2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

# Enabling Aggressive Motion Estimation at Low-drift and Accurate Mapping in Real-time

Ji Zhang and Sanjiv Singh

***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, a camera, and an IMU. Different from traditional methods that use a Kalman filter or factor-graph optimization, the proposed method employs a sequential, multi-layer processing pipeline, solving for motion from coarse to fine. The resulting system enables high-frequency, low-latency ego-motion estimation, along with dense, accurate 3D map registration. Further, the system 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 dark, texture-less, and structure-less environments. During experiments, the system demonstrates 0.22% of relative position drift over 9.3km of navigation and robustness w.r.t aggressive motion such as highway speed driving (up to 33m/s).**

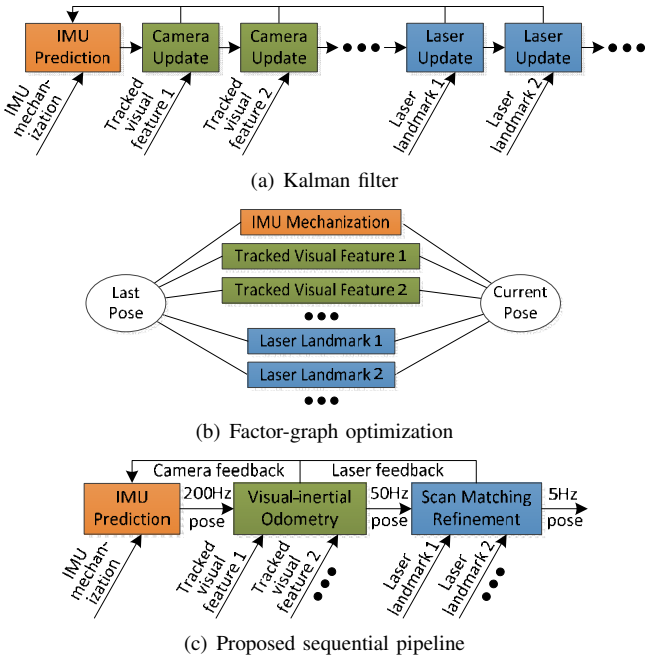
## I. Introduction

We aim at developing a software system for ego-motion estimation and mapping. Specially, we are interested in solving for highly aggressive motion in 6-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. The proposed method enables such high-rate ego-motion estimation, while at the same time develops 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 Fig. 1(a), a Kalman filter based method typically processes individual visual features and laser landmarks in separate steps, while a factor-graph optimization based method combines all sensor data into a full-blown optimization problem (see Fig. 1(b)). In comparison, our system recovers motion through multi-layer processing in a coarse-to-fine manner (see Fig. 1(c)). Starting with motion prediction from an IMU, a visual-inertial coupled method estimates motion and registers laser points locally. Then, a scan matching method further refines the estimated motion and registers point clouds.

The system 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.

J. Zhang and S. Singh are with Kaarta, Inc. Emails: ji@kaarta.com and sanjiv@kaarta.com.




(a) Kalman filter

(b) Factor-graph optimization

(c) Proposed sequential pipeline

Fig. 1. Diagram of the odometry and mapping software system. (a) 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) shows a factor-graph optimization setup. All constraints from the IMU, visual features, and laser landmarks are combined in an optimization problem. (c) 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 and develops a map. From left to right, motion is recovered from coarse to fine and accuracy is improved step by step.

Further, the system is carefully configured to handle sensor degradation. If the camera is non-functional, e.g. due to darkness, dramatic lighting changes, or texture-less environments, or if the laser is non-functional, e.g. due to structure-less environments, the corresponding module is fully or partially bypassed and the rest system is staggered to function reliably. Finally, the system integrates pose outputs from all three modules to realize high-frequency, low-latency motion estimation.

## II. Related Work

This paper is related to vision and laser based state estimation. For vision methods, in addition to stereo cameras [1], [2] and monocular cameras [3], [4] as common choices, RGB-D cameras have gained popularity in recent years. Methods [5]–[7] have shown promising results. Our system is relevant to RGB-D methods since it is assisted by laser

ranging and associates the laser range information to visual features in motion estimation. Alternatively, one can also combine cameras with an IMU. For example, Huang et al. [8] and Li and Mourikis [9] tightly couple a monocular camera and an IMU in a Kalman filter, while other methods [10], [11] use optimization to solve for the motion. In this paper, we prefer optimization-based methods over filter-based methods.

For laser methods, it has been shown that motion can be recovered with a laser itself [12], [13], or optionally assisted by an IMU [14]. Alternatively, one can involve other sensors to provide motion estimates. For example, Droeschel et al's method [15] and Holz and Behnke's method [16] use visual odometry output as motion approximation and further match laser scans to refine the motion. The proposed system is inspired by the same concept, but is more complete with range, vision, and inertial sensors all coupled. The difference is that methods [15], [16] consider the camera and the laser as independent modules, while in our system the modules are highly interactive and dynamically reconfigurable. This allows different combinations in the system adapted to specific environments and motion robust to sensor degradation.

The paper is based on our previous visual odometry [17] and laser odometry [18] methods. The visual odometry is now key-framed and coupled with an IMU. The laser odometry is reimplemented with multi-thread processing. The combined system reaches the level of accuracy that are unachieved in our previous work [17]–[19]. Further, the system 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.

## III. IMU Prediction Subsystem

This subsection describes the IMU prediction subsystem. Let $\boldsymbol{\omega}(t)$ and $\boldsymbol{a}(t)$ be two $3\times1$ vectors indicating the angular rates and accelerations of the IMU frame $\{I\}$. Let $\boldsymbol{b_\omega}(t)$ and $\boldsymbol{b_a}(t)$ be the corresponding biases, and $\boldsymbol{n_\omega}(t)$ and $\boldsymbol{n_a}(t)$ be the noises. Additionally, let $\boldsymbol{g}$ be the constant gravity vector in the world frame $\{W\}$. The IMU measurement terms are,

$$\hat{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}(t) + \boldsymbol{b_\omega}(t) + \boldsymbol{n_\omega}(t), \tag{1}$$

$$\hat{\boldsymbol{a}}(t) = \boldsymbol{a}(t) - {}^I_W\mathbf{R}(t)\boldsymbol{g} + \boldsymbol{b_a}(t) + \boldsymbol{n_a}(t), \tag{2}$$

where ${}^I_W\mathbf{R}(t)$ is the rotation matrix from $\{W\}$ to $\{I\}$. The IMU biases are slowly changing variables. We take the most recently updated biases in (1)-(2) to predict the motion.

The IMU bias correction can be made by feedback from either the camera or the laser. By comparing the estimated motion with IMU integration, we can calculate $\boldsymbol{b_\omega}(t)$ and $\boldsymbol{b_a}(t)$. To reduce high-frequency noises, a sliding window is employed keeping a certain number of biases. The averaged terms are used. Although a rigorous way is to model the biases as random walks and update the biases through optimization [10], [11], we prefer to keep IMU processing in a separate module. This favors dynamic reconfiguration of the system, i.e. the IMU can be coupled with either the camera or the laser. If the camera is non-functional, the IMU biases are corrected by the laser instead (more discussion in Section VI).

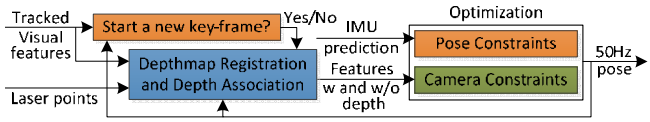



Fig. 2. Diagram of the visual-inertial odometry subsystem.

## IV. Visual-Inertial Odometry Subsystem

The section summarizes the visual-inertial odometry subsystem. A system diagram is shown in Fig. 2. 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.

The method is key-frame based. A new key-frame is determined if a certain number of features lose tracking or the image overlap is below a certain ratio. 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. For a feature that is associated with depth at key-frame $l$, we denote it as $\boldsymbol{X}_l = [x_l,\ y_l,\ z_l]^T$. A feature without depth is denoted as $\bar{\boldsymbol{X}}_l = [\bar{x}_l,\ \bar{y}_l,\ 1]^T$ using normalized coordinates. A feature at frame $c$ is denoted as $\boldsymbol{X}_c = [x_c,\ y_c,\ z_c]^T$ or $\bar{\boldsymbol{X}}_c = [\bar{x}_c,\ \bar{y}_c,\ 1]^T$.

Let $\mathbf{R}^c_l$ and $\boldsymbol{t}^c_l$ be the $3\times3$ rotation matrix and $3\times1$ translation vector between frames $l$ and $c$, where $\mathbf{R}^c_l \in \mathbf{SO}(3)$ and $\boldsymbol{t}^c_l \in \mathbb{R}^3$. The motion function is,

$$\boldsymbol{X}_c = \mathbf{R}^c_l\boldsymbol{X}_l + \boldsymbol{t}^c_l. \tag{3}$$

Let $d_c$ be the depth of $\boldsymbol{X}_c$, where $\boldsymbol{X}_c = d_c\bar{\boldsymbol{X}}_c$. Substituting $\boldsymbol{X}_c$ with $d_c\bar{\boldsymbol{X}}_c$ and combining the 1st and 2nd rows with the 3rd row in (3), respectively, to eliminate $d_c$, we obtain constraints,

$$(\boldsymbol{R}(1) - \bar{x}_c\boldsymbol{R}(3))\boldsymbol{X}_l + t(1) - \bar{x}_c t(3) = 0, \tag{4}$$

$$(\boldsymbol{R}(2) - \bar{y}_c\boldsymbol{R}(3))\boldsymbol{X}_l + t(2) - \bar{y}_c t(3) = 0. \tag{5}$$

Here, $\boldsymbol{R}(h)$ and $t(h)$, $h \in \{1,2,3\}$, are the $h$-th rows of $\mathbf{R}^c_l$ and $\boldsymbol{t}^c_l$. 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 $d_l\bar{\boldsymbol{X}}_l$ and $d_c\bar{\boldsymbol{X}}_c$, respectively, and combining all three rows in (3) to eliminate $d_l$ and $d_c$, we obtain another constraint,

$$[\bar{y}_c t(3) - t(2),\ \ -\bar{x}_c t(3) + t(1),\ \bar{x}_c t(2) - \bar{y}_c t(1)]\mathbf{R}^c_l\bar{\boldsymbol{X}}_l = 0. \tag{6}$$

The motion estimation is to solve an optimization problem combining three sets of constraints: 1) from features with known depth as (4)-(5); 2) from features with unknown depth as (6); and 3) from the IMU prediction. Let us define $\mathbf{T}^b_a$ as a $4\times4$ matrix representing the motion transform between frames $a$ and $b$, $\mathbf{T}^b_a$ corresponds to a set of $\mathbf{R}^b_a$ and $\boldsymbol{t}^b_a$.

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

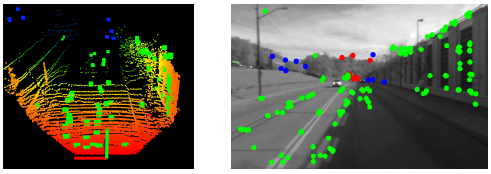
Let $\hat{\boldsymbol{\theta}}_l^c \in \mathfrak{so}(3)$ and $\hat{\boldsymbol{t}}_l^c(\boldsymbol{\theta}_l^c) \in \mathbb{R}^3$ be the 6-DOF motion corresponding to $\hat{\mathbf{T}}_l^c$. Here, the translation from the IMU prediction, $\hat{\boldsymbol{t}}_l^c(\boldsymbol{\theta}_l^c)$, is dependent on the orientation, i.e. the orientation determines projection of the gravity vector through rotation matrix ${}_W^I\mathbf{R}(t)$ in (2), and hence the accelerations being integrated. When calculating $\hat{\boldsymbol{t}}_l^c(\boldsymbol{\theta}_l^c)$, we start at frame $c$ and integrate accelerations inversely w.r.t. time. Let $\boldsymbol{\theta}_l^c \in \mathfrak{so}(3)$ be the rotation vector corresponding to $\mathbf{R}_l^c$ in (3), $\boldsymbol{\theta}_l^c$ and $\boldsymbol{t}_l^c$ are the motion to be solved. The pose constraint is,

$$\Sigma_l^c[(\hat{\boldsymbol{\theta}}_l^c - \boldsymbol{\theta}_l^c)^T,\ (\hat{\boldsymbol{t}}_l^c(\boldsymbol{\theta}_l^c) - \boldsymbol{t}_l^c)^T]^T = \boldsymbol{0}, \tag{7}$$

where $\Sigma_l^c$ is a relative covariance matrix scaling the pose constraint appropriately w.r.t. the camera constraints.

The optimization problem is solved by the Newton gradient-descent method [20] adapted to a robust fitting framework [21] for outlier feature removal. In this problem, the state space contains $\boldsymbol{\theta}_l^c$ and $\boldsymbol{t}_l^c$. In other words, we only solve a marginalized problem where landmark positions are not optimized. This means only six unknowns keeping computation intensity low. The argument is that the method involves laser range measurements to provide precise depth information. Further optimizing the features' depth is practically unnecessary.

The method registers laser points on a depthmap and then associates depth to features. Laser points within the camera FOV are kept. The depthmap is stored in a 2D KD-tree [22] for fast index. In the KD-tree, all laser points are projected onto a unit sphere around the camera center. When associating depth information, we project the features onto the sphere and find the three closest laser points for each feature. The depth is interpolated from the three points assuming a local planar patch in Cartesian space. For 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 [4], [23], where the depth is updated at each frame based on a Bayesian probabilistic mode. Fig. 3 shows an example depthmap and 3D projected features.



(a) (b)

Fig. 3. (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).

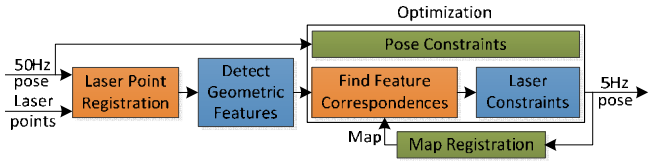



Fig. 4. Diagram of the scan matching subsystem.

## V. Scan Matching Subsystem

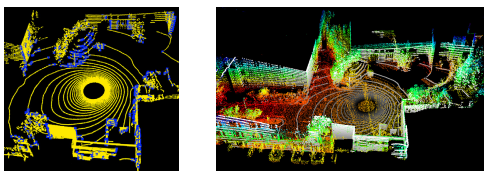
This subsystem further refines motion estimates from the previous module by laser scan matching. A diagram is present in Fig. 4. Upon receiving of laser scans, the method first registers points from a scan into a local point cloud. To do this, we take the odometry estimation from the visual-inertial odometry as key-points, and use IMU measurements to interpolate in between the key-points. Let us use $m \in Z^+$ to indicate the scan number. Let $\mathcal{P}_m$ be the locally registered point cloud from scan $m$. We extract two sets of geometric features from $\mathcal{P}_m$, one with edge points, denoted as $\mathcal{E}_m$, and the other with planar points, denoted as $\mathcal{H}_m$. This is through computation of curvature in $\mathcal{P}_m$. Fig. 5(a) gives an example of detected edge points (blue) and planar points (yellow).

The geometric features are then matched to the map. Let $\mathcal{Q}_{m-1}$ be the map point cloud after processing the last scan. The points in $\mathcal{Q}_{m-1}$ are separated into two sets containing edge points and planar points as well. We use voxels to store the map. For each voxel, we construct two 3D KD-trees [22], each with a set of points. Using KD-trees for individual voxels dramatically accelerates point searching since given a query point, we only need to search in a specific KD-tree from a voxel. When matching scans, we find a cluster of closest points for each point in $\mathcal{E}_m$ and $\mathcal{H}_m$. 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 regarding the distance from a point to the corresponding point cluster,

$$d = f(\boldsymbol{X}_m, \boldsymbol{\theta}_m, \boldsymbol{t}_m), \tag{8}$$

where $\boldsymbol{X}_m$ is a point in $\mathcal{E}_m$ or $\mathcal{H}_m$, and $\boldsymbol{\theta}_m \in \mathfrak{so}(3)$ and $\boldsymbol{t}_m \in \mathbb{R}^3$ indicate the 6-DOF pose of $\mathcal{P}_m$ in the world frame $\{W\}$, w.r.t. $\mathcal{Q}_{m-1}$. Fig. 5(b) shows an example where a scan (gray points) is matched to the map (colored points).

The scan matching is formulated into an optimization problem minimizing the overall distances as (8). The op-



(a) (b)

Fig. 5. (a) Example edge points (blue) and planar points (yellow) detected from a scan. (b) Matching a scan (grey points) to the map (colored points), then the scan is merged with the map to extend the map further.

timization also involves pose constraints. Let $\mathbf{T}_{m-1}$ be the $4 \times 4$ transformation matrix regarding the pose of $\mathcal{P}_{m-1}$ in $\{W\}$, $\mathbf{T}_{m-1}$ is generated by processing the last scan. Let $\hat{\mathbf{T}}^{m}_{m-1}$ be the pose transform between $\mathcal{P}_{m-1}$ and $\mathcal{P}_m$, as provided by the odometry estimation. The predicted pose transform of $\mathcal{P}_m$ is obtained as $\hat{\mathbf{T}}_m = \hat{\mathbf{T}}^{m}_{m-1}\mathbf{T}_{m-1}$. Let $\hat{\boldsymbol{\theta}}_m \in \mathfrak{so}(3)$ and $\hat{\boldsymbol{t}}_m \in \mathbb{R}^3$ be the 6-DOF pose corresponding to $\hat{\mathbf{T}}_m$, and let $\Sigma_m$ be a relative covariance matrix. The constraint is,

$$\Sigma_m[(\hat{\boldsymbol{\theta}}_m - \boldsymbol{\theta}_m)^T,\ (\hat{\boldsymbol{t}}_m - \boldsymbol{t}_m)^T]^T = \boldsymbol{0}. \tag{9}$$

The optimization problem refines $\boldsymbol{\theta}_m$ and $\boldsymbol{t}_m$, which is solved by the Newton gradient-descent method [20] adapted to a robust fitting framework [21]. After a scan is matched, the scan is merged with the map to extend the map further.

The scan matching involves building KD-trees and repetitively finding feature correspondences. The process is time-consuming. We conduct a multi-thread implementation to guarantee the desired frequency. Fig. 6(a) illustrates the case where two matcher programs run 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. The two matchers are called alternatively. On each matcher, $\mathcal{P}_m$, $\mathcal{P}_{m-1}$, ..., are matched with $\mathcal{Q}_{m-2}$, $\mathcal{Q}_{m-3}$, ..., 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 (Fig. 6(b)), and $\mathcal{P}_m$, $\mathcal{P}_{m-1}$, ..., are matched with $\mathcal{Q}_{m-1}$, $\mathcal{Q}_{m-2}$, .... This implementation uses maximally four threads, but we have rarely seen three threads are needed.

## VI. On Robustness

The robustness of the system is determined by its ability to handle sensor degradation. We assume the IMU is always reliable functioning as the backbone in the system. Camera is sensitive to dramatic lighting changes. It also fails in a dark/texture-less environment or when significant motion blur is present causing visual features lose tracking. Laser cannot handle structure-less environments, e.g. 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 [24]. Both the visual-inertial odometry and the scan matching modules formulate and solve optimization problems. When a failure happens, it corresponds to a degraded optimization problem, i.e. some directions of the state space are loosely constrained and noises are dominate. Let $\mathbf{J}$ be the Jacobian matrix associated with the problem, 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, 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,

$$\mathbf{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_6]^T,\ \bar{\mathbf{V}} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_h, 0, ..., 0]^T. \tag{10}$$

When solving an optimization problem, the nonlinear iteration starts with an initial guess. With the sequential pipeline in Fig. 1(c), 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, $\Delta\boldsymbol{x}$ is calculated by solving the linearized system equations. 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,

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \mathbf{V}^{-1}\bar{\mathbf{V}}\Delta\boldsymbol{x}. \tag{11}$$

Let us further explain the intuition behind (11). The system solves for motion in a coarse-to-fine order, starting with the IMU prediction, the following two modules further solve/refine the motion as much as possible, fully (in 6-DOF) if the problem is well-conditioned, and partially (in 0 to 5-DOF) otherwise. If the problem is completely degraded, $\bar{\mathbf{V}}$ is a zero matrix and the previous module's output is kept.

### A. Case Study of Camera Degradation

As shown in Fig. 7(a), if visual features are insufficiently available for the visual-inertial odometry, the IMU prediction fully or partially bypasses the green block to locally register laser points. The laser feedback compensates for the camera feedback to correct 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. If sufficient visual features are found, the laser feedback is not used.

### B. Case Study of Laser Degradation

As shown in Fig. 7(b), 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.

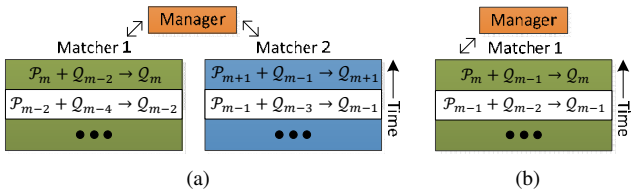



Fig. 6. Illustration of multi-thread scan matching. A manager program calls multiple matcher programs running on separate CPU threads and matches scans to the latest map available. (a) shows a two-thread case. Scans $\mathcal{P}_m$, $\mathcal{P}_{m-1}$, ..., are matched with map $\mathcal{Q}_{m-2}$, $\mathcal{Q}_{m-3}$, ..., on each matcher, giving twice amount of time for processing. In comparison, (b) shows a one-thread case, where $\mathcal{P}_m$, $\mathcal{P}_{m-1}$, ..., are matched with $\mathcal{Q}_{m-1}$, $\mathcal{Q}_{m-2}$, .... The implementation is dynamically configurable using up to four threads.

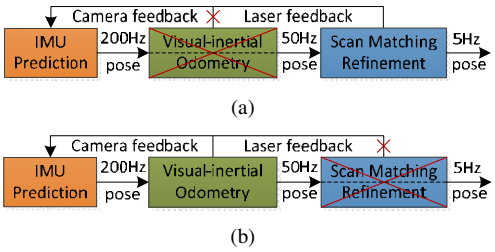



Fig. 7. Case study of camera and laser degradation. (a) If visual features are insufficient for the visual-inertial odometry, the IMU prediction (partially) bypasses the green block to register laser points locally. Correction of velocity drift and biases of the IMU is made with the laser feedback. (b) 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".

### C. Case Study of Camera and Laser Degradation

Finally, let us discuss a complex scenario where both the camera and the laser are degraded. We use the example in Fig. 8 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 (10). 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 VI-A). During the combination, the blue rows are only filled in where the green rows are not present.

## VII. Experiments

Our software system is validated on two sensor suites. In Fig. 9(a), a Velodyne HDL-32E laser scanner is attached to a UI-1220SE monochrome camera and an Xsens MTi-30 IMU. The laser scanner receives 0.7 million points/second at 5Hz. The camera is configured at the resolution of $752 \times 480$ pixels, $76^\circ$ horizontal FOV, and 50Hz frame rate. The IMU frequency is 200Hz. In Fig. 9(b), a Velodyne VLP-16 laser

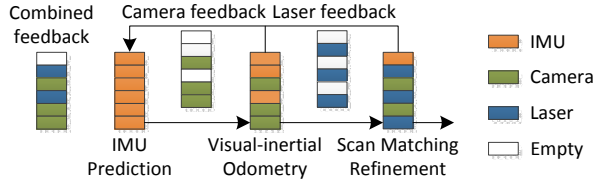



Fig. 8. An example where both the camera and the laser are degraded. A vertical bar represents a 6-DOF pose and each row is a DOF. Starting with the IMU prediction on the left where all six rows are orange, the visual-inertial odometry updates in 3-DOF where the rows become green, then the scan matching 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 the camera feedback is not present.

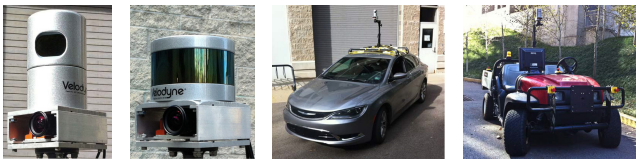



Fig. 9. Sensor suites and vehicles used in experiments. (a) is a Velodyne HDL-32E laser scanner attached with a uEye UI-1220SE monochrome camera and an Xsens MTi-30 IMU. (b) is a Velodyne VLP-16 laser scanner attached with the same camera and IMU. (c) is a passenger vehicle for street driving. (d) is 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.

scanner is attached to the same camera and IMU, receiving 0.3 million points/second at 5Hz. Both sensor suites are attached to the vehicles in Fig. 9(c) and Fig. 9(d).

The software runs on a laptop computer with a 2.6GHz i7 quad-core processor and an integrated GPU, in a Linux system running ROS [25]. Feature tracking runs on the GPU. The rest system consumes 1.5-2.5 threads depending on the amount of laser data. With the sensor suite in Fig. 9(a), the visual-inertial odometry takes about 0.5 thread and the scan matching uses 2 threads, resulting in 2.5 threads being used. However, with the sensor suite in Fig. 9(b), the scan matching takes 1 thread, resulting in 1.5 threads being used in total.

For both sensor suites, we track maximally 300 Harris corners using the Kanade Lucas Tomasi (KLT) method [26]. To evenly distribute the visual features, an image is separated into $5 \times 6$ identical subregions, 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.

### A. Accuracy Tests

We start with evaluating accuracy of the proposed system. The sensor suite in Fig. 9(a) is mounted on the vehicle in Fig. 9(c), driven on structured roads. As shown in Fig. 11, the path goes through vegetated environments, bridges, hilly terrains, and streets with heavy traffic, and finally returns to the starting position. The overall path is 9.3km in length, and the elevation changes over 70m along the path. Except waiting for traffic lights, the vehicle speed is between 9-18m/s (32-65km/h or 20-40 miles/hour) during the test. On the left side of Fig. 11, 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 images logged by the camera.

Additionally, we examine how each module in the system contributes to the overall accuracy. As shown in Fig. 12, we first plot output of the visual-inertial odometry as the green dash-dot curve. This uses the left two modules in

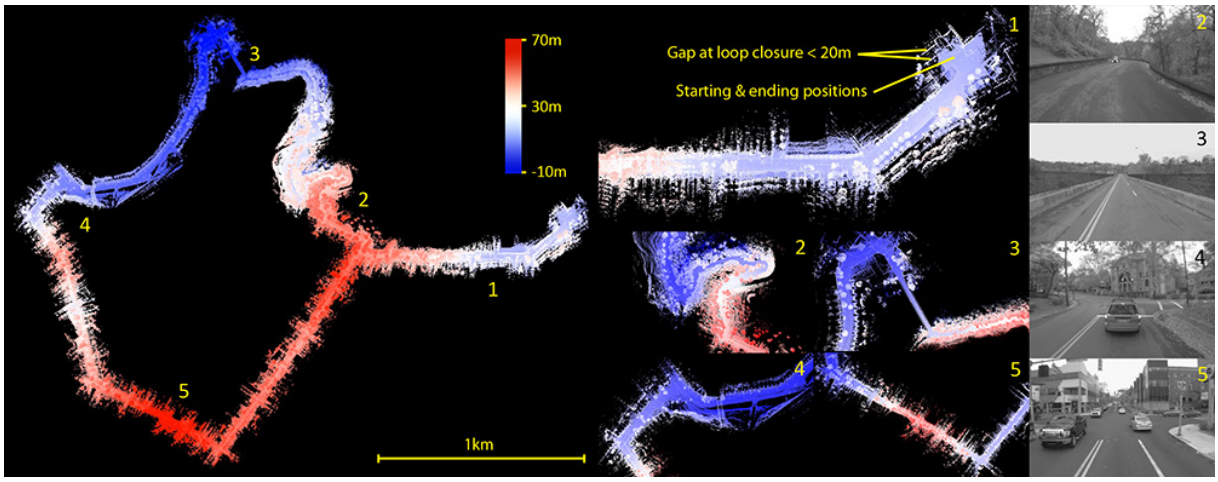



Fig. 11. Accuracy test. The sensor suite in Fig. 9(a) is mounted on the vehicle in Fig. 9(c) for 9.3km of street driving. The path goes through vegetated environments, bridges, hilly terrains, and roads with heavy traffic. The elevation changes over 70m. Except waiting for traffic lights, the vehicle is driven at 9-18m/s. On the left, we show the complete map color coded by elevation. On the right, we show a few close views with corresponding 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 images logged by the camera.

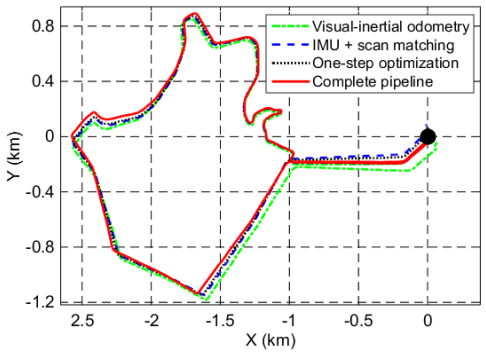



Fig. 12. Estimated trajectories in accuracy test. The trajectories start with the black dot. We compare four system configurations in the test. The green dash-dot curve is from the visual-inertial odometry module (using the left two modules in Fig. 1(c)). The blue dash curve is from the scan matching module with the IMU prediction directly taken as input (leftmost and rightmost modules in Fig. 1(c)). The black dot curve has the system reconfigured to solve one large optimization problem incorporating all constraints, as in Fig. 1(b). The red solid curve is from the proposed data processing pipeline.

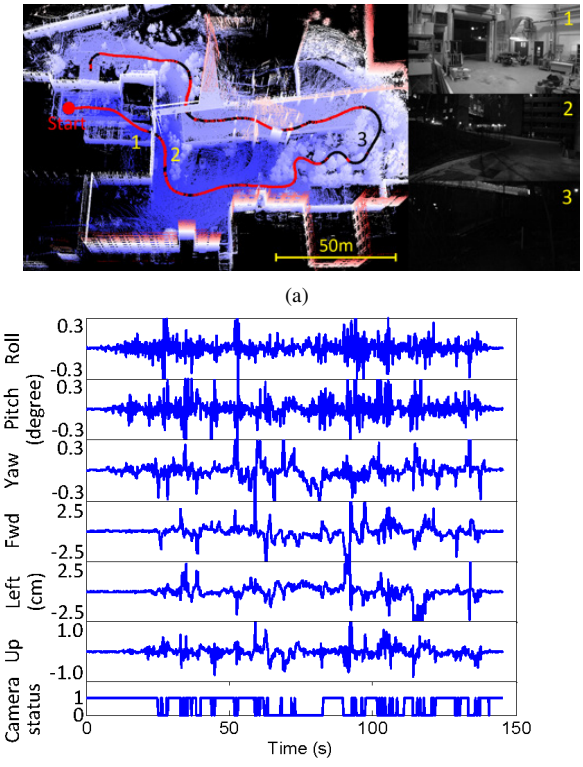
Fig. 1(c). Next, we directly forward the IMU prediction to the scan matching module, bypassing the visual-inertial odometry. This configuration uses the leftmost and the rightmost modules in Fig. 1(c). The result is drawn as the blue dash curve. Finally, we plot output of the complete pipeline as the red solid curve with the least 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 system performance when encountering individual sensor degradation: if scan matching is degraded, the system reduces to a mode indicated by the green dash-dot curve; if vision is degraded, the system reduces to that indicated by the blue dash curve. Further, we reconfigure the system to incorporate all constraints in one large optimization problem as in Fig. 1(b). The system takes the IMU prediction as the initial guess and runs at the laser frequency (5Hz). The system 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, 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.

### *B. Robustness Tests*

We further inspect the system robustness w.r.t. sensor failures. These experiments use the sensor suite in Fig. 9(b) attached to the vehicle in Fig. 9(d). First, we drive the vehicle at night where vision degrades. When 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 Fig. 13(a), the red and the black segments on the trajectory respectively indicate vision is functional and degraded. In Fig. 13(b), 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 corrections on the top six rows become larger because the IMU prediction is less precise in comparison to the visual-inertial odometry.

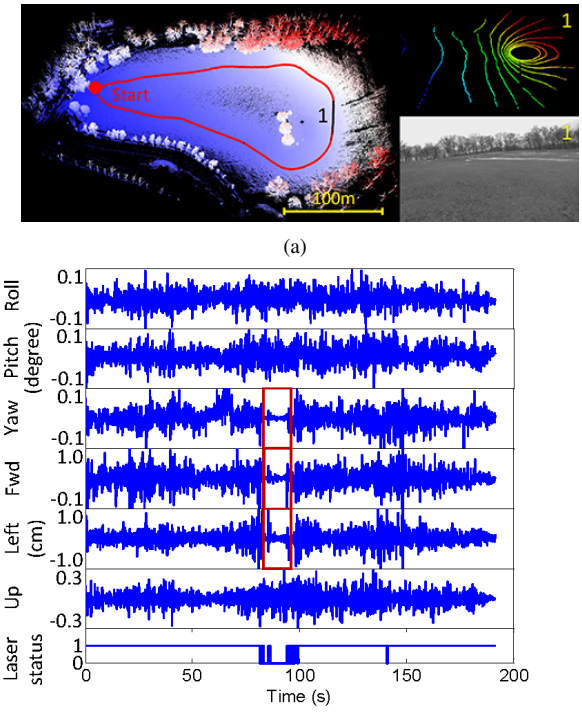
Next, we bring the vehicle to an open area where scan matching degrades due to the planar environment. As shown in Fig. 14(a), when the vehicle researches the rightmost side of the path (black segment), only the flat ground is seen by the laser. The system determines the scan matching is able to refine 3-DOF out of the 6-DOF motion using the method introduced in Section VII. Specifically, roll, pitch, and elevation are well-conditioned, but yaw, forward, left are unsolvable due to the planar scene. The visual-inertial odometry output is used directly in the degraded directions. In Fig. 14(b), we show pose corrections applied by the scan matching. On the bottom row, the laser status being zero indicates partial degradation. Correspondingly, corrections in




(a)

(b)

Fig. 13. Robustness test 1. The sensor suite in Fig. 9(b) is attached to the vehicle in Fig. 9(d) 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.




(a)

(b)

Fig. 14. Robustness test 2. The sensor suite in Fig. 9(b) is attached to the vehicle in Fig. 9(d) 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.

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 Fig. 1(c)) as determined by the method in Section VII.

### C. Aggressive Motion Tests

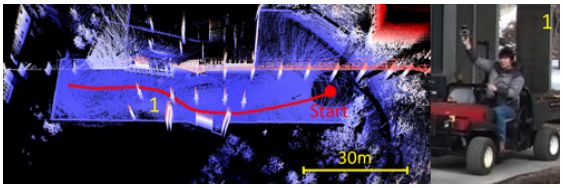
In this section, we evaluate the system performance w.r.t. high-speed rotation and translation. These tests use the sensor suite in Fig. 9(b). First, the sensors are held by a person who drives the vehicle in Fig. 9(d). The vehicle carries power supply and a data processing computer. The person oscillates the sensor suite to introduce fast rotation. In Fig. 15(a), the estimated trajectory is overlayed on the map built, with photos showing the experiment setup. In Fig. 15(b), the estimated orientation is present. During the test, the maximum angular speed exceeds $190°/s$.

Then, we mount the sensors on the vehicle in Fig. 9(c) and drive along a straight path at a high speed. As shown in Fig. 17, the overall path is 701m 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 Fig. 17, and a corresponding image taken from location 1 on the satellite image. The houses are on the left side of the image. In Fig. 17, we plot the linear speed. The maximum speed reaches as high as 33m/s.

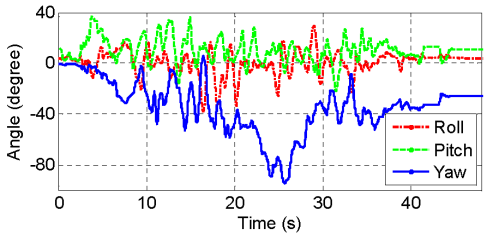
## VIII. Conclusion

We present a data processing pipeline for ego-motion estimation and mapping. The pipeline couples a 3D laser, a camera, and an IMU, running three modules sequentially to produce real-time ego-motion estimation and low-drift map registration. Further, the system is robust to individual sensor failures. Due to degraded environments or aggressive motion, if the camera or the laser is not fully functional, the corresponding module is bypassed and the rest system is staggered to warrant the overall functionality. We validate the system 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


Fig. 17. Aggressive motion test 2. The sensor suite in Fig. 9(b) is mounted to the vehicle in Fig. 9(c), driven at a high speed along the red path. The overall path is 701m and the maximum linear speed is 33m/s. 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.




(a)




(b)

Fig. 15. Aggressive motion test 1. The sensor suite in Fig. 9(b) is held by a person in one hand who drives the vehicle in Fig. 9(d) with the other hand. The person oscillates the sensor suite to introduce fast rotation. (a) shows the estimated trajectory overlayed on the map built and a photo taken from location 1 during the test. (b) shows estimated orientation of the sensor suite. From our analysis, the maximum angular speed exceeds $160^\circ/s$.

with high-rate motion in rotation and translation. Results indicate that the system can conquer all challenging scenarios, producing position drift around $0.2\%$ of the distance traveled and carrying out robustness w.r.t aggressive motion such as highway speed driving.

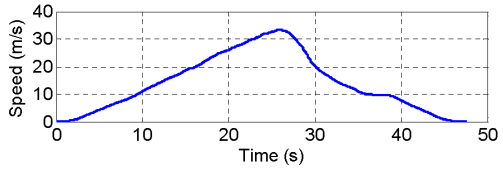



Fig. 17. Estimated linear speed in aggressive motion test 2. The highest speed reaches 33m/s (119km/h or 74 miles/hour) during the test.

## REFERENCES


[1] K. Konolige, M. Agrawal, and J. Sol, "Large-scale visual odometry for rough terrain," *Robotics Research*, vol. 66, pp. 201–212, 2011.

[2] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3D reconstruction in real-time," in *IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany, June 2011.

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

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

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

[6] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[7] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[8] G. Huang, M. Kaess, and J. Leonard, "Towards consistent visual-inertial navigation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Hong Kong, June 2014.

[9] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual–inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[10] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

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

[12] C. H. Tong, S. Anderson, H. Dong, and T. Barfoot, "Pose interpolation for laser-based visual odometry," *Journal of Field Robotics*, vol. 31, no. 5, pp. 731–757, 2014.

[13] M. Bosse and R. Zlot, "Continuous 3D scan-matching with a spinning 2d laser," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[14] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

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

[16] D. Holz and S. Behnke, "Mapping with micro aerial vehicles by registration of sparse 3d laser scans," in *The 13th International Conference on Intelligent Autonomous Systems (IAS)*, Padova, Italy, July 2014.

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

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

[19] ——, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, May 2015.

[20] J. Nocedal and S. Wright, *Numerical Optimization*. New York, Springer-Verlag, 2006.

[21] R. Andersen, *Modern methods for robust regression*. Sage, 2008.

[22] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computation Geometry: Algorithms and Applications (3rd Edition)*. Springer, 2008.

[23] G. Vogiatzis and C. Hernandez, "Video-based, real-time multi-view stereo," *Image and Vision Computing*, vol. 29, no. 7, pp. 434–441, 2011.

[24] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

[25] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *Workshop on Open Source Software (Collocated with ICRA 2009)*, Kobe, Japan, May 2009.

[26] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, Vancouver, Canada, Aug. 1981.