Proc. 7th IEEE Workshop on Applications of Computer Vision / IEEE Workshop on Motion and Video Computing (WACV/MOTION 2005), January 2005, pp. 221--228

Epipolar Constraints for Vision-Aided Inertial Navigation

David D. Diel
Massachusetts Institute of Technology
Email: ddiel@mit.edu

Paul DeBitetto
Draper Laboratory
Email: pdebitetto@draper.com

Seth Teller
Massachusetts Institute of Technology
Email: teller@csail.mit.edu

***Abstract*— This paper describes a new method to improve inertial navigation using feature-based constraints from one or more video cameras. The proposed method lengthens the period of time during which a human or vehicle can navigate in GPS-deprived environments. Our approach integrates well with existing navigation systems, because we invoke general sensor models that represent a wide range of available hardware. The inertial model includes errors in bias, scale, and random walk. Any purely projective camera and tracking algorithm may be used, as long as the tracking output can be expressed as ray vectors extending from known locations on the sensor body.**

A modified linear Kalman filter performs the data fusion. Unlike traditional SLAM, our state vector contains only inertial sensor errors related to position. This choice allows uncertainty to be properly represented by a covariance matrix. We do not augment the state with feature coordinates. Instead, image data contributes *stochastic epipolar constraints* over a broad baseline in time and space, resulting in improved observability of the IMU error states. The constraints lead to a relative residual and associated relative covariance, defined partly by the state history. Navigation results are presented using high-quality synthetic data and real fisheye imagery.

I. Navigation Problem

An Inertial Measurement Unit (IMU) is a common component in modern navigation systems. A typical IMU contains three accelerometers and three gyroscopes that provide information about the motion of a moving body [1]. Unfortunately, the process of extracting body position estimates from IMU output leads to significant drift over time. The Global Positioning System (GPS) provides complementary, absolute position information. However, circumstances such as sky occlusion, hardware failure, and war may disallow GPS signals. Therefore, we turn to vision as an alternative source of information.

We would like to navigate in places where people go on foot or in a wheeled vehicle. Examples include warehouses, factories, offices, homes, city streets, suburbs, highways, rural areas, forests, and caves. These environments happen to share some important visual characteristics: 1) Most of the scene remains stationary with respect to the planet's surface; and 2) For typical scenes, the ratio of body velocity to scene depth lies within a limited range. Other environments that meet these criteria include the sea floor and unexplored planets. The sky, outer space, and most seascapes do not fall in this category.

Variables
t - time
y_{ij} - pixel radiance
ϵ - relatively small number
τ - interval of time (fixed)
$\boldsymbol{b}$ - bias
$\boldsymbol{c}_{ij}$ - camera ray
$\boldsymbol{f}$ - specific force
$\boldsymbol{g}$ - gravity
$\boldsymbol{h}$ - measurement gain
$\boldsymbol{k}$ - Kalman gain
$\boldsymbol{n}$ - white noise
$\boldsymbol{s}$ - scale
$\boldsymbol{u}_{ij}$ - image coordinate
$\boldsymbol{x}$ - body state
$\boldsymbol{z}$ - feature ray
$\boldsymbol{\theta}$ - Euler angles

$\boldsymbol{\rho}$ - imaging parameters
$\boldsymbol{R}$ - rotation matrix
$\boldsymbol{\Lambda}$ - covariance matrix
$\boldsymbol{\Phi}$ - transition matrix
Other Notation
$\diamond_T$ - translation part
$\diamond_R$ - rotation part
$\diamond^{\mathrm{T}}$ - transpose
$\dot{\diamond}$ - 1^{st} temporal derivative
$\ddot{\diamond}$ - 2^{nd} temporal derivative
$\vec{\diamond}$ - unit magnitude
$\bar{\diamond}$ - contains error
$\tilde{\diamond}$ - residual
$\hat{\diamond}$ - estimated
$\Delta\diamond$ - change
$\delta\diamond$ - error
$\diamond[\,]$ - discrete samples

Fig. 1. Symbols and notation. Scalars are regular italic; vectors are bold lowercase; and matrices are bold uppercase. Variables may vary with time unless noted. The $\diamond$ symbol is a placeholder and subscripts $_{ij}$ indicate association with an array.

The goal of navigation is to find the transformations $\boldsymbol{x}[t]$ that relate camera poses to a static reference frame. In a vision-only context, Chiuso et al. have developed a causal (history-based), real-time algorithm to solve for 3D relative scene geometry, while addressing scale ambiguity [2]. Most of their discussion applies here, but we utilize the known characteristics of an inertial sensor in our development. Instead of tackling the Structure From Motion (SFM) problem directly, we aim to navigate first and leave scene geometry to be calculated later.

There exists a critical difference between self-localization and relative target seeking. When the goal is relative positioning for grasping, assembly, or impact, a convergent solution can be derived [3]. A visible target needs no absolute reference frame. However, if we want geo-referenced navigation coordinates in the absence of landmarks, then our sensor combination will face performance limitations. The only source of absolute position information is the planetary gravity potential, and this potential is ambiguous over each iso-layer (the ocean surface is one of them). In essence, the system we have described cannot converge to the actual trajectory. So, we will present a *transient* solution that aims to reduce drift.

II. Approach

Several ideas from machine vision have potential relevance to inertial navigation. The list would certainly include stereo vision, optical flow, pattern recognition, and tracking of points, curves, and regions. Each of these methods produce a different kind of information. Stereo vision may produce dense relative depth estimates within a limited range. Optical flow can produce dense motion fields. Pattern recognition can locate the direction to a unique landmark, which may be linked to absolute coordinates. And finally, tracking can offer data association over multiple frames.

The scope of relevance can be limited by considering system requirements and constraints. We want an automatic method that is robust to visual distractions such as occlusion, variation in lighting, and ambiguities. It should work with one or more cameras, within reasonable computational limits, given no prior knowledge of the environment. These choices bring the focus to either flow-based or tracking-based approaches.

A. Tracking vs. Flow

Both optical flow and tracking provide local estimates of image motion. As a camera translates, light rays from the environment slice through a theoretical unit sphere centered at the camera's focus. Patterns of light appear to flow outward from a point source, around the sides of the sphere, and into a drain at the opposite pole. If the camera also rotates, then a vortex-like flow is superimposed on the flow induced by translation. Despite the naming conventions, both optical flow and tracking are supposed to track the flow. The difference between them lies in the selection of discrete coordinates. Optical flow calculations are defined on a discrete regular mesh, with the assumption of underlying spatial and temporal continuity. Tracking methods tend to utilize discrete particles that retain their identity over a broad baseline in time.

Our method tracks sparse independent corner features for the following reasons: 1) To avoid the texture ambiguity associated with optical flow; 2) To gain leverage against the accumulation of inertial drift over time; 3) To facilitate detection of visual distractions; and 4) To bound computational requirements.

By appealing to feature tracking in general, most assumptions about the visual context can be avoided. For example, one could use infrared imagery to handle low light conditions. Landmarks such as corporate logos and barcodes may provide useful information when they are available, but none are required. Mixed resolutions and dropped frames are also handled. The only requirements are a static electromagnetic environment and sensors selective to an appropriate frequency band.

B. Data Fusion

Probably the most common approach to combining feature observations with inertial data is Simultaneous Localization and Mapping (SLAM) [4][5][6]. SLAM takes on the burden of mapping and the computational complexity of map maintenance in exchange for maximal information usage. Nearly all SLAM solutions are based on the Extended Kalman Filter (EKF) [7], which scales quadratically with the number of states. Extensions such as *Atlas* [8] and *Fast-SLAM* [9] have been developed to manage large maps, but the benefits of mapping may not justify the computational cost. Furthermore, the EKF may be far from optimal due to its linear approximations.

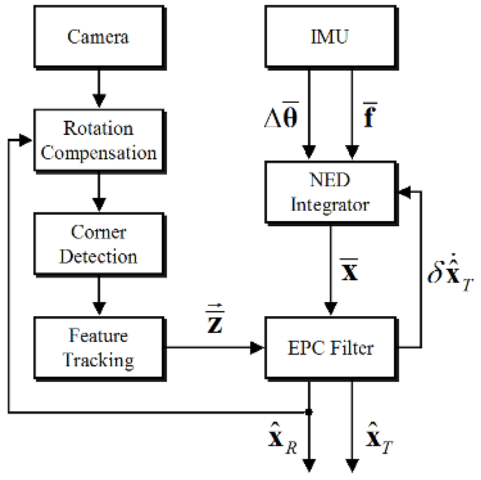

Fig. 2. Proposed block diagram showing sensor fusion in the EPC Filter.

Our approach toward data fusion focuses on the interpretation of visual measurements. As in other methods, we propagate the IMU error state, apply visual updates, and remove translation error. Uncertainty is represented by a covariance matrix. However, our filter does not maintain states for each feature, but instead treats visual measurements as independent *stochastic constraints* on the camera position. Specifically, out-of-plane violations of the epipolar constraint (EPC) are fed into a linear Kalman filter as projected residuals. By disregarding in-plane information, the measurement equation avoids explicit dependency on scene depth. In this framework, computation and memory requirements scale linearly with the number of visible features. For related work in vision-only navigation, see [10][11][12].

C. Gyroscope Reliance

Modern gyroscopes (gyros) can be trusted to provide accurate orientation estimates for periods up to several minutes [13]. For some hardware combinations, gyro accuracy exceeds imaging accuracy by orders of magnitude. Consider a typical wide-field video camera with a pixel separation of 0.25 degrees. A common flight control gyro can maintain similar angular precision for about 10 minutes with no visual assistance. Thus, our method relies on the gyros to compensate for camera rotation, but image data is not used to improve the gyros.

III. IMU Definition

We begin with a simplified strapdown[1] inertial sensor model in a static North-East-Down (NED) visual frame. A strapdown IMU is designed to measure specific force and orientation changes in its own body frame. We assume ideal gyros, and noisy accelerometers, as described in Section II. The gyros output small changes $\Delta\boldsymbol{\theta}[t]$, which can be integrated to find an associated direction cosine matrix $\boldsymbol{R}(t)$. The visual frame moves with the planet, so the planetary rotation rate should be subtracted from the gyro output during integration. The accelerometers measure a non-trivial combination of acceleration and gravitation.

$$\boldsymbol{f} = \boldsymbol{R}^{-1}\left(\ddot{\boldsymbol{x}}_T - \boldsymbol{g}_{app}\right) \tag{1}$$

Note the apparent gravitation term, which has an approximate value of $9.8\frac{m}{s^2}$ near the Earth's surface. To gain more significant digits, planetary rotation effects in the NED frame must be taken into account. These effects depend strongly on latitude and altitude, and weakly on velocity. We use the gradient of the second-order spherical harmonic defined in WGS 84 to model Earth's gravitational field [14][15], and add higher-order terms when they are justified by the sensor accuracy.

Assuming the apparent gravity model accuracy exceeds that of our accelerometers, the primary sources of measurement error are bias, scale and random walk.

$$\bar{\boldsymbol{f}} = \left(\boldsymbol{I} + \text{diag}\left(\delta\boldsymbol{s}\right)\right)\boldsymbol{f} + \delta\boldsymbol{b} + \boldsymbol{n}_w \tag{2}$$

Rewriting as an acceleration measurement yields

$$\begin{aligned}\bar{\ddot{\boldsymbol{x}}}_T &= \boldsymbol{R}\bar{\boldsymbol{f}} + \boldsymbol{g}_{app}\\ &= \left(\ddot{\boldsymbol{x}}_T - \boldsymbol{g}_{app}\right) + \boldsymbol{R}\left(\text{diag}\left(\delta\boldsymbol{s}\right)\boldsymbol{f} + \delta\boldsymbol{b} + \boldsymbol{n}_w\right) + \boldsymbol{g}_{app}\\ &= \ddot{\boldsymbol{x}}_T + \boldsymbol{R}\text{diag}\left(\boldsymbol{f}\right)\delta\boldsymbol{s} + \boldsymbol{R}\delta\boldsymbol{b} + \boldsymbol{n}_w\end{aligned} \tag{3}$$

Therefore, omitting the second order effect of $\text{diag}\left(\delta\boldsymbol{f}\right)\delta\boldsymbol{s}$, the error process may be defined

$$\delta\ddot{\boldsymbol{x}}_T \equiv \bar{\ddot{\boldsymbol{x}}}_T - \ddot{\boldsymbol{x}}_T = \boldsymbol{R}\text{diag}\left(\bar{\boldsymbol{f}}\right)\delta\boldsymbol{s} + \boldsymbol{R}\delta\boldsymbol{b} + \boldsymbol{n}_w \tag{4}$$

which leads to a Linear-Time-Varying (LTV) state-space model for the translation error. Note the nonlinearity with respect to external inputs $\boldsymbol{R}$ and $\bar{\boldsymbol{f}}$, though no elements of $\boldsymbol{\psi}$ appear in $\boldsymbol{\Phi}$:

$$\dot{\boldsymbol{\psi}} = \boldsymbol{\Phi}\boldsymbol{\psi} + \boldsymbol{n} \tag{5}$$

$$\boldsymbol{\psi} \equiv \begin{bmatrix}\delta\boldsymbol{x}_T\\ \delta\dot{\boldsymbol{x}}_T\\ \delta\boldsymbol{b}_{TurnOn}\\ \delta\boldsymbol{b}_{InRun}\\ \delta\boldsymbol{s}_{TurnOn}\\ \delta\boldsymbol{s}_{InRun}\end{bmatrix} \qquad \boldsymbol{n} = \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{n}_w\\ \boldsymbol{0}\\ \boldsymbol{n}_b\\ \boldsymbol{0}\\ \boldsymbol{n}_s\end{bmatrix}$$

$$\boldsymbol{\Phi} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R} & \boldsymbol{R} & \boldsymbol{R}\text{diag}\left(\bar{\boldsymbol{f}}\right) & \boldsymbol{R}\text{diag}\left(\bar{\boldsymbol{f}}\right)\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -\frac{\boldsymbol{I}}{\tau_b} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -\frac{\boldsymbol{I}}{\tau_s}\end{bmatrix}$$

To calculate the body position without image data, one would twice integrate the first form of Equation 3. Given image data, one would also integrate the state space and apply stochastic updates yet to be defined. In our discrete-time implementation, we use the following:

$$\begin{aligned}\boldsymbol{\psi}[t] &= \text{expm}\left(\tau\boldsymbol{\Phi}[t-\tau]\right)\boldsymbol{\psi}[t-\tau]\\ \boldsymbol{\Lambda}[t] &= \text{expm}\left(\tau\boldsymbol{\Phi}[t-\tau]\right)\boldsymbol{\Lambda}[t-\tau]\,\text{expm}\left(\tau\boldsymbol{\Phi}[t-\tau]\right)^{\mathrm{T}} + \boldsymbol{\Lambda}_n\end{aligned} \tag{6}$$

where τ is a small time step, typically in the range of 10 to 400 Hz. Here, expm( ) represents the matrix exponential function. The driving noise is zero mean Gaussian, uncorrelated with itself over time. Therefore, its first order expectation is zero $\text{E}[\boldsymbol{n}] = \boldsymbol{0}$, and its second order expectation is the diagonal matrix $\boldsymbol{\Lambda}_n = \text{E}\left[\boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}\right]$.

IV. Cameras and Tracking

This section describes one way of sampling the space of light rays surrounding a moving body. Any optical system that produces ray-based observations from known relative vantage points may be used. For example, the system might consist of multiple active cameras attached to the body by kinematic linkages. However, for clarity, the discussion will be limited to one calibrated camera having a single focus concentric with the IMU's inertial axes[2].

A. Camera Definition

A calibrated camera can be defined by its transformation from the space of light rays to the image space [16]. To calculate an image coordinate, a ray in the world frame is rotated into the camera frame $\vec{\boldsymbol{c}} = \boldsymbol{R}^{-1}\vec{\boldsymbol{z}}$ and projected to the image space $\boldsymbol{u}_{cam} = \boldsymbol{u}_{cam}\left(\vec{\boldsymbol{c}}\right)$, where $\boldsymbol{u}_{cam} \in [-\boldsymbol{1}\ \ \boldsymbol{1}]$ stretches-to-fill a square array. A forward-right-down frame is associated with the camera, such that $\vec{\boldsymbol{c}} = [1\ 0\ 0]^{\mathrm{T}}$ corresponds to the optical axis.

[1]Similar equations can be derived for a gimballed IMU.

[2]In an extended development, camera offset would appear in Equation 10, and body-relative camera rotation would appear in Equation 7.

Bakstein and Pajdla propose a simple calibration method for radially symmetric camera models [17]. The following model adequately represents our optics:

$$\boldsymbol{u}_{cam} = \frac{r}{\sqrt{1-c_1^2}} \begin{bmatrix} c_3 \\ c_2 \end{bmatrix} \tag{7}$$

with

$$r = \frac{\rho_1 \arccos(c_1) + \rho_2 \arccos(c_1)^2}{1 + \rho_3 \arccos(c_1) + \rho_4 \arccos(c_1)^2}$$

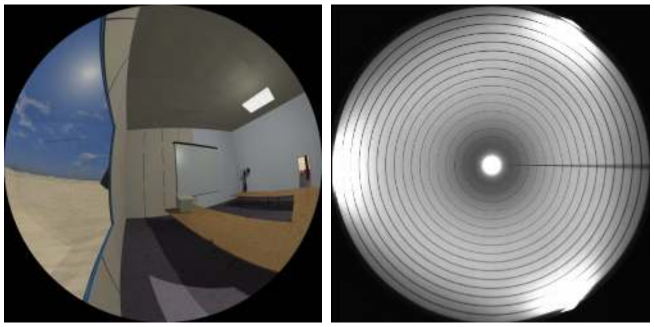
Fig. 3. Left—Synthetic image rendered with POV-Ray. Right—calibration image taken with a commercially available fisheye lens mounted to a 2/3" CCTV camera (lens sold as Kowa LMVZ164, Rainbow L163VCS, or VITEK VTL-1634).

B. Rotation Compensation

We use gyro data to compensate for camera rotation at the most basic level. In a single step, the input images are slightly blurred and warped into a rotationally-fixed projection. To warp an image, the value of each destination pixel is calculated as a weighted sum of pixel values from the source image. We selected the Gall Isographic projection as the destination, but another projection of a sphere could substitute[3]. The Gall inverse projection provides a mapping from its image space to the world ray space

$$\vec{\boldsymbol{z}} = \begin{bmatrix} \cos(\pi u_2)\cos\left(\frac{\pi}{2}u_1\right) \\ \sin(\pi u_2)\cos\left(\frac{\pi}{2}u_1\right) \\ \sin\left(\frac{\pi}{2}u_1\right) \end{bmatrix} \tag{8}$$

where $\boldsymbol{u}$ is stretched to fill an image array with $\frac{j_{max}}{i_{max}} = \sqrt{2}$. Note that a static mapping of $\vec{\boldsymbol{z}}_{ij} = \vec{\boldsymbol{z}}(\boldsymbol{u}_{ij})$ can be pre-calculated for each pixel during algorithm initialization.

C. Tracking

Our tracking system is one of many feature-based alternatives. Every rotation-compensated image passes through the Harris corner detector [18] on its way to the tracker. The resulting "corner strength" image provides a means to select features and to find those features again in future images. If less than a desired number of features have been successfully tracked from previous frames, then new ones are selected from candidate peaks above a threshold. A patch around the peak center is then permanently stored to represent each new feature. In subsequent frames, features are tracked by normalized cross-correlation [19] and located with single-pixel discrete accuracy. A patch can be lost in one of three ways: 1) No strong correlation exists within a local search region; 2) Another feature with a higher peak corner strength already occupies the space; or 3) It gets too close to the image boundary. If the feature survives, then an observation ray $\vec{\boldsymbol{z}}[t]$ is stored along with the feature's current location.

[3]Ideally, one would use a geodesic mesh.

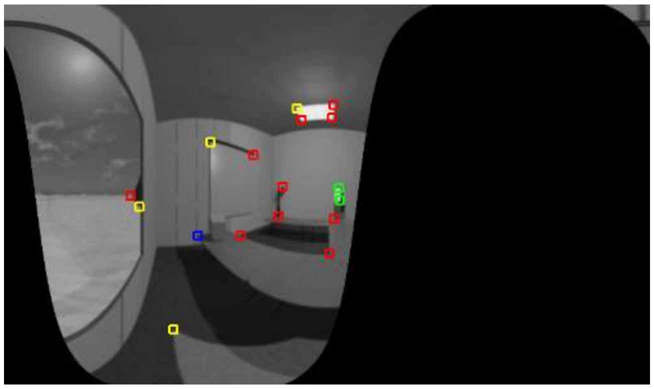
Fig. 4. Demonstration of rotation compensation, corner detection, and tracking applied to the Factory7 scene. Features are boxed and colored for diagnostic purposes.

V. Epipolar Constraint Filter

Consider a single feature tracked over multiple frames. Any two camera poses and a jointly observed feature define a plane, as shown in Figure 5. This well known relationship is often called the *epipolar constraint* [12][20][21], though it appears in various forms and under different names. Using our notation, a basic form is given by

$$(\vec{\boldsymbol{z}}[t_a] \times \vec{\boldsymbol{z}}[t_b]) \circ \Delta\boldsymbol{x}_T = 0 \tag{9}$$

where Δ represents a change between two times t_a and t_b.

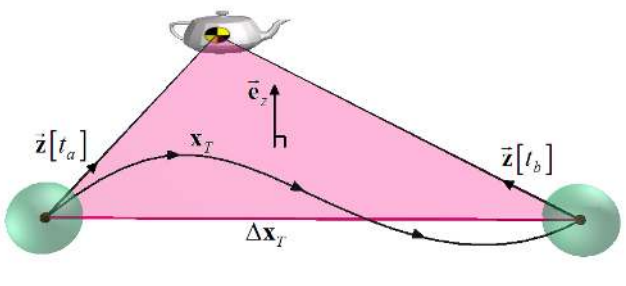

Fig. 5. The epipolar constraint, satisfied by ideal sensor data.

A. *Residual*

Suppose we want to make a filter that "knows" about the epipolar constraint. When sensor noise comes into play, the left-hand-side of Equation 9 may become nonzero. The extra value we would have to add to the right-hand-side to maintain equality is called a *residual*. We propose yet another version of the constraint that yields a more meaningful residual. To the best of our knowledge, this form has not appeared in previous literature:

$$\tilde{\boldsymbol{x}}_T = \left(\boldsymbol{I} - \vec{\boldsymbol{e}}_x \vec{\boldsymbol{e}}_x^{\mathrm{T}}\right) \vec{\boldsymbol{e}}_z \vec{\boldsymbol{e}}_z^{\mathrm{T}} \Delta \boldsymbol{x}_T \overset{?}{=} \mathbf{0} \tag{10}$$

with

$$\vec{\boldsymbol{e}}_x \equiv \frac{\Delta \boldsymbol{x}_T}{\|\Delta \boldsymbol{x}_T\|} \qquad \vec{\boldsymbol{e}}_z \equiv \frac{\vec{\boldsymbol{z}}[t_a] \times \vec{\boldsymbol{z}}[t_b]}{\|\vec{\boldsymbol{z}}[t_a] \times \vec{\boldsymbol{z}}[t_b]\|}$$

The residual $\tilde{\boldsymbol{x}}_T$ of Equation 10 has several desirable properties: 1) It vanishes when the epipolar constraint is satisfied; 2) It depends only on directly measurable quantities, so scene depth does not appear explicitly; 3) It is a vector; and 4) Its direction is always perpendicular to the observation baseline. Figure 6 provides a graphical interpretation. Although this residual was chosen carefully, arguably better forms could exist.

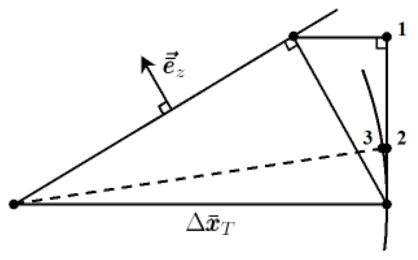

Fig. 6. State update in a case of extreme sensor conflict. 1—residual definition, 2—Kalman state update, 3—final update after spherical normalization.

B. *Stochastic Update*

Nearly all of the tools are now in place to define a stochastic visual update. When error appears in the residual, we can rationally distribute its value between our two sensors. The inertial system uncertainty is properly represented by a dynamically evolving Gaussian distribution, as in Equation 6. If we trust the residual direction and loosely assume Gaussian measurement noise associated with its magnitude, then the Bayes Least Squares (BLS) posterior estimates are given by

$$\boldsymbol{h} = \left[\begin{array}{cc} \frac{\tilde{\boldsymbol{x}}_T^{\mathrm{T}}}{\|\tilde{\boldsymbol{x}}_T\|} & \mathbf{0} \end{array}\right] \quad \text{(measurement gain)} \tag{11}$$

$$\boldsymbol{k} = \boldsymbol{\Lambda}^{-} \boldsymbol{h}^{\mathrm{T}} \left(\boldsymbol{h} \boldsymbol{\Lambda}^{-} \boldsymbol{h}^{\mathrm{T}} + \tilde{\sigma}^2\right)^{-1} \quad \text{(Kalman gain)} \tag{12}$$

$$\boldsymbol{\psi}^{+} = \boldsymbol{\psi}^{-} + \boldsymbol{k} \|\tilde{\boldsymbol{x}}_T\| \quad \text{(state update)} \tag{13}$$

$$\boldsymbol{\Lambda}^{+} = (\boldsymbol{I} - \boldsymbol{k}\boldsymbol{h}) \boldsymbol{\Lambda}^{-} \quad \text{(reduced uncertainty)} \tag{14}$$

Admittedly, we do not know much about the measurement variance $\tilde{\sigma}^2$. It depends on the imaging hardware, the tracking algorithm, the body path, and the scene. From our simulations, we were able to determine strong dependence on the body path and the feature cross product.

$$\tilde{\sigma}^2 \approx \frac{\Delta \bar{\boldsymbol{x}}_T^2 \sigma_{angular}^2}{\left\|\vec{\bar{\boldsymbol{z}}}[t_a] \times \vec{\bar{\boldsymbol{z}}}[t_b]\right\|^2} + \sigma_{tol}^2 \tag{15}$$

Here, $\sigma_{angular}$ is the expected long-term deviation of the tracking algorithm, which we assume to be about six pixels of angular separation, or 1.5 degrees. We also set the noise floor at $\sigma_{tol} = 0.01$ meters.

C. *Multiple Features*

The discussion so far has been limited to a single feature, but multiple features can be handled naturally. Each feature has its own reference time t_a, and all features share the current time t_b. We apply individual stochastic updates in sequential order, beginning with the oldest visible feature and iterating through all currently visible features. Each feature contributes one planar constraint, leading to state observability in multiple dimensions.

Persistent features are generally more valuable than newly acquired ones. For this reason, we choose t_a to be the initial observation time of a given feature. In some cases, this may not be the ideal choice. For instance, if a single GPS update became available, then it would probably be wise to re-initialize many or all features at that time.

D. *State Transfer*

During each integration step, before feature updates are applied, the values of the first six elements in the error state are *transferred*: The position error goes into the position estimate, and the velocity error is fed back into the IMU integration process[4].

$$\hat{\boldsymbol{x}}_T - \delta \boldsymbol{x}_T \Rightarrow \hat{\boldsymbol{x}}_T \ , \ \mathbf{0} \Rightarrow \delta \boldsymbol{x}_T \tag{16}$$

$$\dot{\hat{\boldsymbol{x}}}_T - \delta \dot{\boldsymbol{x}}_T \Rightarrow \dot{\hat{\boldsymbol{x}}}_T \ , \ \mathbf{0} \Rightarrow \delta \dot{\boldsymbol{x}}_T \tag{17}$$

It is well known that pure image projections contain no scale information. After each feature update is applied to the error state, we again transfer the position error. But, instead of Equation 16, we use a *normalized* update equation:

$$\hat{\boldsymbol{x}}_T[t_a] + \frac{\Delta \hat{\boldsymbol{x}}_T - \delta \boldsymbol{x}_T}{\|\Delta \hat{\boldsymbol{x}}_T - \delta \boldsymbol{x}_T\|} \|\Delta \hat{\boldsymbol{x}}_T\| \Rightarrow \hat{\boldsymbol{x}}_T[t_b] \ , \ \mathbf{0} \Rightarrow \delta \boldsymbol{x}_T \tag{18}$$

E. *Relative Covariance*

Suppose the sensor platform has been traveling for some time in a pitch-black room. Suddenly, the lights are turned on, and newly acquired feature rays are stored. The body position estimate at that time would act as a reference for

[4]Details of integration may vary by implementation.

future visual constraints, and the state covariance would also be a reference. At any time, consider what happens to the covariance, given the exact body position:

$$\boldsymbol{\Lambda} \mid \boldsymbol{x}_T = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_{4:18,4:18} - \boldsymbol{\Lambda}_{4:18,1:3}\boldsymbol{\Lambda}_{1:3,1:3}^{-1}\boldsymbol{\Lambda}_{1:3,4:18} \end{bmatrix} \quad (19)$$

Also, consider the algebraic change in covariance:

$$\Delta\boldsymbol{\Lambda} \equiv \boldsymbol{\Lambda}[t_b] - \boldsymbol{\Lambda}[t_a] \quad (20)$$

Assuming that prior knowledge of the state $\boldsymbol{x}_T[t_a]$ does not affect the change in covariance, we make the following approximation:

$$\Delta\boldsymbol{\Lambda} \approx \boldsymbol{\Lambda}[t_b] \mid \boldsymbol{x}_T[t_a] - \boldsymbol{\Lambda}[t_a] \mid \boldsymbol{x}_T[t_a] \quad (21)$$

By doing this, we lose some fidelity of the noise model. However, we avoid the need to maintain complicated relationships between multiple features.

Rearranging Equations 20 and 21 to put the unknown quantity on the left-hand-side yields

$$\begin{aligned} \boldsymbol{\Lambda}[t_b] \mid \boldsymbol{x}_T[t_a] &= \boldsymbol{\Lambda}[t_b] - \boldsymbol{\Lambda}[t_a] + \boldsymbol{\Lambda}[t_a] \mid \boldsymbol{x}_T[t_a] \\ &= \boldsymbol{\Lambda}[t_b] - \boldsymbol{\Lambda}_{ref}[t_a] \end{aligned} \quad (22)$$

with

$$\boldsymbol{\Lambda}_{ref} = \begin{bmatrix} \boldsymbol{\Lambda}_{1:3,1:3} & \boldsymbol{\Lambda}_{1:3,4:18} \\ \boldsymbol{\Lambda}_{4:18,1:3} & \boldsymbol{\Lambda}_{4:18,1:3}\boldsymbol{\Lambda}_{1:3,1:3}^{-1}\boldsymbol{\Lambda}_{1:3,4:18} \end{bmatrix} \quad (23)$$

Since the expression $\boldsymbol{\Lambda}[t_b] - \boldsymbol{\Lambda}_{ref}[t_a]$ includes an approximation, some of its eigenvalues could drop below zero. A covariance matrix by definition must be symmetric and positive semi-definite. Therefore, the approximate relative covariance for the BLS update is defined as follows

$$\boldsymbol{\Lambda}^{-} \equiv \text{EnforceSPD}\left(\boldsymbol{\Lambda}[t_b] - \boldsymbol{\Lambda}_{ref}[t_a]\right) \quad (24)$$

where EnforceSPD( ) brings all eigenvalues of its argument up to ϵ and enforces symmetry. After the BLS update, the reference uncertainty is then reinstated.

$$\boldsymbol{\Lambda}[t_b] = \boldsymbol{\Lambda}^{+} + \boldsymbol{\Lambda}_{ref}[t_a] \quad (25)$$

F. Outlier Protection

Even the best feature tracking algorithms occasionally mis-track. The types of errors observed in tracking are difficult to characterize, and may be different for each algorithm. To identify outliers, we check the validity of each visual measurement by two criteria. First, the sensor conflict must lie within a meaningful range (see Figure 6).

$$\left(\vec{\boldsymbol{z}}[t_a] - \vec{\boldsymbol{z}}[t_b]\right) \circ \vec{\boldsymbol{e}}_x > \left\|\vec{\boldsymbol{z}}[t_a] - \vec{\boldsymbol{z}}[t_b]\right\| \cos\left(\frac{\pi}{4}\right) \quad (26)$$

Second, the residual must have a reasonable probability of being observed, based on the IMU's error distribution. We treat anything beyond 2.5 standard deviations as an outlier.

$$\|\tilde{\boldsymbol{x}}_T\| < 2.5\sqrt{\boldsymbol{h}\boldsymbol{\Lambda}^{-}\boldsymbol{h}^{\mathrm{T}}} \quad (27)$$

If either of these criteria are not met, then the current measurement is excluded from the filter.

G. Numerical Caveats

There are four divide-by-zero traps to avoid in the proposed system. The first two appear when the triangle in Figure 5 becomes degenerate. If either of the denominators in Equation 10 become small, we exclude the corresponding feature from the update phase. The third trap appears when the magnitude of the residual drops below ϵ. In this case, the filter is doing well. The measurement should be included, but calculation of its direction may be numerically unstable. To work around this, we replace Equation 11 with

$$\boldsymbol{h} = \begin{bmatrix} \vec{\boldsymbol{e}}_z^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \quad (28)$$

The fourth trap appears when the inversion in Equation 23 becomes ill-conditioned. We prevent this by adding σ_{tol}^2 to the diagonal elements of $\boldsymbol{\Lambda}_{1:3,1:3}$ during inversion.

VI. Results

To demonstrate the proposed method, we present results from one simulated scene and one real scene. The first scene, entitled "Factory7," runs for 60 seconds, with an IMU sampling frequency of 50Hz, and an image frequency of 10Hz. The second scene, entitled "CSAIL5," runs for about 91 seconds, with an IMU sampling frequency of 100Hz, and an image frequency of 12.6Hz. Both sets of images are hemispherical projections and have an image-circle diameter of about 470 pixels. We would like to emphasize that the same code was applied to both data sets, where the only differences were the camera calibration parameters and an adjusted value for $\sigma_{angular}$.

A. Simulation

In our simulation, the camera moves through a fictitious industrial building as if it were carried by a person. Inertial data and camera poses are generated from a single continuous parametric path. Figure 7 shows an overhead view of the open-loop path used to generate the Factory7 scene.

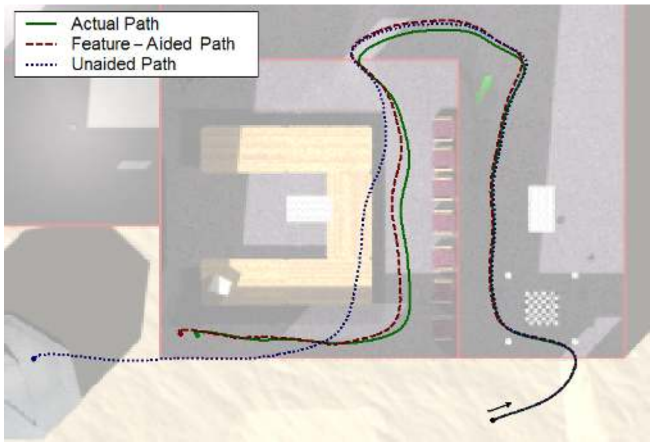

Fig. 7. Navigation results for the synthetic scene.

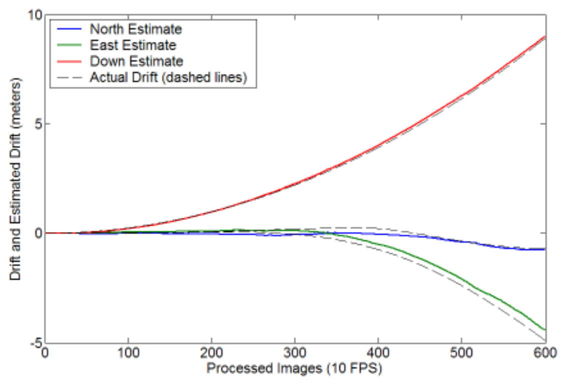

Fig. 8. IMU drift removal plot for the synthetic scene.

The filter can be judged by its ability to estimate and remove translation error. Figure 8 shows the unaided IMU drift, totaling 10.2 meters in one minute, compared to a vision-aided drift of only 1.0 meters. Since the camera motion was mostly in the horizontal plane, the vertical drift was almost entirely removed. Based on multiple simulations, we have also observed that the drift-removal-error remains within a 99.7% confidence ellipsoid, so the filter output obeys the following inequality:

$$\sqrt{\left(\hat{\boldsymbol{x}}_T - \boldsymbol{x}_T\right)^T \Lambda_{1:3,1:3}^{-1} \left(\hat{\boldsymbol{x}}_T - \boldsymbol{x}_T\right)} < 3 \tag{29}$$

B. Real Data

In the CSAIL5 scene, the camera moves through an arbitrary closed-loop path, carried by a person. Therefore, the estimated body path should return to the origin. As shown in Figure 10, the unaided IMU drifts by 262 meters during the run. With visual correction, the drift is reduced to 89 meters (66% improvement). Note how the East axis drift grows faster than the filter can compensate, probably due to excessive orientation error.

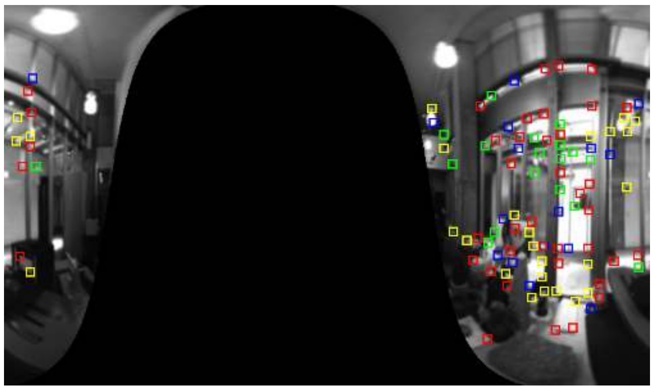

Fig. 9. Tracked features from the CSAIL5 scene.

The camera sits motionless for a short time near both the beginning and the end of the scene. At the beginning, the filter has no effect, because the feature constraints are degenerate. However, during the last 5 seconds, the position error stops growing. This effect could be called a visual zero-velocity update.

Our simulations closely resembled reality, with few exceptions. In real data: 1) Pixel saturation was more common; 2) There were about three times as many visual features per image; 3) The lens deviated from the calibrated model by as much as ± 0.5 degrees; 4) The timing of the images relative to the IMU was only known up to about ± 0.03 seconds; and 5) The initial orientation was roughly estimated.

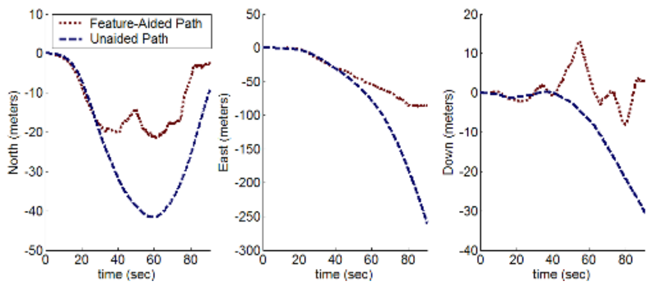

Fig. 10. Estimate of a closed-loop path, with and without visual data. No ground truth for the intermediate path was recorded, but the estimate should return to the origin.

VII. Conclusions

A method of fusing visual and inertial information has been presented. The method was based on a simple IMU error model and a feature tracking system. As expected, the camera's wide-field-of view allowed features to be tracked over extended periods of time. Each feature provided one stochastic epipolar constraint, and multiple features contributed to the overall observability of the IMU error states. Experiments were performed on synthetic and real data. The proposed filter yielded very good drift-reduction as long as the orientation error remained small.

A. Future Work

In the near-term, we would like to determine how imaging parameters, such as resolution and field-of-view, affect overall navigation performance. Errors in camera calibration and signal timing were dominant in our real data, but their effects can be greatly reduced through careful hardware design.

Theoretical extensions to the filter might include state history smoothing (in the backward Kalman filter sense), linearized orientation updates to enable longer excursions, advanced treatment of outliers, and the straightforward application of depth estimation (SFM). To facilitate comparisons with other methods, we plan to post several sets of inertial-visual data on the web at `http://www.mit.edu/~ddiel/DataSets`.

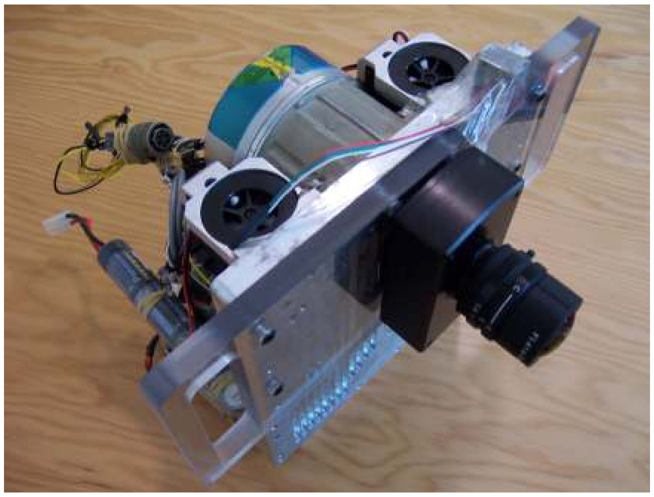

Fig. 11. The hardware used to collect real data.

B. Acknowledgements

Special thanks to James Donna and Brent Appleby for classical inertial navigation models and concepts; J.-J. Slotine and Rami Mangoubi for insights on nonlinearities and error projections; and Scott Rasmussen for hardware assistance. This work was funded by Draper Laboratory under IR&D.

REFERENCES

[1] A. D. King, "Inertial navigation—forty years of evolution," *General Electric Company Review*, vol. 13, no. 3, 1998.

[2] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 523–535, April 2002.

[3] A. Huster, "Relative position sensing by fusing monocular vision and inertial rate sensors," Ph.D. dissertation, Stanford University, July 2003.

[4] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *International Conference on Computer Vision*, Nice, France, October 2003.

[5] R. Eustice, O. Pizarro, and H. Singh, "Visually augmented navigation in an unstructured environment using a delayed state history," *International Conference on Robotics and Automation*, April 2004.

[6] J. J. Leonard, R. J. Rikoski, P. M. Newman, and M. Bosse, "Mapping partially observable features from multiple uncertain vantage points," *International Journal of Robotics Research*, vol. 21, pp. 943–975, 2002.

[7] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," *SPIE AeroSense Symposium*, April 1997.

[8] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An Atlas framework for scalable mapping," in *International Conference on Robotics and Automation*, vol. 2, September 2003, pp. 1899–1906.

[9] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," in *IEEE International Conference on Robotics and Automation*, vol. 2, September 2003, pp. 1985–1991.

[10] A. Ansar and K. Daniilidis, "Linear pose etimation from points or lines," in *European Converence on Computer Vision*, 2002, pp. 282–296.

[11] C.-P. Lu, G. D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 610–622, 2002.

[12] Y. Ma, R. Vidal, S. Hsu, and S. Sastry, "Optimal motion estimation from multiple images by normalized epipolar constraint," *Communications in Information and Systems*, vol. 1, no. 1, pp. 51–74, January 2001.

[13] T. M. Brady, C. E. Tillier, R. A. Brown, A. R. Jimenez, and A. S. Kourepenis, "The inertial stellar compass: A new direction in spacecraft attitude determination," in *16th Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, August 2002.

[14] *Department of Defense World Geodetic System*, 3rd ed., U.S. National Imagery and Mapping Agency, January 2000, TR8350.2.

[15] *WGS84 Implementation Manual*, 2nd ed., EUROCONTROL and IfEN, February 1998.

[16] J. Neumann, C. Fermüller, and Y. Aloimonos, "Eye design in the plenoptic space of light rays," in *Ninth IEEE International Conference on Computer Vision*, October 2003, pp. 1160–1167.

[17] H. Bakstein and T. Pajdla, "Panoramic mosaicing with a 180° field of view lens," in *Third Workshop on Omnidirectional Vision*, June 2002, pp. 60–67.

[18] C. Harris and M. Stephens, "A combined corner and edge detector," *Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[19] J. P. Lewis, "Fast normalized cross-correlation," *Vision Interface*, 1995.

[20] M. Antone and S. Teller, "Scalable extrinsic calibration of omni-directional image networks," *International Journal of Computer Vision*, vol. 49, no. 2–3, pp. 143–174, September–October 2002.

[21] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133–135, 1981.