MC2SLAM: Real-Time Inertial Lidar Odometry Using Two-Scan Motion Compensation

Frank Neuhaus(✉), Tilman Koß, Robert Kohnen, and Dietrich Paulus

University of Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany
{fneuhaus,tkoss,rkohnen,paulus}@uni-koblenz.de

Abstract. We propose a real-time, low-drift laser odometry approach that tightly integrates sequentially measured 3D multi-beam LIDAR data with inertial measurements. The laser measurements are motion-compensated using a novel algorithm based on non-rigid registration of two consecutive laser sweeps and a local map. IMU data is being tightly integrated by means of factor-graph optimization on a pose graph. We evaluate our method on a public dataset and also obtain results on our own datasets that contain information not commonly found in existing datasets. At the time of writing, our method was ranked within the top five laser-only algorithms of the KITTI odometry benchmark.

1 Introduction

Building three-dimensional maps of the world is commonly considered an essential prerequisite for autonomous robots and cars. However, there are other applications where the motion patterns are less constrained and that thus call for very general and robust methods. Examples come from the field of surveying where the goal is to quickly map difficult-to-reach areas using hand-held devices, or applications in disaster response where the goal is to quickly perform damage- or situation assessment. We propose a generic approach to LIDAR odometry[1]—a crucial requirement for full SLAM—that supports all of these use-cases, including automotive and hand-held use. For all of our tests, we use different types of Velodyne LIDAR sensors, which we generically call multi-beam LIDARs (MBL); revolutions of the sensor are generically referred to as *sweeps*.

We use *two* consecutive laser sweeps to estimate the motion during the first sweep. In a single step, our algorithm registers the sweeps against a local map and performs motion compensation, which means that the sweeps are undistorted so

[1] Our implementation is also able to close loops, but to maintain focus of this work, we decided to focus on the odometry part of the SLAM problem, whose accuracy is essential to obtain accurate maps—even when loops are being closed.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-12939-2_5) contains supplementary material, which is available to authorized users.

T. Brox et al. (Eds.): GCPR 2018, LNCS 11269, pp. 60–72, 2019.
https://doi.org/10.1007/978-3-030-12939-2_5

that the ego-motion of the device is taken into account. Combination of these steps ensures that motion compensation quality is on par with the quality of the odometric computations.

To cope with erratic, high-frequency motion—particularly in cases where the system is carried by a human—we bootstrap the motion compensation with predictions based on IMU data. Conversely, motion compensation results are fused with raw inertial measurements by means of factor-graph-based optimization which refines poses and estimates IMU biases. An additional benefit of this fusion is the correct alignment of the map's gravitational vector with that of the real world. The trajectory is automatically leveled, virtually eliminating two dimensions in which the trajectory could otherwise have drifted, namely roll and pitch. Our IMU integration is tight because estimated biases and velocities are used for motion prediction. Results of our algorithm are shown in Fig. 1.

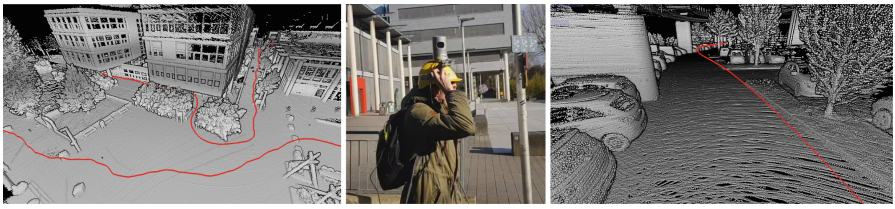

Fig. 1. Results on our own dataset (left), recorded with a head-mounted Velodyne LIDAR (middle). Right: Result on a dataset recorded on a car.

Our contributions in this paper are:

- We describe a novel way to motion-compensate sweeps and show how to tightly integrate an IMU into the system.
- We describe a local map data structure that approximates a Poisson disk downsampling. Despite its simplicity, we show that it yields results equivalent to previous work (see Sect. 5) while being computationally more efficient.
- We show that our method is fast and performs well both on the highly competitive KITTI odometry dataset and on our own challenging datasets recorded with a head-mounted MBL.

2 Related Work

The LOAM-method [21,22,24] proposed by Zhang and Singh is related to our approach. It performs LIDAR odometry only, separating the problem into *odometry* and *mapping*. Within the former, the current sensor sweep is matched against the previous—already motion-compensated—laser sweep only. Motion compensation is performed using a variant of the ICP algorithm that assumes linear motion during the sweep. Instead of exclusively using point-to-plane features that are relatively common within the scan-matching community [15], Zhang and

Singh additionally define so-called *sharp* features. Points are classified as either sharp or planar using a 'smoothness'-metric. Sharp features are then incorporated as point-to-line errors into an ICP error function. The mapping part of their algorithm matches the now-undistorted point cloud against a local map, i. e. is a temporally windowed accumulation of a number of recent point clouds.

Zhang and Singh also propose extensions of their approach to the visual domain [20,23]. This combination is beyond the scope of our work though, since we are currently focusing on the laser domain only.

Bosse et al. [4] present a hand-held 3D laser mapping system based on a spring-mounted 2D LRF whose motion is characterized by a motion model, and an IMU. Their trajectory is sampled at discrete intervals but interpolated in between, making it continuous. Instead of pre-integrating inertial measurements—which is what we propose—they directly integrate individual IMU measurements into least-squares optimization. The underlying assumption is that the trajectory is smooth, which is an assumption we do not have to make.

Moosmann and Stiller [16] as well as Deschaud [6] motion-compensate Velodyne scans based on the previous registration step without any IMU, limiting applications to relatively low-frequency, smooth motion that does not change much between two consecutive laser scans. Both approaches also propose matching the undistorted scan against a local map. Deschaud's approach represents the local map as an implicit moving least squares surface and uses only point-to-plane features selected by a criterion that aims to constrain all six degrees of freedom. At the time of writing, the IMLS method was the best-ranked, fully published method on the KITTI odometry dataset[2].

The approach of Nüchter et al. [18] slices the raw sweep into multiple segments that are assumed rigid. Their algorithm relies on a strong planarity constraint of the ground. They extended their algorithm in [17] to bootstrap it with data from the Google Cartographer library, no longer requiring geometric constraints. The authors mention processing times in the order of days, making the algorithm offline.

There are also commercially available solutions for backpack mapping such as the Leica Pegasus system which is based on two Veloyne LIDARs, cameras, GPS and IMU sensors. However, little is known about the used algorithms. An attempt to quantify the quality of a similar backpack mapping system was conducted by Rönnholm et al. [19] who compared the results with data acquired by a UAV. Kukko et al. [13] used surveyed fiducials for comparison. Revealing absolute accuracies of their backpack mapping system of about two centimeters. However, the analyzed mapping systems make strong use of GPS, making their results not directly comparable to our solution, which does not rely on GPS and instead fuses IMU measurements directly with the laser data.

[2] At the time of writing, the best method using only laser data was the LOAM method by Zhang and Singh. However, the results reported on the KITTI benchmark webpage are no longer equivalent to those reported in their paper [22], indicating that their solution has been updated. The updated algorithm is no longer publicly available.

3 Laser Odometry

Since our MBL sensors are being moved while the devices are measuring, assembling the data of a sweep into a correct 3D scan requires motion compensation. This refers the process of transforming all points within a sweep into a common coordinate frame. The k-th sweep is expected to start at the coordinate frame F_{t_k} with timestamp t_k.

A scan is an aggregate of all measurements acquired during one sweep and is assumed to be a set $\mathcal{S}_k = \{\langle \mathbf{p}_i, t_i \rangle \,|\, i \in \{1, \ldots, n\}\}$ of tuples, each consisting of a point $\mathbf{p}_i$ and a corresponding timestamp t_i. Points are represented in the sensor's local coordinate system at the time they were measured, i. e. F_{t_i}.

3.1 Selection of Query Points

Due to the measuring principle of laser scanners there are significantly more points in the immediate vicinity of the laser than at greater distances. As Deschaud [6] observed in his work, points in close distance are not very effective at constraining the rotation of a scan registration problem. This is because even small rotational errors lead to large, measurable deviations at large distances, while at small distances the deviations are much smaller and therefore disappear in measurement noise. Deschaud proposes a clever sampling strategy that however requires normals which are expensive to compute.

We propose a much simpler approach, where we extract a set of *query points* $\mathcal{Q}_k \subseteq \mathcal{S}_k$ from the point cloud. Points are selected from $\mathcal{S}_k$ so that a minimum distance of $\delta_\mathcal{Q} \approx 20\,\text{cm}$ between the points is maintained within $\mathcal{Q}_k$. Technically, this is achieved by storing points in a uniform grid and performing radius queries against it for every considered point. The minimum distance between the points mitigates the non-uniform point density of laser scans and results in a Poisson disk sampling of the scan. The obtained set of points is further subsampled by randomly selecting $N_\mathcal{Q}$ points.

3.2 Motion Compensation

For motion compensation, we consider the query points from two consecutive sweeps of the Velodyne sensor at once: $\mathcal{Q}_{k,k+1} = \mathcal{Q}_k \bigcup \mathcal{Q}_{k+1}$. This is one of the key differences to prior work, such as [22], where only the current sweep is considered. The use of two sweeps better constrains the optimization problem for the first sweep, because optimization has to ensure, the trajectory is appropriate beyond the first sweep. In our experiments, we observed significantly reduced noise on the estimation results using this method.

The goal of our motion compensation is to approximate a function $\zeta_k : [t_k, t_{k+2}] \to \mathbb{R}^3 \times S^3$ that for a given timestamp returns the relative pose (translation and unit quaternion) from F_{t_i} with $t_k \leq t_i \leq t_{k+2}$ to the coordinate frame F_{t_k} at the start of the first sweep. The trajectory models the motion between the start of the first and the end of the second sweep and can be used to motion-compensate the two sweeps. See Fig. 2 for an overview of the coordinate systems.

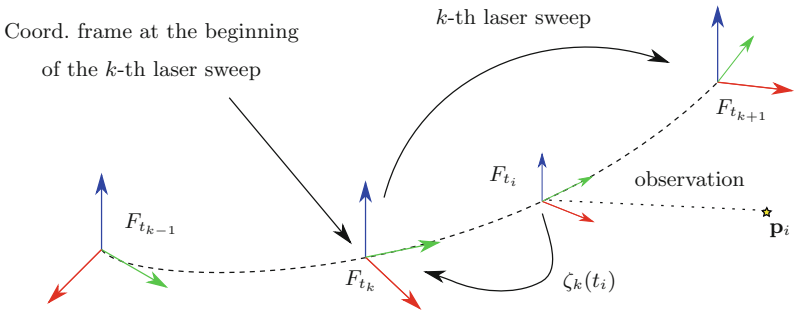

Fig. 2. Overview of the coordinate frames. F_{t_k} refers to the coordinate frame of the laser sensor at the start of the k-th sweep. A point $\mathbf{p}_i$ that is observed from the sensor at time t_i is represented in the coordinate system F_{t_i}. The function $\zeta_k(t)$ introduced in Sect. 3.2 models the relative pose of F_t w.r.t. F_{t_k}.

Trajectory Model. We assume having an initial guess $\bar{\zeta}_k$ for the trajectory. Being a constant during subsequent optimization, it can be arbitrarily complex. In practice, we obtain this trajectory by integrating IMU measurements for the duration of $[t_k, t_{k+2}]$ using the same method shown in Sect. 4. Required biases are taken from the most recent available result from graph optimization. If no IMU is used, $\bar{\zeta}_k$ can be chosen such that it predicts the motion of the LIDAR for the considered duration by linear extrapolation of the previously computed sensor motion. We choose this approach when IMU data is unavailable or when we compare against a non-inertial variant of our algorithm.

For optimization, we model the actual deviation from the predicted trajectory $\bar{\zeta}_k$ with a bias $\mathbf{b} \in \mathbb{R}^6$. We use an exponential map representation for the bias, allowing it to be 'added' on top of the pose from the prediction using a specialized $\boxplus$-operator (see [9]). Hence, our function is linear in the log map of the trajectory. Our complete model for the trajectory can be written as follows:

$$\hat{\zeta}_k(t, \mathbf{b}) = \bar{\zeta}_k(t) \boxplus \left(\mathbf{b} \frac{t - t_k}{t_{k+2} - t_k} \right) \tag{1}$$

Using this model, we are able to transform a point measured at time t to the start of the current (k-th) sweep. We now proceed to define a set of error functions that can be used to determine the unknown bias.

Residuals. We assume already having a local map (see next section) containing reference points $\mathcal{R}_{k-1}$ that are represented in frame $F_{t_{k-1}}$. We transform it to F_{t_k} and obtain $\hat{\mathcal{R}}_k$. In analogy to the point-to-plane ICP algorithm [2,15], every query point $\langle \mathbf{p}_i, t_i \rangle \in \mathcal{Q}_{k,k+1}$ is also transformed into the coordinate frame F_{t_k} using $\hat{\zeta}_k$ in order to determine a set of neighbors in $\hat{\mathcal{R}}_k$ within some radius ϵ. If not enough points are found within the given radius, no residual is generated for the given query point. Otherwise, we extract the normal $\mathbf{n}$ of the reference

'surface' using eigen analysis[3]. The support vector $\mathbf{q}$ of the plane is the mean of the neighbor points. Therefore, our residual functions are defined as

$$r_i(\mathbf{b}) = \mathbf{n}^{\mathrm{T}}(\hat{\zeta_k}(t_i, \mathbf{b}) * \mathbf{p}_i - \mathbf{q}). \tag{2}$$

We use the $*$ operator to indicate rigid transformation of a point with a pose.

Local Reference Map. Since data from one sweep of a MBL is typically very sparse, determining neighbors for a query point $\mathbf{p}_i$ only from *the* most recent compensated point cloud is not sufficient and leads to universally bad performance on all datasets we tested on (see Sect. 5). Instead, we choose a denser approximation of the local vicinity current position: All compensated points within the last $N_{\mathrm{LM}} \approx 100$ sweeps (excluding the k-th sweep) are stored in a local map $\mathcal{R}_k$ at coordinate frame F_{t_k}. By only storing new points whose distance to points already stored exceeds a threshold $\delta_{\mathrm{LM}} \approx 5\,\mathrm{cm}$, we effectively approximate a Poisson disk sampling of the scans. As a result, the covariance estimation in a radius ϵ around a query point $\mathbf{p}_i$ is more robust and not as prone to inhomogeneous point densities of the individual sweeps. Note that we insert *all* points from the laser scan in this set, not only the selected query points.

To accelerate neighbor search, the local map is stored in a uniform grid data structure. All tuples $\langle \mathbf{p}_i, t_i \rangle$ of the laser points are stored in ascending order of timestamps, making it efficient to remove points from old laser scans that are no longer needed in the local map.

Optimization. The goal of motion compensation is to solve the problem

$$\arg\min_{\mathbf{b}} \sum_i \rho(||r_i(\mathbf{b})||^2_{\Sigma_i}), \tag{3}$$

where $||.||^2_{\Sigma_i}$ corresponds to the squared Mahalanobis distance, Σ_i is the variance of the i-th measurement, and where $\rho(.)$ is the Tukey loss function that reduces the influence of outliers. Instead of selecting hard-coded uncertainties Σ_i for the individual residuals, we robustly estimate them from the data. To this end, we compute the median absolute deviation (MAD) of the residuals, which is known to be able to be robust for up to 50% of outliers. It can be used as a consistent estimator of the standard deviation by multiplying it with a scaling factor $c = 1.4826$, which is correct for normally distributed values. Thus, the standard deviation can be estimated using

$$\hat{\sigma}(\{r_i\}) = c \cdot \mathrm{median}(\{|r_i - \mathrm{median}(\{r_i\})|\}). \tag{4}$$

Therefore, we set $\Sigma_i = \hat{\sigma}(\{r_i\})^2$ and solve the problem using the Ceres solver [1]. Having estimated $\mathbf{b}$, we motion-compensate $\mathcal{S}_k$ (not $\mathcal{S}_{k+1}$) and insert its points into the local map $\mathcal{R}_k$. Note, how our motion estimation algorithm uses the sweeps in an overlapping fashion: Two consecutive sweeps are used to motion-compensate only one sweep.

[3] This is relatively efficient, because it is done for the set of query points and not for the whole point cloud.

4 IMU Integration

For mapping and IMU integration, we build an online factor graph for a global optimization problem. Nodes correspond to random variables that represent system's state at the beginning of a sweep. We define random variables

$$\mathbf{x}_t = (\mathbf{p}, \mathbf{q}) \in \mathbb{R}^3 \times S^3 \tag{5}$$

for the pose of the system at time step t, comprising both the position $\mathbf{p} \in \mathbb{R}^3$ and a normalized orientation quaternion $\mathbf{q} \in S^3$. The random variables

$$\mathbf{u}_t = (\mathbf{v}, \mathbf{b}^{\mathrm{a}}, \mathbf{b}^{\mathrm{g}}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \tag{6}$$

are used for the dynamic state of the system and contain linear velocity $\mathbf{v}$ (represented in inertial space) and biases $\mathbf{b}^a$ and $\mathbf{b}^g$ for the accelerometer and gyro. The amount of dynamic state random variables added to the system is configurable. We currently create one for every five system state nodes.

4.1 Cost Functions

We use three types of factors that are defined by their residual function within a least-squares problem.

Pose Prior Factor. We put a prior

$$\mathbf{r}_{\mathrm{Prior}} : \mathbb{R}^3 \times S^3 \to \mathbb{R}^6 \tag{7}$$

$$\mathbf{r}_{\mathrm{Prior}}(\mathbf{x}_t) = \mathbf{x}_t \boxminus (0, 0, 0, 1, 0, 0, 0)^T \tag{8}$$

on the first node so that the optimization problem is constrained. The given seven-dimensional vector represents the world origin with three translational components, which are zero, and four rotational components, which represent a unit quaternion in the order w, x, y, z. The used $\boxminus$ operator computes a 6D pose error on the SE3 manifold (see [9]). The covariance associated with the residual function is chosen such that the rotational part is only constrained along the yaw axis, ensuring that the initialization is still appropriate in cases where the device does not start up in alignment with gravity. The positional part of the first pose is held constant at the world origin during optimization. Hence, the associated covariances are not relevant and we simply set them to 1.

Laser Odometry Factor. The $\hat{\zeta}_k$ function from Sect. 3 allows motion-compensation of the k-th laser sweep. Assuming this sweep begins at time t_1 and ends at t_2, we can obtain an estimate $\hat{\mathbf{z}}_{t_1,t_2}$ for the relative pose between the end- and the beginning of this sweep as $\hat{\mathbf{z}}_{t_1,t_2} = \hat{\zeta}_k(t_2)$. We use this pose in a factor

$$\mathbf{r}_{\mathrm{Odo}} : (\mathbb{R}^3 \times S^3) \times (\mathbb{R}^3 \times S^3) \to \mathbb{R}^6 \tag{9}$$

$$\mathbf{r}_{\mathrm{Odo}}(\mathbf{x}_{t_1}, \mathbf{x}_{t_2}) = (\mathbf{x}_{t_1} \ominus \mathbf{x}_{t_2}) \boxminus \hat{\mathbf{z}}_{t_1,t_2} \tag{10}$$

to softly constrain the relative pose between two consecutive sweep poses in the factor graph. The $\ominus$ operator computes the current relative pose between the two pose nodes. A covariance matrix needed to weigh the residual is obtained as a part of the optimization during motion compensation.

IMU-based Relative Pose Factor. We use a pre-integrated factor

$$\mathbf{r}_{\text{IMU}} : (\mathbb{R}^3 \times S^3) \times (\mathbb{R}^3 \times S^3) \times \mathbb{R}^9 \times \mathbb{R}^9 \to \mathbb{R}^{15} \tag{11}$$

$$\mathbf{r}_{\text{IMU}}(\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \mathbf{u}_{t_1}, \mathbf{u}_{t_2}) = \begin{pmatrix} \hat{\mathbf{p}}_n - \mathbf{p}_{t_2} \\ \log(\hat{\mathbf{q}}_n^{-1}\mathbf{q}_{t_2}) \\ \hat{\mathbf{v}}_n - \mathbf{v}_{t_2} \\ \hat{\mathbf{b}}_n^{\text{g}} - \mathbf{b}_{t_2}^{\text{g}} \\ \hat{\mathbf{b}}_n^{\text{a}} - \mathbf{b}_{t_2}^{\text{a}} \end{pmatrix} \tag{12}$$

that computes the residual[4] between the 'new' state—as predicted by integrating n IMU measurements on top of the 'old' state—and the actual 'new' state. To obtain predictions for the 'new' state, we follow established methods from the vision area [10,14]. Given n IMU measurements between t_1 and t_2, we denote accelerometer and gyro measurements a_j and ω_j and time τ_j, $j = 1, \ldots, n$.

We iteratively integrate over these measurements using simple motion equations to obtain an updated state estimate as well as an associated covariance matrix. This iteration is essentially multiple repetitions of an Extended Kalman Filter prediction. For notational brevity, we define

$$\bar{\omega}_j = \tfrac{1}{2}(\omega_j + \omega_{j+1}) - \mathbf{b}_{t_1}^{\text{g}} \quad \bar{\mathbf{a}}'_j = \tfrac{1}{2}(\mathbf{a}_j + \mathbf{a}_{j+1}) - \mathbf{b}_{t_1}^{\text{a}} - \hat{\mathbf{q}}_j^{-1} * \mathbf{g} \quad \Delta_{\tau_j} = \tau_{j+1} - \tau_j \tag{13}$$

where $\mathbf{g}$ is the gravity vector. In both $\bar{\omega}_j$ as well as $\bar{\mathbf{a}}'_j$, the respective biases are subtracted from the average of successive measurements. In $\bar{\mathbf{a}}'_j$, we also remove the gravity vector from the measurements after transforming into inertial space.

Starting with $\hat{\mathbf{p}}_0 = \mathbf{p}_{t_1}, \hat{\mathbf{q}}_0 = \mathbf{q}_{t_1}, \hat{\mathbf{v}}_0 = \mathbf{v}_{t_1}, \hat{\mathbf{b}}_0^{\text{g}} = \mathbf{b}_{t_1}^{\text{g}}, \hat{\mathbf{b}}_0^{\text{a}} = \mathbf{b}_{t_1}^{\text{a}}$ and an initial covariance matrix $\mathbf{\Sigma}_0 = \mathbf{0}_{15\times 15}$, we apply the following iteration for $j = 0, \ldots, n-1$:

$$\hat{\mathbf{p}}_{j+1} = \hat{\mathbf{p}}_j + \Delta_{\tau_j}(\hat{\mathbf{q}}_j * (\hat{\mathbf{v}}_j + \frac{\Delta_{\tau_j}}{2}\bar{\mathbf{a}}'_j)) \qquad \hat{\mathbf{b}}_{j+1}^{g} = \hat{\mathbf{b}}_j^{g} \tag{14}$$

$$\hat{\mathbf{q}}_{j+1} = \hat{\mathbf{q}}_j \cdot \exp(\Delta_{\tau_j} \cdot \bar{\omega}_j) \qquad \hat{\mathbf{b}}_{j+1}^{a} = \hat{\mathbf{b}}_j^{a} \tag{15}$$

$$\hat{\mathbf{v}}_{j+1} = \hat{\mathbf{v}}_j + \Delta_{\tau_j}(\bar{\mathbf{a}}'_j - [\bar{\omega}_j]^{\times}\hat{\mathbf{v}}_j) \qquad \mathbf{\Sigma}_{j+1} = \mathbf{J}\mathbf{\Sigma}_j\mathbf{J}^T + \mathbf{G}\Delta_{\tau_j}^2\mathbf{N}\mathbf{G}^T \tag{16}$$

Here, $\mathbf{J} \in \mathbb{R}^{15\times 15}$ is the Jacobian of the full state propagation function. $\mathbf{G} \in \mathbb{R}^{15\times 12}$ is the derivative of the same propagation function by the inertial noise variables which are assumed to additively corrupt the inertial measurements. We omit the actual values here for brevity. $\mathbf{N}$ is a block-diagonal matrix with

$$\mathbf{I}_{3\times 3}\sigma_{\text{g}}^2, \mathbf{I}_{3\times 3}\sigma_{\text{bg}}^2, \mathbf{I}_{3\times 3}\sigma_{\text{a}}^2, \mathbf{I}_{3\times 3}\sigma_{\text{ba}}^2 \tag{17}$$

[4] Note that we use a log map of S^3 to $\mathbb{R}^3$ for measuring the 'difference' between the orientations [11].

on the diagonal corresponding to the noise strength of gyroscope, gyroscope bias, accelerometer and accelerometer bias. They can be determined for a given IMU using the Allan variance method [3]. The variable $\mathbf{\Sigma}$ corresponds to the uncertainty of the integration that is used to weigh the residual during optimization.

IMU factors are re-integrated every time the base state $\mathbf{x}_{t_1}$ changes. This could potentially be an issue for very long trajectories that we plan to solve using [7] combined with an incremental least-squares solver [12] in future work.

4.2 Optimization

Optimization of the covariance-weighted squared sum of all residual functions is carried out using the Ceres solver [1]. All occurring quaternions are optimized using a local parametrization included in Ceres that allows for an efficient and mathematically sound optimization on the underlying S^3 manifold.

5 Experiments

KITTI Dataset. The KITTI odometry benchmark [8] is widely accepted for evaluating large-scale, outdoor odometry algorithms. The associated dataset contains 22 sequences featuring laser scans from a Velodyne HDL 64 MBL. Ground truth derived from a high-quality commercial GPS/INS is provided for half of the sequences, serving as a training set, while the rest forms the test set. Scans in the KITTI odometry dataset are already motion-compensated using an unknown trajectory which—as we suppose—is the ground truth trajectory of the vehicle. Scan points are represented at the time where the laser faced exactly forward which complicates processing for our algorithm that requires raw, *uncompensated* Velodyne point clouds. Therefore, for each sweep, we take the relative pose from the previous sweep and use this to undo this motion compensation of the current one, making the data raw as possible. This allows us to run the benchmark without major changes in our method but introduces potential errors into the results.

As Deschaud [6] and others have observed, the point clouds in the dataset suffer from bad intrinsic calibration. Deschaud came up with a simple approximate calibration for the data which we also use for our experiments (see [6]).

The IMU part of our algorithm is not applicable to the KITTI dataset since IMU data is not provided as a part of the odometry benchmark.

Our results on the training set are given in Table 1. It can be seen that we outperform IMLS on 6 from 10 datasets, yet both methods appear to be relatively similar in performance. We believe that the map representation and feature selection advocated by IMLS may result in similar quantitative benefits over LOAM as our proposed method. The main advantage of our algorithm is the runtime, which makes it suitable for real-time operation. It is possible that the two algorithms have other strengths and weaknesses that are not visible in this purely quantitative analysis on a car-mounted dataset, however, no direct in-depth comparison was made in the context of this work.

Table 1. Comparison of the translation drift per meter (in %) of LOAM [24], IMLS [6] and our algorithm on the KITTI training dataset.

Seq.	Environment	LOAM	IMLS	Ours	Seq.	Environment	LOAM	IMLS	Ours
00	Urban	0.78	**0.50**	0.51	06	Urban	0.65	0.33	**0.31**
01	Highway	1.43	0.82	**0.79**	07	Urban	0.63	**0.33**	0.34
02	Urban+Country	0.92	**0.53**	0.54	08	Urban+Country	1.12	**0.80**	0.84
03	Country	0.86	0.68	**0.65**	09	Urban+Country	0.77	0.55	**0.46**
04	Country	0.71	**0.33**	0.44	10	Urban+Country	0.79	0.53	**0.52**
05	Urban	0.57	0.32	**0.27**					

Results on the KITTI test set are provided on the KITTI website[5], where our approach is called MC2SLAM. We hold the shared 4th/5th place of all laser-based algorithms with a translational error of 0.69%. We do however feature a better rotational error and lower run time than the equally ranked IMLS method (see 'Runtime' paragraph below).

Own Datasets. Our own datasets[6] were all recorded on a Velodyne HDL 32 sensor which handily features a built-in IMU. We associate individual timestamps with each point based on the specifications in the manufacturer's datasheet. This is in contrast to the KITTI dataset, where we can only roughly estimate a point's timestamp using its azimuth angle, lowering accuracy of our computations.

For the datasets CAMPUS_DRIVE and FIELD we mounted the sensor on a car, comparable to the KITTI dataset. For CAMPUS_RUN the sensor was mounted on a helmet so that a person could walk around to record data (see Fig. 1). Due to erratic motion from head movements, we expect the IMU to have much more impact in comparison to the very smooth motion patterns of a vehicle.

For large datasets, it is very difficult to determine an accurate, 6-DoF ground truth. Even on the KITTI dataset, the accuracy of the ground truth—especially at small scales—is limited, as has been noticed by other authors [5]. To some extent, specialized evaluation metrics accommodate for these issues and allow to obtain reasonable quantitative results. However, since the wheel-odometry, GPS/INS based solution can not be applied to our head-mounted datasets containing both indoor and outdoor data, we introduce the concept of *checkpoints*. A checkpoint is a point at which a loop in the data could be closed, making it possible to determine the relative pose between the two scans involved. No actual loop closing has to be performed, only the relative pose error at this trajectory position is computed. It represents the amount of drift that has accumulated over the course of the loop. We compute the translational- and rotational drift per meter, comparable to the KITTI evaluation by averaging over the errors

[5] http://www.cvlibs.net/datasets/kitti/eval_odometry.php see entry 'MC2SLAM'.
[6] See URL: https://agas.uni-koblenz.de/data/datasets/mc2slam/.

Table 2. Comparison of the translation drift per meter (in %) using different settings of our algorithm.

Local Map	✓	✗	✓
IMU	✓	✓	✗
CAMPUS_RUN_1	**0.41**	1.90	8.96
CAMPUS_RUN_2	**0.54**	2.13	15.65
CAMPUS_DRIVE	**0.10**	0.64	0.30
FIELD	**0.42**	7.50	0.43

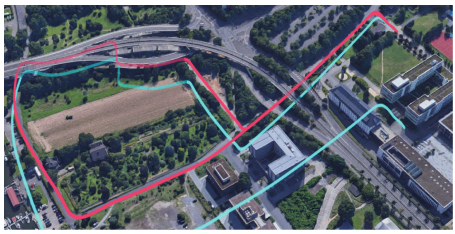

Fig. 3. Trajectory of the FIELD dataset drawn on top of a screenshot from Google Maps with and without using a local map for reference point matching in magenta resp. cyan. (Color figure online)

divided by the loop lengths. We specifically choose this metric to make our results comparable to the KITTI results.

For our datasets, checkpoints have been obtained manually using a specialized tool to perform and verify the registration. All our datasets contain at least one loop between the beginning and the end. Some contain several loops, improving the reliability of the chosen evaluation approach.

Our algorithm randomly selects feature points during motion compensation. For this reason, we ran our algorithm 10 times[7] with different random seeds, in order to obtain a precise measure of accuracy. Our averaged results are shown in Table 2. The results emphasize the importance of a local map, and highlight the versatility of our method: The datasets contain head-mounted in- and outdoor scenarios as well as car-mounted ones. Yet we are seeing low very low drift in all of them. As seen in the plot of the trajectory in Fig. 3, the local map is particularly important to keep the drift low in the long term.

Integration of an IMU is particularly important for head-mounted datasets. For example, we observe that the drift on the CAMPUS_RUN dataset is unacceptable without the integration of an IMU. Motion compensation for those kinds of datasets appears to heavily rely on inertial sensing. For the datasets CAMPUS_DRIVE as well as FIELD, we are seeing smaller benefits from the IMU, which we attribute to the much smoother motion patterns within a car-mounted setting that make linear motion predictions during motion-compensation relatively accurate. However, there are also car-mounted applications in off-road environments, which we did not include in our evaluation. These applications would probably benefit from the IMU integration in a similar way than the head-mounted datasets we evaluated.

For more information about the datasets evaluated in Table 2, please refer to the supplemental material and the given web page.

[7] This number was manually determined to be sufficient to reduce the deviation of the results to negligible values.

Runtime. For our own recorded datasets, the SLAM problem is generally solved in real time on an Intel i7-3700K processor. Precise runtimes were determined by running our algorithm and averaging over 1000 frames of the CAMPUS_RUN_1 dataset: The actual motion compensation algorithm takes about 32 ms per sweep in total. The selection of reference points takes 7.7 ms, estimation of the bias variable $\mathbf{b}$ for the motion compensation trajectory takes 17.6 ms and applying the resulting trajectory to the full sweep $\mathcal{S}_k$ takes 3.7 ms. Insertion of the compensated sweep into the local map is done in a background thread and does not block the main thread. Pose graph optimization is also executed in a separate thread and only needs to be performed every five sweeps (see Sect. 4.2). For this dataset, this step took around 108.6 ms per run.

The runtime of bias estimation depends on the number of query points $N_{\mathcal{Q}}$ matched with the local map. In this example 500 reference points were used for the optimization. While using 1500 points slightly increases accuracy, runtime is more than doubled, increasing the duration of this step from 17.6 ms to 37.8 ms.

6 Conclusion

We have presented an algorithm for laser-based odometry that is applicable to a broad range of domains. The two-scan motion compensation against a local Poisson map yields fast and reliable pose-estimates with low long-term drift, which we were able to show using the KITTI benchmark, as well as on our own datasets. Tight integration of an IMU allows for erratic motion, allowing the use of hand-held or even head-mounted sensors for mapping hard-to-reach areas. In contrast to many existing backpack mapping solutions, our approach does not require GPS, making it suitable for a wide variety of use cases.

Acknowledgement. The authors would like to thank three anonymous reviewers for their helpful comments.

References

1. Agarwal, S., Mierle, K., et al.: Ceres solver. http://ceres-solver.org
2. Besl, P.J., McKay, N.D.: Method for registration of 3-D shapes. In: Sensor Fusion IV: Control Paradigms and Data Structures, vol. 1611, pp. 586–607. International Society for Optics and Photonics (1992)
3. Board, I.: IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros. IEEE Std., pp. 952–1997 (1998)
4. Bosse, M., Zlot, R., Flick, P.: Zebedee: design of a spring-mounted 3-D range sensor with application to mobile mapping. IEEE Trans. Robot. **28**(5), 1104–1119 (2012)
5. Cvišić, I., Petrović, I.: Stereo odometry based on careful feature selection and tracking. In: 2015 European Conference on Mobile Robots, ECMR, pp. 1–6. IEEE (2015)
6. Deschaud, J.: IMLS-SLAM: scan-to-model matching based on 3D data. CoRR abs/1802.08633 (2018). http://arxiv.org/abs/1802.08633

7. Forster, C., Carlone, L., Dellaert, F., Scaramuzza, D.: IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In: Proceedings of Robotics: Science and Systems, Rome, Italy, July 2015. https://doi.org/10.15607/RSS.2015.XI.006
8. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition, CVPR (2012)
9. Hertzberg, C., Wagner, R., Frese, U., Schröder, L.: Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. Inf. Fusion **14**(1), 57–77 (2013)
10. Hesch, J.A., Kottas, D.G., Bowman, S.L., Roumeliotis, S.I.: Consistency analysis and improvement of vision-aided inertial navigation. IEEE Trans. Robot. **30**(1), 158–176 (2014)
11. Kaess, M.: Simultaneous localization and mapping with infinite planes. In: ICRA, vol. 1, p. 2 (2015)
12. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J.J., Dellaert, F.: iSAM2: Incremental smoothing and mapping using the Bayes tree. Int. J. Robot. Res. **31**(2), 216–235 (2012)
13. Kukko, A., Kaartinen, H., Hyyppä, J., Chen, Y.: Multiplatform mobile laser scanning: usability and performance. Sensors **12**(9), 11712–11733 (2012)
14. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual–inertial odometry using nonlinear optimization. Int. J. Robot. Res. **34**(3), 314–334 (2015)
15. Low, K.L.: Linear least-squares optimization for point-to-plane ICP surface registration, no. 4. Chapel Hill, University of North Carolina (2004)
16. Moosmann, F., Stiller, C.: Velodyne SLAM. In: Proceedings of the IEEE Intelligent Vehicles Symposium, pp. 393–398, Baden-Baden, Germany, June 2011
17. Nüchter, A., Bleier, M., Schauer, J., Janotta, P.: Improving Google's cartographer 3D mapping by continuous-time SLAM. Int. Arch. Photogram. Remote Sens. Spat. Inf. Sci. **42**, 543 (2017)
18. Nüchter, A., Borrmann, D., Koch, P., Kühn, M., May, S.: A man-portable, IMU-free mobile mapping system. ISPRS Ann. Photogram. Remote Sens. Spat. Inf. Sci. **2**, 17–23 (2015). https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-3-W5/17/2015/
19. Rönnholm, P., Liang, X., Kukko, A., Jaakkola, A., Hyyppä, J.: Quality analysis and correction of mobile backpack laser scanning data. ISPRS Ann. Photogram. Remote Sens. Spat. Inf. Sci. **3**, 41 (2016)
20. Zhang, J., Kaess, M., Singh, S.: Real-time depth enhanced monocular odometry. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2014, pp. 4973–4980. IEEE (2014)
21. Zhang, J., Kaess, M., Singh, S.: A real-time method for depth enhanced visual odometry. Auton. Robots **41**(1), 31–43 (2017)
22. Zhang, J., Singh, S.: LOAM: lidar odometry and mapping in real-time. In: Robotics: Science and Systems, vol. 2 (2014)
23. Zhang, J., Singh, S.: Visual-lidar odometry and mapping: low-drift, robust, and fast. In: 2015 IEEE International Conference on Robotics and Automation, ICRA, pp. 2174–2181. IEEE (2015)
24. Zhang, J., Singh, S.: Low-drift and real-time lidar odometry and mapping. Auton. Robots **41**(2), 401–416 (2017)