ISPRS Journal of Photogrammetry and Remote Sensing 121 (2016) 48–59

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Localization of a mobile laser scanner via dimensional reduction

Ville V. Lehtola [a,b,*], Juho-Pekka Virtanen [a,b], Matti T. Vaaja [a], Hannu Hyyppä [a], Andreas Nüchter [c,*]

[a] *Institute of Measuring and Modeling for the Built Environment, Aalto University, P.O. box 15800, 00076 Aalto, Finland*
[b] *National Land Survey of Finland, Finnish Geospatial Research Institute FGI, PL 84, 00521 Helsinki, Finland*
[c] *Julius Maximilians University Würzburg, Informatics VII – Robotics and Telematics, 97074 Würzburg, Germany*

A R T I C L E I N F O

Article history:
Received 22 May 2016
Received in revised form 16 August 2016
Accepted 16 September 2016

Keywords:
Localization
Positioning
3D reconstruction
Lidar
Slam

A B S T R A C T

We extend the concept of intrinsic localization from a theoretical one-dimensional (1D) solution onto a 2D manifold that is embedded in a 3D space, and then recover the full six degrees of freedom for a mobile laser scanner with a simultaneous localization and mapping algorithm (SLAM). By intrinsic localization, we mean that no reference coordinate system, such as global navigation satellite system (GNSS), nor inertial measurement unit (IMU) are used. Experiments are conducted with a 2D laser scanner mounted on a rolling prototype platform, VILMA. The concept offers potential in being extendable to other wheeled platforms.

1. Introduction

Localization of a mobile laser scanner (MLS) without using a global reference coordinate system, e.g., satellites, is one of the grand problems in laser scanning research (Lehtola et al., 2015, 2016; Bosse et al., 2012; Bosse and Zlot, 2009; Lauterbach et al., 2015; Liu et al., 2010; Vosselman, 2014; Kaul et al., 2016). Besides its theoretical importance, prominent solutions enable applications in indoor environments, construction and forest settings, and other areas that lack satellite coverage, e.g., planetary sites. If, in addition, the localization is done without any external sensors, such as an inertial measurement unit (IMU), we call it *intrinsic localization*.

The problem is formidable because of its inverse nature. The time-of-flight data of a laser scanner typically results to point clouds of dozens or hundreds of millions of points. Recovering the scanner trajectory by processing this data requires a sophisticated way to reduce its size by identifying invariants, or features, that describe the environment in a sufficient manner. Then the problem is preferably divided into steps, so that one and only one unknown variable is solved in each step, thus keeping the solution space computationally tracktable. Previously, we have tried to apply a 6 DoF semi-rigid SLAM method directly on a 1D trajectory solution in order to correct it into its original physical form (Lehtola et al., 2016), but this approach cannot deal with steep curves because it is based on the iterative closest point (ICP) algorithm. Specifically, when a local minimum in the *n*-scan matching is reached, the iteration gets stuck, and further computation does not help in recovering the actual trajectory.

In this paper, we set out to fix this, and to extend the concept of intrinsic localization from the one-degree-of-freedom solution (Lehtola et al., 2015) back to the six degrees of freedom. This is done in three steps in Section 2. First, local corrections on a horizontal 2D plane are introduced to the 1D trajectory. This is done not by optimizing over the whole 3D point cloud, but rather by dividing the one big problem into smaller ones by considering the trajectory in separate segments. Second, similar local corrections are introduced on a vertical 2D plane. Third, a new local filtering paradigm is introduced to compute features from the time-of-flight measurements to make the previous two steps computationally tracktable. The previously used 6 DoF semi-rigid SLAM implementation is used to bring the trajectory estimates close to the actual trajectory. Results are presented in Section 3, with data gathered using our existing platform, VILMA, and a survey-grade terrestial laser scanner[1] (TLS) is used to provide reference results. Discussion is in Section 4, and Section 5 concludes the paper.

* Corresponding authors at: Institute of Measuring and Modeling for the Built Environment, Aalto University, P.O. box 15800, 00076 Aalto, Finland (V.V. Lehtola).
E-mail address: ville.lehtola@iki.fi (V.V. Lehtola).

[1] Faro Focus 3D 120.

http://dx.doi.org/10.1016/j.isprsjprs.2016.09.004
0924-2716/© 2016 International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS). Published by Elsevier B.V. All rights reserved.

1.1. Related work

Mobile mapping systems deliver 3D data while moving profilers along a trajectory. The trajectory can be recovered by measuring the system motion, combined with extrinsic calibration, i.e., the process of estimating the position and orientation parameters of a sensor system. Recent approaches to calibration of laser scanners, which is also called boresight calibration (Skaloud and Lichti, 2006; Rieger et al., 2010) include statistical methods using sophisticated error functions (Underwood et al., 2009; Sheehan et al., 2011). However, VILMA does not need extrinsic calibration.

The other way to recover the trajectory is intrinsic. Then the focus is on the manipulation of the computational trajectory, commonly known as simultaneous localization and mapping (SLAM). SLAM has long roots in the history of robotics. Approaches include EKF-SLAM (Dissanayake et al., 2000), FastSLAM (Montemerlo et al., 2002), FastSLAM 2.0 (Montemerlo and Thrun, 2007), and GraphSLAM (Thrun and Montemerlo, 2006), including early approaches to 3D mapping (Thrun et al., 2000). Of these, GraphSLAM uses sparse matrices to represent a graph of observation interdependencies, i.e., as extended incidence matrix, and in this sense, its relative in computer vision can be thought to be the bundle adjustment, and its variations, referred broadly to as structure from motion techniques (see e.g. Triggs et al. (2000), and refs there-in). Acquisition of different 3D point clouds from the latter established the need for the well-known iterative closest point (ICP) algorithm that was developed by Besl and McKay (1992), Chen and Medioni (1992), and Zhang (1994). Meanwhile in the robotics community, Lu and Milios (1994) came up with its 2D variant. The input was scan data acquired by a robot with a horizontally mounted profiler, i.e., a 2D safety scanner. Based on this 2D ICP, Lu and Milios (1997) presented an ICP-like GraphSLAM solution, and its extension to 3D scans and poses with six degree of freedom was performed by Borrmann et al. (2008) and Nüchter et al. (2010).

Recent development that begun with approaches that cut the trajectory into segments and performed some globally consistent scan matching on the segments (Stoyanov and Lilienthal, 2009; Bosse and Zlot, 2009), has led towards continuous-time SLAM (Anderson and Barfoot, 2013; Anderson et al., 2014). However, the realization of these often focuses on cameras with rolling shutters instead of event-based vision sensors (Mueggler et al., 2011). Furthermore, the methods of Alismail et al. (2014) and Anderson et al. (2015) are designed for scanners that differ from the ones intended for VILMA, both in terms of the data rate and the modality of the data. To our best knowledge no other intrinsic localization solutions exist. Otherwise, the work closest to ours may be the one of Zhang and Singh (2014) on Lidar Odometry and Mapping (LOAM) that employs external angular measures to perform localization.

1.2. Initial straight trajectory estimate (1 DoF)

The localization of the laser data requires a successful reconstruction of the sensor trajectory. The trajectory $j(t)$ is time-dependent with six degrees of freedom, namely, three from location and three from orientation. We write out

$$j(t) = [\theta(t), \psi(t), \phi(t), x(t), y(t), z(t)]^T \tag{1}$$

where θ is the pitch, ψ is the roll, and ϕ is the yaw angle. Time is denoted by t. Without any reference coordinate system, the successful reconstruction of the trajectory requires that these degrees of freedom are eliminated. Previously, this was done for a holonomic system in one dimension (1D) (Lehtola et al., 2015). We briefly outline this solution here.

To capture a 3D environment with a 2D laser scanner, the scanner has to be rotated about at least one axis. The 2D scanner is mechanically attached onto a round platform, see Fig. 1, so that it can only rotate about one axis of rotation, namely θ. Therefore rotational degrees of freedom are reduced by two, i.e., φ and ψ are constant. The platform does not slip against the floor, and so x, y, and z all become direct functions of θ. This assumption gets relaxed during the SLAM step of Section 2.3.

The main angle of rotation $\theta(t)$ is the path parameter that describes the scanner trajectory, and obtaining it, as follows, solves localization in 1D. The scanner sits on the hypotenuse at a distance of $R_0 - R_1$ from VILMA's central axis, where $R_1 = 0.13$ m, and $R_0 = 0.25$ m is the radius of the metal disk. Assuming that the floor is flat, simple trigonometry is employed to write

$$\theta = \arccos(R_0/d) \tag{2}$$

where $d = d_m + (R_0 - R_1)$, and d_m is the minimum measured distance to the floor over one full 2D circle observation, a so-called slice. Considering the minimum distance to the floor, the scanner's position on disk radius varies between two values, depending on whether the scanner is upside down, Eq. (2), or upside up, in which case $\theta = \pi - \arccos(R_0/d)$, with $d = R_0 + \cos 27.5\,\mathrm{deg}(d_m - R_2)$, and $R_2 = 0.42$ m. Here, the 27.5deg is half of the dead angle of the scanner. The angle θ is incremented by 2π for each cycle that the platform rolls. Each time the 2D scanner is perpendicular towards the floor (PTF), $\theta(t) = \pi + 2\pi n$, $n = 0, 1, 2, \ldots$, the scanning distance reduces to the minimum R_1. We call this a PTF-observation, and keep track of these occurrences in the laser data series obtaining a time series. The PTF observation is robust to error, since data points from a large field of view can be used to interpolate the floor point precisely below the sensor. Also, stochastic errors in PTF observations do not cumulate with time as long as the no-slip condition with the floor applies. Once $\theta(t)$ is obtained, the coordinate transform for the 2D sensor data (X, Z) is obtained considering the trajectory of a contracted cycloid,

$$\begin{cases} x = X \\ y = R_0\theta + (R_0 - R_1)\sin\theta + \sin(\theta)Z\,, \\ z = R_0 + (R_0 - R_1)\cos\theta + \cos(\theta)Z \end{cases} \tag{3}$$

where (x, y, z) are the coordinates of the resulting 3D point cloud, and the platform trajectory is

$$\vec{j}_{1D}(t) = \begin{pmatrix} 0 \\ R_0\theta(t) \\ 0 \end{pmatrix}. \tag{4}$$

2. Back to the six degrees of freedom

In 1D, the localization is done knowing only the initial and the current state of the system, as is obvious from Eq. (4). In two or more dimensions, however, the trajectory reconstruction by path integration requires accurate measuring of the position all along the path. Therefore, we use 6 DoF SLAM. Still, this requires a relatively good initial trajectory estimate, and next we concentrate on how to obtain that. The overall algorithm, and the contribution of this paper, are illustrated in Fig. 2.

Each time-of-flight measurement, or point, is connected to the trajectory, at a position $s(t)$ from where the measure was made at time t. These positions are discretized with respect to time into N slices, where each slice contains one 2D profile, i.e., points from the full rotation of the mirror of the scanner. From the 1D solution of Eq. (4), we have the trajectory divided into parts of length

$$l_n = (\theta_n - \theta_{n-1})R_0, \tag{5}$$

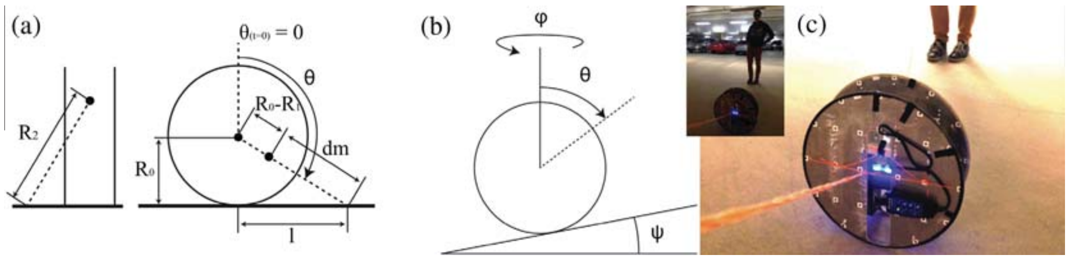

Fig. 1. (a) Technical scheme displaying the constants and the pitch angle θ. (b) Angles describing VILMA's motion on the trajectory, ψ and ϕ. (c) VILMA consists of a FARO Focus3D mounted between two circular plates. Acceleration and steering is done by with the ropes shown in the image.

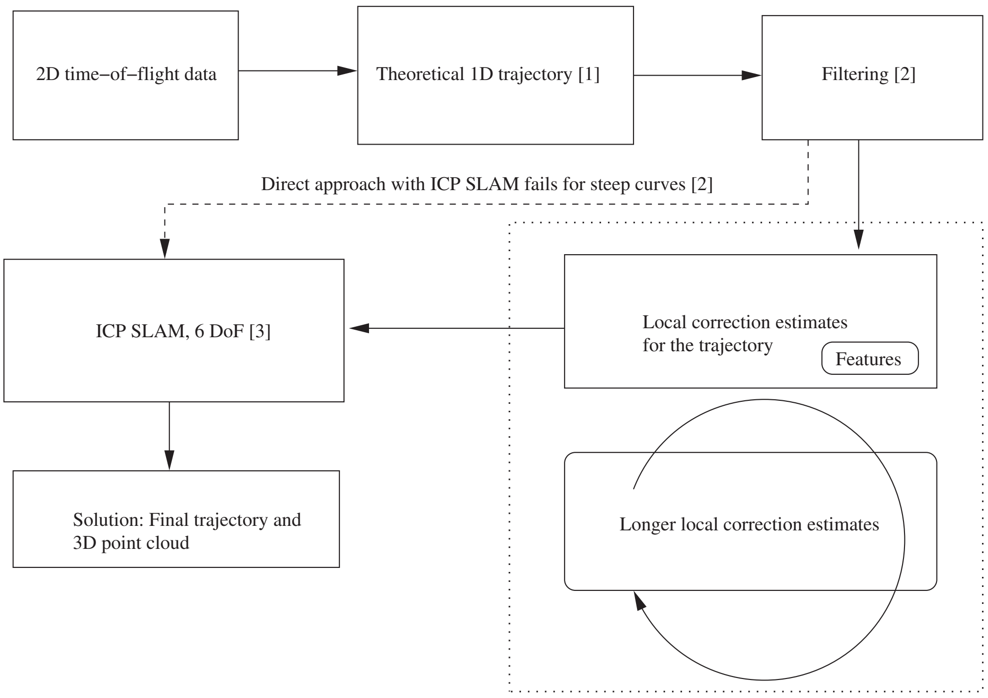
2D time–of–flight data
Theoretical 1D trajectory [1]
Filtering [2]
Direct approach with ICP SLAM fails for steep curves [2]
ICP SLAM, 6 DoF [3]
Local correction estimates for the trajectory
Features
Longer local correction estimates
Solution: Final trajectory and 3D point cloud

Fig. 2. The algorithm pipeline. The previous work [1] and [3] is introduced in Sections 1.2 and 2.3, respectively. In addition, [2] refers to Section 2.4 and Lehtola et al. (2016). The contribution of this paper is surrounded with the dotted line.

where the value for θ_n is obtained from Eq. (2) for each slice n, and $n = 1, 2, \ldots, N$. Hence, we have expressed the knowledge on how far the scanner has traveled in terms of slice-to-slice distances.

What remains to be solved is how to rotate the distances $\{l_n\}$ of Eq. (5) to approximate the underlying physical trajectory, see Fig. 3. Specifically, the horizontal ϕ, and the vertical ψ angles are unknown. For the trajectory, we have

$$\vec{j}(t_N) = \sum_{n=1}^{N} \bar{\mathbf{R}}_n \vec{l}_n, \tag{6}$$

where $\vec{l}_n = (0\ l_n\ 0)^T$, and the rotation matrices are written as

$$\bar{\mathbf{R}}_n = \prod_{i=1}^{n} \mathbf{R}_{\phi,i}\mathbf{R}_{\psi,i}, \tag{7}$$

where

$$\mathbf{R}_{\phi,i} = \begin{pmatrix} \cos\phi_i & -\sin\phi_i & 0 \\ \sin\phi_i & \cos\phi_i & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{R}_{\psi,i} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi_i & -\sin\psi_i \\ 0 & \sin\psi_i & \cos\psi_i \end{pmatrix}.$$

This formulation allows the separate solving of each of the ϕ_i, while ψ_i are kept fixed, and vice versa. We shall return to this later. Note that the angles are solved in a chronological order, and that $\bar{\mathbf{R}}_n$ is a rotation matrix that keeps track of the global orientation of the trajectory.

The straightforward approach of computing a pair-wise correlation for slices n and $n-1$ in order to acquire the rotations ϕ_i, and ψ_i is not possible, since these slices have different pitch angles $\theta_n \neq \theta_{n-1}$, see Eq. (5). In other words, the scanner looks at different places. Hence, a larger portion of the trajectory must be examined for rotation computation. We define a local correlation length ξ along the trajectory that determines what time-of-flight measurements correlate between each other. In other words, it is our *control parameter*. For VILMA, ξ is discretized in terms of cycles, with 1 cycle $= 2\pi R_0 \simeq 1.57$ m. Each cycle equals to two full views of the 3D environment, with the exception that in the direction of the dead angle the environment is captured only once. The PTF measures are employed to count the cycles as they are the most accurate ones of those that describe the scanner movement. Formally, the local correlation length

$$\xi(M) = M(2\pi R_0), \quad M = 1, 2, 3 \ldots,$$

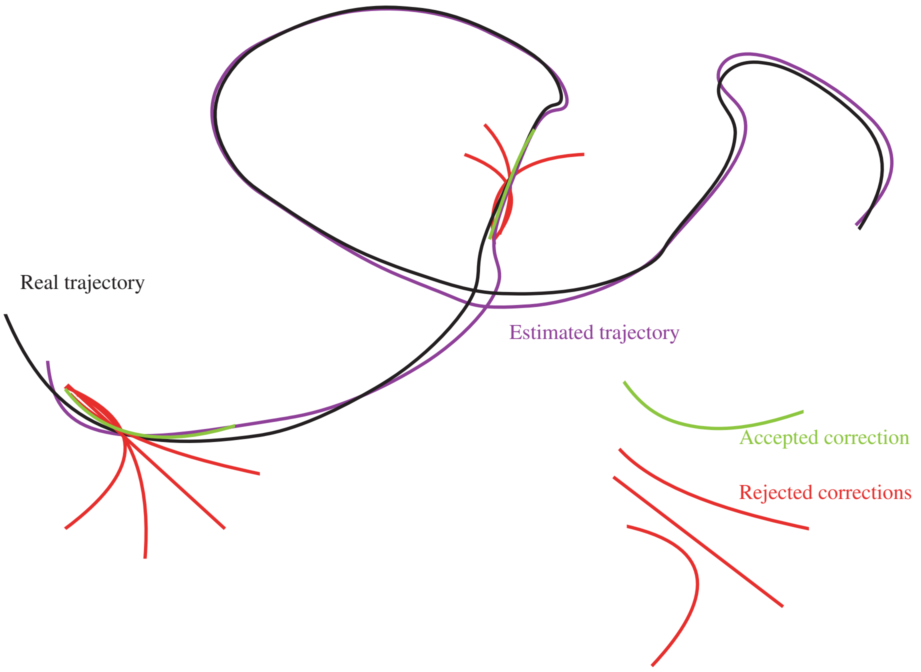

Fig. 3. Illustration of horizontal corrections being introduced to produce a curve-piece estimate of the real trajectory. The estimate (in magenta) is constructed from the accepted (green) corrections. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

i.e., a multiple of subsequent cycles. In Section 3, we examine how changing ξ as a function of M affects the accuracy of the local curve approximation. Note that the amount of slices, i.e., the amount of data, for a given local correction is a *self-adapting* quantity, because the platform velocity is not constant. For VILMA, the amount of slices per cycle varies from about 180 to 600.

2.1. Horizontal corrections (2 DoF)

After obtaining the straight trajectory estimate, horizontal corrections are computed locally for each separate cycle rolled by VILMA, i.e., the local correlation length $\xi = \xi(M = 1)$. These local corrections are defined by the yaw angles ϕ_i, which in turn are incrementally cumulated to create the global trajectory of Eq. (6).

In discretizing the turns, the choice for the rotation kernel, or trial function, is of major importance. Theoretically, all variations for the kernel can be covered with, for example, an infinite Fourier series. In practice, however, we want to limit the use of CPU time by selecting only a few simple kernels. First, the length of the kernel is chosen for convenience to be equal to the local correlation length ξ, if not otherwise mentioned. Second, the shape of the kernel is chosen to be a simple curve,

$$x^2 + y^2 = [R_{trial}(\phi_{trial})]^2, \tag{8}$$

where with a little abuse of notation, y is the initial local advancement direction of the platform, as in Eq. (4), and the trial turn radius R_{trial} is a control parameter obtained from

$$R_{trial} = 2\pi R_o/\phi_{trial}. \tag{9}$$

See Fig. 3 for an illustration of the local correction trial curves of Eq. (8). For VILMA, we define a radius for the maximum turn $R_{tmax} = 2.0$ m. Defining such a radius is important, because not limiting the turn angle easily leads into trivial numerical solutions, where the scanner rotates about its starting location.

To determine the best approximation for the local curvature, we want to minimize an energy E that represents the coherence of the local environment as a function of ϕ, namely

$$\arg\min_{\phi} E(\phi, \{r_i\}),$$

where

$$E = \sum_i \sum_{j \in N(i)} |r_i - r_j| \tag{10}$$

is a sum of (k-d tree) nearest neighbor distances, where self-correlation is excluded. We discretize the range $\phi_{trial} \in [-\phi_{max}, \phi_{max}]$ into 11 different values, computing an energy for each, choosing the value that yields the lowest energy, $\min(E)$. Note that here care is taken to preserve the unimetricity of the local comparison between the trial solutions. In other words, all trials for a segment are computed using the same subset of the point cloud so that the energy of Eq. (10) remains a valid metric. Also, note that Eq. (10) employs the $L1$-norm that is known to be robust with respect to outliers.

A greedy binary search is used, after the above procedure and in near vicinity of the best ϕ_{trial}, to obtain a more accurate final value. Here, the computation continues until a minimum angle step of 0.02deg is reached. Finally, the best kernels, i.e. the best ϕ_i, are selected to form an overall trajectory estimate illustrated in Fig. 3. We call this the *curve-piece estimate.*

2.2. Height corrections (3 DoF)

Height corrections differ from horizontal corrections in two fundamental ways. First, in contrast to the horizontal corrections, which may lead into a looped trajectory, the height corrections are globally constrained, $\psi \in [-\pi/2, \pi/2]$. In other words, gravity keeps us from rolling on vertical surfaces. Second, detecting vertical changes is more difficult than detecting horizontal changes. A slope in a 5deg angle is significantly steep, while a typical horizontal turn may be 90deg. Moreover, the data of the environment is typically more spread out in the two horizontal directions than the vertical one, making the slope detection even harder, see Fig. 5. In order to detect a vertical change, one must therefore significantly increase the local correlation length ξ.

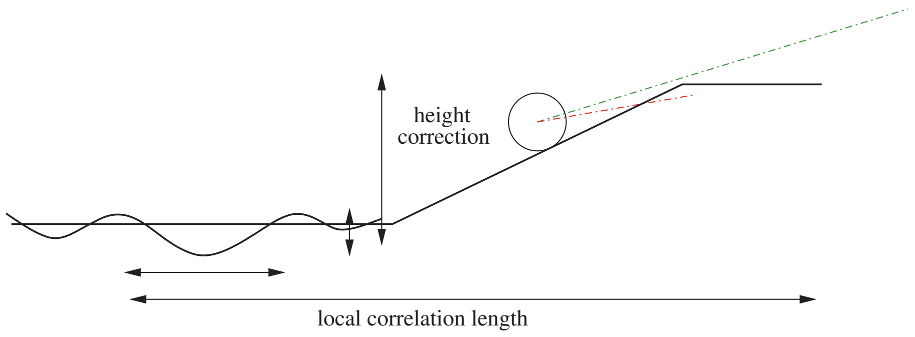

Fig. 4. Schematic for the two vertical correction length scale regimes. Small slopes (on the left) were studied in Lehtola et al. (2016). Here, the scanner is mounting a large slope, which – we propose – can be detected with Eq. (11) using a long local correlation length, $\xi \gg 1$. Note the limited line of sight over the negative angle that is visualized by the red dotted line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

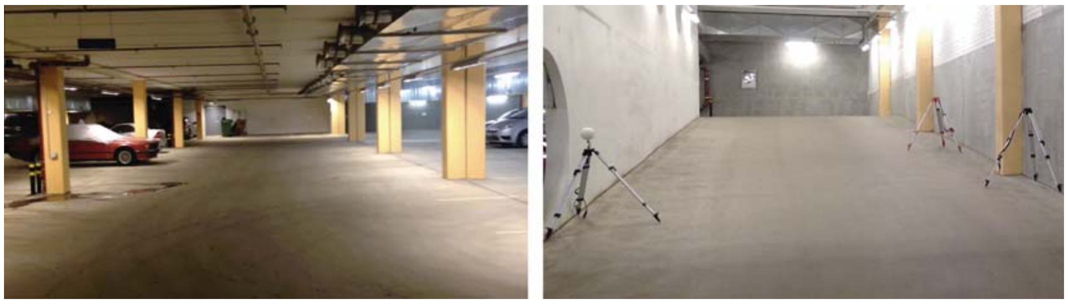

Fig. 5. Images of the indoor environment. Left: On each floor, the visibility is dozens of meters in multiple directions. Right: The ramp is sloped vertically by 5.7 deg, and has a more limited visibility. Tripods for the reference TLS scans are shown.

For purposes of discussion, consider two length scale regimes: one for small height corrections such as minor pits or slopes on the floor, and another for large height corrections, such as major slopes or ramps, see Fig. 4. This helps us to consider the parameter selection associated to the problem of inverse trajectory recovery. In the regime for small corrections, consider a pit of diameter d. The kernel length L_k must be of a comparable size, $L_k \sim d$, to successfully produce a correction estimate. There is, however, a risk of numerical inaccuracy that leads to cumulative vertical drifting, which may also break the global constraint $\psi \in [-\pi/2, \pi/2]$, if the horizontal short curve kernel shown in Fig. 3 is used. Moreover, to counter the lack of data, the local correlation length ξ has to be large (or larger than in the case of horizontal corrections), meaning that L_k and ξ would have to be decoupled. As ψ, and how it should be discretized, is unknown – in addition to L_k and ξ – this results into a rather hard inverse problem with three unknown parameters. Fortunately, it is not necessary to venture into this abyss, since the recovery of small slopes and pits can be achieved by the fine optimization of the trajectory, as in Lehtola et al. (2016), with the 6 DoF semi-rigid SLAM briefed in Section 2.3.

What remains to be dealt with are the large height alterations. Consider now a large slope of a characteristic length D in the regime of large corrections. As, simultaneously, the local correlation length ξ is increased, another problem is introduced. The vertical turn inside the correlated piece of the trajectory needs to be more localized. With a large ξ, we cannot anymore assume that the slope changes uniformly, which questions our previous choice for the kernel. Therefore, we employ the simplest possible kernel, which is an angle located at a pivot point, $p \in \mathbb{R}$. The 2D kernel for (y, z) is written

$$\begin{cases} z = 0, & \text{for} \quad y < |p| \\ z = y \tan\alpha, & \text{for} \quad y \geqslant |p|, \end{cases} \tag{11}$$

where α is the trial vertical angle. For the detection of large slopes, we choose to use only the values $\alpha = 0, -6$ deg, or $+6$ deg. For numerical reasons p is kept inside the middle half of the segment, $1/4 < |p|/\xi < 3/4$.

Now we can ask that if $\xi = 1$ (cycle) would be on the large correction regime for horizontal corrections, how far would the regime lie for vertical corrections. For vertical corrections, difficulties are caused by four reasons.

1. The characteristic turn angle is about 6 deg, being 15$\times$ smaller than the horizontal one of 90 deg.
2. The time-of-flight data is dispersed 10$\times$ more in the horizontal direction, where the range of view is about 35 m, than in the vertical direction, where it is about 3.5 m.
3. Less data is present on the ramp than elsewhere (1.5$\times$), see Table 1.
4. There is a limited line of sight over negative angles, see Fig. 4.

In order to provide a very rough intuitive measure for the reader on how much more difficult the vertical corrections are to compute, we multiply the previous factors together to obtain a total factor of about 225. To counter this challenge of *decrease in correlation*, we propose two measures. The correlation length ξ is increased from 1 cycle to $\sim$20 cycles. Then, the amount of nearest neighbors N to be included in distance metric calculation of Eq. (10) is also increased, from 1 to 10. Multiplying these factors yields 200, which matches the previous very rough measure.

2.3. 6 DoF semi-rigid SLAM

We need a semi-rigid SLAM solution to finalize the recovery of 6deg of freedom for the trajectory, i.e., a SLAM solution that

Table 1
Properties of the data. Discontinuity features are used for the curve-piece trajectory estimate. The *k*-d tree downsampling is made for all SLAM input data using a 10 cm cell size. Ratios for both are displayed to show how much of the original point cloud is conserved.

Data	Loop 1	Loop 2	Loop 3	Ramp 1	Ramp 2
# Cycles	69	64	58	76	70
# Points	146.0 M	161.5 M	120.0 M	150.0 M	106.4 M
# Features	182,400	200,600	144,900	123,500	84,500
Ratio	0.12%	0.12%	0.13%	0.08%	0.08%
#pts, *k*-d tree	6.5 M	7.0 M	5.0 M	5.4 M	3.7 M
Ratio	4.4%	4.3%	4.2%	3.6%	3.4%

transforms the complete trajectory. First, we summarize its basis, a 6 DoF SLAM, which registers 3D point clouds in a globally consistent fashion.

The SLAM is based on the well-known iterative closest points (ICP) algorithm, which minimizes the following error function

$$E(\mathbf{R},\mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N}||\mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t})||^2 \tag{12}$$

iteratively to solve an optimal rotation $\mathbf{T} = (\mathbf{R},\mathbf{t})$, where the tuples $(\mathbf{m}_i, \mathbf{d}_i)$ of corresponding model $\mathbf{M}$ and data points $\mathbf{D}$ are given by minimal distance, i.e., $\mathbf{m}_i$ is the closest point to $\mathbf{d}_i$ within a close limit (Besl and McKay, 1992). Instead of the two-scan-Eq. (12), we look at the *n*-scan case:

$$E = \sum_{j\to k}\sum_{i}\left|\mathbf{R}_j\mathbf{m}_i + \mathbf{t}_j - (\mathbf{R}_k\mathbf{d}_i + \mathbf{t}_k)\right|^2, \tag{13}$$

where *j* and *k* refer to scans of the SLAM graph, i.e., to the graph modeling the pose constraints in SLAM or bundle adjustment. If they overlap, i.e., closest points are available, then the point pairs for the link are included in the minimization. We solve all poses at the same time and iterate as in the original ICP by repeatedly computing closest points and solve the minimization. The derivation of a GraphSLAM method using a Mahalanobis distance that describes the global error of all of the poses:

$$\begin{aligned} W &= \sum_{j\to k}\left(\bar{\mathbf{E}}_{j,k} - \mathbf{E}'_{j,k}\right)^T \mathbf{C}^{-1}\left(\bar{\mathbf{E}}'_{j,k} - \mathbf{E}'_{j,k}\right) \\ &= \sum_{j\to k}\left(\bar{\mathbf{E}}_{j,k} - \left(\mathbf{X}'_j - \mathbf{X}'_k\right)\right)\mathbf{C}_{j,k}^{-1}\left(\bar{\mathbf{E}}'_{j,k} - \left(\mathbf{X}'_j - \mathbf{X}'_k\right)\right). \end{aligned} \tag{14}$$

where $\mathbf{E}'_{j,k}$ is the linearized error metric and the Gaussian distribution is $\left(\bar{\mathbf{E}}_{j,k}, \mathbf{C}_{j,k}\right)$ with computed covariances from scan matching, as given in Borrmann et al. (2008), does not lead to different results (Nüchter et al., 2010). Please note, while there are four closed-form solutions for the original ICP Eq. (12), linearization of the rotation in Eq. (13) or (14) is always required.

Unlike other state-of-the-art algorithms (Stoyanov and Lilienthal, 2009; Bosse and Zlot, 2009), the SLAM is not restricted to purely local improvements. We make no rigidity assumptions, except for the computation of the point correspondences. For processing, we require no explicit model of motion for a vehicle, as it is given initially in the form of the curve-piece trajectory estimate. The semi-rigid SLAM for trajectory optimization works in six DoF, which implies that the trajectory generated by VILMA is also turned into poses with six DoF. The algorithm requires no high-level feature computation, i.e., we require only the points themselves. In this paper, we run SLAM on full time-of-flight data that is outlier-filtered as explained in Section 2.4.

For VILMA, we do not have separate terrestrial 3D scans. In the current state of the art developed by Bosse and Zlot (2009) for improving the overall map quality of mobile mappers in the robotics community, the time is coarsely discretized. This results in a partition of the trajectory into sub-scans that are treated rigidly. Then, rigid registration algorithms, like the ICP and other solutions to the SLAM problem, are employed. Obviously, trajectory errors within a sub-scan cannot be improved in this fashion. Applying rigid pose estimation to this non-rigid problem directly is also problematic since rigid transformations can only approximate the underlying ground truth. When a finer discretization is used, single 2D scan slices or single points result that do not constrain a six DoF pose sufficiently for rigid algorithms.

The mathematical details of our algorithm are given in Elseberg et al. (2013). Essentially, we first split the trajectory into sections and match these sections using the automatic high-precise registration of terrestrial 3D scans, i.e., globally-consistent *n*-scan matching (Borrmann et al., 2008). Here, the graph is estimated using a heuristics that measures the overlap of sections using the number of closest point pairs. After applying globally-consistent scan matching on the sections, the actual semi-rigid matching as described in Elseberg et al. (2013) is applied, using the results of the rigid optimization as starting guess to compute the numerical minimum of the underlying least squares problem. To speed up the calculations, we make use of the sparse Cholesky decomposition by Davis (2005).

For VILMA, the amount of data gathered per distance traveled along the trajectory varies, because the platform velocity is not constant. The Faro Focus 3D operates at 95 Hz, capturing this many slices and a total of ~1 million points per second. Each cycle contains about from 180 to 600 slices. Hence, as input parameters, we choose to conduct a 6 DoF semi-rigid SLAM match between every 300 slices, and use 600 slices to perform each of these.

Similarly to the work of Zhang and Singh (2014), the basics, i.e., 6 DoF SLAM is available as open source in *3DTK – The 3D Toolkit.*[2] The derived semi-rigid part is currently commercialized for industrial applications.

2.4. Filtering

Since the platform movement is not always smooth, the data needs to be filtered for outliers. In Lehtola et al. (2015), filtering was done to improve the visual properties of the final 3D point cloud. Here, in contrast, it is necessary to filter the data before the trajectory computation, which in turn must be done before obtaining the final point cloud. Hence, instead of employing global point cloud properties, the filter needs to rely on the local properties of the measured data.

The local filtering method that some of us introduced in Lehtola et al. (2016) manages to eliminate almost all of the outliers with a total point reduction of mere ~5%. We use it here as well, and explain it briefly. It is based on the concept of support, meaning that among themselves, the time-of-flight measurements must be coherent. Specifically, for each time-of-flight measurement *k*

[2] http://slam6d.sourceforge.net/.

in slice i, the change in range is checked against the previous measurement k in slice $i-1$ and the next k in slice $i+1$. Here, $k \in [1, M]$, where M is the amount of points in one slice, equaling to 8534 for the used Faro 3D scanner. This is called the inter-slice support $\hat{c}_j$. It is cumulated by one for each supporting point. If for both range differences $|\Delta r| > 5.0$ m, i.e., $\hat{c}_j = 0$, then the current point is rejected as an outlier. Next, an intra-slice support c_j check is made by considering 8 nearest neighbors for each point on a slice. Accepted points must have support from at least three neighbors, namely $c_j \geqslant 3$. The support c_j is incremented by one for each pair-wise difference in range measurement $|\Delta r| < 0.15$ m between the point and its neighbor.

2.5. Discontinuity features

We modify the concept of support presented in Section 2.4 to mark significant range discontinuities, or *discontinuity features*, in the environment. Specifically, only those time-of-flight measurements that gain intra-slice support only from either side of their neighborhood are preserved for the curve-piece method. Formally, we first divide the neighborhood N_j of point j so that $N_{j+} = \{j+4, j+3, j+2, j+1\}$ and $N_{j-} = \{j-4, j-3, j-2, j-1\}$. Then we write the discontinuity support condition s_d for point j from N_{j-},

$$s_d(j-) = \begin{cases} 0 & \text{if } \hat{c}_j = 0 \\ 0 & \text{if } c_{j+} \geqslant 1 \\ 0 & \text{if } c_{j-} < 3 \\ 1 & \text{otherwise} \end{cases}, \tag{15}$$

where $\hat{c}_j$ is as in Section 2.4, and

$$c_{j\pm} = \sum_{k \in N_{j\pm}} \Theta_{jk},$$

where in turn the Heaviside function $\Theta_{jk} = 1$, if $|\Delta r_{jk}| < 0.15$ m, and 0 otherwise. The difference of range measures $\Delta r_{jk} = r_j - r_k$. Note that $c_j \geqslant c_{j+} + c_{j-}$ (cf. Section 2.4).

If $s_d(j-) = 1$ or $s_d(j+) = 1$ from Eq. (15), then point j is a discontinuity feature, and it is preserved for the curve-piece method. Otherwise, it is discarded. Intuitively, this leaves us only with points of potential topological interest, for example, those from where a pillar begins to obstruct the view. Then these points can be used to verify whether a trial trajectory improves the result or not. In the case of the pillar, a successfully recovered trajectory should yield a correct representation for that pillar. We shall return to this in Section 3.

We emphasize that the use of features is necessary to make the proposed curve-piece method computationally efficient. Otherwise, the computational load would substantially increase as the amount of points would be 10^3 times higher, see Table 1. Even with k-d tree downsampling, the amount of points would be still about $15\times$ larger, and with a more homogeneous density. We come back to the computational efficiency in Section 3. Finally, note that Eq. (15) relies solely on local data, requiring a cache of three 2D slices to function, or a cache of just one slice if the inter-slice filtering condition is omitted. Therefore, as the size of laser scanners is diminishing (Kostamovaara et al., 2015), this is likely to open new possibilities in on-chip integration of point cloud processing algorithms.

3. Results

To evaluate the proposed method, we gather data by performing three similar but different circular loops with VILMA in an indoor environment that has a sloped floor, see Fig. 5. The start and end positions of the trajectory are intentionally not at the same location, although a closed loop is formed in terms of the 6 DoF semi-rigid SLAM. For vertical correction evaluations, VILMA is steered up a ramp that connects two subsequent floors of the car park. This is done two times to obtain similar, but different, sets of data. In contrast to Lehtola et al. (2015), where the platform was pushed forward, here we steer and pull it by using strings. This allows the level of control required to perform circular movement, but also to climb up the ramp. Conjunctionally, the resulting point cloud is cluttered with data from the two operators performing the steering. As this dynamic clutter is present in all our results, it may be regarded as further evidence towards the robustness of the proposed method. The velocity at which VILMA moves varies a lot, but is typically between 0.3 and 1.0 m/s. For comparison, LOAM scanner is used with a speed of 0.5 m/s (Zhang and Singh, 2014).

3.1. Loops

The three trajectory manipulation phases employed here are visualized in Fig. 6. First, the 1D theoretical trajectory is obtained. Second, the proposed curve-piece estimate is computed using the discontinuity features of Section 2.5. Third, 6 DoF semi-rigid SLAM optimization for the trajectory is conducted employing the full point cloud, filtered as in Section 2.4. Accordingly filtered full point clouds are used also in Fig. 6, with an additional fourth image that shows the reference terrestial laser scanner (TLS) point cloud.

The effect of altering the local correlation length ξ is shown in Fig. 7. The curves on the trajectory are represented with one mode $\xi = 1$ (cycle), and two-mode approximations $\xi = 1$, and then = 5. The notion of approximation follows from the fact that theoretically an infinite sum of modes may be present. In the two-mode approximation, the result trajectory from $\xi = 1$ (cycle) computation is used as input, when the longer $\xi = 5$ (cycles) is used. From Fig. 7 (a), we can see that all of the larger modes improve the result, but are expectedly different from each other. The practical minimum for ξ is about 1 cycle, so that the environment is captured (almost) twice, as discussed in Section 2. In order take longer correlations into account, the algorithm is re-run with $\xi = 4, 5, 6$, and 7 (cycles), and by increasing the number of nearest neighbors from 1 to 10. We found out that if the local correction length ξ is 9 or larger, it is too long for horizontal corrections. In Fig. 7(b)–(d), and for the rest of the paper, we have chosen to use $\xi = 5$ (cycles) for these. The prototype platform is somewhat cumbersome to operate, and to eliminate artifacts originating from setting it in motion, and stopping it, a constant amount of data is cut-off from both ends of the trajectory.

Now, the curve-piece estimate – by definition – has a limited modality and a reduced dimension. Therefore, it cannot by itself recover the underlying physical trajectory, and the use of SLAM is justified. The curve-piece estimate is given to 6 DoF semi-rigid SLAM as input. After convergence, the resulting SLAM trajectories are obtained, see Fig. 7. The convergence is considered in a twofold manner. First, we observe the distance between the two end points of the trajectory R_{ee} as a function of SLAM iterations in Fig. 8. We iterate the trajectory until the end-to-end distance of the trajectory converges, which is enough for a proof-of-concept demonstration. Second, the SLAM processed point clouds are evaluated against TLS reference scans. This is shown for Loop 2 data in Fig. 9. Notably, the accuracy ratio at a 25 m distance is about 1:100 with respect to R_{ee}. The improvement in terms of accuracy with respect to our previous work (Lehtola et al., 2015) is significant, as the method now manages to reproduce pillars as they should be – *straight*, see Fig. 10. From the figure, however, it is clear that the precision of the result points representing the surfaces is still far from the reference result. We select the largest surface, i.e. the floor, to conduct a rough precision evaluation of the result point cloud against a trian-

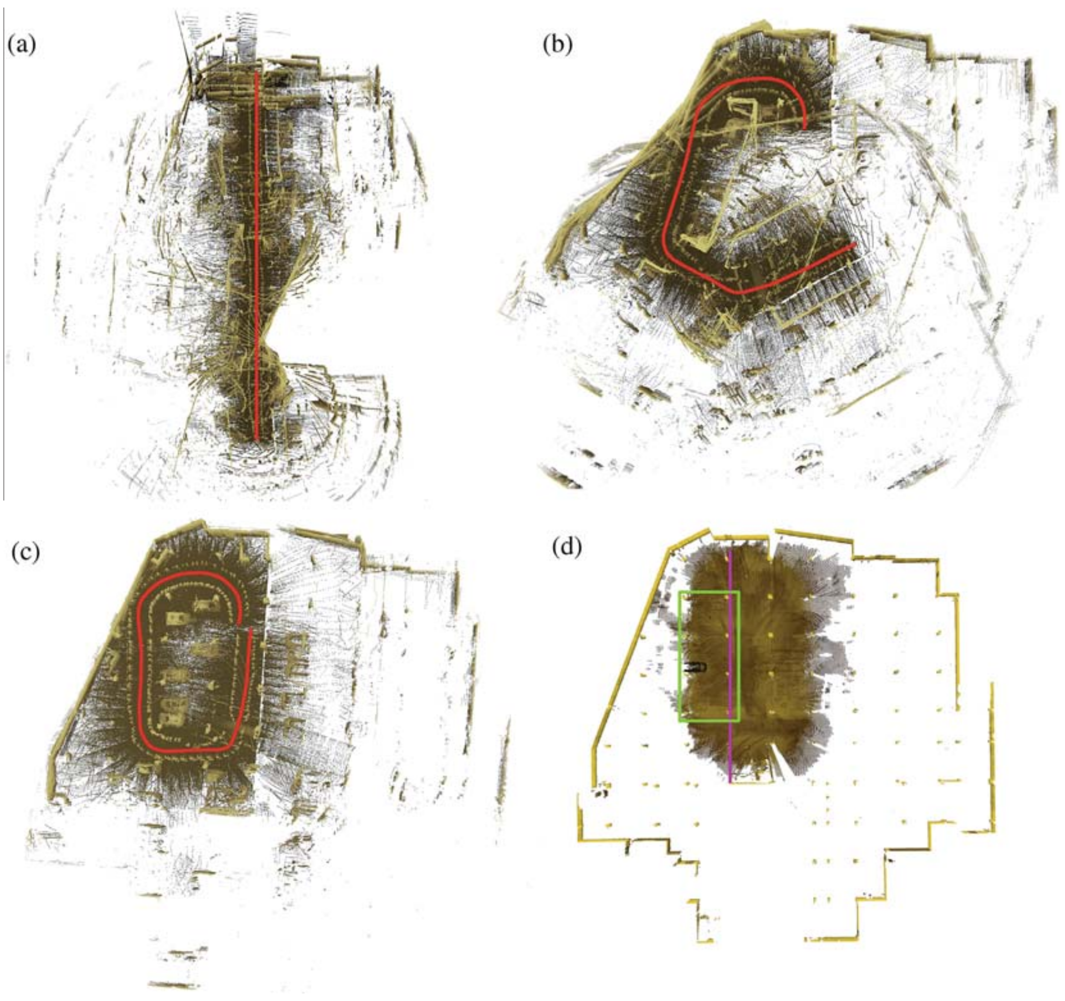

Fig. 6. Full point clouds and red trajectories viewed from the top illustrate the algorithm pipeline as follows. (a) The theoretical 1D solution of Section 1.2. (b) The curve-piece estimate of Section 2.1 after the first phase with $\xi = 1$ cycles. (c) After the second curve-piece estimate phase with $\xi = 5$ cycles, and 6 DoF SLAM of Section 2.3, the result very much resembles (d) the reference TLS scan. In (d), the area surrounded by the green box is enlarged in Fig. 9, and the point cloud profile at the location of the purple line is shown in Fig. 10. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

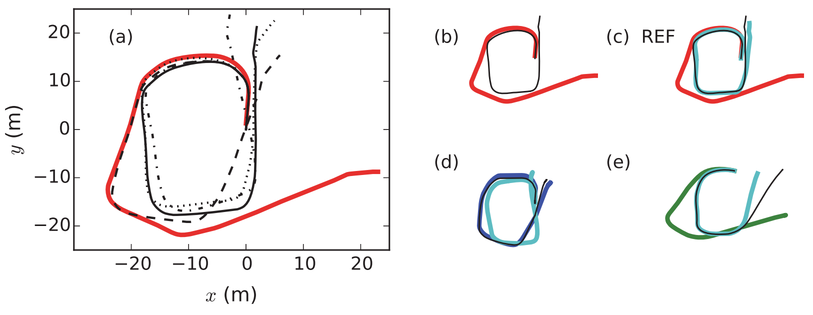

Fig. 7. Approximations of physical trajectories based on curve-piece fitting computations. (a) Obtained trajectories for Loop 1 for different local correlation lengths. First, $\xi = 1$ (red). Then this trajectory is reprocessed with $\xi = 4$, 5 (solid black), 6, and 7. The $\xi = 1$ (red) and $\xi = 5$ data is reproduced in (b), and (c) where also the final (reference) trajectory obtained from 6 DoF semi-rigid SLAM is shown (cyan line). (d,e) Same as (c) but with Loop 2 and Loop 3 data, respectively. See text for details. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

gulated mesh obtained from the reference point cloud. The selected floor area is the one shown in Fig. 9. The obtained average point to mesh distance is 0.04 m, which is similar precision to the one reported in Lehtola et al., 2016, see Fig. 9. Nevertheless, the measuring geometry, and the algorithmic pipeline can likely be improved for better accuracy and precision after a more practical measurement unit is designed. This is what we plan to do in the future.

Snapshot from with-in the point cloud of Loop 1 is illustrated in Fig. 11. The shape of the environment in Fig. 11(a) seems visually preserved in (c) showing the point cloud after the feature filtering of Section 2.5. This reduced point cloud is used for the fast

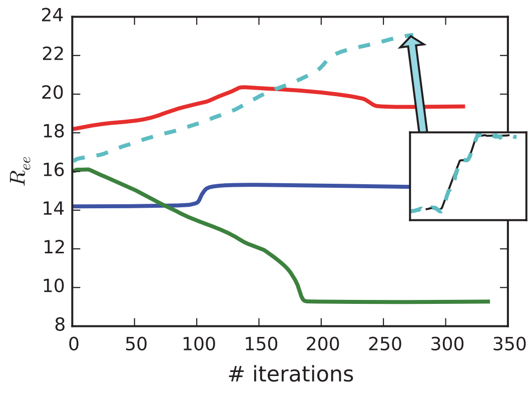

Fig. 8. The end-to-end distance of the trajectory R_{ee} as a function of SLAM iterations. Suitable convergence is reached once the flat line begins. Loop 1 (red), 2 (blue), and 3 (green) data are represented by same colors than in Fig. 7. Dashed cyan line displays Ramp 1 data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

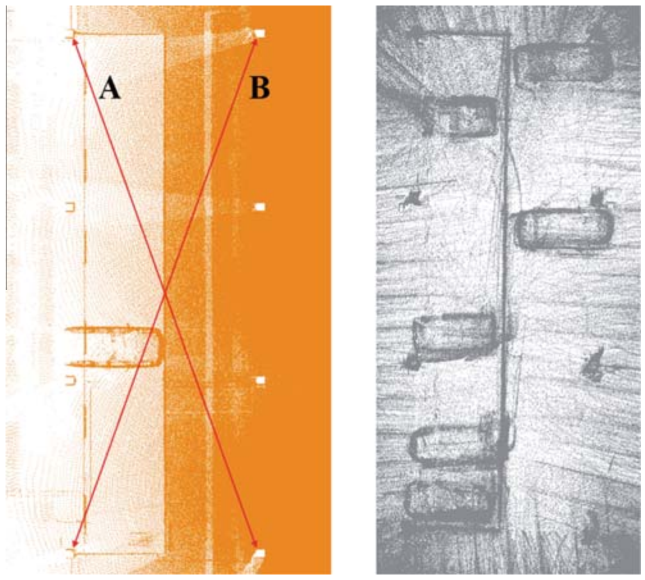

Fig. 9. Top view comparison between the reference TLS point cloud (left), and the result point cloud (right). The overall consistency of the latter is evaluated through distance measures A and B each taken between the middle points of two pillars. Distance A is 24.93 m (TLS) and 25.24 m (result), and distance B is 24.94 m (TLS) and 24.88 m (result). The accuracy ratio is about 1:100. The data were taken on different days, so the cars have moved. This has no significance, since we do the comparison using only the static part of the environment.

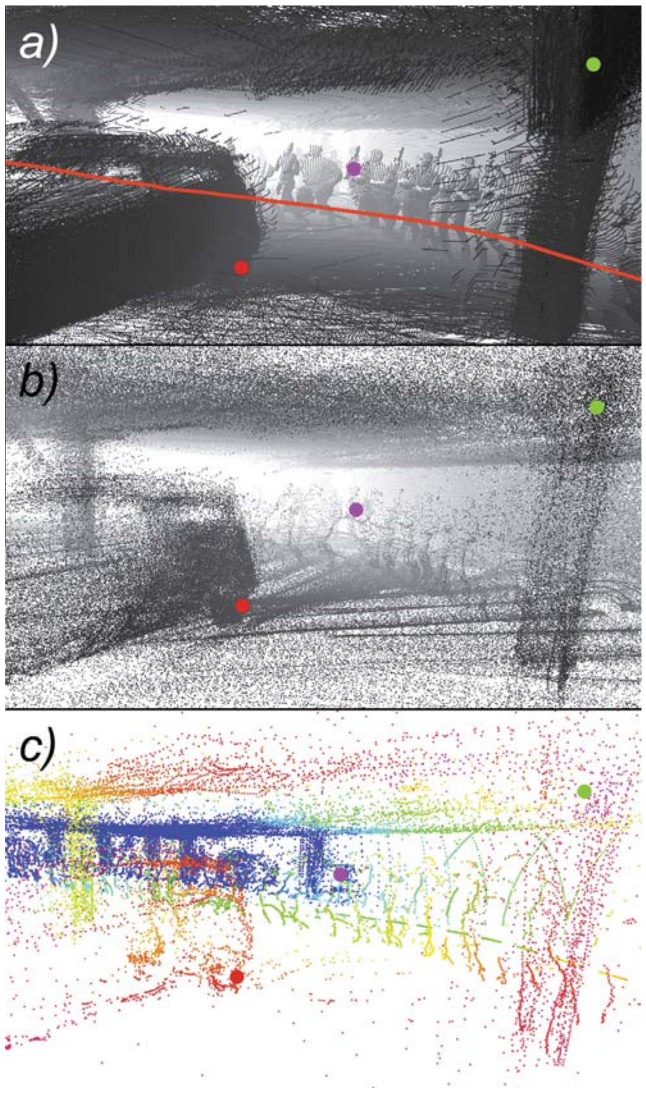

Fig. 11. Snapshots from Loop 1 point clouds after 6 DoF semi-rigid SLAM. (a) Full point cloud. (b) A downsampled representation using k-d tree with 10 cm cell. (c) Point cloud of discontinuity features shows the boundaries of the objects, and is colored for visualization purposes. Note how the point cloud changes for the rear wheel of a car marked with a red dot, for the pillar marked with a green dot, and for the walking authors operating the platform marked with a magenta dot. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

computation of the curve-piece estimate. We discuss its efficiency in Section 3.3.

3.2. The ramp

The ramp is sloped vertically in a 5.7 deg angle. We pretend that this is unknown to us, and say that we want to divide slopes by computation into three categories: −6deg, 0deg, +6deg. This should yield a close enough initial approximation for 6 DoF semi-rigid SLAM to recover the original physical trajectory, while still being robust enough against false positives. The results are shown in Fig. 12. The first upward angle is very robustly discovered, being almost indifferent to the parameter selection. However, the next

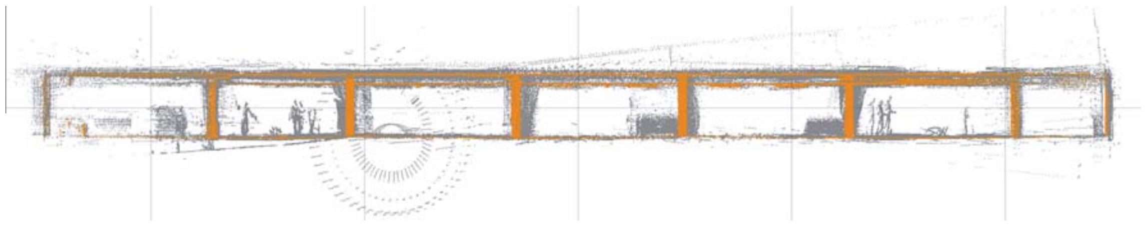

Fig. 10. Profile comparison between the reference TLS point cloud (orange), and the result point cloud (gray). Floor, ceiling, and pillars are visually fairly well matched. Although most part of the point cloud is correctly aligned, there are some artifacts, such as the round circles visible near the third pillar from the left and a sloped ghost floor nearby, which both are most probably a consequence from our prototype having been originally designed for straight, but not circular, movement.

three angles are entwined into one long correlated segment, resulting into one or zero kernel responses, i.e. angles; refer to Eq. (11). Whether one or zero responses occur depends on the choice of parameters and the data itself. In Fig. 12(a), the triple-angle results into a single negative angle turn for ξ in between 16 and 24 cycles. The neighborhood for the energy metric of Eq. (10) is augmented from 1 to 10 neighbors for increased robustness, as discussed in Section 2.2. In (b), the triple-angle results into a single positive angle turn for $\xi = 16$ cycles. The length of the correlated segment is visualized with the magenta plot. The difficulty here is paradoxal. On one hand, to make vertical angle recovery even possible ξ needs to be large. On the other hand, then it is easily so large that it makes subsequent angle detection impossible, if these angles are close to each other. Nevertheless, the accuracy of the trajectory estimate with $\xi = 16$ and = 24 for Ramp 1 data suffices for the 6 DoF semi-rigid SLAM, which then succeeds in discovering the underlying vertical floor profile, see the cyan dotted line in Fig. 12(a). The resulting 3D point cloud is shown in Fig. 13.

We have managed to intrinsically localize a 2D laser scanner on a 2D plane embedded in a 3D space – with one exception, see Fig. 13. The SLAM processed ramp reconstruction contains an artifact, namely a horizontal notch in the middle of the slope. This is formed due to concurrent weaknesses in the processing pipeline. First, during the data capture VILMA was unintentionally rotated on-spot and thus slipped against the ground. The curve-piece estimate cannot recover this, since the computational radius for maximum turn R_{tmax} is limited. Finally, the SLAM algorithm that would likely fix this problem is stuck, because the trajectory segments from different floors incorrectly interact with each other, and prevent the correction. The segment interaction happens in the semi-global matching phase through the SLAM graph, shown in Fig. 14. Currently, the algorithm treats segments with at least 250 point-to-point matches as connected, with each trajectory segment being 600 slices long. Hence, a part of the future work is to develop a way to dissect the points recorded in different floors, or to separate spaces in general, according to their surface normals, for example. Then the semi-global matching, i.e., loop closure, would be executed correctly. Here, the thickness of the ceiling between the two floors is approximately 40 cm, estimated from the reference TLS point cloud. Summing up, the curve-piece method performs rather well, but improvements in SLAM loop closure regarding separate spaces and a new platform design are called for.

3.3. Computational efficiency

For the largest data set of Loop 2, consisting originally of 161 million points, the proposed curve-piece method produces a trajectory estimate in 311 ± 18 s, see Table 2. The two steps to produce the horizontal estimate take ~185 s, regardless of the choice of $\xi \in [2, 9]$ for the second estimation round. For vertical correction, $\xi = 6$ is twice as fast as $\xi = 16$ that produces the best results, showing the effect of the search for the angle pivot point. The increase in the size of the neighborhood in Eq. (10) results into a negligible change in the run time, whether it is 1 or 10. These run times are produced assuming that features from Eq. (15) are

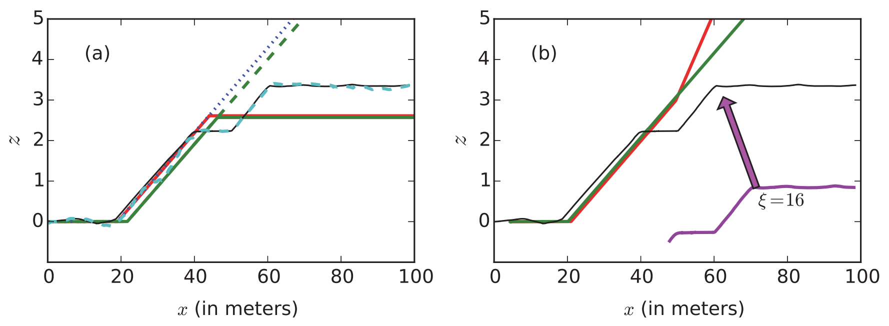

Fig. 12. Vertical component $z(s)$ of the trajectory as a function of the traveled distance s (units in meters), for (a) Ramp 1 and (b) Ramp 2 data. The local correlation length $\xi = 16$ (red), = 24 (green), and = 32 (blue dotted line). Green dotted line is with only one nearest neighbor, while otherwise $N = 10$. Reference trajectory shown with a black solid line is determined from the middle of the ramp using TLS data. The approximation succeeds partly, leaving some vertical angles unrecovered. This is due to long correlation length, visualized by the magenta plot for $\xi = 16$ cycles. Final result from 6 DoF semi-rigid SLAM is shown with a dashed cyan line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

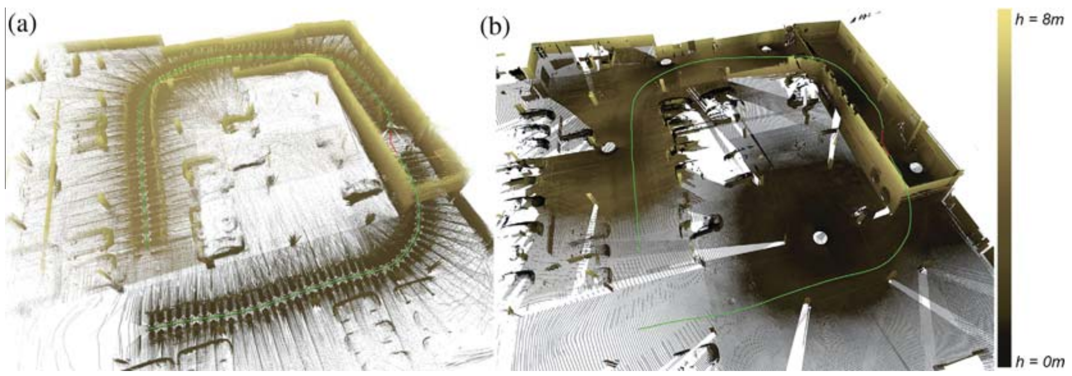

Fig. 13. (a) Ramp 1 point cloud captured with VILMA. An on-spot rotation around z-axis of about 20deg, marked with a red angle, has been corrected manually after obtaining the results of Fig. 12, i.e., after automatic data processing. (b) Reference point cloud from TLS. The trajectory is reproduced for visualization purposes. Roof points have been manually removed from both point clouds. Note that the trajectory ends one floor above its beginning. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

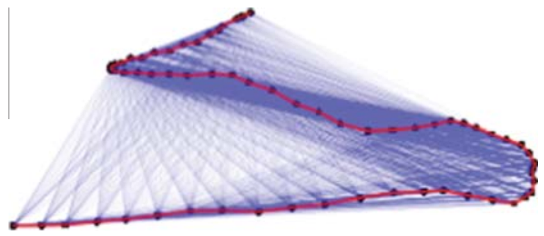

Fig. 14. The connection graph of 6 DoF semi-rigid SLAM for Ramp 1 data at the end of iteration. Black dots represent the 65 locations of trajectory segments, consisting of 600 slices each. Blue lines display the 3920 graph connections. Connections are formed (incorrectly) through the concrete ceiling that is about 40 cm thick. The solid red line is shown to guide the eye.

Table 2
Measured CPU run times in seconds (3.3 GHz Intel Xeon) for the proposed curve-piece method on the largest Loop 2 data, consisting originally 161 M points that are reduced into 200 k discontinuity features. *H* stands for horizontal, and *V* for vertical corrections.

	$H, \xi = 1$	$H, \xi = 5$	$V, \xi = 16$	Total
Run time (s)	98 ± 5	87 ± 4	126 ± 12	311 ± 18

readily available, as it is likely that these can be pre-processed on-the-fly with on-chip integration.

The run time to compute the curve-piece estimate, ~5 CPU minutes, is readily comparable to the time taken to capture the data, which was around 3–4 min. In other words, it performs close to real-time. On the other hand, the 6 DoF semi-rigid SLAM takes about 20–30 CPU minutes per one iteration. Considering the amount of iterations required for convergence shown in Fig. 8, this still poses a big challenge, even though the SLAM is sped up by using multiple processors. Hence, using more than three angular categories for vertical estimates, −6deg, 0deg, +6deg, may be tempting in an attempt reduce the total run time with SLAM, but this is prone to cause problems with robustness, as false positives, i.e., incorrect kernel responses, may appear. In the future, we attempt to open the bottlenecks to achieve real-time trajectory estimation.

4. Discussion

Our starting point was the caveat that the ICP-based SLAM fails to recover steep curves (Lehtola et al., 2016). To obtain the global minimum for an energy metric, given a trajectory *j*, the solution space must be scoured with trajectory trials. The trial set should be chosen so that it is not too sparse, in which case the global minimum may not be found, nor too thick, in which case the computational cost raises unnecessarily. For this purpose, we have proposed the curve-piece method.

We compute the favorability of all trajectory trials with-in certain discretization conditions. This simultaneously means efficiency in computation, since the problem is separated into smaller subproblems. Appropriate care is taken in separation, because the subsample of the point cloud in where the trajectory trials are evaluated must not change, i.e., the comparison must be unimetric.

As all time-of-flight measurements are connected to their respective location on the trajectory, the most convenient way to form the subsamples is to divide the trajectory into segments. Observed points are employed in *k*-d tree nearest neighbor matching to compute a sum of distances that is the proposed trial energy of Eq. (10). It is noteworthy that Eq. (10) does not use any geometric a-priors, such as that the points reside on planes, and that the energy metric employs an outlier resistant L_1 norm. Trial with lowest energy is accepted as the best trajectory estimate. One local solution is computed for each cycle, while keeping track how the pose develops in global system coordinates.

The curve-piece method employs two control parameters for horizontal corrections, the (minimum) local correlation length ξ and the radius for maximum turn R_{tmax}, and two control parameters for vertical corrections, the local correlation length ξ and the slope angle. The first are closely related to the properties of the platform, while the latter are, interestingly, related to the environment. However, the determination of the latter ξ might be automated by setting a lower bound to the amount of terms that must exist in Eq. (10). Also, arguably, the slope angle may be treated as a constant, if the SLAM iteration can recover the physical slope profile. Then the method would consist of only two platform-dependent control parameters. Our approach in intrinsic localization is applicable to any wheeled platform, where the rolling phase of a (no-slip) tire can be mechanically, or through the time-of-flight measurements, connected to the scanner.

Finally, to achieve a computationally effective form for the curve-piece estimate, we have proposed the pre-processing of the data with the local support filter of Eq. (15). Notably, the filter conserves only the range measurements, namely discontinuity features, that are likely the most relevant for localization. However, points that lie on, e.g., a smooth back wall are taken as features, if they are seen around a pillar that blocks the line of sight. If necessary, this can be dealt with another simple range-based condition. The discontinuity feature filter employs two control parameters, one to mark range discontinuities between different scans, and another one for neighboring points inside the same scan. For on-chip integration purposes, these two control parameters should be kept adjustable. The proposed approach is likely to be feasible for a two-orders-of-magnitude higher data capture rate than what is used for real-time computation in Zhang and Singh (2014). Our work includes loop closure with in SLAM, while the one of Zhang and Singh (2014) does not. Future studies should consider the applicability of these discontinuity features as SLAM input to bring this part of the algorithm also to real-time performance.

5. Conclusion

Intrinsic localization allows the pose recovery of a mobile laser scanner without any external sensors such as global navigation satellite systems (GNSS) or an inertial measurement unit (IMU). We have extended the concept of intrinsic localization from the previous theoretical one-dimensional solution onto a 2D manifold that is embedded in a 3D space. Notably, this is a highly non-trivial inverse problem, and therefore we untie the knot in a step-by-step fashion, solving one unknown at a time. First, the position of the scanner is determined with respect to the trajectory length (1D) as in Lehtola et al. (2015). This allows the use of an essential boundary condition – that the trajectory length is fixed – when horizontal turns (2D) and vertical turns (3D) are included with the proposed curve-piece method. Finally, the trajectory is optimized with SLAM (Borrmann et al., 2008) to recover the full six degree of freedom. Pose convergence is examined by point cloud comparisons and observing the end-to-end distance for the trajectory, R_{ee}. The accuracy ratio at a 25 m distance is about 1:100 with respect to it.

We also introduce a new local support filter that conserves only the most meaningful ~0.1% of the time-of-flight measurements. These points reside near range discontinuities by definition, and therefore may be regarded as the most informative points about the environment. This new filter uses a limited cache that allows it to be operated before point cloud registration, also meaning that it can be integrated on chip. Overall, smart filtering reduces data transmission, storage, processing, and work memory requirements, which is especially important in mapping of environments of a large scale. It paves the road for real-time solutions in intrinsic localization.

For future work, we propose an elaborate study of the smart filter properties, specifically for the purposes of fast-processing, completeness, and saliency, but also the on-chip integration of the smart filter. The concept of intrinsic localization is likely to be applicable for various wheeled platforms, and these are to be designed.

Acknowledgments

The authors wish to thank Academy of Finland for funding this research, Grant No. 257755 (VL), CoE-LaSR Grant No. 272195 (VL, J-PV, MV, HH), Strategic Research Council project COMBAT No. 293389 (VL, J-PV, MV, HH), the European Regional Development Fund ‘‘Leverage from the EU 2014–2020” project ‘‘AKAI” No. 301130 (J-PV), and the Aalto University doctoral program (J-PV).

References

Alismail, H., Baker, L.D., Browning, B., 2014. Continuous trajectory estimation for 3D SLAM from actuated lidar. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Anderson, S., Barfoot, T.D., 2013. Towards relative continuous-time SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1033–1040.

Anderson, S., Dellaert, F., Barfoot, T.D., 2014. A hierarchical wavelet decomposition for continuous-time SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Anderson, S., MacTavish, K., Barfoot, T.D., 2015. Relative continuous-time SLAM. Int. J. Robot. Res. (IJRR) 34 (12), 1453–1479.

Besl, P., McKay, N., 1992. A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 14 (2), 239–256.

Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A., Hertzberg, J., 2008. Globally consistent 3D mapping with scan matching. J. Robot. Auton. Syst. (JRAS) 56 (2), 130–142.

Bosse, M., Zlot, R., 2009. Continuous 3D scan-matching with a spinning 2D laser. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09), pp. 4312–4319.

Bosse, M., Zlot, R., Flick, P., 2012. Zebedee: design of a spring-mounted 3-d range sensor with application to mobile mapping. IEEE Trans. Robot. 28 (5), 1104–1119.

Chen, Y., Medioni, G., 1992. Object modelling by registration of multiple range images. Image Vis. Comput. 10 (3), 145–155.

Davis, T.A., 2005. Algorithm 849: a concise sparse Cholesky factorization package. ACM Trans. Math. Softw. 31 (4), 587–591.

Dissanayake, G., Durrant-Whyte, H., Bailey, T., 2000. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00), vol. 2.

Elseberg, J., Borrmann, D., Nüchter, A., 2013. Algorithmic solutions for computing accurate maximum likelihood 3D point clouds from mobile laser scanning platforms. Rem. Sens. 5 (11), 5871–5906.

Kaul, L., Zlot, R., Bosse, M., 2016. Continuous-time three-dimensional mapping for micro aerial vehicles with a passively actuated rotating laser scanner. J. Field Robot. 33 (1), 103–132. http://dx.doi.org/10.1002/rob.21614.

Kostamovaara, J., Huikari, J., Hallman, L., Nissinen, I., Nissinen, J., Rapakko, H., Avrutin, E., Ryvkin, B., 2015. On laser ranging based on high-speed/energy laser diode pulses and single-photon detection techniques. IEEE Photon. J. 7 (2), 1–15.

Lauterbach, H.A., Borrmann, D., Heß, R., Eck, D., Schilling, K., Nüchter, A., 2015. Evaluation of a backpack-mounted 3D mobile scanning system. Rem. Sens. 7 (10), 13753–13781.

Lehtola, V.V., Virtanen, J.-P., Kukko, A., Kaartinen, H., Hyyppa, H., 2015. Localization of mobile laser scanner using classical mechanics. ISPRS J. Photogram. Rem. Sens. 99 (0), 25–29. URL http://www.sciencedirect.com/science/article/pii/S0924271614002585 .

Lehtola, V.V., Virtanen, J.-P., Rönnholm, P., Nüchter, A., 2016. Localization corrections for mobile laser scanner using local support-based outlier filtering. ISPRS Ann. Photogram. Rem. Sens. Spatial Inform. Sci. http://dx.doi.org/10.5194/isprs-annals-III-4-81-2016.

Liu, T., Carlberg, M., Chen, G., Chen, J., Kua, J., Zakhor, A., 2010. Indoor localization and visualization using a human-operated backpack system. In: 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN). IEEE, pp. 1–10.

Lu, F., Milios, E., 1994. Robot pose estimation in unknown environments by matching 2D range scans. In: IEEE Computer Vision and Pattern Recognition Conference (CVPR '94), pp. 935–938.

Lu, F., Milios, E., 1997. Globally consistent range scan alignment for environment mapping. Auton. Robots 4 (4), 333–349.

Montemerlo, M., Thrun, S., 2007. FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics. Springer.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al., 2002. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: AAAI/IAAI, pp. 593–598.

Mueggler, E., Gallego, G., Scaramuzza, D., 2011. Continuous-time trajectory estimation for event-based vision sensors. In: Proceedings of Robotics Science and Systems (RSS).

Nüchter, A., Elseberg, J., Schneider, P., Paulus, D., 2010. Study of parameterizations for the rigid body transformations of the scan registration problem. J. Comput. Vis. Image Underst. (CVIU) 114 (8), 963–980.

Rieger, P., Studnicka, N., Pfennigbauer, M., 2010. Boresight alignment method for mobile laser scanning systems. J. Appl. Geodesy 4 (1), 13–21.

Sheehan, M., Harrison, A., Newman, P., 2011. Self-calibration for a 3D laser. Int. J. Robot. Res. 31 (5), 675–687.

Skaloud, J., Lichti, D., 2006. Rigorous approach to bore-sight self-calibration in airborne laser scanning. ISPRS J. Photogram. Rem. Sens. 61 (1), 47–59.

Stoyanov, T., Lilienthal, A.J., 2009. Maximum likelihood point cloud acquisition from a mobile platform. In: Proceedings of the IEEE International Conference on Advanced Robotics (ICAR '09), pp. 1–6.

Thrun, S., Burgard, W., Fox, D., 2000. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). IEEE, San Francisco, CA.

Thrun, S., Montemerlo, M., 2006. The graph slam algorithm with applications to large-scale mapping of urban structures. Int. J. Robot. Res. (IJRR) 25 (5–6), 403–429.

Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W., 2000. Vision algorithms: theory and practice. In: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings. Springer, Berlin Heidelberg, pp. 298–372 (Chapter Bundle Adjustment — A Modern Synthesis).

Underwood, J.P., Hill, A., Peynot, T., Scheding, S.J., 2009. Error modeling and calibration of exteroceptive sensors for accurate mapping applications. J. Field Robot. 27 (1), 2–20.

Vosselman, G., 2014. Design of an indoor mapping system using three 2D laser scanners and 6 DOF SLAM. ISPRS Ann. Photogram. Rem. Sens. Spatial Inform. Sci. 2 (3), 173.

Zhang, J., Singh, S., 2014. LOAM: lidar odometry and mapping in real-time. In: Proceedings of Robotics Science and Systems (RSS). Berkeley, CA, USA.

Zhang, Z., 1994. Iterative point matching for registration of free-form curves. Int. J. Comput. Vis. (IJCV) 13 (12), 145–155.