remote sensing

MDPI

Article

On-the-Fly Camera and Lidar Calibration

Balázs Nagy [1,2,*] and Csaba Benedek [1,2,3]

[1] Machine Perception Research Laboratory, Institute for Computer Science and Control, Kende Str. 13-17, 1111 Budapest, Hungary; benedek.csaba@sztaki.hu
[2] 3in-PPCU Research Group, Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, 2500 Esztergom, Hungary
[3] Faculty of Informatics, University of Debrecen, Kassai út 26, 4028 Debrecen, Hungary
* Correspondence: balazs.nagy@sztaki.hu

Received: 27 February 2020; Accepted: 31 March 2020; Published: 2 April 2020

check for updates

Abstract: Sensor fusion is one of the main challenges in self driving and robotics applications. In this paper we propose an automatic, online and target-less camera-Lidar extrinsic calibration approach. We adopt a structure from motion (SfM) method to generate 3D point clouds from the camera data which can be matched to the Lidar point clouds; thus, we address the extrinsic calibration problem as a registration task in the 3D domain. The core step of the approach is a two-stage transformation estimation: First, we introduce an object level coarse alignment algorithm operating in the Hough space to transform the SfM-based and the Lidar point clouds into a common coordinate system. Thereafter, we apply a control point based nonrigid transformation refinement step to register the point clouds more precisely. Finally, we calculate the correspondences between the 3D Lidar points and the pixels in the 2D camera domain. We evaluated the method in various real-life traffic scenarios in Budapest, Hungary. The results show that our proposed extrinsic calibration approach is able to provide accurate and robust parameter settings on-the-fly.

Keywords: lidar; camera; extrinsic calibration; registration

---

1. Introduction

Nowadays, state-of-the-art autonomous systems rely on wide range of sensors for environment perception such as optical cameras, radars and Lidars, therefore efficient sensor fusion is a highly focused research topic in the fields of self-driving vehicles and robotics. Though the resolution and the operation speed of these sensors have significantly improved in recent years, and their prices have become affordable in mass production, their measurements have highly diverse characteristics, which makes the efficient exploitation of the multimodal data challenging. While real time Lidars, such as Velodyne's rotating multi-beam (RMB) sensors provide accurate 3D geometric information with relatively low vertical resolution, optical cameras capture high resolution and high quality image sequences enabling to perceive low level details from the scene. A common problem with optical cameras is that extreme lighting conditions (such as dark, or strong sunlight) largely influence the captured image data, while Lidars are able to provide reliable information much less depending on external illumination and weather conditions. On the other hand, by simultaneous utilization of Lidar and camera sensors, accurate depth with detailed texture and color information can be obtained in parallel from the scenes.

Accurate Lidar and camera calibration is an essential step to implement robust data fusion, thus, related issues are extensively studied in the literature [1–3]. Existing calibration techniques can be grouped based on a variety of aspects [1]: based on the level of user interaction they can be semi- or fully automatic, methodologically we can distinguish target-based and target-less approaches, and in the term of operational requirements offline and online approaches can be defined (see Section 2).

In this paper, we propose a new fully automatic and target-less extrinsic calibration approach between a camera and a rotating multi-beam (RMB) Lidar mounted on a moving car. Our new method consists of two main steps: an object level matching algorithm performing a coarse alignment of the camera and Lidar data, and a fine alignment step which implements a control point based point level registration refinement. Our method relies on only the raw camera and Lidar sensor streams without using any external Global Navigation Satellite System (GNSS) or Inertial Measurement Unit (IMU) sensors. Moreover, it is able to automatically calculate the extrinsic calibration parameters between the Lidar and camera sensors on-the-fly which means we only have to mount the sensors on the top of the vehicle and start driving in a typical urban environment. In the object level coarse alignment stage, we first obtain a synthesized 3D environment model from the consecutive camera images using a Structure from Motion (SfM) pipeline then we extract object candidates from both the generated model and the Lidar point cloud. Since the density of the Lidar point cloud quickly decreases as a function of the distance from the sensor, we only consider key-points extracted from robustly observable landmark objects for registration. In the first stage, the object-based coarse alignment step searches for a rigid-body transformation, assuming that both the Lidar and the SfM point clouds accurately reflect the observed 3D scene. However in practice various mismatches and inaccurate scaling effects may occur during the SfM process; furthermore, due to the movement of the scanning platform ellipsoid-shape distortions can also appear in the Lidar point cloud. In the second stage, in order to compensate for these distortions of the point clouds, we fit a Non-Uniform Rational B-Spline (NURBS) curve to the extracted registration landmark key-points, which step enables to flexibly form the global shape of the point clouds.

The outline of the paper is as follows: in Section 2 we give a detailed insight into the literature of camera-Lidar calibration, Section 3 introduces the proposed method and finally in Section 4 we quantitatively and qualitatively evaluate our fully automatic and target-less approach in real urban environment and we compare the performance of the proposed method against state-of-the-art target-based [2] and target-less calibration techniques [4,5].

A preliminary stage of the proposed method has been introduced in [6,7]. This paper presents a more elaborated model, with a significant extension using a curve-based non-rigid transformation component, while we also provide here more extensive evaluation and comparison against several reference techniques.

2. Related Works

As mentioned above, extrinsic calibration approaches can be methodologically divided into two main categories: target-based and target-less methods.

As their main characteristics, target-based methods use special calibration targets such as 3D boxes [2], checkerboard patterns [8], a simple printed circle [9], or a unique polygonal planar board [10] during the calibration process. In the level of user interactions, we can subdivide target-based methods into semi-automatic and fully-automatic techniques. Semi-automatic methods may consist of many manual steps, such as moving the calibration patterns in different positions, manually localizing the target objects both in the Lidar and in the camera frames, and adjusting the parameters of the calibration algorithms. Though semi-automatic methods may yield very accurate calibration, these approaches are very time consuming and the calibration results highly depend on the skills of the operators. Moreover, even a well calibrated system may periodically need re-calibration due to artifacts caused by vibration and sensor deformation effects.

Fully-automatic target-based methods attempt to automatically detect previously defined target objects, then they extract and match features without user intervention: Velas et al. [11] detect circular holes on planar targets, Park et al. [10] calibrate Lidar and camera by using white homogeneous target objects, Geiger et al. [8] use corner detectors on multiple checkerboards and Rodriguez et. al. [12] detect ellipse patterns automatically. Though the mentioned approaches do not need operator interactions, they still rely on the presence of calibration targets, which often should be arranged in complex setups (i.e., [8] uses 12 checkerboards). Furthermore, during the calibration both the platform and the targets must be motionless.

On the contrary, target-less approaches rely on features extracted from the observed scene without using any calibration objects. Some of these methods use motion-based [13–15] information to calibrate the Lidar and camera, while alternative techniques [1,5] attempt to minimize the calibration errors using only static features.

Among motion-based approaches, Huang and Stachniss [14] improve the accuracy of extrinsic calibration by the estimation of the motion errors, Shiu and Ahmad [13] approximate the relative motion parameters between the consecutive frames, and Shi et al. [16] calculate sensor motion by jointly minimizing the projection error between the Lidar and the camera residuals. These methods estimate first the trajectories of the camera and Lidar sensors either by visual odometry and scan matching techniques, or by exploiting IMU and GNSS measurements. Thereafter they match the recorded camera and Lidar measurement sequences assuming that the sensors are rigidly mounted to the platform. However, the accuracy of these techniques strongly depends on the performance of trajectory estimation, which may suffer from visual featureless regions, low resolution scans [17], lack of hardware trigger based synchronization between the camera and the Lidar [16], or urban scenes without sufficient GPS coverage.

We continue the discussion with single frame target-less and feature-based methods. Moghadam et al. [5] attempt to detect correspondences by extracting lines both from the 3D Lidar point cloud and the 2D image data. While this method proved to be efficient in indoor environments, it requires a large number of line correspondences, a condition which cannot be often satisfied in outdoor scenes. A mutual information based approach has been introduced in [18] to calibrate different range sensors with cameras. Pandey et al. [1] attempt to maximize the mutual information using the camera's grayscale pixel intensities and the Lidar reflectivity values. Based on Lidar reflectivity values and grayscale images Napier et al. [19] minimize the correlation error between the Lidar and the camera frames. Scaramuzza et al. [4] introduce a new data representation called the Bearing Angle image (BA) which is generated from the Lidar's range measurements. Using conventional image processing operations, the method searches for correspondences between the BA and the camera image. As a limitation, target-less feature-based methods require a reasonable initial transformation estimation between the different sensors measurement [16], and mutual information based matching is sensitive to inhomogeneous point cloud inputs and illumination artifacts, which are frequently occurring problems when using RMB Lidars [1].

In this paper, we propose a two-stage fully automatic target-less camera-Lidar calibration method, which requires neither hardware trigger based sensor synchronization, nor accurate self-localization, trajectory estimation or simultaneous localization and mapping (SLAM) implementation. It can also be used by experimental platforms with ad-hoc sensor configurations, since after mounting the sensors to the car's roof top, all registration parameters are automatically obtained during driving. Failure of the SfM point cloud generation or SfM-Lidar point cloud matching steps in challenging scene segments does not ruin the process, as the estimation only concerns a few consecutive frames, and it can be repeated several times for parameter re-estimation.

Note that there exist a few end-to-end deep learning based camera and Lidar calibration methods [3,20] in the literature, which can automatically estimate the calibration parameters within a bounded parameter range based on a sufficiently large training dataset. However, the trained models cannot be applied for arbitrary configurations, and re-training is often more resource intensive than applying a conventional calibration approach. In addition, the failure case analysis and analytical estimation of the limits of operations are highly challenging for black box deep learning approaches.

3. Proposed Approach

Rotating multi-beam Lidar sensors such as Velodyne HDL64, VLP32 and VLP16 are able to capture point cloud streams in real time (up to 20 frames/s) providing accurate 3D geometric information (up to 100 m) for autonomous vehicles, however, the spatial resolution of the measurement is quite limited and typical ring patterns appear in the obtained point clouds. While most of the online, target-less calibration approaches attempt to extract feature correspondences from the 2D image and the 3D point cloud data, such as various key-points, lines and planes, we turn to a structural approach to eliminate the need for unreliable cross-domain feature extraction and matching.

Our proposed approach is an automatic process consisting of a number of algorithmic steps for Lidar-camera sensor calibration, as presented in (Figure 1). To avoid sensitive feature matching (2/3D interest points, line and planar segments), we propose a two-stage calibration method where first we use a Structure from Motion (SfM) [21] based approach to generate a 3D point cloud from the consecutive image frames recorded by the moving vehicle (see Figure 2). In such manner, the calibration task can be defined as a point cloud registration problem. Then, a robust object-based coarse alignment method [22] is adopted to estimate an initial translation and rotation between the Lidar point cloud and the synthesized 3D SfM data. In this step, connected point cloud components called abstract objects are extracted first, followed by the calculation of the best object level overlap between the Lidar and the synthesized SfM point clouds, based on the extracted object centers. A great advantage of the method is that although the number of the extracted object centers can be different in the two point clouds, the approach is still able to estimate a robust transformation. Following the coarse initial transformation estimation step we decrease the registration error using the Iterated Closest Point (ICP) point level method. Thereafter, to compensate the effects of the non-linear local distortions of the point clouds (SfM errors, and shape artifacts due to platform motion), we introduce a novel elastic registration refinement transform, which is based on non-uniform rational basis spline (NURBS) approximation (see details later). Finally, we approximate the 3D–2D transformation between the Lidar points and the corresponding image pixels by an optimal linear homogeneous coordinate transform.

In our experiments, we observed a common problem that in SfM point clouds, dynamic objects such as vehicles and pedestrians often fall apart into several blobs due to several occlusions and artifacts during the SfM processing. This phenomenon significantly reduced the performance and robustness of the matching. To handle this issue we introduce a pre-processing step: before the coarse alignment we eliminate the dynamic objects both from the Lidar and the synthesized SfM point cloud data by applying state-of-the-art object detectors. We use Mask R-CNN [23] to provide an instance level semantic segmentation on images, while the Point pillars method [24] to detect dynamic objects in the Lidar point cloud.

As output, the proposed approach provides a 4×3 $\hat{T}$ matrix which represents an optimized linear homogeneous transform between the corresponding 3D Lidar points and the 2D image pixels. To generate $\hat{T}$, our algorithm calculates three matrices (T_1, T_2 and T_3) and a non-rigid transformation ($\mathcal{T}_\star$): The first matrix, T_1 is calculated during the SfM point cloud synthesis, and it represents the (3D-2D) projection transformation between the synthesized SfM point cloud and the first image of the given image sequence which was used to create the SfM point cloud. Hence matrix T_1 can be used to project the synthesized 3D points to the corresponding pixel coordinates of the images. Matrix T_2 represents the coarse rigid transform (composed by a translation and rotation components) between the Lidar and SfM point clouds, estimated by the object level alignment step, while T_3 is the output of the ICP-based registration refinement. The non-rigid body transformation $\mathcal{T}_\star$ scales and deforms the local point cloud parts compensating the distortion effects of the SfM and Lidar point cloud synthesis processes. Finally, we approximate the cascade of the obtained transforms T_1, T_2, T_3, $\mathcal{T}_\star$ with a single global homogeneous linear coordinate transform $\hat{T}$, using the EPnP [25] algorithm:

$$(T_1, T_2, T_3, \mathcal{T}_\star) \xrightarrow{\text{EPnP}} \hat{T}. \tag{1}$$

Note that we will also refer to the composition of the two 3D–3D rigid transform components as $T_R = T_2 \cdot T_3$. The steps of the proposed approach are detailed in the following subsections.

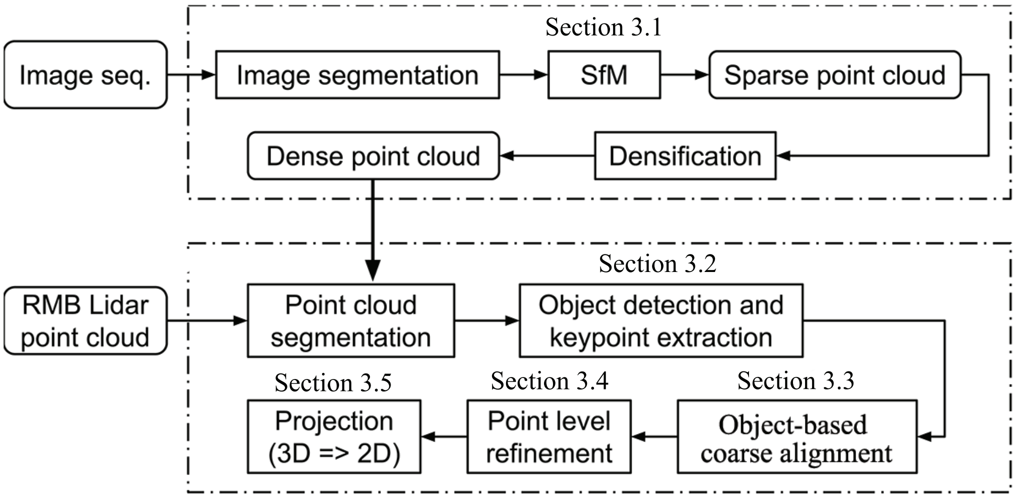


Figure 1. Workflow of the proposed approach.

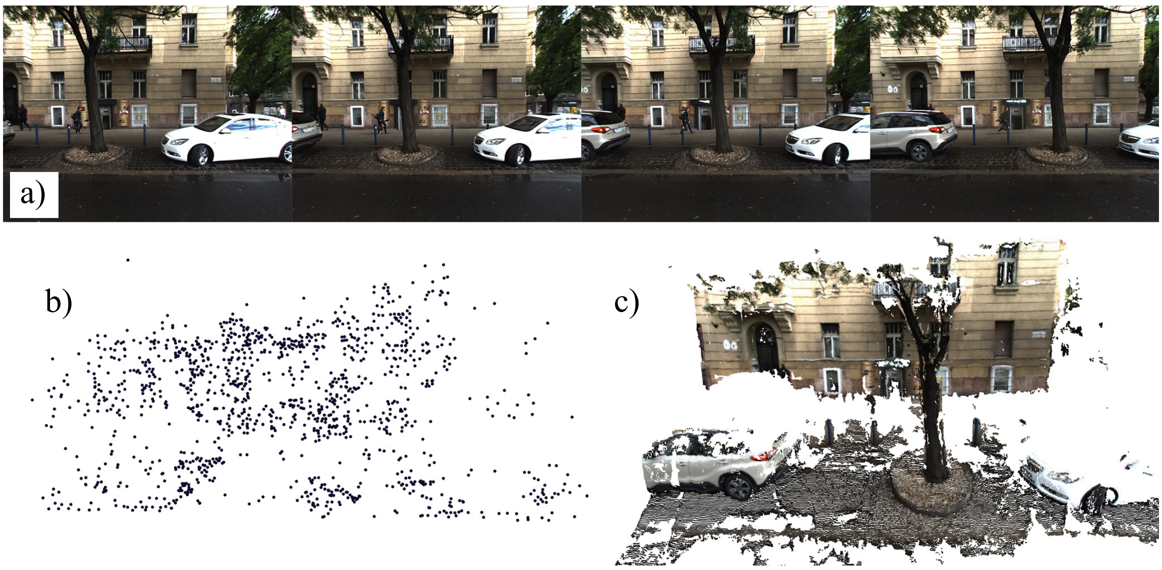


Figure 2. SfM point cloud generation (**a**) 4 from a set of 8 images to process. (**b**) Generated sparse point cloud (2041 points). (**c**) Densified point cloud (257,796 points).

3.1. Structure from Motion (SfM)

As a first step, we generate a 3D synthesized point cloud from consecutive camera frames reinterpreting the calibration task as a point cloud registration problem. To generate the synthesized SfM point clouds we modified and extended the SfM pipeline [21] from the OpenMVG (Url: https://github.com/openMVG/openMVG) library [26] which we describe in the following.

To generate the synthesized point cloud, we start capturing 1288×964 pixel sized image frames and we select a set of N ($N \geq 3$) images ($N = 8$ is used in this paper). We include the current frame among the N frames to be processed if there exists a global movement of at least $th_{move} \geq 10$ pixels between consecutive frames. After gathering the N frames, we execute the following steps:

1. Image rectification: First we rectify the selected images (Figure 2a) using the intrinsic camera parameters.
2. Semantic segmentation: Using a state-of-the-art instance level segmentation approach called Mask R-CNN [23] we label the dynamic objects (Figure 3a) such as vehicles and pedestrians on the rectified frames.
3. Consecutive frame correspondences: L_2 fast cascade matching is used to match the extracted SIFT feature points between the consecutive frames.

4. Structure from motion calculation: In this step we perform a SfM pipeline to generate a sparse point cloud (Figure 2b) from the selected image sequence and we also store the 2D image pixel coordinates (Figure 4) from all images that contributed to the calculation of the current 3D points. We assign unique IDs to all 2D interest points in all images and propagate the point IDs through the SfM pipeline along with the associated Mark R-CNN semantic segmentation labels into the generated point cloud. Thus, we will know which 2D points from which frames contributed to the calculation of each 3D point in the produced point cloud. (Figure 3b).
5. Transformation calculation: Based on point density we select M points (in this paper $M = 45$ and the point labels cannot be dynamic objects) from each processed frame and using the stored 3D-2D point correspondences (Figure 4a,b). We apply the EPnP [25] algorithm to estimate a transformation matrix T_1 of Equation (1) which is able to project the 3D points onto the corresponding 2D image pixels. Note that different T_1 matrices belong to every frame of the N images used to create the SfM point cloud. In practice, we calculate T_1 only for a single frame, selected via time stamp comparison with the actual Lidar point cloud. To validate if the estimated projection matrix is correct, we re-project the 3D points to the corresponding images (Figure 4c) and we calculate the re-projection error based on the stored 3D-2D point associations. Using our method, we measured 1.02 pixel error in average which we determined to be enough empirically to achieve a robust calibration result.
6. Sparse point cloud densification: Figure 2c demonstrates the result of the densification step by using the PatchMatch [27] algorithm from OpenMVS (Url: http://cdcseacave.github.io/openMVS) library without any modification. We use the densified point cloud (Figure 2c) and the estimated projection matrix in the next steps of the proposed approach.

Since our modified SfM pipeline can be performed on-the-fly, during the data capturing process we can periodically calculate the synthesized dense SfM point cloud from selected N consecutive frames. Thus, the method can efficiently operate on experimental test platforms, where the movements of the vehicle may cause some displacements of the temporarily fixed sensors. Note that even industrial sensor configurations may need re-calibration at specific time intervals, which is also straightforward using our technique, since to perform our whole pipeline takes about 2–3 min (using an Intel Core i7 Coffee Lake processor), and it is not required to stop driving or using any calibration targets.

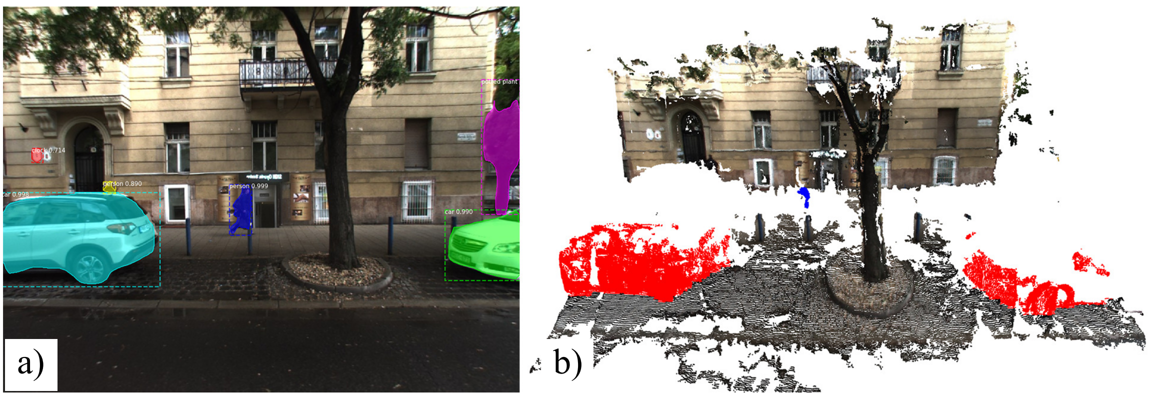


Figure 3. (**a**) Mask R-CNN [23] based instance level semantic segmentation. (**b**) Marking dynamic objects in the 3D SfM point clouds based on the 2D semantic segmentation (red: vehicle, blue: pedestrian).

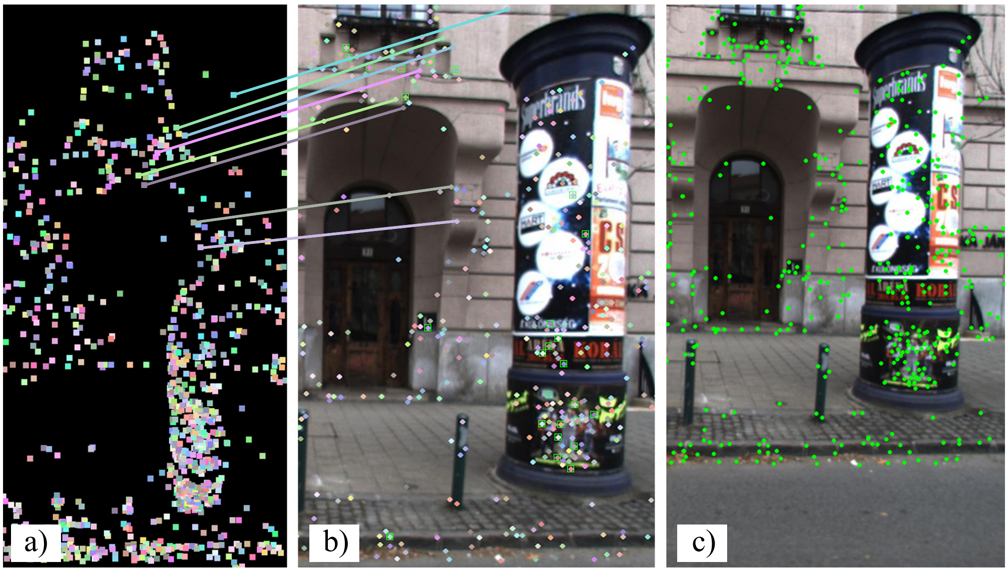


Figure 4. (**a**) Sparse cloud with each point assigned a unique color. (**b**) One frame showing color coded 2D points that contribute to the 3D point with the same color in (**a**), also showing example correspondences. (**c**) Re-projection error.

3.2. Point Cloud Registration

In the next phase of the proposed algorithm, we aim to transform the Lidar point cloud and the previously synthetized SfM point cloud (Section 3.1) to the same coordinate system.

Several point level registration methods can be found in the literature. Many of them are variants or extensions of the classical iterative ICP or the Normal Distributions Transform (NDT) [28] methods, which often fail if the initial translation is large between the point clouds. In addition, they are usually inefficient when the density characteristics of the two point clouds are quite different, furthermore the typical ring patterns of RMB Lidar data may mislead the registration process by finding irrelevant correspondences on the ground level instead of finding associations between the foreground regions [17]. In order to avoid such registration artifacts, we proposed an object level point cloud alignment approach [22] to estimate an initial translation and rotation between the point clouds.

We found that since the proposed SfM (Section 3.1) pipeline operates on image sequences of $N = 8$ frames, the global diameter of the generated SfM model is usually much smaller than the diameter of the recorded RMB Lidar point cloud. Therefore, to facilitate the registration, in the further processing steps we only consider the central segment of the Lidar measurement with a 15 m radius around the sensor position.

3.2.1. Object Detection

Instead of extracting local feature correspondences from the global scene, we detect and match various landmark objects in the SfM-based and the Lidar point clouds. In the next step based on the landmark object locations we estimate the transform which provides the best overlap between the two object sets.

As a preprocessing step, we first detect the dynamic object regions which can mislead the alignment, thus it is preferred to remove their regions before the matching step. On the one hand, based on the predicted semantic labels in the camera images (Section 3.1), we determine the camera-based 3D locations of the vehicles and pedestrians by projecting the 2D labels to the synthesized 3D SfM point cloud (see Figure 3). On the other hand, to eliminate the dynamic objects from the Lidar point cloud we adopt a state-of-the-art object detector called Point pillars [24] (see Figure 5b), which takes the full point cloud as input and provides bounding box candidates for vehicles and pedestrians. As a result of the filtering, during the subsequent alignment estimation we can rely on more compact and robustly detectable static scene objects such as columns, tree trunks and different street furniture elements.

After the dynamic object filtering step we also remove the ground/road regions by applying a spatially adaptive terrain filter [17], then we extract connected component blobs (objects) from both the synthesized SfM and the Lidar point clouds.

Euclidean point distance based clustering methods are often used to extract object candidates from point clouds, for example [29] used kD-tree and Octree space partition structures to efficiently find neighboring points during the clustering process. However these methods often yield invalid objects, either by merging nearby entities to large blobs, or by over-segmenting the scene and cutting the objects into several parts. Furthermore, building the partition tree data structure frame by frame causes noticeable computational overload. Börcs et al. [30] proposed an efficient 2D pillar-structure to robustly extract object candidates using a standard connected component algorithm on a two level hierarchical 2D occupancy grid, however that method erroneously merges multiple objects above the same 2D cell even if they have different elevation values (such as hanging tree crowns).

To handle the merging problem of the 2D grid based approaches alongside the height dimension we propose an efficient sparse voxel structure (SVS) to extract connected components called abstract objects from the point cloud data (Figure 6d,e). While using only the pillar structure, which is similar to a traditional 2D grid approach [30], the objects may be merged alongside the height dimension (see Figure 6d), in a second step focusing on the vertical point distribution we can split the erroneously merged regions using the voxel level representation of the SVS (see Figure 6e).

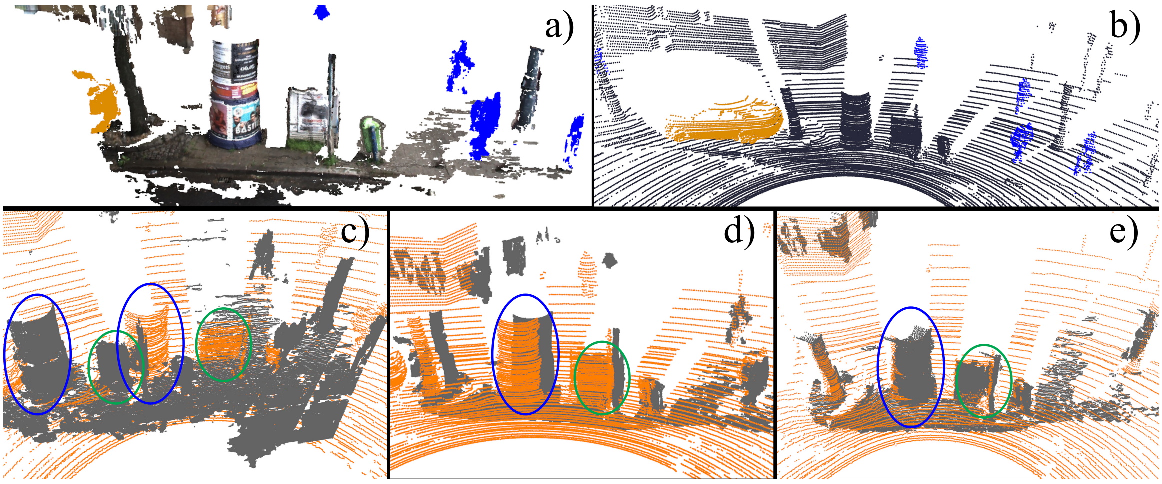


Figure 5. Demonstration of the object-based coarse alignment step (**a**) Mask R-CNN [23] based dynamic object filtering. (**b**) Point pillars [24] based dynamic object filtering. (**c**) Initial translation between the synthesized (dark grey) and the Lidar (orange) point cloud. (**d**) Object-based coarse alignment without dynamic object removal (**e**) Proposed object-based alignment after removing dynamic objects. Identically colored ellipses mark the corresponding objects.

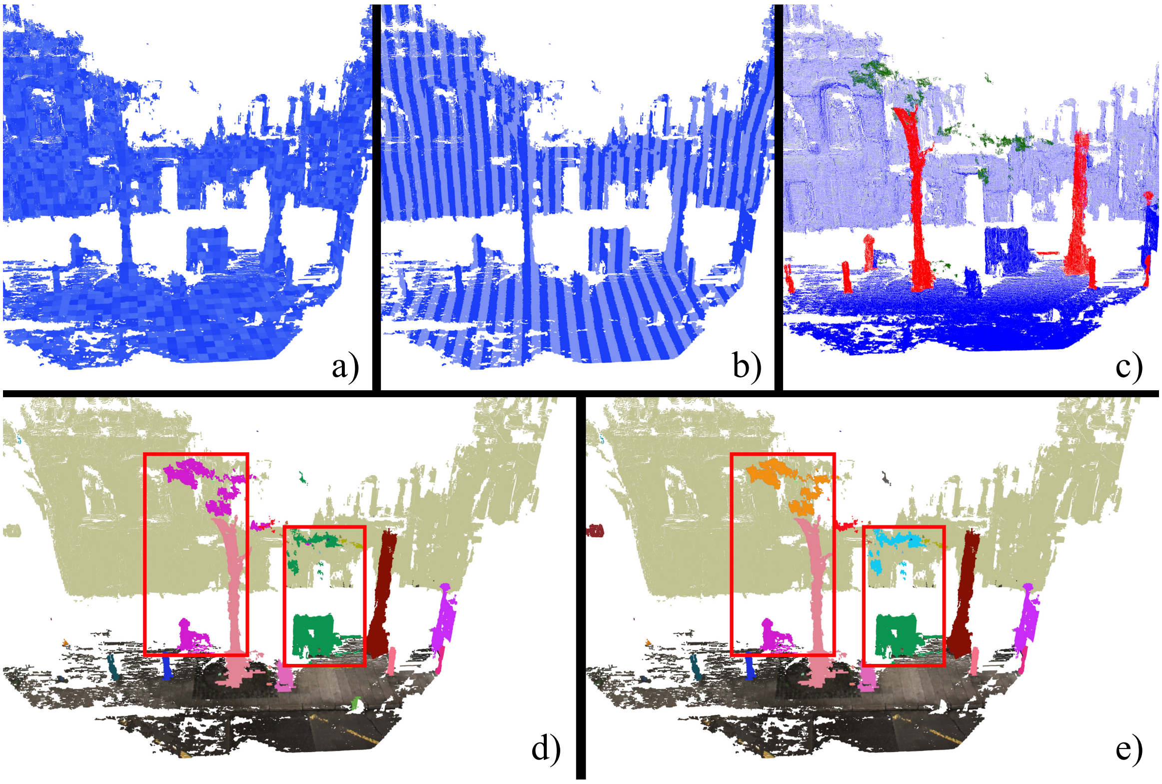


Figure 6. (**a**) Voxel representation of the proposed SVS. (**b**) Pillars representation of the proposed SVS. (**c**) Linearity and flatness features computed based on eigen value analysis. Blue: flat voxels, red: high linearity, green: scattered region. (**d**) Extracted objects from the point cloud using pillar representation (**e**) Extracted objects from the point cloud using voxel representation

The most elementary building block of the SVS is the voxel (see Figure 6a) which is a cube volume of the 3D space containing a local part of the given point cloud. Furthermore the vertically aligned voxels form a so called pillar structure (see Figure 6b) which is an extension of the voxels to be able to operate as a simple 2D grid similarly to [30]. Creating a fully dense voxel model based on the bounding box dimensions of the point cloud is quite inefficient, since in a typical urban environment only a few segments of the bounding space contain 3D points. To ensure memory efficiency for our SVS, we only create a voxel if we find any data point in the corresponding volume segment. In our case, the average size of the point cloud bounding boxes is around $30 \times 30 \times 5$ m, and we found that the optimal voxel resolution for object separation in a typical urban environment is 0.2 m [30]. While in this case a full voxel model would require the management of around 562,500 voxels, our proposed SVS usually contains less than 30,000 voxels, which yields a great memory and computational gain, while efficient neighborhood search can be maintained with dynamically created links within the new data structure. Note that we can find alternative sparse voxel structures in the literature such as [31,32], however those methods are based on Octree structures, and they were designed for computer graphics related tasks such as ray tracing. On the other hand our SVS fully exploits the fact that in an urban environment most of the objects are horizontally distributed along the road, while vertical object occlusions are significantly less frequent.

Next, we describe the object extraction process using our SVS. First, similarly to [30] we detect the ground regions using a spatially adaptive terrain filtering technique. Second, using the pillars structure of the SVS, we can quickly extract connected components (objects) by merging the neighboring pillars into object candidates. Third, considering the voxel level resolution of the SVS, we separate the erroneously merged blobs along the height dimension.

After the extraction of abstract objects, we assign different attributes to them, which will constraint the subsequent object matching steps (Section 3.2.2). We calculate the following three attributes for each object blob o:

1. Centroid ($o.c$): The center of mass of the local point cloud belonging to the object.
2. Bounding box ($o.b$): minimal and maximal point coordinate values along each dimension.
3. Shape category ($o.s$): we assign to each object the shape category linear, flat or scattered object (see Figure 6c).

The assigned shape category is based on a preliminary voxel level segmentation of the scene (see Figure 7). For each voxel, we determine a linearity and a flatness descriptor, by calculating the principal components and the eigen values $(\lambda_1 \geq \lambda_2 \geq \lambda_3)$ of the point set in the voxel, using the Principal Components Analysis (PCA) algorithm. By examining the variance in each direction, we can determine the linearity and flatness properties as follows [33]:

$$(1)\ Linearity = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$
$$(2)\ Flatness = 1 - \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$$

1. If λ_1 is large ($\lambda_1 > 0.7$) compared to the other two eigenvalues, it means that the included points lie on a line, so the voxel is part of a linear structure such as pole or wire.
2. If the total variance is defined by the two largest eigenvalues that means points lie on a plane.
3. If $Flatness > 0.6$ the voxel is classified as flat voxel. Typically wall segments and parts of larger objects are built from flat voxels.
4. Unstructured regions can also be detected using the third eigenvalues. If $\lambda_3 > 0.1$ that means point scattering is high in the region. Vegetation and noise can be detected efficiently using this property.

Note that the above threshold values should be empirically defined, since they depend on the characteristics of the given point cloud. Following the guidelines of [33], we fine-tuned the threshold parameters based on our experiments. Figure 6c demonstrates the voxel level classification based on the linearity and flatness values. Finally, for each extracted abstract object we determine its shape category by a majority voting of its included voxels.

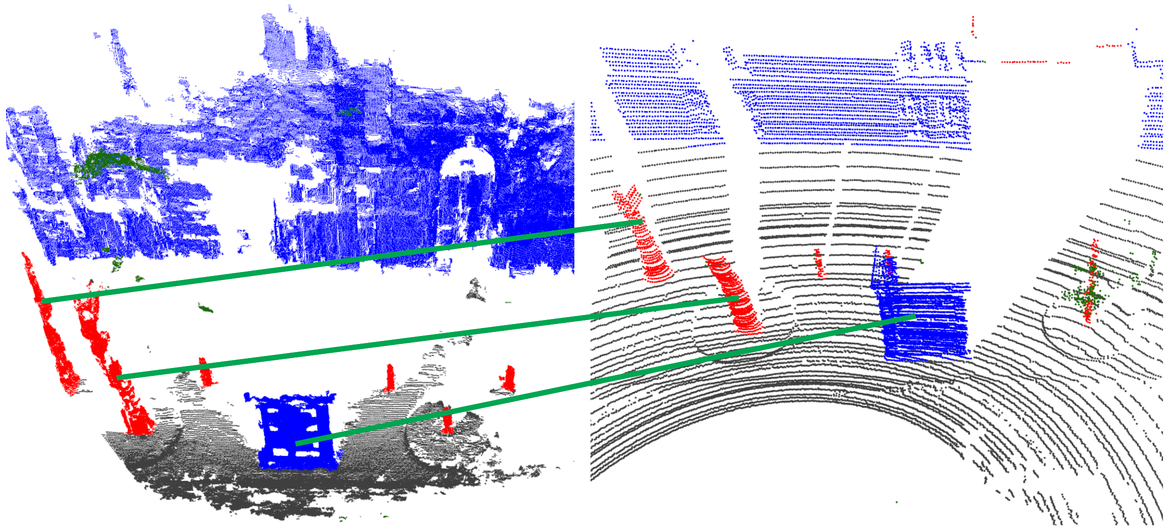
3.2.2. Object Based Alignment

At this point, our aim is to estimate a sufficient initial alignment between the synthesized SfM and Lidar point clouds based on the above detected abstract objects (Figure 7). In order to find the optimal matching, we use the extracted two sets of object centroids from the SfM-based and the Lidar point clouds, respectively, and we search for an optimal rigid transform, T_2 of Equation (1), between the two point sets by an iterative voting process in the Hough space [22]. We observed that the search for the translation component of the transform can be limited to a 15m radius search area (which is equal to the base size of the considered point cloud segments), while it is also crucial to roughly estimate the rotation component around the upwards axis. On the other hand, the other two rotation components are negligible, as the road elevation is usually quite short within the local road segments covered by the considered point clouds, thus the minor rotation errors caused by ignoring them can be corrected later through point level transformation refinement.

Estimation of the proper 3D translation vector (dx, dy, dz) among the three coordinate axes and the α rotation value around the upwards axis is equivalent to finding an optimal 3D rigid body transformation in the following form:

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

The above transformation estimation can be defined as a finite search problem by discretization of the possible parameter space into equal bins. Considering that our transformation has four degrees of freedom (3D translation vector and a rotation component around the upwards axis) we allocate a 4D accumulator array $A[\alpha, d_x, d_y, d_z]$.



a) SfM point cloud b) Lidar point cloud

Figure 7. Landmark correspondences between the detected objects in the SfM and Lidar point clouds. Assignment estimation is constrained by the computed Linearity and Flatness values. Red color represents linear voxels, blue color denotes flat structures and green marks noisy unstructured regions.

Let us denote the abstract object sets extracted from the SfM and the Lidar point clouds by $C1$ and $C2$ respectively. During the assignment, we attempt to match all possible $o1 \in C1$ and $o2 \in C2$ object pairs, however to speed up the process and increase its robustness we also introduce a compatibility measurement between the objects. As discussed in Section 3.2.1, during the object detection step we assigned a bounding box $(o.b)$ and a shape category $(o.s)$ parameter to each extracted abstract object. We say that $o1 \in C1$ and $o2 \in C2$ are compatible, if the side length ratios of their bounding boxes are between predefined values $(0.7 \leq \frac{o1.b_x}{o2.b_x}, \frac{o1.b_y}{o2.b_y}, \frac{o1.b_z}{o2.b_z} \leq 1.4)$ and their shape categories are the same $(o1.s = o2.s)$.

Algorithm 1 shows the pseudo code of the proposed object-based alignment algorithm. The approach iterates through all the compatible $o1 \in C1$ and $o2 \in C2$ object pairs and it rotates $o2$ with all the possible α values. We calculate the Euclidean distance between the rotated $o2'.c$ and the $o1.c$ centroid points in the following way:

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = o1.c - \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} o2.c \qquad (2)$$

Algorithm 1 Object-based point cloud alignment. Input: two point clouds $F1$ and $F2$, output: the estimated $TR = T_R$ rigid transformation between them. $Rot(\alpha)$ denotes a rotation matrix around the upwards axis.

```
procedure ALIGNMENT(F1, F2, TR)
    C1 ← ObjectDetect(F1)
    C2 ← ObjectDetect(F2)
    Initialize 4D accumulator array A
    for all o1 ∈ C1 do
        for all o2 ∈ C2 do
            if iscompatible(o1, o2) then
                for α ∈ [0, 359] do
                    o2'.c ← Rot(α) * o2.c
                    (dx, dy, dz) ← o1.c − o2'.c
                    A[α, dx, dy, dz] ← A[α, dx, dy, dz] + 1
                end for
            end if
        end for
    end for
    α, dx, dy ← FindMaximum(A)
    F1, T2 ← TransformCloud(F1, α, dx, dy, dz)
    F1, T3 ← ICP(F1, F2)
    TR ← T3 * T2
end procedure
```

During a given iteration step the method addresses the accumulator array with the actual α, it calculates the corresponding dx, dy and dz translation components using Equation (2), then it increases the evidence of the transformation defined by the (α, dx, dy, dz) quadruple. At the end of the iterative process, the maximal vote value in the accumulator array determines the best transformation parameters, i.e., the best α rotation and the dx, dy and dz translation components. Thereafter, using the estimated transformation matrix the Lidar point cloud is transformed to the coordinate system of the synthesized SfM model.

The above point cloud registration process is demonstrated in Figure 5. Note that as we mentioned in the beginning of Section 3.2.1, we aimed to restrict the object matching step to static landmark objects, therefore we first removed the vehicle and pedestrian regions from the 3D scene (see Figure 5a,b). As Figure 5c–e confirm, this step may have a direct impact on the quality of the object-based alignment algorithm: subfigure (c) shows the initial alignment of the Lidar and the SfM point clouds with a large translation error, subfigure (d) shows the result of the proposed coarse alignment algorithm without eliminating the dynamic objects from the point clouds in advance, while subfigure (e) demonstrates the output of the proposed complete algorithm which provides a significantly improved result.

To eliminate the discretization error of the object-based alignment we can refine the estimation of the rigid transformation using a standard ICP [28] method which yields a T_3 matrix of Equation (1), so that the unified rigid transform between the SfM and Lidar point clouds is represented as $T_R = T_2 \cdot T_3$.

3.3. Control Curve Based Refinement

In the previous section, we estimated an optimal rigid transformation $T_R = T_2 \cdot T_3$ between the synthesized SfM model and the Lidar point cloud, which is composed of a global translation and a rotational component. However, during the SfM pipeline various artifacts may occur such as deformations due to invalid depth calculation, effects which can significantly distort the geometry of the observed 3D scene. On the other hand, due to the vehicle motion, the Lidar point clouds may also suffer from shape distortion as detailed in the Introduction. To handle the local point cloud deformation problems during the point cloud registration process, we propose a control curve based refinement method, that calculates the non-rigid $\mathcal{T}_\star$ transform component of Equation (1) (see Figure 8).

To estimate the local shape deformations of the point clouds, we use the extracted object centroids as control points, and we fit separate NURBS curves to the control points of the SfM and the Lidar point clouds, respectively. As shown in Figure 8a, after the rigid body transformation based alignment there are point cloud regions which fit quite well, while in other (distorted) segments we observe significant local alignment errors.

For this reason, we introduce a non-rigid curve-based point cloud registration refinement step (see Figure 9):

1. Using the extracted object centers as control points, we fit NURBS curves to the SfM and the Lidar point clouds. In Figure 9 red dots mark the control points and bluish curves demonstrate the NURBS curves. Based on the assignment results of the object-based alignment step (Section3.2.2), we can utilize the correspondences between the control points to obtain the coherent objects pairs.
2. Let us assume that based on the object alignment, the curve segments between O_1 and O_2 belong together. Between each neighboring control point pair, we subdivide the curve segment into equal bins, so that we assign the first bin of the Lidar curve segment to the first bin of the SfM curve segment.
3. In the next step we assign each point of the Lidar point cloud to the closest bin of the Lidar based NURBS curve.
4. Taking the advantage that a NURBS curve is locally controllable, we can transform the local parts of the Lidar point cloud separately. We calculate a translation between each corresponding bins of the SfM and Lidar based curve segments by subtracting the coordinates of the Lidar bins from the corresponding SfM bins. Since we assigned each Lidar point to one of the curve bins, we can transform these points using the translation vector of the given curve segment, which points from the given Lidar bin to the corresponding SfM bin. Since each part of the Lidar point cloud is transformed according to the corresponding curve segment, this process implements a non-rigid body transformation.

Figure 8b show the results of the curve-based registration refinement step, demonstrating a significant improvement in the initially misaligned point cloud segments.

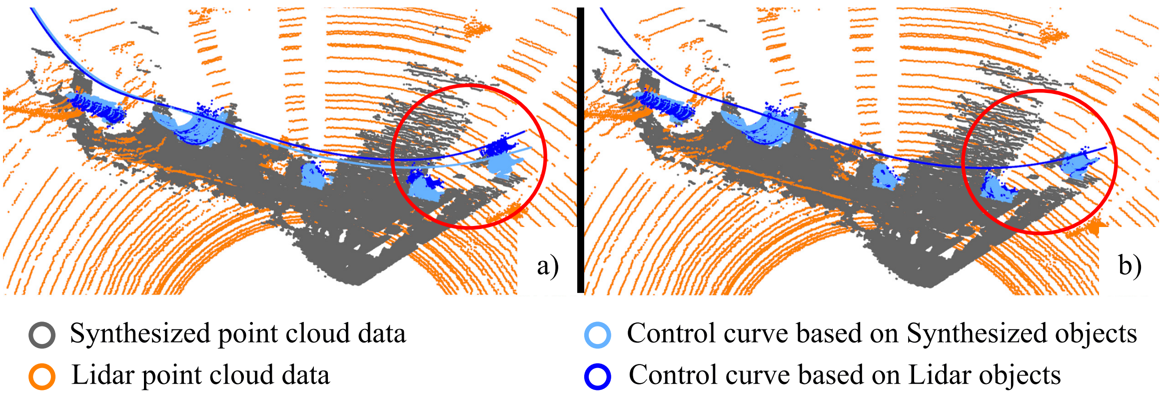


Figure 8. Demonstration of the control curve-based registration refinement step. Image (**a**): output of rigid body transform based registration (T_R) with local registration artifacts; Image (**b**) corrections via registration refinement ($\mathcal{T}_\star$)

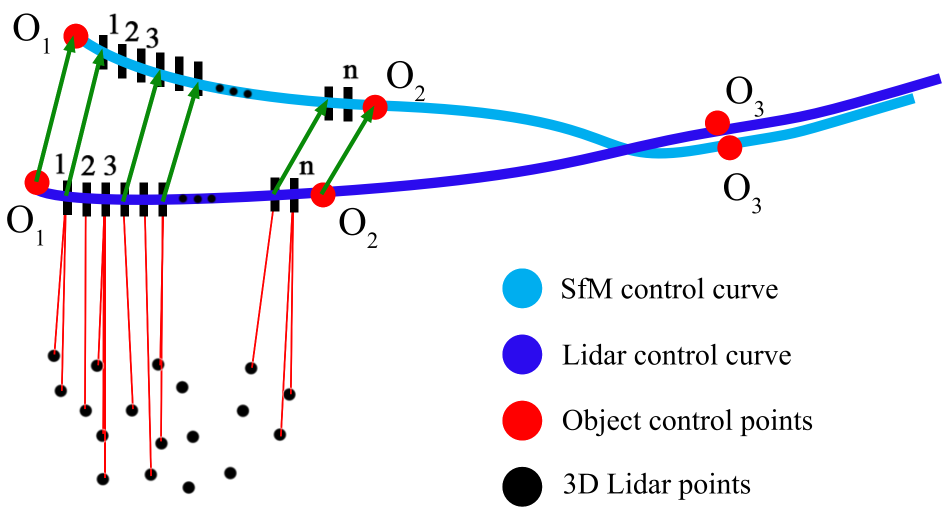


Figure 9. Demonstration of the curve-based refinement step.

3.4. 3D Point Projection Calculation

Our final goal is to estimate a transformation $\hat{T}$ which is able to project the 3D points from the original coordinate system of the Lidar sensor directly to the corresponding 2D image pixels. Figure 10 demonstrates the proposed workflow of the point cloud projection. From $N = 8$ consecutive image frames using an SfM pipeline we estimate a dense SfM point cloud, and we also calculate and store a unique T_1 projection transformation between the generated dense SFM point cloud and the original image frames. Thereafter we take the Lidar point cloud with the closest time stamp to the first frame of the actual N-frame sequence and we estimate a transformation which is able to transform the Lidar point cloud to the coordinate system of the SfM point cloud. The estimated transformation consists of a rigid body component composed by an object-based (T_2) and a point-based (ICP) term (T_3), as well as a NURBS-based non-rigid transform component ($\mathcal{T}_\star$).

Since the non-rigid body transform $\mathcal{T}_\star$ cannot be defined in a closed form as a simple matrix multiplication, we calculate an approximate direct transformation between the original Lidar point cloud domain and the image pixels for preserving computational efficiency of the data fusion step. For this reason, with applying the above estimated transformation components (T_1, T_2, T_3 and $\mathcal{T}_\star$) we project the centroid points of the extracted Lidar objects onto the camera images and we store the 3D-2D correspondences. In the next step, upon the available 3D-2D point correspondences we estimate a single homogeneous linear transformation $\hat{T}$ which can be used to project the raw Lidar points directly to the image domain (see Figure 11). $\hat{T}$ is calculated by the OpenCV *solvePnP* function, which is based on the Levenberg-Marquardt optimization method [34]. As a final result, Figure 11 shows the projection of the Lidar data to the image domain.

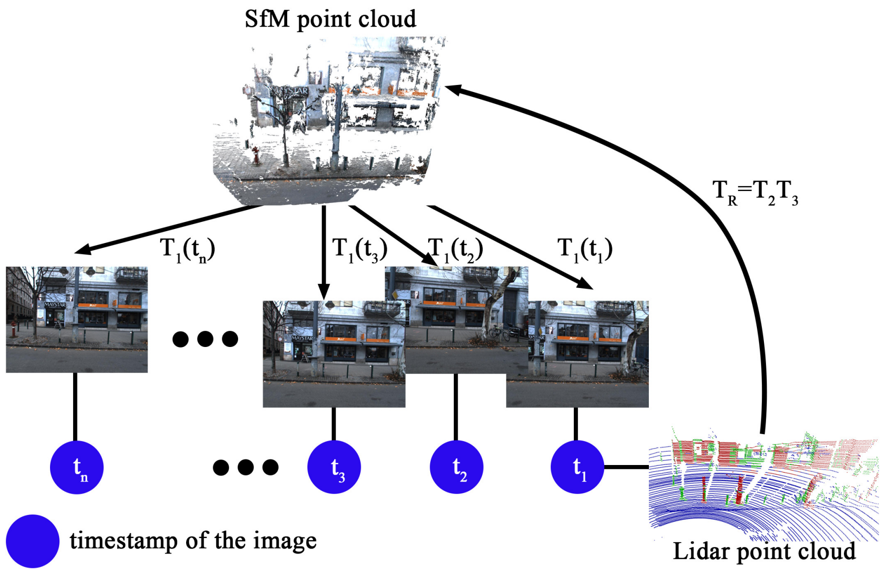


Figure 10. Workflow of the projection calculation process.

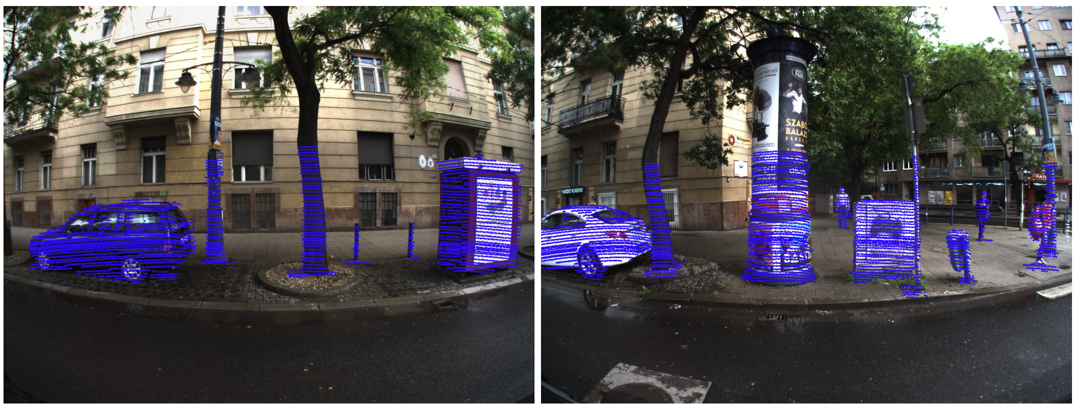


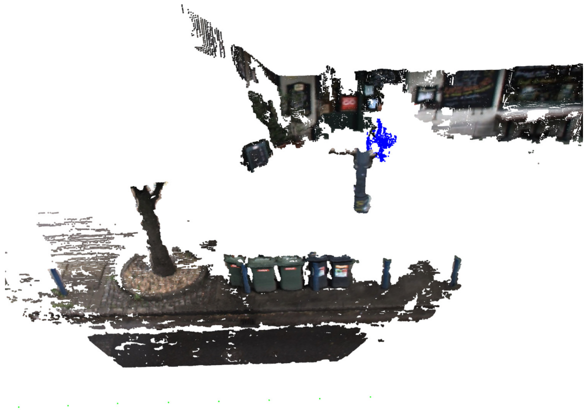
Figure 11. Qualitative point cloud projection results onto the image domain based on the proposed fully automatic, target-less camera-Lidar calibration approach. Blue color denotes the projected 3D points which belong to the object candidates detected during the coarse alignment process.

4. Experiments and Results

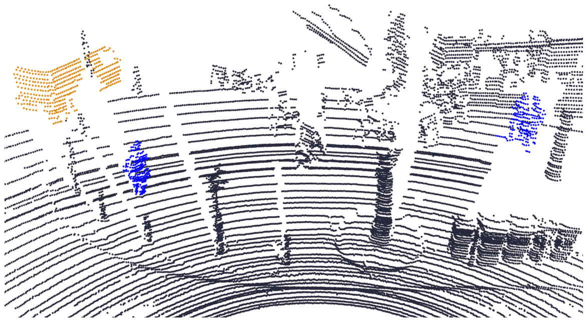
To evaluate our proposed target-less and fully automatic self-calibration method we compared it to two state-of-the-art target-less techniques [4,5], and an offline target-based method [2]. We quantitatively evaluated the considered methods by measuring the magnitude and standard deviation of the pixel level projection error values both in the x and y directions along the image axes (Table 1).

We collected test data from overall 10 km long road segments in different urban scenes such as boulevards, main roads, narrow streets and large crossroads. To take into account the daily traffic changes, we recorded measurements both in rush hours in heavy traffic, and outside the peak hours as well. Since using the rotating multi-beam Lidar technology, the speed of the moving platform influences the shape of the recorded point cloud, we separately evaluated the results for measurements captured by slow and fast vehicle motion, respectively. Therefore we defined two test sets: set Slow contains sensor data captured at speed level between 5 and 30 km/h, while set Fast includes test data captured at speed above 30 km/h.

Figures 12 and 13 visually demonstrate the steps of the proposed workflow.

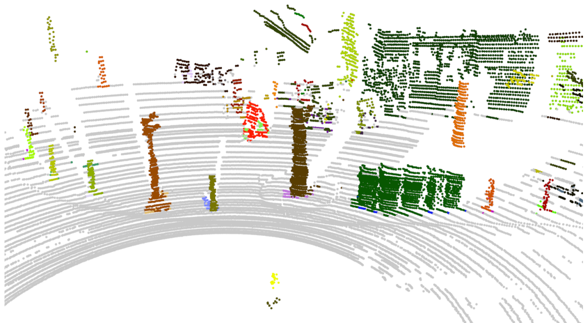


(**a**) Pedestrian detection (blue) in the SfM cloud [23]



(**b**) Pedestrian (blue) and vehicle (orange) detection in the Lidar cloud [24]

(**c**) Abstract object extraction in the SfM cloud

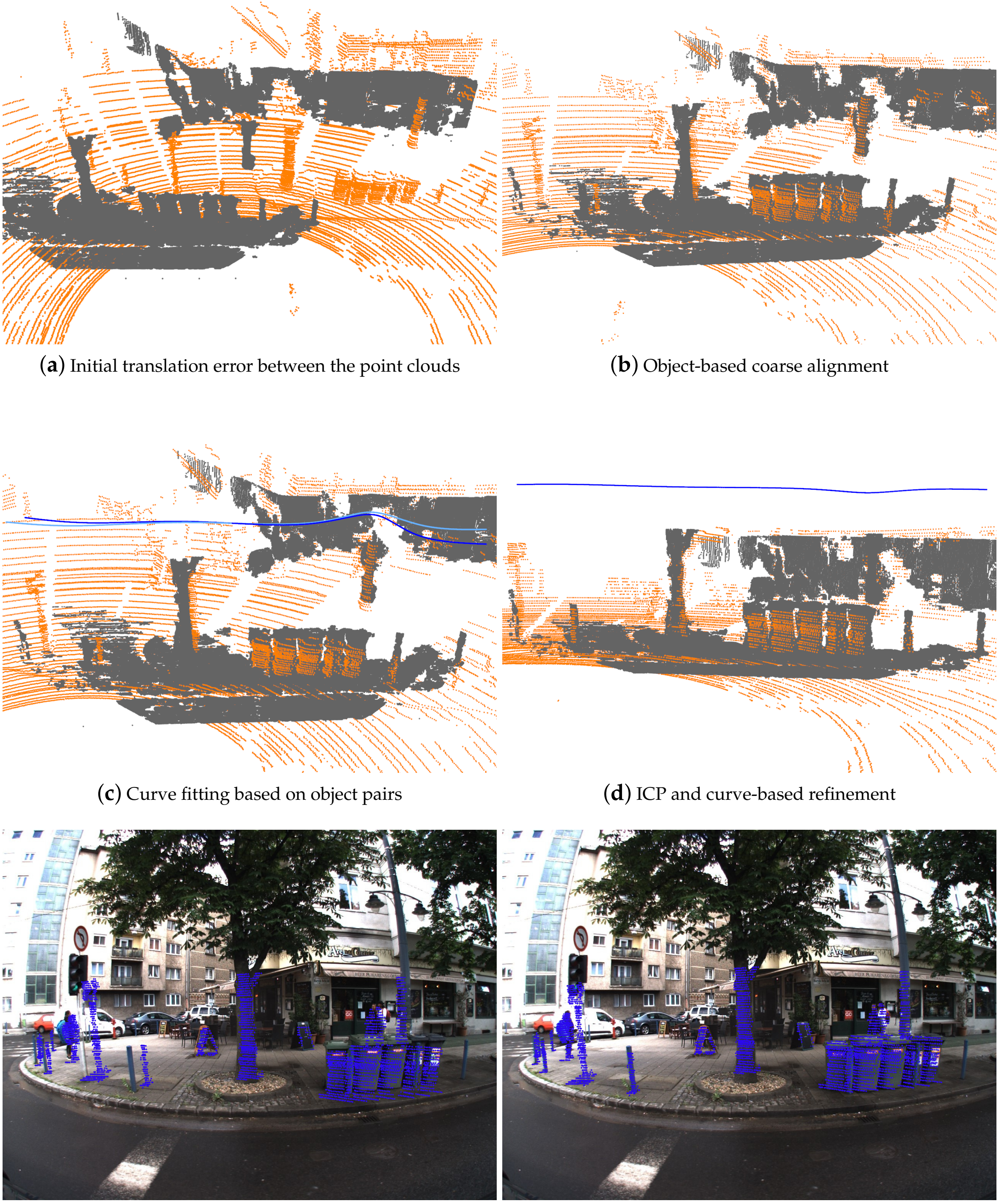


(**d**) Abstract object extraction in the Lidar cloud

Figure 12. Demonstration of intermediate steps of the proposed method in a selected scene. Results of the (**a**,**b**) dynamic object filtering and (**c**,**d**) 3D abstract object extraction steps, which are used as input of the proposed object-based alignment technique (Section 3.2.2).

4.1. Quantitative Evaluation and Comparison

We measured the projection error of the proposed and the reference methods by projecting the 3D Lidar points onto the 2D image pixels, which was followed by visual verification. Before starting the test, we carefully calibrated the Lidar and camera sensors using the offline target-based reference method [2], which served as a baseline calibration for the whole experiment. During our 10 km long test drive, we chose 200 different key frames of the recorded measurement sequence, and by each key frame, based on the actual a Lidar point cloud and $N = 8$ consecutive camera images, we executed new calibration processes in parallel by our proposed approach and by the remaining automatic reference techniques [4,5]. We paid attention to collect these key frames from diverse scenarios (main roads, narrow streets, etc.), and we also noticed the corresponding vehicle speed values to assign each key frame either to the Slow or to the Fast test set. We evaluated the performance of all considered methods by each key frame (i.e., after each re-calibration). To calculate the pixel projection error we manually selected 10 well defined feature points (e.g., corners) from different regions of the 3D point cloud, and using the calibration results of the different methods we projected these 3D feature points to the image domain. Finally we measured the pixel errors versus the true positions of the corresponding 2D feature points positions detected (and manually verified) on the images.



(**a**) Initial translation error between the point clouds (**b**) Object-based coarse alignment

(**c**) Curve fitting based on object pairs (**d**) ICP and curve-based refinement

(**e**) Projection result after the coarse alignment (**f**) Projection result after the refinement

Figure 13. Qualitative results of the proposed registration and projection steps.

As shown in Table 1, both in the Slow and Fast test sets our proposed approach notably outperforms the two target-less reference methods [4,5] in mean pixel error, and it also exhibits lower error deviation which indicates a more robust performance. Although in the Slow test set, in terms of accuracy, the proposed approach is slightly outperformed by the considered offline target-based calibration technique [2], it should be emphasized that the operational circumstances are quite different there. On one hand, according to our experiences the calibration of the camera-Lidar system using [2] takes more than two hours with several manual interaction steps, furthermore if the sensors are slightly displaced during the drive due to vibration, one must stop the measuring platform and re-calibrate the system offline. On the other hand, our proposed approach is able to calibrate the camera and Lidar on-the-fly and in a fully automatic way, even during driving without stopping the car.

We can also observe in Table 1, that in the Fast test set our proposed method can reach or even surpass the accuracy of the target-based method, since we can adaptively handle the vehicle speed dependent shape deformations of the recorded RMB Lidar point clouds. By increasing the speed of the scanning platform, we can observe that the shape of the obtained Lidar point cloud becomes distorted, and gets stretched in the direction of the movement. Since the offline, target-based method cannot re-calibrate the system during the drive, its accuracy may decline when the car accelerates. On the contrary, our proposed method is able to re-calibrate the system at different speed levels, and taking the advantage of the non-rigid transformation component, it can be better adapted to the non-linear shape distortions of the point cloud.

Table 1. Performance comparison of the target-less proposed method with two state-of-the-art target-less reference methods and with a target-based (supervised) reference technique. Notes: * Test set names Slow and Fast refer to the speed of the data acquisition platform.

Set *	Method	x-Error (pixel)		y-Error (pixel)	
		Avg.	Dev.	Avg.	Dev.
Slow	Target-based ref. [2]	1.13	0.27	1.75	0.37
	Bearing angle [4]	4.56	2.15	4.58	1.74
	Line-based [5]	3.36	1.45	3.47	0.98
	Proposed	2.58	0.73	2.82	0.88
Fast	Target-based ref. [2]	3.04	0.74	2.74	0.41
	Bearing angle [4]	4.87	2.46	4.42	1.91
	Line-based [5]	5.01	1.93	4.01	1.49
	Proposed	2.84	0.95	3.14	0.82

To quantify the improvements of the consecutive calibration steps of the proposed method, we also compared the pixel level projection errors at three different stages of our algorithm pipeline. In Table 2, the first row shows the results of using the object-based coarse alignment (OBA) step on the raw Lidar frames; the second row refers to using OBA with preliminary dynamic object filtering (DOF), finally the third row displays the error parameters obtained by the complete workflow including the ICP and curve-based refinement steps. Since applying DOF can prevent us from considering several invalid object matches, we can see a large improvement in the robustness between the first and second stages, furthermore by exploiting the point level ICP and curve-based refinement steps (third stage), the projection accuracy can be significantly upgraded.

Table 2. Performance evaluation of the proposed method at different calibration steps. First we measure the projection error using only the object-based coarse alignment, then we extended the method with the dynamic object filtering based on Mask R-CNN and PointPillars techniques, and finally we measure the impact of the ICP and the proposed curve-based refinement.

Step of Our Method	x-Error (pixel)		y-Error (pixel)	
	Avg.	Dev.	Avg.	Dev.
Object-based alignment (OBA) on raw frames	4.78	2.26	5.21	2.47
OBA with preliminary dynamic object filtering	3.46	0.85	4.47	1.24
Final result also using ICP and NURBS-based refinement	2.58	0.73	2.83	0.88

4.2. Robustness Analysis of the Proposed Method

Since our proposed method is able to re-calibrate the sensors periodically during the movement of the sensor platform, it is important to measure the robustness of the approach. We recall here that during the proposed algorithm pipeline, we already attempt to internally estimate the quality of the calibration: First following the SfM point cloud synthesis we re-project the generated point cloud to the original image pixels, and only continue the process if the re-projection error is smaller than 2 pixels. Second, following the estimation of both the rigid, and the NURBS-based non-rigid transformation components, we calculate the point level Hausdorff distance between the registered SfM and Lidar point clouds, and only accept the new calibration if the distance is lower than 5 cm. In summary, if these threshold-based constraints are not satisfied we reject to apply the current transformation estimation and we start a new calibration process.

In Section 4.1, we manually measured the errors based on 10 selected feature points in each Lidar frame. Due to the lack of availability of a Ground Truth (GT) transform [35], there is no straightforward way to extend this validation step for all points of the Lidar point cloud. However, we have performed an additional experiment to quantitatively measure the reliability and repeatability of the proposed approach. Although as discussed in Section 4.1, the target-based reference (TBR) method [2] itself may also suffer betweem one and three pixel registration errors, we observed that these errors can usually be considered constant when driving with an approximately standard average speed. Exploiting this fact, we used here TBR as a baseline technique, and we re-calibrated with 500 randomly selected seed frames the Lidar and the camera sensors using the proposed approach.

By each re-calibration step, we calculated the average point projection differences in pixels between our actual calibration and the TBR registration considering all points of the Lidar point cloud. Figure 14 shows the differences measured in pixels in the x and y directions over the subsequent calibration trials. Based on this experiment the proposed approach proved to be quite stable again, as the measured differences range between 1.2 and 1.8 pixels in the x direction, and 1.0 and 1.4 pixels in the y direction, without significant outlier values.

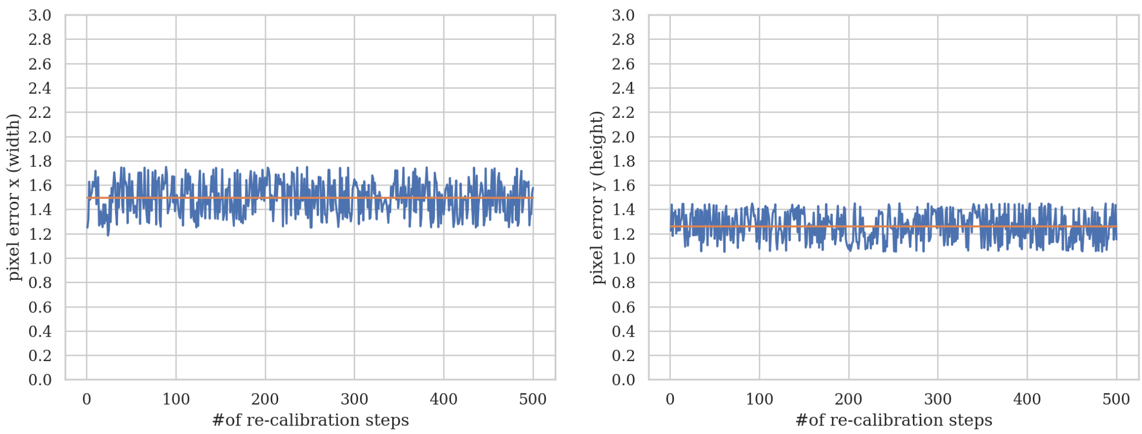


Figure 14. Robustness analysis of the proposed method.

We also examined the impact of the roughness of discretization in the accumulator space in the voting process during the object-based registration. If we increase the step size of the α rotation parameter around the upward axis, or if we divide the space of the possible translation components dx, dy and dz into larger bins, we lose accuracy, however the algorithm may show an increased robustness. On the contrary, if we choose a smaller rotation step size and we increase the resolution of the possible translation space we can reach more accurate results; however, we must also expect an increasing number of unsuccessful registration attempts, as the maximum value in the voting array may become ambiguous. Based on our experiments we used an optimal rotation step size of 0.25 degrees for α, and an optimal resolution of 0.2 m for the translation components dx, dy and dz. These experiments are consistent with our previous observations regarding an optimal cell resolution in the case of RMB Lidar sensors [36], keeping in mind that the accuracy can be enhanced in the subsequent point level ICP step.

There can be situations when the SfM pipeline cannot produce suitable synthesized point clouds. For example, the strong sunlight may cause burnout in the images limiting the traceable features during the SfM process. However, the proposed approach is able to periodically repeat the calibration pipeline and we should only update the calibration when the current estimation improves the previously used one in terms of Hausdorff error between the SfM and Lidar point clouds. Since the object-based alignment step of the proposed approach greatly depends on the landmark candidates (objects) used for registration, we preliminary analyze the scene using the computed linearity and flatness values and the extracted bounding boxes, and we only start the calibration pipeline if the number of the proper object candidates (column shaped objects such as traffic signs, tree trunks and poles) is more than three. Typically in main roads and larger crossroads the scenes contain several column shaped landmark objects, where the proposed self-calibration algorithm works more robustly and accurately.

4.3. Implementation Details and Running Time

We implemented the proposed method using C++ and OpenGL. An NVIDIA GTX1080Ti GPU with 11 GB RAM was used to speed up the calculations. The running time of the entire calibration pipeline is less than 3 min. For data acquisition we used 1288×964 resolution RGB cameras and a Velodyne HDL-64E RMB Lidar sensor.

5. Conclusions

This paper proposed a new target-less, automatic camera-Lidar calibration approach which can be performed on-the-fly, i.e., during the driving without stopping the scanning platform. Our method does not rely on any external sensors such as GNSS and IMU and we do not use hardware-based time synchronization between the Lidar and the camera. The method can periodically re-calibrate the sensors to handle small sensor displacements and it also adapts to the distortion of the RMB Lidar point cloud.

We quantitatively evaluated and compared our method to state-of-the-art target-less techniques and we proved that our approach surpasses them. Furthermore, we have shown that in some cases the proposed method even reaches the accuracy of the considered target-based reference method.

Author Contributions: All authors contributed to the conceptualization, methodology, and writing—editing the paper. Further specific contributions: software, B.N.; data acquisition B.N.; validation and comparative state-of-the-art analysis, B.N.; supervision and funding acquisition C.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research, Development and Innovation Fund, under grant number K-120233; by the European Union and the Hungarian Government from the project 'Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications' under grant number EFOP-3.6.2-16-2017-00013 (B.N.), and from the project 'Intensification of the activities of HU-MATHS-IN—Hungarian Service Network of Mathematics for Industry and Innovation' under grant number EFOP-3.6.2-16-2017-00015 (C.B.); by the ÚNKP-19-3 New National Excellence Program of the Ministry of Human Capacities (B.N.); and by the Michelberger Master Award of the Hungarian Academy of Engineering (C.B.).

Acknowledgments: The authors thank László Tizedes for his software development work on SZTAKI's autonomous vehicle platform, and for his help in test data acquisition; Zoltán Pusztai for help in using the referred target-based semi-automatic calibration technique; and Prof. Tamás Szirányi for providing remarks and advices regarding the project. Special thanks to Levente Kovács who gave important advises and technical help.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

SfM	Structure from Motion
RMB	rotating multi-beam

NURBS	Non-uniform rational B-spline
SVS	Sparse Voxel Structure
ICP	Iterative Closest Point
NDT	Normal Distributions Transform
IMU	Inertial Measurement Units
GNSS	Global Navigation Satellite System

References

1. Pandey, G.; McBride, J.R.; Savarese, S.; Eustice, R.M. Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information. *J. Field Robot.* **2015**, *32*, 696–722. [CrossRef]
2. Pusztai, Z.; Eichhardt, I.; Hajder, L. Accurate Calibration of Multi-LiDAR-Multi-Camera Systems. *Sensors* **2018**, *18*, 2139. [CrossRef] [PubMed]
3. Iyer, G.; Ram, R.K.; Murthy, J.K.; Krishna, K.M. CalibNet: Geometrically Supervised Extrinsic Calibration using 3D Spatial Transformer Networks. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
4. Scaramuzza, D.; Harati, A.; Siegwart, R. Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 4164–4169.
5. Moghadam, P.; Bosse, M.; Zlot, R. Line-based extrinsic calibration of range and image sensors. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 3685–3691.
6. Nagy, B.; Kovács, L.; Benedek, C. Online Targetless End-to-End Camera-LIDAR Self-calibration. In Proceedings of the International Conference on Machine Vision Applications (MVA), Tokyo, Japan, 27–31 May 2019.
7. Nagy, B.; Kovács, L.; Benedek, C. SFM and Semantic Information Based Online Targetless Camera-Lidar Self-Calibration. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–29 September 2019; IEEE: Taipei, Taiwan, 2019.
8. Geiger, A.; Moosmann, F.; Car, O.; Schuster, B. Automatic camera and range sensor calibration using a single shot. In Proceedings of the IEEE International Conference on Robotics and Automation, St. Paul, MN, USA, 14–18 May 2012; pp. 3936–3943.
9. Alismail, H.; Baker, L.; Browning, B. Automatic calibration of a range sensor and camera system. In Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, Zurich, Switzerland, 13–15 October 2012; pp. 286–292.
10. Park, Y.; Yun, S.; Won, C.S.; Cho, K.; Um, K.; Sim, S. Calibration between Color Camera and 3D LIDAR Instruments with a Polygonal Planar Board. *Sensors* **2014**, *14*, 5333–5353. [CrossRef] [PubMed]
11. Velas, M.; Spanel, M.; Materna, Z.; Herout, A. Calibration of RGB Camera With Velodyne LiDAR. In *WSCG 2014 Communication Papers Proceedings;* Union Agency: Plzeň, Czech Republic, 2014, pp. 135–144.
12. Rodriguez F., S.A.; Fremont, V.; Bonnifait, P. Extrinsic calibration between a multi-layer lidar and a camera. In Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Seoul, Korea, 20–22 August 2008; pp. 214–219.
13. Shiu, Y.C.; Ahmad, S. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AXDXB. *IEEE Trans. Robot. Autom.* **1989**, *5*, 16–29. [CrossRef]
14. Huang, K.; Stachniss, C. Extrinsic multi-sensor calibration for mobile robots using the Gauss Helmert model. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1490–1496.
15. Strobl, K.H.; Hirzinger, G. Optimal hand-eye calibration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China, 9–15 October 2006; pp. 4647–4653.
16. Shi, C.; Huang, K.; Yu, Q.; Xiao, J.; Lu, H.; Xie, C. Extrinsic Calibration and Odometry for Camera-LiDAR Systems. *IEEE Access* **2019**, *7*, 120106–120116. [CrossRef]
17. Józsa, O.; Börcs, A.; Benedek, C. Towards 4D Virtual City Reconstruction From Lidar Point Cloud Sequences. In Proceedings of the ISPRS Workshop on 3D Virtual City Modeling, Istanbul, Turkey, 27–29 November 2013; Volume II-3/W1, pp. 15–20.

18. Wang, R.; Ferrie, F.P.; Macfarlane, J. Automatic registration of mobile LiDAR and spherical panoramas. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, USA, 16–21 June 2012; pp. 33–40.
19. Napier, A.; Corke, P.; Newman, P. Cross-calibration of push-broom 2D LIDARs and cameras in natural scenes. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 3679–3684.
20. Schneider, N.; Piewak, F.; Stiller, C.; Franke, U. RegNet: Multimodal sensor registration using deep neural networks. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 1803–1810.
21. Moulon, P.; Monasse, P.; Marlet, R. Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 3248–3255. [CrossRef]
22. Nagy, B.; Benedek, C. Real-time point cloud alignment for vehicle localization in a high resolution 3D map. In Proceedings of the Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving (ECCV), Munich, Germany, 14 September 2018.
23. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988. [CrossRef]
24. Lang, A.H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; Beijbom, O. PointPillars: Fast Encoders for Object Detection From Point Clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
25. Lepetit, V.; Moreno-Noguer, F.; Fua, P. EPnP: An Accurate O(n) Solution to the PnP Problem. *Int. J. Comput. Vis.* **2008**, *81*, 155–166. [CrossRef]
26. Moulon, P.; Monasse, P.; Perrot, R.; Marlet, R. OpenMVG: Open Multiple View Geometry. In Proceedings of the Workshop on Reproducible Research in Pattern Recognition, Cancún, Mexico, 4 December 2016; pp. 60–74.
27. Barnes, C.; Shechtman, E.; Finkelstein, A.; Goldman, D.B. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.* **2009**, *28*, 24. [CrossRef]
28. Magnusson, M.; Nuchter, A.; Lorken, C.; Lilienthal, A.J.; Hertzberg, J. Evaluation of 3D registration reliability and speed—A comparison of ICP and NDT. In Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3907–3912. [CrossRef]
29. Rusu, R.; Cousins, S. 3D is here: Point cloud library (PCL). In Proceedings of the International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011.
30. Börcs, A.; Nagy, B.; Benedek, C. 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*; Springer: Zürich, Switzerland, 2015; Volume 8926, pp. 628–639.
31. Samuli, L.; Tero, K. Efficient Sparse Voxel Octrees. *IEEE Trans. Vis. Comput. Graph.* **2010**, *17*, 1048–59. [CrossRef]
32. Viktor, K.; Erik, S.; Ulf, A. High Resolution Sparse Voxel DAGs. *ACM Trans. Graph.* **2013**, *32*, 101:1–101:13. [CrossRef]
33. Matti, L.; Anttoni, J.; Juha, H.; Jouko, L.; Harri, K.; Antero, K.; Eetu, P.; Hannu, H. Object Classification and Recognition From Mobile Laser Scanning Point Clouds in a Road Environment. *IEEE Trans. Geosci. Remote Sens.* **2015**. [CrossRef]
34. Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math.* **1944**, *2*, 164–168. [CrossRef]
35. Kim, E.S.; Park, S.Y. Extrinsic Calibration between Camera and LiDAR Sensors by Matching Multiple 3D Planes. *Sensors* **2019**, *20*, 52. [CrossRef] [PubMed]
36. Börcs, A.; Nagy, B.; Baticz, M.; Benedek, C. A Model-based Approach for Fast Vehicle Detection in Continuously Streamed Urban LIDAR Point Clouds. In *Workshop on Scene Understanding for Autonomous Systems at ACCV, Lecture Notes in Computer Science*; Springer Science+Business Media: Berlin/Heidelberg, Germany, 2014.