VILENS: Visual, Inertial, Lidar, and Leg Odometry for All-Terrain Legged Robots

David Wisth, *Graduate Student Member, IEEE*, Marco Camurri, *Member, IEEE*, and Maurice Fallon, *Senior Member, IEEE*

***Abstract*—We present visual inertial lidar legged navigation system (VILENS), an odometry system for legged robots based on factor graphs. The key novelty is the tight fusion of four different sensor modalities to achieve reliable operation when the individual sensors would otherwise produce degenerate estimation. To minimize leg odometry drift, we extend the robot's state with a linear velocity bias term, which is estimated online. This bias is observable because of the tight fusion of this preintegrated velocity factor with vision, lidar, and inertial measurement unit (IMU) factors. Extensive experimental validation on different ANYmal quadruped robots is presented, for a total duration of 2 h and 1.8 km traveled. The experiments involved dynamic locomotion over loose rocks, slopes, and mud, which caused challenges such as slippage and terrain deformation. Perceptual challenges included dark and dusty underground caverns, and open and feature-deprived areas. We show an average improvement of 62% translational and 51% rotational errors compared to a state-of-the-art loosely coupled approach. To demonstrate its robustness, VILENS was also integrated with a perceptive controller and a local path planner.**

***Index Terms*—Field robots, legged robots, localization, sensor fusion.**

Fig. 1. VILENS has been tested on a variety of platforms. *Top-Left:* ANYmal B300 at the Fire Service College in Moreton-On-Marsh (U.K.). *Top-Right:* ANYmal B300 modified for the DARPA SubT Challenge (Urban Circuit) in Olympia (Washington, USA) [10]. *Bottom-Left:* ANYmal C100 in a limescale mine in Wiltshire (U.K.). *Bottom-Right:* ANYmal C100 in an abandoned mine in Seemühle (Switzerland).

I. Introduction

THE increased maturity of quadruped robotics has been demonstrated in initial industrial deployments, as well as impressive results achieved by academic research. State estimation plays a key role in field deployment of legged machines: Without an accurate estimate of its location and velocity, the robot cannot build up a useful representation of its environment, or plan and execute trajectories to reach desired goal positions.

Most legged robots are equipped with a high-frequency (>250 Hz) proprioceptive state estimator for control and local mapping purposes. These are typically implemented as nonlinear filters fusing high-frequency signals, such as kinematics and inertial measurement unit (IMU) [1]–[3]. In ideal conditions (i.e., high friction, rigid terrain, slow speeds), these estimators have a limited (yet unavoidable) drift that is acceptable for local mapping and control.

However, deformable terrain, leg flexibility, and foot slippage can degrade estimation performance up to the point where local terrain reconstruction is unusable and multistep trajectories cannot be executed, even over short ranges. This problem is more evident when a robot is moving dynamically and can be the limiting factor when crossing rough terrain. This estimate is also affected by modeling errors, such as inaccurate leg lengths or nonzero contact point size.

Recent works have attempted to improve kinematic-inertial estimation accuracy by reducing the convergence time using invariant observer design [3] or by detecting unstable contacts and reducing their influence on the overall estimation [4]–[6].

Other approaches have incorporated additional exteroceptive sensing into the estimator to help reduce the pose error. These included either tightly coupled methods fusing camera, IMU, and kinematics [7] or loosely coupled methods combining lidar in addition to the other sensors [8], [9]. These approaches model the contact locations as being fixed and affected only by Gaussian noise. Both assumptions are broken when there is nonrigid terrain, kinematic chain flexibility, or mild but repeated foot

Manuscript received 7 January 2022; revised 6 May 2022; accepted 7 July 2022. This work was supported in part by the EU H2020 Projects THING and MEMMO and in part by the UKRI-funded ORCA Robotics Hub under Grant EP/R026173/1. The work of Maurice Fallon was supported by a Royal Society University Research Fellowship. The work of David Wisth was supported by a Google DeepMind studentship. This article was recommended for publication by Associate Editor Luca Carlone and Editor Francois Chaumette upon evaluation of the reviewers' comments. *(Corresponding author: David Wisth.)*

The authors are with the Oxford Robotics Institute, University of Oxford, OX1 2JD Oxford, U.K. (e-mail: davidw@robots.ox.ac.uk; mcamurri@robots.ox.ac.uk; mfallon@robots.ox.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2022.3193788.

Digital Object Identifier 10.1109/TRO.2022.3193788

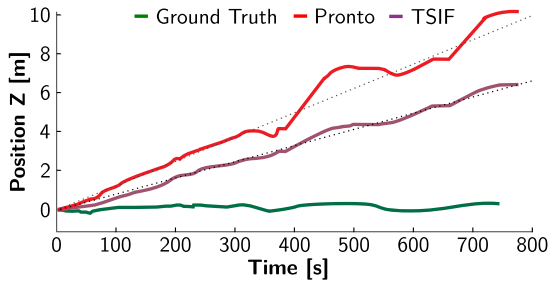

Fig. 2. Comparison between estimated robot elevation by Pronto [4] (red) and TSIF [2] (purple) kinematic-inertial state estimators, against ground truth (green) on the SMR experiment (see Section VI-B). Despite local fluctuations, the drift has a characteristic linear growth for a particular gait and terrain type. For example, between 350 s and 450 s, the robot walks over soft gravel, increasing the drift rate.

slippage. When these occur, fusion with exteroceptive sensors becomes nontrivial.

In our work, we aim to fuse all four sensor modalities (IMU, kinematics, lidar, and camera) in a tightly coupled fashion, with particular focus on the proper integration of leg kinematics in presence of non-ideal contacts, when slippage or terrain deformation occurs.

A. Motivation

Our work is motivated by the challenges and limitations of state estimation for the deployment of legged robots in extreme environments, as illustrated in Fig. 1. The DARPA Subterranean (SubT) Challenge [10] involves the deployment of a team of autonomous robots navigating unknown, dark, and unstructured underground environments. In such scenarios, individual sensor modalities can fail (e.g., due to camera blackouts or degenerate geometries for lidar), so robust sensor fusion is paramount. Additional requirements include limited computational budget and the need for high-frequency output to update the local footstep planner. For these reasons, we aim to use all of the sensors available on the robot (IMU, kinematics, lidar, and camera) to form constraints for lightweight sliding window optimization, as it can be more accurate than filter-based approaches [11].

When both lidar and camera fail at the same time, IMU integration alone would rapidly lead to divergence. Leg kinematics can prevent this, but special care has to be taken to fuse it with the other sensors in an effective way, especially in the presence of foot impacts and terrain deformation. Fig. 2 shows this effect on the ANYmal B300 robot traveling over various terrains. Two different kinematic-inertial estimators suffer from continuous elevation drift, which is locally approximately linear (see dashed lines in Fig. 2). Fahmi et al. [12] demonstrated that this drift can be caused by nonrigid, and nonstatic interaction of the legs and terrain during contact events.

An example of a foot contact event on soft gravel is shown in Fig. 3. The contact point was nonstatic throughout the entire sequence, violating the key assumption of most leg odometry algorithms. This is mainly due to the deformation of the ground and rubber foot, as well as the nonzero contact point size (the foot has a hemispherical profile). This can be considered as a systematic modeling error. Drift will be accumulated each time the robot steps on this terrain, leading to a biased estimate.

Fig. 3. Example of a foot contact sequence while trotting on gravel. (a) After the foot touches the ground, (b) both the terrain and robot's rubber foot deform as the controller increases the applied force. During the stance phase, (c) the contact point changes as the foot rolls over its hemispherical profile before (d) finally breaking contact.

One approach would be to further model the dynamic properties of the robot, such as torque-dependent bending [13], or to model the terrain directly within the estimator. However, this would be robot specific and terrain dependent—improving performance in one situation but degrading it elsewhere. Additionally, threshold-based methods only reject the most significant slippage or deformation events and ignore the small error accumulated with each footstep.

Inspired by the IMU bias estimation and preintegration methods from the work by Forster et al. [14], we instead propose to extend the state with a velocity bias term to estimate and reject these effects. This bias is observable when doing tight fusion with exteroceptive sensors (see Appendix C). This novel leg odometry factor computes a velocity measurement from kinematic sensing, preintegrates it, and estimates its bias to compensate for the characteristic drift of the leg odometry on slippery or deformable ground.

B. Contributions

This article makes the following contributions, significantly extending our previous work [15].

1) A novel factor that incorporates joint kinematic measurements. In contrast to the work in[15], this factor computes base velocities from the joint states, rather than taking them from an external filter. This avoids potential correlation due to double IMU usage and allows better modeling of the noise associated with the kinematics. A full derivation of the noise propagation is also provided in the Appendix.
2) Support and testing on a wider range of sensor modalities, including fisheye cameras and lidar. To the best of our knowledge, this is the first smoothing algorithm to combine inertial, legged, visual, and lidar data into a single-factor graph. In particular, in addition to the high-frequency lidar feature tracking from the work by Wisth et al. [16], we also introduce a lidar odometry factor for low-frequency accurate lidar registration. The wide support of complementary sensor modalities is the key to operation in adverse operational conditions, such as the DARPA SubT Challenge.
3) Extensive evaluation in a broad set of scenarios including different models of the ANYmal quadruped. In contrast to the work in [15], which was validated only on datasets, we tested the algorithm onboard the robot and integrated

it with both a dynamic perceptive controller [17] and an autonomous exploration planner [18].

The remainder of this article is presented as follows: In Section II, we review the literature on legged state estimation with a focus on sensor fusion via smoothing methods and contact modeling; Section III formally defines the problem addressed by the article and provides the required mathematical background; Section IV describes the factors used in our proposed graph formulation; Section V presents the implementation details of our system; Section VI presents the experimental results and discussion; Section VIII concludes with final remarks.

II. Related Work

We define multisensor state estimation as the joint processing of multiple proprioceptive and/or exteroceptive sensors to estimate the pose and velocity of a mobile robot [19]. Multisensor estimation can be divided into two broad categories: 1) *filtering* and 2) *smoothing*. Filtering methods such as MSC-KF [20] restrict the inference process to the latest state of the system while smoothing methods also estimate all or part of the past states. In Section II-A, we describe smoothing methods applied to multisensor estimation on legged robots. In Section II-B, we review the relevant methods for modeling nonlinearities at the contact point during leg odometry.

A. Multisensor Smoothing on Legged Robots

Multisensor smoothing for legged robots typically involves the fusion of IMU, leg odometry, and visual tracking within a probabilistic graphical model framework, such as factor graphs. Hartley et al. [21] proposed the first method to incorporate leg odometry into a factor graph. They extended the floating base state with the feet contact locations and defined two new factors to incorporate forward kinematics, where zero velocity constraints on the contact point of each foot were imposed. The two new factors were then combined with the preintegrated IMU factor from the work by Forster et al. [14] and relative pose measurements from the SVO visual odometry system [22]. In [7], the same authors extended this work to support multiple footsteps in the same kinematics factor. Both works were demonstrated on the Cassie bipedal robot through experiments in controlled environments lasting <2 m.

Fourmy et al. [23] proposed a factor-graph-based state estimation system for the Solo12 quadruped robot [24]. It fused both kinematic and dynamic information to estimate the base frame of the robot and performed online calibration of the offset between the base frame and center of mass. The authors claimed this was important for control since the center of mass of their robot was not known precisely from the CAD model. However, only very limited experimental results were presented, where the robot's torso moved while the feet were stationary. In the related field of motion planning, Xie et al. [25] modeled both the kinematics and dynamics of a 3-DoF manipulator using factor graphs. This approach took into account dynamics, contact forces, and joint actuation limits.

In our prior work [26], we proposed a tightly coupled visual-inertial-legged system based on the iSAM2 solver [27] running on the ANYmal robot. The method tracked visual features from a RealSense D435i stereo camera and optimized them as landmarks in a factor graph. Leg odometry was only loosely coupled with relative pose factors formed using the internal state estimator running on the robot [2]. This method was demonstrated through outdoor experiments in urban and industrial scenarios.

All these works were based on the assumption of a stationary point of foot contact. This assumption is violated every time there are slippages, or deformations of the leg and/or the ground. Contact detection methods can help to reject sporadic slippage or deformation events. However, when these occur regularly, they need to be modeled.

B. Modeling Contact Deformation and Slippage

In legged robotics, slippage and/or deformation have typically been addressed by assuming the contact location of a stance foot is entirely static throughout the stance period (yet affected by Gaussian noise). Thus, the main focus has been on detecting and ignoring the feet that are not in fixed contact with the ground. This is a relatively simple task when a foot is equipped with force/torque sensors. In this case, a high vertical component of the measured force would imply that the contact force is within the friction cone and, therefore, nonslipping. However, residual errors due to model uncertainty or deformation might persist. Fahmi et al. [12] have shown that incorrect contact detections on soft ground (i.e., detecting a "rigid" contact while the leg and ground are still deforming) are a key contributor to leg odometry drift.

Bloesch et al. [1] proposed an Unscented Kalman Filter design that fused IMU and differential kinematics. The approach used a threshold on the Mahalanobis distance of the filter innovation to infer velocity measurement outliers (caused by misclassified contact legs), which were then ignored. More recently, the idea was extended by Kim et al. [6] where the threshold on the Mahalanobis distance was replaced with a tunable threshold on the contact foot velocity estimated from the previous state.

For systems without feet sensors, more sophisticated methods were proposed. Hwangbo et al. [28] presented a probabilistic approach where information from kinematics, differential kinematics, and dynamics was fused within a hidden Markov model (HMM). This approach was later integrated with a dynamic trotting controller and demonstrated on the ANYmal robot walking on ice [5].

Instead of an HMM, Bledt et al. [29] proposed to fuse information from kinematics and dynamics, as well as additional input from the controller's gait cycle within a Kalman filter. Their work was demonstrated on the Cheetah 3 robot walking on rubble.

Hwangbo et al. [28] and Bledt et al. [29] focused more on detecting the contact as early as possible for control purposes, rather than determining the contact periods that would minimize the state estimation error.

Camurri et al. [4] proposed a contact detector that learned the optimal force threshold to detect a foot in contact for a specific gait, and an impact detector that adapted the measurement covariance online to reject unreliable measurements.

More recently, Lin et al. [30] proposed a contact estimation method based on a deep neural network taking IMU and kinematics as inputs and producing a binary contact state as output. The binary contact state was then used as input to the Invariant Kalman Filter [3] to estimate the state of the MIT Mini Cheetah robot walking on eight different terrains including grass, gravel, and concrete.

In our previous work [15], we used the approach from the work in[4] to fuse each leg's contribution into a single velocity measurement for our proposed factor graph method. The contact nonlinearities were modeled as a bias term on the linear velocity measurements from leg odometry. This could reduce the inconsistency between leg and visual odometry and provide a more robust pose and velocity estimate. However, the system was still dependent on an external filter to access the velocity estimates from the kinematics. Furthermore, lidar, which is now a common sensor on quadruped robots, was not used for estimation.

In contrast to the work in [15], in this work, we compute the velocity measurements from kinematics internally, instead of receiving them from an external filter. This has several benefits: It eliminates the double usage of the IMU signal (from the filter and from the preintegrated IMU factor), which breaks the independent measurement assumption; it properly models the error propagation from the joints to the feet, thereby estimating the covariance from leg kinematics more accurately; it allows for better integration, as the optimized IMU and kinematics biases are directly accessible from the kinematics module, whereas the external filter did not. Finally, it simplifies the overall estimation architecture, eliminating the dependence on an external component. We also introduce a lidar registration factor with local submapping and tested our algorithm online and onboard the robot in conjunction with a perceptive controller and a local path planner (as opposed to operation in postprocessing).

III. Problem Statement

Our objective is to estimate the history of poses and velocities of a legged robot equipped with a combination of sensors, including cameras (mono or stereo), IMUs, lidars, and joint sensors (encoders and torque sensors).

A. Notation

Scalars are lowercase italics $(a, b, c, \dots)$, matrices are uppercase Roman bold $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots)$, and vectors are lowercase Roman bold $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots)$. Reference frames are indicated in typeface $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \dots)$ and for physical quantities, we follow the frame decorator rules from the work in [31]. States and landmarks are bold italics $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \dots)$, whereas sensor measurements are uppercase calligraphics $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots)$. Finally, sets of time indices for measurements are uppercase sans $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \dots)$. Where appropriate, a time-dependent quantity will be shortened as $a(t_i) = a_i$.

B. Frame Definitions

The relevant reference frames are specified in Fig. 4 and include the fixed-world frame W, the base frame B, the IMU frame I, the camera frame C (left camera, when stereo), and the lidar frame L. When a foot is in contact with the ground, a contact frame K, fixed to Earth, is defined. Finally, a local anchor frame A, also fixed to Earth, is defined for lidar feature tracking, as detailed in Section IV-E.

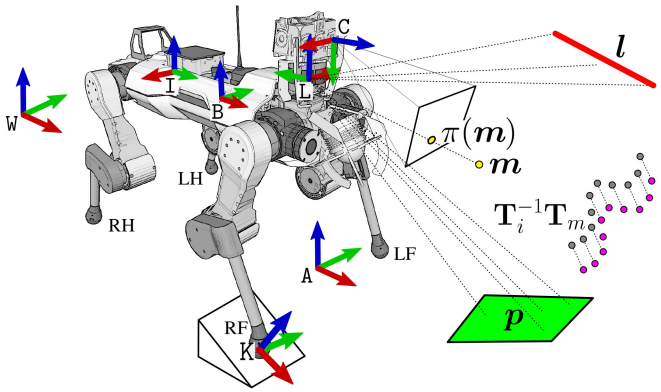

Fig. 4. Reference frames and landmark conventions. The world frame W is fixed to Earth. The base frame B, the camera's optical frame C, the lidar frame L, and the IMU frame I are all rigidly attached to the robot's chassis. The feet are conventionally named: Left front (LF), right front (RF), left hind (LH), and right hind (RH). When a foot touches the ground (e.g., RF), a contact frame K (perpendicular to the ground and parallel to W's y-axis) is defined. The primitives tracked by the system are points $\boldsymbol{m}$, lines $\boldsymbol{l}$, and planes $\boldsymbol{p}$. To improve numerical stability, when a new plane feature is detected, an additional local fixed anchor frame A is defined. Finally, a relative pose factor (between times t_i and t_m in figure) is created using lidar registration.

C. State Definition

The robot state at time t_i is defined as follows:

$$\boldsymbol{x}_i \triangleq [\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_i^g, \mathbf{b}_i^a, \mathbf{b}_i^\omega, \mathbf{b}_i^v] \in \mathrm{SO}(3) \times \mathbb{R}^{15} \quad (1)$$

where $\mathbf{R}_i \in \mathrm{SO}(3)$ is the orientation; $\mathbf{p}_i \in \mathbb{R}^3$ is the position; $\mathbf{v}_i \in \mathbb{R}^3$ is the linear velocity; $\mathbf{b}_i^g, \mathbf{b}_i^a \in \mathbb{R}^3$ are the usual IMU gyroscope and accelerometer biases. We expand the state with angular and linear velocity biases $\mathbf{b}_i^\omega, \mathbf{b}_i^v \in \mathbb{R}^3$ to model slippage, deformations, and other kinematics inaccuracies at the contact point.

Unless otherwise specified, the position ${}_{\mathtt{W}}\mathbf{p}_{\mathtt{WB}}$ and orientation $\mathbf{R}_{\mathtt{WB}}$ of the base are expressed in world coordinates, velocities of the base ${}_{\mathtt{B}}\mathbf{v}_{\mathtt{WB}}, {}_{\mathtt{B}}\boldsymbol{\omega}_{\mathtt{WB}}$ are in base coordinates, IMU biases ${}_{\mathtt{I}}\mathbf{b}^g$, ${}_{\mathtt{I}}\mathbf{b}^a$ are expressed in the IMU frame, and the velocity biases are expressed in the base frame, ${}_{\mathtt{B}}\mathbf{b}_{\mathtt{WB}}{}^\omega$, ${}_{\mathtt{B}}\mathbf{b}_{\mathtt{WB}}{}^v$.

In addition to the robot state, we also estimate the position of visual and lidar landmarks. Visual landmarks $\boldsymbol{m}$ are parameterized as 3-D points in Euclidean space and projected onto the image plane via the function $\pi(\cdot)$ (yellow dot in Fig. 4). Lidar landmarks are parameterized as planar and linear geometric primitives (green plane $\boldsymbol{p}$ and red line $\boldsymbol{l}$ in Fig. 4), as detailed in [16]. For brevity, we will later refer to any of the above landmarks with $\boldsymbol{f}$.

We define the history of states and landmarks, visible up to the current time t_k, as

$$\mathcal{X}_k \triangleq \{\boldsymbol{x}_i, \boldsymbol{m}_\ell, \boldsymbol{p}_\ell, \boldsymbol{l}_\ell\}_{i \in \mathsf{K}_k, \ell \in \mathsf{F}_k} \quad (2)$$

where K_k and F_k are the sets of all keyframe and landmark indices, respectively.

D. Measurements Definition

We denote with $\mathcal{I}_{ij}$ the IMU measurements received between two consecutive keyframes i and j. Each measurement includes the proper acceleration $\tilde{\mathbf{a}}$ and the rotational velocity $\tilde{\boldsymbol{\omega}}$, both expressed in the IMU frame. Similarly, we define the kinematics measurements $\mathcal{K}_{ij}$, which include the joint positions $\tilde{\boldsymbol{\alpha}}$ and velocities $\tilde{\dot{\boldsymbol{\alpha}}}$. The (mono or stereo) camera images and lidar point clouds collected at time t_i are expressed with $\mathcal{C}_i$ and $\mathcal{L}_i$, respectively. In practice, camera and lidar measurements are received at different times and frequencies, so they are first synchronized before being integrated into the graph (see Section V-A).

The full set of measurements within the smoothing window is defined as

$$\mathcal{Z}_k \triangleq \{\mathcal{I}_{ij}, \mathcal{K}_{ij}, \mathcal{C}_i, \mathcal{L}_i\}_{i,j\in\mathsf{K}_k}. \tag{3}$$

E. Maximum-A-Posteriori Estimation

We wish to maximize the likelihood of the measurements $\mathcal{Z}_k$ given the history of states and landmarks $\mathcal{X}_k$

$$\mathcal{X}_k^\star = \arg\max_{\mathcal{X}_k} p(\mathcal{X}_k|\mathcal{Z}_k) \propto p(\mathcal{X}_0)p(\mathcal{Z}_k|\mathcal{X}_k). \tag{4}$$

Given that the measurements are assumed to be conditionally independent and corrupted by white Gaussian noise, (4) can be formulated as a least squares minimization problem of the following form:

$$\begin{aligned}\mathcal{X}_k^\star = \arg\min_{\mathcal{X}_k} &\|\mathbf{r}_0\|^2_{\Sigma_0} + \sum_{i\in\mathsf{K}_k}\Big(\|\mathbf{r}_{\mathcal{I}_{ij}}\|^2_{\Sigma_{\mathcal{I}_{ij}}} + \|\mathbf{r}_{\mathcal{K}_{ij}}\|^2_{\Sigma_{\mathcal{K}_{ij}}} \\ &+ \|\mathbf{r}_{\mathbf{b}_{ij}}\|^2_{\Sigma_{\mathbf{b}_{ij}}} + \|\mathbf{r}_{\mathcal{L}_i}\|^2_{\Sigma_{\mathcal{L}_i}} + \sum_{\ell\in\mathsf{F}_i}\|\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{f}_\ell}\|^2_{\Sigma_{\boldsymbol{x}_i,\boldsymbol{f}_\ell}}\Big)\end{aligned} \tag{5}$$

where each term is the squared residual error associated to a factor type, weighted by the inverse of its covariance matrix, which will be detailed in Section IV. All the residuals (except the state prior $\mathbf{r}_0$) are added whenever a new keyframe i is created. These include preintegrated IMU $\mathbf{r}_{\mathcal{I}_{ij}}$ and kinematic velocity $\mathbf{r}_{\mathcal{K}_{ij}}$, IMU and kinematic velocity biases $\mathbf{r}_{\mathbf{b}_{ij}}$, lidar odometry $\mathbf{r}_{\mathcal{L}_i}$, and landmark primitives $\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{f}_\ell}$. The latter are further divided into point, line, and plane residuals

$$\begin{aligned}\sum_{\ell\in\mathsf{F}_i}\|\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{f}_\ell}\|^2_{\Sigma_{\boldsymbol{x}_i,\boldsymbol{f}_\ell}} &= \sum_{\ell\in\mathsf{M}_i}\|\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{m}_\ell}\|^2_{\Sigma_{\boldsymbol{x}_i,\boldsymbol{m}_\ell}} \\ &+ \sum_{\ell\in\mathsf{L}_i}\|\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{l}_\ell}\|^2_{\Sigma_{\boldsymbol{x}_i,\boldsymbol{l}_\ell}} + \sum_{\ell\in\mathsf{P}_i}\|\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{p}_\ell}\|^2_{\Sigma_{\boldsymbol{x}_i,\boldsymbol{p}_\ell}}.\end{aligned} \tag{6}$$

IV. Factor Graph Formulation

In this section, we describe the measurements, residuals, and covariances of the factors that form the graph shown in Fig. 5. For convenience, we summarize the IMU factors from the work by Forster et al. [14] in Section IV-A; our novel velocity factor is detailed in Section IV-B; Sections IV-C and IV-D describe the bias and stereo visual residuals, which are adapted from the work in [14] and [15] to include the velocity bias term and support for fisheye cameras, respectively. We also briefly introduce the lidar factor residuals from the work by Wisth et al. [16] in Section IV-E. Finally, the novel lidar registration residual is detailed in Section IV-F.

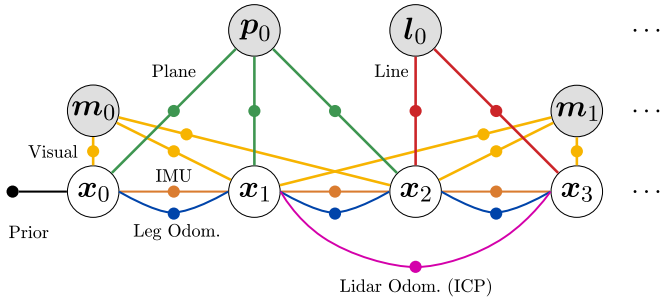

Fig. 5. VILENS factor graph structure. The factors are prior (black), visual (yellow), lidar planes (green), lidar lines (red), preintegrated IMU (orange), preintegrated velocity (from leg kinematics, blue), and lidar odometry (from ICP registration, magenta). State nodes are white, whereas landmarks are gray.

A. Preintegrated IMU Factors

As in [14], the IMU measurements are preintegrated to constrain the pose and velocity between two consecutive nodes of the graph, as well as provide high-frequency state updates between them. This uses a residual of the form

$$\mathbf{r}_{\mathcal{I}_{ij}} = \left[\mathbf{r}^\mathsf{T}_{\Delta\mathbf{R}_{ij}}, \mathbf{r}^\mathsf{T}_{\Delta\mathbf{v}_{ij}}, \mathbf{r}^\mathsf{T}_{\Delta\mathbf{p}_{ij}}\right] \tag{7}$$

where the individual elements of the residual are defined as

$$\mathbf{r}_{\Delta\mathbf{R}_{ij}} = \mathrm{Log}\left(\Delta\tilde{\mathbf{R}}_{ij}(\mathbf{b}_i^g)\right)\mathbf{R}_i^\mathsf{T}\mathbf{R}_j \tag{8}$$

$$\mathbf{r}_{\Delta\mathbf{v}_{ij}} = \mathbf{R}_i^\mathsf{T}\left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}\right) - \Delta\tilde{\mathbf{v}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) \tag{9}$$

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{p}_{ij}} = \mathbf{R}_i^\mathsf{T}&\left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\right) \\ &- \Delta\tilde{\mathbf{p}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a).\end{aligned} \tag{10}$$

As explained in more detail in [14], this formulation does not require recomputation of the integration between the two keyframes connected by the factor every time the linearization point changes. This makes the fusion of the high-frequency signal from the IMU and lower frequency measurements from camera and lidar computationally feasible.

Equations (8)–(10) depend on the optimized states and preintegrated measurements $\Delta\tilde{\mathbf{R}}_{ij}, \Delta\tilde{\mathbf{v}}_{ij}, \Delta\tilde{\mathbf{p}}_{ij}$. For convenience, we report them here (for the incorporation of the bias update and other details, see [14])

$$\Delta\tilde{\mathbf{R}}_{ij} = \prod_{k=i}^{j-1}\mathrm{Exp}\left(\left(\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g\right)\Delta t\right) \tag{11}$$

$$\Delta\tilde{\mathbf{v}}_{ij} = \sum_{k=i}^{j-1}\Delta\tilde{\mathbf{R}}_{ik}\left(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a\right)\Delta t \tag{12}$$

$$\Delta\tilde{\mathbf{p}}_{ij} = \sum_{k=i}^{j-1} \left[\Delta\tilde{\mathbf{v}}_{ik}\Delta t + \frac{1}{2}\Delta\tilde{\mathbf{R}}_{ik}\left(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a\right)\Delta t^2 \right]. \quad (13)$$

These are associated to the corresponding preintegrated noises, which can be expressed in iterative form

$$\begin{bmatrix} \delta\boldsymbol{\phi}_{i,k+1} \\ \delta\mathbf{v}_{i,k+1} \\ \delta\mathbf{p}_{i,k+1} \end{bmatrix} = \begin{bmatrix} \Delta\tilde{\mathbf{R}}_{k,k+1}^{\mathsf{T}} & \mathbf{0} & \mathbf{0} \\ -\Delta\tilde{\mathbf{R}}_{ik}(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a)^{\wedge}\Delta t & \mathbf{I} & \mathbf{0} \\ -\frac{1}{2}\Delta\tilde{\mathbf{R}}_{ik}(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a)^{\wedge}\Delta t^2 & \mathbf{I}\Delta t & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\phi}_{i,k} \\ \delta\mathbf{v}_{i,k} \\ \delta\mathbf{p}_{i,k} \end{bmatrix} + \begin{bmatrix} \mathbf{J}_R^k\Delta t & \mathbf{0} \\ \mathbf{0} & \Delta\tilde{\mathbf{R}}_{ik}\Delta t \\ \mathbf{0} & \frac{1}{2}\Delta\tilde{\mathbf{R}}_{ik}\Delta t^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}^g \\ \boldsymbol{\eta}^a \end{bmatrix} \quad (14)$$

from which the covariance $\boldsymbol{\Sigma}_{\mathcal{I}_{ij}}$ can be computed.

B. Preintegrated Leg Odometry Factors

Inspired by the preintegrated IMU factor, we define here the preintegrated velocity factor for leg odometry. This factor is used to estimate the position of the robot from high-frequency joint kinematics measurements.

To derive a velocity signal from leg kinematics, we first need to estimate the stance legs, i.e., the set of legs in stable contact with the ground (see Section IV-B1). Then, we fuse the odometry from these legs as a combined velocity measurement (see Section IV-B2). To model the nonlinearities at the contact point, the linear velocity bias, and its residuals are introduced in Section IV-B3. The preintegrated measurements and their residuals are described in Sections IV-B4 and IV-B5, respectively. Finally, a special case of this twist bias is outlined in Section V-C.

1) Stance Estimation: The feet of the robot are approximated as points. For simplicity, the contact point is assumed to be on a fixed point at the center of the foot of the robot, which in our case is a soft rubber sphere with a 2 cm radius. Since no direct force measurement is assumed to be available, we compute the ground reaction force (GRF) at each foot from the dynamics equation of motion

$$\mathbf{f} = -\left(\mathbf{J}_p(\boldsymbol{\alpha})^{\mathsf{T}}\right)^{\dagger}\left(\boldsymbol{\tau} + \mathbf{h}(\boldsymbol{\alpha}) + F^{\mathsf{T}}\begin{bmatrix}\dot{\boldsymbol{\omega}} \\ \dot{\mathbf{v}}\end{bmatrix}\right) \quad (15)$$

where † indicates the Moore–Penrose pseudoinverse operation, $\boldsymbol{\alpha}$ are the joint positions, $\mathbf{J}_p(\cdot) \in \mathbb{R}^{3\times 3}$ is the Jacobian of the forward kinematics function relative to the point foot (later indicated with $\mathbf{f}_p(\cdot) \in \mathbb{R}^3$), $\boldsymbol{\tau}$ are the joint torques, $\mathbf{h}(\boldsymbol{\alpha})$ are the Coriolis terms, and F is the matrix of spatial forces required at the floating base to support unit accelerations about each joint variable [32]. In practice, both $\dot{\boldsymbol{\omega}}$ and $\dot{\mathbf{v}}$ are obtained from the IMU and projected into the base frame, where the angular acceleration is computed by numerical differentiation and the linear velocity is computed by compensating for the gravity using the latest orientation estimate.

For each leg that is in contact with the ground, we assume *rigid, nonslipping contact*. By thresholding the vertical component of the GRF for each foot, we get the set of binary contact states for all the legs. We indicate the subset of the legs in contact as $S \subseteq L$, with $L = \{\text{LF}, \text{RF}, \text{LH}, \text{RH}\}$ (see Fig. 4).

2) Combined Kinematic Velocity Measurement: Given a leg $s \in S$, we can compute the linear velocity of the robot's floating base at time t_i as follows:

$$\mathbf{v}_s = -\mathbf{J}_p(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}} - \boldsymbol{\omega} \times \mathbf{f}_p(\boldsymbol{\alpha}). \quad (16)$$

Both the joint positions and velocities are measured from encoders and corrupted by additive zero-mean Gaussian noise

$$\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha} + \boldsymbol{\eta}^{\alpha} \quad (17)$$

$$\tilde{\dot{\boldsymbol{\alpha}}} = \dot{\boldsymbol{\alpha}} + \boldsymbol{\eta}^{\dot{\alpha}} \quad (18)$$

where $\boldsymbol{\eta}^{\alpha}, \boldsymbol{\eta}^{\dot{\alpha}}$ are available from the sensor specifications. After substituting (17) and (18) into (16), we can formulate a linear velocity measurement [1]

$$\mathbf{v}_s = -\mathbf{J}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha}) \cdot (\tilde{\dot{\boldsymbol{\alpha}}} - \boldsymbol{\eta}^{\dot{\alpha}}) - \boldsymbol{\omega} \times \mathbf{f}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha}). \quad (19)$$

As detailed in Appendix A, the noise terms from (19) can be separated, so the measurement can be expressed as

$$\tilde{\mathbf{v}}_s = \mathbf{v}_s + \boldsymbol{\eta}_s^v \quad (20)$$

$$\boldsymbol{\eta}_s^v = -\left(\mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\tilde{\dot{\boldsymbol{\alpha}}} + \boldsymbol{\omega}^{\wedge}\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\right)\boldsymbol{\eta}^{\alpha} - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}} \quad (21)$$

where $\mathbf{H}_p(\cdot) \in \mathbb{R}^{3\times 3\times 3}$ is the Hessian of the forward kinematics function $\mathbf{f}_p(\boldsymbol{\alpha})$.

Equation (20) is valid only when the leg s is in contact with the ground. Since foot contacts happen intermittently while the robot walks, multiple legs can be in contact simultaneously. As each velocity measurement from (20) is associated with a Gaussian noise term $\boldsymbol{\eta}_s^v$, it would be possible to add to the graph one independent velocity measurement per stance leg. This would, however, unnecessarily increase the graph complexity, as a closed form for fusing the measurements from the stance legs into one can be computed instead [33]. Additionally, treating the leg velocities separately would require explicit handling of contact switching every time a new step is made to ensure all measurements are used [7].

From (21), since $\boldsymbol{\eta}_s^v$ is a linear combination of zero-mean Gaussians, it is also a zero-mean Gaussian with covariance $\boldsymbol{\Sigma}_s$. Therefore, a combined velocity measurement for all the stance legs can then be computed as a weighted average of the velocity measurements of each stance leg, based on the information matrix $\boldsymbol{\Omega}_s = \boldsymbol{\Sigma}_s^{-1}$

$$\boldsymbol{\Sigma}_v = \left(\sum_{s\in S}\boldsymbol{\Omega}_s\right)^{-1} \quad (22)$$

$$\tilde{\mathbf{v}} = \boldsymbol{\Sigma}_v \sum_{s\in S}\left(\boldsymbol{\Omega}_s\tilde{\mathbf{v}}_s\right) \quad (23)$$

$$\boldsymbol{\eta}^v \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_v) \quad (24)$$

where the compound velocity measurement noise $\boldsymbol{\eta}^v$ is sampled from a zero-mean Gaussian with covariance $\boldsymbol{\Sigma}_v$.

We now have a linear velocity measurement[1] $\tilde{\mathbf{v}}$ and its noise $\boldsymbol{\eta}^v$ that can be used for the preintegrated velocity factor

$$\tilde{\mathbf{v}} = \mathbf{v} + \boldsymbol{\eta}^v. \quad (25)$$

[1]Not to be confused with the IMU preintegrated velocity measurement $\Delta\tilde{\mathbf{v}}$.

3) Velocity Bias: On slippery or deformable terrains, the constraint from (16) will not be true, resulting in incorrect leg velocities, and thus drift in the final odometry estimate. In our experience, this velocity drift is locally constant and is gait- and terrain-dependent (see Sections I-A and V-C, and Figs. 2 and 3).

For these reasons, we relax (16) by adding a slowly varying bias term $\mathbf{b}^v$ to (25)

$$\tilde{\mathbf{v}} = \mathbf{v} + \mathbf{b}^v + \boldsymbol{\eta}^v. \tag{26}$$

This term incorporates the characteristic drift caused by leg or terrain compression, slippage, and impacts occurring at the contact point.

4) Preintegrated Velocity Measurements: We derive the preintegrated linear velocity and noise only, as the preintegrated rotation measurement $\Delta\tilde{\boldsymbol{\Theta}}_{ij}$ and noise $\delta\boldsymbol{\theta}_{ij}$ have the same form as (11) and (14)

$$\Delta\tilde{\boldsymbol{\Theta}}_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}\left((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^{\omega})\,\Delta t\right) \tag{27}$$

$$\delta\boldsymbol{\theta}_{ij} = \sum_{k=i}^{j-1} \Delta\tilde{\boldsymbol{\Theta}}_{k+1,j}^{\mathsf{T}} \mathbf{J}_R^k \boldsymbol{\eta}^{\omega} \Delta t. \tag{28}$$

The position at time $t_j = t_i + \Delta t$ is

$$\mathbf{p}(t_j) = \mathbf{p}(t_i) + \int_{t_i}^{t_j} \mathbf{v}(\tau)\,\mathrm{d}\tau. \tag{29}$$

Assuming constant velocity between t_i and t_j, we can iteratively calculate the position in discrete time domain form[2]

$$\mathbf{p}_j = \mathbf{p}_i + \sum_{k=i}^{j-1} \left[\mathbf{R}_k(\tilde{\mathbf{v}}_k - \mathbf{b}_i^v - \boldsymbol{\eta}_k^v)\Delta t\right]. \tag{30}$$

From (30), a relative measurement can be obtained[3]

$$\Delta\boldsymbol{\kappa}_{ij} = \mathbf{R}_i^{\mathsf{T}}(\mathbf{p}_j - \mathbf{p}_i) = \sum_{k=i}^{j-1} \left[\Delta\mathbf{R}_{ik}(\tilde{\mathbf{v}}_k - \mathbf{b}_i^v - \boldsymbol{\eta}_k^v)\Delta t)\right]. \tag{31}$$

With the substitution $\Delta\mathbf{R}_{ik} = \Delta\tilde{\boldsymbol{\Theta}}_{ik}\mathrm{Exp}(-\delta\boldsymbol{\theta}_{ik})$ to include the preintegrated rotation measurements [from (27)–(28)], and the approximation $\mathrm{Exp}(\boldsymbol{\phi}) \simeq \mathbf{I} + \boldsymbol{\phi}^{\wedge}$, (31) becomes

$$\Delta\boldsymbol{\kappa}_{ij} \simeq \sum_{k=i}^{j-1} \left[\Delta\tilde{\boldsymbol{\Theta}}_{ik}(I - \delta\boldsymbol{\theta}_{ik}^{\wedge})(\tilde{\mathbf{v}}_k - \mathbf{b}_i^v - \boldsymbol{\eta}_k^v)\Delta t\right]. \tag{32}$$

By separating the measurement and noise components of (32) and ignoring higher order terms, we can define the *preintegrated leg odometry position measurement* $\Delta\tilde{\boldsymbol{\kappa}}_{ij}$ and *noise* $\delta\boldsymbol{\kappa}_{ij}$ as

$$\Delta\tilde{\boldsymbol{\kappa}}_{ij} \triangleq \sum_{k=i}^{j-1} \left[\Delta\tilde{\boldsymbol{\Theta}}_{ik}(\tilde{\mathbf{v}}_k - \mathbf{b}_i^v)\Delta t\right] \tag{33}$$

$$\delta\boldsymbol{\kappa}_{ij} \triangleq \sum_{k=i}^{j-1} \left[\Delta\tilde{\boldsymbol{\Theta}}_{ik}\boldsymbol{\eta}_k^v\Delta t - \Delta\tilde{\boldsymbol{\Theta}}_{ik}(\tilde{\mathbf{v}}_k - \mathbf{b}_i^v)^{\wedge}\delta\boldsymbol{\theta}_{ik}\Delta t\right]. \tag{34}$$

Note that both quantities still depend on the twist biases $\mathbf{b}^{\omega}, \mathbf{b}^v$. When these change, we would like to avoid the recomputation of (33). Given a small change $\delta\mathbf{b}$ such that $\mathbf{b} = \bar{\mathbf{b}} + \delta\mathbf{b}$, we use a first-order approximation to find the new measurement, as done in [14]

$$\Delta\tilde{\boldsymbol{\kappa}}_{ij}(\mathbf{b}^{\omega}, \mathbf{b}^v) \simeq \Delta\tilde{\boldsymbol{\kappa}}_{ij}(\bar{\mathbf{b}}^{\omega}, \bar{\mathbf{b}}^v) + \frac{\partial\Delta\tilde{\boldsymbol{\kappa}}_{ij}}{\partial\mathbf{b}^{\omega}}\delta\mathbf{b}^{\omega} + \frac{\partial\Delta\tilde{\boldsymbol{\kappa}}_{ij}}{\partial\mathbf{b}^v}\delta\mathbf{b}^v. \tag{35}$$

5) Residuals: The factor residuals include rotation and translation

$$\mathbf{r}_{\mathcal{K}_{ij}} = \left[\mathbf{r}_{\Delta\tilde{\boldsymbol{\Theta}}_{ij}}^{\mathsf{T}},\ \mathbf{r}_{\Delta\boldsymbol{\kappa}_{ij}}^{\mathsf{T}}\right] \tag{36}$$

$$\mathbf{r}_{\Delta\boldsymbol{\Theta}_{ij}} = \mathrm{Log}\left(\Delta\tilde{\boldsymbol{\Theta}}_{ij}(\mathbf{b}_i^{\omega})\right)\mathbf{R}_i^{\mathsf{T}}\mathbf{R}_j \tag{37}$$

$$\mathbf{r}_{\Delta\boldsymbol{\kappa}_{ij}} = \mathbf{R}_i^{\mathsf{T}}(\mathbf{p}_j - \mathbf{p}_i) - \Delta\tilde{\boldsymbol{\kappa}}_{ij}(\mathbf{b}_i^{\omega}, \mathbf{b}_i^v). \tag{38}$$

Since (37) has the same form as (8), when the same angular velocity measurements are used for both the IMU and leg odometry factors, the following relations hold:

$$\Delta\tilde{\boldsymbol{\Theta}}_{ij} = \Delta\tilde{\mathbf{R}}_{ij} \qquad \mathbf{b}^{\omega} = \mathbf{b}^g \qquad \delta\boldsymbol{\theta}_{ij} = \delta\boldsymbol{\phi}_{ij}. \tag{39}$$

In this case, we can avoid double counting the IMU signal by setting the rotational residual $\mathbf{r}_{\Delta\boldsymbol{\Theta}_{ij}}$ to zero. This was not possible in [15], where an external filter was used.

6) Covariance: After simple manipulation of (34), the covariance of the residual $\mathbf{r}_{\mathcal{K}_{ij}}$ can be computed iteratively

$$\boldsymbol{\Sigma}_{i,k+1}^{\mathcal{K}} = \mathbf{A}\boldsymbol{\Sigma}_{i,k}^{\mathcal{K}}\mathbf{A}^{\mathsf{T}} + \mathbf{B}\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{\mathcal{K}}\mathbf{B}^{\mathsf{T}} \tag{40}$$

where the first term evolves from an initial condition of $\boldsymbol{\Sigma}_{i,i}^{\mathcal{K}} = \mathbf{0}$, whereas the second term $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{\mathcal{K}}$ is fixed and taken from sensor specifications. The complete derivation of the multiplicative terms $\mathbf{A}$ and $\mathbf{B}$ are detailed in Appendix B.

C. Bias Residuals

The bias terms are intended to change slowly and are, therefore, modeled as a Gaussian random walk. The residual term for the cost function is as follows:

$$\|\mathbf{r}_{\mathbf{b}_{ij}}\|_{\Sigma_{\mathbf{b}_{ij}}}^2 \triangleq \|\mathbf{b}_j^g - \mathbf{b}_i^g\|_{\Sigma_{\mathbf{b}_{ij}^g}}^2 + \|\mathbf{b}_j^a - \mathbf{b}_i^a\|_{\Sigma_{\mathbf{b}_{ij}^a}}^2 + \|\mathbf{b}_j^v - \mathbf{b}_i^v\|_{\Sigma_{\mathbf{b}_{ij}^v}}^2 \tag{41}$$

where the covariance matrices are determined by the expected rate of change of these quantities. In particular, $\boldsymbol{\Sigma}_{\mathbf{b}_{ij}^g}$, $\boldsymbol{\Sigma}_{\mathbf{b}_{ij}^a}$ are available from IMU specifications while $\boldsymbol{\Sigma}_{\mathbf{b}_{ij}^v}$ depends on the drift rate of the leg odometry bias, which is found empirically.

[2]For simplicity, we keep the symbol $\boldsymbol{\eta}^v$ for the noise in the discrete domain.

[3]The variable name change is to avoid confusion with the position measurements and noise from the IMU factor.

TABLE I
SENSOR SPECIFICATIONS OF THE EXPERIMENTAL PLATFORMS

Sensor	Model	Hz	Specs
Sensors common to ANYmal B300 and C100			
Encoder	ANYdrive	400	*Res:* $<0.025°$
Torque	ANYdrive	400	*Res:* <0.1 N.m
Lidar	Velodyne VLP-16	10	*Res:* $16\,\text{px} \times 1824\,\text{px}$
ANYmal B300 (experiments: SMR, FSC)			
IMU	Xsens MTi-100	400	*Init Bias:* $0.2\,°/\text{s} \mid 5\,\text{mg}$ *Bias Stab:* $10\,°/\text{h} \mid 15\,\text{mg}$
Gray Stereo Camera	RealSense D435i	30	*Res:* $848\,\text{px} \times 480\,\text{px}$ *FoV (Diag.):* $100.6°$
ANYmal B300 (experiment: SUB)			
IMU	Xsens MTi-100	400	*Init Bias:* $0.2\,°/\text{s} \mid 5\,\text{mg}$ *Bias Stab:* $10\,°/\text{h} \mid 15\,\text{mg}$
RGB Mono Camera	FLIR BFS-U3-16S2C-CS	30	*Res:* $1440\,\text{px} \times 1080\,\text{px}$ *FoV (Diag.):* $150°$
ANYmal C100 (experiments: LSM, SMM)			
IMU	Epson G365	400	*Init Bias:* $0.1\,°/\text{s} \mid 3\,\text{mg}$ *Bias Stab:*$1.2\,°/\text{h} \mid 15\,\text{mg}$
Gray Stereo Camera	Sevensense Alphasense	30	*Res:* $720\,\text{px} \times 540\,\text{px}$ *FoV (Diag.):* $165.4°$

D. Visual Factors

We use two main factors related to visual measurements. The first is a traditional reprojection error given by [26]

$$\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{m}_\ell} = \begin{pmatrix} \pi_u^L(\mathbf{R}_i, \mathbf{p}_i, \boldsymbol{m}_\ell) - u_{i,\ell}^L \\ \pi_u^R(\mathbf{R}_i, \mathbf{p}_i, \boldsymbol{m}_\ell) - u_{i,\ell}^R \\ \pi_v(\mathbf{R}_i, \mathbf{p}_i, \boldsymbol{m}_\ell) - v_{i,\ell} \end{pmatrix} \tag{42}$$

where $(u^L, v), (u^R, v)$ are the pixel locations of the detected landmark. If only a monocular camera is available then the second element of (42) is not used.

The second factor uses the overlapping fields of view of the lidar and camera sensors (where applicable) to provide depth estimates for visual features, as described in [16].

In addition to the standard camera projective model, in this article, we also introduce support for fisheye cameras with equidistant distortion [34]. As demonstrated in [35], a high camera FoV allows for tracking of features for longer periods of time, but for large open areas, the loss of pixel density becomes significant for a diagonal FoV of 180° or more. To maximize versatility in underground scenarios, our fisheye configurations have a moderately large diagonal FoV of 150–165° (see Table I).

To add the landmark to the graph, we first detect and track features in the original, distorted image to avoid costly image undistortion. We then undistort the individual feature locations before adding them to the factor graph using (42).

As shown in Fig. 6, the distortion far from the focal point can be quite large. This means the angular resolution of the camera varies across the image. To correctly model the visual feature uncertainty $\boldsymbol{\Sigma}_{\boldsymbol{x}_i,\boldsymbol{m}_\ell}$, we first select a set of points around the landmark in the distorted image, based on the specified visual feature tracker uncertainty. We then undistort these points and use a least-squares method to fit an ellipse to these points. For ellipse fitting we use at least six points since the covariance ellipse is, in general, not aligned to the x or y axes. This gives us an accurate uncertainty matrix in the undistorted image coordinates. Since the distortion is constant, the undistortion map and covariance ellipses can be precomputed for efficiency and, thus, have no impact on runtime performance.

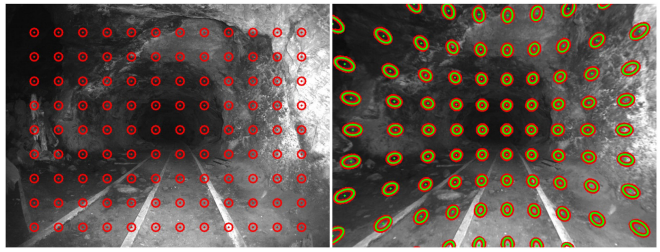

Fig. 6. Example of fisheye covariance warping and fitting. *Left:* Original samples (red points) and 1-σ bound of 9 pixels (red circles) in the distorted image. *Right:* the undistorted points from the 1-σ bound (red) and their ellipse fit (green) in the undistorted image. Notice that the undistorted covariance becomes larger near the edge of the image where there is greater distortion.

We use $\Sigma_{\boldsymbol{x}_i,\boldsymbol{m}_\ell}$ of between 1 and 2 pixels (adapted by the fisheye covariance warping if applicable).

E. Plane and Line Factors

We extract and track geometric primitives, specifically planes and lines over successive lidar scans. This is similar to how visual features are tracked. Planes are defined using the Hessian norm form [16]

$$\boldsymbol{p} = \left\{ \langle \hat{\mathbf{n}}, d \rangle \in \mathbb{R}^4 \,|\, \hat{\mathbf{n}} \cdot (x, y, z) + d = 0 \right\} \tag{43}$$

while lines use the minimal parameterization from [36]

$$\boldsymbol{l} = \left\{ \langle \mathbf{R}, (a, b) \rangle \in \text{SO}(3) \times \mathbb{R}^2 \right\}. \tag{44}$$

In this article, we extend the plane and line factors from the work in [16] to support local linearization points. This is important in large-scale environments, where the transformation of the plane from world to sensor coordinates can cause numerical instability in the optimization (small changes in the angle of the normal can cause very large changes in the position of the plane or line). This instability increases as the sensor gets further from the origin of the world frame.

Therefore, inspired by the work in [37], we introduce a local linearization frame for planes and lines, which we call the anchor frame $\mathtt{A}$. This decreases convergence time [37] and increases numerical stability. $\mathtt{A}$ is arbitrarily defined at the first frame in which the landmark is observed.

When a plane $\tilde{\boldsymbol{p}}_\ell$ is measured at time t_i, the corresponding residual is the difference between $\tilde{\boldsymbol{p}}$ and the estimated plane $\boldsymbol{p}_\ell$ is transformed into the local reference frame

$$\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{p}_\ell} = \left(\left(\mathbf{T}_{\mathtt{WA}}^{-1} \mathbf{T}_{\mathtt{WB}} \right) \otimes \boldsymbol{p}_\ell \right) \ominus \tilde{\boldsymbol{p}}_\ell \tag{45}$$

where $\mathbf{T}_{\mathtt{WA}}$ is the pose of the robot at the time where the plane is first detected, $\mathbf{T}_{\mathtt{WB}} = [\mathbf{p}_i, \mathbf{R}_i]$ is the current pose estimate of the robot.

Similarly, the residual between a measured line $\tilde{\boldsymbol{l}}_i$ and its prediction is defined as follows:

$$\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{l}_\ell} = \left(\left(\mathbf{T}_{\mathtt{WA}}^{-1}\mathbf{T}_{\mathtt{WB}}\right) \boxtimes \boldsymbol{l}_\ell\right) \boxminus \tilde{\boldsymbol{l}}_\ell \tag{46}$$

where $\otimes$, $\boxtimes$ apply a transformation and $\ominus$, $\boxminus$ are difference operators to planes and lines, respectively [16].

The line and plane covariances $\boldsymbol{\Sigma}_{\boldsymbol{x}_i,\boldsymbol{l}_\ell}$, $\boldsymbol{\Sigma}_{\boldsymbol{x}_i,\boldsymbol{p}_\ell}$ are determined by analyzing the covariance of the inlier points for each feature. This results in typical covariances of 10 cm and 3 deg.

F. Lidar Registration Factor

The lidar feature tracking allows for continuous motion estimation at the full lidar frame rate. Approaches such as in [38] instead accumulate the features into a local submap for a certain number of frames before integrating them into the factor graph. Local submapping allows for accurate pose estimation refinement but at a lower frequency. In this work, in addition to the feature tracking, we also integrate fine-grained lidar registration with local submapping. The registration is based on the iterative closest point (ICP) approach by Pomerleau et al. [39]. To make the registration more robust, a local submap is maintained with all the scans successfully registered in the past 5 m traveled. The submaps constitute the reference cloud of the ICP registration process.

Although ICP typically runs at a lower frequency than the other signals, it is able to take advantage of the high-quality motion prior provided by the other modules that run at a higher frequency (e.g., IMU) to motion correct scans.

ICP odometry measurements are added to the factor graph as relative pose factors between nonconsecutive keyframes i and m (e.g., nodes 1 and 3 in Fig. 5)

$$\mathbf{r}_{\mathcal{L}_i} = \Phi(\widetilde{\mathbf{T}}_i^{-1}\widetilde{\mathbf{T}}_m, \mathbf{T}_i^{-1}\mathbf{T}_m) \tag{47}$$

where $\mathbf{T}$ is the estimated pose, $\widetilde{\mathbf{T}}$ is the estimate from the ICP module, and Φ is the lifting operator defined in [14]. Note that ICP registration is prone to failure in environments with degenerate geometries, such as long tunnels. For this reason, a robust cost function is used to reject the factor in such degenerate situations.

For simplicity, the lidar registration covariance $\boldsymbol{\Sigma}_{\mathcal{L}}$ is set as a constant value, determined empirically during preliminary experiments and left unchanged for all our experiments, with potential outliers being handled by the robust cost function.

V. Implementation

The architecture of the VILENS state estimator is shown in Fig. 7. Three parallel threads process the preintegration of IMU and kinematics, camera feature tracking, lidar processing (both feature tracking and ICP registration) and a fourth carries out the subsequent optimization. A forward propagated state from the IMU factor is output from the preintegration thread at IMU frequency (i.e., 400 Hz). This is used to motion correct and synchronize the lidar point clouds and is available for high-frequency tasks such as control (for simplicity, we assume $\mathcal{L}_i$ is already undistorted in Fig. 7). When a new keyframe is processed, the preintegrated measurements and tracked landmarks are collected

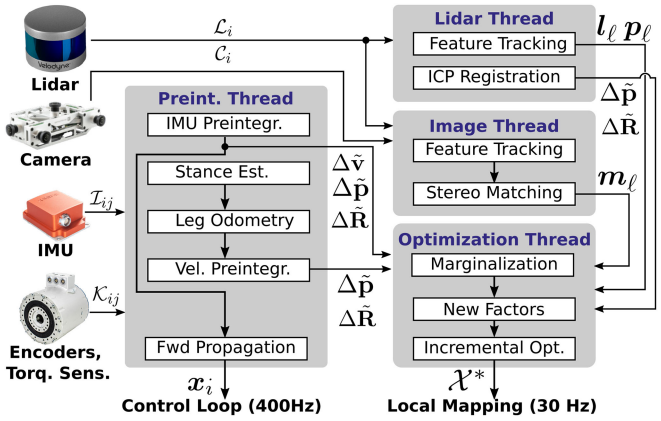

Fig. 7. Block diagram of the VILENS algorithm. Three independent threads (preintegration, camera, and lidar) process the data while a fourth thread performs the optimization on the data already processed by the other threads.

by the optimization thread, whereas the other threads process the next set of measurements. When the optimization step is complete, the optimal set of states is produced at the keyframe rate for use by local mapping and path planning.

The factor graph is solved using a fixed lag smoothing framework based on the efficient incremental optimization solver iSAM2 [27], implemented as part of the GTSAM library [11]. For these experiments, we use a lag time of 5 s. All visual and lidar factors are added to the graph using the dynamic covariance scaling (DCS) [40] robust cost function to reduce the effect of outliers.

A. Sensor Synchronization

To perform joint optimization across all the sensor modalities, the measurements first need to be synchronized. As in [26], the IMU and kinematic measurements, which have high frequency, are interpolated to match the camera timestamps. To synchronize the lidar measurements with the camera, we instead follow the approach described in [16]. When a new point cloud is received, its points are motion compensated using the IMU propagated state (from the preintegrated IMU factor) using the closest camera keyframe timestamp as reference, instead of the start or end of the timestamp, as commonly done. In this way, the minimum number of nodes will be added to the graph. The optimization is performed *jointly* between IMU, kinematics, camera, and lidar inputs. This also ensures a fixed output frequency, i.e., the camera keyframe rate.

B. Forward Kinematics and Dynamics

The forward kinematics and dynamics are implemented in RobCoGen [41], which is a computationally efficient kinematics and dynamics solver, as demonstrated in this comparison paper [42]. RobCoGen takes the current joint configuration and base state to calculate the position, velocity, and force at each of the feet. The covariance values for the kinematics are taken directly from the encoder datasheet.

C. Twist Bias Modes

The introduction of a twist bias is motivated by nonideal contact events, which can cause drift in the estimated velocity. Therefore, if no new contact events are occurring (i.e., the robot is not making any new steps), this velocity bias should not be present.

To account for this, when no footsteps are being made, we disable the twist bias and simply use relative pose factors instead. This is important for legged robots since the robot can still move (e.g., roll and pitch) while the feet are stationary. This is in contrast to the zero velocity mode presented in [15] where access to the contact states was not possible due to the use of an external filter to process the kinematics.

In practice, we detect this *low drift* state when the majority of the robot's feet (i.e., 3 or 4 feet) are in constant contact with the ground for an extended period of time (i.e., more than 200 ms). As soon as new footsteps are made we exit this low-drift state and return to normal operation.

D. Visual and Point Cloud Feature Tracking

We detect visual features using the FAST corner detector and track them between successive frames using the KLT feature tracker. Outliers are rejected using a RANSAC-based fundamental matrix consistency check (similar to [43]). For point cloud features, we use the feature tracking approach based on the work in [16] to extract and track geometric primitives (namely planes $\boldsymbol{p}_\ell$ and lines $\boldsymbol{l}_\ell$) over time.

Keyframes are added when the mean visual feature movement between consecutive frames is greater than a threshold (1 pixel), with minimum (5 Hz) and maximum (15 Hz) keyframe frequency bounds.

E. Zero-Velocity Update Factors

To limit drift and factor graph growth, we continuously query if the robot is stationary by using a voting mechanism. If the majority of sensor modalities detect no motion, then we add a zero velocity constraint to the graph. This method can detect when the robot is stationary, even when one or more sensor inputs are not available (e.g., legs are not in contact with the ground).

For example, the IMU and leg odometry threads report zero velocity when position (rotation) is less than 0.1 mm (0.5°) between two keyframes. The image thread reports zero velocity when the average feature movement between frames is less than 0.5 pixels. The lidar thread reports zero velocity when the motion induced by tracking planes and lines is less than 0.2 m and 5°, respectively.

F. Calibration

We use the open-source camera and IMU calibration toolbox Kalibr [44] to compute the intrinsic and extrinsic calibrations of the cameras. The IMU, lidar, and kinematics positions are known from accurate CAD models and are rigidly attached to the frame of the robot.

All sensors are hardware-synchronized where possible (using either precision time protocol, EtherCAT, or hardware triggering). The only exception is the Realsense D435i cameras, which are software-synchronized using the manufacturer's driver, before removing any clock drift using continuous-time maximum likelihood estimation [45].

G. Initialization

We initialize the system by averaging the first 1 s of IMU data at system start-up (assuming the IMU is stationary). This allows us to find the gyroscope bias and initial estimates for the orientation of the gravity vector. As they cannot be estimated from stationary data [46], accelerometer biases are initially assumed to be either zero or a known constant value. The accelerometer biases become observable after the robot starts moving, thanks to the features tracked by the exteroceptive sensors, as demonstrated in [47]. The scale is known from either stereo correspondence or lidar measurements.

VI. Experimental Results

In this section, we will describe the platforms and the dataset used in our evaluations (see Sections VI-A and VI-B). We will then demonstrate how the tight integration of kinematics into a factor graph framework allows for an easy combination of the different sensor modalities (see Section VI-C). Finally, we will end with an analysis of specific challenging situations in the dataset, and discuss computational performance (see Sections VII and VI-F).

A. Experimental Platforms

The platforms used for our experiments are the ANYbotics ANYmal B300 [48] and C100 quadrupeds (see Fig. 1). Both robots have 4 identical legs giving a total of 12 active Degrees-of-Freedom (DoF) and are equipped with an IMU and a Velodyne VLP-16 lidar. Each leg has joint encoders and torque sensors. In some of the experiments, the robots were modified from the stock version to compete in the DARPA SubT Challenge as detailed in [10]. This gives a total of three different sensor configurations (see Fig. 1 and Table I). Note that, even though the experiments were performed using the default leg configuration of ANYmal with inward pointing knees, VILENS is agnostic to the number and configuration of legs used.

B. Dataset

To evaluate our proposed algorithm, we have collected datasets in a variety of test environments lasting a total of 2 h and traveling 1.8 km. This includes data from the Urban Circuit of the DARPA SubT Challenge. The dataset is composed of the following experiments.

1) SMR: Swiss Military Rescue Facility, Wangen, CH. Trotting over concrete and gravel followed by loops on the grass with different gaits (ANYmal B300, 106 m, 13 m).
2) FSC: Fire Service College, Moreton-in-the-Marsh, U.K. Three loops of an outdoor industrial environment with

Fig. 8. Onboard camera images showing challenging environments from the experiments. *Top-Left:* Wet and oily concrete with reflections and lack of features at the horizon (FSC); *Top-Right:* Complete darkness (image manually enhanced in postprocessing). *Bottom-Left:* Low light, overexposure, and dust particles (LSM); *Bottom-Right:* Long straight corridors and low light (SMM).

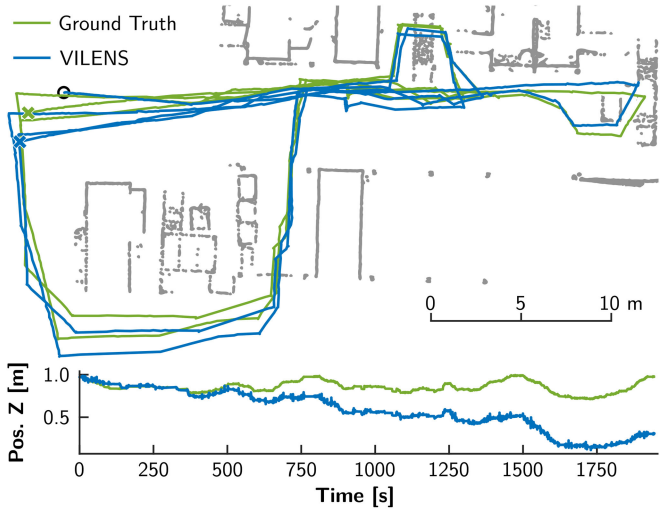

Fig. 9. Aerial view of the estimated and ground truth trajectories on the FSC experiment (240 m traveled). The start of the trajectory is marked with a circle. The ending location is marked with a cross. Note that since VILENS is an odometry system, no loop closures have been performed.

standing water, oil residue, gravel, and mud (ANYmal B300, 240 m, 34 m).

3) SUB: DARPA SubT Urban Beta course, Satsop, WA, USA.[4] Autonomous exploration of a dark underground inactive nuclear powerplant (ANYmal B300, 490 m, 60 m).
4) LSM: A decommissioned limestone mine, Wiltshire, U.K. Teleoperated exploration of an unlit mine with several loops (ANYmal C100, 474 m, 20 m).
5) SMM: Seemühle mine, CH. Autonomous exploration with Cerberus SubT exploration system (ANYmal C100, 522 m, 17 m).

Images from the onboard cameras in each of the experiments are shown in Fig. 8, which illustrate various challenges including slippery ground (affecting kinematics), reflections and darkness (affecting vision), and long corridors (affecting lidar point cloud registration). Different copies of the robots were used in each experiment.

In the SMR and FSC experiments, ground truth was generated by tracking the robot using a Leica TS16 laser tracker and then estimating its orientation using a optimization-based method (as described in [15]). In LSM, SUB, and SMM experiments, an ICP-based method was to align the current lidar scan to an accurate prior map made with survey-grade lidar scanners (as described in [45]).

C. Multisensor Fusion Comparison

To evaluate the performance of the system we compute the mean relative pose error (RPE) over distances of 10 m traveled for different module combinations of VILENS (at 15 Hz). We use a letter to indicate the sensor modalities used as follows: lidar features (L), visual features (V), IMU preintegration (I), preintegrated leg kinematics (K), and ICP registration (R).

The set of combinations tested are as follows:

1) VILENS-LVI: Lidar and visual feature tracking with preintegrated IMU but no leg kinematics. This is the same configuration as [16].
2) VILENS-LVIK: Lidar and visual feature tracking, with preintegrated IMU and including leg kinematics.
3) VILENS-IR: Only ICP registration and preintegrated IMU. As a fair comparison, we use the IMU to undistort point clouds. In this configuration, the output is limited by the frequency of ICP and thus can only run at 2 Hz.
4) VILENS: Lidar and visual feature tracking, preintegrated IMU, leg kinematics, and ICP registration. This is the full algorithm proposed in this article. (Where no letters are appended to the word VILENS we mean this version.)

Note that the same settings have been used for all configurations, with just different modules activated for each experiment.

For comparison, we also tested CompSLAM [9], which is a loosely coupled filter combining lidar-inertial odometry (LOAM [49]), visual-inertial odometry (ROVIO [50]), and kinematic-inertial odometry (TSIF [2]). It uses heuristics to switch between modalities. LOAM is its primary modality and has a frequency of 5 Hz. CompSLAM was also tested and deployed on the ANYmal robots of the winning team of the DARPA SubT Challenge [10].

Figs. 9 and 10 illustrate the high-level performance of VILENS. It can be seen that the estimated trajectory closely matches the ground truth, showing the accuracy of this approach. Note that VILENS is an odometry system—No loop closures are performed in this system. However, VILENS can be integrated with an external SLAM system, such as in [51].

Quantitative results are summarized in Table II. Across the entire dataset, the best performing algorithm was the complete

[4]This is the same sequence used in [16], except here we additionally process leg kinematics.

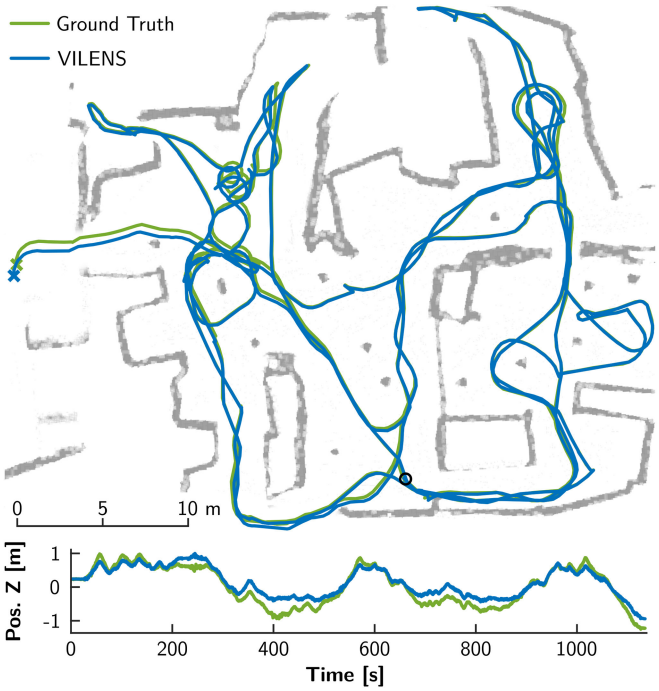

Fig. 10. Top-down view of the trajectory estimate by VILENS (blue) and the ground truth (green) on the 474-m LSM experiment. The start of the trajectory is indicated by a black circle, whereas the end is indicated by a cross. Note that since VILENS is an odometry system, no loop closures have been performed.

TABLE II
EXPERIMENTAL RESULTS

10 m RPE					
Translation μ (σ)[m]					
Data	C-S* [9]	V-LVI	V-LVIK	V-IR†	VILENS
SMR	0.28 (0.14)	0.14 (0.12)	0.15 (0.12)	0.24 (0.15)	**0.12** (0.11)
FSC	0.16 (0.08)	0.20 (0.09)	0.24 (0.17)	0.15 (0.09)	**0.15** (0.07)
SUB	0.20 (0.14)	0.11 (0.07)	0.11 (0.08)	0.10 (0.08)	**0.05** (0.03)
LSM	0.27 (0.15)	0.34 (0.30)	0.29 (0.33)	0.10 (0.08)	**0.04** (0.04)
SMM	0.36 (0.14)	0.74 (0.99)	0.74 (0.90)	0.27 (0.22)	**0.12** (0.08)
Mean	0.25 (0.13)	0.31 (0.31)	0.31 (0.32)	0.17 (0.12)	**0.10** (0.07)
Rotation μ (σ)[°]					
Data	C-S* [9]	V-LVI	V-LVIK	V-IR†	VILENS
SMR	3.14 (1.83)	**1.18** (0.97)	1.38 (1.05)	1.69 (0.93)	1.30 (1.07)
FSC	2.36 (1.07)	1.30 (0.90)	1.17 (0.94)	2.03 (0.86)	**1.14** (0.78)
SUB	0.99 (0.62)	0.75 (0.40)	0.74 (0.43)	1.88 (1.13)	**0.56** (0.34)
LSM	1.80 (0.94)	2.32 (1.73)	1.92 (1.51)	1.44 (0.71)	**0.59** (0.39)
SMM	1.42 (0.73)	4.73 (6.60)	4.38 (5.26)	2.73 (1.73)	**1.19** (0.61)
Mean	1.94 (1.04)	2.06 (2.12)	1.92 (1.84)	1.95 (1.07)	**0.96** (0.64)

"C-S" = CompSLAM, "V-" = VILENS-, *Output at 5 Hz, †Output at 2 Hz.
The boldface values represent the algorithm with the lowest RPE.

VILENS system, with an overall mean RPE error of just 0.96% in translation and 0.0956 deg/m in rotation.

Compared to the ICP and IMU only (VILENS-IR) solution VILENS provides a higher frequency and higher accuracy estimation, as well as robustness against degenerate scenarios for lidar, including long tunnels. The improvement in performance is due to the incorporation of different sensor modalities at a higher frequency. This created a smoother state estimate, which improved lidar motion correction, provided a better prior for ICP, and allowed for inaccurate ICP estimates to be rejected using robust cost functions.

Additionally, VILENS outperforms the loosely coupled approach, CompSLAM, by an average of 62% in translation and 51% in rotation.

Accurate, high-frequency state estimation is also a requirement for terrain mapping, as discussed in Section VI-F.

TABLE III
ABLATION STUDY OF ONLINE VELOCITY BIAS ESTIMATION

10-m RPE				
	Translation μ (σ)[m]		Rotation μ (σ)[°]	
Data	VILENS-NO-BIAS	VILENS	VILENS-NO-BIAS	VILENS
SMR	0.12 (0.12)	**0.12** (0.11)	1.30 (0.94)	**1.30** (1.07)
FSC	0.21 (0.15)	**0.15** (0.07)	1.24 (0.79)	**1.14** (0.78)
SUB	0.06 (0.05)	**0.05** (0.03)	0.64 (0.41)	**0.56** (0.34)
LSM	0.05 (0.04)	**0.04** (0.04)	0.64 (0.41)	**0.56** (0.34)
SMM	0.13 (0.09)	**0.12** (0.08)	**1.12** (0.62)	1.19 (0.61)
Mean	0.11 (0.09)	**0.10** (0.07)	1.02 (0.65)	**0.96** (0.64)

The boldface values represent the algorithm with the lowest RPE.

D. Twist Bias Ablation Study

In this section, we quantitatively evaluate the effect of the velocity bias estimation. As shown in Table III, even when using all available sensor modalities (including cameras and lidar), the biased signal from kinematics can still negatively affect the outcome. On average, adding online bias estimation reduces the RPE on average by 9.0% / 5.9% in translation/rotation. However, the effect of bias estimation is most obvious when exteroceptive sensors are degraded (see Section VII-B).

E. Internal Kinematics Comparison

In this section, we evaluate quantitatively the benefit of incorporating the kinematics inside the factor graph, rather than relying on an external filter. Fig. 11 shows a comparison between the estimated velocity when using internal kinematics and receiving the output from an external kinematic-inertial filter [2]. The ground truth was obtained by differentiation of the 200-Hz motion capture data. Note that in this configuration, we are only using kinematic-inertial inputs (no exteroceptive sensors) to highlight the difference between the two approaches. It can be seen that using internal kinematics improves the velocity tracking and, in turn, reduces position drift by 45%.

F. Integration With Local Planner and Perceptive Controller

This section will provide a qualitative demonstration of VILENS running onboard the ANYmal robot. The limited onboard computation is shared with other processes running on the robot, including control, terrain mapping, and sensor drivers.

Constructing an accurate local terrain map around the robot is crucial for perceptive locomotion and path planning. Locomotion controllers plan footstep placements on these maps, such as in [52].

The current approach for the ANYmal robot is to use the kinematic-inertial estimator, TSIF [2], to feed the local elevation mapping system [53]. TSIF can suffer from significant drift,

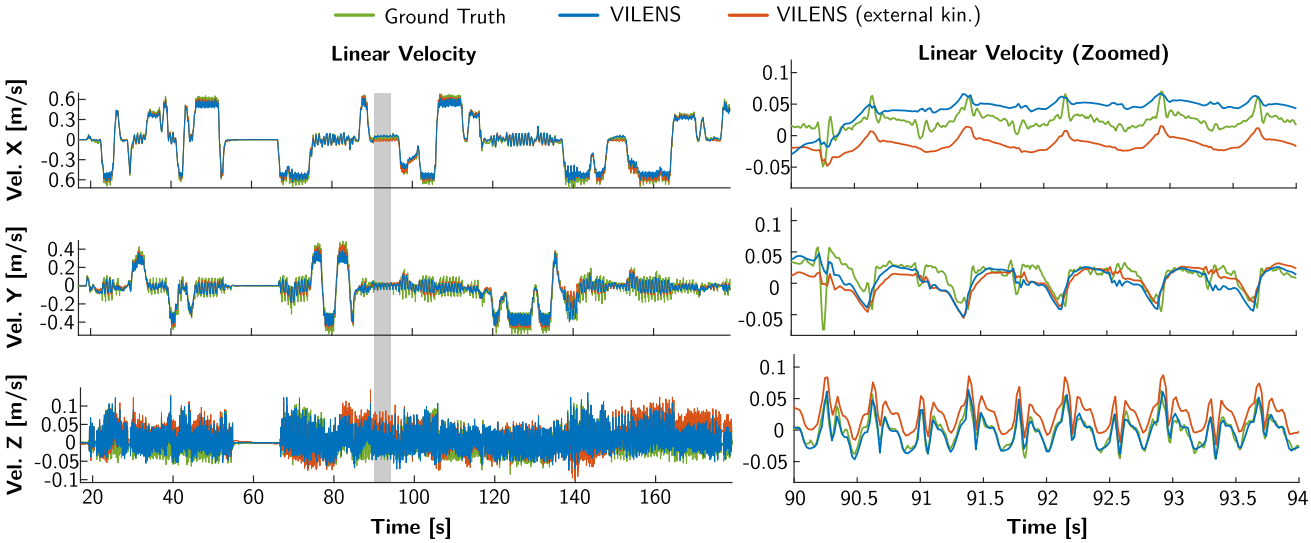

Fig. 11. *Left:* Linear velocity comparison between VILENS fusing IMU and leg kinematics using the external filter from TSIF [2] (orange) and fusing the kinematics inside the optimizer (blue) against ground truth (green). *Right:* Zoomed in plot corresponding to the gray area of the complete experiment. The better velocity tracking allows for a reduction in position drift.

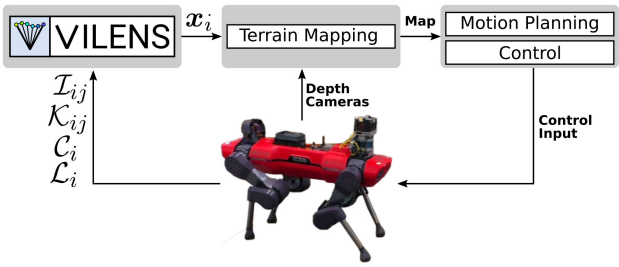

Fig. 12. *Terrain Mapping Pipeline:* The VILENS state estimator produces a high frequency (400 Hz), low drift state estimate, $\boldsymbol{x}$, for the terrain mapping module [53]. This local terrain map can then be used by other modules, such as perceptive motion planning and control [17], or local path planning for obstacle avoidance [18].

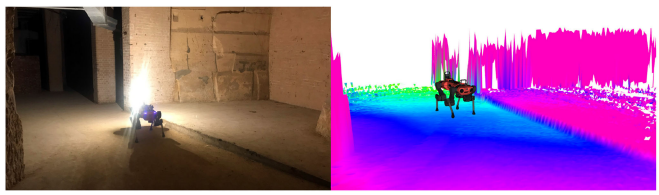

Fig. 13. Accurate terrain map generated with the VILENS state estimate can be fed into the local path planner [18] for autonomous exploration. The ledge near the robot is clearly defined in the elevation map, allowing the local planner to create routes on this obstacle.

which creates "phantom obstacles" in the terrain map. The effect of this is that the robot exhibits undesirable behavior such as poorly placed foot landings or planning suboptimal paths. In the worst case, the robot may fall and get irreversibly damaged or become stuck because the local path planner is unable to find a feasible solution.

Fig. 12 shows the pipeline used for the terrain mapping with VILENS. The inputs to the terrain mapping module [53] are the VILENS state estimate, and the point clouds from several downward-facing depth cameras on the robot's body. These cameras are not the same as the ones used for state estimation and are not triggered at the same time. Therefore, accurate terrain mapping depends on a smooth, accurate, and high-frequency state estimate to avoid interpolation errors.

From our experience, terrain mapping frequencies of $\geq$15 Hz are required for dynamic locomotion over rough terrain. This means that registration-based algorithms with a low-frequency output (such as ICP or LOAM) are not suitable for this purpose.

1) Terrain Map Integration 1: Local Path Planning: In the first experiment, we evaluate the quality of online local terrain mapping for local path planning. During the LSM experiment the terrain map was successfully used in the loop by a local path planning algorithm, GBPlanner [18], for over 30 m without failure. Fig. 13 shows an example of the accuracy of the terrain map compared to the ground truth.

2) Terrain Map Integration 2: Perceptive Control: In the second experiment, we demonstrate terrain mapping for perceptive control using RLOC [17] during a 5-m outdoor experiment over grass, gravel, curbs, and slopes. Fig. 14 shows the accuracy of the terrain map, which allowed the controller to plan and execute precise steps on the terrain. By reducing the amount of drift in the state estimate, the quality of terrain map improves, allowing for more accurate motion planning. Future work could involve tighter integration of local terrain mapping and contact point locations estimated by VILENS.

VII. Discussion

A. Evolution of Velocity Bias

In Fig. 15, we compare the actual and the estimated velocity error from kinematics in the z-axis, which is where most of the drift occurs for this particular sequence due to the presence of gravel. The orange line indicates the error between the true robot

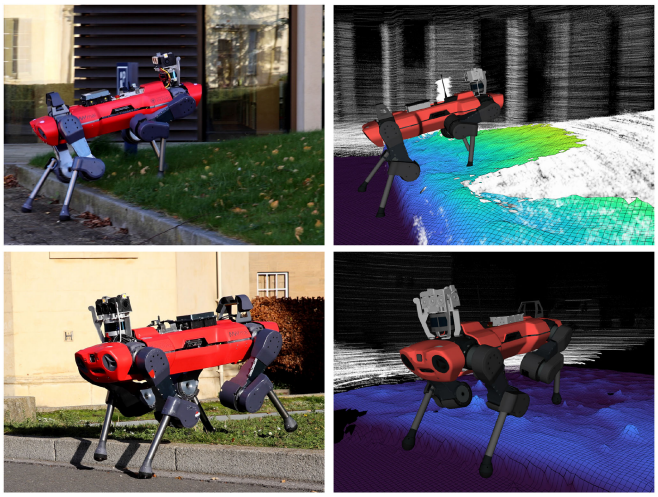

Fig. 14. ANYmal robot walking over curbs, slopes, and other rough terrain using the RLOC perceptive controller [17]. The accurate terrain map produced by using the VILENS estimate enabled the successful execution of the trajectory.

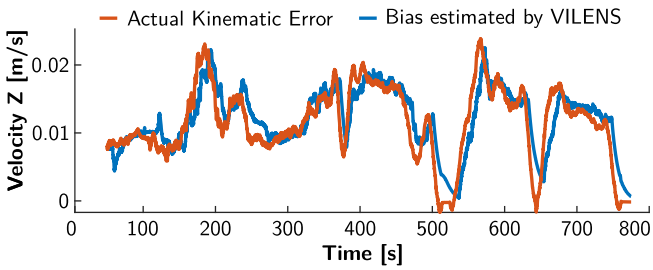

Fig. 15. Comparison between the actual velocity error (from the kinematic odometry in the z-axis inferred using a Leica tracker) and the bias estimated by VILENS during the SMR experiment. Using exteroceptive sensing, VILENS is able to accurately and stably track this effect by modeling the bias term $\mathbf{b}^v$.

velocity and the one perceived by the leg odometry, whereas the blue line shows the VILENS estimate of the same error, which is modeled as a velocity bias term of the leg odometry factor $\mathbf{b}^v$. The sequence analyzed is the same as the one shown in Fig. 2. The high degree of correlation between the two signals demonstrates the effectiveness of leg odometry velocity bias estimation.

B. Performance in Underconstrained Environments

In contrast to other recent loosely-coupled approaches to multisensor state estimation [9], [54], VILENS naturally handles degenerate scenarios without requiring hard switches which are typically hand engineered.

An example is shown in Fig. 16, where the ANYmal robot walks very close to a wall in a long, straight tunnel. At time 843 s, the robot turns toward the straight part of the tunnel. Due to the degenerate geometry, the number of lidar features gradually drops as the robot goes deeper into the tunnel (maroon line, bottom plot). While the lidar feature tracking can still provide partial constraints to the system, ICP is unstable and close to divergence, so it was not included in the analysis. At the same time, the scene changes abruptly causing a period of underexposure (picture at the top), with a dramatic drop

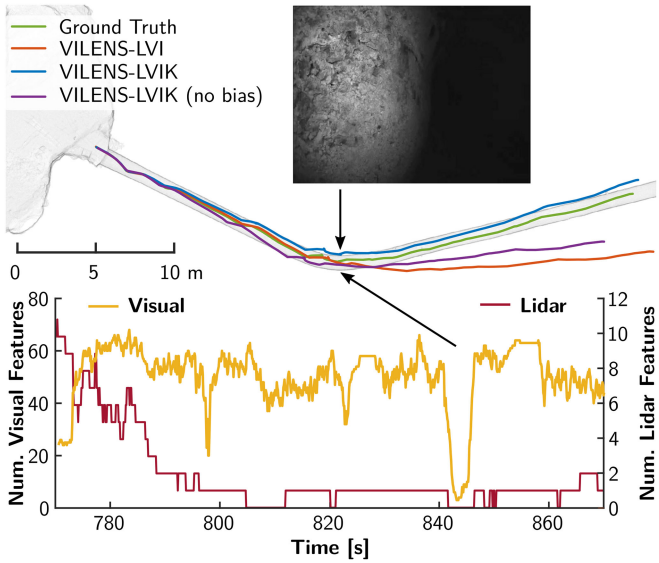

Fig. 16. *Top:* Comparison between the trajectory estimates for different VILENS configurations and ground truth (in green) for the tunnel sequence of the SMM experiment. The lack of visual features causes a large rotation error when not using leg odometry (orange line). Adding leg odometry improves the estimate but still accumulates drift (purple line). With the online velocity bias estimation, the drift is reduced even further (blue line). *Bottom:* Number of lidar (in red) and visual (in yellow) features tracked over time. Note that the lidar features plot has been slightly smoothed over a 1-s window for clarity.

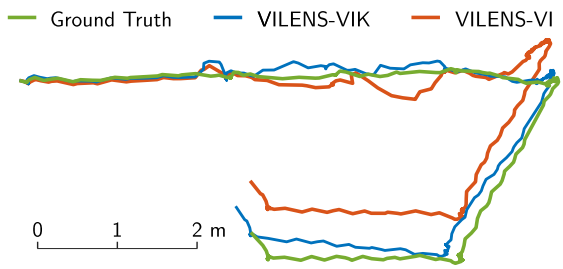

Fig. 17. Top-down comparison of VILENS-VIK and VILENS-VI trajectories aligned with ground truth while crossing the puddle shown in Fig. 8 Top-Left.

in the number of visual features (yellow line, bottom plot). Because of this, the VILENS-LVI configuration (i.e., VILENS without leg odometry and ICP, as in [16]) accumulates a large angular error (orange line). Instead, the addition of leg odometry factor improves the estimate (VILENS-LVIK, blue). To show the benefit of velocity bias estimation, we also show the same VILENS-LVIK configuration but without online bias estimation. Because the terrain is loose, the drift is accumulated faster (purple line).

C. Leg Odometry in Visually Degraded Scenarios

Fig. 17 shows a situation from FSC experiment where the robot crosses a large puddle. VILENS-VI tracks the features on the standing water, causing drift on the xy-plane. Instead, VILENS-VIK maintains a better pose estimate by relying on leg odometry with the bias estimation.

TABLE IV
ANALYSIS OF TIMING FOR DIFFERENT PARTS OF VILENS

Module	Timing $\mu(\sigma)$ [ms]	Freq. [Hz]
IMU	0.05 (0.12)	400
Leg kinematics	0.07 (0.30)	400
Visual features with lidar depth	9.48 (7.69)	10
Lidar point cloud features	19.72 (6.24)	10
Lidar ICP	149.75 (59.23)	2
Optimization	8.65 (3.25)	10

TABLE V
FREQUENCY AND LATENCY OF OUTPUTS. LATENCY IS COMPARED TO THE LOWEST LATENCY INPUT SIGNAL (IMU)

Output type	Mean latency [ms]	Freq. [Hz]
IMU forward-propagated	2.3	400
Factor graph optimized	95.2	10
ICP optimized	395.2	2

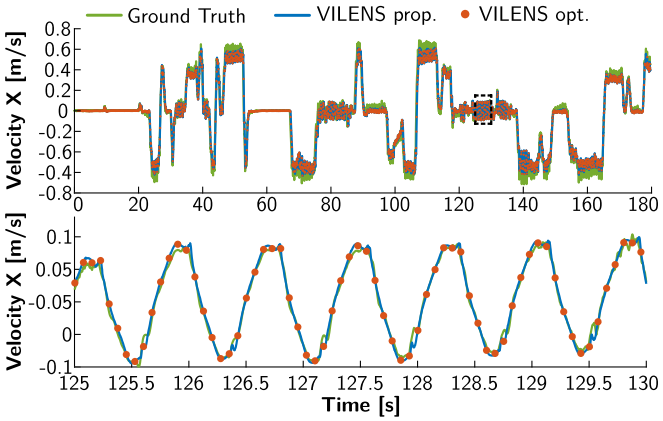

Fig. 18. *Top:* Comparison between the optimized (10 Hz, orange dots) and IMU forward propagated (400 Hz, blue line) velocity estimates against the ground truth from motion capture (green line). *Bottom:* Zoomed-in view corresponding to the dashed box at time interval 125–130 s. The close tracking of the ground truth shows that IMU forward propagation provides an accurate, high-frequency, and low-latency prediction of the state between optimizations.

D. Timing Analysis

An important consideration in a multisensor fusion algorithm is computation time, which generally increases with the number of input sensors.

Table IV shows a summary of the computation time for the components of VILENS. The timing tests were performed on a laptop equipped with an Intel E-2186 M processor (6 cores/12 threads @ 2.9 GHz base frequency) and 16 GB of RAM. A key benefit of using this type of factor-graph based approach is that the lightweight visual and point cloud features allow for accurate, low latency state estimation at relatively low computational expense. This also allows more expensive modalities, such as ICP, to run at a lower frequency. This saves computation and preserves the benefits of low-drift ICP state estimation.

Table V highlights the three different types of outputs from the VILENS system, with differing levels of latency and accuracy for different purposes. Fig. 18 shows how the high-frequency forward propagated estimate closely matches Vicon ground truth. This is important in some applications to capture the high-frequency behavior of the robot between optimizations.

VIII. CONCLUSION

This article presented VILENS, a multisensor fusion algorithm that can seamlessly fuse inertial, legged, lidar, and visual sensing within the same factor graph. The tight fusion of all these sensor modalities allows the algorithm to overcome the challenging operational conditions of autonomous navigation in underground environments as well as large open areas. In these conditions, individual sensor modalities such as visual-inertial or lidar-inertial odometry would fail.

A particular contribution of the article was the integration of the leg kinematic measurements as a dedicated residual for the factor graph, rather than relying on an external filter. This enables tighter integration and better noise modeling. The leg odometry drift, typically occurring due to leg compression at the impact, was modeled as a bias term of the velocity measurement and was proven to be effective in challenging episodes where both the lidar and camera sensors are deprived. In these situations, our system falls back to fusing IMU and leg odometry with no hard switching required.

We demonstrated the robustness of our method in field experiments with several quadruped platforms in challenging scenarios including slippery and deformable terrain, water reflections, complete darkness, and degenerate geometries, such as long corridors lasting longer than 2 h.

The high frequency as well as the smoothness of the output was further demonstrated by running the algorithm online and onboard the robot to enable local elevation mapping, which was used by a perceptive controller [17] and a local path planner [18] to cross an obstacle course and to autonomously explore a mine, respectively.

APPENDIX
PREINTEGRATED LEG ODOMETRY FACTOR

In this Appendix, we provide a derivation of the measurements and covariance for the preintegrated leg odometry factor described in Section IV-B. Without loss of generality, we will focus on cases when only one leg is in contact with the ground. As explained in Section IV-B2, when there are multiple stance legs, the compound measurement and noise can be computed independently from each stance leg and the fused via a weighted average.

A. Velocity Measurement and Noise

When a leg is in rigid, nonslipping contact with the ground, the robot's linear velocity can be computed from the position and velocity of the feet

$$\mathbf{v} = -\mathbf{J}_p(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}} - \boldsymbol{\omega} \times \mathbf{f}_p(\boldsymbol{\alpha}) \tag{48}$$

where $\mathbf{f}_p(\cdot)$ is the forward kinematics function and $\mathbf{J}_p(\cdot)$ is its Jacobian. To compute a velocity measurement for the factor graph, we need to linearly separate the noise from the rest of (48). Remembering from (17) to (18) that the joint states are corrupted

by zero-mean Gaussian noise, such that $\boldsymbol{\alpha} = \tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha}$, $\dot{\boldsymbol{\alpha}} = \dot{\tilde{\boldsymbol{\alpha}}} - \boldsymbol{\eta}^{\dot{\alpha}}$, we first start by separating the noise from the differential kinematics term

$$\begin{aligned}\mathbf{J}_p(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}} &= \mathbf{J}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha})(\dot{\tilde{\boldsymbol{\alpha}}} - \boldsymbol{\eta}^{\dot{\alpha}}) \\ &= \mathbf{J}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha})\dot{\tilde{\boldsymbol{\alpha}}} - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha})\boldsymbol{\eta}^{\dot{\alpha}} \\ &\simeq \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} - \frac{\partial}{\partial \tilde{\boldsymbol{\alpha}}}\left(\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}}\right)\boldsymbol{\eta}^{\alpha} - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}} \quad (49)\end{aligned}$$

where we applied the Taylor expansion of the product $\mathbf{J}_p(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}}$, ignoring the second-order terms. Since $\boldsymbol{\alpha}$ and $\dot{\boldsymbol{\alpha}}$ are independent, $\dot{\boldsymbol{\alpha}}$ can be taken out of the derivative in (49)

$$\frac{\partial}{\partial \tilde{\boldsymbol{\alpha}}}\left(\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}}\right)\boldsymbol{\eta}^{\alpha} = \frac{\partial}{\partial \tilde{\boldsymbol{\alpha}}}\left(\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\right)\dot{\tilde{\boldsymbol{\alpha}}}\boldsymbol{\eta}^{\alpha} = \mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}}\boldsymbol{\eta}^{\alpha} \quad (50)$$

where $\mathbf{H}_p(\cdot) \in \mathbb{R}^{3\times 3\times 3}$ is the Hessian of the forward kinematics function $\mathbf{f}_p(\boldsymbol{\alpha})$. Note that, since the forward kinematics is a vector function, the Hessian becomes a tensor of rank 3, so the product with $\mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} = \mathbf{H}_p(\tilde{\boldsymbol{\alpha}}) \bar{\times}_2 \dot{\tilde{\boldsymbol{\alpha}}}$ is a two-mode vector product [55]. In our implementation, the calculation of (50) is performed numerically.

Substituting (49)–(50) into (48) leads to

$$\begin{aligned}\mathbf{v} = &-\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} - \mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}}\boldsymbol{\eta}^{\alpha} - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}} \\ &- \boldsymbol{\omega} \times \mathbf{f}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha}). \quad (51)\end{aligned}$$

It has been demonstrated in [7] that the noise from the last term of (51) can be separated as follows:

$$\mathbf{f}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha}) \approx \mathbf{f}_p(\tilde{\boldsymbol{\alpha}}) - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\alpha}. \quad (52)$$

From the above relations, (48) becomes

$$\begin{aligned}\mathbf{v} &= -(\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} - \mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}}\boldsymbol{\eta}^{\alpha} - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}}) \\ &\quad - \boldsymbol{\omega}^{\wedge}\mathbf{f}_p(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\eta}^{\alpha}) \\ &= -\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} + \mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}}\boldsymbol{\eta}^{\alpha} + \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}} \\ &\quad - \boldsymbol{\omega}^{\wedge}(\mathbf{f}_p(\tilde{\boldsymbol{\alpha}}) - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\alpha}) \\ &= -\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} + \mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}}\boldsymbol{\eta}^{\alpha} + \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}} \\ &\quad - \boldsymbol{\omega}^{\wedge}\mathbf{f}_p(\tilde{\boldsymbol{\alpha}}) + \boldsymbol{\omega}^{\wedge}\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\alpha} \\ &= \underbrace{-\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} - \boldsymbol{\omega}^{\wedge}\mathbf{f}_p(\tilde{\boldsymbol{\alpha}})}_{\tilde{\mathbf{v}}} \\ &\quad - \underbrace{\left(-\left(\mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} + \boldsymbol{\omega}^{\wedge}\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\right)\boldsymbol{\eta}^{\alpha} - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}}\right)}_{\boldsymbol{\eta}^{v}} \\ &\triangleq \tilde{\mathbf{v}} - \boldsymbol{\eta}^{v}. \quad (53)\end{aligned}$$

Equation (53) still depends on the rotational velocity, which is measured by the IMU and is affected by measurement noise and bias. If we replace the angular velocity with the measured values, we have

$$\begin{aligned}\mathbf{v} = &-\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} - (\tilde{\boldsymbol{\omega}} - \mathbf{b}^{\omega} - \boldsymbol{\eta}^{\omega})^{\wedge}\mathbf{f}_p(\tilde{\boldsymbol{\alpha}}) \\ &+ \left(\mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} + (\tilde{\boldsymbol{\omega}} - \mathbf{b}^{\omega} - \boldsymbol{\eta}^{\omega})^{\wedge}\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\right)\boldsymbol{\eta}^{\alpha} + \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{v}. \quad (54)\end{aligned}$$

By applying the inversion rule for the cross product and the distributive property, we have

$$\begin{aligned}\mathbf{v} = &-\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} + \mathbf{f}_p(\tilde{\boldsymbol{\alpha}})^{\wedge}\tilde{\boldsymbol{\omega}} - \mathbf{f}_p(\tilde{\boldsymbol{\alpha}})^{\wedge}\mathbf{b}^{\omega} - \mathbf{f}_p(\tilde{\boldsymbol{\alpha}})^{\wedge}\boldsymbol{\eta}^{\omega} \\ &+ \left(\mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} - \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})^{\wedge}\tilde{\boldsymbol{\omega}} + \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})^{\wedge}\mathbf{b}^{\omega}\right. \\ &+ \left.\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})^{\wedge}\boldsymbol{\eta}^{\omega}\right)\boldsymbol{\eta}^{\alpha} + \mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}}. \quad (55)\end{aligned}$$

After rearrangement and removal of the second-order terms, the noise can be again separated as follows:

$$\begin{aligned}\mathbf{v} \simeq &\underbrace{-\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} - \tilde{\boldsymbol{\omega}}^{\wedge}\mathbf{f}_p(\tilde{\boldsymbol{\alpha}}) + \mathbf{b}^{\omega\wedge}\mathbf{f}_p(\tilde{\boldsymbol{\alpha}})}_{\tilde{\mathbf{v}}} \\ &\underbrace{-\left(-\left(\mathbf{H}_p(\tilde{\boldsymbol{\alpha}})\dot{\tilde{\boldsymbol{\alpha}}} + \tilde{\boldsymbol{\omega}}^{\wedge}\mathbf{J}_p(\tilde{\boldsymbol{\alpha}}) - \mathbf{b}^{\omega\wedge}\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\right)\boldsymbol{\eta}^{\alpha}\right.}_{} \\ &\underbrace{\left.-\mathbf{J}_p(\tilde{\boldsymbol{\alpha}})\boldsymbol{\eta}^{\dot{\alpha}} + \mathbf{f}_p(\tilde{\boldsymbol{\alpha}})^{\wedge}\boldsymbol{\eta}^{\omega}\right)}_{\boldsymbol{\eta}^{v}} \\ \simeq &\ \tilde{\mathbf{v}} - \boldsymbol{\eta}^{v}. \quad (56)\end{aligned}$$

B. Iterative Noise Propagation

By substitution of $\boldsymbol{\eta}^{v}$ from (56) into (34), we can express the preintegrated noises $\delta\boldsymbol{\theta}, \delta\boldsymbol{\kappa}$ in an iterative form

$$\begin{aligned}\begin{bmatrix}\delta\boldsymbol{\theta}_{i,k+1} \\ \delta\boldsymbol{\kappa}_{i,k+1}\end{bmatrix} = &\underbrace{\begin{bmatrix}\Delta\tilde{\boldsymbol{\Theta}}_{k,k+1}^{\mathsf{T}} & \mathbf{0} \\ -\Delta\tilde{\boldsymbol{\Theta}}_{i,k}(\tilde{\mathbf{v}}_k - \mathbf{b}_i^{v})^{\wedge}\Delta t\ \mathbf{I}\end{bmatrix}}_{\mathbf{A}}\begin{bmatrix}\delta\boldsymbol{\theta}_{i,k} \\ \delta\boldsymbol{\kappa}_{i,k}\end{bmatrix} \\ &+ \underbrace{\begin{bmatrix}\mathbf{J}_R^k\Delta t & \mathbf{0} & \mathbf{0} \\ -\mathbf{f}_p(\tilde{\boldsymbol{\alpha}}_k)^{\wedge} & \boldsymbol{\chi}_k & \mathbf{J}_p(\tilde{\boldsymbol{\alpha}}_k)\end{bmatrix}}_{\mathbf{B}}\begin{bmatrix}\boldsymbol{\eta}^{\omega} \\ \boldsymbol{\eta}^{\alpha} \\ \boldsymbol{\eta}^{\dot{\alpha}}\end{bmatrix} \quad (57)\end{aligned}$$

where

$$\boldsymbol{\chi}_k = \mathbf{H}_p(\tilde{\boldsymbol{\alpha}}_k)\dot{\tilde{\boldsymbol{\alpha}}}_k + \tilde{\boldsymbol{\omega}}_k^{\wedge}\mathbf{f}_p(\tilde{\boldsymbol{\alpha}}_k) - (\mathbf{b}^{\omega})^{\wedge}\mathbf{J}_p(\tilde{\boldsymbol{\alpha}}_k). \quad (58)$$

Equation (57) can be expressed more compactly as

$$\boldsymbol{\eta}_{i,k+1}^{\Delta} = \mathbf{A}\boldsymbol{\eta}_{i,k}^{\Delta} + \mathbf{B}\boldsymbol{\eta}_k \quad (59)$$

where $\boldsymbol{\eta}_k = [\boldsymbol{\eta}^{\omega}\ \boldsymbol{\eta}^{\alpha}\ \boldsymbol{\eta}^{\dot{\alpha}}]^{\mathsf{T}}$.

From the linear expression (59) and given the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\eta}} \in \mathbb{R}^{9\times 9}$ of the raw gyro and joint states noises $\boldsymbol{\eta}_k$, the covariance for the factor can be computed iteratively

$$\boldsymbol{\Sigma}_{i,k+1} = \mathbf{A}\boldsymbol{\Sigma}_{i,k}\mathbf{A}^{\mathsf{T}} + \mathbf{B}\boldsymbol{\Sigma}_{\boldsymbol{\eta}}\mathbf{B}^{\mathsf{T}} \quad (60)$$

starting from the initial condition $\boldsymbol{\Sigma}_{i,i} = \mathbf{0}$.

C. Observability of the Linear Velocity Bias

In (26), we have introduced the velocity bias term $\mathbf{b}^{v}$ to relax the nonslip relation from (16). Intuitively, $\mathbf{b}^{v}$ represents the velocity of the contact point. If we rewrite (26) and ignore Gaussian noise terms, the kinematic velocity bias $\mathbf{b}^{v}$ can be expressed as

$$\mathbf{b}^{v} = \tilde{\mathbf{v}}_{\text{kin}} - \mathbf{v} \quad (61)$$

where $\tilde{\mathbf{v}}_{\text{kin}}$ is the estimated velocity from kinematics and $\mathbf{v}$ is the true velocity.

Bloesch et al. [1] proved that, if the contact point is stationary (i.e., $\mathbf{b}^v = 0$), absolute position and yaw are the only unobservable states (excluding degenerate cases). Therefore, kinematic-inertial measurements alone can only estimate $\tilde{\mathbf{v}}_{\text{kin}}$. The same unobservable states have been proved by Hesch et al. [47] for a typical visual-inertial system. Therefore, the robot's linear velocity $\mathbf{v}$ can be observed from IMU and camera measurements only.

Since VILENS combines kinematic, inertial, and camera measurements, we speculate that the linear velocity bias $\mathbf{b}^v$ should be observable, as confirmed empirically in our experiments. An analytic proof of observability and convergence is left to future work.

ACKNOWLEDGMENT

This research has been conducted as part of the ANYmal research community. Special thanks to the CERBERUS DARPA SubT Team for providing the data from the challenge runs and the hardware support.

REFERENCES

[1] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, "State estimation for legged robots on unstable and slippery terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 6058–6064.

[2] M. Bloesch, M. Burri, H. Sommer, R. Siegwart, and M. Hutter, "The two-state implicit filter recursive estimation for mobile robots," *IEEE Robot. Automat. Lett.*, vol. 3, no. 1, pp. 573–580, Jan. 2018.

[3] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended Kalman filtering for robot state estimation," *Int. J. Robot. Res.*, vol. 39, no. 4, pp. 402–430, 2020.

[4] M. Camurri et al., "Probabilistic contact estimation and impact detection for state estimation of quadruped robots," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 1023–1030, Apr. 2017.

[5] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4170–4176, Oct. 2019.

[6] J.-H. Kim et al., "Legged robot state estimation with dynamic contact event information," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6733–6740, Oct. 2021.

[7] R. Hartley, M. G. Jadidi, L. Gan, J. Huang, J. W. Grizzle, and R. M. Eustice, "Hybrid contact preintegration for visual-inertial-contact state estimation using factor graphs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3783–3790.

[8] M. Camurri, M. Ramezani, S. Nobili, and M. Fallon, "Pronto: A multi-sensor state estimator for legged robots in real-world scenarios," *Front. Robot. AI*, vol. 7, pp. 1–18, 2020.

[9] S. Khattak, H. Nguyen, F. Mascarich, T. Dang, and K. Alexis, "Complementary multi-modal sensor fusion for resilient robot pose estimation in subterranean environments," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2020, pp. 1024–1029.

[10] M. Tranzatto et al., "CERBERUS: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA subterranean challenge," *Field Robot.*, vol. 2, pp. 274–324, 2022.

[11] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations Trends Robot.*, vol. 6, pp. 1–139, 2017.

[12] S. Fahmi, G. Fink, and C. Semini, "On state estimation for legged locomotion over soft terrain," *IEEE Sens. Lett.*, vol. 5, no. 1, pp. 1–4, Jan. 2021.

[13] T. Koolen et al., "Design of a momentum-based control framework and application to the humanoid robot atlas," *Int. J. Humanoid Robot.*, vol. 13, no. 1, pp. 1–34, 2016.

[14] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[15] D. Wisth, M. Camurri, and M. Fallon, "Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 392–398.

[16] D. Wisth, M. Camurri, S. Das, and M. Fallon, "Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1004–1011, Apr. 2021.

[17] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Trans. Robot.*, pp. 1–20, 2022, doi: 10.1109/TRO.2022.3172469.

[18] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *J. Field Robot.*, vol. 37, no. 8, pp. 1363–1388, 2020.

[19] T. D. Barfoot, *State Estimation for Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[20] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 3565–3572.

[21] R. Hartley et al., "Legged robot state-estimation through combined forward kinematic and preintegrated contact factors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4422–4429.

[22] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 15–22.

[23] M. Fourmy, T. Flayols, N. Mansard, and J. Solà, "Contact forces pre-integration for the whole body estimation of legged robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 1372–1378.

[24] F. Grimminger et al., "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3650–3657, Apr. 2020.

[25] M. Xie, A. Escontrela, and F. Dellaert, "A factor-graph approach for optimization problems with dynamics constraints," 2020, *arXiv:2011.06194*.

[26] D. Wisth, M. Camurri, and M. Fallon, "Robust legged robot state estimation using factor graph optimization," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4507–4514, Oct. 2019.

[27] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "ISAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, 2012.

[28] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter, "Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3872–3878.

[29] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim, "Contact model fusion for event-based locomotion in unstructured terrains," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4399–4406.

[30] T.-Y. Lin, R. Zhang, J. Yu, and M. Ghaffari, "Legged robot state estimation using invariant Kalman filtering and learned contact events," in *Proc. Conf. Robot Learn.*, 2022, pp. 1057–1066.

[31] P. Furgale, "Representing robot pose: The good, the bad, and the ugly," 2014. [Online]. Available: http://paulfurgale.info/news/2014/6/9/representing-robot-pose-the-good-the-bad-and-the-ugly

[32] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Germany: Springer, 2008.

[33] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Trans. Robot.*, vol. 30, no. 3, pp. 679–693, Jun. 2014.

[34] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1335–1340, Aug. 2006.

[35] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 801–808.

[36] C. J. Taylor and D. J. Kriegman, "Minimization on the lie group SO(3) and related manifolds," Yale Univ., New Haven, CT, *Tech. Rep. 9405*, 1994.

[37] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4605–4611.

[38] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[39] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Auton. Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[40] K. MacTavish and T. D. Barfoot, "At all costs: A comparison of robust cost functions for camera correspondence outliers," in *Proc. Conf. Comput. Robot Vis.*, 2015, pp. 62–69.

[41] M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "RobCoGen: A code generator for efficient kinematics and dynamics of articulated robots, based on domain specific languages," *J. Softw. Eng. Robot.*, vol. 7, no. 1, pp. 36–54, 2016.
[42] S. M. Neuman, T. Koolen, J. Drean, J. E. Miller, and S. Devadas, "Benchmarking and workload analysis of robot dynamics algorithms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 5235–5242.
[43] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[44] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1280–1286.
[45] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The newer college dataset: Handheld lidar, inertial and vision with ground truth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4353–4360.
[46] A. Martinelli, "Closed-form solution of visual-inertial structure from motion," *Int. J. Comput. Vis.*, vol. 106, no. 2, pp. 138–152, Jan. 2014.
[47] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Camera-IMU-based localization: Observability analysis and consistency improvement," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 182–201, 2014.
[48] M. Hutter et al., "ANYmal – A highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 38–44.
[49] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Auton. Robots*, vol. 41, no. 2, pp. 401–416, 2017.
[50] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1053–1072, 2017.
[51] M. Ramezani, G. Tinchev, E. Iuganov, and M. Fallon, "Online lidar-slam for legged robots with robust registration and deep-learned loop closure," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4158–4164.
[52] O. Melon, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9805–9811.
[53] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3019–3026, Oct. 2018.
[54] M. Palieri et al., "LOCUS: A multi-sensor lidar-centric solution for high-precision odometry and 3D mapping in real-time," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 421–428, Apr. 2021.
[55] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

David Wisth (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering and the M.Eng. degree in mechatronics engineering from the University of Melbourne, Parkville, VIC, Australia, in 2012 and 2014, respectively, and the D.Phil. degree in engineering science from the University of Oxford, Oxford, U.K., in 2022.

From 2015 to 2017, he was a Project Engineer with Siemens, Australia. His research interests include multisensor fusion, factor graphs, and legged robots.

Dr. Wisth's awards and honors include graduating as valedictorian from his M.Eng. degree, and Best Student Paper Award (Finalist) at the IEEE International Conference on Robotics and Automation in 2021.

Marco Camurri (Member, IEEE) received the B.Eng. and M.Eng. degrees in computer engineering from the University of Modena and Reggio Emilia, Modena, Italy, in 2009 and 2012, respectively, and the Ph.D. degree in advanced robotics from the Istituto Italiano di Tecnologia (IIT), Genoa, Italy, in 2017.

From 2017 to 2020, he was a Postdoc with IIT for one year and the University of Oxford for two years. Since 2021, he has been a Senior Research Associate with Oxford Robotics Institute, University of Oxford, Oxford, U.K. His research interests include legged state estimation, mobile robot perception, sensor fusion, mapping, and mobile autonomous navigation.

Maurice Fallon (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from University College Dublin, Dublin, Ireland, in 2004 and the Ph.D. degree in acoustic source tracking from the University of Cambridge, Cambridge, U.K., in 2008.

From 2008 to 2012, he was a Postdoc and a Research Scientist with MIT Marine Robotics Group working on SLAM. Later, he was the Perception Lead of MIT's team in the DARPA Robotics Challenge. Since 2017, he has been a Royal Society University Research Fellow and an Associate Professor with the University of Oxford, Oxford, U.K. He leads the Dynamic Robot Systems Group, Oxford Robotics Institute. His research interests include probabilistic methods for localization, mapping, dynamic motion planning, and multisensor fusion.