Laser-visual-inertial Odometry and Mapping with High Robustness and Low Drift

Article *in* Journal of Field Robotics · August 2018

DOI: 10.1002/rob.21809

2 authors:

Ji Zhang
Carnegie Mellon University
30 PUBLICATIONS 927 CITATIONS

SEE PROFILE

Sanjiv Singh
Carnegie Mellon University
225 PUBLICATIONS 7,791 CITATIONS

SEE PROFILE

Received: 10 November 2017 | Revised: 23 March 2018 | Accepted: 9 July 2018

DOI: 10.1002/rob.21809

WILEY

REGULAR ARTICLE

Laser–visual–inertial odometry and mapping with high robustness and low drift

Ji Zhang | Sanjiv Singh

Kaarta, Inc., Pittsburgh, Pennsylvania

Correspondence
Ji Zhang, Kaarta, Inc., Pittsburgh, PA 15213.
Email: ji@kaarta.com

Abstract

We present a data processing pipeline to online estimate ego-motion and build a map of the traversed environment, leveraging data from a 3D laser scanner, a camera, and an inertial measurement unit (IMU). Different from traditional methods that use a Kalman filter or factor-graph optimization, the proposed method employs a sequential, multilayer processing pipeline, solving for motion from coarse to fine. Starting with IMU mechanization for motion prediction, a visual–inertial coupled method estimates motion; then, a scan matching method further refines the motion estimates and registers maps. The resulting system enables high-frequency, low-latency ego-motion estimation, along with dense, accurate 3D map registration. Further, the method is capable of handling sensor degradation by automatic reconfiguration bypassing failure modules. Therefore, it can operate in the presence of highly dynamic motion as well as in the dark, texture-less, and structure-less environments. During experiments, the method demonstrates 0.22% of relative position drift over 9.3 km of navigation and robustness w.r.t. running, jumping, and even highway speed driving (up to 33 m/s).

KEYWORDS
mapping, position estimation

1 | INTRODUCTION

This paper aims at developing a method for online ego-motion estimation with data from a 3D laser scanner, a camera, and an inertial measurement unit (IMU). The estimated motion further registers laser points to build a map of the traversed environment. In many real-world applications, ego-motion estimation and mapping must be conducted in real time. In an autonomous navigation system, the map is crucial for motion planning and obstacle avoidance, while the motion estimation is important for vehicle control and maneuver. Very often, high-accuracy global positioning system (GPS)–inertial navigation system (INS) solutions are impractical when the application is GPS-denied, lightweight, or cost-sensitive. The proposed method utilizes only perception sensors without reliance on GPS.

Specially, we are interested in solving for extremely aggressive motion. Yet, it remains nonobvious how to solve the problem reliably in 6 degrees of freedom (DOF), in real time, and in a small form factor. The problem is closely relevant to sensor degradation due to sparsity of the data during dynamic maneuver. To the best of our knowledge, the proposed method is by far the first to enable such high-rate ego-motion estimation capable of handling running and jumping (see Figure 1 as an example), while at the same time develop a dense, accurate 3D map, in the field under various lighting and structural conditions, and using only sensing and computing devices that can be easily carried by a person.

The key reason that enables this level of performance is our novel way of data processing. As shown in Figure 2a, a standard Kalman filter-based method typically uses IMU mechanization in a prediction step followed by update steps seeded with individual visual features or laser landmarks. On the other hand, a factor-graph optimization-based method combines constraints from all sensors in one optimization problem (see Figure 2b). In comparison, our modularized pipeline sequentially recovers motion in a coarse-to-fine manner (see Figure 2c). Starting with motion prediction from an IMU, a

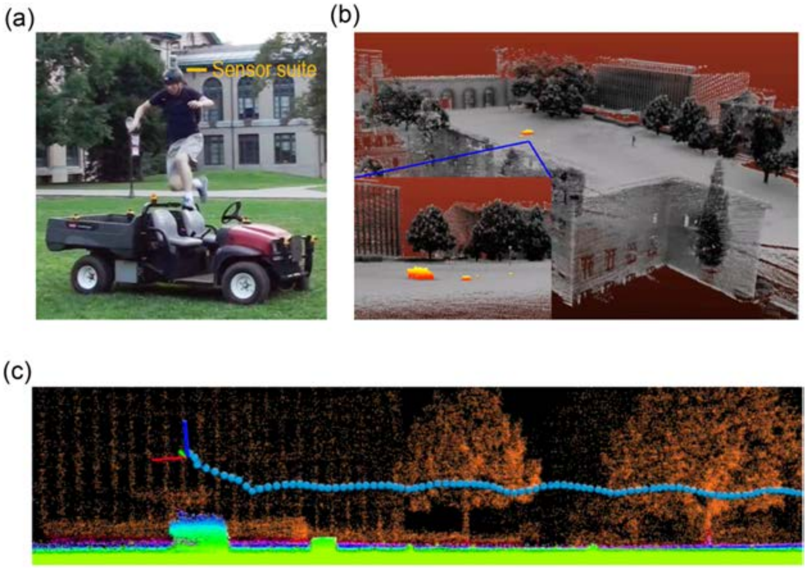


FIGURE 1 Representative results of our odometry and mapping data processing pipeline. (a) An operator carries the sensor suite on a helmet and a processing computer in a backpack, running, and jumping over a vehicle. The method estimates ego-motion of the sensor suite in real time, as well as develops a precise 3D map using only onboard processing. (b) It presents the map built with the vehicle labeled in orange. (c) It shows the estimated trajectory. The coordinate frame in (c) illustrates the sensor pose at the point in (a) [Color figure can be viewed at wileyonlinelibrary.com]

visual–inertial coupled method estimates motion and registers laser points locally. Then, a scan matching method further refines the estimated motion. The last module also registers laser points to build a map.

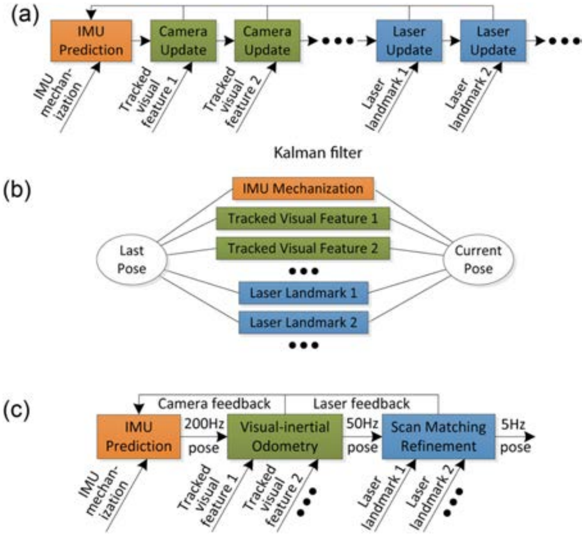


FIGURE 2 Diagram of the odometry and mapping data processing pipeline. (a) It shows a standard Kalman filter setup. IMU mechanization is used for prediction, then each visual feature and laser landmark seeds an individual update step. (b) It shows a factor-graph optimization setup. All constraints from the IMU, visual features, and laser landmarks are combined in an optimization problem. (c) It presents the proposed sequential data processing pipeline. Starting with IMU mechanization for prediction, a visual–inertial coupled method estimates ego-motion; then, a scan matching method further refines the estimated motion. From left to right, motion is recovered from coarse to fine and accuracy is improved step by step to a high level. Further, both modules on the right provide feedback to correct velocity drift and biases of the IMU. IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

The method design follows a key insight: Drift in ego-motion estimation has a lower frequency than a module's own frequency. The three modules are therefore arranged in decreasing order of frequency. High-frequency modules are specialized to handle aggressive motion, while low-frequency modules cancel drift from the previous modules. The sequential processing also favors computation: Modules in the front take less computation and execute at high frequencies, giving sufficient time to modules in the back for thorough processing. The method is therefore able to achieve a high level of accuracy while running online in real time.

Further, the data processing pipeline is carefully designed to handle sensor degradation. If the camera is nonfunctional, for example, due to darkness, dramatic lighting changes, or texture-less environments, or if the laser scanner is nonfunctional, for example, due to structure-less environments, the corresponding module is bypassed and the rest of the pipeline is staggered to function reliably. The method is tested through a large number of experiments and results show that it can produce high accuracy over several kilometers of navigation and robustness w.r.t. environmental degradation and aggressive motion. The main contributions of the paper are as follows:

- We propose a modularized data processing pipeline to leverage range, vision, and inertial sensing for motion estimation and mapping through mulilayer optimization. Therefore, it achieves high accuracy and low drift.
- The proposed pipeline is dynamically reconfigurable. It fully or partially bypasses failure modules and combines the rest of the pipeline to handle sensor degradation. Therefore, it can handle environmental degradation and aggressive motion.
- The proposed pipeline employs a two-level voxel representation and a multithread processing implementation to accelerate scan matching.

- The proposed pipeline is extendable to localization on an existing map and therefore can enable collaborative mapping, for example, between ground and air (more details in Section 10.3).
- The proposed pipeline is thoroughly tested with a large number of datasets in various difficult environments, with moving object, aggressive motion, and in large scale.

The rest of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we make assumptions, define coordinate systems, and state the problem. In Section 4, we introduce the IMU prediction module. In Sections 5 and 6, we summarize the visual–inertial odometry module and the scan matching module, respectively. Section 7 discusses the transform integration and Section 8 the robustness in degraded environmental conditions. The extension to localization on an existing map is presented in Section 8. Experiment results are shown in Section 10 and conclusion is made in Section 11.

2 | RELATED WORK

This paper is most related to vision- and laser-based state estimation. For vision-based methods, stereo camera systems (Corke, Strelow, & Singh, 2004; Konolige, Agrawal, & Sol, 2011; Maimone, Cheng, & Matthies, 2007; Nister, Naroditsky, & Bergen, 2006) are commonly used. These systems benefit from the baseline between the two cameras as a reference to determine scale of the motion estimation. However, if a monocular camera is used (Engel, Koltun, & Cremers, 2016; Forster, Pizzoli, & Scaramuzza, 2014; Klein & Murray, 2007; Newcombe, Lovegrove, & Davison, 2011), without aiding from additional sensors or assumptions about motion, scale is generally unsolvable. In recent years, RGB-D cameras have gained popularity in the research community. These cameras provide depth information associated with individual pixels and hence can help determine scale easily. With RGB-D cameras, methods (Engelhard, Endres, Hess, Sturm, & Burgard, 2011; Henry, Krainin, Herbst, Ren, & Fox, 2012; Huang et al., 2011; Kerl, Sturm, & Cremers, 2013; Whelan, Johannsson, Kaess, Leonard, & McDonald, 2013) have shown promising results. However, these methods only utilize the image areas with coverage of depth information, possibly causing large image areas being wasted if in an open environment where depth can only be sparsely available. The visual–inertial odometry involved in the pipeline is closely relevant to RGB-D methods since the method is assisted by laser ranging and associates the range information to visual features. In comparison, our method involves both features with and without depth to maximize features' usage. Further, it also retrieves depth by triangulation using previously estimated motion for features without range coverage.

Another category is to couple cameras with an IMU, where scale constraints are provided from IMU accelerations. To this end, Huang, Kaess, and Leonard (2014) and Li and Mourikis (2013) tightly couple a monocular camera and an IMU in a Kalman filter. Weiss et al. (2013) loosely couple an IMU with an independent monocular visual odometry method (Klein & Murray, 2007). Other methods (Forster, Carlone, Dellaert, & Scaramuzza, 2015; Leutenegger, Lynen, Bosse, Siegwart, & Furgale, 2015) use optimization to solve for the motion. We prefer optimization-based methods over filter-based methods. The visual–inertial odometry in our pipeline involves constraints from visual features and an IMU.

When using laser scanners for motion estimation, the downside is from the fact that laser points are received continuously at different time stamps. When the scanning rate is slow, a scan cannot be consider as a rigid body but distortion is present due to external motion of the laser scanner. To date, it has been shown that motion can be recovered with a laser scanner itself (Ceriani, Sanchez, Taddei, Wolfart, & Sequeira, 2015; Velas, Spanel, & Herout, 2016; Wei, Wu, & Fu, 2015). This requires a motion model being involved. Tong, Anderson, Dong, and Barfoot (2014) model the motion as constant velocity or with Gaussian processes. The method matches visual features from images generated by laser intensity returns. Combining a 2D laser scanner and an IMU, Bosse and Zlot's method (Bosse, Zlot, & Flick, 2012; Zlot et al., 2014) uses IMU mechanization as the motion model. The method matches spatiotemporal patches formed by laser points to estimate sensor motion and correct IMU biases in offline batch optimization.

The same problem of motion distortion is observed with rolling-shutter cameras, that is, image pixels are perceived continuously over time, resulting in image distortion caused by extrinsic motion of the camera. A few visual odometry methods use an IMU to compensate for the rolling-shutter effect given readout time of the pixels (Guo et al., 2014; Li & Mourikis, 2014). Further, Furgale, Barfoot, and Sibley (2012) show that the continuous motion of the camera can be modeled with a set of temporal basis functions.

An alternative way is to use other sensors to assist laser scanners. Scherer et al. (2012) navigation system uses stereo visual odometry (Geiger, Ziegler, & Stiller, 2011) loosely coupled with an IMU to estimate motion of a microhelicopter. The estimated motion registers laser points on a map. Also taking visual odometry output as motion approximation, Droeschel, Stuckler, and Behnke (2014) method and Holz and Behnke (2014) method further match laser scans to refine the motion. The proposed pipeline is inspired by the same concept, but is more complete with range, vision, and inertial sensors all coupled. The difference is that methods (Droeschel et al., 2014; Holz & Behnke, 2014) consider the camera and the laser scanner as independent modules, while in our pipeline the modules are highly interactive and dynamically reconfigurable. This allows both the camera and the laser scanner to correct velocity drift and biases of the IMU. In the case that the camera or the laser scanner degrades, the other one substitutes and couples with the IMU. More importantly, it enables different combinations in the pipeline adapted to specific environments and motion, which ensures the robustness w.r.t. environmental degradation and aggressive motion that typically cause different modules to fail.

The proposed method is based on our pervious work where a visual odometry method (Zhang, Kaess, & Singh, 2014) and a laser odometry method (Zhang & Singh, 2014) are proposed separately. In Zhang and Singh (2017a), the work is extended to perform localization based on existing maps. The visual odometry method is now key-framed and coupled with an IMU. The laser odometry method is further improved by a two-level voxel representation and multithread processing for better real-time performance. The fully integrated pipeline reaches the level of accuracy that is unachieved in our previous work (Zhang et al., 2014; Zhang & Singh, 2014, 2015). Further, the method now handles sensor failures. By combining functioning modules, it can reliably operate in the presence of aggressive motion as well as in low-light, texture-less, and structure-less environments. The paper is an extended version of our conference papers (Zhang & Singh, 2017a, 2017b) with more technical details and experiment results.

3 | ASSUMPTIONS, COORDINATES, AND PROBLEM

3.1 | Assumptions and coordinate systems

Considering a sensor system including a laser scanner, a camera, and an IMU, we assume that the camera is modeled by a pinhole camera model (Hartley & Zisserman, 2000) and the intrinsic parameters are known (Zhang, 2000). The extrinsic parameters among the three sensors are calibrated. The relative pose between the camera and the laser scanner is obtained based on (Unnikrishnan & Hebert, 2005), and the relative pose between the laser scanner and the IMU is calibrated in the same way as Geiger, Lenz, Stiller, and Urtasun (2013) by solving a hand–eye problem (Horaud & Dornaika, 1995). As extrinsic calibration is made, we use a single coordinate system for the camera and the laser scanner. This is chosen to be the camera coordinate system—all laser points are projected into the camera coordinate system in preprocessing. For simplicity of notations, we also assume that the IMU coordinate system is parallel to the camera coordinate system—IMU measurements are rotationally corrected upon receiving. We define coordinate systems in the following (see Figure 3 for illustration),

- Camera coordinate system $\{C\}$ is originated at the camera optical center. The x-axis points to the left, the y-axis points upward, and the z-axis points forward coinciding with the camera principal axis.
- IMU coordinate system $\{I\}$ is originated at the IMU measurement center. The x-, y-, and z-axes are parallel to $\{C\}$ pointing to the same directions.
- World coordinate system $\{W\}$ is the gravity aligned coordinate system coinciding with $\{C\}$ at the start.

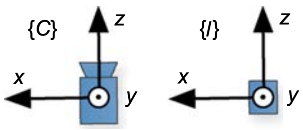


FIGURE 3 Illustration of coordinate systems in top-down view. $\{C\}$ is the camera coordinate system. All laser points are converted into $\{C\}$ in preprocessing. $\{I\}$ is the inertial measurement unit coordinate system [Color figure can be viewed at wileyonlinelibrary.com]

3.2 | Maximum a posterior (MAP) estimation problem

A state estimation problem can be formulated as a MAP estimation problem. We follow the definition in Dellaert and Kaess (2006). Define $\mathscr{X} = \{\mathbf{x}_i\}$, $i \in \{1, 2, \ldots, m\}$, as the set of system states, $\mathscr{U} = \{\mathbf{u}_i\}$, $i \in \{1, 2, \ldots, m\}$, as the set of control inputs, and $\mathscr{Z} = \{\mathbf{z}_k\}$, $k \in \{1, 2, \ldots, n\}$, as the set of landmark measurements. Given the proposed pipeline, $\mathscr{Z}$ is composed of both visual features and laser landmarks. The joint probability of the system is defined as follows,

$$P(\mathscr{X}|\mathscr{U}, \mathscr{Z}) \propto P(\mathbf{x}_0) \prod_{i=1}^{m} P(\mathbf{x}_i|\mathbf{x}_{i-1}, u_i) \prod_{k=1}^{n} P\left(z_k|\mathbf{x}_{i_k}\right), \tag{1}$$

where $P(\mathbf{x}_0)$ is a prior of the initial system state, $P(\mathbf{x}_i|\mathbf{x}_{i-1}, u_i)$ represents the motion model, and $P(z_k|\mathbf{x}_{i_k})$ represents the landmark measurement model. For each problem formulated as (1), there is a corresponding Bayesian belief network representation of the problem (Kaess, Ranganathan, & Dellaert, 2008). The MAP estimation is to maximize (1). Under the assumption of zero-mean Gaussian noise, the problem is equivalent to a least-squares problem,

$$\mathscr{X}^* = \arg\min_{\mathscr{X}} \sum_{i=1}^{m} ||r_{\mathbf{x}_i}||^2 + \sum_{k=1}^{n} \left|\left|r_{\mathbf{z}_k}\right|\right|^2. \tag{2}$$

Here, $r_{\mathbf{x}_i}$ and $r_{\mathbf{z}_k}$ are residual errors associated with the motion model and the landmark measurement model, respectively.

The standard way of solving (2) is to combine all sensor data, i.e., visual features, laser landmarks, and IMU measurements, into a large factor-graph optimization problem. The proposed data processing pipeline, instead, formulates multiple small optimization problems and solves the problems in a coarse-to-fine manner. We give the problem statement as,

> **Problem 1** *Given data from a laser scanner, a camera, and an IMU, formulate and solve problems as (2) to determine poses of $\{C\}$ w.r.t $\{W\}$, then use the estimated poses to register laser points and build a map of the traversed environment in $\{W\}$.*

4 | IMU PREDICTION SUBSYSTEM

4.1 | IMU mechanization

This subsection describes the IMU prediction subsystem. As our method considers $\{C\}$ as the fundamental sensor coordinate system, we also characterize the IMU w.r.t. $\{C\}$. Recalling in Section 3.1, we have stated that $\{I\}$ and $\{C\}$ are parallel coordinate systems. Let $\boldsymbol{\omega}(t)$ and $\boldsymbol{a}(t)$ be two 3×1 vectors indicating the angular rates and accelerations of $\{C\}$ at time t. Let $\boldsymbol{b}_\omega(t)$ and $\boldsymbol{b}_{\boldsymbol{a}}(t)$ be the corresponding biases, and $\boldsymbol{n}_\omega(t)$ and $\boldsymbol{n}_{\boldsymbol{a}}(t)$ be the corresponding noises. All above terms are defined in $\{C\}$. Additionally, let $\boldsymbol{g}$ be the constant gravity vector in $\{W\}$. The IMU measurement terms are,

$$\hat{\omega}(t) = \boldsymbol{\omega}(t) + \boldsymbol{b}_{\omega}(t) + \boldsymbol{n}_{\omega}(t), \tag{3}$$

$$\hat{\boldsymbol{a}}(t) = \boldsymbol{a}(t) - {}^{C}_{W}\mathbf{R}(t)\boldsymbol{g} - {}^{I}_{C}\mathbf{t}||\boldsymbol{\omega}(t)||^2 + \boldsymbol{b}_a(t) + \boldsymbol{n}_a(t), \tag{4}$$

where ${}^{C}_{W}\mathbf{R}(t)$ is the rotation matrix from $\{W\}$ to $\{C\}$, and ${}^{I}_{C}\boldsymbol{t}$ is the translation vector between $\{C\}$ and $\{I\}$.

Note that the term ${}^{I}_{C}\boldsymbol{t}||\boldsymbol{\omega}(t)||^2$ represents the centrifugal force due to the fact that the rotation center (origin of $\{C\}$) is different from the origin of $\{I\}$. State-of-the-art visual–inertial navigation methods (Huang et al., 2014; Li & Mourikis, 2013) tend to model the motion in $\{I\}$ to eliminate this term. Since our method uses visual features both with and without depth information (discussed in Section 5.3), converting features without depth from $\{C\}$ to $\{I\}$ is not straight forward. We model the motion in $\{C\}$ instead. Practically, the camera and the IMU are mounted close to each other to maximally reduce effect of the term.

The IMU biases are slowly changing variables. We take the most recently updated biases for motion integration. First, (3) is integrated over time. Then, the resulting orientation is used with (4) for integration over time twice to obtain translation.

4.2 | Bias correction

The IMU bias correction can be made by feedback from either the camera or the laser scanner. Each one contains the estimated incremental motion over a short amount of time. When calculating the biases, we model the biases to be constant during the incremental motion. Still starting with (3), by comparing the estimated orientation with IMU integration, we can calculate $\boldsymbol{b}_{\omega}(t)$. The updated $\boldsymbol{b}_{\omega}(t)$ is used in one more round of integration to recompute the translation, which is compared with the estimated translation to calculate $\boldsymbol{b}_{\boldsymbol{a}}(t)$.

To reduce the effect of high-frequency noises, a sliding window is used keeping a certain number of biases. The averaged biases from the sliding window are used. In this implementation, the length of the sliding window functions as a parameter determining update rate of the biases. We are aware that a rigorous way is to model the biases as random walks and update the biases through optimization (Forster et al., 2015; Leutenegger et al., 2015). However, we prefer this nonstandard implementation to keep IMU processing in a separate module. The implementation favors dynamic reconfiguration of the pipeline, that is, the IMU can be coupled with either the camera or the laser scanner. If the camera is nonfunctional, the IMU biases are corrected by the laser scanner instead (more discussion in Section 8).

5 | VISUAL–INERTIAL ODOMETRY SUBSYSTEM

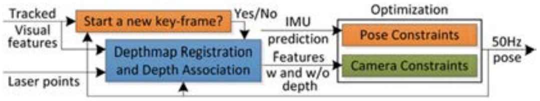


FIGURE 4 Diagram of the visual–inertial odometry subsystem. IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

This section summarizes the visual–inertial odometry subsystem. This is based on a method proposed in our previous work (Zhang et al., 2014). A system diagram is shown in Figure 4. The method couples vision with an IMU. Both provide constraints to an optimization problem that estimates incremental motion. At the same time, the method associates depth information to visual features. If a feature is located in an area where laser range measurements are available, depth is obtained from laser points. Otherwise, depth is calculated from triangulation using the previously estimated motion sequence. As the last option, the method can also use features without any depth by formulating constraints in a different way. This is true for those features which do not have laser range coverage or cannot be triangulated due to the fact that they are not tracked long enough or located in the direction of camera motion.

5.1 | Camera constraints

The visual–inertial odometry is a key-frame-based method. A new key-frame is determined if more than a certain number of features lose tracking or the image overlap is below a certain ratio. Here, let us use right superscript $l \in Z^+$ to indicate the last key-frame, and $c, c \in Z^+$ and $c > l$, to indicate the current frame. As we have discussed, the method combines features with and without depth. For a feature that is associated with depth at key-frame l, we denote it as $\boldsymbol{X}_l = [x_l, y_l, z_l]^T$ in $\{C_l\}$. Correspondingly, a feature without depth is denoted as $\overline{\boldsymbol{X}}_l = [\overline{x}_l, \overline{y}_l, 1]^T$ using normalized coordinates instead. Note that $\boldsymbol{X}_l$, $\overline{\boldsymbol{X}}_l$, x_l, and $\overline{x}_l$ are different from $\mathscr{X}$ and $\boldsymbol{x}$ in (1) which represent the system state. We only associate depth to features at key-frames, for two reasons: (a) Depth association takes some processing and executing at key-frames only helps reduce computation intensity; and (b) laser points need to be registered on a depthmap (more discussion in Section 5.3); however, the transform to frame c has not been established at this point and the depthmap is not available at frame c. We denote a normalized feature in $\{C_c\}$ as $\overline{\boldsymbol{X}}_c = [\overline{x}_c, \overline{y}_c, 1]^T$.

Let $\mathbf{R}_l^c$ and $\boldsymbol{t}_l^c$ be the 3×3 rotation matrix and 3×1 translation vector between frames l and c, where $\mathbf{R}_l^c \in \mathbf{SO}(3)$ and $\boldsymbol{t}_l^c \in \mathbb{R}^3$, $\mathbf{R}_l^c$ and $\boldsymbol{T}_l^c$ form an $\mathbf{SE}(3)$ transformation (Murray & Sastry, 1994). The motion function between frames l and c is written as,

$$\boldsymbol{X}_c = \mathbf{R}_l^c \boldsymbol{X}_l + \boldsymbol{t}_l^c. \tag{5}$$

Recalling that $\boldsymbol{X}_c$ has unknown depth, let d_c be the depth, where . Substituting $\boldsymbol{X}_c$ with $d_c\overline{\boldsymbol{X}}_c$ and combining the first and second rows with the third row in (5) to eliminate d_c, we have,

$$(\boldsymbol{R}(1) - \overline{x}_c\boldsymbol{R}(3))\boldsymbol{X}_l + t(1) - \overline{x}_c t(3) = 0, \tag{6}$$

$$(\boldsymbol{R}(2) - \overline{y}_c\boldsymbol{R}(3))\boldsymbol{X}_l + t(2) - \overline{y}_c t(3) = 0. \tag{7}$$

Here, $\boldsymbol{R}(h)$ and $t(h)$, $h \in \{1, 2, 3\}$, are the hth rows of $\mathbf{R}_l^c$ and $\boldsymbol{t}_l^c$. In the case that depth in unavailable to a feature, let d_l be the unknown depth at key-frame l. Substituting $\boldsymbol{X}_l$ and $\boldsymbol{X}_c$ with $d_l\overline{\boldsymbol{X}_l}$ and $d_c\overline{\boldsymbol{X}}_c$, respectively, and combining all three rows in (5) to eliminate d_l and d_c, we obtain another constraint,

$$[\bar{y}_c t(3) - t(2), -\bar{x}_c t(3) + t(1), \bar{x}_c t(2) - \bar{y}_c t(1)]\mathbf{R}_l^c\overline{\mathbf{X}_l} = 0. \tag{8}$$

5.2 | Motion estimation

The motion estimation is to solve an optimization problem combining three sets of constraints: (a) From features with known depth as (6) and (7); (b) from features with unknown depth as (8); and (c) from the IMU prediction. Let us define $\mathbf{T}_a^b$ as the 4×4 transformation matrix between frames a and b,

$$\mathbf{T}_a^b = \begin{bmatrix} \mathbf{R}_a^b & \boldsymbol{t}_a^b \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \tag{9}$$

where $\mathbf{R}_a^b$ and $\boldsymbol{t}_a^b$ are the corresponding rotation matrix and translation vector. Further, let θ_a^b be a 3×1 vector corresponding to $\mathbf{R}_a^b$ through an exponential map (Murray & Sastry, 1994), where $\theta_a^b \in so(3)$. The normalized term $\theta/||\theta||$ represents direction of the rotation, and $||\theta||$ is the rotation angle. Each $\mathbf{T}_a^b$ corresponds to a set of θ_a^b and $\boldsymbol{t}_a^b$ containing 6-DOF motion of the camera.

To formulate the IMU pose constraints, we take the solved motion transform between frames l and $c-1$, namely $\mathbf{T}_l^{c-1}$. From IMU mechanization, we obtain a predicted transform between the last two frames $c-1$ and c, denoted as $\hat{\mathbf{T}}_{c-1}^c$. The predicted transform at frame c is calculated as,

$$\hat{\mathbf{T}}_l^c = \hat{\mathbf{T}}_{c-1}^c \mathbf{T}_l^{c-1}. \tag{10}$$

Let $\hat{\theta}_l^c$ and $\hat{\boldsymbol{t}}_l^c$ be the 6-DOF motion corresponding to $\hat{\mathbf{T}}_l^c$. It is worth to mention that the IMU-predicted translation, $\hat{\boldsymbol{t}}_l^c$, is dependent on the orientation, that is, the orientation determines projection of the gravity vector through rotation matrix ${}_W^C\mathbf{R}(t)$ in (4), and hence the accelerations being integrated. We formulate $\hat{\boldsymbol{t}}_l^c$ as a function of θ_l^c and rewrite it as $\hat{\boldsymbol{t}}_l^c(\theta_l^c)$ from now on. At this point, we should clarify that the 200 Hz pose and so as the 50 Hz pose in Figure 2c are in indeed pose functions. When calculating $\hat{\boldsymbol{t}}_l^c(\theta_l^c)$, we start at frame c and integrate accelerations inversely w.r.t. time. Let θ_l^c be the rotation vector corresponding to $\mathbf{R}_l^c$ in (5), θ_l^c and $\boldsymbol{t}_l^c$ are the motion to be solved. The constraints are expressed as,

$$\Sigma_l^c \left[\left(\hat{\theta}_l^c - \theta_l^c\right)^T, \ \left(\hat{\boldsymbol{t}}_l^c(\theta_l^c) - \boldsymbol{t}_l^c\right)^T \right]^T = \mathbf{0}, \tag{11}$$

where Σ_l^c is a relative covariance matrix scaling the pose constraints appropriately w.r.t. the camera constraints.

In the visual–inertial odometry subsystem, the pose constraints fulfill the motion model and the camera constraints fullfill the landmark measurement model in (2). The optimization problem is solved by the Newton gradient-descent method (Nocedal & Wright, 2006) adapted to a robust fitting framework (Andersen, 2008) for outlier feature removal. In this problem, the state space contains θ_l^c and $\boldsymbol{t}_l^c$. In other words, we do not perform full-scale MAP estimation but only solve a marginalized problem. The landmark positions are not optimized. This means only six unknowns in the state space keeping computation intensity low. The argument is that the method involves laser range measurements to provide precise depth information to features, warranting motion estimation accuracy. Further optimizing the features' depth in bundle adjustment is practically unnecessary.

5.3 | Depth association

The method registers laser points on a depthmap using previously estimated motion. Laser points within the camera field of view are kept for a certain amount of time. The depthmap is downsampled to keep a constant density and stored in a two-dimensional K-D tree (de Berg, Cheong, van Kreveld, & Overmars, 2008) for fast index. In the K-D tree, all laser points are projected onto a unit sphere around the camera center (as in Figure 5). A point is represented by its two angular coordinates. When associating depth to features, we project the features onto the sphere. The three closest laser points are found on the sphere for each feature. Then, we further check their validity by calculating distances among the three points in Cartesian space. If a distance is larger than a threshold, the chance that the points are from different objects, for example, a wall and an object in front of the wall, is high and the validity check fails. Finally, the depth is interpolated from the three points assuming a local planar patch in Cartesian space.

For those features without laser range coverage, if they are tracked over a certain distance and not located in the direction of camera motion, we triangulate them using the image sequences where the features are tracked. This uses a similar procedure as Forster et al. (2014) and Vogiatzis and Hernandez (2011), where the depth is updated at each frame based on a Bayesian probabilistic mode. Figure 6 shows an example depthmap and 3D projected features. The green points have depth from the depthmap, the blue points are by triangulation, and the red points have unknown depth.

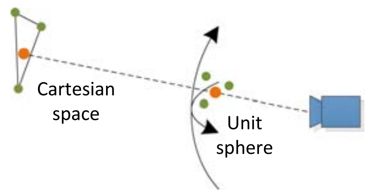


FIGURE 5 Illustration of feature depth association. Features (orange) and laser points on the demthmap (green) are projected from Cartesian space onto a unit sphere. Then, the three closest laser points on the sphere are found for each feature using a 2D K-D tree. The three points form a local planar patch in Cartesian space, and the depth is interpolated for the feature [Color figure can be viewed at wileyonlinelibrary.com]

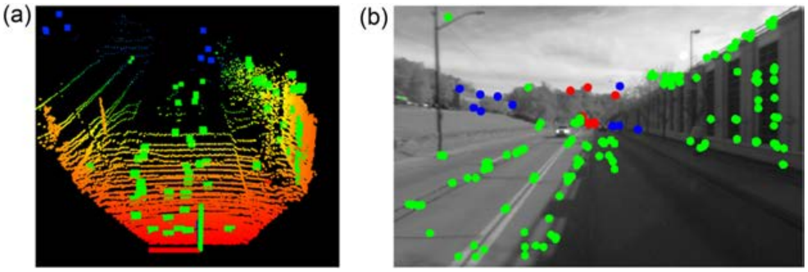


FIGURE 6 (a) Example depthmap (colored points) and 3D projected visual features. The green points are features whose depth is from the depthmap. The blue points are by triangulation. (b) Corresponding features in an image. The red points have unknown depth, hence are not drawn in (a) [Color figure can be viewed at wileyonlinelibrary.com]

6 | SCAN MATCHING SUBSYSTEM

This subsystem further refines motion estimates from the previous module by scan matching. The method is based on our previous work published in Zhang and Singh (2014). A diagram is present in Figure 7. The subsystem registers laser points in a local point cloud using provided odometry estimation. Then, geometric features are detected from the point cloud and matched to the map. The scan matching minimizes the feature-to-map distances, similar to many existing methods (François Pomerleau & Siegwart, 2015). However, the odometry estimation from the previous module also provides pose constraints in the optimization. The implementation uses voxel representation of the map. Further, it can dynamically configure to run on one to multiple central processing unit (CPU) threads in parallel.

6.1 | Laser constraints

When receiving laser scans, the method first registers points from a scan into a common coordinate system. Let us use $m \in Z^+$ to indicate the scan number. Recall that we use the camera coordinate system for both the camera and the laser scanner. For scan m, we associate it with the camera coordinate system at the beginning of the scan, denoted as $\{C_m\}$. To locally register the laser points, we take the odometry estimation from the visual–inertial odometry as key-poses, and use IMU measurements to interpolate in between the key-poses.

Let $\mathscr{P}_m$ be the locally registered point cloud from scan m. We extract two sets of geometric features from $\mathscr{P}_m$, one on sharp edges, namely edge points and denoted as $\mathscr{E}_m$, and the other on local planar surfaces, namely planar points and denoted as $\mathscr{H}_m$. When extracting geometric features, we use the term defined in Zhang and Singh (2014) to evaluate smoothness of local surfaces around the points. Denote a point as $\boldsymbol{X}^i_m = [x^i_m,\ y^i_m,\ z^i_m]^T$, $\boldsymbol{X}^i_m \in \mathscr{P}_m$. The set of surrounding points of $\boldsymbol{X}^i_m$ in $\mathscr{P}_m$ is denoted as $\mathscr{C}^i_m$. Given that orientations of the points around $\boldsymbol{X}^i_m$ from a laser scanner distribute evenly, the smoothness is defined as,

$$c^i_m = \frac{1}{\left|C^i_m\right| \cdot \left\| \boldsymbol{X}^i_m \right\|} \left\| \sum_{X^j_m \in C^i_m, j \neq i} \left(\boldsymbol{X}^i_m - \boldsymbol{X}^j_m \right) \right\|. \tag{12}$$

Edge points and planar points are extracted with large and small c^i_m values, respectively. We avoid selecting points whose neighbor points are already selected, points on boundaries of occluded regions (point B in Figure 8), or points whose local surfaces are close to be parallel to laser beams (point D in Figure 8). These points are likely to contain large noises or change positions over time as the sensor moves. Figure 9a gives an example of detected edge points (blue) and planar points (yellow).

The geometric features are then matched to the current map built. Let $\mathscr{Q}_{m-1}$ be the map point cloud after processing the last scan, $\mathscr{Q}_{m-1}$ is defined in $\{W\}$. The points in $\mathscr{Q}_{m-1}$ are separated into two sets containing edge points and planar points, respectively. We use voxels to store the map truncated at a certain distance around the sensor. For each voxel, we construct two 3D K-D trees (de Berg et al., 2008), one for edge points and the other for planar points. Using K-D trees for individual voxels accelerates point searching since given a query point, we only need to search in a specific K-D tree associated with a single voxel (more discussion in Section 6.3).

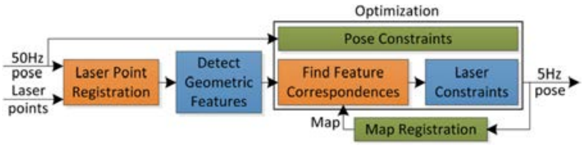


FIGURE 7 Diagram of the scan matching subsystem. MAP: maximum a posterior [Color figure can be viewed at wileyonlinelibrary.com]

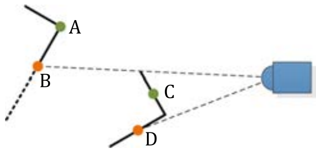


FIGURE 8 Illustration of geometric feature selection. Point A is a good edge point but point B is not. This is because point B is on the boundary of an occluded region, making it appear to be an edge point. Point C is a good planar point but point D is not, because the local surface of point D is close to be parallel to the laser beam. Points B and D are likely to contain large noises or change positions as the sensor moves, hence are not selected [Color figure can be viewed at wileyonlinelibrary.com]

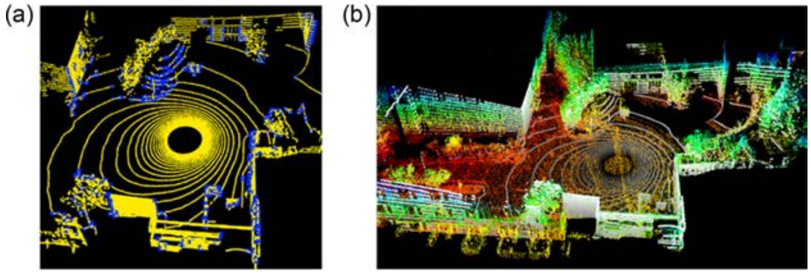


FIGURE 9 (a) Example edge points (blue) and planar points (yellow) detected from a scan. (b) Matching a scan (grey points) to the map (colored points), and then, the scan is merged with the map to extend the map further [Color figure can be viewed at wileyonlinelibrary.com]

When matching scans, we first project $\mathscr{E}_m$ and $\mathscr{H}_m$ into $\{W\}$ using the best guess of motion available, then for each point in $\mathscr{E}_m$ and $\mathscr{H}_m$, a cluster of closest points are found from the corresponding set on the map. To verify geometric distributions of the point clusters, we examine the associated eigenvalues and eigenvectors. Specifically, one large and two small eigenvalues indicate an edge line segment, and two large and one small eigenvalues indicate a local planar patch. If the matching is valid, an equation is formulated as for the distance from a point to its correspondence,

$$d^i_m = f\left(\mathbf{X}^i_m, \theta_m, \mathbf{t}_m\right), \tag{13}$$

where $\mathbf{X}^i_m \in \mathscr{E}_m$ or $\mathbf{X}^i_m \in \mathscr{H}_m$, $\theta_m \in \mathfrak{so}(3)$ and $\mathbf{t}_m \in \mathbb{R}^3$ indicate the 6-DOF pose of $\{C_m\}$ in $\{W\}$. Specifically, if $\mathbf{X}^i_m$ is an edge point, (13) describes the distance between $\mathbf{X}^i_m$ and the corresponding edge line segment,

$$d^i_m = \frac{\left|\left(\hat{\mathbf{X}}^i_m(\theta_m, \mathbf{t}_m) - \hat{\mathbf{X}}^j_{m-1}\right) \times \left(\hat{\mathbf{X}}^i_m(\theta_m, \mathbf{t}_m) - \hat{\mathbf{X}}^k_{m-1}\right)\right|}{\left|\hat{\mathbf{X}}^j_{m-1} - \hat{\mathbf{X}}^k_{m-1}\right|}, \tag{14}$$

where $\hat{\mathbf{X}}^i_m(\theta_m, \mathbf{t}_m)$ is the projected point of $\hat{\mathbf{X}}^i_m$ into $\{W\}$ using θ_m and $\mathbf{t}_m$, $\hat{\mathbf{X}}^j_{m-1}$ and $\hat{\mathbf{X}}^k_{m-1}$ are two points located on the edge line segment in $\{W\}$. If $\mathbf{X}^i_m$ is a planar point, (13) describes the distance between $\mathbf{X}^i_m$ and the corresponding local planar patch,

$$d^i_m = \frac{\left|\begin{matrix}\left(\hat{\mathbf{X}}^i_m(\theta_m, \mathbf{t}_m) - \hat{\mathbf{X}}^j_{m-1}\right) \\ \left(\left(\hat{\mathbf{X}}^j_{m-1} - \hat{\mathbf{X}}^k_{m-1}\right) \times \left(\hat{\mathbf{X}}^j_{m-1} - \hat{\mathbf{X}}^l_{m-1}\right)\right)\end{matrix}\right|}{\left|\left(\hat{\mathbf{X}}^j_{m-1} - \hat{\mathbf{X}}^k_{m-1}\right) \times \left(\hat{\mathbf{X}}^j_{m-1} - \hat{\mathbf{X}}^l_{m-1}\right)\right|}, \tag{15}$$

where $\hat{\mathbf{X}}^j_{m-1}$, $\hat{\mathbf{X}}^k_{m-1}$, $\hat{\mathbf{X}}^l_{m-1}$ are three points located on the local planar patch in $\{W\}$. Figure 9b shows an example where a scan is matched. The gray points are from a scan, and the colored points are from the map.

6.2 | Motion estimation

The scan matching is formulated into an optimization problem minimizing the overall distances described by (13). The optimization also involves pose constraints from prior motion. Let $\mathbf{T}_{m-1}$ be the 4×4 transformation matrix as for the pose of $\{C_{m-1}\}$ in $\{W\}$, $\mathbf{T}_{m-1}$ is generated by processing the last scan. Let $\hat{\mathbf{T}}^m_{m-1}$ be the pose transform from $\{C_{m-1}\}$ to $\{C_m\}$, as provided by the odometry estimation. Similar to (10), the predicted pose transform of $\{C_m\}$ in $\{W\}$ is,

$$\hat{\mathbf{T}}_m = \hat{\mathbf{T}}^m_{m-1}\mathbf{T}_{m-1}. \tag{16}$$

Let $\hat{\theta}_m$ and $\hat{\mathbf{t}}_m$ be the 6-DOF pose corresponding to $\hat{\mathbf{T}}_m$, and let Σ_m be a relative covariance matrix. The constraints are,

$$\Sigma_m[(\hat{\theta}_m - \theta_m)^T, \ (\hat{\mathbf{t}}_m - \mathbf{t}_m)^T]^T = 0. \tag{17}$$

Equation (17) refers to the case that the prior motion is from the visual–inertial odometry, assuming the camera is functional. Otherwise, the constraints are from the IMU prediction. Let us use $\hat{\theta}'_m$ and $\hat{\mathbf{t}}'_m(\theta_m)$ to denote the same terms by IMU mechanization. $\hat{\mathbf{t}}'_m(\theta_m)$ is a function of θ_m due to the fact that integration of accelerations is dependent on the orientation (same with $\hat{\mathbf{t}}^c_l(\theta^c_l)$ in (11)). The IMU pose constraints are,

$$\Sigma'_m[(\hat{\theta}'_m - \theta_m)^T, (\hat{\mathbf{t}}'_m(\theta_m) - \mathbf{t}_m)^T]^T = 0, \tag{18}$$

where Σ'_m is the corresponding relative covariance matrix. In the optimization problem, (17) and (18) are linearly combined into one set of constraints. The linear combination is determined by working

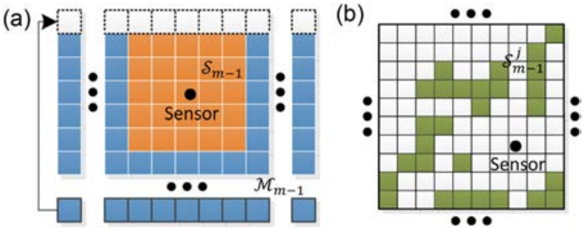


FIGURE 10 (a) Voxels on the map $\mathscr{M}_{m-1}$ (all voxels in (a)), and voxels surrounding the sensor $\mathscr{S}_{m-1}$ (orange voxels). $\mathscr{S}_{m-1}$ is a subset of $\mathscr{M}_{m-1}$. If the sensor approaches the boundary of the map, voxels on the opposite side of the boundary (bottom row) are moved over to extend the map boundary. Points in moved voxels are cleared and the map is truncated. (b) Each voxel $j \in \mathscr{S}_{m-1}$ (an orange voxel in (a)) is formed by a set of voxels $\mathscr{S}^j_{m-1}$ that are a magnitude smaller (all voxels in (b) $\in \mathscr{S}^j_{m-1}$). Before scan matching, we project points in $\mathscr{E}_m$ and $\mathscr{H}_m$ onto the map using the best guess of motion. Voxels in $\{\mathscr{S}^j_{m-1}\}$, $j \in \mathscr{S}_{m-1}$ occupied by points from $\mathscr{E}_m$ and $\mathscr{H}_m$ are labeled in green. Then, map points in green voxels are extracted as $\mathscr{Q}_{m-1}$. The figure is drawn in 2D but the algorithm is implemented in 3D [Color figure can be viewed at wileyonlinelibrary.com]

mode of the visual–inertial odometry (recapped in Section 8). The optimization problem refines $\boldsymbol{\theta}_m$ and $\boldsymbol{t}_m$, which is solved by the Newton gradient-descent method (Nocedal & Wright, 2006) adapted to a robust fitting framework (Andersen, 2008).

6.3 | Map in voxels

The points on the map are kept in voxels. We use a two-level voxel implementation as illustrated in Figure 10a. Let us use $\mathscr{M}_{m-1}$ to indicate the set of voxels on the map after processing the last scan. Voxels surrounding the sensor form a subset of $\mathscr{M}_{m-1}$, denoted as $\mathscr{S}_{m-1}$. Given a 6-DOF sensor pose, $\hat{\boldsymbol{\theta}}_m$ and $\hat{\boldsymbol{t}}_m$, there is a corresponding $\mathscr{S}_{m-1}$ which moves with the sensor on the map. When the sensor approaches the boundary of the map, as an example in Figure 10a, voxels on the opposite side of the boundary are moved over to extend the map boundary. Points in moved voxels are cleared resulting in truncation of the map.

As illustrated in Figure 10b, each voxel $j \in \mathscr{S}_{m-1}$ is formed by a set of voxels that are a magnitude smaller, denoted as $\mathscr{S}^j_{m-1}$. Voxels in $\mathscr{S}^j_{m-1}$ only last for one scan. Before matching scans, we project points in $\mathscr{E}_m$ and $\mathscr{H}_m$ onto the map using the best guess of motion and fill them into $\{\mathscr{S}^j_{m-1}\}$, $j \in \mathscr{S}_{m-1}$. Voxels occupied by points from $\mathscr{E}_m$ and $\mathscr{H}_m$ are labeled in green. Then, map points in green voxels are extracted to form $\mathscr{Q}_{m-1}$ and stored in 3D K-D trees for scan matching. An example of $\mathscr{Q}_{m-1}$ is the colored points in Figure 9b. Upon completion of scan matching, the scan is merged into the green voxels with the map. After that, the map points are downsized to maintain a constant density.

One question as for the algorithm implementation is why using two levels of voxels as described above. The explanation is that we use $\mathscr{M}_{m-1}$ to keep the map and $\{\mathscr{S}^j_{m-1}\}$, $j \in \mathscr{S}_{m-1}$, to retrieve the map around the sensor for scan matching. The map is truncated only when the sensor approaches the map boundary. In other words, if the sensor navigates inside the map, no truncation is needed. Voxels in $\mathscr{M}_{m-1}$ are stationary which store points on the map until the map is truncated in the associated voxels. Voxels in $\{\mathscr{S}^j_{m-1}\}$, $j \in \mathscr{S}_{m-1}$, are momentary and to retrieve the map for an instant scan. Another consideration is that using a single level of voxels in the size of $\{\mathscr{S}^j_{m-1}\}$, $j \in \mathscr{S}_{m-1}$, to keep the entire map is practically unfeasible due to computational limits. Table 1 compares CPU processing time using different voxel and K-D tree configurations. The time is averaged from multiple datasets collected from different types of environments covering confined and open, structured and vegetated areas. We see that using only one level of voxels, $\mathscr{M}_{m-1}$, results in about twice of processing time for K-D tree building and querying. This is because the second level of voxels, $\{\mathscr{S}^j_{m-1}\}$, $j \in \mathscr{S}_{m-1}$, help retrieve the map precisely. Without these voxel, more points are contained in $\mathscr{Q}_{m-1}$ and built into the K-D trees. Also, we see that using K-D trees for each voxel helps reduce processing time sightly in comparison to using K-D trees for all voxels in $\mathscr{M}_{m-1}$.

6.4 | Parallel processing

The scan matching is time-consuming and takes major computation in the pipeline. This involves building K-D trees, repetitively querying geometric feature points, and matrix manipulations for nonlinear optimization. Without taking advantage from a powerful graphics processing unit (GPU), the paper employs a CPU-based multithread implementation warranting the desired frequency. Figure 11a illustrates the case where two scans are matched in parallel. Upon receiving of a scan, a manager program arranges it to match with the latest map available. In a clustered environment with plenty of structures, matching is slow and may not complete before arrival of the next scan. Two matcher programs are called alternatively. On each matcher, $\mathscr{P}_m$, $\mathscr{P}_{m-1}$, ..., are matched with $\mathscr{Q}_{m-2}$, $\mathscr{Q}_{m-3}$, ..., respectively, giving twice amount of time for processing. On the other hand, in a clean environment with few structures, computation is light. Only the first matcher is called as in Figure 11b, and $\mathscr{P}_m$, $\mathscr{P}_{m-1}$, ..., are matched with $\mathscr{Q}_{m-1}$, $\mathscr{Q}_{m-2}$, ..., respectively. The implementation is configured to use maximally four threads; however it is uncommon that more than two threads are needed.

An alternative way is to process a single scan on multiple CPU threads in parallel. However, this will unavoidably cause efficiency loss. Lu and Hart's research (Lu & Hart, 2014) indicates that constructing K-D trees using four threads with low-dimensional data results in 87.5% of the original efficiency. Further, our multithread implementation is dynamically configured to arrange all processing on the minimum number of threads. This ensures low-latency ego-motion estimation as it leaves room on other threads for real-time processing of the IMU prediction and visual–inertial odometry.

7 | TRANSFORM INTEGRATION

The final motion estimation is integration of outputs from the three modules in Figure 2c. As illustrated in Figure 12, the 5-Hz scan matching output (blue section) fundamentally warrants accuracy. The 50-Hz visual–inertial odometry output (green section) and the 200-Hz IMU prediction (orange section) are integrated to the front for high-frequency motion estimates.

TABLE 1 Comparsion of average CPU processing time on K-D tree operation

	One-level voxels		Two-level voxels	
Task	K-D trees for all voxels	K-D trees for each voxel	K-D trees for all voxels	K-D trees for each voxel
Build (times per K-D tree; ms)	54	47	24	21
Query (times per point; ns)	4.2	4.1	2.4	2.3

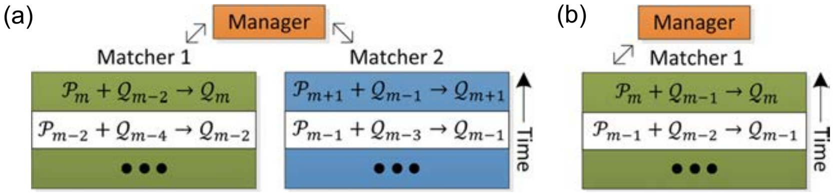

(a)
Manager
Matcher 1
Matcher 2
$\mathcal{P}_m + \mathcal{Q}_{m-2} \rightarrow \mathcal{Q}_m$
$\mathcal{P}_{m-2} + \mathcal{Q}_{m-4} \rightarrow \mathcal{Q}_{m-2}$
$\mathcal{P}_{m+1} + \mathcal{Q}_{m-1} \rightarrow \mathcal{Q}_{m+1}$
$\mathcal{P}_{m-1} + \mathcal{Q}_{m-3} \rightarrow \mathcal{Q}_{m-1}$
●●●
Time
(b)
Manager
Matcher 1
$\mathcal{P}_m + \mathcal{Q}_{m-1} \rightarrow \mathcal{Q}_m$
$\mathcal{P}_{m-1} + \mathcal{Q}_{m-2} \rightarrow \mathcal{Q}_{m-1}$
●●●
Time

FIGURE 11 Illustration of multithread scan matching. A manager program calls multiple matcher programs running on separate CPU threads and matches scans to the latest map available. (a) It shows a two-thread case. Scans $\mathscr{P}_m$, $\mathscr{P}_{m-1}$, ..., are matched with map $\mathscr{Q}_{m-2}$, $\mathscr{Q}_{m-3}$, ..., on each matcher, giving twice amount of time for processing. In comparison, (b) shows a one-thread case, where $\mathscr{P}_m$, $\mathscr{P}_{m-1}$, ..., are matched with $\mathscr{Q}_{m-1}$, $\mathscr{Q}_{m-2}$, ... The implementation is dynamically configurable using up to four threads [Color figure can be viewed at wileyonlinelibrary.com]

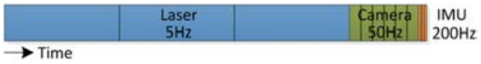


FIGURE 12 Illustration of transform integration. The final motion estimation is integration of the 5-Hz scan matching output (blue), the 50-Hz visual–inertial odometry output (green), and the 200-Hz IMU prediction (orange), at the IMU frequency. The density of the vertical line segments in each section indicates the motion estimation frequency from the corresponding module. IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

8 | ON ROBUSTNESS

The robustness of the pipeline is determined by its ability to handle sensor degradation. We assume the IMU is always reliable functioning as the backbone in the pipeline. Camera is sensitive to dramatic lighting changes. It also fails in a dark or texture-less environment or when significant motion blur is present causing visual features lose tracking. Laser scanner cannot handle structure-less environments, for example, a scene that is dominant by a plane. Further, the same degradation can be caused by sparsity of the data due to aggressive motion.

The method that we use to deal with these failures is originally proposed in Zhang, Kaess, and Singh (2016). Both the visual–inertial odometry and the scan matching modules formulate and solve optimization problems as (2). When a failure happens, it corresponds to a degraded optimization problem, that is, some directions of the state space are loosely constrained and noises are dominated in determining the solution in these directions. Let **J** be the Jacobian matrix associated with the current pose in (2), our method starts with computing eigenvalues, denoted as $\lambda_1, \lambda_2, ..., \lambda_6$, and eigenvectors, denoted as $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_6$, of $\mathbf{J}^T\mathbf{J}$. Here, six eigenvalues–eigenvectors are present because the state space contains 6-DOF motion of the sensor. Without loosing generality, $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_6$ are sorted in decreasing order. Each eigenvalue describes how well the solution is conditioned in the direction of its corresponding eigenvector. By comparing the eigenvalues to a threshold determined in a single degradation test as conducted in Zhang et al. (2016), we can separate well-conditioned directions from degraded directions in the state space. Let h, $h = 0, 1, ..., 6$, be the number of well-conditioned directions. Here. we define two matrices,

$$\mathbf{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_6]^T, \quad \overline{\mathbf{V}} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_h, 0, ..., 0]^T. \tag{19}$$

When solving an optimization problem, the nonlinear iteration starts with an initial guess. With the sequential pipeline in Figure 2c, the IMU prediction provides the initial guess for the visual–inertial odometry, whose output is taken as the initial guess for the scan matching. For the last two modules, let $\boldsymbol{x}$ be a solution and $\Delta\boldsymbol{x}$ be an update of $\boldsymbol{x}$ in a nonlinear iteration. Given the Newton gradient-descent method (Nocedal & Wright, 2006) used by the paper, $\Delta\boldsymbol{x}$ is calculated as,

$$\Delta\boldsymbol{x} = -(\mathrm{J}^T\mathrm{J})^{-1}\mathrm{J}^T\boldsymbol{b}. \tag{20}$$

Here, $\boldsymbol{b}$ is a matrix containing residuals of the linearized problem. During the optimization process, instead of updating $\boldsymbol{x}$ in all directions, we only update $\boldsymbol{x}$ in well-conditioned directions, keeping the initial guess in degraded directions instead,

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \mathrm{V}^{-1}\overline{\mathrm{V}}\Delta\boldsymbol{x}. \tag{21}$$

Let us further explain the intuition behind (21). The pipeline solves for motion in a coarse-to-fine order, starting with the IMU prediction, and the following two modules further solve or refine the motion as much as possible, fully (in 6-DOF) if the problem is well-conditioned, and partially (in zero to 5-DOF) otherwise. If the problem is completely degraded, $\overline{\mathbf{V}}$ becomes a zero matrix and the previous module's output is kept.

Recap 1 *Let us recap on the pose constraints described in (17) and (18). In fact, the two equations are linearly combined in the scan matching problem. Let us use* $\mathbf{V}_V$ *and* $\overline{\mathbf{V}}_V$ *to denote the matrices defined in (19) containing eigenvectors from the visual–inertial odometry module,* $\overline{\mathbf{V}}_V$ *represents well-conditioned directions in the subsystem, and* $\mathbf{V}_V - \overline{\mathbf{V}}_V$ *represents degraded directions. The combined constraints are as follows:*

$$\begin{gathered}\Sigma_m \mathbf{V}_V^{-1}\overline{\mathbf{V}}_V\left[\left(\hat{\theta}_m - \theta_m\right)^T, \left(\hat{\boldsymbol{t}}_m - \boldsymbol{t}_m\right)^T\right]^T + \Sigma'_m \mathbf{V}_V^{-1} \\ (\mathbf{V}_V - \overline{\mathbf{V}}_V)\left[(\hat{\theta}'_m - \theta_m)^T,\ \left(\hat{\boldsymbol{t}}'_m(\theta_m)\right) - \boldsymbol{t}_m)^T\right]^T = \mathbf{0}.\end{gathered} \tag{22}$$

In a normal case where the camera is functional, $\overline{\mathbf{V}}_V = \mathbf{V}_V$ *and (22) is composed of pose constraints from the visual–inertial odometry as (17). On the other hand, if the camera is completely degraded,* $\overline{\mathbf{V}}_V$ *is a zero matrix and (22) is composed of pose constraints from the IMU prediction as (18).*

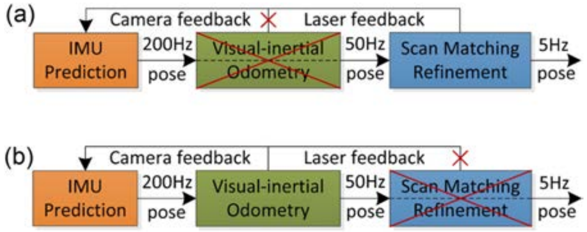


FIGURE 13 Case study of camera and laser degradation. (a) If visual features are insufficient for the visual–inertial odometry, the IMU prediction (partially) bypasses the green block to register laser points locally. Correction of velocity drift and biases of the IMU is made with the laser feedback. (b) If environmental structures are insufficient for the scan matching, the visual–inertial odometry output (partially) bypasses the blue block to register laser points on the map. Here, the dashed line segments indicate "bypass." IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

8.1 | Case study of camera degradation

As shown in Figure 13a, if visual features are insufficiently available for the visual–inertial odometry, the IMU prediction bypasses the green block fully or partially, depending on the number of well-conditioned directions in the visual–inertial odometry problem, and locally registers laser points for the scan matching. The bypassing IMU prediction is subject to drift. The laser feedback compensates for the camera feedback correcting velocity drift and biases of the IMU, only in directions where the camera feedback is unavailable. In other words, the camera feedback has a higher priority, due to the higher frequency making it more suitable. When sufficient visual features are found, the laser feedback is not used.

8.2 | Case study of laser degradation

As shown in Figure 13b, if environmental structures are insufficient for the scan matching to refine motion estimates, the visual–inertial odometry output fully or partially bypasses the blue block to register laser points on the map. If well-conditioned directions exist in the scan matching problem, the laser feedback contains refined motion estimates in those directions. Otherwise, the laser feedback becomes empty.

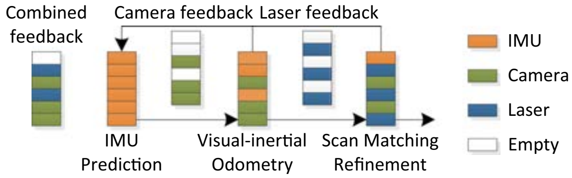


FIGURE 14 An example where both the camera and the laser scanner are degraded. A vertical bar represents a 6-DOF pose and each row is a DOF. Starting with the IMU prediction on the left where all six rows are orange, the visual–inertial odometry updates in 3-DOF where the rows become green, then the scan matching updates in another 3-DOF where the rows turn blue. The camera and the laser feedback is combined as the vertical bar on the left. The camera feedback has a higher priority: Blue rows from the laser feedback are only filled in if green rows from the camera feedback are not present. DOF: degree of freedom; IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

8.3 | Case study of camera and laser degradation

Finally, let us discuss a complex scenario where both the camera and the laser scanner are degraded. We use the example in Figure 14 to illustrate this scenario. A vertical bar with six rows represents a 6-DOF pose where each row is a DOF, corresponding to an eigenvector in (19). In this example, both the visual–inertial odometry and the scan matching update 3-DOF motion, leaving the motion unchanged in the other 3-DOF. Starting with the IMU prediction on the left where all six rows are orange, the visual–inertial odometry updates in 3-DOF where the rows change to green, then the scan matching updates in 3-DOF further where the rows turn blue. The camera and the laser feedback contains updates from each module on the green and the blue rows, respectively (white means empty). The feedback is combined upon receiving by the IMU prediction module as the vertical bar on the left. The camera feedback has a higher priority than the laser feedback (discussed in Section 8.1). During the combination, the blue rows are only filled in if the green rows are not present.

In reality, however, the visual–inertial odometry and the scan matching execute at different frequencies and have each own degraded directions. We take the poses from the scan matching output and use IMU messages to interpolate in between the poses. This way, we create an incremental motion that is time aligned with the visual–inertial odometry output. Let θ_{c-1}^{c} and $\boldsymbol{t}_{c-1}^{c}$ be the 6-DOF motion estimated by the visual–inertial odometry between frames $c-1$ and c, where $\theta_{c-1}^{c} \in \mathfrak{so}(3)$ and $\boldsymbol{t}_{c-1}^{c} \in \mathbb{R}^3$. Let θ'^{c}_{c-1} and $\boldsymbol{t}'^{c}_{c-1}$ be the corresponding terms estimated by the scan matching after time interpolation. Consider $\mathbf{V}_V$ and $\overline{\mathbf{V}}_V$ to be the matrices defined in (19) containing eigenvectors from the visual–inertial odometry module, $\overline{\mathbf{V}}_V$ represents well-conditioned directions, and $\mathbf{V}_V - \overline{\mathbf{V}}_V$ represents degraded directions. Let $\mathbf{V}_S$ and $\overline{\mathbf{V}}_S$ be the same matrices from the scan matching module. The following equation calculates the combined feedback, $\boldsymbol{f}_C$,

$$\boldsymbol{f}_C = \boldsymbol{f}_V + \mathbf{V}_V^{-1}(\mathbf{V}_V - \overline{\mathbf{V}}_V)\boldsymbol{f}_S, \tag{23}$$

where $\boldsymbol{f}_V$ and $\boldsymbol{f}_S$ represent the camera and the laser feedback,

$$\boldsymbol{f}_V = \mathbf{V}_V^{-1}\overline{\mathbf{V}}_V\left[\left(\theta_{c-1}^{c}\right)^T,\ \left(\boldsymbol{t}_{c-1}^{c}\right)^T\right]^T, \tag{24}$$

$$\boldsymbol{f}_S = \mathbf{V}_S^{-1}\overline{\mathbf{V}}_S\left[\left(\theta'^{c}_{c-1}\right)^T,\ \left(\boldsymbol{t}'^{c}_{c-1}\right)^T\right]^T. \tag{25}$$

Note that $\boldsymbol{f}_C$ only contains solved motion in a subspace of the state space. We take the motion from the IMU prediction, namely $\hat{\theta}_{c-1}^{c}$ and $\hat{\boldsymbol{t}}_{c-1}^{c}$, and project it to the nullspace of $\boldsymbol{f}_C$,

$$\boldsymbol{f}_I = \mathbf{V}_V^{-1}(\mathbf{V}_V - \overline{\mathbf{V}}_V)\mathbf{V}_S^{-1}(\mathbf{V}_S - \overline{\mathbf{V}}_S)\left[\left(\hat{\theta}_{c-1}^{c}\right)^T,\ \left(\hat{\boldsymbol{t}}_{c-1}^{c}\right)^T\right]^T. \tag{26}$$

Next, we use $\tilde{\theta}^c_{c-1}(\boldsymbol{b}_\omega(t))$ and $\tilde{\boldsymbol{t}}^c_{c-1}(\boldsymbol{b}_\omega(t), \boldsymbol{b}_a(t))$ to denote the IMU-predicted motion formulated as functions of $\boldsymbol{b}_\omega(t)$ and $\boldsymbol{b}_a(t)$), through integration of (3) and (4). The orientation $\tilde{\theta}^c_{c-1}(\boldsymbol{b}_\omega(t))$ is only relevant to $\boldsymbol{b}_\omega(t)$, but the translation $\tilde{\boldsymbol{t}}^c_{c-1}(\boldsymbol{b}_\omega(t), \boldsymbol{b}_a(t))$ is dependent on both $\boldsymbol{b}_\omega(t)$ and $\boldsymbol{b}_a(t)$). The biases can be calculated by solving the following equation,

$$\left[\left(\tilde{\theta}^c_{c-1}(\boldsymbol{b}_\omega(t))\right)^T,\ \left(\tilde{\boldsymbol{t}}^c_{c-1}(\boldsymbol{b}_\omega(t), \boldsymbol{b}_a(t))\right)^T \right]^T = \boldsymbol{f}_C + \boldsymbol{f}_I. \qquad (27)$$

When the pipeline functions normally, $\boldsymbol{f}_C$ spans the state space, and $\mathbf{V}_V - \overline{\mathbf{V}}_V$ and $\mathbf{V}_S - \overline{\mathbf{V}}_S$ in (26) are zero matrices. Correspondingly, $\boldsymbol{b}_\omega(t)$ and $\boldsymbol{b}_a(t)$ are calculated from $\boldsymbol{f}_C$. In a degraded case, the IMU-predicted motion, $\hat{\theta}^c_{c-1}$ and $\hat{\boldsymbol{t}}^c_{c-1}$, is used in directions where the motion is unsolvable (e.g., white row of the combined feedback in Figure 14). The result is that the previously calculated biases are kept in these directions.

9 | LOCALIZATION ON EXISTING MAPS

When a map is available, our data processing pipeline can be extended to utilize the map for localization. This uses a scan matching method similar to Section 6. The method extracts and matches two types of geometric features, on edges and local planar surfaces. The feature points from the map are precomputed and stored in voxels. By matching feature points from scans to the map, the localization solves an optimization problem minimizing the overall distances between the feature points and their correspondences.

In comparison to Section 6, the difference is that the localization does not match individual scans but stacks a number of scans for batch processing. Scans are stacked only when the sensor suite is moving, to prevent redundant data being collected from the same location. Thanks to the high-accuracy ego-motion estimation, scans are precisely registered in a local coordinate frame where drift is negligible over seconds of time. Figure 15a shows a single scan utilized in Section 6 for scan matching at 5 Hz, and Figure 15b presents stacked scans over 2 s and matched in the localization at 0.5 Hz. One can see the stacked scans contain more detailed structural information. Further, execution at a low frequency keeps the CPU usage to be minimal (about 10% of a CPU thread) for onboard processing.

10 | EXPERIMENTS

10.1 | Tests with single-axis scanners

Our odometry and mapping method is first validated on two sensor suites. In Figure 16a, a Velodyne HDL-32E laser scanner is attached to a UI-1220SE monochrome camera and an Xsens MTi-30 IMU. The laser scanner has 360° horizontal field of view (FOV), 40° vertical FOV, and receives 0.7 million points per second at 5 Hz spinning rate. The camera is configured at the resolution of 752 × 480 pixels, 76° horizontal FOV, and 50 Hz frame rate. The IMU frequency is set at 200 Hz. In Figure 16b, a Velodyne VLP-16 laser scanner is attached to the same camera and IMU. This laser scanner has 360° horizontal

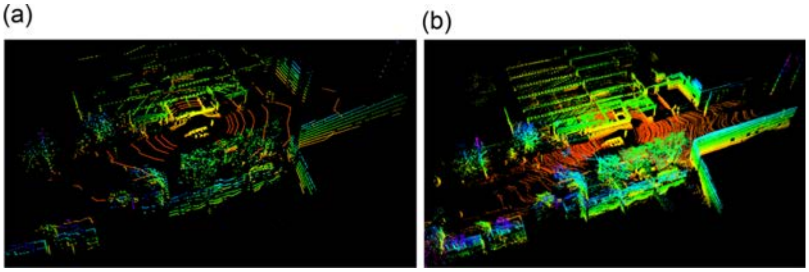


FIGURE 15 (a) Scans involved in Section 6 for ego-motion estimation, and (b) in localization on an existing map. In Section 6, each individual scan is matched at 5 Hz. While in localization, a number of locally registered scans are stacked and matched at 0.5 Hz [Color figure can be viewed at wileyonlinelibrary.com]

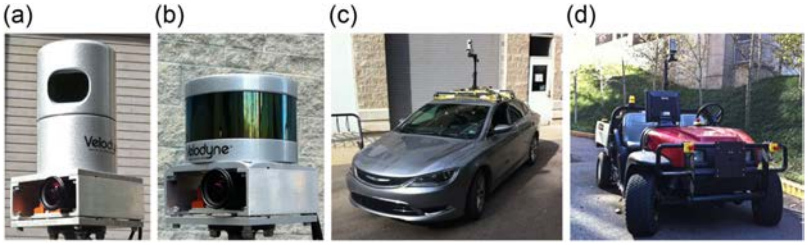


FIGURE 16 Sensor suites and vehicles used in experiments. (a) It is a Velodyne HDL-32E laser scanner attached with a uEye UI-1220SE monochrome camera and an Xsens MTi-30 IMU. (b) It is a Velodyne VLP-16 laser scanner attached with the same camera and IMU. (c) It is a passenger vehicle for street driving. (d) It is a utility vehicle for off-road driving. Each sensor suite in (a) and (b) is attached to both vehicles in (c) and (d) for experiment validation. IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Average CPU processing time using the sensor suites in Figure 16a,b

		Visual–inertial odometry (time per image frame)		
Environment	Senor suite	GPU tracking (ms)	CPU tracking (ms)	Scan matching (time per laser scan; ms)
Structured	Figure 16a	4.8	14.3	148
	Figure 16b	4.2	12.9	103
Vegetated	Figure 16a	5.5	15.2	267
	Figure 16b	5.1	14.7	191

Note. CPU: central processing unit; GPU: graphics processing unit.

FOV, 30° vertical FOV, and receives 0.3 million points per second at 5 Hz spinning rate. Both sensor suites are attached to the vehicles in Figure 16c,d for data collection, which are driven on streets and in off-road terrains, respectively.

For both sensor suites, we track maximally 300 Harris corners using the Kanade–Lucas–Tomasi method (Lucas & Kanade, 1981). To evenly distribute the visual features, an image is separated into 5 × 6 identical subregions, and each subregion provides up to 10 features. When a feature loses tracking, a new feature is generated to maintain the feature number in each subregion.

The software runs on a laptop computer with a 2.6-GHz i7 quad-core processor (two threads on each core and eight threads overall) and an integrated GPU, in a Linux system running robot operating system (Quigley et al., 2009). We implement two versions of the software with visual feature tracking running on GPU and CPU, respectively. The processing time is shown in Table 2. The time used by the visual–inertial odometry (middle module in Figure 2c) does not vary much w.r.t. the environment or sensor configuration. For the GPU version, it consumes about 25% of a CPU thread executing at 50 Hz. For the CPU version, it takes about 75% of a thread. The sensor suite in Figure 16a results in slightly more processing time than the one in Figure 16b. This is because the scanner receives more points and the program needs more time to maintain the depthmap and associate depth to the visual features.

The scan matching (rightmost module in Figure 2c) consumes more processing time which also varies w.r.t. the environment and sensor configuration. With the sensor suite in Figure 16a, the scan matching takes about 75% of a thread executing at 5 Hz if operated in structured environments. In vegetated environments, however, more points are registered on the map and the program typically consumes about 135% of a thread (running on multiple threads). With the sensor suite in Figure 16b, the scanner receives less number of points. The scan matching uses about 50–95% of a thread depending on the environment. The time used by the IMU prediction (leftmost module in Figure 2c) is neglectable compared to the other two modules.

10.1.1 | Accuracy tests

We first conduct tests to evaluate accuracy of the proposed method. In these tests, the sensor suite in Figure 16a is used. We first mount the sensors on the vehicle in Figure 16d driving around the university campus. After 2.7 km of driving within 16 min, a campus map is built (shown in Figure 17a). The average speed over the test is 2.8 m/s. In addition to the overall map, we present three close views on the right for readers to inspect the local registration accuracy. The corresponding locations are labeled with numbers 1–3 on the map.

To evaluate motion estimation drift over the test, we align the estimated trajectory and registered laser points on a satellite image in Figure 17b. Here, laser points on the ground are manually removed. By matching the trajectory with streets on the satellite

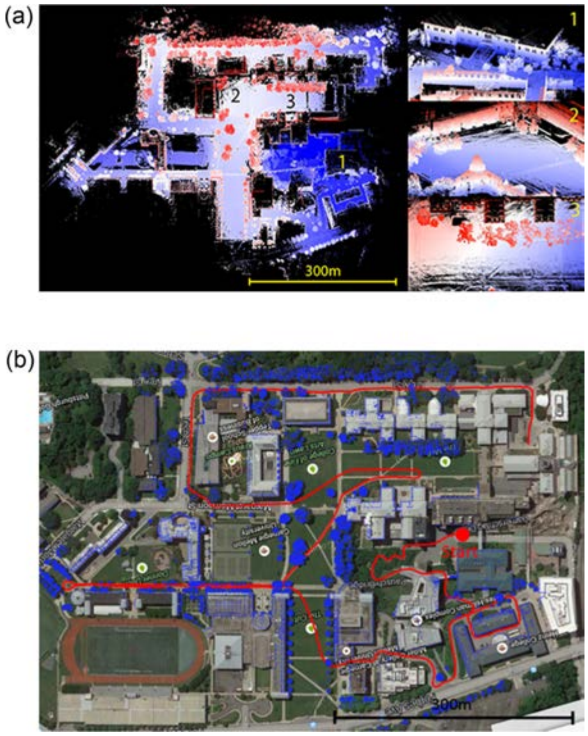


FIGURE 17 Accuracy Test 1. The sensor suite in Figure 16a is mounted on the vehicle in Figure 16d to map the university campus. The overall path is 2.7 km in length, finished in 16 min with an average driving speed of 2.8 m/s. (a) It shows the map built and three close views labeled with numbers 1–3. The corresponding locations on the map are marked with the same numbers. (b) It shows the vehicle trajectory (red) and registered laser points (blue) overlayed on a satellite image. Laser points on the ground are manually removed. The relative position drift at the end is < 0.09% of the distance traveled [Color figure can be viewed at wileyonlinelibrary.com]

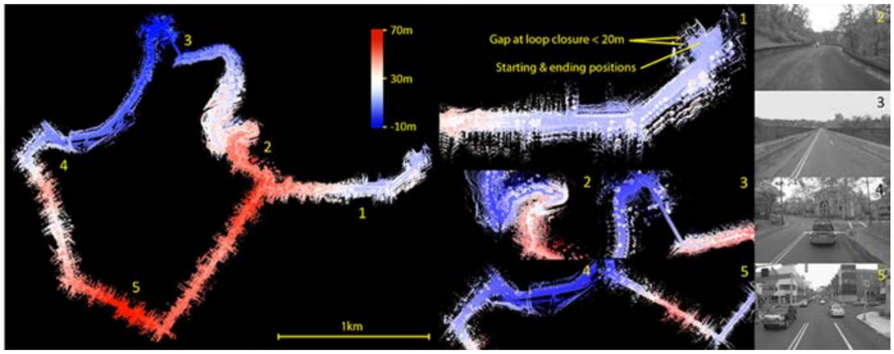


FIGURE 18 Accuracy Test 2. The sensor suite in Figure 16a is mounted on the vehicle in Figure 16c for 9.3 km of street driving. The path goes through vegetated environments, bridges, hilly terrains, and roads with heavy traffic. The elevation changes over 70 m. Except waiting for traffic lights, the vehicle is driven at 9–18 m/s. On the left, we show the complete map color coded by elevation. On the right, we show a few close views with locations labeled with numbers 1–5 on the map. In close view 1, we present the starting and the ending positions. Because of drift, a building is registered into two, one during the vehicle leaves from the start and the other during the vehicle returns at the end. We manually measure the gap to be < 20 m, resulting in a relative position error at the end to be < 0.22% of the distance traveled. Close views 2–5 show more details with corresponding images logged by the camera [Color figure can be viewed at wileyonlinelibrary.com]

image, we are able to determine an upper bound of the horizontal error to be < 1.0 m. By comparing buildings on the same floor, we further determine the vertical error to be < 2.0 m. This gives an overall relative position drift at the end to be < 0.09% of the distance traveled. We are aware that precision cannot be guaranteed for the measurements; hence, we only calculate an upper bound of the position drift.

Further, we conduct a more comprehensive test with the same sensors mounted on the vehicle in Figure 16c. The vehicle is driven on structured roads for 9.3 km of travel. As shown in Figure 18, the path goes through vegetated environments, bridges, hilly terrains, and streets with heavy traffic, and finally returns to the starting position. The elevation changes over 70 m along the path. Except waiting for traffic lights, the vehicle speed is between 9 and 18 m/s during the test. On the left side of Figure 18, we show the complete map color coded by elevation. On the right, we present a few close views with corresponding locations labeled with numbers 1–5 on the map. In particular, close view 1 shows the starting and the ending positions. Carefully examining the figure, we see that a building is registered into two. This is because of motion estimation drift over the path, while one is registered when the vehicle leaves from the start and the other when the vehicle returns at the end. We measure the gap to be < 20 m, which results in a relative position error at the end to be < 0.22% of the distance traveled. We show more details in close views 2–5 with corresponding images logged by the camera.

Additionally, we examine how each module in the pipeline contributes to the overall accuracy. As shown in Figure 19, we first plot output of the visual–inertial odometry as the green dash–dot curve. This uses the left two modules in Figure 2c. Next, we directly forward the IMU prediction to the scan matching module, bypassing the visual–inertial odometry. This configuration uses the leftmost and the rightmost modules in Figure 2c. The result is drawn as the

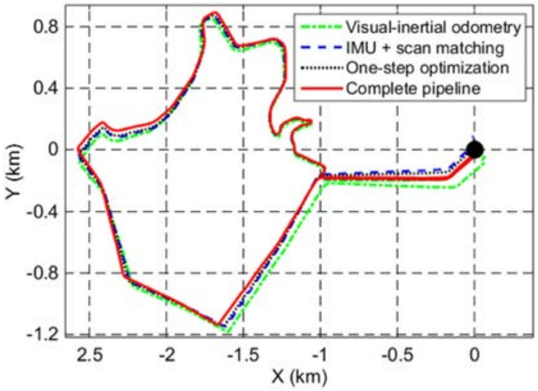


FIGURE 19 Estimated trajectories in Accuracy Test 2. The trajectories start with the black dot. We compare four configurations in the test. The green dash-dot curve is from the visual–inertial odometry module (using the left two modules in Figure 2c). The blue dash curve is from the scan matching module with the IMU prediction directly taken as input (leftmost and rightmost modules in Figure 2c). The black dot curve has the pipeline reconfigured to solve one large optimization problem incorporating all constraints, as in Figure 2b. The red solid curve is from the proposed data processing pipeline. IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 3 Comparsion of relative position errors as percentages of the distance traveled in Accuracy Test 2

Configuration	1× speed (%)	2× speed (%)
Visual–inertial odometry	0.93	1.47
IMU + scan matching	0.51	0.89
One-step optimization	0.48	1.02
Complete pipeline	0.22	0.26

Note. The errors at 1× speed correspond to the trajectories in Figure 19. IMU: inertial measurement uni.

TABLE 4 Further comparsion of relative position errors in Accuracy Test 2, with two existing methods, LOAM (Zhang & Singh, 2014) and V-LOAM (Zhang & Singh, 2015)

Method	LOAM	V-LOAM	Complete pipeline
Accuracy	0.39%	0.33%	0.22%

blue dash curve. Finally, we plot output of the complete pipeline as the red solid curve, with the least amount of drift. The position errors of the first two configurations are about four and two times larger.

We can consider the green dash-dot curve and the blue dash curve as the expected performance when encountering individual sensor degradation: If scan matching is degraded, the pipeline reduces to a mode indicated by the green dash–dot curve; if vision is degraded, the pipeline reduces to that indicated by the blue dash curve. Further, we reconfigure the pipeline to incorporate all constraints in one large optimization problem as in Figure 2b. It takes the IMU prediction as the initial guess and runs at the laser scanning frequency (5 Hz). The method produces a trajectory as the black dot curve. The resulting accuracy is only little better in comparison to the blue dash curve which uses the IMU directly coupled with the laser scanner, passing the visual–inertial odometry. The result indicates that the high-frequency advantage of the camera is unexplored if solving the problem with all constraints stacked together.

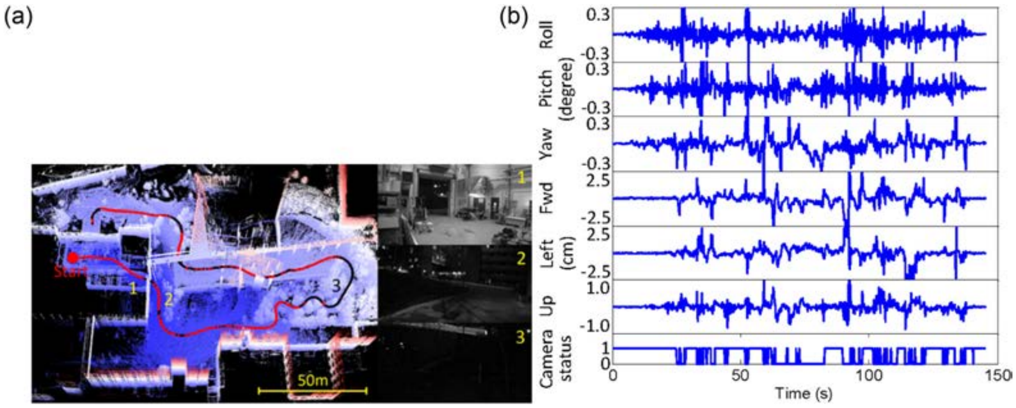


FIGURE 20 Robustness Test 1. The sensor suite in Figure 16b is attached to the vehicle in Figure 16d driven from indoor to outdoor. The test is conducted at night. Frequently, the camera cannot capture enough visual features and the visual–inertial odometry module is bypassed. In (a), we show the estimated trajectory overlayed on the map built. The red segments indicate vision is functional and the black segments indicate degradation. Also, we show three images logged by the camera from locations 1–3 labeled on the map. Location 1 is indoor and locations 2–3 are outdoor. In (b), we show pose corrections applied by the scan matching to refine motion estimates. On the bottom row, the camera status being one indicates functioning. When the camera status is zero, corrections on the top six rows become larger because the IMU prediction produces more drift than the visual–inertial odometry. IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

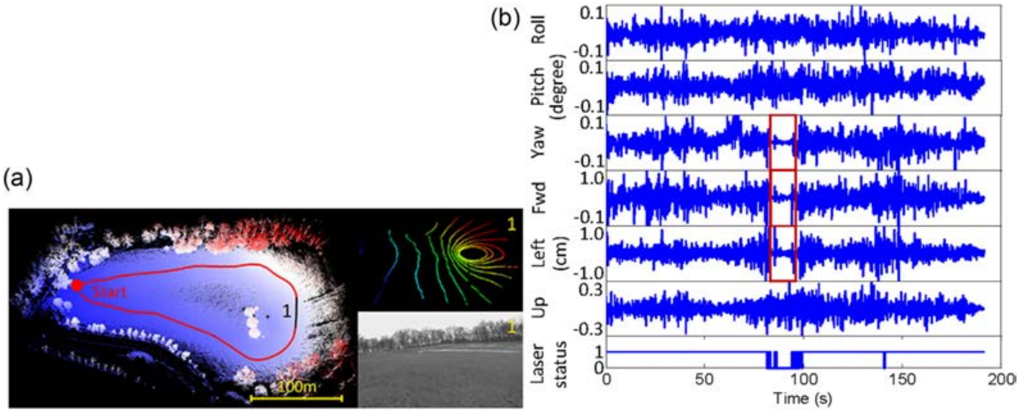


FIGURE 21 Robustness Test 2. The sensor suite in Figure 16b is attached to the vehicle in Figure 16d driven in an off-road terrain. In (a), when the vehicle reaches the rightmost side of the path, only the flat ground is seen, causing the scan matching to partially degrade. The corresponding trajectory is drawn in black. Here, we determine the scan matching is able to refine 3-DOF out of the 6-DOF motion, which are roll, pitch, and elevation. The other 3-DOF are unsolvable due to the planar scene, where the pose is directly taken from the visual–inertial odometry. In addition, we show a laser scan and an image logged from Location 1 labeled on the map. In (b), we show pose corrections applied by the scan matching. On the last row, the laser status being one indicates functioning. When the laser status is zero, pose corrections in degraded directions (in the red boxes) become much smaller. DOF: degree of freedom [Color figure can be viewed at wileyonlinelibrary.com]

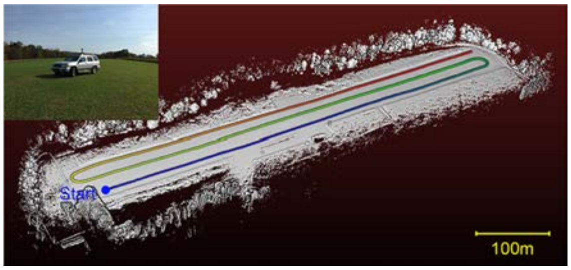


FIGURE 22 Robustness Test 3. Succeeding test of laser degradation on a flat airport runway. The airport is about 500 m long. The path starts in blue and ends in red, traversing the runway three times. The overall length of the trajectory is over 1.6 km. During the test, the scan matching module is partially bypassed similar to Figure 21 [Color figure can be viewed at wileyonlinelibrary.com]

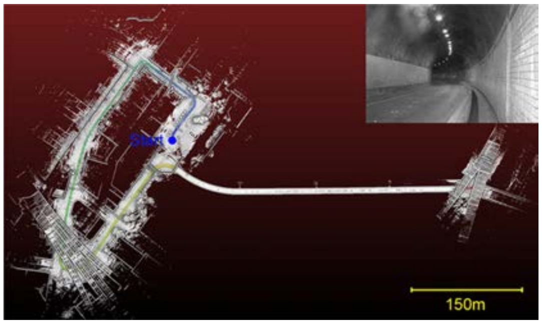


FIGURE 23 Robustness Test 4. Succeeding test of laser degradation in a smooth tunnel. The tunnel is 380 m in length, with a 45° curve close to its left end (shown in the photo at the upper-right corner). Similar to Figure 21, the scan matching module is partially bypassed while passing through the tunnel [Color figure can be viewed at wileyonlinelibrary.com]

We would like to further understand how the modules incorporate in the pipeline. To this end, we compare accuracy of the pipeline running at the original 1× speed and an accelerated 2× speed. When running at 2× speed, we skip every other data frame for all three sensors, resulting in much more aggressive motion through the test. The results are listed in Table 3. At each speed, we evaluate three configurations. As we can see, when running at 2× speed, the accuracy of the visual–inertial odometry and the IMU + scan matching configurations reduce significantly, by 0.54% and 0.38% of the distance traveled in comparison to the accuracy at 1× speed. However, the complete pipeline reduces accuracy very little, only by 0.04%. The results indicate that the camera and the laser scanner compensate for each other keeping the overall accuracy. This is especially true when the motion is aggressive.

Finally, we compare the proposed method with two of our existing methods. LOAM (Zhang & Singh, 2014) is a method that uses an IMU and a 3D laser scanner for ego-motion estimation. V-LOAM (Zhang & Singh, 2015) uses a camera and a 3D laser scanner instead. Using data from Accuracy Test 2, the results are shown in Table 4. Here, we can see that the complete pipeline utilizing all laser, visual, and inertial data produces the highest accuracy among the three.

10.1.2 | Robustness tests

We further inspect the robustness w.r.t. sensor failures. Specifically, we carry out test cases to produce vision degradation due to low-light and scan matching degradation due to lack of structures. Here, it is worth to mention that the same mechanism in the pipeline that ensures the robustness w.r.t. environmental degradation also warrants the robustness w.r.t. and aggressive motion. This is because both affect the processing in a similar way. The problem caused by

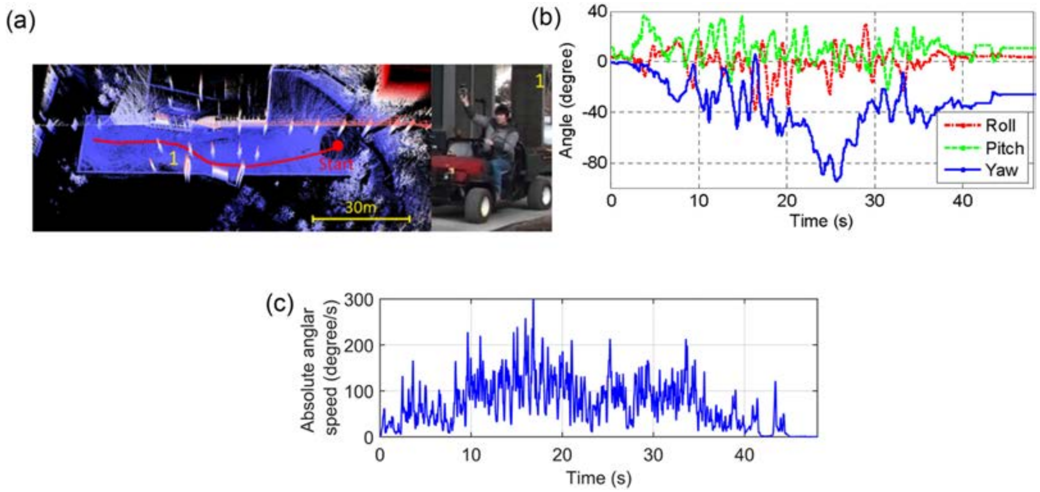


FIGURE 24 Aggressive Motion Test 1. The sensor suite in Figure 16b is held by a person in one hand who drives the vehicle in Figure 16d with the other hand. The person oscillates the sensor suite to introduce fast rotation. (a) It shows the estimated trajectory overlayed on the map built and a photo taken from Location 1 during the test. (b) It shows the estimated orientation. (c) This is the estimated absolute angular speed. The maximum angular speed exceeds 250°/s [Color figure can be viewed at wileyonlinelibrary.com]

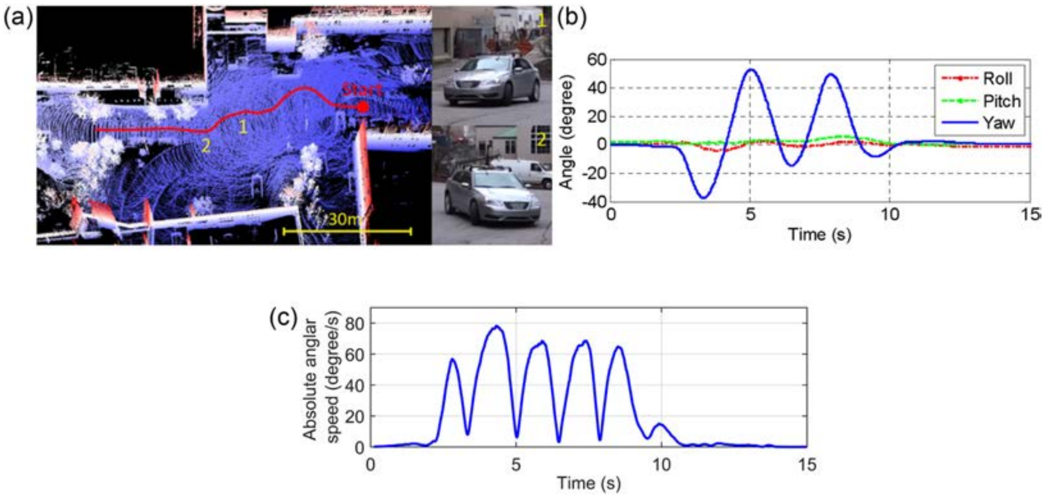
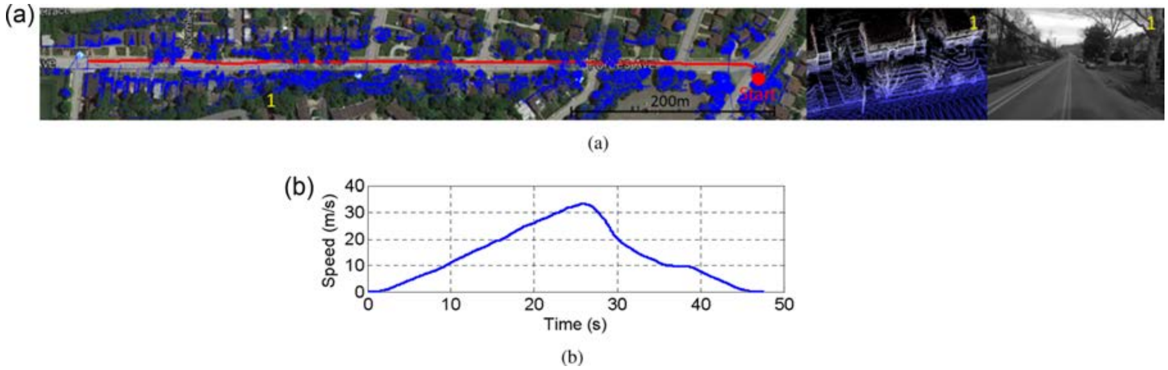


FIGURE 25 Aggressive Motion Test 2. The sensor suite in Figure 16b is mounted to the vehicle in Figure 16c. The vehicle is driven along an "S"-shaped path. The sensor rotates over 330° within 8 s during the test [Color figure can be viewed at wileyonlinelibrary.com]



(a)

(b)

FIGURE 26 Aggressive Motion Test 3. The sensor suite in Figure 16(b) is mounted to the vehicle in Figure 16(c), driven at a high speed along the red path. The overall path is 701 m and the maximum linear speed is 33 m/s. In (a), the blue points are laser points overlayed on a satellite image. We show three mapped houses and a corresponding image taken from Location 1 labeled on the satellite image. Meanwhile, the three houses are on the left side of the image. In (b), we present the estimated linear speed through the test [Color figure can be viewed at wileyonlinelibrary.com]

environmental degradation is due to the fact that the environment does not contain sufficient information, resulting in an ill-conditioned problem as discussed in Section 8. The problem caused by aggressive motion is because of sparsity of the data in which insufficient information is captured from the environment, resulting in the same ill-conditioned problem. Both problems are handled using the method introduced in Section 8.

The experiments use the sensor suite in Figure 16b attached to the vehicle in Figure 16d. First, we drive the vehicle at night where vision degrades. When an insufficient number of visual features are tracked, the visual–inertial odometry module is bypassed, and the IMU prediction is directly sent to the scan matching module. As shown in Figure 20a, the red and the black segments on the trajectory respectively indicate vision is functional and degraded. In Figure 20b, we show pose corrections applied by the scan matching for motion estimation refinement. On the bottom row, the camera status being zero indicates degradation. Correspondingly, pose

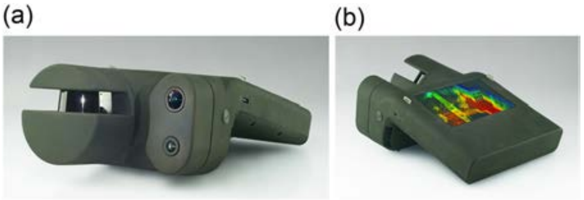


FIGURE 27 (a) Front and (b) back views of a custom-built Contour. The device includes a spinning 2D Hokuyo UTM-30LX-EW laser scanner functioning as a 3D scanner, a wide-angle camera at 640 × 512 pixels for motion estimation, an HD color camera at 1600 × 1200 pixels for point cloud colorization, and an Xsens MTi-20 IMU. The device is also equipped with an embedded i7 computer for online data processing and a touch-screen monitor. IMU: inertial measurement unit [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 5 Average CPU processing time on Contour in Figure 27

	Visual–inertial odometry (time per image frame; ms)		
Environment	**GPU tracking**	**CPU tracking**	**Scan matching (time per laser scan; ms)**
Structured	6.4	16.7	162
Vegetated	6.9	18.7	343

Note. CPU: central processing unit; GPU: graphics processing unit.

corrections on the top six rows become larger because the IMU prediction is less precise in comparison to the visual–inertial odometry.

Next, we bring the vehicle to an open area where scan matching degrades due to the planar environment. As shown in Figure 21a, when the vehicle research the rightmost side of the path (black segment), only the flat ground is seen by the laser scanner. The method determines the scan matching is able to refine 3-DOF out of the 6-DOF motion using the method introduced in Section 8.

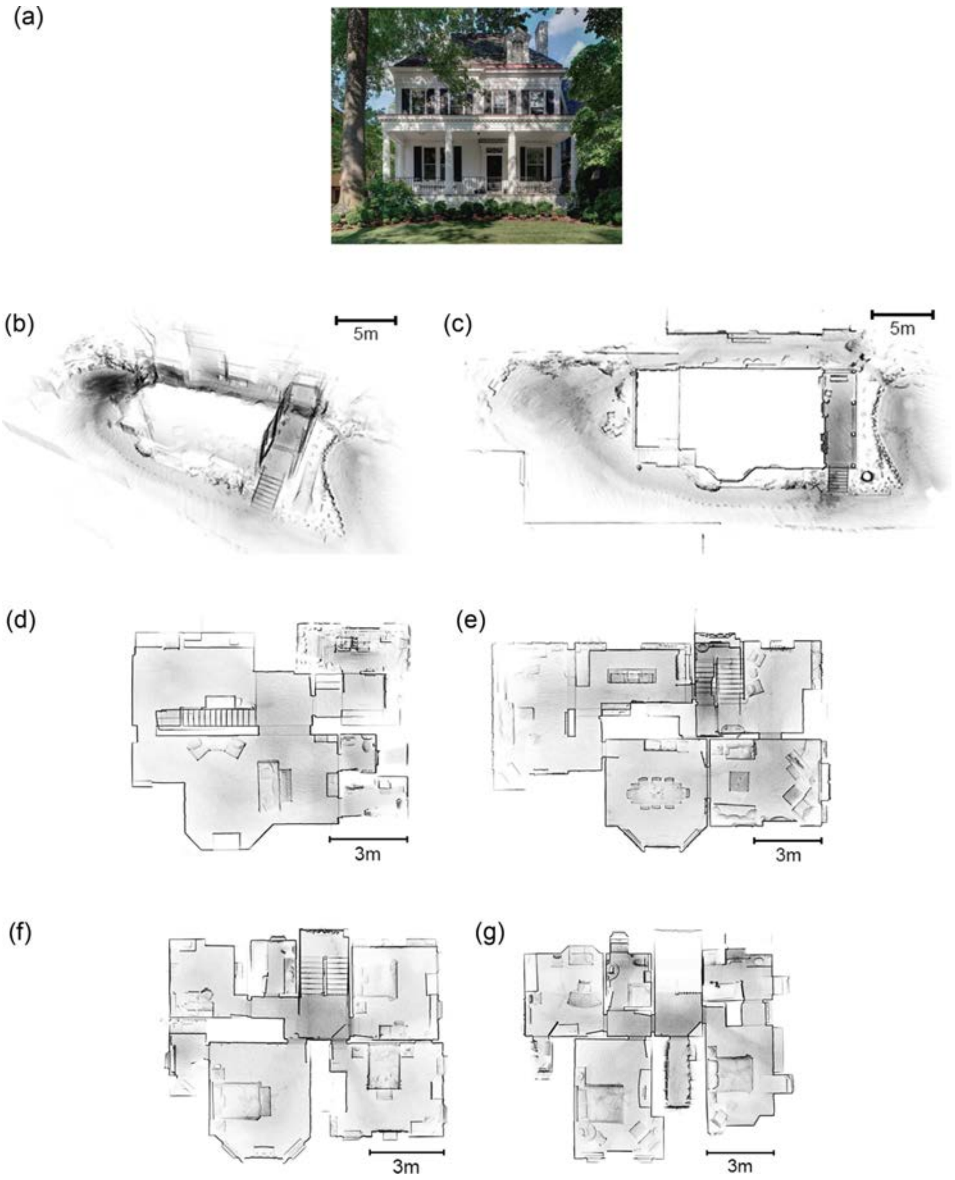


FIGURE 28 (a) A photo and (b)–(g) maps built from a four-floor residential house using Contour in Figure 27. (b) and (c) are horizontal and top-down views of the surrounding. (d–g) These are, respectively, the basement, first floor, second floor, and third floor of the interior. The device is moved at around 0.5 m/s over the test [Color figure can be viewed at wileyonlinelibrary.com]

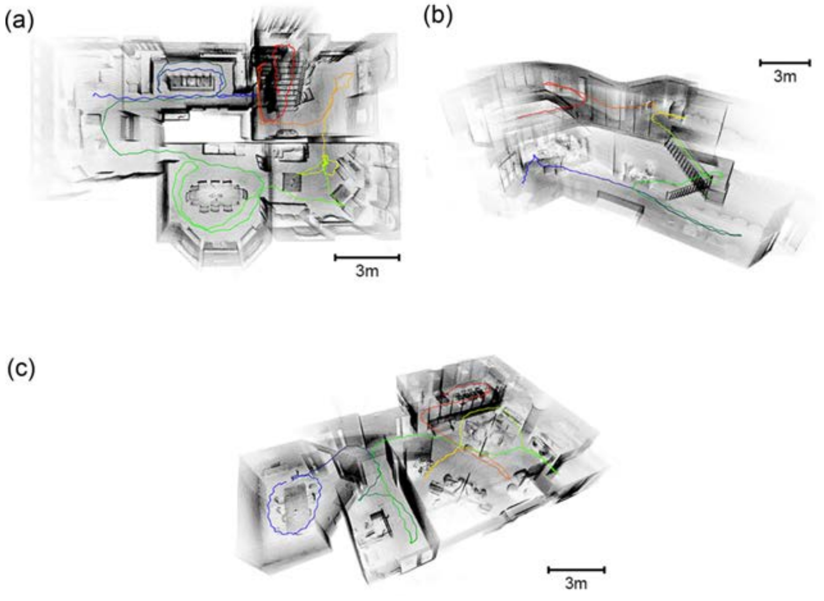


FIGURE 29 Three maps built by Contour in Figure 27 with sensor paths inserted. (a) This is the same area as Figure 28e. (b) and (c) are from a different building. The device is moved at around 0.5 m/s. Each of the three tests lasts about 2 min [Color figure can be viewed at wileyonlinelibrary.com]

Specifically, roll, pitch, and elevation are well-conditioned, but yaw, forward, and left are unsolvable due to the planar scene. The visual–inertial odometry output is used directly in the degraded directions. In Figure 21b, we show pose corrections applied by the scan matching. On the bottom row, the laser status being zero indicates partial degradation. Correspondingly, corrections in the degraded directions (labeled in the red boxes) are much smaller because the corrections are only applied in well-conditioned directions of the problem (rightmost module in Figure 2c) as determined by the method in Section 8.

Further, we show in Figures 22 and 23 succeeding results of laser degradation. Figure 22 is conducted on a flat airport runway which measures about 500 m in length. The figure shows the registered map and sensor path starting in blue and ending in red. The overall length of the trajectory is over 1.6 km. A photo from the test site is shown at the upper-left corner. Figure 23 shows a smooth tunnel that is 380 m long. In both tests, the method encounters laser degradation similar to Figure 21, and corresponding, the scan matching module is partially bypassed to produce these results.

10.1.3 | Aggressive motion tests

In this section, we evaluate the method performance w.r.t. high-speed rotation and translation. As we have discussed, the pipeline uses the method in Section 8 to hold the robustness w.r.t. aggressive motion. The tests use the sensor suite in Figure 16b. First, the sensors are held by a person who drives the vehicle in Figure 16d. The vehicle carries power supply and a data processing computer.

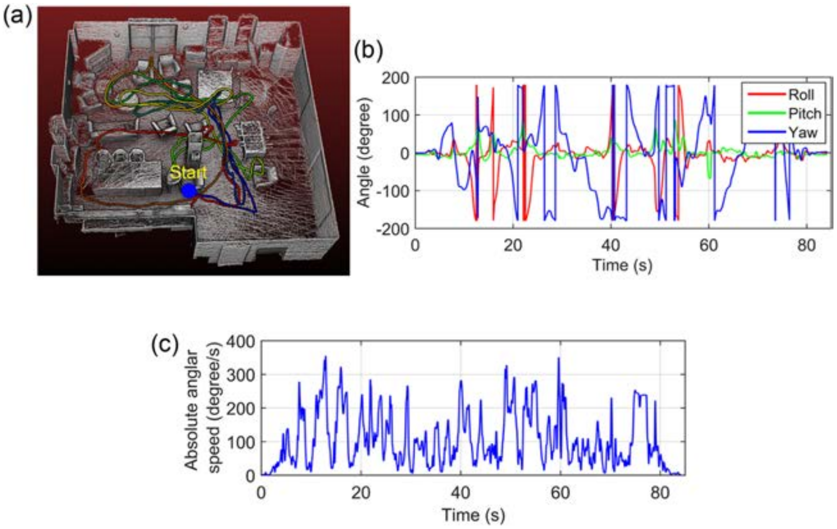


FIGURE 30 Aggressive motion test with Contour in Figure 27. The device is hand-carried and rotated fast inside a room. (a) It shows the estimated trajectory overlayed on the map built. (b) It shows the estimated orientation. (c) This is the estimated absolute angular speed. The maximum angular speed is as high as 370°/s. During 82 s of the test, the accumulated rotation reaches $6.8^{\circ} \times 10^{3}$, leading to an average angular speed of 83°/s [Color figure can be viewed at wileyonlinelibrary.com]

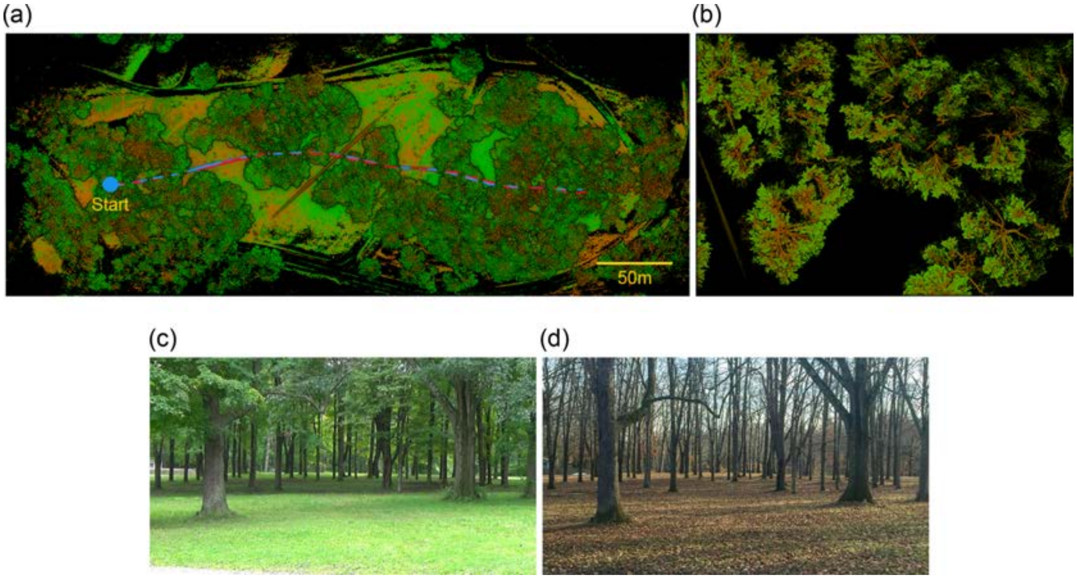


FIGURE 31 Test of localization robustness w.r.t. environmental changes. The test uses the sensor suite in Figure 16b. A map is built by walking through the forest in the summer, and a localization run is made by walking one more time in the winter. The speed is about 2 m/s for both runs. (a) It shows the overall merged map where green and brown points are from the summer and winter, respectively. The blue trajectory is associated with the green points and the red trajectory is associated with the brown points. (b) It shows a close view of the merged map, where one can see the brown points are only on tree branches and the green points are on both tree branches and leaves. (c) and (d) are two photos of the same area in the forest in the summer and winter. We can see dramatic changes in the environment [Color figure can be viewed at wileyonlinelibrary.com]

The person oscillates the sensor suite to introduce fast rotation. Next, the sensors are mounted to the vehicle in Figure 16c driven along an "S"-shaped path. Results of the two tests are in Figures 24 and 25. In Figures 24a and 25a the estimated trajectories are overlayed on the maps built, with photos showing experiment setups. In Figures 24b and 25b the estimated orientations are present. In Figures 24c and 25c the estimated absolute angular speeds are shown. For the first test, the maximum angular speed exceeds 250°/s. For the second test, the accumulative rotation is over 330° in 8 s.

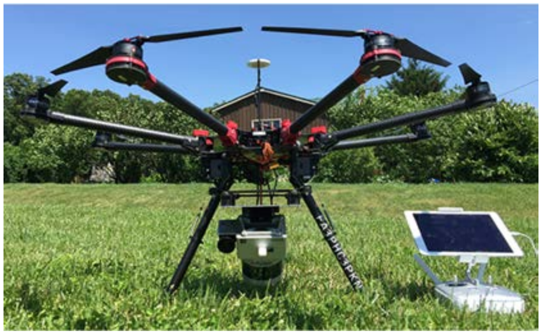


FIGURE 32 Drone platform used in air-ground collaborative mapping test. This is a DJI S1000 aircraft mounted with a sensor suite which has the same sensor setup as in Figure 16b [Color figure can be viewed at wileyonlinelibrary.com]

Finally, we mount the sensors on the vehicle in Figure 16c and drive along a straight path at a high speed. As shown in Figure 26a, the overall path is 701 m in length. The blue points are laser points registered and overlayed on a satellite image. We see the mapped trees and houses are well aligned with the satellite image. Through this comparison, we believe the horizontal position error is < 1.0 m, resulting in a horizontal position drift to be < 0.15% of the distance traveled. For the vertical drift, however, we do not have a means to evaluate. We show three mapped houses in close views on the right side of Figure 26a, and a corresponding image taken from Location 1 on the satellite image. The houses are on the left side of the image. In Figure 26b, we plot the linear speed. The maximum speed reaches as high as 33 m/s (119 km/hr or 74 mi/hr).

10.2 | Tests with custom-built Contour

Our odometry and mapping method is further validated on a custom-built Contour. As shown in Figure 27, the device includes a 2D Hokuyo UTM-30LX-EW laser scanner acquiring 43.2 thousand points per second. The laser scanner is attached to a motor-encoder shaft spinning at 1 Hz, functioning as a 3D scanner. The device also includes a wide-angle camera configured at the resolution of 640 × 512 pixels for motion estimation, an HD color camera at 1600 × 1200 pixels for point cloud colorization, and an Xsens MTi-20 IMU. An onboard embedded computer with an 1.8 GHz i7 dual-core processor (four threads overall) runs the data processing software and displays mapping results on a touch-screen monitor in real time.

We use the same data processing pipeline as in Figure 2c except that the scan matching module runs at 1 Hz instead of 5 Hz. This is due to the fact that the 3D scanner on Contour has a lower spinning rate. We track maximally 300 visual features. The CPU processing time is shown in Table 5. Note that the embedded computer in Contour is less powerful than the laptop we test with the Velodyne scanners. When running feature tracking on GPU, the visual–inertial odometry consumes about 35% of a CPU thread executing at 50 Hz. If running feature tracking on CPU, however, the processing time is about 85% of a thread. The scan matching takes between 15 and 35% of a thread executing at 1 Hz, depending on the type of environment.

Figure 28 shows results from a four-floor residential house. Figure 28a is a photo of the house. Figure 28b,c is maps of surrounding of the house, in perspective view and top-down view. Figure 28d–g is, respectively, the basement, first floor, second floor, and third floor of the interior. Further, we show three maps in Figure 29 with the sensor paths inserted. Here, Figure 29a is the same area as Figure 28e. Figure 29b,c is from a different building. In all tests, the device is moved at a speed around 0.5 m/s, resulting in each of the three maps in Figure 29 built in about 2 min. The exterior map in Figure 28b,c takes about 6 min due to its larger scale and more details to cover. Due to difficulty to acquire ground truth, meanwhile, we can only let readers visually inspect quality of the maps.

Finally, we evaluate the performance in aggressive motion. Contour is hand-carried by a person who rotates the device fast inside a room while walking and traversing the room. The results are shown in Figure 30. In Figure 30a, we present the estimated trajectory overlayed on the map built. In Figure 30b, we show the estimated orientation. In Figure 30c, we show the estimated absolute angular speed. From the results, we can see that the maximum angular speed is up to 370°/s. During 82 s of the test, the accumulated rotation is about 6. 8° × 10^3. This results in an average angular speed of 83°/s over the test.

10.3 | Tests of localization and map merging

We verify localization robustness w.r.t. environmental changes. The experiment uses the sensor suite in Figure 16b. As shown in Figure 31, the test is conducted in a forest. A map is built by an operator holding the sensor suite and walking through the environment in the summer. The map covers over 300 m area. Then, a

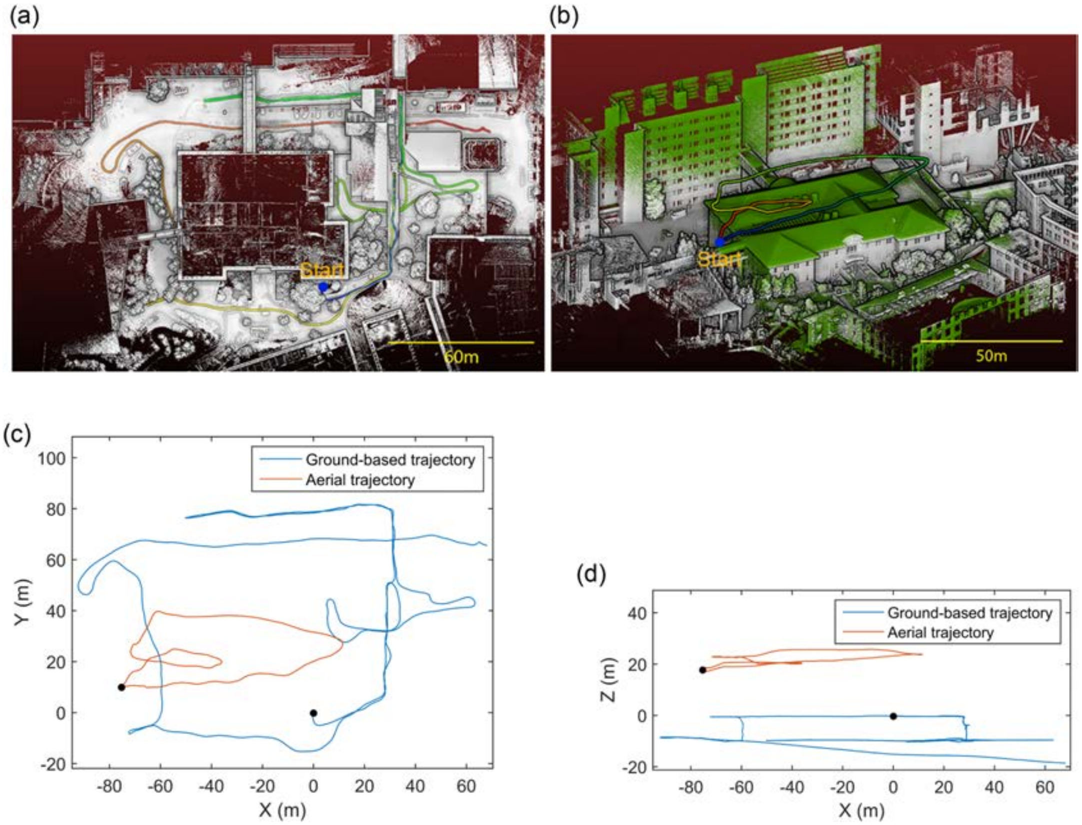


FIGURE 33 Air-ground collaborative mapping test. (a) It shows the ground-based map and sensor path produced by an operator holding the sensor suite and walking around a building at 1–2 m/s for 914 m of travel. The path starts in blue and ends in red. In (b), the same sensor suite is mounted to the drone in Figure 32 and flown over the building at 2-3 m/s for 269 m. The green points are mapped from the air and the colored curve is the sensor path. (c) and (d) are sensor paths from the ground and air in top-down and side views, respectively. The paths start from the black dots [Color figure can be viewed at wileyonlinelibrary.com]

localization run is made in the winter by waling through the environment one more time. The walking speed is about 2 m/s for both runs. Figure 31a shows the overall map merged from the summer and winter, where green and brown points are from the summer and winter, respectively. The blue trajectory is associated with the green points, and the red trajectory is associated with the brown points. Figure 31b presents a close view of the merged map. We can see that the brown points are only on tree branches while the green points are on both tree branches and leaves. From inspecting the merged map, we estimate the localization error is <2 cm through the test. By offline reprocessing logged data, we validate that the method produces the same result by localizing summer data on a map built in the winter. Figure 31c,d shows two photos from the same area in the forest as data is logged in the summer and winter. We can see dramatic environmental changes between the two runs.

Finally, we conduct a test to validate the localization with the drone in Figure 32. This is a DJI S1000 aircraft mounted with a sensor suite on the bottom. The sensor suite has the same setup as in Figure 16b. The test starts with ground-based mapping where an operator holds the sensor suite and walks around a building. The operator walks at 1–2 m/s over 914 m of travel, following the colored trajectory. A map is produced as in Figure 33a which covers surroundings of the building in detail. As expected, the roof of the building is empty on the map. Second, the sensor suite is attached to the drone to fly over the building. The flight is at 2–3 m/s for 269 m. During the flight, we utilize the ground-based map for localization. The localized scans form an aerial map sharing the same coordinate frame as the ground-based map, shown as the green points in Figure 33b. The colored curve is the sensor path in the air. In Figure 33c,d, we present the sensor paths from the ground and air, in top-down and side views. In this test, the localization takes 214 ms for processing on average. Running at 0.5 Hz, it consumes 10.7% of a CPU thread to execute.

11 | CONCLUSION

We present a data processing pipeline for ego-motion estimation and mapping. The pipeline couples a 3D laser scanner, a camera, and an IMU, running three modules sequentially to produce real-time ego-motion estimation. The coarse-to-fine data processing generates high-rate estimation and registers low-drift maps over a long distance of travel. Further, the method is robust to individual sensor failures. Due to degraded environments or aggressive motion, if the camera or the laser scanner is not fully functional, the corresponding module is bypassed and the rest of the pipeline is staggered to warrant the overall functionality. We validate the method through a large number of experiments. In particular, we conduct tests to evaluate the accuracy and robustness over several kilometers of travel, in complex road conditions, with dramatic lighting changes and structural degradation, and with high-rate motion in rotation and translation. Results indicate that the method can conquer all challenging scenarios, producing position drift around 0.2% of the distance traveled and carrying out robustness w.r.t running, jumping motion, and even driving at highway speed.

ACKNOWLEDGMENT

Special thanks are given to K. D., W. S., M. B., D. D., S. H., E. A., B. B., V. G., and L. N. for their insightful inputs and invaluable help.

ORCID

Ji Zhang http://orcid.org/0000-0002-3122-3106

REFERENCES

Andersen, R. (2008). *Modern methods for robust regression*. Los Angeles: Sage.

de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). *Computation Geometry: Algorithms and Applications* (3rd ed). Springer-Verlag.

Bosse, M., Zlot, R., & Flick, P. (2012). Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, *28*(5), 1104–1119.

Ceriani, S., Sanchez, C., Taddei, P., Wolfart, E., & Sequeira, V. (2015). Pose interpolation slam for large maps using moving 3d sensors. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany.

Corke, P., Strelow, D., & Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan.

Dellaert, F., & Kaess, M. (2006). Square RootSAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, *25*(12), 1181–1204.

Droeschel, D., Stuckler, J., & Behnke, S. (2014). Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner. *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China.

Engel, J., Koltun, V., & Cremers, D. (2016). Direct sparse odometry. Retrieved from: http://arXiv.org/abs/arXiv:1607.02565.

Engelhard, N., Endres, F., Hess, J., Sturm, J., & Burgard, W. (2011). Real-time 3D visual SLAM with a hand-held RGB-D camera. *RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden.

Forster, C., Carlone, L., Dellaert, F., & Scaramuzza, D. (2015). Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. *Robotics: Science and Systems (RSS)*, Rome, Italy.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China.

FrançoisPomerleau, F. C., & Siegwart, R. (2015). A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, *4*(1), 1–104.

Furgale, P., Barfoot, T. D., & Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, (32), 1229–1235.

Geiger, A., Ziegler, J., & Stiller, C. (2011). Stereoscan: Dense 3D reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany.

Guo, C. X., Kottas, D. G., DuToit, R. C., Ahmed, A., Li, R., & Roumeliotis, S. I. (2014). Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps. In *Proceedings of Robotics: Science and Systems*, Berkeley, CA.

Hartley, R., & Zisserman, A. (2000). *Multiple View Geometry in computer vision*. Cambridge Press.

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2012). RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, *31* (5), 647–663.

Holz, D. & Behnke, S. (2014). Mapping with micro aerial vehicles by registration of sparse 3d laser scans. In *the 13th International Conference on Intelligent Autonomous Systems (IAS)*, Padova, Italy.

Horaud, R., & Dornaika, F. (1995). Hand-eye calibration. *The international journal of robotics research*, *14*(3), 195–210.

Huang, A., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., & Roy, N. (2011). Visual odometry and mapping for autonomous flight using an RGB-D camera. *International Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona.

Huang, G., Kaess, M., & Leonard, J. (2014). Towards consistent visual-inertial navigation. *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong.

Kaess, M., Ranganathan, A., & Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, *24*(6), 1365–1378.

Kerl, C., Sturm, J., & Cremers, D. (2013). Robust odometry estimation for RGB-D cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany.

Klein, G. and Murray, D. (2007). Parallel tracking amd mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, Nara, Japan.

Konolige, K., Agrawal, M., & Sol, J. (2011). Large-scale visual odometry for rough terrain. *Robotics Research*, *66*, 201–212.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., & Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, *34*(3), 314–334.

Li, M., & Mourikis, A. (2014). Vision-aided inertial navigation with rolling-shutter cameras. *International Journal of Robotics Research*, *33*(11), 1490–1507.

Li, M., & Mourikis, A. I. (2013). High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, *32*(6), 690–711.

Lu, V. & Hart, J. (2014). Multicore construction of k-d trees for high dimensional point data. In *International Conference on Advances in Big Data Analytics (ABDA)*, Las Vegas, NV.

Lucas, B. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, Vancouver, Canada.

Maimone, M., Cheng, Y., & Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, *24* (2), 169–186.

Murray, R., & Sastry, S. (1994). *A mathematical introduction to robotic manipulation*. CRC Press.

Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain.

Nister, D., Naroditsky, O., & Bergen, J. (2006). Visual odometry for ground vechicle applications. *Journal of Field Robotics*, *23*(1), 3–20.

Nocedal, J., & Wright, S. (2006). *Numerical Optimization*. New York: Springer-Verlag.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., … Ng, A. (2009). ROS: An open-source robot operating system. Workshop on Open Source Software (Collocated with ICRA 2009), Kobe, Japan.

Scherer, S., Rehder, J., Achar, S., Cover, H., Chambers, A., Nuske, S., & Singh, S. (2012). River mapping from a flying robot: state estimation, river detection, and obstacle mapping. *Autonomous Robots*, *32*(5), 1–26.

Tong, C. H., Anderson, S., Dong, H., & Barfoot, T. (2014). Pose interpolation for laser-based visual odometry. *Journal of Field Robotics*, *31*(5), 731–757.

Unnikrishnan, R. & Hebert, M. (2005). Fast extrinsic calibration of a laser rangefinder to a camera. Technical report, Technical Report #CMU-RI-TR-05-09, Robotics Institute, Carnegie Mellon University.

Velas, M., Spanel, M., & Herout, A. (2016). Collar line segments for fast odometry estimation from velodyne point clouds. In *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden.

Vogiatzis, G., & Hernandez, C. (2011). Video-based, real-time multi-view stereo. *Image and Vision Computing*, *29*(7), 434–441.

Wei, C., Wu, T., & Fu, H. (2015). Plain-to-plain scan registration based on geometric distributions of points. In *IEEE International Conference on Information and Automation (ICIA)*, Lijiang, China.

Weiss, S., Achtelik, M., Lynen, S., Achtelik, M., Kneip, L., Chli, M., & Siegwart, R. (2013). Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, *30*(5), 803–831.

Whelan, T., Johannsson, H., Kaess, M., Leonard, J., & McDonald, J. (2013). Robust real-time visual odometry for dense RGB-D mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany.

Zhang, J., Kaess, M., & Singh, S. (2014). Real-time depth enhanced monocular odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL.

Zhang, J., Kaess, M., & Singh, S. (2016). On degeneracy of optimization-based state estimation problems. In *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden.

Zhang, J. & Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. *Robotics: Science and Systems Conference (RSS)*, Berkeley, CA.

Zhang, J. & Singh, S. (2015). Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA.

Zhang, J. & Singh, S. (2017a). Aerial and ground-based collaborative mapping: An experimental study. In *the 11th International Conference on Field and Service Robotics (FSR)*, Zurich, Switzerland.

Zhang, J. & Singh, S. (2017b). Enabling aggressive motion estimation at low-drift and accurate mapping in real-time. In *IEEE International Conference on Robotics and Automation (ICRA)*, Singapore.

Zhang, Z. (2000). A flexible new technique for camera calibration. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1330–1334.

Zlot, R., Bosse, M., Greenop, K., Jarzab, Z., Juckes, E., & Roberts, J. (2014). Efficiently capturing large, complex cultural heritage sites with a handheld mobile 3D laser mapping system. *Journal of Cultural Heritage*, *15*(6), 670–678.

How to cite this article: Zhang J, Singh S. Laser–visual–inertial odometry and mapping with high robustness and low drift. *J Field Robotics*. 2018;1–23. https://doi.org/10.1002/rob.21809