Vision-Aided Inertial Navigation with Line Features and a Rolling-Shutter Camera

Hongsheng Yu, Anastasios I. Mourikis
Dept. of Electrical and Computer Engineering, University of California, Riverside
E-mail: {hyu|mourikis}@ece.ucr.edu

***Abstract*— The vast majority of existing methods for vision-aided inertial navigation rely on the detection and tracking of point features in the images. However, in several man-made environments, such as indoor office spaces, *straight line features* are prevalent, while point features may be sparse. Therefore, developing methods that will enable the use of straight-line features for vision-aided inertial navigation can lead to improved performance. While limited prior work on the subject exists, it assumes the use of a global-shutter camera, i.e., a camera in which all image pixels are captured simultaneously. Most low-cost cameras, however, use rolling-shutter (RS) image capture, which renders the existing methods inapplicable. To address these limitations, we here present an algorithm for vision-aided inertial navigation that employs both point and line features, and is capable of operation with RS cameras. The two key contributions of this work are (i) a novel parameterization for 3D lines, which is shown to exhibit better linearity properties than existing ones, and (ii) a novel approach for the use of line observations in images. This approach forgoes line-fitting and does not assume that a straight line in 3D projects to a straight line in the image, and is thus suitable for use with RS cameras. Our results demonstrate that our proposed estimator formulation leads to improved precision, in point-feature-poor environments.**

I. Introduction

In this paper, we focus on the problem of motion estimation using inertial measurements and visual observations of line features with a rolling-shutter (RS) camera. A significant body of literature has focused on the problem of motion estimation using cameras and inertial measurement units (IMUs). However, prior work typically assumes that the camera has a global shutter (GS), i.e., that all the pixels in an image are captured at the same time. By contrast, most low-cost cameras typically use CMOS sensors with a RS, capturing each row of pixels at a slightly different time instant. To develop high-precision localization methods, suitable for low-cost robots and for indoor navigation using consumer-grade cameras, the RS effect must be explicitly addressed in the design of estimation algorithms.

The (only few) methods that have been proposed to date for motion estimation using an IMU and a RS camera rely on the detection and tracking of point features in the scene. Point features are commonly used in motion-estimation methods, as they are abundant in many real-world environments, and well-established algorithms exist for their detection and tracking. In several man-made environments however, such as indoors and in urban areas, *straight-line* features are equally common. In this work, we present an algorithm that is able to use line features for motion estimation with a RS camera, either as an alternative to or in addition to point features.

As discussed in Section II, the subject of motion and structure estimation from line features has been extensively studied. However, almost all prior work assumes that a GS camera is used. This is a significant assumption, since it guarantees that straight lines in the scene project into straight lines in the image. Consequently, the first step of all methods for line-based motion estimation using GS cameras is to perform straight-line detection and line-fitting, in order to obtain the equations of image lines. When a RS camera is used, straight lines do *not* in general project into straight lines in the images, and therefore these methods are not applicable. If restrictive assumptions on the camera motion (e.g., constant velocity) are imposed, it is possible to obtain a parametric description of the projection of a straight line (see, e.g., [1]). However, such assumptions do not hold in general-motion cases, and thus imposing them would lead to loss of estimation precision.

The main contribution of this paper is a formulation of line-based visual-inertial motion estimation with a RS camera that addresses the above challenges. Specifically, we develop a measurement model for straight lines that is based on using the observations of *all pixels on the projection of a line individually*. This approach makes a line-fitting step to compute the equations of image lines unnecessary, and can thus be employed with either a RS or a global-shutter camera. This measurement model is used in conjunction with a novel minimal parameterization for straight lines. We demonstrate experimentally that this parameterization exhibits better linearity than parameterizations proposed in prior work, and is thus better suited for use in linearization-based estimators.

The new formulation of the line-measurement equations is general, and can be employed for estimation in a variety of settings (e.g., using either known or unknown lines), with either an extended Kalman filter (EKF) or an iterative-minimization method. We here assume that the positions and directions of the lines in the scene are not known *a priori*, and present a visual-inertial odometry algorithm based on the hybrid-EKF algorithm of [2]. In order to enable the use of a RS camera, we employ the formulation of [3], which makes it possible to use the RS-camera measurements *without* imposing any assumptions on the camera motion. Our simulation and experimental results demonstrate that the proposed method for using the line measurements yields

high-precision state estimates.

II. Related Work

We begin by discussing related work, divided into three areas:

a) Motion Estimation with a Rolling-Shutter Camera: The use of a RS camera, as opposed to a GS one, for motion estimation requires special treatment: with a RS camera each image row is captured at a slightly different time instant, and therefore from a different camera pose. Since including one state for each image row (the "exact" approach) is computationally intractable, existing methods employ some assumption about the nature of the camera trajectory [4]–[7]. By contrast, we here employ the formulation of [3], which imposes *no* assumptions on the form of the camera trajectory itself, and is thus able to model arbitrarily complex motions. Instead, it uses an approximate representation for the time-evolution of the estimation *errors* during the image readout time. Since these errors are typically small, this leads to only small modeling inaccuracies.

b) Structure from Motion and SLAM using Line Features: In both the computer vision and robotics communities, several approaches have been proposed for estimating the motion of camera and scene structure based on line features (see, e.g., [8]–[12] and references therein). However these approaches all employ GS cameras, and thus suffer from the limitations discussed in Section I. Similarly, the work of [13], which employs line-feature observations in conjunction with inertial measurements, also utilizes a GS camera. By contrast, motion estimation using straight-line features and RS camera is a less-explored topic. In [1] a bundle-adjustment method that employs constant-velocity motion assumptions is employed, while in [12] prior information about the lines' directions in space is assumed. By contrast, in our work none of these assumptions are necessary.

c) Line Measurement Model and Line Parameterizations: When a GS camera is used, the projection of a 3D line onto the image plane is a straight line. This property, which is employed in all prior work on motion estimation with lines and a GS camera, is no longer valid when a RS camera is used. To address this issue, we here propose a new way of using the "raw" pixel measurements belonging to line features, which is applicable with both types of cameras.

In addition to the way in which line measurements are processed, the parameterization of a line will also have great impact on the performance of any linearization-based estimator. Previous work has used Plücker coordinates [9] or a pair of vectors [14], [15] to represent 3D lines in space. However, these parameterizations are not minimal, and this overparameterization can lead to numerical issues in EKF-based estimators. We here propose a novel, minimal error parameterization for 3D lines, which has favorable characteristics. Specifically, it has inverse-depth properties, and is anchored in one of the camera poses from which the line was observed. These properties are desirable in EKF-SLAM, as discussed in [16]. Moreover, we demonstrate that the proposed parameterization has favorable linearity characteristics, through an analysis similar to that in [16].

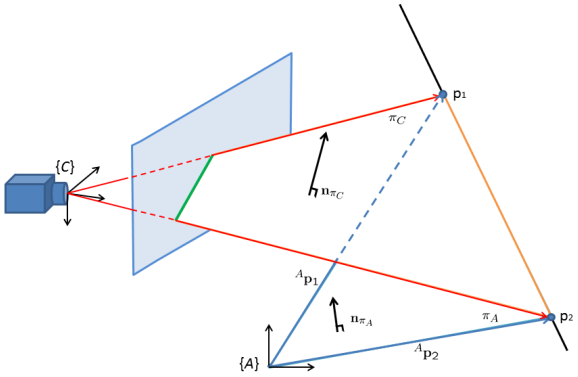

Fig. 1. Illustration of the line-projection geometry. The origin of the anchor frame $\{A\}$ and the line define a plane with normal vector $\mathbf{n}_{\pi_A}$, while the origin of the camera frame $\{C\}$ and the line define a plane with normal vector $\mathbf{n}_{\pi_C}$.

III. Line Parameterization

We begin by discussing the line parameterization we employ in our work, which we term the two-point inverse depth parameterization (TPIDP). For each line we employ an "anchor frame," $\{A\}$, which is a known, constant coordinate frame (e.g., the first frame from which the line was observed), based on which the parameterization is derived. Let the position and orientation of the anchor frame with respect to the global reference frame, $\{G\}$, be denoted as ${}^G\mathbf{p}_A$ and ${}^G_A\mathbf{R}$, respectively[1]. To present our proposed line parameterization, we note that when a line is observed from a camera frame $\{C\}$, the normal vector of the plane defined by the line and the camera optical center, $\mathbf{n}_{\pi_C}$, is the only quantity needed in order to derive the measurement equations (see Section IV-A). To describe the line parameters that we define, we proceed to obtain an expression for $\mathbf{n}_{\pi_C}$.

We start by considering two distinct points, $\mathbf{p}_1$ and $\mathbf{p}_2$ on the line. The unit vector normal to the plane defined by the origin of the camera frame and the line, expressed with respect to the camera frame, can be computed as:

$$ {}^C\mathbf{n}_{\pi_C} \sim {}^C\mathbf{p}_1 \times {}^C\mathbf{p}_2 $$

where $\sim$ denotes equality up to a normalizing scale factor. Using the equations relating the points' coordinates in $\{C\}$ with the points' coordinates in $\{A\}$, ${}^C\mathbf{p}_i = {}^C_A\mathbf{R}\,{}^A\mathbf{p}_i + {}^C\mathbf{p}_A$, $i = 1, 2$, we obtain:

$$ {}^C\mathbf{n}_{\pi_C} \sim {}^C_A\mathbf{R}\left({}^A\mathbf{p}_1 \times {}^A\mathbf{p}_2\right) + {}^C\mathbf{p}_A \times \left({}^C_A\mathbf{R}\left({}^A\mathbf{p}_2 - {}^A\mathbf{p}_1\right)\right) \tag{1} $$

[1]Notation: The preceding superscript for vectors (e.g., G in ${}^G\mathbf{p}$) denotes the frame of reference with respect to which quantities are expressed. ${}^X_Y\mathbf{R}$ is the rotation matrix rotating vectors from frame $\{Y\}$ to $\{X\}$. ${}^X\mathbf{p}_Y$ represents the origin of frame Y with respect to frame X. $\mathbf{I}$ represents the identity matrix, and $\mathbf{0}$ the zero matrix. Finally, $\hat{a}$ is the estimate of a variable a, and $\tilde{a} \doteq a - \hat{a}$ is the error of the estimate.

We now note that the term ${}^A\mathbf{p}_1 \times {}^A\mathbf{p}_2$ can be written as:

$${}^A\mathbf{p}_1 \times {}^A\mathbf{p}_2 = ||{}^A\mathbf{p}_1||\,||{}^A\mathbf{p}_2||\sin\langle {}^A\mathbf{p}_1, {}^A\mathbf{p}_2\rangle {}^A\mathbf{n}_{\pi_A}$$

where $\mathbf{n}_{\pi_A}$ is the unit vector normal to the plane defined by the line and the origin of $\{A\}$ (see Fig. 1). Using this result, and expressing the quantities in the above equation with respect to the global reference frame, $\{G\}$, we obtain:

$$\begin{aligned} {}^C\mathbf{n}_{\pi_C} \sim\ & {}^C_G\mathbf{R}\Big(||{}^A\mathbf{p}_1||\,||{}^A\mathbf{p}_2||\sin\langle {}^A\mathbf{p}_1, {}^A\mathbf{p}_2\rangle {}^G\mathbf{n}_{\pi_A} + \\ & \left({}^G\mathbf{p}_C - {}^G\mathbf{p}_A\right) \times \left({}^G_A\mathbf{R}\,{}^A\mathbf{p}_1 - {}^G_A\mathbf{R}\,{}^A\mathbf{p}_2\right) \Big) \\ \sim\ & {}^C_G\mathbf{R}\left({}^G\mathbf{n}_{\pi_A} + \left({}^G\mathbf{p}_C - {}^G\mathbf{p}_A\right) \times {}^G\mathbf{v}_\ell\right) \end{aligned} \tag{2}$$

where

$${}^G\mathbf{v}_\ell = \frac{1}{||{}^A\mathbf{p}_1||\,||{}^A\mathbf{p}_2||\sin\langle {}^A\mathbf{p}_1, {}^A\mathbf{p}_2\rangle} \left({}^G_A\mathbf{R}\,{}^A\mathbf{p}_1 - {}^G_A\mathbf{R}\,{}^A\mathbf{p}_2\right)$$

Let us now examine the terms appearing in (2). First, we have the camera position and rotation with respect to the global frame, ${}^G\mathbf{p}_C$ and ${}^C_G\mathbf{R}$. Second, we have the position of the origin of the anchor frame, ${}^G\mathbf{p}_A$, which is a known constant. Third, we have the vector ${}^G\mathbf{n}_{\pi_A}$, which is the unit vector formed by the origin of $\{A\}$ and the line, expressed with respect to the global frame. This vector depends on the line's position and orientation in space, and therefore it will constitute part of the line parameters. Finally, turning our attention to the vector ${}^G\mathbf{v}_\ell$, we note that this vector is a linear combination of ${}^A\mathbf{p}_1$ and ${}^A\mathbf{p}_2$ (expressed in $\{G\}$). Since both vectors lie in the plane normal to ${}^G\mathbf{n}_{\pi_A}$, ${}^G\mathbf{v}_\ell$ must also lie in this plane, and thus can be written as a linear combination of two basis vectors within this plane. We can therefore write ${}^G\mathbf{v}_\ell$ as:

$${}^G\mathbf{v}_\ell = a_1\,{}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_1 + a_2\,{}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_2 \tag{3}$$

where $\mathbf{w}_1$ and $\mathbf{w}_2$ are two known constant vectors. Note that, for any choice of these two vectors, both ${}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_1$ and ${}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_2$ lie in the plane normal to $\mathbf{n}_{\pi_A}$, and therefore these are two valid basis vectors, assuming they are linearly independent. In our work, to ensure linear independence, we select $\mathbf{w}_i = {}^G_A\mathbf{R}\left({}^A\mathbf{p}_i \times {}^A\mathbf{n}_{\pi_A}\right)$, $i = 1, 2$.

To summarize, a 3D line in our work is parameterized by the unit vector ${}^G\mathbf{n}_{\pi_A}$ and the parameters a_1, a_2. Since the unit vector has two degrees of freedom, this parameterization corresponds to a total of four degrees of freedom, as required for a 3D line. We note that since the norms of the vectors ${}^A\mathbf{p}_1$ and ${}^A\mathbf{p}_2$ (i.e., the "depth" of these points with respect to the anchor frame $\{A\}$) appear in the denominator of the expression for ${}^G\mathbf{v}_\ell$, the parameters a_1 and a_2 have units of "inverse depth", hence the name of our proposed parameterization.

Given this line parameterization, the vector normal to the plane defined by the camera optical center and the line is given by equations (2) and (3). How this normal vector is used in order to construct a measurement model in our EKF estimator is explained in the next section. As a final remark, we note that to ensure a minimal representation for the errors, the error-state for a given line parameterization $\mathbf{f}_L = [{}^G\mathbf{n}_{\pi_A}^T \quad a_1 \quad a_2]^T$ is defined as:

$$\tilde{\mathbf{f}}_L = \begin{bmatrix} k_1 & k_2 & \tilde{a}_1 & \tilde{a}_2 \end{bmatrix}^T$$

where $\tilde{a}_1$ and $\tilde{a}_2$ represent the errors in the estimates of a_1 and a_2, while k_1 and k_2 are used to define a minimal representation for the error in the unit vector ${}^G\mathbf{n}_{\pi_A}$. Specifically, to a first-order approximation, the relationship between the true and estimated unit vectors is given by:

$${}^G\mathbf{n}_{\pi_A} = {}^G\hat{\mathbf{n}}_{\pi_A} + \mathbf{B}_{\mathbf{n}_{\pi_A}} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{4}$$

where $\mathbf{B}_{\mathbf{n}_{\pi_A}}$ is a matrix whose columns are chosen to be two orthogonal vectors perpendicular to ${}^G\hat{\mathbf{n}}_{\pi_A}$.

IV. Estimator formulation

We now turn to the problem of using line measurements to track the motion of a platform equipped with an IMU and a RS camera in an unknown environment. The EKF-based estimator that we employ to track this state is a modification of the "hybrid" filter proposed in [3]. In our prior work, the hybrid estimator was employed in conjunction with point-feature measurements only. Here, in addition to point features, we utilize the observations of straight-line features in the environment. Since the details of the hybrid estimator have been presented in prior work, we here briefly describe the structure of the estimator (see Algorithm 1), and refer the reader to [3] for further details.

Our goal is to estimate the pose of the moving platform with respect to a global coordinate frame $\{G\}$. To this end, we affix a "body" coordinate frame, $\{B\}$, to the IMU, and track the motion of this frame with respect to $\{G\}$. We here assume that the camera intrinsics and the extrinsic calibration between the camera and IMU are known through prior offline calibration, but stress that these are not strict requirements for our work.

The state vector of the EKF at time-step k is given by:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \mathbf{x}_{B_1}^T & \cdots & \mathbf{x}_{B_m}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_s^T \end{bmatrix}^T \tag{5}$$

where $\mathbf{x}_{E_k}$ is the "evolving state" of the IMU, comprising the current body-frame position, velocity, orientation, as well as the time-varying IMU biases; the states $\mathbf{x}_{B_j}$, $j = 1 \ldots m$ are the body states corresponding to the time instants the past m images were recorded; and $\mathbf{f}_i$, for $i = 1, .., s$ are the currently visible features. These features include *both* points, which are being represented in inverse-depth parameterization (IDP) [18], and straight lines, which are represented in the TPIDP parameterization described in the preceding section.

When an IMU measurement is received, it is used to propagate the evolving state and covariance. On the other hand, when a new image is received, the sliding window of states is augmented with a new state. Note that each image is sampled over a time interval of non-zero duration (the rolling shutter readout time). By convention, we here consider that the timestamp associated with each image corresponds to the time instant the *middle* row of the image is captured.

Algorithm 1 Hybrid EKF algorithm

Propagation: Propagate the state vector and the state covariance matrix using the IMU readings.

Update: When camera measurements become available:

- Augment the sliding window with a new state, and begin image processing.
- For each feature track that is complete after m or fewer images, do the following
  - Obtain an estimate for the feature's parameters (IDP for points or TPIDP for lines) using all its observations.
  - Compute the residuals associated with all the feature measurements and their Jacobians, and apply the method of [17] to remove the effect of the feature-estimate error.
  - Perform a Mahalanobis-distance gating test.
- For the features included in the state vector, compute the residuals and measurement Jacobians.
- Perform an EKF update using all the features.
- Initialize into the state vector features that are still actively tracked after m images.

State Management:

- Remove from the state vector features that are no longer tracked.
- Remove all sliding-window states that have no active feature tracks associated with them.

Therefore the state corresponding to each image in the filter represents the body-frame state at that time instant.

The images are processed to extract and match point and line features, which are processed in one of two ways: if a feature's track is lost after m or fewer images, it is used to provide constraints involving the poses of the sliding window. For this purpose, the multi-state-constraint method of [17], [19] is employed, which makes it possible to use the feature measurements without including the feature in the EKF state vector. On the other hand, if a feature is still being tracked after m frames, it is initialized in the state vector and any subsequent observations of it are used for updates as in the EKF-SLAM paradigm.

At each time step, the hybrid filter processes a number of features with each of the two approaches. For each feature the appropriate residuals and Jacobian matrices are computed, and a Mahalanobis-distance gating test is performed. All the features that pass the gating test are then employed for an EKF update. At the end of the update, features that are no longer visible and old sliding-window states with no active feature tracks associated with them are removed. Note that, to ensure the correct observability properties of the linearized system model, and thus improve the estimation accuracy and consistency, the hybrid filter employs fixed linearization points for each state [19].

A. Line measurement model

We now describe the formulation of the measurement model that we employ for the straight-line features detected in the images. In prior work, the standard practice is to employ line fitting in the images to obtain the equations of the lines in the images, and to subsequently relate these equations to the configuration of the line and camera in space. However, when a moving RS camera observes straight lines, these are no longer guaranteed to project into straight lines in the images. Therefore, our approach completely forgoes the line-fitting step. Instead, all the pixels that are deemed to belong to a straight line (see Section VI for a description of our line-detection strategy) are directly used to define measurement residuals for the EKF update.

Specifically, let us consider an image point that belongs to the projection of a 3D line, and lies in the i-th row of the RS image. If we denote the normalized coordinates of the point as (u_i, v_i), then the following equation is satisfied:

$$\begin{bmatrix} u_i & v_i & 1 \end{bmatrix} {}^{C_i}\mathbf{n}_{\pi_C} = 0$$

where C_i is the camera pose at time t_i, i.e., at the time instant the i-th image row was recorded, and ${}^{C_i}\mathbf{n}_{\pi_C}$ is defined by equations (2) and (3). We note that the above equation involves the noise-less image projection coordinates, as well as the true line parameters. In any real-world estimation problem, however, the measurements will be corrupted by noise, and only estimates of the line parameters are available. Therefore, using these quantities, we obtain a *residual*:

$$\begin{aligned} r_i &= \begin{bmatrix} \mathbf{z}_i^T & 1 \end{bmatrix} {}^{C_i}\hat{\mathbf{n}}_{\pi_C} \\ &= \begin{bmatrix} \mathbf{z}_i^T & 1 \end{bmatrix} {}^{C}_{G}\hat{\mathbf{R}}(t_i) \left({}^{G}\hat{\mathbf{n}}_{\pi_A} + \left({}^{G}\hat{\mathbf{p}}_C(t_i) - {}^{G}\mathbf{p}_A \right) \times {}^{G}\hat{\mathbf{v}}_\ell \right) \end{aligned} \tag{6}$$

where

$${}^{G}\hat{\mathbf{v}}_\ell = \hat{a}_1 {}^{G}\hat{\mathbf{n}}_{\pi_A} \times \mathbf{w}_1 + \hat{a}_2 {}^{G}\hat{\mathbf{n}}_{\pi_A} \times \mathbf{w}_2$$

In the above, $\mathbf{z}_i$ represents the normalized-image measurement vector,

$$\mathbf{z}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} + \boldsymbol{\eta}_i$$

where $\boldsymbol{\eta}_i$ is the image measurement noise, modelled as zero-mean, Gaussian, with covariance matrix $\sigma^2\mathbf{I}_2$, and $\hat{\cdot}$ denotes the estimated value of a quantity. By employing linearization, we can express the residual, up to a first order approximation, as a function of the errors in the state estimates, the line-parameter estimates, and the measurement noise:

$$r_i \simeq \mathbf{H}_{\boldsymbol{\theta}} \tilde{\boldsymbol{\theta}}_B(t_i) + \mathbf{H}_{\mathbf{p}} {}^{G}\tilde{\mathbf{p}}_B(t_i) + \mathbf{H}_{\mathbf{f}} \tilde{\mathbf{f}}_L + \boldsymbol{\Gamma}_i \boldsymbol{\eta}_i \tag{7}$$

where $\tilde{\boldsymbol{\theta}}_B(t_i)$ and ${}^{G}\tilde{\mathbf{p}}_B(t_i)$ are the errors in the orientation and position estimates at time instant t_i, $\mathbf{H}_{\boldsymbol{\theta}}$ and $\mathbf{H}_{\mathbf{p}}$ are the corresponding Jacobians, $\tilde{\mathbf{f}}_L$ is the error in the estimates of the line parameters and $\mathbf{H}_{\mathbf{f}}$ is the corresponding Jacobian, and $\boldsymbol{\Gamma}_i$ is the Jacobian of r_i with respect to $\boldsymbol{\eta}_i$.

Having defined a residual in (6) and its linearized expression in (7), we can now directly apply the method of [3] for performing an EKF update. Specifically, for each image

TABLE I
LINEARITY INDEX OF DIFFERENT LINE PARAMETERIZATIONS

Parameterization	Average Linearity Index
TPIDP	0.000049
OIDP	0.0013
ODP	0.0149
Roberts	0.0045
Ortho	0.0017
Cayley	0.0055

one body state is included in the state vector of the EKF, corresponding to the time instant the middle row of the image was captured. In order to be able to compute residuals for pixels captured in any possible row of the image, we employ IMU propagation within the image readout interval, to compute the state estimates corresponding to all N image rows, i.e., the state estimates at the N different time instants given by:

$$t_i = t_o + \frac{it_r}{N}, \quad i \in \big[-\frac{N}{2}, \ \frac{N}{2}\big] \tag{8}$$

where t_r is the image-readout time of the RS sensor. These state estimates are employed for computing the residuals shown in (6). Note that, since our IMU propagation method does not make any assumption about the form of the trajectory [19], we are able to compute the residuals *without* any assumptions on the trajectory (e.g., without assuming a constant-velocity model).

On the other hand, assumptions are necessary when computing the EKF Jacobians. To see why, note that the linearized expression in (7) involves the errors of the state estimates at time instant t_i, not at t_o, which is the time instant represented in the EKF state vector. Since it is computationally intractable to include in the EKF state vector one body-state corresponding to each image row, we must approximate the errors $\tilde{\boldsymbol{\theta}}_B(t_i)$ and ${}^G\tilde{\mathbf{p}}_B(t_i)$ as a function of the errors $\tilde{\boldsymbol{\theta}}_B(t_o)$ and ${}^G\tilde{\mathbf{p}}_B(t_o)$. To this end, either a zero-order of a first-order linear approximation can be used, as explained in [3].

V. SIMULATION RESULTS

In this section we present simulation results which demonstrate the properties of the proposed line parameterization compared to alternatives, as well as the performance of the proposed visual-inertial odometry estimator which processes both point and line measurements.

A. Comparison of Line Parameterizations

We first examine the linearity characteristics of the proposed line parameterization. We note that, as discussed in [16], [18], one of the key properties a state parameterization must have, when used in an EKF estimator, is that it has to result in measurement models with "small" nonlinearity. This is necessary, as large nonlinearities result in non-Gaussian errors, and violate the small linearization-error assumptions of the EKF, leading to poor performance. Hence, we here compare the linearity of the proposed TPIDP against the following parameterizations: (i) the orientation-depth parameterization (ODP) employed in [13], (ii) the orientation-depth parameterization with the depth represented by its inverse, which we term orientation-inverse depth (OIDP), (iii) the Cayley parameterization of [20], (iv) the Roberts parameterization [21], and (v) the orthonormal representation (Ortho) of [22]. Note that, while additional parameterizations have been proposed in the literature (e.g., Plücker coordinates, two-point parameterizations), we are only interested in minimal-error-representation parameterizations, due to their numerical advantages in EKF-based estimation.

The metric that we use to measure linearity is the measurement-linearity index defined in [16]. Specifically, if we denote by $\mathbf{y}$ the $n \times 1$ vector containing the line parameters and the poses from which this line was observed, the linearity index is defined as:

$$\lambda = \|\epsilon\|_2, \ \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_q \end{bmatrix}, \ \epsilon_i \simeq \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} H_{ijk}P_{jk} \tag{9}$$

where q is the number of pixel observations corresponding to the line, $H_{ijk} = \frac{\partial^2 r_i}{\partial \mathbf{y}_j \partial \mathbf{y}_k}$ is the second-order derivative of the measurement residual function (6), and P_{jk} is the (j,k) element of the joint covariance matrix of the errors in $\mathbf{y}$. Intuitively, the above metric evaluates the magnitude of the second-order and higher terms in the measurement model, which are ignored during the EKF's linearization.

To evaluate the linearity of the different parameterizations, we perform Monte-Carlo simulations. In each trial, we randomly generate a line feature and m camera poses, with m randomly selected between 8 and 15. The joint covariance matrix of the camera poses is selected to be identical to the joint covariance matrix observed at a certain time in one of our real-world experiments. For each randomly generated configuration we generate simulated line-pixel measurements, and employ them to estimate the line parameters using least-squares minimization. Subsequently, following the approach in [23], we compute the joint covariance matrix of the line and camera poses, and use it to compute the linearity index λ as shown in (9).

In Table I we present the average linearity index, averaged over the 6000 Monte-Carlo trials, for all the compared parameterizations. Recall that in this table smaller values indicate smaller nonlinearity, and are preferable. We can clearly see that the linearity index associated with the TPIDP parameterization is substantially smaller than the index associated with all other parameterizations. Specifically, it is approximately 3.7 percent of the linearity index of the second-best parameterization, which is the OIDP. This result demonstrates that the TPIDP results in a measurement model that is "closer to linear", and thus we expect to obtain improved performance when using this parameterization in an EKF-based estimator. Moreover, this result indicates that the covariance matrix computed by linearization of the measurement models using TPIDP provides a more accurate

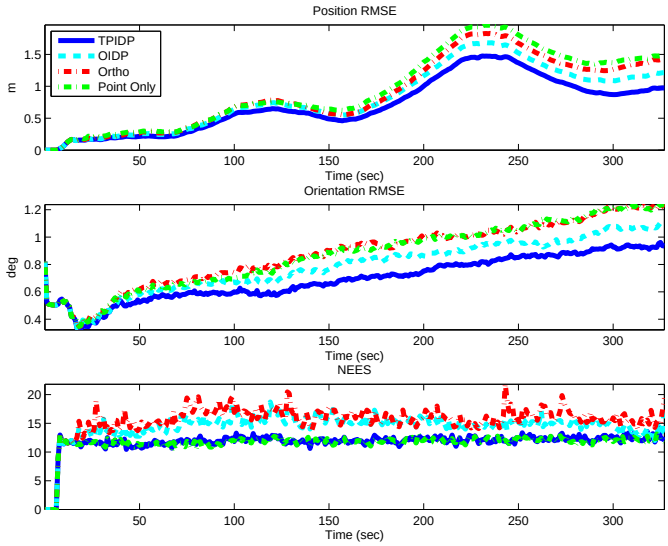

Fig. 2. Hybrid EKF simulations: average NEES and RMSE over 100 Monte-Carlo trials

description of the uncertainty.

B. Comparison of the Hybrid EKF With Different Parameterizations

We next examine the effect of using straight-line measurements, and parameterizing these lines in different ways, in the hybrid EKF estimator. To this end, we perform Monte-Carlo simulations in a simulation environment that emulates a real-world dataset. Specifically, in each Monte-Carlo trial, we generate a ground-truth trajectory that follows the trajectory of the actual dataset, and in each image we generate simulated point and straight-line feature measurements with characteristics (noise levels, feature numbers, track lengths) as in the real-world dataset. The trajectory's length was 402 m, lasting about 328 s. The average number of point features per image is 30, while the average number of line features is 15, and each line consists of 20 pixels on average. We process the data by the following methods: (i) the hybrid EKF estimator that uses point features only, (ii) the proposed hybrid EKF that uses both points and lines, using the TPIDP representation for the lines included in the state, (iii) the hybrid EKF using points and lines with the OIDP representation, and (iv) the hybrid EKF using points and lines with the orthonormal representation of [22]. The three line parameterizations used here are the best three parameterizations in terms of linearity, as shown in the preceding tests.

The metrics we use to evaluate the performance of the different methods are (i) the root-mean square error (RMSE) of the orientation and position estimates, and (ii) the normalized estimation error squared (NEES) for the IMU state consisting of the orientation, position, and velocity. The RMSE gives us a measure of the accuracy of the estimator, while the NEES provides us with a measure of consistency [19], [24]. Specifically, for zero-mean Gaussian errors the NEES should equal the dimension of the error-state, i.e., 9 in this case.

TABLE II
AVERAGE RMSE AND NEES FOR FIG. 2

Simulation Settings	Position RMSE (m)	Orientation RMSE (deg)	NEES
TPIDP	0.7059	0.6964	11.8096
OIDP	0.8192	0.7875	14.4869
Ortho	0.8947	0.8793	15.5772
Points Only	0.9554	0.8699	11.6510

Larger values indicate that there exist unmodelled errors (e.g., linearization errors) in the estimator, and that the covariance matrix reported by the EKF underestimates the actual magnitude of the estimation errors. Examining both metrics provides us with a more complete picture of the estimator's performance.

Both the RMSE and NEES are averaged over 100 Monte-Carlo trials, and the results are plotted over time in Fig. 2. Table II lists the average values over all Monte-Carlo trials and all time instants. From these results, we can first observe that, as expected, using line features in addition to point features provides additional information to the estimator, and leads to lower estimation errors. Moreover, we can clearly see that the TPIDP parameterization results in both lower estimation errors, as well as in lower NEES, compared to the alternative parameterizations examined. These results agree with the linearity-index results presented earlier, and demonstrate the advantages of the proposed TPIDP representation for EKF-based estimation using line features.

We note that the NEES values computed using all three line parameterizations, as well as for the point-only EKF, are higher than the "ideal" value of 9. While TPIDP performs better than the other parameterizations, it also appears to be mildly inconsistent. We attribute this result primarily to the small number of features used. Being able to track only a small number of features, which is typical of low-texture indoor environments (such as office environments with textureless walls), results in increased errors and thus more pronounced nonlinearities in all the methods tested.

VI. REAL-WORLD EXPERIMENT

In addition to the simulation tests, we also tested our proposed approach in a real-world experiment, which was conducted using the sensors of a Nexus 4 device. The experiment was conducted in an indoor office area of the UCR Engineering Building, where line features are the most visually dominant ones. The device was hand-held by a person walking the halls and stairwells between three floors of the building during the experiment. For point-feature extraction and matching, Shi-Tomasi features are used [25], and matched by normalized cross-correlation. To detect and match line features, we first use the Canny edge detector [26] to identify edge pixels. The normalized coordinates of these pixels are computed using the pre-computed calibration parameters, and subsequently we perform partial RS compensation. Specifically, we employ the rotation estimates from the IMU to remove the rotational component of the RS distortion. The resulting compensated coordinates

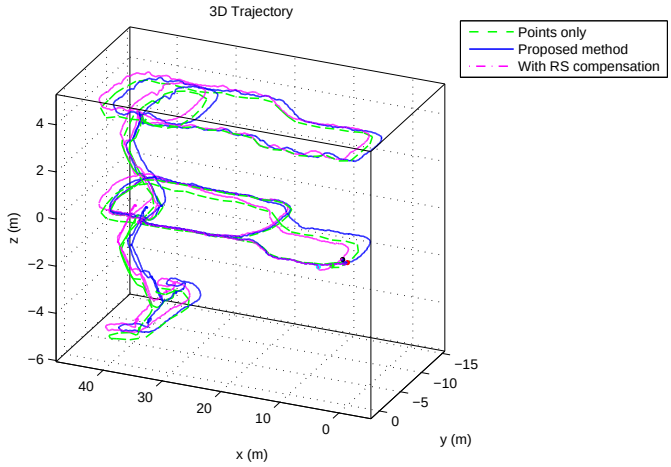

Fig. 3. Real world experiment: Estimated trajectory when (i) using point features alone, (ii) using both points and straight lines with the proposed method, and (iii) using both points and straight lines, where partial RS compensation (for rotation only) is employed for the line features. The red dot represents the start of the trajectory, while the green dot the end-point of the estimated trajectory using points only, the black dot the estimated end-point with the proposed method, and the cyan dot the end-point for the method with partial RS compensation.

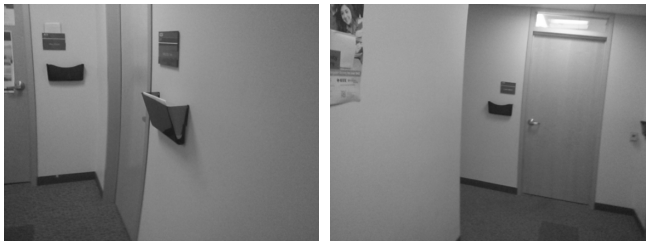

Fig. 4. Sample images recorded during the experiment.

are employed for straight-line detection using a split-and-merge approach. We point out that, in our proposed method, compensation is performed *only* to aid in detecting straight lines in the environment, and the compensated coordinates are *not* used for EKF updates. For line matching, a template is generated at the midpoint of each line, and matching is performed by normalized-cross correlation.

In this experiment, an average of 29.71 point features and 16 line segments are extracted per image. The trajectory length of the experiment is approximately 400 m. The IMU sample rate is 200 Hz, while the images are captured at 22 Hz (sample images are shown in Fig. 4). All the data are post-processed off-line on a desktop computer, to enable comparing the performance of different approaches. In this test, we are comparing (i) the hybrid EKF estimator using point features only, (ii) the proposed hybrid EKF using point and line features, and (iii) the hybrid EKF using point and line features concurrently, but using the RS-compensated coordinates for the lines (computed as described above), and processing them as if they come from a global-shutter camera. Since the ground truth for the entire trajectory is not available, the fact that the trajectory starts and ends at the same point is used to evaluate the performance of the algorithms.

The 3D trajectory estimated by each approach is shown in Fig. 3. The computed final position errors for each of the methods are 1.48 m for the proposed method (corresponding to 0.37% of the travelled distance), 3.05 m for the hybrid EKF that uses point measurements only and 4.05 m for the hybrid EKF that uses both points and lines with partial RS compensation. Thus we can clearly see that processing line features, especially in environments that are poor in point features and rich in straight lines, can lead to measurable performance gains in vision-aided inertial navigation, compared to algorithms that only employ point features. Furthermore, we notice that simply performing RS-compensation, and treating the resulting measurements as if they come from a global shutter camera, leads to worse performance than the proposed approach. This is due to the fact that RS compensation can only be reliably performed for the rotational effects, while the effects of the camera's translational motion during the image readout time cannot be exactly compensated for.

VII. Conclusion

In this paper, we present two main contributions. First, we propose a novel parameterization for 3D lines, which is shown to exhibit better linearity properties than alternatives, and thus is better suited for use in linearization-based estimators such as the EKF. Second, we describe a method for processing the observations of lines that is suitable for use with RS sensors, which are found in the majority of low-cost cameras. Our approach forgoes line-fitting, and relies on processing the measurement of each line pixel individually, thus avoiding assumptions on the shape of the projection of a line in the image. These two contributions are employed in conjunction with the hybrid EKF estimator for visual-inertial odometry [3]. The simulation and experimental results we present demonstrate that the proposed approach leads to an improvement in performance, compared to using point features alone, and that the proposed line parameterization outperforms those proposed in prior work.

Acknowledgement

This work was supported by the National Science Foundation (grant no. IIS-1117957, IIS-1253314 and IIS-1316934).

References

[1] O. A. Aider, A. Bartoli, and N. Andreff, "Kinematics from lines in a single rolling shutter image," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, Jun 2007, pp. 1 – 6.

[2] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, Jul. 2012.

[3] M. Li. and A. I. Mourikis, "Vision-aided inertial navigation with a rolling-shutter camera," *International Journal of Robotics Research*, vol. 33, no. 11, pp. 1490–1507, 2014.

[4] J. Hedborg, P. Forssen, M. Felsberg, and E. Ringaby, "Rolling shutter bundle adjustment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, June 2012, pp. 1434 –1441.

[5] M. Li, B. Kim, and A. I. Mourikis, "Real-time motion estimation on a cellphone using inertial sensing and a rolling-shutter camera," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 4697–4704.

[6] C. Guo, D. Kottas, R. DuToit, A. Ahmed, R. Li, and S. Roumeliotis, "Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[7] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, Jun. 2013, pp. 1360 – 1367.

[8] C. J. Taylor and D. J. Kriegman, "Structure and motion from line segments in multiple images," *IEEE Transacions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 11, pp. 1021 – 1032, 1995.

[9] A. Bartoli and P. Sturm, "The 3D line motion matrix and alignment of line reconstructions," *International Journal of Computer Vision*, vol. 57, no. 3, pp. 159–178, 2004.

[10] M. Chandraker, J. Lim, and D. Kriegman, "Moving in stereo: Efficient structure and motion using lines," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Kyoto, Japan, October 2009, pp. 1741 – 1748.

[11] J. Witt and U. Weltin, "Robust stereo visual odometry using iterative closest multiple lines," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Nov 2013, pp. 4164 – 4171.

[12] A. Elqursh and A. M. Elgammal, "Line-based relative pose estimation." in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Providence, RI, June 2011, pp. 3049–3056.

[13] D. G. Kottas and S. I. Roumeliotis, "Efficient and consistent vision-aided inertial navigation using line observations," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 1540 – 1547.

[14] P. Smith, I. Reid, and A. Davison, "Real-time monocular SLAM with straight lines," in *Proceedings of the British Machine Vision Conference*, Edinburgh, Sep 2006, pp. 3.1–3.10.

[15] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM," in *Proceedings of the British Machine Vision Conference*, Edinburgh, Sep 2006, pp. 2.1–2.10.

[16] J. Sola, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parameterization on monocular EKF-SLAM with points and lines," *International Journal of Computer Vision*, vol. 97, no. 3, pp. 339–368, 2012.

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

[18] J. Montiel, J. Civera, and A. Davison, "Unified inverse depth parametrization for monocular SLAM," in *Proceedings of Robotics: Science and Systems*, Philadelphia, PA, Aug. 2006, pp. 81–88.

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

[20] L. Zhang and R. Koch, "Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 904–915, 2014.

[21] K. Roberts, "A new representation for a line," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Ann Arbor, MI, Jun 1988, pp. 635 – 640.

[22] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[23] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation: Supplemental materials," 2012, www.ee.ucr.edu/∼mli/SupplMaterialsRSS2012.pdf.

[24] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, 2001.

[25] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, WA, June 1994, pp. 593–600.

[26] J. Canny, "A computational approach to edge detection," *IEEE Transacions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.