2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Visual-Inertial Odometry with Point and Line Features

Yulin Yang, Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang

Abstract— **In this paper, we present a tightly-coupled monocular visual-inertial navigation system (VINS) using points and lines with degenerate motion analysis for 3D line triangulation. Based on line segment measurements from images, we propose two sliding window based 3D line triangulation algorithms and compare their performance. Analysis of the proposed algorithms reveals 3 degenerate camera motions that cause triangulation failures. Both geometrical interpretation and Monte-Carlo simulations are provided to verify these degenerate motions which prevent triangulation. In addition, commonly used line representations are compared through a monocular visual SLAM Monte-Carlo simulation. Finally, real-world experiments are conducted to validate the implementation of the proposed VINS system using the "closest point" line representation.**

I. Introduction and Related Work

Over the past decade, visual-inertial navigation systems (VINS) have seen a great increase in popularity due to their ability to provide accurate localization solutions while utilizing only low-cost inertial measurement units (IMUs) and cameras. The affordability, size, and light-weight nature of these sensors make them ideal for deployments in a wide-range of applications such as unmanned aerial vehicles (UAVs) [1] and mobile devices [2].

When fusing camera and IMU data, a key question is how to best to utilize the rich amount of information available in the images. In particular, most VINS can be categorized by whether they use *indirect* or *direct* image processing techniques. Direct visual methods provide motion estimates by minimizing costs involving the raw pixel intensities captured by the camera [3, 4]. By contrast, indirect systems typically extract geometric features from the pixels and track their motion across the image plane with the most common being points [5–7] and/or lines [8–13]. Point features correspond to 3D positions in the space that are detected as "corners" on the image plane. Lines, which are most commonly seen in human-built environments, are detected as straight edges in the image. Quite a few navigation works have been devoted to utilizing both geometric features to achieve robust and accurate estimation.

In particular, leveraging the multi-state constraint Kalman filter (MSCKF) framework [5], Kottas et al. [8] proposed the use of a unit quaternion with distance scalar to model a line and incorporate line features into visual-inertial odometry (VIO). In addition, they provided an observability analysis showing that VINS with a single line feature suffers from 5 unobservable directions. Later on, Kottas et al. [9] and Guo et al. [14] proposed the utilization of line constraints (e.g., parallel lines and vertical lines) to improve the estimation accuracy in structured environments such as indoor or Manhattan world, where all lines lay along the 3 major directions. Yu et al. [11] designed a VINS system with line features suitable for rolling-shutter cameras. Utilizing the Plücker representation and orthonormal error states [15] for lines, Heo et al. [12] implemented an invariant MSCKF with point and line features using a Lie group representation of the state. Zheng et al. [13] proposed representing a line with two 3D endpoints and designed a stereo visual-inertial navigation system leveraging both points and lines. He et al. [10] used IMU preintegration [16] to design a batch-optimization based visual-inertial SLAM with point and line features, where line features are represented in Plücker coordinates [17].

In many of these previous works, different representations for line features were used without considering their impacts on estimator performance. Leveraging our previous work [18], which offered a brief survey of line representations and proposed a "closest point" representation, in this paper we conduct a performance evaluation for the commonly used line parameterizations. In addition, we investigate and identify the degenerate motions for line feature triangulation which cause poor line estimates and have large practical impacts for monocular VINS which might leverage line features. The main contributions of this paper can be listed as follows:

- We present a tightly-coupled monocular visual-inertial navigation system which leverages point and line features and performs online spatial and temporal calibration.
- We propose two sliding window based line feature triangulation algorithms and compare their performances. We identify 3 degenerate motions that cause these line triangulation methods to fail.
- We investigate commonly used line representations and numerically compare their performances in a line feature based monocular visual-inertial SLAM scenario.
- Real-world experiments are performed to validate the designed system with the CP line representation, and the performances are shown to be improved than the system using points only.

II. Problem Formulation

In order to properly contextualize our line analysis, we first formulate the standard visual-inertial odometry problem

This work was partially supported by the University of Delaware (UD) College of Engineering, the ARL (W911NF-19-2-0226, JWS 10-051-003) the DTRA (HDTRA1-16-1-0039), and Google Daydream. P. Geneva was also partially supported by the Delaware Space Grant College and Fellowship Program (NASA Grant NNX15AI19H).

Y. Yang, K. Eckenhoff and G. Huang are with the Department of Mechanical Engineering, University of Delaware, Newark, DE 19716, USA. Email: {yuyang,keck,ghuang}@udel.edu

P. Geneva is with the Department of Computer Information and Science, University of Delaware, Newark, DE 19716, USA. Email: {pgeneva}@udel.edu

when using both point and line features. We define the state vector for visual-inertial odometry as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_{calib}^\top & t_d & \mathbf{x}_c^\top \end{bmatrix}^\top \tag{1}$$

where $\mathbf{x}_I$ denotes the IMU state, $\mathbf{x}_{calib}$ denotes the rigid transformation between the IMU and camera, t_d represents the time-offset and $\mathbf{x}_c$ represents the cloned IMU states. At time step k, the state vector can be written as:

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_G\bar{q}^\top & {}^{I_k}\mathbf{b}_g^\top & {}^G\mathbf{v}_{I_k}^\top & {}^{I_k}\mathbf{b}_a^\top & {}^G\mathbf{p}_{I_k}^\top \end{bmatrix}^\top \tag{2}$$

where ${}^{I_k}_G\bar{q}$ denotes JPL quaternion [19], representing the rotation from global frame $\{G\}$ to IMU frame $\{I\}$ at time step k. ${}^G\mathbf{v}_{I_k}$ and ${}^G\mathbf{p}_{I_k}$ represents the IMU velocity and position in the global frame at time step k, respectively. ${}^{I_k}\mathbf{b}_g$ and ${}^{I_k}\mathbf{b}_a$ represents the gyroscope and accelerometer biases. We define the error states of the IMU state as:

$$\tilde{\mathbf{x}}_{I_k} = \begin{bmatrix} \delta^{I_k}_G\theta^\top & {}^{I_k}\tilde{\mathbf{b}}_g^\top & {}^G\tilde{\mathbf{v}}_{I_k}^\top & {}^{I_k}\tilde{\mathbf{b}}_a^\top & {}^G\tilde{\mathbf{p}}_{I_k}^\top \end{bmatrix}^\top \tag{3}$$

where the error, $\tilde{\mathbf{x}}$, is defined as the difference between the true state, $\mathbf{x}$, and estimated state $\hat{\mathbf{x}}$, that is $\tilde{\mathbf{x}} := \mathbf{x} - \hat{\mathbf{x}}$. However, for the JPL quaternion, the error state is defined as $\delta\theta$, where we have:

$$\bar{q} = \delta\bar{q} \otimes \hat{\bar{q}} \simeq \begin{bmatrix} \frac{1}{2}\delta\theta^\top & 1 \end{bmatrix}^\top \otimes \hat{\bar{q}} \tag{4}$$

with $\otimes$ denotes the quaternion multiplication [19].

In addition to this IMU navigation state, we also estimate the spatial calibration, $\mathbf{x}_{calib}$, between the IMU and camera. In particular, our state vector contains the rotation from the IMU frame to the camera frame, ${}^C_I\bar{q}$, as well as the translation from camera to IMU ${}^C\mathbf{p}_I$. Explicitly, we have:

$$\mathbf{x}_{calib} = \begin{bmatrix} {}^C_I\bar{q}^\top & {}^C\mathbf{p}_I^\top \end{bmatrix}^\top \tag{5}$$

Moreover, due to the nature of electronic hardware (e.g., asynchronous clocks, data transmission delays and electronic triggering delays), the timestamps reported by each of the sensors will differ from the "true" time that the measurements were recorded. In this work, we treat the IMU clock as the true time and estimate the offset of the aiding sensor relative to this base clock [20, 21]. We model the time offset t_d as a constant value:

$$t_d = t_C - t_I \tag{6}$$

where t_C is the time recorded on the sensor measurements, and t_I is the corresponding true IMU time.

If there are m cloned IMU poses at the time step k (corresponding to the poses of the IMU at the *true* imaging times), then $\mathbf{x}_{ck}$ can be written as:

$$\mathbf{x}_{c_k} = \begin{bmatrix} {}^{I_{k-1}}_G\bar{q}^\top & {}^G\mathbf{p}_{I_{k-1}}^\top & \cdots & {}^{I_{k-m}}_G\bar{q}^\top & {}^G\mathbf{p}_{I_{k-m}}^\top \end{bmatrix}^\top \tag{7}$$

A. System Dynamic Model

The local linear acceleration $\mathbf{a}$ and angular velocity measurements ω are modeled with additive noises and biases:

$$\mathbf{a}_m = \mathbf{a} + {}^I_G\mathbf{R}\,{}^G\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a \tag{8}$$
$$\omega_m = \omega + \mathbf{b}_g + \mathbf{n}_g \tag{9}$$

where $\mathbf{n}_g$ and $\mathbf{n}_a$ are the continuous-time Gaussian noises that contaminate the IMU readings. The dynamic model for this system is given by [19]:

$$\begin{aligned} &{}^I_G\dot{\bar{q}}(t) = \frac{1}{2}\Omega\left({}^I\omega(t)\right){}^I_G\bar{q}(t), \quad {}^G\dot{\mathbf{p}}_I(t) = {}^G\mathbf{v}_I(t), \quad {}^G\dot{\mathbf{v}}_I(t) = {}^G\mathbf{a}(t) \\ &\dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}, \quad \dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t), \quad \dot{t}_d = 0 \\ &\dot{\mathbf{x}}_{calib}(t) = \mathbf{0}_{6\times 1}, \quad \dot{\mathbf{x}}_c(t) = \mathbf{0}_{6m\times 1} \end{aligned} \tag{10}$$

where $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ denote the zero-mean Gaussian noises driving the IMU gyroscope and accelerometer biases, ${}^G\mathbf{g}$ denotes gravity, $\lfloor\cdot\rfloor$ represents the skew matrix, $\Omega(\omega) \triangleq \begin{bmatrix} -\lfloor\omega\rfloor & \omega \\ -\omega^\top & 0 \end{bmatrix}$ and continuous time variable $\mathbf{x}(t)$ denotes the state value at time t. After linearization, the error state dynamic equation are:

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}(t) &\simeq \begin{bmatrix} \mathbf{F}_I(t) & \mathbf{0}_{15\times(6m+7)} \\ \mathbf{0}_{(6m+7)\times 15} & \mathbf{0}_{(6m+7)} \end{bmatrix} \tilde{\mathbf{x}}(t) + \begin{bmatrix} \mathbf{G}_I(t) \\ \mathbf{0}_{(6m+7)\times 12} \end{bmatrix} \mathbf{n}(t) \\ &= \mathbf{F}(t)\tilde{\mathbf{x}}(t) + \mathbf{G}(t)\mathbf{n}(t) \end{aligned} \tag{11}$$

where $\mathbf{F}_I(t)$ and $\mathbf{G}_I(t)$ are the continuous-time IMU error state and noise Jacobians matrices, respectively, and $\mathbf{n}(t) = [\mathbf{n}_g^\top \ \mathbf{n}_{wg}^\top \ \mathbf{n}_a^\top \ \mathbf{n}_{wa}^\top]^\top$ represents the system noises modeled as a zero-mean white Gaussian process with autocorrelation $\mathbb{E}[\mathbf{n}(t)\mathbf{n}^\top(t)] = \mathbf{Q}\delta(t-\tau)$.

To propagate the covariance $\mathbf{P}_{k|k}$ at time step k, the state transition matrix $\Phi_{(k+1,k)}$ from time t_k to t_{k+1} can be computed by solving $\dot{\Phi}_{(t,k)} = \mathbf{F}(t)\Phi_{(t,k)}$ with identity initial conditions. Thus, the discrete-time noise covariance and the propagated covariance can be written as:

$$\mathbf{Q}_k = \int_{t_k}^{t_{k+1}} \Phi_{(k+1,\tau)}\mathbf{G}(\tau)\mathbf{Q}\mathbf{G}^\top(\tau)\Phi_{(k+1,\tau)}^\top \mathbf{d}\tau \tag{12}$$
$$\mathbf{P}_{k+1|k} = \Phi_{(k+1,k)}\mathbf{P}_{k|k}\Phi_{(k+1,k)}^\top + \mathbf{Q}_k \tag{13}$$

B. Point Measurement Model

As the camera moves through an environment, point feature measurements can be extracted and tracked between images. These camera measurements are described by:

$$\mathbf{z}_p = \Pi({}^C\mathbf{x}_p) + \mathbf{n}_f, \quad \Pi([x\ y\ z]^\top) = \begin{bmatrix} \frac{x}{z} & \frac{y}{z} \end{bmatrix}^\top \tag{14}$$

where ${}^C\mathbf{x}_p$ represents the 3D position of the point feature expressed in the camera frame. According to our time offset definition (6), the feature ${}^C\mathbf{x}_p$ in the sensor frame with reported time stamp t corresponds to the time $t - t_d$ in the IMU base clock. Hence, we have:

$${}^C\mathbf{x}_p = {}^C_I\mathbf{R}\,{}^I_G\mathbf{R}(t-t_d)\left({}^G\mathbf{x}_p - {}^G\mathbf{p}_I(t-t_d)\right) + {}^C\mathbf{p}_I \tag{15}$$

where ${}^I_G\mathbf{R}(t-t_d)$ and ${}^G\mathbf{p}_I(t-t_d)$ represent the IMU pose at time $t-t_d$, which will be denoted as time step k for simplicity in the ensuing derivations.

C. Line Measurement Model

Similar to [15], we adopt a simple projective line measurement model which describes the distance of two line endpoints, $\mathbf{x}_s = [u_s\ v_s\ 1]^\top$ and $\mathbf{x}_e = [u_e\ v_e\ 1]^\top$, to the projected line segment in the image:

$$\mathbf{z}_l = \begin{bmatrix} \frac{\mathbf{x}_s^\top \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} & \frac{\mathbf{x}_e^\top \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} \end{bmatrix}^\top \tag{16}$$

where (u, v) are the coordinates of the point on the image plane, while $\mathbf{l} = [l_1 \ l_2 \ l_3]^\top$ is the 2D image line projected from the 3D line ${}^C\mathbf{x}_l$ expressed in the camera frame.

D. Multi-State Constraint Kalman Filter (MSCKF)

Each feature measurement, whether it be a point (14) or line measurement (16), can be written generically as:

$$\mathbf{z} = \mathbf{h}\left(\mathbf{x}, {}^G\mathbf{x}_f\right) + \mathbf{n}_z \tag{17}$$

where ${}^G\mathbf{x}_f$ represents the feature (either a point ${}^G\mathbf{x}_p$, or line ${}^G\mathbf{x}_l$). If ${}^G\mathbf{x}_f$ is kept in the state vector, the problem size (and thus the computational burden) will grow unbounded, quickly preventing real-time estimation. One way to avoid this is to use a null space operation [22] to marginalize these features.

To perform this, the MSCKF maintains a sliding window of stochastically cloned historical IMU poses corresponding to past imaging times in the state vector, and accumulates the corresponding feature measurements collected over this window. In our implementation, we clone these poses at the "true" imaging times in order to estimate the time offset as in [20]. By stacking the measurements corresponding to a single feature and using the current estimates of the IMU clones in the sliding window, we can triangulate this feature to form its estimate, ${}^G\hat{\mathbf{x}}_f$. The feature measurement (17) can then be linearized as:

$$\tilde{\mathbf{z}} \simeq \mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{H}_f {}^G\tilde{\mathbf{x}}_f + \mathbf{n}_z \tag{18}$$

where $\mathbf{H}_x$ and $\mathbf{H}_f$ represent the Jacobians w.r.t. the state $\mathbf{x}$ and feature, respectively. Decomposing $\mathbf{H}_f$ with QR factorization gives the following:

$$\mathbf{H}_f = \begin{bmatrix}\mathbf{Q}_e & \mathbf{Q}_n\end{bmatrix}\begin{bmatrix}\mathbf{R}_\Delta \\ \mathbf{0}\end{bmatrix} = \mathbf{Q}_e\mathbf{R}_\Delta \tag{19}$$

Note that $\mathbf{Q}_n^\top$ is the left null space of $\mathbf{H}_f$, i.e., $\mathbf{Q}_n^\top\mathbf{H}_f = \mathbf{0}$. Multiplying (18) to the left by $\mathbf{Q}_n^\top$ yields a new measurement model independent of the feature error:

$$\tilde{\mathbf{z}}' \simeq \mathbf{H}_x'\tilde{\mathbf{x}} + \mathbf{n}_z' \tag{20}$$

As this new measurement relates only to quantities already contained in the state vector, the standard EKF update can be performed. Utilizing this null space operation for every feature tracked over the sliding window keeps the problem size (and thus the computational cost) bounded.

III. Line Representation and Jacobians

A. Line Representation

In order to incorporate line measurements into the estimator, we need to find an appropriate representation for these features.

In our previous work [18], we summarized several line error representations (including orthonormal, quaternion, and closest point) shown in Table I and Figure 1a.

Note that $\bar{q}_l$ is a unit quaternion and $\bar{q}_l = [\mathbf{q}_l^\top \ q_l]^\top$. Given the 3D positions of two points $\mathbf{p}_{f1}$ and $\mathbf{p}_{f2}$ (expressed in the same frame) corresponding to the same line $\mathbf{x}_l$, we can obtain its Plücker coordinate (Model 1 in Table I) as [15, 23]:

$$\begin{bmatrix}\mathbf{n}_l \\ \mathbf{v}_l\end{bmatrix} = \begin{bmatrix}\lfloor\mathbf{p}_{f1}\rfloor\mathbf{p}_{f2} \\ \mathbf{p}_{f2} - \mathbf{p}_{f1}\end{bmatrix} \tag{21}$$

TABLE I: Line representation and corresponding error states

Model #	Line	Error states
1: Orthonormal	$\mathbf{n}_l$, $\mathbf{v}_l$	$\delta\theta_l$, $\delta\phi_l$
2: Quaternion	d_l, $\bar{q}_l$ with $\mathbf{R}(\bar{q}_l) = [\mathbf{n}_e, \mathbf{v}_e, \lfloor\mathbf{n}_e\rfloor\mathbf{v}_e]$	$\delta\theta_l$, $\tilde{d}_l$
3: Closest Point (CP)	$\mathbf{p}_l = d_l\bar{q}_l$	$\mathbf{p}_l = \hat{\mathbf{p}}_l + \tilde{\mathbf{p}}_l$

where $\mathbf{n}_l$ represents the normal direction of the plane constructed by the two points and the origin while $\mathbf{v}_l$ represents the line direction. The distance from the origin to the line can be computed as $d_l = \|\mathbf{n}_l\| / \|\mathbf{v}_l\|$. A minimal orthonormal error state ($\delta\theta_l$ and $\delta\phi_l$) is introduced by [15] for line feature-based structure from motion (SfM). Based on Model 1, we can conclude the basic geometric elements of a line, including a unit normal direction $\mathbf{n_e} = \frac{\mathbf{n}_l}{\|\mathbf{n}_l\|}$, a unit line direction $\mathbf{v_e} = \frac{\mathbf{v}_l}{\|\mathbf{v}_l\|}$, and the distance scalar d_l (see Fig. 1a).

With these geometric elements, Kottas et al. [8] proposed the use of a unit quaternion $\bar{q}_l$ and a distance scalar d_l to represent a line (Model 2), where the quaternion describes the line direction:

$$\mathbf{R}(\bar{q}_l) = \begin{bmatrix}\mathbf{n_e} & \mathbf{v_e} & \lfloor\mathbf{n_e}\rfloor\mathbf{v_e}\end{bmatrix}, \quad \bar{q}_l \simeq \begin{bmatrix}\frac{1}{2}\delta\theta_l \\ 1\end{bmatrix} \otimes \hat{\bar{q}}_l \tag{22}$$

where $\delta\theta_l$ represents the error state of the line quaternion, while $\mathbf{R}(\bar{q}_l)$ refers to the rotation matrix associated with $\bar{q}_l$. The 4D minimal error states of the line include the quaternion error angle and the distance scalar error: $[\delta\theta_l^\top \ \tilde{d}_l]^\top$.

More importantly, if we multiply the unit quaternion $\bar{q}_l$ with the distance scalar d_l, we obtain a 4D vector, which can be considered as the "closest point" for a line in the 4D vector space (i.e., Model 3):

$$\mathbf{p}_l = d_l\bar{q}_l = d_l\begin{bmatrix}\mathbf{q}_l^\top & q_l\end{bmatrix}^\top = \hat{\mathbf{p}}_l + \tilde{\mathbf{p}}_l \tag{23}$$

where $\tilde{\mathbf{p}}_l$ is the 4D error state for the closest point of a line. In this paper, we will compare the performance of these line representations within a SfM framework.

B. Line Measurement Jacobians

We use the CP representation to show how to compute the measurement Jacobians for line measurements (16). For the line projection in CP form, we have the following equalities:

$$\mathbf{l} = [\mathbf{K} \ \ \mathbf{0}_3]{}^C\mathbf{L}, \quad {}^C\mathbf{L} = \begin{bmatrix}{}^Cd_l{}^C\mathbf{n}_e^\top & {}^C\mathbf{v}_e^\top\end{bmatrix}^\top \tag{24}$$

$${}^C\mathbf{L} = \begin{bmatrix}{}^C_I\mathbf{R} & \lfloor{}^C\mathbf{p}_I\rfloor{}^C_I\mathbf{R} \\ \mathbf{0}_3 & {}^C_I\mathbf{R}\end{bmatrix}\begin{bmatrix}{}^I_G\mathbf{R}(t-t_d) & -{}^I_G\mathbf{R}(t-t_d)\lfloor{}^G\mathbf{p}_I(t-t_d)\rfloor \\ \mathbf{0}_3 & {}^I_G\mathbf{R}(t-t_d)\end{bmatrix}{}^G\mathbf{L} \tag{25}$$

Therefore, the measurement Jacobians can be written as:

$$\frac{\partial\tilde{\mathbf{z}}}{\partial{}^G\tilde{\mathbf{p}}_I} = \frac{\partial\tilde{\mathbf{z}}}{\partial\tilde{\mathbf{l}}}\frac{\partial\tilde{\mathbf{l}}}{\partial{}^C\tilde{\mathbf{L}}}\frac{\partial{}^C\tilde{\mathbf{L}}}{\partial{}^I\tilde{\mathbf{L}}}\frac{\partial{}^I\tilde{\mathbf{L}}}{\partial{}^G\tilde{\mathbf{p}}_I} \tag{26}$$

$$\frac{\partial\tilde{\mathbf{z}}}{\partial\delta\theta_I} = \frac{\partial\tilde{\mathbf{z}}}{\partial\tilde{\mathbf{l}}}\frac{\partial\tilde{\mathbf{l}}}{\partial{}^C\tilde{\mathbf{L}}}\frac{\partial{}^C\tilde{\mathbf{L}}}{\partial{}^I\tilde{\mathbf{L}}}\frac{\partial{}^I\tilde{\mathbf{L}}}{\partial\delta\theta_I} \tag{27}$$

$$\frac{\partial\tilde{\mathbf{z}}}{\partial\tilde{\mathbf{x}}_{calib}} = \frac{\partial\tilde{\mathbf{z}}}{\partial\tilde{\mathbf{l}}}\frac{\partial\tilde{\mathbf{l}}}{\partial{}^C\tilde{\mathbf{L}}}\frac{\partial{}^C\tilde{\mathbf{L}}}{\partial\tilde{\mathbf{x}}_{calib}} \tag{28}$$

$$\frac{\partial\tilde{\mathbf{z}}}{\partial{}^G\tilde{\mathbf{p}}_l} = \frac{\partial\tilde{\mathbf{z}}}{\partial\tilde{\mathbf{l}}}\frac{\partial\tilde{\mathbf{l}}}{\partial{}^C\tilde{\mathbf{L}}}\frac{\partial{}^C\tilde{\mathbf{L}}}{\partial{}^I\tilde{\mathbf{L}}}\frac{\partial{}^I\tilde{\mathbf{L}}}{\partial[\delta\theta_l^\top \ \tilde{d}_l]^\top}\frac{\partial[\delta\theta_l^\top \ \tilde{d}_l]^\top}{\partial{}^G\tilde{\mathbf{p}}_l} \tag{29}$$

Please refer to Appendix I for detailed derivations.

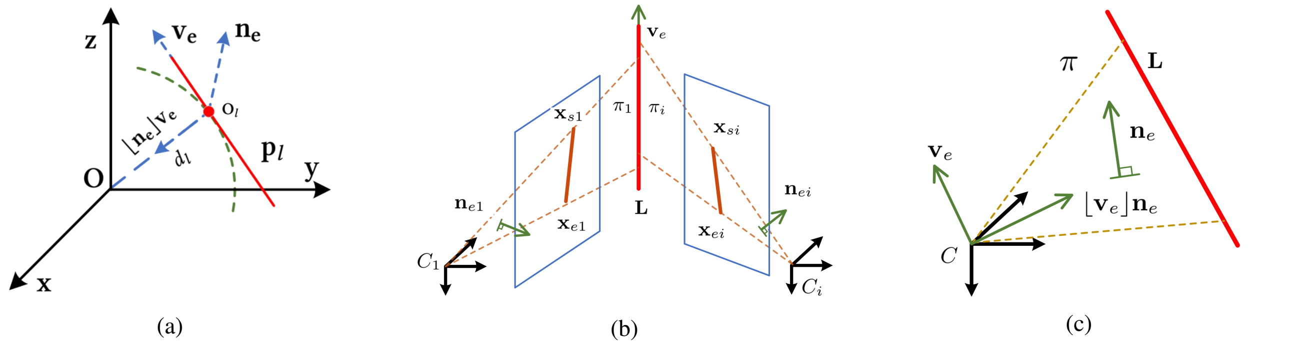

Fig. 1: Basic geometric elements for a 3D line (a). Sliding window-based line feature triangulation (b). Degenerate motion analysis for line feature triangulation (c).

IV. Line Triangulation

In order to utilize line features in the MSCKF, an estimate of its 3D line parameters is needed to linearize the measurement model (17). From the above section, it is clear that we need the basic geometric elements (e.g., $\mathbf{v}_e$, $\mathbf{v}_e$ and d_l) for a line feature.

Image line detectors such as Line Segment Detector (LSD) [24] extract two arbitrary endpoints of the line segment in each image. Consider line feature, $\mathbf{x}_l$, which has been detected and tracked over the sliding window, yielding a collection of line segment endpoint pairs, we propose two algorithms for line feature triangulation based on these stacked endpoint measurements.

A. Algorithm A

Denoting the endpoints for a line in the i-th image in the sliding window as $\mathbf{x}_{si}$ and $\mathbf{x}_{ei}$, see Figure 1b, we obtain the normal direction of the plane π_i formed by the line $\mathbf{x}_l$ and the i-th camera center:

$$^{C_i}\mathbf{n}_{ei} = \frac{\lfloor \mathbf{x}_{si} \rfloor \mathbf{x}_{ei}}{\left\| \lfloor \mathbf{x}_{si} \rfloor \mathbf{x}_{ei} \right\|} \tag{30}$$

Since line $\mathbf{x}_l$ resides on every plane π_i, we have the following constraint:

$$\underbrace{\begin{bmatrix} \vdots \\ {}^{C_i}\mathbf{n}_{ei}^{\top}\, {}^{C_1}_{C_i}\mathbf{R}^{\top} \\ \vdots \end{bmatrix}}_{\mathbf{N}} {}^{C_1}\mathbf{v}_{e1} = \mathbf{0} \tag{31}$$

Therefore, $^{C_1}\mathbf{v}_{e1}$ can be found as the unit vector minimizing the error on this constraint, which is given by the eigenvector corresponding to the smallest eigenvalue of $\mathbf{N}^{\top}\mathbf{N}$.

The transformation of a line expressed in frame C_i to a representation in frame C_1 is given by:

$$\begin{bmatrix} {}^{C_1}d_l {}^{C_1}\mathbf{n}_{e1} \\ {}^{C_1}\mathbf{v}_{e1} \end{bmatrix} = \begin{bmatrix} {}^{C_1}_{C_i}\mathbf{R} & \lfloor {}^{C_1}\mathbf{p}_{C_i} \rfloor {}^{C_1}_{C_i}\mathbf{R} \\ \mathbf{0}_3 & {}^{C_1}_{C_i}\mathbf{R} \end{bmatrix} \begin{bmatrix} {}^{C_i}d_l {}^{C_i}\mathbf{n}_{ei} \\ {}^{C_i}\mathbf{v}_{ei} \end{bmatrix} \tag{32}$$

$$\Rightarrow {}^{C_1}d_l {}^{C_1}\mathbf{n}_{e1} - {}^{C_i}d_l {}^{C_1}_{C_i}\mathbf{R} {}^{C_i}\mathbf{n}_{ei} = \lfloor {}^{C_1}\mathbf{p}_{C_i} \rfloor {}^{C_1}_{C_i}\mathbf{R} {}^{C_i}\mathbf{v}_{ei} \tag{33}$$

$$\Rightarrow {}^{C_1}d_l \mathbf{b}_i^{\top} {}^{C_1}\mathbf{n}_{e1} = \mathbf{b}_i^{\top} \lfloor {}^{C_1}\mathbf{p}_{C_i} \rfloor {}^{C_1}_{C_i}\mathbf{R} {}^{C_i}\mathbf{v}_{ei} \tag{34}$$

where $\mathbf{b}_i = \lfloor {}^{C_1}\mathbf{v}_{e1} \rfloor {}^{C_1}_{C_i}\mathbf{R} {}^{C_i}\mathbf{n}_{ei}$ is a unit vector perpendicular to ${}^{C_1}_{C_i}\mathbf{R} {}^{C_i}\mathbf{n}_{ei}$. Given all the measurements from $i = 2 \dots m$, we build a linear system as:

$${}^{C_1}d_l \begin{bmatrix} \vdots \\ \mathbf{b}_i^{\top} {}^{C_1}\mathbf{n}_{e1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{b}_i^{\top} \lfloor {}^{C_1}\mathbf{p}_{C_i} \rfloor {}^{C_1}_{C_i}\mathbf{R} {}^{C_i}\mathbf{v}_{ei} \\ \vdots \end{bmatrix} \tag{35}$$

By solving the above system, we obtain $^{C_1}d_l$. After this step, we have recovered all the required line parameters: $^{C_1}\mathbf{n}_{e1}$ eq. (30), $^{C_1}\mathbf{v}_{e1}$ eq. (32) and $^{C_1}d_l$ eq. (35). The triangulated 3D line feature can be transformed to other frames (i.e. global frame G) as needed. Note that this is a generic algorithm and we have made no assumptions on the correspondences of the endpoints.

B. Algorithm B

One of the classical methods to triangulate line features is based on the two intersecting planes (e.g., π_1 and π_i). The dual Plücker matrix $\mathbf{L}^*$ can be computed as:

$$\mathbf{L}^* = \pi_1 \pi_i^{\top} - \pi_i \pi_1^{\top} = \begin{bmatrix} \lfloor {}^{C_1}\mathbf{v}_{e1}^{(i)} \rfloor & {}^{C_1}d_l^{(i)} {}^{C_1}\mathbf{n}_{e1}^{(i)} \\ -{}^{C_1}d_l^{(i)} ({}^{C_1}\mathbf{n}_{e1}^{(i)})^{\top} & 0 \end{bmatrix} \tag{36}$$

where $\pi_1 = \begin{bmatrix} {}^{C_1}\mathbf{n}_{e1}^{\top} & 0 \end{bmatrix}^{\top}$ and $\pi_i = \begin{bmatrix} {}^{C_1}\mathbf{n}_{ei}^{\top} & {}^{C_1}\mathbf{n}_{ei}^{\top} {}^{C_1}\mathbf{p}_{C_i} \end{bmatrix}^{\top}$. The line geometric elements $^{C_1}d_l^{(i)}$, $^{C_1}\mathbf{n}_{e1}^{(i)}$ and $^{C_1}\mathbf{v}_{e1}^{(i)}$ are computed based on π_1 and π_i. In this work, we offer a generalization of this method for m measurements. In particular, we solve for the line parameters using:

$${}^{C_1}\mathbf{n}_{e1} = \sum_{i=2}^{m} {}^{C_1}\mathbf{n}_{e1}^{(i)} \Big/ \left\| \sum_{i=2}^{m} {}^{C_1}\mathbf{n}_{e1}^{(i)} \right\| \tag{37}$$

$${}^{C_1}\mathbf{v}_{e1} = \sum_{i=2}^{m} {}^{C_1}\mathbf{v}_{e1}^{(i)} \Big/ \left\| \sum_{i=2}^{m} {}^{C_1}\mathbf{v}_{e1}^{(i)} \right\| \tag{38}$$

$${}^{C_1}d_l = \frac{\sum_{i=2}^{m} {}^{C_1}d_l^{(i)}}{m-1} \tag{39}$$

After linear triangulation, we perform nonlinear least squares to refine the line estimates, noting again that we have made no assumption of the endpoint correspondences.

C. Degenerate Motion Analysis for Triangulation

When using a monocular camera, the ability to perform line feature triangulation is heavily dependent on the sensor motion. In particular, we identify degenerate motions that cause the line feature parameters to become unobservable, thereby causing triangulation failures (see Fig. 1c and Tab. II). Letting C denote the center of the camera frame and $\mathbf{L}$ is the line feature and formulate a plane π as shown in Fig. 1c.

- If the monocular camera moves along the direction $\mathbf{v}_e$ of $\mathbf{L}$ or toward $\mathbf{L}$ with direction $\lfloor \mathbf{v}_e \rfloor \mathbf{n}_e$, the camera will stay in the same plane, π. As a result, each of the ${}^{C_i}\mathbf{n}_{ei}$ will be parallel to each other, causing matrix $\mathbf{N}$ in (32) to become rank 1, thereby causing ambiguity in the solution for ${}^{C_1}\mathbf{v}_{e1}$. In addition, without ${}^{C_1}\mathbf{v}_{e1}$, ${}^{C_1}d_l$ becomes unsolvable.
- If the monocular camera undergoes pure rotation (no translation), the camera will also stay in the plane π, causing degeneracy as in the previous case.
- The effective motion for line triangulation is the motion $\mathbf{n}_e$ perpendicular to the plane π as the shown in Fig 1c.

Note that for a monocular camera, any combination of the listed 3 degenerate motions (e.g., planar motions in plane π shown in Fig. 1c) will also cause triangulation failure.

Interestingly, we see that for stereo cameras, if both cameras remain in the plane during the motion (such as when the platform translation and camera to camera offset remain in the plane), we will still have degenerate motion. This is because triangulation requires that we measure $\mathbf{L}$ from different views along $\mathbf{n}_e$. In such a case, even stereo vision cannot guarantee proper line triangulation.

TABLE II: Summary of degenerate motion for line feature triangulation with monocular camera

Motion	Solvable	Unsolvable
Along line direction $\mathbf{v}_e$	$\mathbf{n}_e$	$\mathbf{v}_e$ and d
Toward line $\lfloor \mathbf{v}_e \rfloor \mathbf{n}_e$	$\mathbf{n}_e$	$\mathbf{v}_e$ and d
Pure rotation	$\mathbf{n}_e$	$\mathbf{v}_e$ and d
Perpendicular to plane $\mathbf{n}_e$	$\mathbf{n}_e$,$\mathbf{v}_e$ and d	-
Random motion	$\mathbf{n}_e$,$\mathbf{v}_e$ and d	-

V. Simulations

We performed two Monte-Carlo simulations to verify the proposed line feature triangulation algorithms and different line representations.

A. Line Triangulation Simulation

We first used Monte-Carlo simulations to verify both the proposed sliding-window based line feature triangulation algorithms and our degenerate motion analysis. 8 lines were generated (see Fig. 2) and observed by a monocular camera from 20 poses in space, with the lines being placed about 2m in front of the poses. Similar to the real-world line segment detector (LSD), our simulated monocular camera collected the two endpoint measurements of lines in its view, with each endpoint bearing measurement corrupted by 2-pixel Gaussian random noise, while we assume no correspondences between these endpoints. 3 different camera motions (including straight line motion, planar motion and 3D sinusoidal motion) were simulated to verify the degenerate motions. During triangulation, we disturbed the true camera poses (both the orientation and position) with random noises as:

$$\bar{q}_m = \begin{bmatrix} \frac{1}{2}\mathbf{n}_\theta \\ 1 \end{bmatrix} \otimes \bar{q}, \quad \mathbf{p}_{Cm} = \mathbf{p}_C + \mathbf{n}_p \tag{40}$$

where $\mathbf{n}_\theta$ and $\mathbf{n}_p$ represents the white Gaussian noises added to the camera pose estimates, with $\sigma_\theta = 0.01rad$ and $\sigma_p = 0.005m$, respectively. while $\bar{q}_m$ and $\mathbf{p}_m$ represent the disturbed camera orientation and position estimates which were used by the triangulation algorithms. When evaluating the triangulated line errors, we transferred the estimated line parameters into CP form and computed the errors in Euclidean space.

For each motion profile we generated 30 sets of data and computed the root mean square error (RMSE) [25] for the line accuracy evaluation. As shown in Fig. 2, we see that for all the tested motion patterns, the proposed Algorithm A outperformed Algorithm B. Since lines 1, 5 and 8 are horizontal lines, when the camera performs straight line motion along this direction (shown in Fig. 2 left), their estimates can not be accurately triangulated. For the planar trajectory, the camera moved in the same plane formulated by the camera center and line 5, therefore, line 5's triangulation still failed. For lines 1 and 8, their accuracy was slightly improved over the 1D motion case because this planar motion is not strictly degenerate for them. Finally, in the 3D motion case, all lines were successfully triangulated with relatively low error due to the fact that all degenerate cases were avoided.

B. Line Representation Simulation

When fusing line information into any estimator, it is vital to be able to determine which of those possible line representations yields the best performance. In order to test the effect of line representation choice, we performed a Monte-Carlo simulations using a visual line-SLAM system. A line map (64 lines in total) in an indoor room was generated while a monocular camera was simulated to follow a sinusoidal trajectory (with 150 simulated poses), as shown in the right of Fig. 3.

To simplify the simulation, we simulated relative pose odometry measurements for the camera (which were also disturbed with pose noises (40)). In order to test the robustness of the line representations, we performed tests using 3 different image ise levels (2, 6 and 8 pixels) to corrupt the line endpoint measurements (as shown in the right of Fig. 3). To simplify the simulation, the camera traversed a single loop of this trajectory as shown in the left of Fig. 3. The line features were triangulated using proposed Algorithm A. To solve the visual line-SLAM problem, we formulated it as a Maximum Likelihood Estimation (MLE) problem which can be computed as an instance of Nonlinear Least Squares [26]. In this simulation, we allowed 5 Gauss-Newton iterations for each representation for a fair comparison. We ran 30 Monte-Carlo simulations and computed the RMSE for the camera poses to evaluate the accuracy.

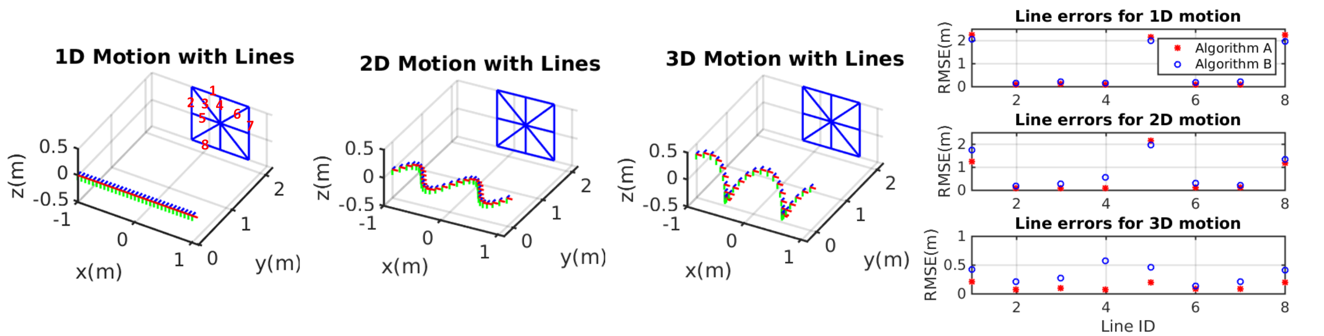

Fig. 2: Simulation setup for line triangulation. 8 lines are simulated with IDs in the left figure. 3 motion patterns are simulated, including (left) straight line motion, (middle-left) 2D planar motion and (middle-right) 3D motion. Note that 2D planar motion is in the plane formed by the initial camera position and line 5. On the far right, line triangulation RMSE (computed in CP form) across the Monte-Carlo simulations for the three tested motion profiles.

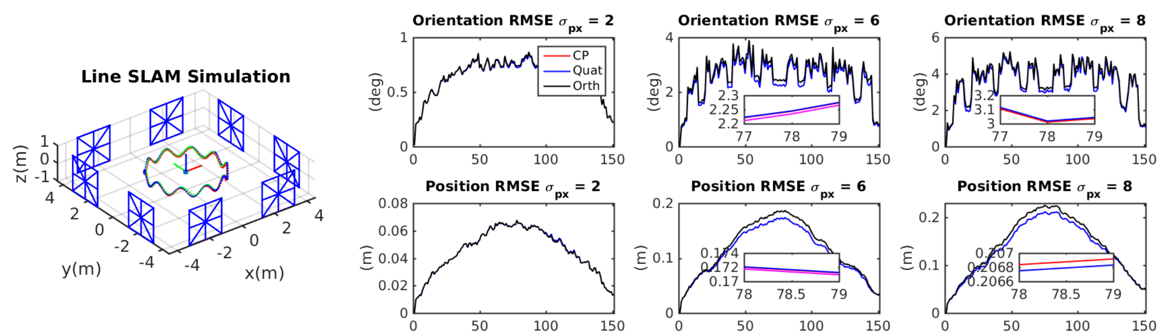

Fig. 3: Simulation environmental setup for visual line-SLAM (left). Average orientation and position RMSE for visual SLAM with line features in different representations over the 30 Monte-Carlo simulation runs (right).

As shown in Fig. 3, all 3 representations gave similar SLAM performance. However, as we increase the measurement noise levels, the Plücker representation with orthonormal error states tends to perform slightly worse than the others (e.g., the CP and quaternion representations). Note that in all the noise levels tested, the CP and quaternion line representations yielded similar performance. These results indicate that one of these two representations should be used in practice, in particular for low-cost visual sensors which display a large amount of noise.

VI. Experiments

For our real-world experiments, we implemented the proposed triangulation and MSCKF line feature update. A sparse point feature pipeline first extracts FAST [27] features which are then tracked frame to frame using KLT [28], after which 8-point RANSAC is used to reject outliers. For the line segments, we leverage the Line Segment Detector (LSD) [24] implementation within OpenCV [29] to detect lines and extract their descriptors using Line Band Descriptors (LBD) [30]. Incoming grayscale monocular images are first undistorted to allow for extraction of *straight* line segments in the image plane. We found that that we were able to robustly match lines without RANSAC by first matching using a knn-matcher [29]. We additionally enforced that extracted lines had "significant" length to prevent noisy detections of small line segments. The line detection within OpenCV is the major bottleneck for processing time within the estimator, and limits the processing frame rate to 15 Hz, while the point-only MSCKF can run up to 50 Hz on an Intel Xeon E3-1505Mv6 with 3.00GHz.

We validated the designed system on the Vicon room

TABLE III: Average absolute trajectory error for the different Euroc Mav datasets. Averaged over 25 runs, both orientation and position errors are reported.

Algorithm		V1_01	V1_02	V1_03	V2_01	V2_02
MSCKF	(deg)	6.663	1.675	1.557	1.529	1.930
	(m)	0.154	0.114	0.186	0.121	0.137
MSCKF w/ Lines	(deg)	6.582	1.656	1.957	1.324	1.710
	(m)	0.149	0.119	0.139	0.096	0.117

TABLE IV: Relative pose error for different segment lengths. Units for orientation and position errors are in degrees and meters respectively.

Segment Length	MSCKF Ori.	MSCKF Pos.	MSCKF w/ Lines Ori.	MSCKF w/ Lines Pos.
8m	1.1688	0.1244	1.1181	0.1155
16m	1.4193	0.1531	1.3238	0.1410
24m	1.4355	0.1667	1.3643	0.1533
32m	1.7254	0.1963	1.5318	0.1768
40m	1.9270	0.2292	1.7210	0.2020

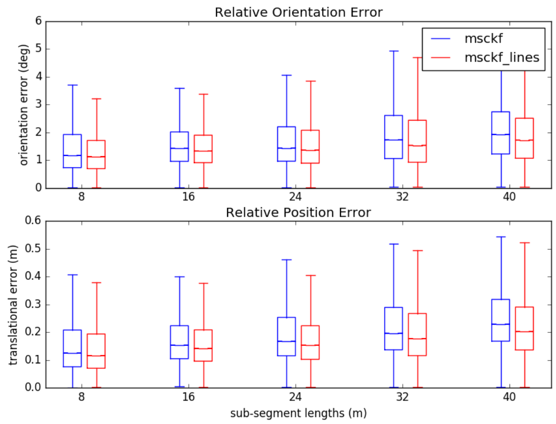

Fig. 4: Boxplot of the relative trajectory error statistics. The middle box spans the first and third quartiles, while the whiskers are the upper and lower limits. Plot best seen in color.

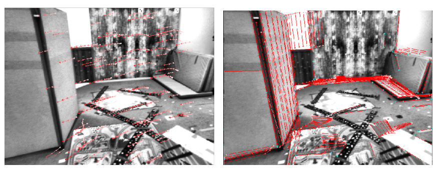

Fig. 5: Example tracking Euroc Mav datasets [31] for both point features (left) and line tracks (right). A history of tracks has been overlaid for each point/line in the image.

Euroc Mav datasets [31]. The Euroc Mav datasets provide stereo greyscale images at 20Hz, of which we only use the left image for our monocular system, a 200Hz ADIS16448 IMU, and Vicon groundtruth trajectories. Each dataset has a dynamic aerial trajectory, of average length of 70 meters, that each exhibit varying degrees of motion blur and textureless regions, making it a challenging scenario for VINS. We tracked 100 sparse feature points, while we tracked around 20 lines in each frame (Figure 5 shows example extracted image) with a sliding window size of 15 poses. Lines that have been lost or reached the maximum window size are first triangulated using the method proposed in Section IV-A, after which they are used to update the state following Section II-D. We found that in most of these datasets we had around 4 lines at each timestep that can be used in update.

Shown in Table III, we calculated the average absolute trajectory error (ATE) [32] for the proposed method and the standard MSCKF [5] for each of the Vicon room datasets. We found that due to the extreme image blur and motion we were unable to run both our standard MSCKF and the proposed on the V2_03 dataset and thus do not report its result. The average ATE is calculated over 25 runs to account for the randomness in the feature frontends. It is clear that the proposed point and line-aided MSCKF is able to either match the performance of the standard point MSCKF or in most cases outperform it in terms of orientation and position accuracy. This is due to the inclusion of additional pose constraints from the free line measurements into the estimator.

We further evaluated the proposed method by looking at the relative pose errors (RPE) [32] in Figure 4 and Table IV. This RPE is calculated over all dataset runs and thus shows the accuracy and variance for different distance segment lengths. It is clear that adding lines consistently decreases both the orientation and position error as the segment length grows. It is interesting that as the segment length grows, the difference in position accuracy also grows.

A. Remarks

It should be pointed out that, we empirically found that there are a few cases where the inclusion of free line features in VIO does not improve the estimation performance much, as shown in the V1_02 and V1_03 datasets. This is likely due to three key issues inherent to line features: (i) structure of the explored environment, (ii) motion of the monocular camera relative to the features, (iii) and poor visual line tracking. Through our experiments, we saw that free-line tracking and estimates of unstructured environments will be of low quality and provide little information to the estimator. We also found that line triangulation and estimation can suffer if the VIO platform traveled in the degenerate directions (see Table II), and these degenerate motions may often occur for free lines. To address these issues in the future, we will adopt long-term robust line features into our VIO state, avoid these degenerate triangulation cases and leverage the loop-closure properties of lines.

VII. Conclusions and Future Work

In this paper, we designed a tightly-coupled monocular visual-inertial navigation system using point and line features with online spatial and temporal calibration. We also proposed two sliding window based 3D line triangulation algorithms and compared their performances. Based on the proposed algorithms, we identified 3 degenerate motions for monocular camera that cause the line triangulation to fail and validated their existence through simulation. Monte-Carlo simulations with a visual line SLAM setup were also performed to compare commonly used line feature representations, and the results showed that the CP and quaternion representations performed better than the orthonormal representation in scenarios with high measurement noises. Finally real world experiments were performed to verify the implemented system using the CP line representation. In the future we plan to investigate inverse depth line representations, different line outlier rejection methods, and feature constraints (e.g., parallel lines, point-on-lines, etc.).

Appendix I
Line Measurement Jacobians

The jacobians for the CP line measurements are as follows:

$$\frac{\partial \tilde{\mathbf{z}}_l}{\partial \tilde{\mathbf{l}}} = \frac{1}{l_n}\begin{bmatrix} u_1 - \frac{l_1 e_1}{ln^2} & v_1 - \frac{l_2 e_1}{ln^2} & 1 \\ u_2 - \frac{l_1 e_2}{ln^2} & v_2 - \frac{l_2 e_2}{ln^2} & 1 \end{bmatrix} \tag{41}$$

$$\frac{\partial \tilde{\mathbf{l}}}{\partial {}^C\tilde{\mathbf{L}}} = \begin{bmatrix} \mathbf{K} & \mathbf{0}_3 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} f_v & 0 & 0 \\ 0 & f_u & 0 \\ -f_v c_u & -f_u c_v & f_u f_v \end{bmatrix} \tag{42}$$

$$\frac{\partial^C\tilde{\mathbf{L}}}{\partial^I\tilde{\mathbf{L}}} = \begin{bmatrix} {}^C_I\hat{\mathbf{R}} & \lfloor {}^C\hat{\mathbf{p}}_I \rfloor {}^C_I\hat{\mathbf{R}} \\ \mathbf{0}_3 & {}^C_I\hat{\mathbf{R}} \end{bmatrix} \tag{43}$$

where $e_1 = \mathbf{l}^\top \mathbf{x}_1$, $e_2 = \mathbf{l}^\top \mathbf{x}_e$. $\mathbf{K}$ is the projection matrix for line features, f_u, f_v, c_u and c_v are the corresponding camera intrinsic parameters.

$$\frac{{}^I\tilde{\mathbf{L}}}{{}^G\tilde{\mathbf{L}}} = \begin{bmatrix} {}^I_G\hat{\mathbf{R}} & -{}^I_G\hat{\mathbf{R}} \lfloor {}^G\hat{\mathbf{p}}_I \rfloor \\ \mathbf{0}_3 & {}^I_G\hat{\mathbf{R}} \end{bmatrix} \tag{44}$$

$$\frac{\partial^C\tilde{\mathbf{L}}}{\partial\tilde{\mathbf{x}}_{calib}} = \begin{bmatrix} {}^I\hat{d}_l \lfloor {}^C_I\hat{\mathbf{R}} {}^I\hat{\mathbf{n}}_e \rfloor + \lfloor {}^C\hat{\mathbf{p}}_I \rfloor \lfloor {}^C_I\hat{\mathbf{R}} {}^I\hat{\mathbf{v}}_e \rfloor & -\lfloor {}^C_I\hat{\mathbf{R}} {}^I\hat{\mathbf{v}}_e \rfloor \\ \lfloor {}^C_I\hat{\mathbf{R}} {}^I\hat{\mathbf{v}}_e \rfloor & \mathbf{0}_3 \end{bmatrix} \tag{45}$$

The Jacobians w.r.t. the pose can be written as:

$$\frac{\partial^I\tilde{\mathbf{L}}}{\partial\delta\theta_I} = \begin{bmatrix} {}^G\hat{d}_l \lfloor {}^I_G\hat{\mathbf{R}} {}^G\hat{\mathbf{n}}_{\mathbf{e}} \rfloor - \lfloor {}^I_G\hat{\mathbf{R}} \lfloor {}^G\hat{\mathbf{p}}_I \rfloor {}^G\hat{\mathbf{v}}_{\mathbf{e}} \rfloor \\ \lfloor {}^I_G\hat{\mathbf{R}} {}^G\hat{\mathbf{v}}_{\mathbf{e}} \rfloor \end{bmatrix} \tag{46}$$

$$\frac{\partial^I\tilde{\mathbf{L}}}{\partial^G\tilde{\mathbf{p}}_I} = \begin{bmatrix} {}^I_G\hat{\mathbf{R}} \lfloor {}^G\hat{\mathbf{v}}_{\mathbf{e}} \rfloor \\ \mathbf{0}_3 \end{bmatrix} \tag{47}$$

The Jacobians w.r.t. the CP line features can be written as:

$$\frac{\partial^G\tilde{\mathbf{L}}}{\partial[\delta\theta_l^\top \ \tilde{d}_l]^\top} = \begin{bmatrix} {}^G\hat{d}_l \lfloor {}^G\hat{\mathbf{R}}_l \mathbf{e}_1 \rfloor & {}^G\hat{\mathbf{R}}_l \mathbf{e}_1 \\ \lfloor {}^G\hat{\mathbf{R}}_l \mathbf{e}_2 \rfloor & \mathbf{0}_{3\times 1} \end{bmatrix} \tag{48}$$

$$\frac{\partial[\delta\theta_l^\top \ \tilde{d}_l]^\top}{\partial^G\tilde{\mathbf{p}}_l} = \begin{bmatrix} \frac{2}{\hat{d}} \left(\hat{q}_l \mathbf{I}_3 - \lfloor \hat{\mathbf{q}}_l \rfloor \right) & -\frac{2}{\hat{d}} \hat{\mathbf{q}}_l \\ \hat{\mathbf{q}}_l^\top & \hat{q}_l \end{bmatrix} \tag{49}$$

References

[1] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J Taylor, and Vijay Kumar. "Robust stereo visual inertial odometry for fast autonomous flight". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 965–972.

[2] Kejian Wu, Ahmed Ahmed, Georgios A. Georgiou, and Stergios I. Roumeliotis. "A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices". In: *Robotics: Science and Systems*. 2015.

[3] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM". In: *European conference on computer vision*. Springer. 2014, pp. 834–849.

[4] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2018), pp. 611–625.

[5] A. I. Mourikis and S. I. Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation". In: *International Conference on Robotics and Automation*. Rome, Italy, Apr. 2007, pp. 3565–3572.

[6] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. "Keyframe-based visual-inertial odometry using nonlinear optimization". In: *International Journal of Robotics Research* (Dec. 2014).

[7] T. Qin, P. Li, and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Transactions on Robotics* 34.4 (Aug. 2018), pp. 1004–1020.

[8] Dimitrios G Kottas and Stergios I Roumeliotis. "Efficient and consistent vision-aided inertial navigation using line observations". In: *International Conference on Robotics and Automation*. Karlsruhe, Germany, May 2013, pp. 1540–1547.

[9] Dimitrios Kottas and Stergios Roumeliotis. "Exploiting Urban Scenes for Vision-aided Inertial Navigation". In: *Proc. of the Robotics: Science and Systems Conference*. Berlin, Germany, June 2013.

[10] Yijia He, Ji Zhao, Yue Guo, Wenhao He, and Kui Yuan. "Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features". In: *Sensors* 18.4 (2018), p. 1159.

[11] H. Yu and A. I. Mourikis. "Vision-aided inertial navigation with line features and a rolling-shutter camera". In: *International Conference on Robotics and Intelligent Systems*. Hamburg, Germany, 2015, pp. 892–899.

[12] Sejong Heo, Jae Hyung Jung, and Chan Gook Park. "Consistent EKF-based visual-inertial navigation using points and lines". In: *IEEE Sensors Journal* (2018).

[13] F. Zheng, G. Tsai, Z. Zhang, S. Liu, C. Chu, and H. Hu. "Trifo-VIO: Robust and Efficient Stereo Visual Inertial Odometry Using Points and Lines". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 3686–3693.

[14] C. X. Guo, K. Sartipi, R. C. DuToit, G. A. Georgiou, R. Li, J. O'Leary, E. D. Nerurkar, J. A. Hesch, and S. I. Roumeliotis. "Resource-Aware Large-Scale Cooperative Three-Dimensional Mapping Using Multiple Mobile Devices". In: *IEEE Transactions on Robotics* 34.5 (Oct. 2018), pp. 1349–1369.

[15] Adrien Bartoli and Peter Sturm. "Structure From Motion Using Lines: Representation, Triangulation and Bundle Adjustment". In: *Computer Vision and Image Understanding* 100.3 (Dec. 2005), pp. 416–441.

[16] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry". In: *IEEE Transactions on Robotics* 33.1 (Feb. 2017), pp. 1–21. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2597321.

[17] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, U. K.: Cambridge University Press, 2000.

[18] Yulin Yang and Guoquan Huang. "Aided Inertial Navigation: Unified Feature Representations and Observability Analysis". In: *Proc. IEEE International Conference on Robotics and Automation*. Montreal, Canada, May 2019.

[19] Nikolas Trawny and Stergios I. Roumeliotis. *Indirect Kalman Filter for 3D Attitude Estimation*. Tech. rep. University of Minnesota, Dept. of Comp. Sci. & Eng., Mar. 2005.

[20] M. Li and A. I. Mourikis. "Online Temporal Calibration for Camera-IMU Systems: Theory and Algorithms". In: *International Journal of Robotics Research* 33.7 (June 2014), pp. 947–964.

[21] Tong Qin and Shaojie Shen. "Online Temporal Calibration for Monocular Visual-Inertial Systems". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*. Madrid, Spain, Oct. 2018.

[22] Yulin Yang, James Maley, and Guoquan Huang. "Null-Space-based Marginalization: Analysis and Algorithm". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vancouver, Canada, Sept. 2017.

[23] X. Zuo, J. Xie, Y. Liu, and Guoquan Huang. "Robust Visual SLAM with Point and Line Features". In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vancouver, Canada, Sept. 2017.

[24] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. "LSD: A fast line segment detector with a false detection control". In: *IEEE transactions on pattern analysis and machine intelligence* 32.4 (2010), pp. 722–732.

[25] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. New York: Academic Press, 1988.

[26] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A general framework for graph optimization". In: *Proc. of the IEEE International Conference on Robotics and Automation*. Shanghai, China, May 2011, pp. 3607–3613.

[27] E. Rosten, R. Porter, and T. Drummond. "Faster and Better: A Machine Learning Approach to Corner Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (Jan. 2010), pp. 105–119.

[28] S. Baker and I. Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". In: *International Journal of Computer Vision* 56 (Mar. 2004), pp. 221–255.

[29] OpenCV Developers Team. *Open Source Computer Vision (OpenCV) Library*. Available: http://opencv.org.

[30] Lilian Zhang and Reinhard Koch. "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency". In: *Journal of Visual Communication and Image Representation* 24.7 (2013), pp. 794–805.

[31] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1157–1163.

[32] Zichao Zhang and Davide Scaramuzza. "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7244–7251.