Improving trajectory estimation using 3D city models and kinematic point clouds

Lukas Lucks1 | Lasse Klingbeil2 | Lutz Plümer3 | Youness Dehbi2

1Fraunhofer (IOSB) Institute of Optronics, System Technologies and Image Exploitation, Ettlingen, Germany
2Institute for Geodesy and Geoinformation, University of Bonn, Bonn, Germany
3Faculty of Geosciences and Environmental Engineering, Southwest Jiaotong University, Chengdu, China

Correspondence
Lukas Lucks, Fraunhofer (IOSB) Institute of Optronics, System Technologies and Image Exploitation, Ettlingen, Germany.
Youness Dehbi, Institute for Geodesy and Geoinformation, University of Bonn, Bonn, Germany.
Email: lukas.lucks@iosb.fraunhofer.de (L.L.), dehbi@igg.uni-bonn.de (Y.D.)

[Correction added on 26 February 2021, after first online publication: Youness Dehbi was designated as corresponding author.]

Abstract
Accurate and robust positioning of vehicles in urban environments is of high importance for autonomous driving or mobile mapping. In mobile mapping systems, a simultaneous mapping of the environment using laser scanning and an accurate positioning using global navigation satellite systems are targeted. This requirement is often not guaranteed in shadowed cities where global navigation satellite system signals are usually disturbed, weak or even unavailable. We propose a novel approach which incorporates prior knowledge (i.e., a 3D city model of the environment) and improves the trajectory. The recorded point cloud is matched with the semantic city model using a point-to-plane iterative closest point method. A pre-classification step enables an informed sampling of appropriate matching points. Random forest is used as classifier to discriminate between facade and remaining points. Local inconsistencies are tackled by a segmentwise partitioning of the point cloud where an interpolation guarantees a seamless transition between the segments. The general applicability of the method implemented is demonstrated on an inner-city data set recorded with a mobile mapping system.

1 | INTRODUCTION

The accurate determination of the trajectory of a vehicle in an urban environment is a key requirement in many applications. In the case of autonomous driving, it is necessary to localize a vehicle in real time within a map in order to navigate it to the desired location. In the context of mobile mapping, laser scanners and cameras mounted
on a vehicle are used to generate high-precision 3D information about the surroundings of the moving car. Here, the trajectory, which may be derived in a post-processing step, is used to register all sensor data within a global coordinate system.

In both examples, global navigation satellite systems (GNSS) such as GPS, Glonass, Galileo or Beidou are the key to the absolute accuracy of the trajectory, while the integration of other sensors, such as inertial measurement units (IMUs), odometers, cameras or scanners increases the relative accuracy and smoothness of the trajectory or helps to bridge GNSS outages. Regardless of the type of GNSS processing method used (carrier-phase-based differential GNSS usually used in mobile mapping or code-based GNSS usually used in autonomous driving), the urban environment imposes strong systematic errors on the observations due to multipath effects and non-line-of-sight signal reception. Even in differential GNSS processing, these errors can be of the order of several meters. It is also possible that a GNSS position cannot be processed at all over a longer period of time and that the other relative sensors cannot prevent the trajectory estimation from drifting without bound. Figure 1 (left) illustrates an example of such errors by means of a trajectory that has been recorded in an inner-city area. It can be stated that the trajectory intersects with building elements which do not match the true driving path.

One option to increase the absolute accuracy in weak GNSS conditions is to integrate prior knowledge about the environment into the estimation process. In map matching, prior knowledge is represented by a street map. The initially estimated trajectory is then matched against this map on the basis of coordinates or curvature profiles. The transformation of this match is then used to improve the initial estimate.

In this contribution, we show that it is also possible to use a 3D model of the environment as prior knowledge in the form of less of detail 1 (LOD1) city models. By comparing the 3D information based on some initially estimated trajectory and the corresponding laser scanner observations with the georeferenced 3D model from a database, we update the estimate and improve its absolute accuracy. The complementarity of both GNSS and 3D building models is a key issue here. In addition, height information is integrated from an elevation model. In this manner, the city model contributes to high accuracy in shadowed cities, exactly where GNSS signals are usually weak or disturbed. We implement an overall processing chain from the detection of facades and roads in point clouds, to the matching between those and the models and the update of the trajectory estimate. We show the general applicability of our procedure on a 5 km long trajectory, which has been recorded with a mobile mapping system within a 2 × 1.5 km test area in the city of Bonn, Germany. It is shown that the inaccurately false estimated trajectory in Figure 1 (left) has been updated and corrected by integrating GNSS information and the city model of the surrounding area (right).

As a post-processing step, it is also possible to integrate this in an online fashion, as would be necessary in the example of autonomous driving. In this case, the transformation needs to be calculated based on the very last point cloud section, that is generated using the trajectory output of an online estimator such as the Kalman filter. The transformation would be treated as an absolute measurement and integrated into the filtering algorithm accordingly.

**Figure 1** Trajectory recorded with a mobile mapping system in Bonn, Germany. The trajectory intersects with buildings (left). The corrected trajectory incorporating a 3D city model of the surrounding area (right)
The remainder of this article is structured as follows. Section 2 gives a review of the most closely related research. Section 3 presents the theoretical background of the methods used. Section 4 introduces our experiments and discusses our results. Section 5 summarizes and concludes.

2 | RELATED WORK

For our approach, four fields are relevant: (1) trajectory estimation in mobile mapping systems; (2) simultaneous localization and mapping (SLAM); (3) map matching; and (4) 3D point cloud interpretation.

2.1 | Trajectory estimation in mobile mapping systems

In mobile mapping systems, the trajectory is usually estimated by a combination of GNSS and IMU. The GNSS processing method (differential or absolute, code or carrier phase based) and the local GNSS conditions determine the absolute position accuracy of the trajectory, while the quality of the IMU determines the accuracy of the orientation component. The IMU also influences the length of a period in which the trajectory can still be estimated, even if the GNSS is interrupted or severely disturbed. Because the laser scans are usually transformed into the global coordinate system in an open loop manner using the trajectory parameters, any error in the trajectory estimation directly affects the quality of the resulting point cloud (Glennie, 2007). Because of this, mobile mapping systems mostly use high-grade IMUs with relative carrier-phase-based GNSS processing. A good overview of mobile mapping technology can be found in Puente, González-Jorge, Martínez-Sánchez, and Arias (2013). In the proposed method, we extend the classical mobile mapping approach by using additional information from an a priori known 3D city model.

2.2 | SLAM-based approaches

One way of overcoming the problem of systematically incorrect or missing absolute position data is to incorporate observations from the object space, such as the scan points or images, into the estimation procedure. It is quite common to use control points along the path to adjust and evaluate the trajectory, and therefore the resulting point cloud (Clancy, 2011; Gräfe, 2007). Eckels and Nolan (2013) use multiple passes with the system along the same trajectory to detect and adjust errors resulting from GNSS insufficiencies. Another option to improve the trajectory accuracy is to estimate it simultaneously with the 3D map by integrating the scanner data and the navigation sensor information in a single adjustment (simultaneous localization and mapping). This is a common technique in robotic applications (Chang, Niu, Liu, Tang, & Qian, 2019; Stachniss, 2009; Thrun, Burgard, & Fox, 2005) and has also been applied to 3D laser scanning in urban environments (Jutzi, Weinmann, & Meidow, 2013; Nüchter, Lingemann, Hertzberg, & Surmann, 2007). Others (e.g., Zhang & Singh, 2014) divide the SLAM approach into a high-frequency, low-fidelity LiDAR odometry part, and a lower-frequency part, optimizing the map on a global scale. In any of the above cases, the map itself is part of the optimization process. In contrast to this, our approach seeks to optimize the absolute trajectory accuracy by introducing additional information into the estimation process.

2.3 | Map matching

Various types of maps have been integrated to improve trajectory estimation. One example is the use of street networks and preliminary trajectories. In this context, Haunert and Budig (2012) developed a method for matching
GNSS trajectories with road data which deals with incomplete or missing roads. They applied a hidden Markov model (HMM) together with a discrete set of candidate matches and off-road candidates for each point of the trajectory. Osogami and Raymond (2013) applied inverse reinforcement learning for the estimation of the importance of the number of turns compared to the travel distance. The estimated importance is then incorporated into the transition probability of an HMM as in the case of Haunert and Budig (2012). Also based on HMMs, Viel et al. (2018) proposed a map-matching algorithm which is able to deal with noisy and sparse cellular fingerprint observations instead of GNSS positions. De Paula Veronese, Badue, Cheein, Guivant, and De Souza (2019) used matching with a road network to generate virtual GNSS measurements, which are then used to update the drifting solutions from a LiDAR odometry-based approach.

Based on LiDAR for observing the surrounding environment and an a priori known 3D point cloud map for position estimation within the map, Javanmardi, Gu, Javanmardi, and Kamijo (2019) proposed a method for the self-localization of autonomous vehicles. They derive different abstract maps from a previously recorded accurate point cloud and use them to self-localize a vehicle in these maps. Although the matching approach is in general similar to ours, we focus on the applicability of official 3D models, which are available on the fly via geo-web services. Matching with other known features of the environment has been presented in Bureick, Vogel, Neumann, Unger, and Alkhatib (2019). Here, the scan lines of a unmanned aerial vehicle (UAV) based laser scanner and plane parameters from a simulated city model are used to create constraints, which are used in an iterative Kalman filter for the trajectory estimation of the UAV. Although the general idea is similar, in contrast to our work, this approach is used for UAVs and does not use real-world experimental data for evaluation.

2.4 | Three-dimensional point cloud interpretation

The interpretation of LiDAR point clouds for building modeling has been intensively investigated. Niemeyer, Rottensteiner, and Sörgel (2012) applied conditional random fields for the classification of airborne LiDAR point clouds. Conditional random fields allows for the incorporation of contextual information for the discrimination between five classes: buildings, low vegetation, trees, natural ground, and asphalt ground. To this end, intensity information as well as geometric features have been exploited. The latter describes the local context of a single point by its position, its normal and the distribution of positions and normals of its parametrized neighborhood. A data-driven approach is followed by Zhou and Neumann (2008), developing features for the reconstruction of buildings efficiently and robustly. In this context, trees are identified and eliminated in a first step, then ground points and roof patches are extracted using a distance-based region growing. Finally, simple and correct polygonal mesh models are generated based on the roof boundaries in a bottom-up manner. Chehata, Guo, and Mallet (2009) proposed a feature selection method for urban classification using random forests. They separated the designed features into five groups: height-based, waveform echo-based, eigenvalue-based, local-plane-based and full-waveform LiDAR features. Rusu, Blodow, and Beetz (2009) introduced the fast point feature histogram (FPFH) as discriminative descriptor for the detection of objects in point clouds. Rouhani, Lafarge, and Alliez (2017) proposed a method for segmenting textured meshes extracted from multi-view stereo based on Markov random fields. Weinmann, Jutzi, Hinz, and Mallet (2015) presented an approach for 3D scene analysis from 3D point clouds consisting of a neighborhood selection step followed by feature extraction as well as selection and classification in the last step. Landrieu and Simonovsky (2017) presented an approach where a captured scene is divided into a so-called superpoint graph. The latter divides the point cloud into geometrically homogeneous elements which are assigned to semantic classes based on a graph convolutional network.

Summarizing, we present an iterative closest point method (ICP) based approach for trajectory estimation involving 3D city models for the informed sampling of matching candidates. Our method can cope with measurements acquired from an inaccurate low-cost GPS sensor without the need for overlapping scan profiles for LiDAR odometry. Furthermore, we performed a quality assessment of volunteered geographic information (VGI) and
investigated whether OpenStreetMap (OSM) data can meet the expectations of our method, comparing them with authoritative data. We also integrate geo-web services in our workflow, which contain besides the building models a terrain model for a more accurate matching of the point cloud. The feasibility of our approach is demonstrated based on a newly captured large-scale data set, which is associated with both high-accurate point cloud and trajectory as ground truth.

3 | METHODOLOGY

In this section we present our approach from a methodological point of view and give insights into the different components of our method. The workflow is depicted in Figure 2. Furthermore, the steps performed are summarized and described in Algorithm 1 for the sake of readability. The input data consist of a potentially inaccurate GNSS trajectory and a 3D kinematic point cloud generated using this trajectory. In addition, a 3D city model of the region surveyed is derived on demand via a web feature service (WFS). In order to take height information into account, an elevation model of the area of interest is requested using a web coverage service (WCS). More details on the data and the experimental settings are provided in Section 4.1. In a first step, the point cloud is interpreted pointwise using supervised machine learning methods. This yields discrimination between facade points and remaining points (see line 1 in Algorithm 1). On the other hand, street points are classified in order to integrate the street-level elevation in the matching step afterwards (see line 2 in Algorithm 1). These steps are elaborated in Sections 3.1 and 3.2. To this end, pre-designed features are extracted from the point cloud beforehand. It can be

FIGURE 2  Overview of the workflow of our approach. An inaccurate trajectory and point cloud issue from mobile mapping (top). The point cloud is partitioned according to the captured trajectory shape. The partitions together with pre-classified building and street points (right) are a prerequisite for a registration step building upon a 3D city model from a geo-web service (left). The transformations acquired are applied to the original data, resulting in an improved trajectory and point cloud (bottom). A seamless transition between single partitions is ensured by an interpolation step
seen that the point cloud is characterized by a drift which is obvious after comparing the outward (green) and the return (red) journey in Figure 2. In order to overcome this deficiency, an automatic piecewise segmentation of the point cloud based on the GNSS trajectory is performed (see line 3 in Algorithm 1). The procedure is based on the Douglas–Peucker simplification algorithm (Douglas & Peucker, 1973), enabling a subsequent registration of each segment with the city model using the ICP method (Zhang, 2014).

**Algorithm 1: Improving trajectory using 3D city models**

**Input:** OT ... original trajectory, P_{cloud} ... captured point cloud, B ... building model, DEM ... digital elevation model

**Output:** IT ... improved trajectory, IC ... improved point cloud

1. \[ [P_{facades}] = CLASSIFY_FACADE_POINTS(P_{cloud}) \]
   // section 3.1
2. \[ [P_{street}] = CLASSIFY_STREET_POINTS(P_{cloud}) \]
   // section 3.2
3. \[ [QT_1, \ldots, QT_n; QC_1, \ldots, QC_n] = PARTITION_TRAJECTORY(OT, P_{facade}; P_{street}) \]
   // section 3.3
4. For \( i = 1 \ldots n \) do
5. \[ [T_i, QC_i] = COMPUTE_TRANSFORMATION(QC_i, B, DEM) \]
6. \[ [IT_1, IC_i] = IMPROVE_TRAJECTORY(QT_1, T_1) \]
   // section 3.4
7. \[ [IT, IC] = INTERPOLATE_TRAJECTORY(IT_1, \ldots IT_n; IC_1, \ldots IC_n) \]
   // section 3.5

The main idea of our approach is to combine the inaccurate GNSS track and the point cloud with more accurate information from the 3D city model. This leads to an improvement of the georeferencing of the scene captured and the trajectory as well. To this end, the precise 3D city model is incorporated in the ICP. This model represents important background knowledge for finding appropriate matching points based on a learned classifier, which is the topic of Section 3.1. The matching step is elaborated in Section 3.4. For each segment, we acquire a local transformation between the point cloud and the corresponding building model (see lines 4–6 in Algorithm 1). For a seamless transition between underlying consecutive segments, we performed an interpolation step which will be explained in Section 3.5 (see line 7 in Algorithm 1).

### 3.1 | Point-based facade classification

The mobile mapping system provides a huge and dense 3D point cloud. To reduce the number of points and the computation time accordingly, we specified at most 10 cm as the minimum distance between two points, obtaining a still detailed point cloud consisting of roughly 60 million points. A more detailed data description is given in Section 4.1. In order to enable an informed sampling of appropriate points for registration later on, a pre-classification of the input 3D point cloud is of great importance. This step leads to an efficient selection of matching points according to the ICP algorithm. For this purpose, an active sampling of points belonging to facades is a key issue. We address this task as a binary classification problem where a point-based discrimination is enabled. This enriches the points as geometric entities by class labels, leading to a semantic segmentation of the point cloud. In this manner, we are able to differentiate between building points and remaining points using random forests (Breiman, 2001).
as a robust classifier in a supervised machine learning task. For each point in the 3D point cloud, we derive a set of geometric features for a binary classification. We apply features which characterize the local context of a single point by its position, its normal, and the distribution of positions and normals of its neighborhood.

To express these properties, we calculate several features based on the eigenvalues of the covariance matrix as proposed by Hackel, Wegner, and Schindler (2016). The covariance matrix is calculated from a local neighborhood around each point. A planarity feature is used to distinguish between human-made objects and vegetation. Further, a verticality feature separates horizontal objects from vertical ones. We use the point density calculated from a local neighborhood around each point as an additional feature. The point density is calculated in both 2D and 3D space. To further describe the local geometry around a point, we use point feature histograms (PFHs). Due to the high number of points, we used the so-called fast point feature histograms (FPFHs) as an approximation of the PFHs to simplify the calculation (Rusu et al., 2009). The FPFH and the covariance matrix-based features were derived in an offline pre-processing step based on the Point Cloud Library (http://pointclouds.org) and the open source software CloudCompare (https://www.danielgm.net/cc/).

In addition to the geometric features, we make use of intensity information measured by the mobile mapping system. This is particularly important for discriminating between vegetation and building points based on the surface properties. All in all, 14 features and a further 33 FPFH descriptors were used to classify the building points. The features used are shown in Table 1. As already mentioned, the classification results lead to appropriate sample points for the subsequent registration using the ICP approach.

### 3.2 Feature-based street detection

In order to consider elevation information and exploit the 3D structure of the building models, our aim is to align the trajectory and the point cloud along the building facades and the horizontal street level as well. Otherwise, the vertical position of the point cloud would only be estimated oriented by the facade heights. To this end, street points are of great interest. In this context, a two-stage classification of these points was performed in addition to facade points in the preceding step. In a first step, we safely assume that street points are characterized by a verticality value of less than 0.1. This feature is calculated based on the covariance matrix as mentioned in Section 3.1. Accordingly, flat

| TABLE 1 | Features used for point-based classification of facades |
| --- | --- |
| Intensity |  |
| Sum of eigenvalues |  |
| Omnipvariance |  |
| Eignentropy |  |
| Anisotropy |  |
| Planarity |  |
| Verticality |  |
| Linearity |  |
| Sphericity |  |
| First component after principal components analysis |  |
| Second component after principal components analysis |  |
| Surface variation |  |
| Point density 2D |  |
| Point density 3D |  |
| FPFH (33 descriptors) |  |
areas are maintained within the point cloud as possible road candidates as depicted in Figure 3, which illustrates the stepwise filtering of road points. Remaining points are then filtered by their distance to the trajectory. Points with an absolute distance greater than 2.0 m are subsequently rejected, leading to our points of interest.

3.3 | Trajectory partitioning

The next step is to perform a segmentation of the point cloud. As the underlying point cloud is not consistent, a registration based on the whole recorded scene would improve the georeferencing but would not correct the drifts within the point cloud. Thus, the idea is to take the temporal information into account and to partition the point cloud into several segments oriented by the trajectory. In this manner, a better local matching with the corresponding 3D building model can be achieved since each segment is registered separately. The main reason supporting this choice is that the point cloud accuracy is time-dependent. The uncertainties and GPS errors, for instance, are different according to the temporal and spatial distribution of the trajectory captured. Figure 8 shows two inconsistent point clouds of the same facade captured by the outward and returning journey (red versus green). In this context, a transformation based on all the points will lead to incorrect alignment of the point cloud. Likewise, a matching with the entire point cloud, even in one driving direction, leads to discrepancies which are attributed to inhomogeneous errors over time. These deficiencies are overcome using a segmentwise registration of the point cloud. In this context, the errors are safely assumed to be locally constant for one single segment.

The first step toward an automatic point cloud segmentation is an appropriate trajectory partitioning. The pre-classified point cloud is automatically partitioned into several segments. In order to get convenient segments for the subsequent matching step, we focus on turns in the trajectory which have been extracted using the Douglas–Peucker algorithm. The latter expects the original trajectory $OT$ (black) as input as illustrated in Figure 4 and generalizes it leading to a simplified trajectory $ST$ (green) with $n$ vertices $v_1, \ldots, v_n$. The latter consists of a sequence of...
segments \((s_1, \ldots, s_{n-1})\) with \(s_i = v_{i+1}v_i\) for \(i = 1, \ldots, n-1\). Each green vertex \(v_i\) corresponds to a turn in the trajectory according to a simplification parameter. Based on the simplified trajectory, we get our targeted segmented trajectory \(QT\) consisting of paths \(q_1, \ldots, q_n\) as presented on the right of Figure 4. Each path \(q_i\) is defined as follows:

\[
q_i = \begin{cases} 
    v_1c_1, & i = 1, \\
    c_{i-1}v_{i}c_i, & i = 2, \ldots, n-1 \\
    c_{n-1}v_n, & i = n,
\end{cases}
\tag{1}
\]

where each \(c_i\) is the middle of the segment \(s_i\) from the simplified trajectory \(ST\). Oriented by the segments of the trajectory, we also divide the point cloud into single parts which will be presented in Section 4.2. The resulting point clouds provide a basis for the registration with the building models.

### 3.4 Point cloud registration

The registration of the point cloud is performed using the ICP algorithm. For this purpose, the algorithm calculates and applies a transformation between two point sets iteratively, assuming that every point in the first point set corresponds to another point in the second. In general, the transformation of a 3D point from its source position \(p_S\) to a targeted position \(p_D\) can be expressed by a 3D rotation matrix \(R\) and a translation vector \(t\) as described in the following equation:

\[
p_D = Rp_S + t
\tag{2}
\]

The aim of the ICP algorithm is to estimate such a transformation from two corresponding point sets \(P\) and \(M\). The rotation \(R\) and the translation \(t\) are estimated by minimizing the distances between the point sets using the following loss function:

\[
L(R, t) = \sum_{i=1}^{N} \| m_i - R p_i - t \|, \quad \forall p_i \in P, \quad \forall m_i \in M
\tag{3}
\]

This transformation can also be performed by first reducing the points using the centroids \(\mu_S\) and \(\mu_D\) in the source and destination coordinate system respectively as follows:

\[
p_D = R (p_S - \mu_S) + \mu_D
\tag{4}
\]

substituting \(t\) by \(\mu_D - R \mu_S\) in Equation (2). In this way, both point sets are moved to the origin of the coordinate system in such a way that they share the same centroid, and thus the rotation can easily be determined. We used rotation matrices based on quaternions, which have significant advantages such as reducing the rotation axes and omitting the trigonometric terms. The determination of the rotation quaternions was performed by singular value decomposition (Förstner & Wrobel, 2016).

The ICP algorithm builds on two given point sets which have to be matched. In our case, however, we have to deal with one point set which has to be directed according to a 3D building model. Thus, we make use of the so-called point-to-plane ICP instead (Low, 2004). Figure 5 depicts the main idea of this method. In this context, building facades are modeled as planes. A matching point (red) is a point lying on the plane at a minimal distance to the corresponding point (green) of the point cloud. Furthermore, each matching point needs to be included in the polygon defined by the facade boundary (light blue). In this work, relevant facade planes are selected based
on a bounding box around the captured point cloud. As mentioned in Section 3.1, appropriate points for ICP are pre-filtered based on a classification step. We used trimmed ICP as a robust version of ICP which does not take matching points at a large distance to source points into account (Chetverikov, Svirko, Stepanov, & Krsek, 2002). In this context, larger-distance quantiles of all paired points are removed to identify mismatches. The interested reader is referred to Pomerleau, Colas, Siegwart, and Magnenat (2013) for the comparison of ICP variants.

The point cloud is characterized by an accuracy of several meters. In this context, a mismatch between points from the point cloud and the corresponding building facade has to be avoided and, hence, a false transformation can be prevented. In particular, this is important after the partitioning of the point cloud into small segments due to possible loss of spatial information, which especially occurs when driving straight ahead where most facades are parallel to the driving direction. Together with the inaccurate georeferencing of the point cloud, the approach might not be able to sufficiently transform such segments. The ICP algorithm cannot find a unique assignment to associate the point cloud to the model and may converge to an inaccurate solution. In order to overcome this problem, we used the already mentioned modification of the Douglas–Peucker algorithm for the partitioning instead of a simple solution such as a length-oriented partitioning of the trajectory.

In addition to the building models, the information from a digital elevation model (DEM; see Figure 2) was exploited for the determination of the vertical position of the point cloud, and thus of the trajectory. Therefore, the segmentwise transformation calculated by the final points is extended by an additional translation, adapting Equation (4) as follows:

\[ p_D = R(p_S - \mu_S) + \mu_D + t_Z \]

where \( t_Z \) is a translation along the Z-axis. The translation vector is calculated based on the street points together with the corresponding DEM. To this end, the nearest point on the DEM is identified for each street point. Based on these point pairs, the average height difference is calculated and used to determine \( t_Z \).

### 3.5 Interpolation after the piecewise transformation

The registration provides a set of transformation parameters \( T_i \) for each segment \( i \) consisting of a rotation matrix \( R_i \) and a translation vector \( t_i \). Unfortunately, the ICP solutions may differ slightly for each segment. If the points of each segment are transformed using only the local transformation, overlapping areas or gaps in the trajectory and
the point cloud may occur in the boundaries of two consecutive segments. In order to overcome these artifacts, we performed an interpolation between two subsequent segments \(i\) and \((i+1)\). To do so, each point \(p_S\) is transformed twice according to \(T_i\) and \(T_{i+1}\), resulting in two different solutions:

\[
\begin{align*}
    p_{T_i} & = R_i p_S + t_i \\
    p_{T_{i+1}} & = R_{i+1} p_S + t_{i+1}
\end{align*}
\]

The points obtained are then weight-averaged for the final interpolated solution \(p_I\) using the equation

\[
p_I = w_i p_{T_i} + w_{i+1} p_{T_{i+1}}
\]

The weights \(w_i\) and \(w_{i+1}\) are calculated as follows:

\[
\begin{align*}
    w_i & = \frac{d_i}{d_i + d_{i+1}} \\
    w_{i+1} & = \frac{d_{i+1}}{d_i + d_{i+1}}
\end{align*}
\]

where \(d_i\) and \(d_{i+1}\) represent the distance between the point \(p_S\) and the corresponding centroid of the point sets from segments \(i\) and \((i+1)\), respectively.

4 | EXPERIMENTS

This section gives insight into the data used in the different steps of our approach and presents our experiments and results in detail.

4.1 | Data settings

4.1.1 | Three-dimensional point cloud and trajectory

The 3D point cloud was recorded using a mobile mapping system (Figure 6), consisting of a high-precision profile laser scanner (Zoller & Fröhlich GmbH, 2020) and a navigation-grade IMU (IMAR inav-FJI-LSURV). It also included
a multi-constellation, multi-frequency GNSS receiver. For the reference trajectory, the data from the IMU and the GNSS receiver were processed to a highly accurate 6D trajectory using the Waypoint Inertial Explorer software (NovAtel Inc., 2020). This trajectory was then used to transform the laser scanner measurements to a global coordinate system. The laser scanner has a distance measurement accuracy of the order of 1 mm. The trajectory parameters (position and orientation) from the GNSS/IMU were then used to register all scan points to a globally consistent georeferenced point cloud. Taking the uncertainty of the GNSS/IMU unit into account, this results in a point cloud accuracy of the order of a few centimeters (Heinz, Holst, Kuhlmann, & Klingbeil, 2020). This highly accurate point cloud was taken as ground truth for the labeling of street and facade points used in the classification task described in Section 3. In our experiments we artificially reduced the trajectory estimation accuracy to a few meters, mimicking multipath and other GNSS errors. This degraded trajectory, when combined with the scanner data, leads to a new point cloud with an overall accuracy of several meters, which is then used to test our algorithms.

We employed a 2 km × 1.5 km area in the city of Bonn, Germany, where a 5 km long trajectory was driven within a time of about 20 min. As described above, the trajectory was accurately processed as a reference and artificially degraded by using only code-based GNSS observations. This degraded trajectory is called in the rest of this paper the initial trajectory. The point cloud derived from the initial trajectory (Figure 7, top) was then used as a data input for the proposed algorithm. The segmentwise difference between the initial trajectory and the reference trajectory is shown in Figure 15 in blue, quantified by mean squared errors (MSEs) which depict a deviation up to 13 m.

4.1.2 Three-dimensional city model and elevation model

A further important input for our approach are building models in the surroundings of our driving area. In this context, we use building models of the environment with LoD1 according to the standardized CityGML format.
(Gröger, Kolbe, Nagel, & Häfele, 2012). This level of detail includes models derived from footprints with extruded heights. Figure 7 (bottom) visualizes an excerpt of the city model of our test area. The buildings are acquired by performing a WFS request specifying a bounding box of the region of interest. The WFS is hosted on a geo-server (http://geoserver.org/) providing CityGML models from open authoritative data sources from OpenGeodata. NRW (https://www.opengeodata.nrw.de/produkte/) of the state of North Rhine-Westphalia in Germany (https://www.bkg.bund.de/EN/Home/home.html). The 2D position accuracy amounts to few centimeters, enabling us to improve the accuracy of the point clouds and in turn the accuracy of the trajectory. The height, however, is characterized by a lower accuracy (around 5 m).

The availability of authoritative data cannot always be ensured. For this reason, we evaluate whether the quality of VGI data meets the expectations of our approach. In this context, we expand our web service engine with a backend providing data derived from OSM (openstreetmap.org). As stated by Fan, Zipf, Fu, and Neis (2014), the completeness of OSM’s building footprints is high, but some building attributes are mostly missing. In particular, only few buildings are recorded with attributes related to height information such as numbers of stories, which is relevant in our context. In order to overcome this deficiency, we enriched the affected footprints with facade surfaces by extruding the boundaries of the corresponding ground surfaces. The extrusion height is set to the value of the projection of the highest laser beam on the Z-plane.

As already mentioned, our aim was to align the point cloud both vertically and horizontally. We therefore requested an elevation model of the region of interest on the fly based on a freely available WCS by Geobasis NRW (https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/index.html).

4.1.3 | Coordinate systems

Both the 3D point cloud and the 3D city model are represented in a Universal Transverse Mercator (UTM) coordinate system. The heights of the 3D points are ellipsoidal, however, as GNSS systems provide these by default, whereas those of the city model are represented as physical heights with respect to the German main height network. In this context, an adjustment step is needed in order to map both heights. To this end, the geoid undulation had to be determined. We made use of a precise network of measured heights provided by the German federal agency for cartography.

4.2 | Experimental results

4.2.1 | Point-based classification

To classify the 3D point cloud into “building” and “no-building” points, we used the random forest classifier (Breiman, 2001). For this purpose, an automatic labeling of the point cloud was performed based on the underlying precise city model. In this context, the semantically rich CityGML model was used to select facade or ground surfaces by extruding the boundaries of the corresponding ground surfaces. The extrusion height is set to the value of the projection of the highest laser beam on the Z-plane.

As already mentioned, our aim was to align the point cloud both vertically and horizontally. We therefore requested an elevation model of the region of interest on the fly based on a freely available WCS by Geobasis NRW (https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/index.html).

Table 2 | Results of the point-based classification of building and no-building points

| Class       | No-building | Building | Precision (%) |
|-------------|-------------|----------|---------------|
| Pred. no-building | 48,561,230  | 298,926  | 99.39         |
| Pred. building   | 942,141     | 3,922,521| 80.63         |
| Recall (%)       | 98.10       | 92.92    | 97.68         |

Notes: The lower left value corresponds to the overall accuracy.
surfaces. The point cloud derived from the reference trajectory was involved due to its accurate georeferencing. Consequently, labeled training data could be safely generated. Points are considered as building points if their distance to the city model is less than 0.5 m and their verticality value is greater than 0.5. In a similar way, no-building points are labeled when the distance to the building models exceeds 1 m and the points verticality is less
than 0.5. The points that do not meet these requirements remain unlabeled. For the evaluation of the resulting model, a threefold cross-validation was performed. Overall, we achieved an accuracy of 97.68%. The detailed confusion matrix is shown in Table 2. Our points of interest (i.e., the building class) are predicted with a precision of about 80.63%. With regard to the training data, the classifier is much more restrictive concerning the selection of building points. Nevertheless, the large number of correctly predicted examples enables an informed sampling of facade points later on. In particular, the recall value achieved (approx. 92.92%) is more significant, since false positive examples directly influence the matching between point cloud and building model. The 7% error rate is attributed to the presence of concurrent vertical objects (i.e., underpasses, large motorway signs or noise barriers) which are difficult to distinguish from building facades, as will be shown later in this section. As mentioned, we used the trimmed ICP to reduce the negative influence of points of this kind.

4.2.2 | Segmentation and registration of the point cloud

The result of our approach and its impact on the point cloud are exemplarily illustrated in Figure 8 where an example of the point cloud is shown before (left) and after (right) the alignment of the point cloud with the city model. The green points in Figure 8 stem from the outward journey, whereas the red points belong to scans from the return journey. A deviation between the two original point clouds on the left can be seen. On the right, this deviation cannot be observed after the application of our approach. The same point cloud is illustrated at the bottom of the figure projected on an orthophoto of the driving area. Besides the consistency, an improvement of the

![Figure 10](image_url) **Figure 10** Residuals with respect to the reference trajectory for each coordinate component during a drive. The vertical gray lines represent the segment boundaries.
absolute position is achieved. The deviation to the building on the left has been corrected, enabling point locations that even reflect the building positions.

**Figure 11** Impact of the interpolation on the trajectory (green). The trajectories matched based on local transformations for segments $S_i$ and $S_{i+1}$: (blue) transformed by $T_i$ and (red) by $T_{i+1}$. The dotted sections show the transformed parts of the next partition. It can be seen that the interpolated trajectory converges to the local solution of each partition (on the left the green line converges to the red, and on the right to the blue trajectory).

**Figure 12** Influence of elevation data on the improvement of the trajectory.
The point cloud which has been previously partitioned oriented by turns in the underlying trajectory as depicted on the right of Figure 9. As mentioned, the trajectory was simplified using the Douglas–Peucker algorithm preserving inflection points, which mostly correspond to turns (see the blue vertices on the left). Subsequently, this simplification step enables the acquisition of final segments including turning points. This acquired segmented trajectory is then used to segment the point cloud in the last step using the middle points of the simplified trajectories (cf. Section 3.3 and Figure 4). The resulting partitioned trajectory is shown in the middle.

4.2.3 Improvement of the trajectory

The visual analysis of the results shows that our method improves the point cloud significantly, as mentioned above. The improvement of the point cloud position implies also an improvement in the trajectory. For a more rigorous evaluation of the accuracy improvement, we performed a quantitative quality assessment using the highly accurate reference trajectory and our data set. The results are presented in Figure 10, showing the differences $\Delta X$, $\Delta Y$ and $\Delta Z$ between the coordinates of the reference trajectory and the three trajectories: initial (blue), transformed without interpolation (yellow) and transformed with interpolation (green). The vertical gray lines represent the partition boundaries. The differences have been calculated pointwise based on the GPS time-stamps. Obviously, the transformation and interpolation lead to meaningful improvements in all dimensions. In particular, the peaks of the initial trajectory (blue) have been significantly flattened. The segmentwise transformed trajectory (yellow) shows a reduction of up to 6.8 m in the $X$-direction and 11.3 m in the $Y$-direction, while an improvement of 7.3 m is achieved in the $Z$-direction. Regarding the deviation of the $X$ and $Y$ coordinates, it is remarkable that the improvement is only possible at segment level. In most cases, the deviation of the transformed trajectories shows a nearly constant offset to the initial trajectory.

As expected, the transformed points show discontinuities at the segment boundaries. These discontinuities have been smoothed by the interpolation (green). The interpolation influences the result in different ways. For several segments, a significant reduction of the deviation can be observed, but in few cases a deterioration is also evident. For instance, the interpolation leads to an increase of the deviations in the $Z$-direction for the second segment compared to the initial trajectory. This can be explained by the poor preconditions of the previous segment. A close examination revealed that this segment is a small one, leading to a few facade points. Besides the segment size, vegetation-related occlusions also contributed to the lack of sampling points for the ICP algorithm.

Further, we observed that the associated building model in that region is lower than the captured facade. All these

**FIGURE 13** Three-dimensional visualization of the trajectory results: initial trajectory (blue), reference trajectory (green), improved trajectory (red) and the building models (dark blue)
described inaccuracies have an impact on the determination of the transformation. Due to the interpolation, this transformation has a negative impact on the position of the trajectory of the following segment.

Figure 11 illustrates the positive impact of our interpolation step which can be seen in the resulting trajectory (green). Transformed trajectories according to transformations $T_i$ and $T_{i+1}$ (see Equation 7) are illustrated in blue and red, respectively. Transformed parts of the consecutive partition are highlighted with dotted lines. The interpolated trajectory (green) converges to the local solution of each partition (red on the left and blue on the right).

**FIGURE 14** Impact of our approach on the recorded trajectory based on different building models. The reference trajectory is displayed as a green line, the initial trajectory as a blue line and the transformed trajectory as a red line. The underlying point cloud is shown in blue: (Top left) input data; (top right) results based on OSM data; (bottom left) results based on cadastral data; (bottom right) results based on the extended OSM data
4.2.4 | Influence of the elevation data

In order to assess the impact of the incorporation of the elevation data on the resulting trajectory, we calculated the MSE of each segment after the interpolated transformation as shown in Figure 12. Compared to the initial trajectory, lower MSE values can be observed confirming the coordinatewise results in Section 4.2.3. In particular, the consideration of a DEM (see Equation 5) leads to an error reduction of 1 m on average. Omitting the elevation information (Equation 4) results in errors between 0.8 and 3.9 m, compared to 0.6 and 2.5 m taking the elevation information into account. Figure 13 illustrates a 3D visualization of the trajectories with respect to the 3D building models. An improvement in the Z-direction can be noticed where the corrected trajectory (red) is almost identical to the reference trajectory (green).

4.2.5 | Suitability of OSM data

For an assessment of the VGI data quality, we evaluated our method based on three different data sources (authoritative data, OSM data, and extruded OSM data) as explained in Section 4.1. The results concerning the OSM data are depicted rightmost in Figure 14 (top). Compared to the initial trajectory (blue) and the one obtained by our method (red), the resulting trajectory deviates furthest from the reference (green). As mentioned in Section 4.1, we extend these data by extruding the footprints of the missing buildings. In this context, the results obtained show an improvement, which can be observed at the bottom of the same figure on the right. At the bottom on the left, the authoritative 3D city model delivers the most accurate achievements.
For further evaluation, we calculate the MSE by segment relative to the initial trajectory based on the three data sets. The results show that the pure OSM data lacking height attributes are not suitable. The MSEs range from about 10 m up to over 100 m and even lead to a substantial degradation of the initial trajectory. In contrast, the extruded data reduced the MSE dramatically. Huge differences to the authoritative data cannot be observed, as can be shown in Figure 15. All in all, we can state that OSM data alone are not suitable for improving trajectories. Nevertheless, they are good enough for our purposes after an extrusion step. In particular, if the authoritative 3D city models are not available, they seem to be a good alternative.

4.2.6 | Discussion

To sum up, our approach turns out to be very successful for the improvement of trajectories recorded by a mobile mapping system. The integration of background knowledge in the form of 3D city models is a key issue. Nevertheless, perfect matching on the whole trajectory is not possible, as can be observed in Figure 10. In this context, we investigated possible error sources causing the deficiencies mentioned. We stated that misclassifications within the point cloud can considerably influence the result. For instance, a wall in Figure 16 (rightmost green region on the right) belonging to a pedestrian and cyclists underground (see bottom) was interpreted as a facade. At this position, the ICP tries to match with a falsely expected facade, leading to unsatisfactory results as shown in Figure 17. Such problems can be overcome by the elimination of points lying below street level. However, outdated city models would also lead to similar problems. An unsatisfactory alignment may also be caused by a sudden change in the position of the initial trajectory resulting from deviations of the GPS sensor. Figure 18 visualizes...
such a case where the initial trajectory (blue) has drifted. This leads to larger deviations with respect to the refer-
ence trajectory (green). In order to deal with this problem, drifted locations can be detected and considered as
partition boundaries. Another aspect worth investigating is to retrieve other partitioning parameters rather than
turns. In this context, an evaluation of the influence of segment sizes on the accuracies achieved will be subject of
future work. The quality assessment of VGI data (i.e., OSM) integrating geo-web services in our workflow revealed
that these data alone are not suitable for improving trajectories. An extrusion step leads, however, to satisfactory
results.

The next step is to integrate the transformations gained as measurements in the trajectory filtering process. In
this context, it is necessary to investigate how often the correction must be applied. It would also be possible to
integrate this transformation in the form of constraints on the trajectory estimation process. In order to restrict
the search space of relevant surfaces for ICP matching, pre-filtering could be performed. In this way, surfaces lying
on planes orthogonal to the drive direction can be omitted to enable more efficient and robust matching.
CONCLUSION AND OUTLOOK

This article presents a novel approach to improving trajectories using 3D city models and kinematic point clouds for highly accurate positioning. A combination of 3D city models and mobile mapping systems enables us to bridge weak GNSS signals and drifts characterizing drives within shadowed cities. We consecutively registered segments of the kinematic point cloud incorporating the building and elevation models of the surrounding area using a point-to-plane trimmed ICP and an informed sampling of pre-classified matching candidates. The resulting transformation was applied to update the trajectory using an interpolation between segments. Together with the new trajectory, they were used to update the point cloud. Compared to other approaches, our method can cope with measurements acquired from an inaccurate low-cost GPS sensor without the need for overlapping scan profiles for LiDAR odometry. We showed that 3D city models are suitable for improving the positioning of vehicles in shadowed cities. The general functionality of the method and the impact on both point cloud and trajectory were demonstrated using a newly captured large-scale data set, which is associated with both a highly accurate point cloud and trajectory as ground truth. The application of our approach in real time will be the subject of future work.

ACKNOWLEDGMENTS

We gratefully acknowledge the open data sets from Geobasis NRW and OpenStreetMap. The authors thank Viktor Stroh for his assistance in the setup of the WFS and WCS and Johannes Leonhardt for performing some implementation tasks. We also thank Erik Heinz for providing the data set. Open access funding enabled and organized by Projekt DEAL.

ORCID

Youness Dehbi https://orcid.org/0000-0003-0133-4099

REFERENCES

Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32.
Bureick, J., Vogel, S., Neumann, I., Unger, J., & Alkhatib, H. (2019). Georeferencing of an unmanned aerial system by means of an iterated extended Kalman filter using a 3D city model. PFG-Journal of Photogrammetry, Remote Sensing & Geoinformation Science, 87, 229–247.
Chang, L., Niu, X., Liu, T., Tang, J., & Qian, C. (2019). GNSS/INS/LiDAR-SLAM integrated navigation system based on graph optimization. Remote Sensing, 11, 1009.
Chehata, N., Guo, L., & Mallet, C. (2009). Airborne LiDAR feature selection for urban classification using random forests. International Archives of Photogrammetry, Remote Sensing & Spatial Information Sciences, 38(3/W8), 207–212.
Chetverikov, D., Svirko, D., Stepanov, D., & Krsek, P. (2002). The trimmed iterative closest point algorithm. In Proceedings of the 16th International Conference on Pattern Recognition, Québec City, QC, Canada (pp. 545–548). Piscataway, NJ: IEEE.
Clancy, S. (2011). The importance of applied control. LiDAR Magazine, 1, 2.
De Paula Veronese, L., Badue, C., Cheein, F. A., Guivant, J., & De Souza, A. F. (2019). A single sensor system for mapping in GNSS-denied environments. Cognitive Systems Research, 56, 246–261.
Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica, 10, 112–122.
Eckels, R., & Nolan, J. (2013). Mobile laser scanning: Field methodology for achieving the highest accuracy at traffic speed. In Proceedings of the 18th Association of Public Authority Surveyors Conference, Canberra, ACT, Australia (pp. 218–230). Bathurst, NSW, Australia: APAS.
Fan, H., Zipf, A., Fu, Q., & Neis, P. (2014). Quality assessment for building footprints data on OpenStreetMap. International Journal of Geographical Information Science, 28, 700–719.
Förstner, W., & Wrobel, B. (2016). Photogrammetric computer vision: Geometry, orientation and reconstruction. Berlin, Germany: Springer.
Glennie, C. (2007). Rigorous 3D error analysis of kinematic scanning LIDAR systems. Journal of Applied Geodesy, 1, 147–157.
Gräfe, G. (2007). Quality management in kinematic laser scanning applications. In Proceedings of the Fifth International Symposium on Mobile Mapping Technology, Padua, Italy (pp. 1–6).

Gröger, G., Kolbe, T. H., Nagel, C., & Häfele, K. H. (Eds.) (2012). OGC City Geography Markup Language (CityGML) Encoding Standard, Version 2.0. Wayland, MA: Open Geospatial Consortium.

Hackel, T., Wegner, J. D., & Schindler, K. (2016). Contour detection in unstructured 3D point clouds. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV (pp. 1610–1618). Piscataway, NJ: IEEE.

Haunert, J. H., & Budig, B. (2012). An algorithm for map matching given incomplete road data. In Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Redondo Beach, CA (pp. 510–513). New York, NY: ACM.

Heinz, E., Holst, C., Kuhlmann, H., & Klingbeil, L. (2020). Design and evaluation of a permanently installed plane-based calibration field for mobile laser scanning systems. Remote Sensing, 12, 555.

Javanmardi, E., Gu, Y., Javanmardi, M., & Kamijo, S. (2019). Autonomous vehicle self-localization based on abstract map and multi-channel LiDAR in urban area. International Association of Traffic & Safety Sciences Research, 43, 1–13.

Jutzi, B., Weinmann, M., & Meidow, J. (2013). Improved UAV-borne 3D mapping by fusing optical and laser scanner data. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 40(l/W2), 223–228.

Landrieu, L., & Simonovsky, M. (2017). Large-scale point cloud semantic segmentation with superpoint graphs. Preprint, arXiv:1711.09869.

Low, K. L. (2004). Linear least-squares optimization for point-to-plane ICP surface registration (Technical Report TR04-004). Chapel Hill, NC: Department of Computer Science, University of North Carolina.

Niemeyer, J., Rottensteiner, F., & Sörgel, U. (2012). Conditional random fields for LiDAR point cloud classification in complex urban areas. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 1(3), 263–268.

NovAtel Inc. (2020). Waypoint Inertial Explorer 8.80 post processing software. Retrieved from http://www2.novatel.com/waypointrelease

Nüchter, A., Lingemann, K., Hertzberg, J., & Surmann, H. (2007). 6D SLAM: 3D mapping outdoor environments. Journal of Field Robotics, 24, 699–722.

Osogami, T., & Raymond, R. (2013). Map matching with inverse reinforcement learning. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China (pp. 2547–2553). New York, NY: ACM.

Pomerleau, F., Colas, F., Siegwart, R., & Magnenat, S. (2013). Comparing ICP variants on real-world data sets. Autonomous Robots, 34, 133–148.

Puente, I., González-Jorge, H., Martí-nez-Sánchez, J., & Arias, P. (2013). Review of mobile mapping and surveying technologies. Measurement, 46, 2127–2145.

Rouhani, M., Lafarge, F., & Alliez, P. (2017). Semantic segmentation of 3D textured meshes for urban scene analysis. ISPRS Journal of Photogrammetry & Remote Sensing, 123, 124–139.

Rusu, R., Blodow, N., & Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan (pp. 3212–3217). Piscataway, NJ: IEEE.

Stachniss, C. (2009). Robotic mapping and exploration. Berlin, Germany: Springer-Verlag.

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. Cambridge, MA: MIT Press.

Viel, A., Gubiani, D., Gallo, P., Montanari, A., Dalla Torre, A., Pittino, F., & Marshall, C. (2018). Map matching with sparse cellular fingerprint observations. In Proceedings of the 2018 IEEE Conference on Ubiquitous Positioning, Indoor Navigation and Location-Based Services, Wuhan, China (pp. 1–10). Piscataway, NJ: IEEE.

Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhood, relevant features and efficient classifiers. ISPRS Journal of Photogrammetry & Remote Sensing, 105, 286–304.

Zhang, J., & Singh, S. (2014). LOAM: LiDAR odometry and mapping in real-time. In Proceedings of the 2014 Robotics: Science and Systems Conference, Berkeley, CA (pp. 1–9).

Zhang, Z. (2014). Iterative closest point (ICP). In K. Ikeuchi (Ed.), Computer vision (pp. 433–434). Boston, MA: Springer.

Zhou, Q. Y., & Neumann, U. (2008). Fast and extensible building modeling from airborne LiDAR data. In Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Irvine, CA (pp. 7.1–7.8). New York, NY: ACM.

Zoller & Fröhlich GmbH. (2020). Z+F Profiler 9012A, 2D laser scanner (Technical report). Retrieved from https://www.zf-laser.com/fileadmin/editor/Datenblatter/Z_F_PROFILER_9012_Datasheet_E_final_compr.pdf

How to cite this article: Lucks L, Klingbeil L, Plümer L, Dehbi Y. Improving trajectory estimation using 3D city models and kinematic point clouds. Transactions in GIS. 2021;25:238–260. https://doi.org/10.1111/tgis.12719