The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Inertial-Aided KLT Feature Tracking for a Moving Camera

Myung Hwangbo, Jun-Sik Kim, and Takeo Kanade

Abstract— **We propose a novel inertial-aided KLT feature tracking method robust to camera ego-motions. The conventional KLT uses images only and its working condition is inherently limited to small appearance change between images. When big optical flows are induced by a camera-ego motion, an inertial sensor attached to the camera can provide a good prediction to preserve the tracking performance. We use a low-grade MEMS-based gyroscope to refine an initial condition of the nonlinear optimization in the KLT. It increases the possibility for warping parameters to be in the convergence region of the KLT.**

For longer tracking with less drift, we use the affine photometric model and it can effectively deal with camera rolling and outdoor illumination change. Extra computational cost caused by this higher-order motion model is alleviated by restraining the Hessian update and GPU acceleration. Experimental results are provided for both indoor and outdoor scenes and GPU implementation issues are discussed.

I. INTRODUCTION

Feature tracking is a front-end to many vision applications from optical flow to object tracking and 3D reconstruction. Higher-level computer vision algorithms require, therefore, robust tracking performance no matter how a camera moves. KLT (Kanade-Lucas-Tomasi) feature tracking which uses template image alignment techniques has been extensively studied from the seminal work by Lucas and Kanade [1], Shi and Tomasi [2], to the unifying work by Baker and Matthews [3].

The fundamental assumption of KLT feature tracking is small spatial and temporal change of appearance across image sequence. Hence it is naturally vulnerable to fast rotational and big shake motions of a camera. Figure 1 shows two common tracking failures of interest: The first is when the pan motion of a camera induces large optical flows, which easily occur in a hand-held device. It fails because the amount of translation is out of the search region even a multi-scale method can afford. The second is when the roll motion of a camera produces large translation as well as rotational deformation of a template. In this case the translation motion model is not enough to explain template deformation but more general warping models are required such as an affine-photometric model [4]. However, it causes extra computational cost in registering and tracking steps due to increased numbers of the parameters.

These two examples address two issues we focus on in this paper: *search region* and *tracking motion model* (image warping function) of the KLT when a camera moves fast. Mathematically speaking the KLT is a solver of nonlinear optimization problem and has a limited convergence region for a true global minimum. So more assurance for avoiding local minima is offered when a new search region is given from a external inertial sensor instead of simply using the last tracking state. A higher-order tracking model is preferred for our inertial-aided tracking method since image sequences of interest contain more appearance changes than image-only methods expect.

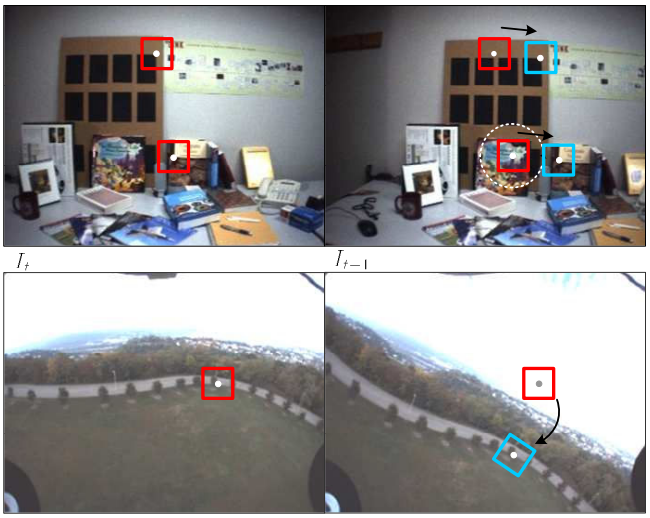

Fig. 1. (Top) An abrupt change of the pan angle in a hand-held camera generates big optical flows that even a multi-scale feature tracker cannot cover. (Bottom) The camera rolling occured in an aerial vehicle requires a rotational component in addition to translation in a tracking motion model.

An image sensor is no longer the only sensor found in up-to-date camera systems. For example, image stabilization (IS) driven by integrated inertial sensors becomes *de facto* in small gadgets to remove unwanted jitters. The higher level of integration for visual and inertial sensors has been studied for gaze stabilization [5], object detection [6], and structure from motion [7]. They are based on how humans make the vestibular system collaborate with visual movements [8]. Particularly for camera motion estimation drawn from feature tracking, there have been two main approaches on how to combine them as seen in Figure 2. The first is that camera motions considered independently from both sensors are fused in a certain way (*e.g.*, Kalman filter), and then feed-backed to feature tracking. In visual SLAM [9] and Augemented Reality [10] [11] [12] applications, feature tracking or feature matching uses the knowledge of reconstructed or given position of 3D visual landmarks. The second is that inertial measurements are deeply coupled with feature tracking algorithms in order to improve a parameter search

M. Hwangbo, J. Kim, and T. Kanade are with Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh 15213, USA {myung, kimjs, tk}@cs.cmu.edu

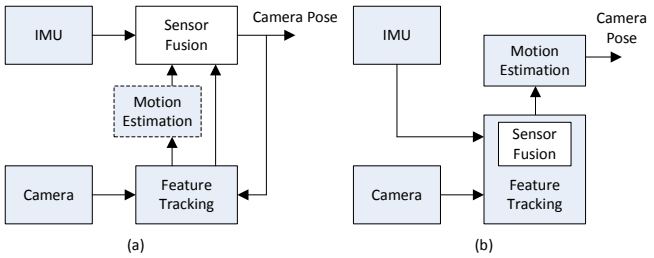

Fig. 2. Two main approaches for feature-based motion estimation that fuses inertial and visual sensors: (a) Camera motion is considered independently from both sensors and then fused in a certain way. (b) Inertial measurements are deeply coupled with a feature tracking algorithm to improve a parameter search problem.

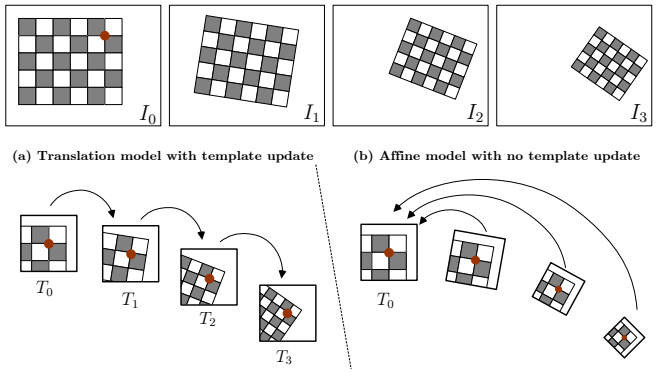

Fig. 3. For long-term feature tracking template update is necessary. A low-order warping function (*e.g.* translation only) requires frequent template updates to keep up with the appearance change. A high-order warping function (*e.g.* affine) is more flexible to template deformation but its Hessian computation is expensive.

problem [13]. Gray and Veth [14] use an inertial sensor to aid the search for matched SIFT descriptors in a predicted feature space. Makadia and Daniilidis [15] use a sensed gravity direction to reduce the number of model parameters and use the Hough transform for direct estimation of camera motion instead of feature correspondence. Our proposed method also uses instantaneous rotation obtained from inertial sensors to update the parameter of a tracking motion model. Here we only focus on *gyroscopic* sensors because large optical flows is mainly due to camera rotation in video-rate feature tracking applications.

This paper is organized as follows. Section II describes the derivation of tracker update rules for a high-order tracking model we choose. Section III shows how to embed directly inertial sensor (gyroscope) measurements into the KLT tracking algorithm. Section IV gives brief discussion on GPU implementation for realtime video-rate tracking and Section V demonstrates the improved performance of our method.

II. Feature Tracking Problem

Feature tracking is a sequence of search operation that locates a feature point along incoming images. For small appearance change between temporally adjacent images it can be formulated as a nonlinear optimization problem (1) and a set of parameters $\delta\mathbf{q}$ will be sought that minimizes the cost of intensity difference e between the template T and a new image I_{t+1}.

$$e = \sum_{\mathbf{x}\in\mathcal{A}} s(\mathbf{x})\big[\,T(\mathbf{x}) - I_{t+1}(\mathbf{w}(\mathbf{x};\mathbf{p}_t,\delta\mathbf{p}))\,\big]^2 \qquad (1)$$

where $\mathbf{x} = (x,y)^\top$ is a pixel coordinate, $\mathbf{w}(\cdot)$ is a tracking motion model (image warping function), $\mathbf{p}_t$ is a vector of warping parameters, s is a weighting function, and $\mathcal{A}$ is the area of a template window.

There are two main iterative image alignment approaches for template matching: *forward* and *inverse* methods [3]. Both are equivalent up to a first-order approximation but key differences are which image is used as a reference and how to update a warping parameter during iterations. The original Lucas-Kanade algorithm [1] is the forward method and every iteration it recomputes the Hessian from the gradient of the warped image of I_{t+1}. On the contrary the inverse method uses a fixed Hessian from the template gradient and consequently it leads to significant computational efficiency over the forward method.

A. Inverse Compositional Image Alignment

This alignment method starts with switching the roles of the image $I_{t+1}(=I)$ and the template T from (1) to (2). The Gauss-Newton method to solve for $\delta\mathbf{p}$ involves two steps: (i) linearize the cost e by the first-order Taylor expansion and (ii) find a local minimum by taking the partial derivative with respect to $\delta\mathbf{p}$. Linearizing at the current warping parameter, *i.e.*, $\delta\mathbf{p} = \mathbf{0}$, gives:

$$e = \sum_{\mathbf{x}\in\mathcal{A}} \big[\,I(\mathbf{w}(\mathbf{x};\mathbf{p}_t)) - T(\mathbf{w}(\mathbf{x};\delta\mathbf{p}))\,\big]^2 \qquad (2)$$

$$\approx \sum_{\mathbf{x}\in\mathcal{A}} \big[\,I(\mathbf{w}_\mathbf{x}(\mathbf{p}_t)) - T(\mathbf{w}_\mathbf{x}(\mathbf{0})) - \mathbf{J}(\mathbf{x})\delta\mathbf{p}\,\big]^2 \qquad (3)$$

where $\mathbf{w}(\mathbf{x};\cdot) = \mathbf{w}_\mathbf{x}(\cdot)$ for brevity and $s(\mathbf{x}) = 1$ here. $\mathbf{J} = \frac{\partial T}{\partial \mathbf{p}}\big|_{\mathbf{p}=\mathbf{0}}$ is called the steepest decent image and it remains unchanged until the template T is updated. The partial derivative of (3) with respect to $\delta\mathbf{p}$ is

$$\frac{\partial e}{\partial \delta\mathbf{p}} = 2\sum_{\mathbf{x}\in\mathcal{A}} \mathbf{J}^\top\big[\,I(\mathbf{w}_\mathbf{x}(\mathbf{p}_t)) - T(\mathbf{x}) - \mathbf{J}(\mathbf{x})\delta\mathbf{p}\,\big] = 0 \qquad (4)$$

Rearranging (4) gives a closed form solution for $\delta\mathbf{p}$ at a local minimum. It iterates until $\|\delta\mathbf{p}\| < \epsilon$ for a small ϵ due to linearization error.

$$\delta\mathbf{p} = \mathbf{H}^{-1}\sum_{\mathbf{x}\in\mathcal{A}} \mathbf{J}^\top\big[\,I(\mathbf{w}_\mathbf{x}(\mathbf{p}_t)) - T(\mathbf{x})\,\big] \qquad (5)$$

$$\mathbf{w}_\mathbf{x}(\mathbf{p}_{t+1}) \leftarrow \mathbf{w}_\mathbf{x}(\mathbf{p}_t)\circ\mathbf{w}_\mathbf{x}^{-1}(\delta\mathbf{p}) \qquad (6)$$

where $\mathbf{H} = \sum \mathbf{J}^\top\mathbf{J}$ is called the *Hessian* (precisely a first-order approximation to the Hessian).

The inverse method takes computational benefit from fixed $\mathbf{J}$ and $\mathbf{H}$ while in the forward method derived from (1) they changes every iteration. See [3] for more details.

B. Affine Photometric Warping

The choice of a tracking motion model determines the level of allowable image deformation of a template window. We employ the affine-photometric warping proposed by [4] for more robust tracking to camera rotation and outdoor illumination condition. This model has total 8 parameters, $\mathbf{p} = (a_1, \ldots, a_6, \alpha, \beta)$ in (7). The affine warp $(\mathbf{A}, \mathbf{b})$ is for spatial deformation mainly to deal with translation and rolling motion of a camera. For illumination change the scale-and-offset model (α, β) treats contrast variation of a template image. The scale α compensates for the change of an ambient light and the bias β does for the change of a directed light.

$$T(\mathbf{x}; \mathbf{p}) = (\alpha + 1)\, T(\mathbf{A}\mathbf{x} + \mathbf{b}) + \beta \tag{7}$$

$$\text{where } \mathbf{A} = \begin{bmatrix} 1 + a_1 & a_2 \\ a_3 & 1 + a_4 \end{bmatrix},\ \mathbf{b} = \begin{bmatrix} a_5 \\ a_6 \end{bmatrix}$$

By using the chain rule the steepest decent image $\mathbf{J}$ can be computed from the partial derivative at $\mathbf{p} = \mathbf{0}$. $\nabla T = (T_x, T_y)$ is the template gradient image.

$$\mathbf{J} = \frac{\partial T}{\partial \mathbf{p}}\Big|_{\mathbf{p}=\mathbf{0}} = \begin{bmatrix} \frac{\partial T}{\partial \mathbf{a}} & \frac{\partial T}{\partial \alpha} & \frac{\partial T}{\partial \beta} \end{bmatrix} = \begin{bmatrix} \frac{\partial T}{\partial \mathbf{a}} & T & 1 \end{bmatrix} \tag{8}$$

$$\frac{\partial T}{\partial \mathbf{a}}\Big|_{\mathbf{a}=\mathbf{0}} = \frac{\partial T}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \mathbf{a}}\Big|_{\mathbf{a}=\mathbf{0}} = \frac{\partial T}{\partial \mathbf{x}} \frac{\partial \mathbf{w}}{\partial \mathbf{a}} = \nabla T \frac{\partial \mathbf{w}}{\partial \mathbf{a}} \tag{9}$$

$$= \nabla T \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix} \tag{10}$$

Then

$$\mathbf{J} = \begin{bmatrix} xT_x & yT_x & xT_y & yT_y & T_x & T_y & T & 1 \end{bmatrix} \tag{11}$$

In (5) the update, $\delta\mathbf{p}$, requires the inversion of the symmetric 8×8 Hessian matrix for this model, $\mathbf{H} = \sum_{\mathcal{A}} \mathbf{J}^\top \mathbf{J}$. So much more expensive computation, $\mathcal{O}(n^3)$ with Gaussian elimination method, is involved when compared with a 2×2 Hessian in the translation model.

$$(\mathbf{A}, \mathbf{b})_{t+1} \leftarrow (\mathbf{A}_t \delta\mathbf{A}^{-1},\ \mathbf{b}_t - \mathbf{A}_t \delta\mathbf{b}) \tag{12}$$

$$\alpha_{t+1} \leftarrow (\alpha_t + 1)/(\delta\alpha + 1) \tag{13}$$

$$\beta_{t+1} \leftarrow \beta_t - (\alpha_t + 1)\delta\beta \tag{14}$$

Finally from the inverse compositional update rule in (6) the warping parameters $\mathbf{p}_{t+1}$ of the affine-photometric model are propogated as above.

C. Template Update

For long-term feature tracking template update is a mandatory step to keep a plausible track. It usually occurs when the current view point of a camera is quite different from that the template has been taken from and thus it is hard to fit correctly with a chosen tracking motion model. An inherent problem with the template update is accumulation of the localization error of a feature. There exists computational trade-off between frequency of the template update and the order of a motion model. Figure 3 shows two extreme cases of which difference is whether the template is renewed every certain frames or remains unchanged across images. Less template update requires a tracking motion model to have higher degrees of freedom to adjust bigger image deformation in an incoming image. Nonetheless the Hessian is updated less frequently of which computation is expensive for a higher order model.

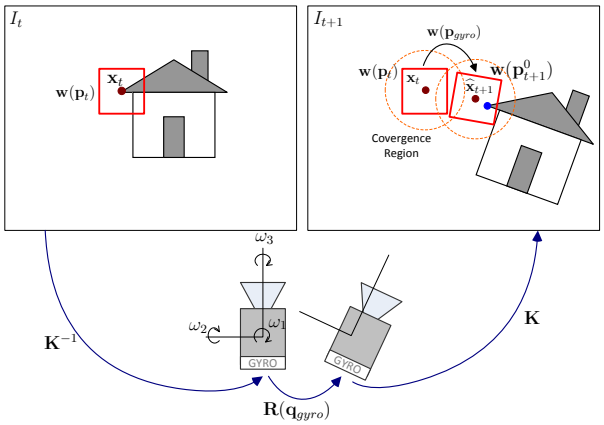

Fig. 4. The inertial-aid KLT for a new image I_{t+1} starts at $\mathbf{p}^0_{t+1}$ instead of $\mathbf{p}_t$ which is not in the covergence region $\mathcal{C}$. The initial condition is refined by the 2D homography $\mathcal{H}$ computed from the instantaneous camera rotation $\mathbf{q}_t^{t+1}(gyro)$ from the IMU and a camera calibration matrix $\mathbf{K}$.

We choose carefully the times of the template update by monitoring how good a current tracking is. The measures for tracking quality are squared error, normalized correlation, and shearness. Since tracking a feature location is the main goal of this paper rather than the template itself, the window around the last feature position is captured as a new template if required.

III. Inertial Fusion for Robust Tracking

One fundamental assumption for all the KLT-based tracking algorithms is a small inter-frame change (*e.g.* less than three-pixel translation with a 9×9 template window). Otherwise the nonlinear optimization (1) may fail to converge because the convergence region $\mathcal{C}$ for $\delta\mathbf{p}$ is limited to a concave region in which the first-order approximation (4) is close enough for a correct update direction. In the inverse image alignment method, therefore, the success of tracking given T and I_{t+1} heavily depends on whether a starting point $\mathbf{p}^0_{t+1}$ of the optimization is in $\mathcal{C}$ or not. The image-only tracking method simply uses $\mathbf{p}^0_{t+1} = \mathbf{p}_t$.

Our goal is to increase a chance of the convergence by relocating $\mathbf{p}^0_{t+1}$ to $\mathcal{C}$ as close as possible with the aid from the inertial sensor. Here we only focus on *gyroscopic* sensors because large optical flows is mainly due to camera rotation in video-rate feature tracking applications. The knowledge of the instantaneous camera rotation $\mathbf{q}_t^{t+1}(gyro)$ from a tri-axial gyroscope is directly coupled with the current warping parameter $\mathbf{p}_t$ to predict $\mathbf{p}^0_{t+1}$.

A. Camera Ego-Motion Compensation

Figure 4 illustrates the main idea of the paper about how the gyroscope helps feature tracking more robust to big optical flows. If the two images I_t and I_{t+1} are assumed to be taken by a pure camera rotation $\mathbf{R}_t^{t+1}$ the motions of all the feature points can be described by a single 2D homography $\mathcal{H}$ once a camera calibration matrix $\mathbf{K}$ is known [16].

$$\mathcal{H} = \mathbf{K}^{-1}\mathbf{R}_t^{t+1}\,\mathbf{K} \tag{15}$$

The homography $\mathcal{H}$ is a higher-order deformation than the affine warp $(\mathbf{A}, \mathbf{b})$ we use for the spatial tracking motion model. Hence the prediction affine warp is appropriately extracted componentwise from the homography. At first the homography needs to be normalized by $\mathcal{H}_{33}$. The linear transformation part $\mathbf{A}_{pred}$ is the same as an upper 2×2 matrix of $\mathcal{H}$ in order to copy affine components (rotation, scaling, and shear) but projectivity. The translation part $\mathbf{b}_{pred}$ is the transfer amount of position by the 2D homography. $\mathbf{x}_{\mathcal{H}} \equiv \mathcal{H}\,[\,\mathbf{x}\;1\,]^\top$ is a transferred position of $\mathbf{x}$ by $\mathcal{H}$. The photometric parameters remains unaffected.

$$\mathbf{A}_{pred} = \mathcal{H}_{2\times 2} \tag{16}$$
$$\mathbf{b}_{pred} = \mathbf{x}_{\mathcal{H}} - \mathbf{x} \tag{17}$$
$$\alpha_{pred} = \beta_{pred} = 0 \tag{18}$$

Given the prediction warp by the inertial sensor the initial parameter $\mathbf{p}_{t+1}^0$ is obtained by the forward composition of the warping function with $\mathbf{p}_t$.

$$\begin{aligned}\mathbf{w}(\mathbf{x};\mathbf{p}_{t+1}^0) &= \mathbf{w}(\mathbf{x};\mathbf{p}_{pred}) \circ \mathbf{w}(\mathbf{x};\mathbf{p}_t) \\ &= \mathbf{w}(\mathbf{w}(\mathbf{x};\mathbf{p}_{pred}),\,\mathbf{p}_t)\end{aligned} \tag{19}$$

The chance of $\mathbf{p}_{t+1}^0 \in \mathcal{C}$ would be greater than that of $\mathbf{p}_t \in \mathcal{C}$ as the bigger camera rotation is involved.

B. Camera-IMU Synchronization

High-precision synchronization of raw sensors usually requires hardware-level triggering based on a precise clock. For a COTS-based sensor solution with a generic computer, however, exact sampling time of sensor data is obscure. It is hindered by unknown delays in data bridges such as communication and buffering between low-level embedded systems.

If the delays between the sensors and a computer that fuses both data are unknown but fixed, Figure 6 shows a easy and simple way to identify the time lag between two measurement sequences. We first repeat a small sinusoidal motion to the camera in one direction and then compute the average of the optical flow magnitude between images from many feature tracks. Note that the optical flow and gyroscope data have an identical phase since they are induced by the same camera motion. The phase lag ϕ_o between the two signals reveals the time offset t_o of both sensors. The FFT can be used to derive a correct phase lag from repeated camera motions.

Once t_o is obtained, the camera rotation $\mathbf{q}_t^{t+1}$ can be computed properly from the interpolation of asynchronous gyroscope measurements. Let $\boldsymbol{\omega} = [\,\omega_1\;\omega_2\;\omega_3\,]^\top$ be angular

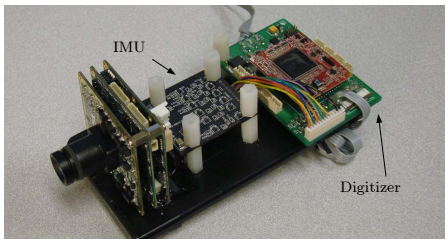

Fig. 5. The camera-IMU system for the inertial-aided feature tracking

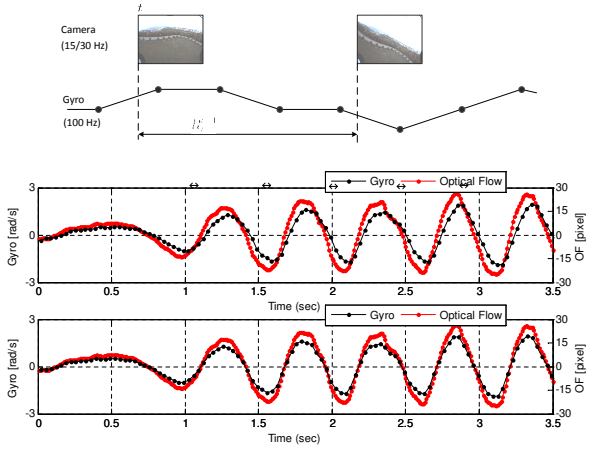

Fig. 6. Synchronization of the camera and the IMU: (Top) Asynchronous gyroscope data with image timestamps are linearly interpolated to get the rotation between I_t and I_{t+1}. (Bottom) By applying a sinusoidal camera motion, the time offset t_o in synchronization can be identified from the phase difference ϕ_o between the IMU and optical flow signals.

rates from gyroscopes. For a strap-down IMU configuration the use of quaternion in (20) is fast and effective for integrating $\boldsymbol{\omega}$ to rotation angle $\mathbf{q}$.

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} = \frac{1}{2}\begin{bmatrix} \mathbf{0} & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix}\mathbf{q} \tag{20}$$

From the initial $\mathbf{q}_t = [\,1\;0\;0\;0\,]^\top$ $\mathbf{q}_{t+1}$ is computed by a numerical integration with proper quaternion normalization.

C. IMU Noise and Camera-IMU Calibration Error

The prediction error in $\mathbf{p}_{pred}$ is caused by two main factors: modeling error and calibration error. Modeling error means that neither camera translation nor projectivity is considered in the prediction warp. We intentionally ignore camera translation since noisy MEMS-based accelerometers is insufficient to provide robust traveling distance and direction of the camera. The other includes misalignment of camera/IMU, camera calibration error, bias and variance of IMU noise. Particularly we focus how much IMU noise is allowable in a given convergence region $\mathcal{C}$ of feature tracking.

For the simplicity of analysis, the error of translation distance $\|\mathbf{b}_{pred}\|$ due to IMU noise is investigated in a pinhole camera model. For a small rotation, the variances for

Algorithm 1: Inertial-Aid KLT Feature Tracking

$n_{iter}, n_{fmin}, p_{max} \leftarrow$ fixed numbers
Compute the gradient $\nabla \mathbf{I}_{t+1}$
if $n_{feature} < n_{f,min}$ **then**
  Find new features from cornerness of $\nabla \mathbf{I}_{t+1}$
  Compute $\mathbf{H}$ and $\mathbf{H}^{-1}$
  Fill in lost slots of feature table
Get camera rotation $\mathbf{R}(\mathbf{q})$ from IMU
for *pyramid level* = p_{max} **to** 0 **do**
  forall *features* **do**
    Update initial warp $\mathbf{w}_{t+1}$ from $\mathbf{w}_{imu}$ using (19)
    Warp image $I_{t+1}(\mathbf{w}(\mathbf{x}, \mathbf{p}_{t+1}))$
    Compute error $e = T - I_{t+1}(\mathbf{w}(\mathbf{x}, \mathbf{p}_{t+1}))$
    Compute update direction $\delta\mathbf{p}$ using (5)
    for $k = 1$ **to** n_{iter} **do**
      Line search for best scale s^*
    Update parameter with $s^*\delta\mathbf{p}$ using (12)-(14)
  Remove features that $e > e_{thresh}$

camera panning and rolling can be approximated from (15) and (17) respectively for a small rotation.

$$\sigma^2_{\|\mathbf{b}\|,pred} = (kf)^2 \, \Delta t_s \, \sigma^2_{\omega,x} \tag{21}$$

$$\sigma^2_{\|\mathbf{b}\|,pred} = r^2 \, \Delta t_s \, \sigma^2_{\omega,z} \tag{22}$$

where k is a pixel scale factor, f is a focal length, r is the distance in pixels from a camera center, and Δt_s is a image sampling time interval. Let the camera be a 1/4" CCD sensor and 640 $\times$ 480 resolution ($k = 126.0$) at 30Hz with a lens of 60° field-of-view (f = 2mm). Given a typical variance of MEMS-based gyroscopes $\sigma_\omega = 0.01$ rad/s, the variances are $\sigma_{\|\mathbf{b}\|,pred} = 0.4$ pixel for pan or tilt motion and $\sigma_{\|\mathbf{b}\|,pred} = 0.8$ pixel for roll motion at 120 pixels from the center (r = 120). For the 15 $\times$ 15 template size, the errors are negligible when compared with the convergence region shown in Figure 11.

IV. GPU Implementation

In practice the affine-photometric tracking is barely used due to the computational complexity. In template registration time complexity of the Hessian inverse is generally $O(n^3)$ where n is the number of parameters. Compared with the translation model (n = 2), therefore, computation load of the affine-photometric model (n = 8) increases at least 64 times. In template tracking the complexity is also $O(n^2)$ in updating warping parameters. When several hundreds of features are tracked, the increased burden can be alleviated by parallelized hardware computation. We implemented our high order model on a GPU as Sinha *et al.* [17] has done with the translation warping model which is valid for the ground vehicle application.

However, note that the GPU tend to lose its computational efficiency and is even worse than the CPU when a target algorithm has complicated decision flows depending on the current evaluation of input data. In this case the GPU has difficulty to use concurrent threads that have the same length of instructions with different inputs. It violates the SIMD principle of parallel computing. Algorithm 1 contains two routines of which internal iterations depend on inputs: sorting cornerness measures and inversion of the Hessian. Gaussian elimination involved with the 8×8 Hessian inverse is a quite complicated thread in the GPU side. Even with extra burden of CPU-GPU data transfer, they run faster on the CPU. Based on NVIDIA CUDA library [18], therfore, we adopt a CPU-GPU hybrid approach in which particularly the cornerness sort and the Hessian inversion are computed in the CPU.

V. Experiments

The experiments are performed on two kinds of scenes: the desk scene (640 $\times$ 480 at 30Hz) from a hand-held device and the outdoor aerial scene (320 $\times$ 240 at 15Hz) from a MAV. Both uses the same camera system in Figure 5. We use the multi-resolution pyramid approach to increase the search region as explained in the GPU implementation.

A. Camera-IMU system

The IMU is placed right behind of the camera and the relation between both sensors are calibrated. The camera is the Sentech USB 2.0 color board camera and the IMU is the O-Navi Gyroscube tri-axis MEMS analog module. Three orthogonal angular rates in the range of $\pm 150°$ are sampled at 100 Hz with a 11-bit resolution digitizer.

B. Results

We have run feature tracking algorithms in two ways for the given scenes. One is purely vision-based tracking that does not use the IMU information but the images only. We call it the *image-only* method here. The other is our inertial-aided method that the camera-ego rotation is compensated. The advantages of the inertial-aided feature tracker over the image-only method are summarized as follows.

- Higher tracking success rate
- Longer tracking length with less drift

Figure 7 compares the inertial-aided and the image-only feature tracking on the desk scene. We rotates the camera so that it undergoes three different rotations - panning, tilting, and rolling sequentially. We apply this set of the camera rotations both slowly and fast in order to see when the IMU fusion makes the difference between two methods. The IMU measurements are plotted together with the number of tracked features for this purpose. In Figure 7(d) at the slow camera motion no big difference in the number of tracked features is noticeable between both methods. When the magnitude of the camera motion increases twice in Figure 7(e), however, the image-only method starts to drop features significantly whenever there is a peak in angular rates and it has good tracks only when the camera nearly stops to switch the direction. On the contrary the inertial-aided method maintains the good performance only losing 10% $\sim$ 20% of features. The robust performance of the inertial-aided method are shown in Figure 7(a)-(c) when the

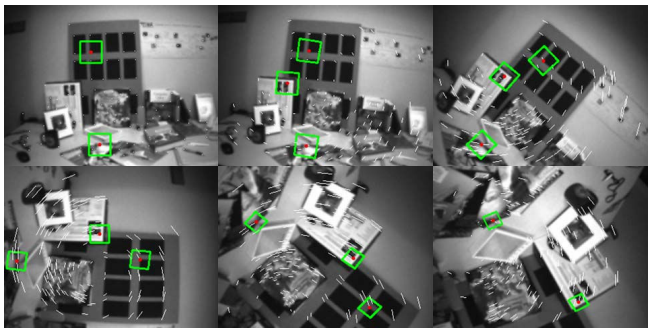

Fig. 9. The templates are well tracked by the affine motion model when the camera moves forward with a rolling motion in the desk scene. All the degrees of freedom of the affine warping are revealed in template deformations colored in green.

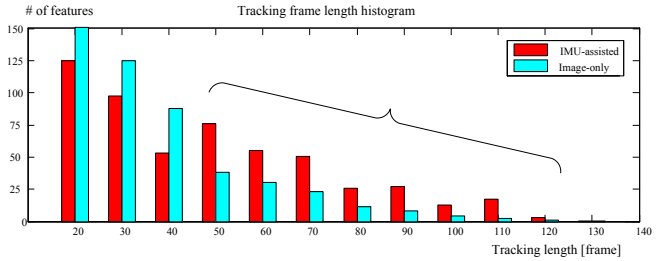

Fig. 10. The tracking length histogram of 500 features in the desk scene with no new registration: The inertial-aided KLT has more features in regions of the longer tracking length as marked by the curly bracket.

fast camera motions occur. The green dots indicate predicted feature locations by the IMU. See how close the green dots are to the red dots - the tracked points - to verify the effect of the better initialization for feature tracking.

The spatial warping of a square template by the affine transformation is shown in Figure 9. All the degrees of freedom of this motion model can be clearly seen when the camera moves forward with a rolling motion. Here the templates translate and rotate due to the camera rotation. They scale down as the camera approaches to the desk and get sheared as the angle to the desk plane changes. It demonstrates the affine motion model is good enough to track templates deformed by camera rotation. Figure 10 compares the tracking length histograms of both image-only and inertial-aided methods for the desk scene. It evaluates the capability for a long-frame tracking. In this experiment, no new features are registered once 500 features are selected at the first image frame. At the high frame length bins marked by the curly bracket, the inertial-aided tracker has more features. In other words, the image-only tracker tends to lose tracking quickly at the lower frame length.

Figure 8 shows outdoor experiments taken by a micro aerial vehicle. The scene has been taken when the aerial vehicle is banking to the left and back to a level flight. Large optical flows occur when the vehicle changes its rolling angle. The inertial-aided method provides quite better optical flows sufficient to perceive the current vehicle motion relative to the ground plane. This scene has the regions difficult to track. A group of trees has repeated textures and the grass field is very homogeneous. In these regions a tracker is susceptible to get in a wrong local minimum. Since the inertial-aided tracker narrows down a search area, however, stable tracking results are available.

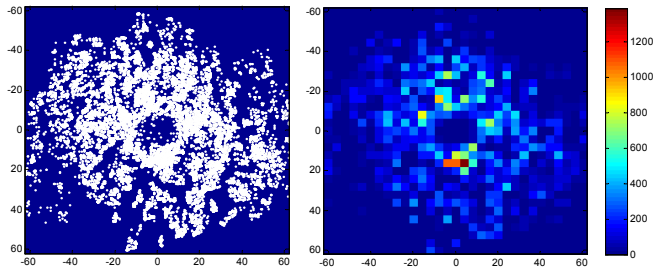

Fig. 11. Feature translations rescued by the inertial-aid tracking in the desk scene: Tested by 15×15 template size. No points appears for small translations because the image-only KLT can also track. The inertial-aid KLT can cover up to 60 pixel translation.

C. GPU Performance

The upper row of Figure 12 shows the comparison when the numbers of features changes with 25×25 template size and all four levels of pyramids. One can see that the tracking time has no increase on most GPUs when the number of features increases. It is because the processing time on a GPU is determined by the worst case. Note that even the

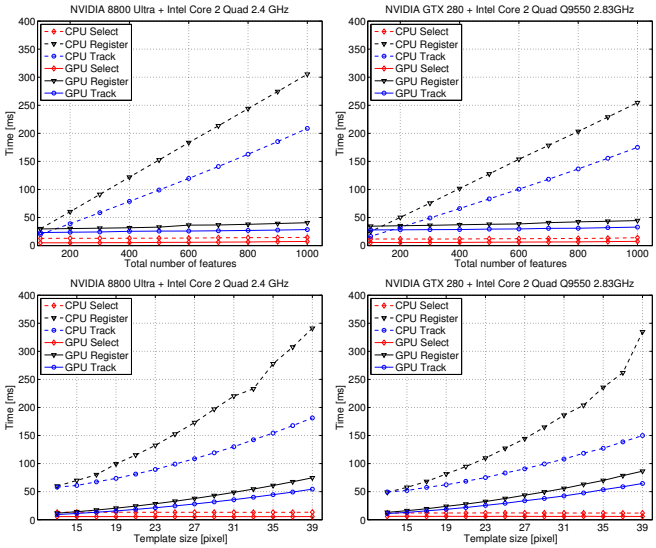

Fig. 12. Performance comparison between the CPU and GPU implementations on various platforms with respect to the number of features using 25×25 templates (upper row), and the template size with 500 features (lower row). Dashed lines are for the CPU and solid lines for the GPU. The GPU has no performance degrade when the number of features increases.

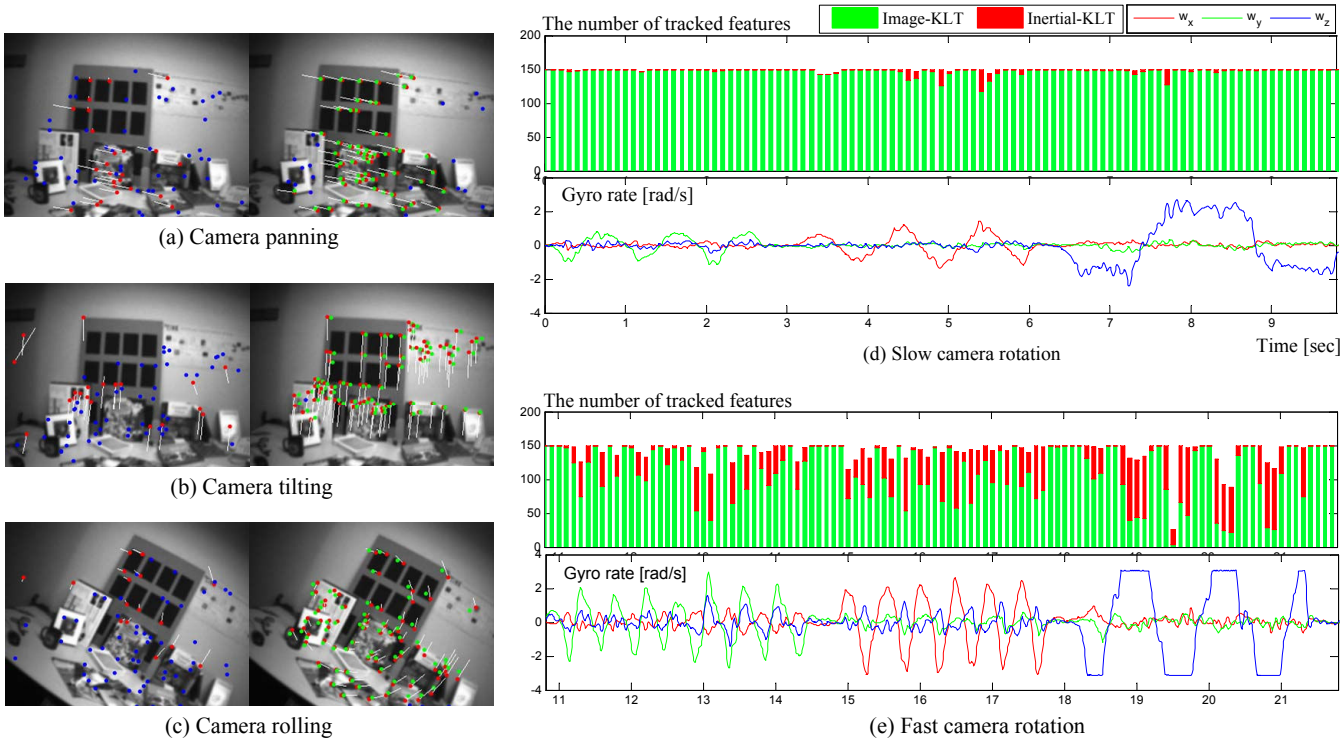

Fig. 7. Inertial-aided *vs.* Image-only KLT on the desk scene: The camera is rotated in (a) pan, (b) tilt, and (c) roll motions sequentially. Two different motion sets, (d) and (e), are applied in terms of the frequency and the magnitude of the angular rates. When the magnitude of motions increases twice in (e), significant amount of features fails in the image-only KLT but the inertial-aided KLT maintains the good performance in terms of the number of tracked features. (Right graphs) The red area in the bar indicates the number of features rescued by the inertial-aid KLT. (Left images) The green dots are predicted locations by the IMU. See how close the green dots are to the red ones from the purpose of the better initialization for feature tracking.

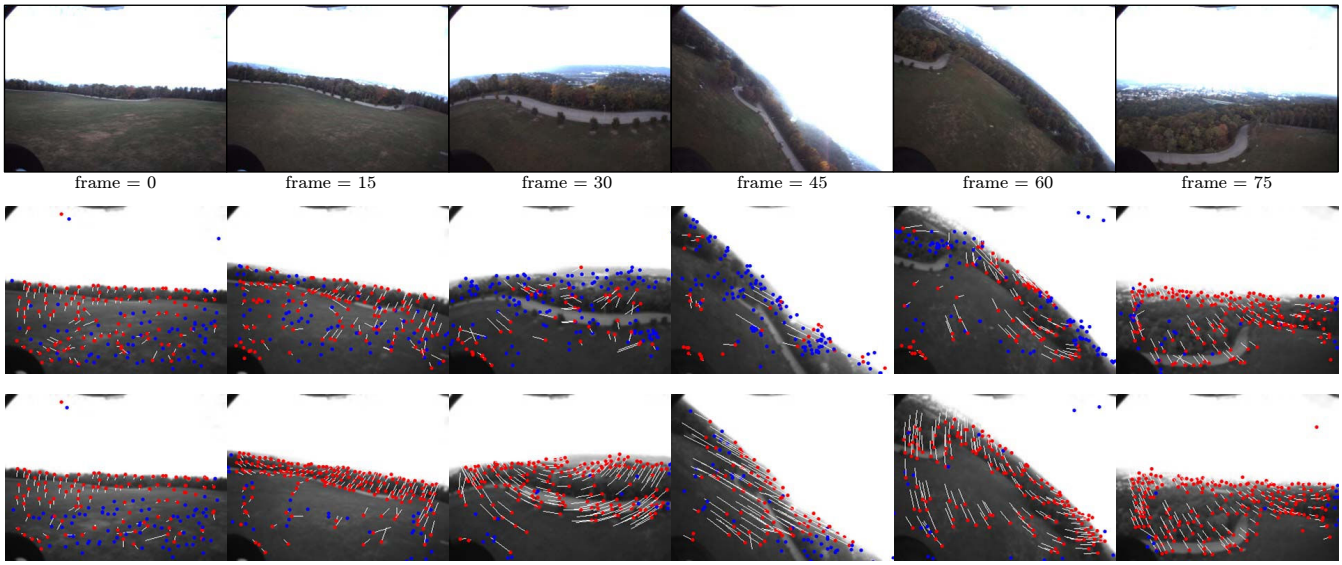

Fig. 8. Inertial-aided *vs.* Image-only feature tracking on the aerial scene: (Top) Input image sequence (Middle) The image-only feature tracking results. (Bottom) The inertial-aided feature tracking results. The aerial vehicle is banking to the left and back to a level flight. The inertial-aided method generates quite better optical flows sufficient to perceive the current vehicle motion relative to the ground plane. Red dots are for good tracks and blue dots are for lost ones.

worst GPU runs faster than the best CPU. In the lower row of Figure 12, the processing time of tracking 500 features increases quadratically with respect to the template size but its rate is much smaller than that of the CPU.

VI. Conclusion

We propose an enhanced KLT feature tracking method assisted by a low-cost IMU for a freely moving camera in 3D. We use the rotational information from the IMU to give better initial estimates of motion parameters for template warping. By compensating a camera-ego motion, search ranges for the motion parameters become much narrower. This simple assistance from the IMU brings significant improvement: longer feature tracking under abrupt camera motions, overall speed-up due to less frequent feature selection. With the affine-photometric motion model our inertial-aided tracker shows very stable and robust results under big shake and fast rolling motions of the camera. We also show that this high-order motion model is implemented for a video-rate feature tracking up to 1000 features by the SIMD property of a GPU.

References

[1] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, (Vancouver, Canada), pp. 674–679, 1981.

[2] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, (Seattle), June 1994.

[3] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *Int. J. Comput. Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[4] H. Jin, P. Favaro, and S. Soatto., "Real-time feature tracking and outlier rejection with changes in illumination," in *Proc. of the Int'l. Conf. on Computer Vision (ICCV)*, July 2001.

[5] F. Panerai and G. Sandini, "Visual and inertial integration for gaze stabilization.," in *Proc. Int'l Symp. Intelligent Robotic Systems (IROS)*, 1997.

[6] J. Lobo and J. Dias, "Vision and inertial sensor cooperation using gravity as a vertical reference," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1597–1608, 2003.

[7] T. Okatani and K. Deguchi, "Robust estimation of camera translation between two images using a camera with a 3d orientation sensor," in *Proc. IEEE Int'l Conf. on Pattern Recognition (ICPR)*, 2002.

[8] R. H. Carpenter, *Movements of the Eyes*. London Pion Limited, 2 ed., 1988.

[9] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[10] J. Chandaria, G. Thomas, B. Bartczak, K. Koeser, R. Koch, M. Becker, G. Bleser, D. Stricker, C. Wohlleber, M. Felsberg, F. Gustafsson, J. Hol, T. B. Schn, J. Skoglund, P. J. Slycke, and S. Smeitz, "Real-time camera tracking in the MATRIS project," in *IBC2006*, (Amsterdam), 2006.

[11] G. Bleser, C. Wohlleber, M. Becker, and D. Stricker, "Fast and stable tracking for ar fusing video and inertial sensor data," in *Int'l Conf in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, (Plzen, Czech Republic), pp. 109–115, 2006.

[12] Y. Yokokohji, Y. Sugawara, and T. Yoshikawa, "Accurate image overlay on video see-through HMDs using vision andaccelerometers," in *Proc. IEEE Virtual Reality*, 2000.

[13] S. You, U. Neumann, and R. Azuma, "Hybrid inertial and vision tracking for augmented reality registration," in *Proc. IEEE Virtual Reality*, 1999.

[14] J. Gray and M. Veth, "Deeply-integrated feature tracking for embedded navigation," in *ION International Technical Meeting Program*, (Anaheim, USA), 2009.

[15] A. Makadia and K. Daniilidis, "Correspondenceless ego-motion estimation using an IMU," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 3534–3539, 2005.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge, 2 ed., 2003.

[17] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Gpu-based video feature tracking and matching," Tech. Rep. Technical Report 06-012, Department of Computer Science, UNC Chapel Hill, May 2006.

[18] *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*, 2008.