD thesis, Örebro University, December 2009.

[7] P. Moghadam, M. Bosse, and R. Zlot. Line-based extrinsic calibration of range and image sensors. *2013 IEEE International Conference on Robotics and Automation*, pages 3685–3691, 2013.

[8] P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *2013 IEEE International Conference on Computer Vision*, pages 3248–3255, Dec 2013.

[9] B. Nagy and C. Benedek. Real-time point cloud alignment for vehicle localization in a high resolution 3d map. In *Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving (ECCV)*, Munchen, Germany, 2018.

[10] 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*, 32:696–722, 2015.

[11] Y. Park, S. M. Yun, C. S. Won, K. Cho, K. Um, and S. Sim. Calibration between color camera and 3d lidar instruments with a polygonal planar board. In *Sensors*, 2014.

[12] Z. Pusztai, I. Eichhardt, and L. Hajder. Accurate calibration of multi-lidar-multi-camera systems. In *Sensors*, 2018.

[13] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain. A real-time matching system for large fingerprint databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):799–813, Aug 1996.

[14] S. Rodriguez Florez, V. Fremont, and P. Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 214 – 219, 09 2008.

[15] D. L. Rosenband. Inside waymo's self-driving car: My favorite transistors. In *2017 Symposium on VLSI Circuits*, pages C20–C22, June 2017.

[16] D. Scaramuzza, A. Harati, and R. Siegwart. Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4164–4169, 2007.

[17] Z. Taylor and J. Nieto. A mutual information approach to automatic calibration of camera and lidar in natural environments. 12 2012.

[18] M. Velas, M. Spanel, Z. Materna, and A. Herout. Calibration of rgb camera with velodyne lidar. 2014.

[19] R. Wang, F. P. Ferrie, and J. Macfarlane. Automatic registration of mobile lidar and spherical panoramas. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 33–40, June 2012.

[20] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.