A Robust Pixel-Aware Gyro-Aided KLT Feature Tracker for Large Camera Motions

Weibo Huang and Hong Liu*

***Abstract*—Tracking fails in the optimization step of conventional KLT (Kanade-Lucas-Tomasi) feature tracker mainly due to the inadequate initial condition that falls out of the convergence region, especially when a camera rotates rapidly or shakes severely. To overcome the problem, we propose a pixel-aware gyro-aided KLT feature tracker that remains accurate and robust under fast camera-ego motion conditions. In particular, we develop a pixel-aware gyro-aided feature prediction algorithm to predict the initial optical flow and obtain the patch deformation matrix of each feature point. It increases the probability of initial estimates to locate in its convergence region. Unlike the existing methods, which assume all the tracked feature pairs were constrained by the same homography prediction matrix, our prediction matrix is adjustable for each feature as it considers the pixel coordinates in the prediction process. A geometric validation based on homography and fundamental check is also adopted to remove outlier tracks. Experimental results on both public datasets and real-world sequences demonstrate that the feature tracking accuracy and robustness can be significantly improved by the proposed method. To facilitate further development, the code is publicly available at https://github.com/weibohuang0314/pixel_aware_gyro_aided_klt_feature_tracker.**

I. Introduction

Feature tracking is a process of determining and maintaining the location of visually interesting points as they move about in moving video. The process is essential for many vision applications [1], for example, microvision-based motion measurement [2], gesture detection and recognition [3], 3D coordinate measurement [4], and autonomous mobile robot navigation [5]. One common solution is similar to the feature matching procedure that detects and describes interesting points on each image and then matches them by finding similar descriptors. This solution has been widely adopted as the detector and descriptor are robust to illumination change and large camera motions. However, the feature detection and descriptor matching are time-consuming, leading to the unsuitable for the applications like visual odometry on lightweight computational platforms. To this end, many researchers begin to study more efficient methods, for example, using the appearance PatchMatch-based technique [6] to reduce feature detection or avoid descriptor generation and matching process.

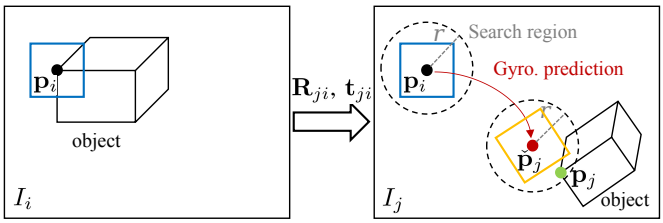
This work is supported by National Natural Science Foundation of China (NSFC, No.62073004), Shenzhen Fundamental Research Program (No.GXWD20201231165807007-20200807164903001, No.JCYJ20190808182209321).

W. Huang is with Key Laboratory of Machine Perception, Shenzhen Graduate School, Peking University, Beijing, China (e-mail: weibohuang@pku.edu.cn).

H. Liu is with Key Laboratory of Machine Perception, Shenzhen Graduate School, Peking University, Beijing, China, and also with School of Artificial Intelligence, Chongqing University of Technology, Chongqing, China (corresponding author, e-mail: hongliu@pku.edu.cn).

In the PatchMatch-based feature tracking domain, the KLT feature tracker [7]–[10] is one of the most popular methods. It considers local information derived from a small patch surrounding each interesting point and searches for the best similar patch within a search region. The success of KLT feature trackers depends on two assumptions: data conservation and spatial coherence. Data conservation is derived from the observation that the observed objects generally persist in time. Thus, the intensity of a small region in two consecutive images remains similar, although its position is changing. The spatial coherence expects the motion in a small region to be constant. These assumptions are simplifications and hence may be violated in practices. For example, motion boundaries or fast camera motions violate the common assumption that the optical flow varies smoothly. Pyramidal implementation of the classical Lucas-Kanade [11] or considering the affine deformation between images [12] can handle this challenge to some extent. However, when the camera undergoes large rotations or displacement, it still requires to increase the number of pyramidal layers or enlarge the search region. As a result, it is difficult for the conventional methods to set a uniform pyramidal layer or search region to suit all scenarios.

To overcome the problem, we propose a pixel-aware gyro-aided KLT feature tracker for improving the feature tracking performance for large camera motions, without the need to adjust the pyramidal layers or the search regions. The motivation is that the optical flow of feature points mainly comes from the rotation movement of the camera rather than the translation movement, while the camera rotation can be estimated by integrating gyroscope measurements. The idea of incorporating gyroscope measurements to aid feature tracking has been investigated in the literature (see Section II). These methods first integrated gyroscope measurements to obtain the related rotation between two images. Then, a homography prediction matrix was obtained by multiplying the integrated result and the camera intrinsic matrix. After that, the initial estimates of matched features were predicted by performing a feature homography transformation and then be refined by finding the best-matched patches nearby the predicted estimates. The existing methods have achieved impressive performance. However, most of them assumed that all the feature pairs in two images were constrained by the same homography prediction matrix. In contrast, we find that the feature prediction accuracy and tracking performance can be further improved by considering the pixel coordinates of each feature in the prediction process. In our method, both the feature pixel coordinates and the gyroscope measurements are involved in the prediction process, thus the prediction



First frame I_i Second frame I_j

Fig. 1: A toy example that a camera undergoes large motion, with rotation $\mathbf{R}_{ji}$ and translation $\mathbf{t}_{ji}$. The source point $\mathbf{p}_i$ and the target point $\mathbf{p}_j$ are the same corner of the object observed in I_i and I_j respectively. r is the radius of nearest neighbor search region. It is obvious that $\mathbf{p}_j$ is located out of the search region centered on $\mathbf{p}_i$. $\check{\mathbf{p}}_j$ is the position predicted by the proposed pixel-aware gyro-aided feature prediction algorithm. $\mathbf{p}_j$ is located within the search region centered on $\check{\mathbf{p}}_j$. The blue square patch around $\mathbf{p}_i$ is deformed due to camera rotation. The patch after deformed is shaped in orange square. **Best viewed in color.**

matrix is adjustable and specific for each feature. Note that the mistracked problem is inevitable in feature tracking, we also introduce a geometric validation mechanism using score-ratio-based homography/fundamental model selection to tackle this problem.

A toy example explaining the idea of our pixel-aware gyro-aided feature tracker is shown in Fig. 1. The first/reference frame I_i and the second/current frame I_j denotes two 2D images captured by a common camera. The camera undergoes a rotation $\mathbf{R}_{ji}$ and a translation $\mathbf{t}_{ji}$ from time i to time j. Points $\mathbf{p}_i$ and $\mathbf{p}_j$ are the same corners of an object observed in I_i and I_j respectively. The pair $(\mathbf{p}_i, \mathbf{p}_j)$ is denoted as a tracked feature pair. In the following, we denote $\mathbf{p}_i$ as a source interesting feature given by users, and $\mathbf{p}_j$ as an unknown target/matched feature to be tracked. The dashed circle with radius r denotes a search region. Note that $\mathbf{p}_j$ is out of the search region around $\mathbf{p}_i$, if we set $\mathbf{p}_i$ as the initial estimate of $\mathbf{p}_j$ and try to refine it by finding the best-matched patch through minimizing a tracking energy function, the optimization would be divergent due to the bad search region. To tackle this problem, one intuitive solution is to set a large radius so that $\mathbf{p}_j$ is located within the search region. However, it is difficult to pick a uniform radius for all situations since the camera movement is unpredictable. Besides, the growth of search regions may quadratically increase the computational cost for patch matching. Instead, we propose to predict the position of $\mathbf{p}_j$ using gyroscope measurements. As shown in Fig. 1, the predicted result $\check{\mathbf{p}}_j$ is close to $\mathbf{p}_j$. Therefore, $\mathbf{p}_j$ can be successfully tracked by searching the best-matched patch inside the search region around $\check{\mathbf{p}}_j$. In this way, one can use a small search region even though the camera undergoes large motions.

It is also worth noting that the perspective view of the object on I_j is deformed compared with that in I_i due to camera rotations. To find the best-matched patch of the blue square source patch centered on $\mathbf{p}_i$, the affine deformation of target patch needs to be taken into account. Most of the existing methods optimize the affine parameters along with optical flows in a tracking energy function, which increases the computational cost and thus requires a graphics processing unit (GPU) for real-time performance [12], [13]. Instead, we found that the patch affine deformation matrix for each feature can be well calculated by the proposed pixel-aware feature prediction algorithm, without the need for further optimization.

Contributions of this paper include the following:

1) A pixel-aware gyro-aided KLT feature tracker is designed to track features for large camera motions, with a publicly available source code for facilitating the community.
2) A feature prediction algorithm using gyroscope measurements and considering pixel coordinates of each feature is proposed to predict the position and patch affine deformation of target features.
3) A geometric validation mechanism using score-ratio-based homography/fundamental model selection is introduced to filter out mistracks.

Algorithm 1: Pixel-aware gyro-aided feature tracker

Input:
I_i, I_j: two images at time i and j;
$\mathbf{K}$: camera intrinsic matrix;
$\mathbf{R}_c^b$: relative rotation between the camera and gyro;
$\bar{\mathbf{b}}_{g_i}$: gyro bias at time i;
$\{\mathbf{p}_i^1, \mathbf{p}_i^2, ..., \mathbf{p}_i^n\}$: n interesting feature points on I_i;
$\{w_{b_i}, w_{b_{i+1}}, ..., w_{b_{j-1}}\}$: gyro measurements between time i and j.
Output: Feature points $\{\mathbf{p}_j^1, \mathbf{p}_j^2, ..., \mathbf{p}_j^n\}$ tracked on I_j.

Initialize $w = 10$, $L = 3$, $\Gamma = T_H = 5.99$, $T_F = 3.84$;
Integrate the relative gyroscope rotation $\mathbf{R}_{b_{ij}}$ using Eq. (3);
Estimate the relative camera rotation $\mathbf{R}_{ji}$ using Eq. (4);
for *each* $\mathbf{p}_i^k (1 \leq k \leq n)$ **do**
    Predict each feature point $\check{\mathbf{p}}_j^k$ using Eq. (12), check image boundary, and initialize optical flow with $\check{\mathbf{d}} = \check{\mathbf{p}}_j^k - \mathbf{p}_i^k$;
    Estimate the affine deformation matrix $\mathbf{A}$ using Eq. (15);
    Refine $\mathbf{d}$, α, β by optimizing the energy function shown in Eq. (7) using a pyramidal implementation with a maximum level of L;
    The final tracked feature point is $\mathbf{p}_j^k = \mathbf{p}_i^k + \mathbf{d}$;
end
Compute $\mathbf{H}_{ij}$ and $\mathbf{F}_{ij}$ as described in Section V-1);
Compute scores S_H and S_F using Eq. (17) and score ratio R_H using Eq. (18);
if $R_H > 0.45$ **then**
    Select homography model and reject the corresponding outliers;
end
else
    Select fundamental model and reject the corresponding outliers;
end

The pseudocode of our method is shown in Algorithm 1. The remaining part of this paper is organized as follows: Section II reviews the related work. Section III introduces the preliminary knowledge about the gyroscope model, gyroscope integration, camera rotation representation, and feature tracking energy function. Section IV details the proposed pixel-aware gyro-aided feature prediction algorithm, including the feature position prediction and the affine deformation matrix estimation. Section V introduces the geometric validation mechanism. Experiments and analyses are performed in Section VI. Conclusions and future work are described in Section VII.

II. Related Work

Descriptor-based Feature Tracking. The feature tracking can be implemented by matching features between consecutive images. Traditionally, the distinctive feature points and descriptors are first obtained by hand-crafted local feature detectors and descriptors like SIFT [14] and ORB [15], or by learning-based methods like LIFT [16], LF-Net [17], MagicPoint [18], and SuperPoint [19]. The obtained descriptors are invariant to local geometric and photometric transformations. The candidate correspondences can then be obtained by variants of the nearest neighbor or brute force matching on descriptors. Following, the ambiguous and non-distinctive matches are removed by using the second nearest neighbor ratio test, or enforcing matches to be mutual nearest neighbors, or using the random sample consensus (RANSAC) scheme to estimate geometric transformation and then filter out outliers. This technology routine works well for many applications [20], [21] but has the disadvantage of discarding many correct matches, which can be problematic for challenging scenes, such as repetitive and textureless areas.

Descriptor-free Feature Tracking. Descriptor-free methods usually use optical flows or directly learn the feature matches to avoid the descriptor generation and matching phases. For example, Rocco *et al.* [22] developed a neighborhood consensus network for dense matching. It learned local geometric constraints between neighboring correspondences without the need for global geometric models. The computational efficiency of this work was improved in [23] by using sparse convolutions. Sun *et al.* [24] presented a detector-free matching approach, named LoFTR, to establish pixel-wise semi-dense matches with Transformers [25] in a coarse-to-fine manner. Although the learning-based methods have achieved good accuracy and robustness, they still require a GPU to accelerate the computation and it is hard to achieve real-time (e.g., 30 Hz) performance.

Traditionally, the KLT feature tracker [7]–[12] is an extensively used optical flow-based algorithm for feature tracking [26]–[28]. Comparative studies indicate that the Lucas-Kanade algorithms provided accurate results while being significantly more efficient than other optical flow methods [29], [30]. There have been several works to further improve the robustness and runtime performance of the KLT tracker. For example, Senst *et al.* [31] used integral images to speed up the computation of sparse optical flow fields. A strategy for adapting the window size to cope with the generalized aperture problem was introduced in [32]. Ramakrishnan *et al.* [33] also introduced an adaptive window size strategy to tackle the distortion problems. Sinha *et al.* [34], Zach *et al.* [35], and Fassold *et al.* [36] improved the runtime performance by paralleling the algorithm and porting it onto a GPU. Although the KLT tracker has been improved in the literature, it still becomes vulnerable to large inter-image appearance changes. Mathematically, the KLT is a gradient search of the difference between the template and a new image. Initial parameters of KLT optimization are usually set to the same values of the last iteration in a reference frame. If large camera motions cause this initial condition to fall out of the convergence region of the current frame, then it may fail to find the true matched feature. Therefore, adequate estimation of initial parameters is critical for coping with large camera motions.

Gyro-aided Feature Tracking. A gyroscope commonly provides discrete-time samples of angle rates, which can be used to estimate a camera's inter-frame rotation. When a camera rotates quickly, the knowledge of inter-frame rotation can provide good local search regions in which the parameters are more likely to converge to their true solutions. Several efforts have been made to combine the gyroscope data with tracking algorithms. For example, Li *et al.* [37] proposed an unsupervised learning approach that fused gyroscope data into optical flow learning. You *et al.* [38] used gyroscopes to predict the image velocity of feature points in the image plane and then corrected the predicted feature positions by doing local searches. Hwangbo *et al.* [39], [40] presented a modification of the pyramidal KLT algorithm for scenarios where large camera rotation or pan/tilt motion was involved. The authors used gyroscope measurements to estimate a single 2D homography matrix between two consecutive images, which was then used to predict feature positions and pre-warp templates for initializing the KLT feature tracker. The affine photometric model used in [39], [40] has eight parameters allowing robust tracking to camera rotation and outdoor illumination. However, this model leads to a significant computational cost, thus it requires a GPU for ensuring good tracking performance and high frame rates. Ryu *et al.* [41] proposed an IMU-aided feature tracking solution for video stabilization, in which a translational motion model was adopted for computation efficiency in a general CPU. Ravichandar *et al.* [42] proposed a user-driven gyro-aided mutual information tracker that used the information provided by users for template localization. Yao *et al.* [43] also used gyroscope data to estimate a 2D homography matrix for initializing positions of static objects and then used an iterative Earth Mover's Distance algorithm to track moving objects. To deeply exploit the optical flows predicted by gyroscopes, Poling *et al.* [44] proposed to regularize the tracking energy function of KLT by adding a term that penalizes deviations from the prior predicted flows. Chermak *et al.* [45], [46] proposed an IMU-assisted KLT tracker, which was robust to scale change between consecutive images due to the incorporation of an accelerometer sensor. However, this tracker was limited to visual or visual-inertial odometry systems since the IMU information had to be updated over time for continuously and efficiently using accelerometer measurements. The gyro-aided

KLT feature tracking has been extensively explored in the methods mentioned above. However, these methods simply assume that the tracked feature pairs from two images were constrained by the same homography matrix. In contrast, we show in this work that the tracking performance, e.g., the prediction accuracy and the tracking rate, can be improved by considering the feature pixel coordinates when performing the prediction process.

III. Preliminary

This section discusses some preliminary knowledge, including the gyroscope model, gyroscope integration, representation of camera rotations, and the energy function for feature tracking.

A. Gyroscope Model and Integration

The gyroscope measurement model can be formulated as:

$$\omega_b = \bar{\omega}_b + \mathbf{b}_g + \eta_g, \tag{1}$$

where ω_b is the gyroscope output and $\bar{\omega}_b$ is the angle rate that represents the physical dynamic motion property of the sensor suite. The gyroscope output subjects to white sensor noise η_g and slow time-varying bias $\mathbf{b}_g$. In practice, the bias $\mathbf{b}_g$ can be easily obtained by averaging gyroscope outputs collected when the sensor is stationary [47].

Given the gyroscope measurements $\{\omega_{b_i}, \omega_{b_{i+1}}, ..., \omega_{b_{j-1}}\}$ between time i and j, the relative gyroscope rotation $\mathbf{R}_{b_{ij}}$ can be integrated as:

$$\mathbf{R}_{b_{ij}} = \prod_{k=i}^{j-1} \mathrm{Exp}\left(\left(\omega_{b_k} - \mathbf{b}_{g_k} - \eta_{g_k}\right)\Delta t\right), \tag{2}$$

where Δt is the gyroscope sampling interval. $\mathrm{Exp}(\cdot)$ is the "vectorized" version of *exponential map* that transforms a rotational vector $\phi \in \mathbb{R}^3$ to a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ [48], [49]. Since the gyroscope bias changes slowly over time, it can be assumed to remain constant during the time period (i, j), i.e., $\mathbf{b}_{g_{i+1}} = \ldots = \mathbf{b}_{g_j} \approx \bar{\mathbf{b}}_{g_i}$, with $\bar{\mathbf{b}}_{g_i}$ is the gyroscope bias at time i. If the sensor noise is ignored, an approximated expression can be obtained as follows:

$$\mathbf{R}_{b_{ij}} \approx \prod_{k=i}^{j-1} \mathrm{Exp}\left(\left(\omega_{b_k} - \bar{\mathbf{b}}_{g_i}\right)\Delta t\right). \tag{3}$$

B. Camera Rotation Representation

Given the extrinsic orientation $\mathbf{R}_c^b$ between the camera and the gyroscope, the relative camera rotation $\mathbf{R}_{ji}$ from time i to time j can be obtained by transforming from gyroscope rotation, as follows:

$$\mathbf{R}_{ji} = \mathbf{R}_b^c \cdot {\mathbf{R}_{b_{ij}}}^T \cdot \mathbf{R}_c^b = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix}, \tag{4}$$

where $\mathbf{R}_b^c$ is the inverse of $\mathbf{R}_c^b$. $r_{11}, \ldots, r_{33}$ are the nine elements of $\mathbf{R}_{ji}$. $\mathbf{r}_1^T$, $\mathbf{r}_2^T$, and $\mathbf{r}_3^T$ are three row vectors of $\mathbf{R}_{ji}$.

C. Feature Tracking Energy Function

Let I_i and I_j be two 2D images. The two quantities $I_i(\mathbf{p}) = I_i(u, v)$ and $I_j(\mathbf{p}) = I_j(u, v)$ are then the gray-scale values of the two images at location $\mathbf{p} = [u, v]^T$, where u and v are two pixel coordinates of a generic image point $\mathbf{p}$. Considering an image point $\mathbf{p}_i = [u_i, v_i]^T$ on I_i, the goal of feature tracking is to find the location $\mathbf{p}_j = [u_j, v_j]^T = [u_i + du, v_i + dv]^T$ on I_j such as $I_i(\mathbf{p}_i)$ and $I_j(\mathbf{p}_j)$ are "similar". The vector $\mathbf{d} = [du, dv]^T$ is the image velocity at $\mathbf{p}_i$, which is known as the optical flow at $\mathbf{p}_i$. In addition to a translation component $\mathbf{d}$, we also assume that the image undergoes an affine deformation and an illumination change between I_i and I_j in the vicinity of $\mathbf{p}_i$ and $\mathbf{p}_j$.

For the assumption of affine deformation, we introduce an affine transformation matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 1 + d_{xx} & d_{xy} \\ d_{yx} & 1 + d_{yy} \end{bmatrix}, \tag{5}$$

where the four coefficients d_{xx}, d_{xy}, d_{yx}, and d_{yy} characterize the affine deformation of the image patch. For the assumption of illumination change, we introduce two coefficients α and β to account for changes in image contrast and image brightness respectively. The pixel error in the vicinity of $\mathbf{p}_i$ is defined as:

$$e(\mathbf{x}|\mathbf{p}_i) = (1 + \alpha) I_i(\mathbf{x} + \mathbf{p}_i) - I_j(\mathbf{A}\mathbf{x} + \mathbf{d} + \mathbf{p}_i) - \beta, \tag{6}$$

where $\mathbf{x} = [x, y]^T, x, y \in [-w, w]$. The integer variable w is the half-width of a patch, which lets the size of patch window as $(2w + 1) \times (2w + 1)$.

The objective of feature tracking is then to find the vector $\mathbf{d}$ and the coefficients α and β that minimize the patch energy function ϵ defined as follows:

$$\epsilon(\mathbf{d}, \alpha, \beta) = \epsilon(du, dv, \alpha, \beta) = \sum_{x=-w}^{w} \sum_{y=-w}^{w} \|e(\mathbf{x}|\mathbf{p}_i)\|^2. \tag{7}$$

Note that we do not optimize the affine deformation matrix $\mathbf{A}$ in (7) since this matrix is estimated by the proposed pixel-aware gyro-aided feature prediction algorithm. The detail of the affine deformation estimation is derived in Section IV-B.

IV. Pixel-Aware Gyro-Aided Feature Prediction

The fundamental assumption for appearance PatchMatch-based feature tracking algorithms like KLT is a small inter-frame change. Considering only the equation (7), it is preferable to have $|du| \leq w \cdot 2^L$ and $|dv| \leq w \cdot 2^L$, with L is the maximum pyramidal level if a pyramidal implementation is applied. For example, if we set $w = 5$ and $L = 3$, then the successfully tracked features should satisfy the condition of $|du| \leq 40$ and $|dv| \leq 40$. In practice, this condition may be easily violated due to large camera motions. To overcome this problem, one intuitive approach is to set a larger patch window. However, it may quadratically increase the computational complexity. On the contrary, we in this paper propose to utilize gyroscope measurements to predict prior pieces of information, i.e., the initial estimates of optical flow $\check{\mathbf{d}}$ and the affine deformation matrix $\mathbf{A}$, for facilitating the feature tracking process. This idea benefits us to adopt a small patch window for solving equation (7) even though the camera undergoes large motions.

A similar idea has been investigated in the literature [38]–[44]. However, most of the existing methods only used gyroscope measurements to perform the feature prediction since they simply assumed that all tracked feature pairs were constrained by the same homography matrix. Unlike these methods, our method can adjust the feature prediction matrix for each feature pair according to the source feature's pixel coordinates, which benefits us to achieve accurate feature prediction and thus improves the tracking performance.

A. Feature Position Prediction

Points in the camera reference space are projected according to the pinhole camera model [50]. Denoting the coordinate of a 3D point $\mathbf{P}$ in the camera coordinate system of frame I_i as $\mathbf{P}_i = [X_i, Y_i, Z_i]^T$, the projection onto the image plane can be represented as:

$$\mathbf{p}_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \frac{1}{Z_i} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \stackrel{\text{def}}{=} \frac{1}{Z_i}\mathbf{K}\mathbf{P}_i, \quad (8)$$

where the 2D image point $\mathbf{p}_i$ is represented in homogeneous coordinate for convenience. $\mathbf{K}$ is the camera intrinsic matrix. f_x, f_y are the focal lengths. c_x, c_y are the principal point coordinates. Using (8), the back-projection equation can be represented as: $\mathbf{P}_i = Z_i\mathbf{K}^{-1}\mathbf{p}_i$. Note that the scalar Z_i is usually called the depth value, which is not the distance from the origin of the camera coordinate system to the 3D point but the projection of distance on the z-axis, thus the depth value is changed when the camera undergoes pure-rotation motion.

Assuming the coordinate of 3D point $\mathbf{P}$ in the camera coordinate system of frame I_j is represented as $\mathbf{P}_j = [X_j, Y_j, Z_j]^T$, if the camera undergoes a rotation $\mathbf{R}_{ji}$ and a translation $\mathbf{t}_{ji}$ ($\mathbf{t}_{ji} = [t_1, t_2, t_3]^T$) from time i to time j, then $\mathbf{P}_j$ can be transformed from $\mathbf{P}_i$ as follows:

$$\mathbf{P}_j = \mathbf{R}_{ji}\mathbf{P}_i + \mathbf{t}_{ji} \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{r}_1^T\mathbf{P}_i + t_1 \\ \mathbf{r}_2^T\mathbf{P}_i + t_2 \\ \mathbf{r}_3^T\mathbf{P}_i + t_3 \end{bmatrix}. \quad (9)$$

Substituting the back-projection equation of (8) into (9) and using the projection of $\mathbf{p}_j$, the relationship between $\mathbf{p}_i$ and $\mathbf{p}_j$ can be derived as follows:

$$\begin{aligned} \mathbf{p}_j &= \frac{1}{Z_j}\mathbf{K}\mathbf{P}_j \\ &= \frac{1}{Z_j}\mathbf{K}(\mathbf{R}_{ji}\mathbf{P}_i + \mathbf{t}_{ji}) \\ &= \frac{Z_i}{\mathbf{r}_3^T\mathbf{P}_i + t_3}\mathbf{K}\mathbf{R}_{ji}\mathbf{K}^{-1}\mathbf{p}_i + \frac{\mathbf{K}\mathbf{t}_{ji}}{\mathbf{r}_3^T\mathbf{P}_i + t_3} \\ &= \frac{\mathbf{K}\mathbf{R}_{ji}\mathbf{K}^{-1}\mathbf{p}_i}{\mathbf{r}_3^T\mathbf{K}^{-1}\mathbf{p}_i + t_3/Z_i} + \frac{\mathbf{K}\mathbf{t}_{ji}}{Z_i\mathbf{r}_3^T\mathbf{K}^{-1}\mathbf{p}_i + t_3}. \end{aligned} \quad (10)$$

Discussion: In (10), the prediction of $\mathbf{p}_j$ from $\mathbf{p}_i$ involves camera rotation and translation. However, large optical flows are mainly due to camera rotation rather than translation simply because cameras are often free to rotate much faster than they can move through their environments (or relative to other visible objects). For illustration, consider a common camera that has a 50° diagonal angle of view, a focal length of 400, a resolution of 600×800 pixels, and a frame rate of 30 fps. If the camera achieves rotation rates of $330^\circ/s$, it will induce motion in the image plane of up to 22% (220 pixels) of the image diagonal between consecutive frames. For a non-rotating camera viewing an object placed at 3 meters far away from the camera, to generate a comparable flow in the image plane, the camera or object should move at least $178.2\ \mathrm{km/h}$. Objects and cameras are not moving nearly this fast in many situations, and camera rotation is, therefore, the dominant source of optical flow.

In practical applications, the following two cases often occur. Case 1: the camera undergoes small or no translation motion like pure rotation; Case 2: the camera translation motion between two consecutive images is much less than the feature depth. Using these two cases, we can make the following approximation:

$$\left.\begin{array}{l} \text{case 1}: \mathbf{t}_{ji} \to \mathbf{0} \\ \text{case 2}: \|\mathbf{t}_{ji}\| \ll Z_i \end{array}\right\} \Rightarrow \left.\begin{array}{l} t_3/Z_i \approx 0 \\ \mathbf{t}_{ji}/(Z_i\mathbf{r}_3^T\mathbf{K}^{-1}\mathbf{p}_i + t_3) \approx \mathbf{0} \end{array}\right\}. \quad (11)$$

According the above discussion, the equation (10) can be simplified as:

$$\check{\mathbf{p}}_j \approx \underbrace{\frac{1}{\mathbf{r}_3^T\mathbf{K}^{-1}\mathbf{p}_i}\mathbf{K}\mathbf{R}_{ji}\mathbf{K}^{-1}}_{feature\ prediction\ matrix} \cdot \mathbf{p}_i, \quad (12)$$

where $\check{\mathbf{p}}_j$ denotes the predicted position of $\mathbf{p}_j$. The initial estimate of optical flow can then be calculated as $\check{\mathbf{d}} = \check{\mathbf{p}}_j - \mathbf{p}_i$. It is worth noting that many existing methods [38]–[44] assume that all the feature pairs satisfy a single 2D homography prediction matrix as $\check{\mathbf{p}}_j \approx \mathbf{K}\mathbf{R}_{ji}\mathbf{K}^{-1}\mathbf{p}_i$, which is only determined by the gyroscope measurements and the camera intrinsic matrix. We call these methods as "single homography prediction" in the following. Different from the existing methods, the prediction equation shown in (12) is specific for each feature since the pixel coordinates are involved in the coefficient term $1/(\mathbf{r}_3^T\mathbf{K}^{-1}\mathbf{p}_i)$. As a result, equation (12) is termed as "pixel-aware gyro-aided feature prediction". Fig. 2 shows the comparison between the optical flows predicted by the single homography prediction and our pixel-aware prediction. As shown in Fig. 2 (e) and (f), when a camera performs pure rotation around the z-axis of the camera coordinate system, the two methods predict the same optical flows since all the feature depths remain unchanged. However, when the camera rotates around the x-axis (see Fig. 2 (a) and (b)) or the y-axis (see Fig. 2 (c) and (d)), the optical flows predicted by single homography prediction are asymmetric and unreasonable, while our predictions are reasonable since they are axisymmetric. The reason is that our method accounts for the depth change when deriving the feature prediction matrix, while most existing methods ignore the impact of depth change. The advantage of our method is further analyzed in the experimental section.

B. Affine Deformation Matrix Estimation

In our method, the patch affine deformation is taken into account when minimizing the patch energy function (7). As shown in Fig.1, the image patch around $\mathbf{p}_i$ is drawn as a

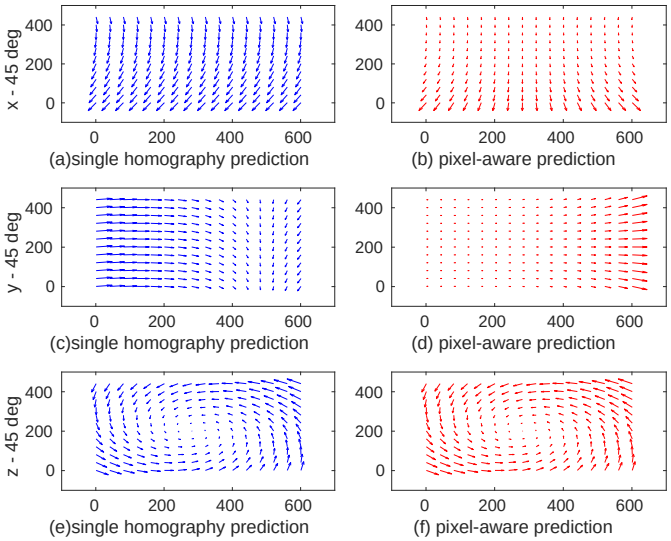


Fig. 2: Predicted optical flows when a camera respectively rotates 45 degrees around the x-axis, y-axis, and z-axis of the camera coordinate system. (a), (c), and (e): the optical flows predicted by single homography prediction that most existing methods used; (b), (d), and (f): the optical flows predicted by our pixel-aware gyro-aided feature prediction. The optical flows in (b) and (d) are axisymmetric and thus more reasonable than (a) and (c).

blue square. Due to camera rotation, its best-matched patch is deformed as indicated by the orange square patch around $\check{\mathbf{p}}_j$. In the following, we provide the detailed derivation for estimating the affine deformation matrix $\mathbf{A}$ of each feature pair $(\mathbf{p}_i, \mathbf{p}_j)$.

Letting the size of patch windows as $(2w+1)\times(2w+1)$, and denoting the four corners of the patch around $\mathbf{p}_i$ as $\mathbf{p}_i^{tl}$, $\mathbf{p}_i^{tr}$, $\mathbf{p}_i^{bl}$, and $\mathbf{p}_i^{br}$, we have:

$$\begin{bmatrix} \mathbf{p}_i^{tl\,T} \\ \mathbf{p}_i^{tr\,T} \\ \mathbf{p}_i^{bl\,T} \\ \mathbf{p}_i^{br\,T} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_i{}^T \\ \mathbf{p}_i{}^T \\ \mathbf{p}_i{}^T \\ \mathbf{p}_i{}^T \end{bmatrix} + \underbrace{\begin{bmatrix} -w & -w \\ w & -w \\ -w & w \\ w & w \end{bmatrix}}_{\mathbf{B}^T}, \tag{13}$$

where $\mathbf{B} \in \mathbb{R}^{2\times 4}$ is a fixed corner coordinate shift matrix determined by the patch window size. Using the equation (12), the four corners of the deformed patch around $\mathbf{p}_j$ can be predicted. Here, we denote the corresponding predicted corners as $\check{\mathbf{p}}_j^{tl}$, $\check{\mathbf{p}}_j^{tr}$, $\check{\mathbf{p}}_j^{bl}$, and $\check{\mathbf{p}}_j^{br}$. A coordinate shift matrix $\mathbf{C} \in \mathbb{R}^{2\times 4}$ can then be defined as:

$$\mathbf{C}^T = \begin{bmatrix} du^{tl} & dv^{tl} \\ du^{tr} & dv^{tr} \\ du^{bl} & dv^{bl} \\ du^{br} & dv^{br} \end{bmatrix} = \begin{bmatrix} (\check{\mathbf{p}}_j^{tl} - \check{\mathbf{p}}_j)^T \\ (\check{\mathbf{p}}_j^{tr} - \check{\mathbf{p}}_j)^T \\ (\check{\mathbf{p}}_j^{bl} - \check{\mathbf{p}}_j)^T \\ (\check{\mathbf{p}}_j^{br} - \check{\mathbf{p}}_j)^T \end{bmatrix}, \tag{14}$$

where $[du^{(\cdot)}, dv^{(\cdot)}]^T$ is the vector pointed from $\check{\mathbf{p}}_j$ to $\check{\mathbf{p}}_j^{(\cdot)}$. Using the matrices $\mathbf{B}$ and $\mathbf{C}$, the patch affine deformation matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$ for point $\mathbf{p}_i$ can be calculated as follows:

$$\mathbf{A} = \mathbf{C}\mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}. \tag{15}$$

V. Geometric Validation

Assuming there are N feature points in the first image and most of them have been tracked in the second image. In real applications, the mistracked problem that the tracked result is a few pixels away from its true position is inevitable due to the local minimum of the feature tracking energy function. In addition, feature pairs belonging to moving objects should be filtered out because they are inconsistent with camera motion constraints. To overcome the problem, we propose to use geometric validation to filter out mistracked features. First, two geometric models, i.e., a homography matrix assuming a planar scene and a fundamental matrix assuming a nonplanar scene, are computed using all the tracked feature pairs. Then, a score-ratio-based homography/fundamental model selection inspired by Mur-Artal's work [20], is adopted to select a suitable model. Finally, the feature pairs with large reprojection errors are identified as mistracked features. The steps of geometric validation are as follows:

1) Parallel computation of the two models: The homography matrix $\mathbf{H}_{ij}$ and fundamental matrix $\mathbf{F}_{ij}$ are modeled as:

$$\mathbf{p}_i = \mathbf{H}_{ij}\mathbf{p}_j, \qquad \mathbf{p}_i^T\mathbf{F}_{ij}\mathbf{p}_j = 0. \tag{16}$$

They can be computed in parallel threads with the normalized four-point DLT and eight-point algorithms, respectively, inside a RANSAC scheme as explained in [50]. After that, a tracking score S_M for each model M (H for the homography matrix, F for the fundamental matrix) is computed as:

$$S_M = \sum_k^N \left(\rho_M(d_{ij}^2(\mathbf{p}_i^k, \mathbf{p}_j^k, M)) + \rho_M(d_{ji}^2(\mathbf{p}_i^k, \mathbf{p}_j^k, M))\right),$$
$$\rho_M(d^2) = \begin{cases} \Gamma - d^2, & \text{if } d^2 < T_M \\ 0, & \text{if } d^2 \geq T_M, \quad outliers \end{cases} \tag{17}$$

where d_{ij}^2 and d_{ji}^2 are the symmetric transfer errors from one frame to the other. For the homography model, $d_{ij} = \mathbf{p}_i - \mathbf{H}_{ij}\mathbf{p}_j$ and $d_{ji} = \mathbf{p}_j - \mathbf{H}_{ij}^{-1}\mathbf{p}_i$. For the fundamental model, d_{ij} is the distance from point $\mathbf{p}_i$ to the epipolar line $l_i = \mathbf{F}_{ij}\mathbf{p}_j$, and d_{ji} is the distance from point $\mathbf{p}_j$ to the epipolar line $l_j = \mathbf{F}_{ij}^T\mathbf{p}_i$. T_M is the outlier rejection threshold based on the χ^2 test at 95% ($T_H = 5.99$, $T_F = 3.84$). Γ is defined equal to T_H so that both models score equally for the same d in their inlier region.

2) Score-ratio-based model selection: The homography can well explain the cases where the scene is planar, nearly planar or the camera moves with low parallax. However, a fundamental matrix can also be found in these cases although it is not well constrained. On the other hand, a nonplanar scene with enough parallax can only be explained by the fundamental model, but a homography matrix can also be found explaining a subset of matches if they lie on a plane or they have low parallax. As a consequence, it is necessary to introduce a criterion for model selection. In our method, we use the score ratio to select the proper model. The score ratio is computed as follows:

$$R_H = \frac{S_H}{S_H + S_F}. \tag{18}$$

Following the work of [20], the homography model is selected if $R_H > 0.45$, which adequately captures the planar and low parallax cases. Otherwise, the fundamental model is selected. Once a model is selected, the corresponding outliers are identified as mistracked features.

VI. Experiments

The performance of the proposed pixel-aware gyro-aided KLT feature tracker, referred to as "GA KLT - Ours", is examined and compared with a pyramidal image-only KLT tracker implemented based on OpenCV 3.4.7 [51] that regards source points as initial estimates of target points; a gyro-aided KLT tracker[1] proposed by Hwangbo [39], [40] that uses a single 2D homography matrix for feature prediction, which is referred to as "GA KLT - Hwangbo"; a learning-based local feature matching algorithm named SuperGlue [52]. For ablation study, a simplified configuration of the proposed method that uses a single 2D homography prediction to replace equation (12) is also evaluated, which is referred to as "GA KLT - Ours (sc)".

In the following, the experiments are performed on public and real-world sequences. There include an indoor DeskScene sequence[2] (640×480 at 30 Hz, gyro at 100 Hz) taken by a hand-held device in front of a desk [39]; eleven indoor sequences (752×480 at 20 Hz, gyro at 200 Hz) of the EuRoC dataset [53] captured by a micro aerial vehicle (MAV) that flew under conditions ranging from slow flights with good visual conditions to dynamic flights with motion blur and poor illumination; and seven real-world sequences recorded from a hand-held Intel RealSense Depth Camera D435i[3] device (640×480 at 15 Hz, gyro at 250 Hz). For convenient, we term the inliers that pass the geometric validation as *good tracks*, and the predicted features that satisfy the image boundary constraint as *valid predicts*. The rate of good tracks (RGT), i.e., $N_{good\_tracks}/N_{features\_to\_be\_tracked} * 100\%$, the rate of good predicts (RGP), i.e., $N_{good\_tracks}/N_{valid\_predicts} * 100\%$, the tracking scores (TS), the prediction errors, i.e., the euclidian distance between the predicted position and its true position, and the tracking frame length are reported. All experiments are carried out with an Intel CPU i7-9750H (12 cores @2.60 GHz), a GeForce GTX 1660Ti GPU laptop computer with 16 GB RAM. The proposed method and its simplified configuration are performed on pure CPU, while the "GA KLT - Hwangbo" and SuperGlue are performed rely on GPU.

A. *DeskScene Sequence*

The tracking performance of the proposed method and other compared methods are exhaustively evaluated on the indoor DeskScene sequence. It contains good illumination conditions and various types of camera motions including panning, tilting, and moving forward with rolling, which benefits us to analyze the tracking performance under different motion conditions. To test the robustness performance of these methods for fast-moving cameras, the image stream is interval-sampled with a rate of 1, 2, 3, 4 to proportionally speed up camera movement. The processed sequences are more challenging since the rotation and translation are faster and the field of view (FOV) overlap is lower than the raw sequence. For a fair comparison, all methods use the same features to perform the tracking process, and the patch size is equally set as 21×21 pixels. In the following, we first evaluate the performance of RGT and RGP under various interval-sampled rates and then present the histogram of feature tracking frame length. After that, a comparison of initial prediction errors is provided.

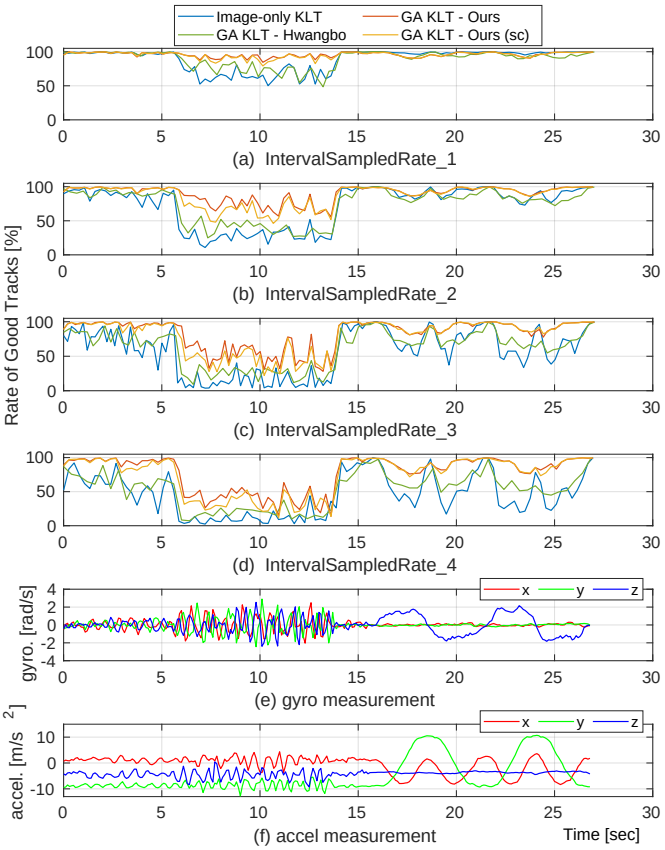


Fig. 3: Feature tracking on the DeskScene sequence. (a) - (d) show the RGT when the image stream is interval-sampled with rates of 1, 2, 3, 4 for proportionally speeding up camera movements. (e) and (d) are the gyroscope and accelerometer measurements. **Best viewed in color.**

1) Performance of RGT and RGP: In this experiment, the maximum number of features to be tracked in each frame is set as 500. Specifically, we first detect 500 features on the first frame and then track and keep them in the following frames. If the number of good features (i.e., inliers that pass the geometric validation) is less than 500, new features will be detected and registered for making up. The feature tracked results w.r.t different interval-sampled rates are shown in Fig. 3 (a)-(d). The gyroscope and accelerometer measurements of the sequence are also plotted in Fig. 3 (e) and (f). For panning (0-6s) and moving forward with rolling motions (16-22s), Fig. 3 (a) shows that almost all features can be tracked by all the methods on the raw sequence. However, with the interval-sampled rate increases, the RGT of "Image-only KLT"

[1] http://www.cs.cmu.edu/~myung/IMU_KLT/
[2] http://www.cs.cmu.edu/~myung/IMU_KLT/data_desk_scene.zip
[3] https://www.intelrealsense.com/depth-camera-d435i/

(a) GA KLT - Ours (patch size: 21x21) (b) GA KLT - Ours (sc) (patch size: 21x21) (c) Image-only KLT (patch size: 21x21) (d) Image-only KLT (patch size: 41x41)

Fig. 4: **Qualitative feature tracking.** The feature tracked results of different methods when the camera undergoes an up-tilting motion. Top row: the reference frame; Bottom row: the final tracked results in current frame; Middle row: the intermediate results to explain the feature tracking process. The results on each column are respectively: (a): GA KLT - Ours; (b): GA KLT - Ours (sc); (c)-(d): Image-only KLT. The patch sizes are set as 21×21 pixels for (a)-(c) and 41×41 for (d). In the top and bottom rows, green dots are for good tracks, red dots are for lost tracks, and blue dots are the newly detected and registered features. In the middle row, yellow circles are for predicted feature positions, green dots are for patch-matched results, and red circles are for the tracks that would be filtered out by the geometric validation. The white lines show the optical flows.

TABLE I: Rates of Good Tracks, Good Predicts, and Tracking Scores under Different Interval-Sampled Rates.

Tracker	Interval-Sampled Rates				avg.	avg.	avg.
	1	2	3	4	RGT	RGP	TS
Image-only KLT	88.87	71.11	56.02	44.48	65.12	79.71	3647.81
GA KLT - Hwangbo	89.67	73.68	61.81	52.57	69.43	84.63	3953.91
GA KLT - Ours (sc)	95.17	86.85	78.50	72.15	83.17	89.79	4768.89
GA KLT - Ours	**95.76**	**89.66**	**81.81**	**75.29**	**85.63**	**92.51**	**4906.78**

(i.e., blue curves) and "GA KLT - Hwangbo" (i.e., green curves) decreases moderately, nevertheless the performance degradations of our method (i.e., orange curves) and its simplified configuration (i.e., yellow curves) are negligible. For tilting motions (6-13s), the gyroscope and accelerometer measurements change drastically due to the severe shaking of the device. Since the FOV overlap between consecutive frames drops with the increase of interval-sampled rates, the RGT of all methods decreases significantly. However, the RGTs of our method and its simplified configuration are still higher than that of "Image-only KLT" and "GA KLT - Hwangbo".

The rates of good tracks, good predicts, and the average tracking scores over the whole sequence are shown in Table I. The average number of good tracks of "GA KLT - Our (sc)" and "GA KLT - Hwangbo" are respectively 415.85 (83.17%) and 347.15 (69.43%), which are both higher than the 325.6 (65.12%) of "Image-only KLT" tracker. The result verifies that the feature tracking performance can be improved by incorporating gyroscope measurements for predicting feature positions. It is also worth noting that the average number of good tracks of "GA KLT - Ours" is 428.15 (85.63%), which is even higher than that of "GA KLT - Our (sc)". The out-performance indicates that the performance can be further improved by considering pixel coordinates in the feature prediction.

An example of feature tracking of different methods when the camera undergoes an up-tilting motion is shown in Fig. 4. The patch sizes are set as 21×21 pixels for Fig. 4 (a) - (c) and 41×41 pixels for Fig. 4 (d). Compare Fig. 4 (a) with Fig. 4 (b), it shows that most of the features located in the bottom left region of the reference frame are lost tracked by the "GA KLT - Ours (sc)", while they can be successfully tracked by "GA KLT - Ours". The number of good tracks in Fig. 4 (a) is 494, which is 45.7% higher than that, i.e., 339,

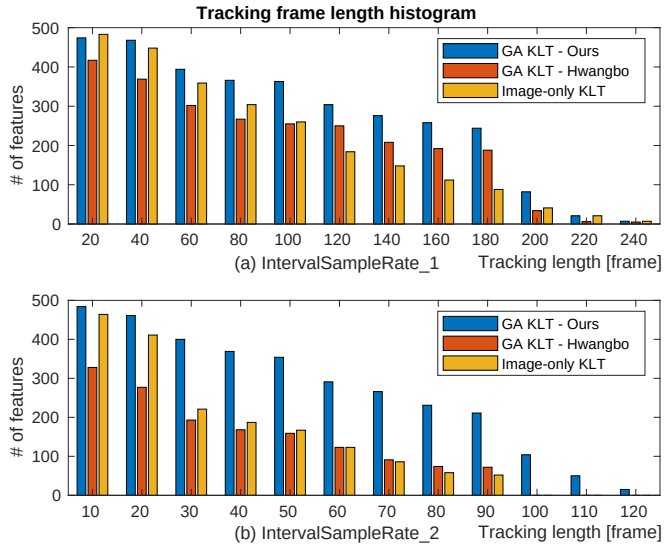


Fig. 5: The tracking frame length histogram of 500 features in the DeskScene with no new registration when the image stream is interval-sampled with rates of 1 (a) and 2 (b) for speeding up camera movements.

in Fig. 4 (b). The average prediction errors are respectively 12.05 pixels and 17.67 pixels in Fig. 4 (a) and Fig. 4 (b). These out-performances illustrate that incorporating feature pixel coordinates in the prediction process improves the feature tracking performance and the prediction accuracy.

2) Performance of Tracking Frame Length: To evaluate the ability of long frame tracking when the camera moves at different speeds, we compare the tracking length histograms of "GA KLT - Ours", "GA KLT - Hwangbo", and "Image-only KLT" when the interval-sampled rate of DeskScene sequence is set to 1 and 2. In this experiment, once 500 features are selected on the first image frame, no new features would be registered on the following sequences. The histograms in Fig. 5 show that our method has more features in both the raw and the speeded-up sequences. Compare Fig. 5 (a) and Fig. 5 (b), it shows that the quantity of tracked features of "GA KLT - Hwangbo" and "Image-only KLT" decreases when the interval-sampled rate increases, while our method achieves consistent results. The out-performance owns to the accurate initial predictions of the proposed method.

3) Initial Prediction Errors: Fig. 6 plots the error distribution curves of initial predictions when the interval-sampled rate is set to 2, where the initial prediction error of "Image-only KLT" is the distance between the source feature $\mathbf{p}_i$ and the target feature $\mathbf{p}_j$, while the error of "GA KLT - Ours" and "GA KLT - Hwangbo" [39], [40] is the distance between the predicted position $\check{\mathbf{p}}_j$ and $\mathbf{p}_j$. The corresponding statistic results, i.e., the mean, maximum, minimum, standard deviation, and median values, are shown in Table II. The results show that "GA KLT - Hwangbo" reduces the mean and standard deviation of initial prediction errors from 22.57 and 14.17 pixels of "Image-only KLT" to 8.56 and 5.04 pixels, while the values can be further reduced to 6.67 and 4.27 pixels by our method.

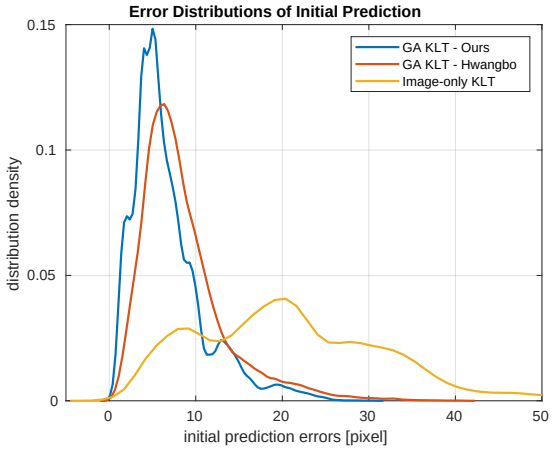


Fig. 6: Error distribution curves of initial predictions for the "GA KLT - Ours", "GA KLT - Hwangbo", and "Image-only KLT" when the interval-sampled rate is 2.

TABLE II: Statistics of Initial Prediction Errors (Unit: pixel).

	mean	max	min	staDev	median
GA KLT - Ours	6.76	30.57	0.19	4.27	5.66
GA KLT - Hwangbo	8.56	40.92	0.24	5.04	7.39
Image-only KLT	22.57	114.01	0.09	14.17	20.35

B. *EuRoC Dataset*

The tracking performance of the proposed method is also evaluated using the indoor EuRoC dataset [53]. This dataset was recorded by a Skybotix stereo VI-sensor [54] installed on a MAV. It includes eleven sequences. Five of them were collected in a large industrial machine hall (denoted as MH01 to MH05), and the other six were collected in a lab room (denoted as V101 to V103 and V201 to V203). These sequences are classified into easy, medium, and difficult grades by considering the illumination, texture, motion velocity, and motion blur.

In this experiment, our method is compared with the "Image-only KLT" tracker and a learning-based local feature matching algorithm named SuperGlue [52]. The SuperGlue accepts two sets of interest points with their descriptors as input and learns their matches with a graph neural network (GNN). Since SuperGlue learns priors over geometric transformations and regularities of the 3D world through end-to-end training from image pairs, it achieves impressive performance and sets a new state-of-the-art in learning-based local feature matching domain. In the public code of SuperGlue, the authors use SuperPoint [19], a self-supervised interest point detection and description approach, to detect features and descriptors of two images and then match them using a GNN. For a fair comparison, the feature matches output by SuperGlue are checked by our geometric validation algorithm. In addition, we utilize SuperPoint as the feature detector to perform the proposed method and the "Image-only KLT" tracker. Different from the processing of the DeskScene sequence that keeps good tracks and registers new features on the following frames, in this experiment, we abandon the previous good tracks and re-detect all features on each reference frame. The patch sizes

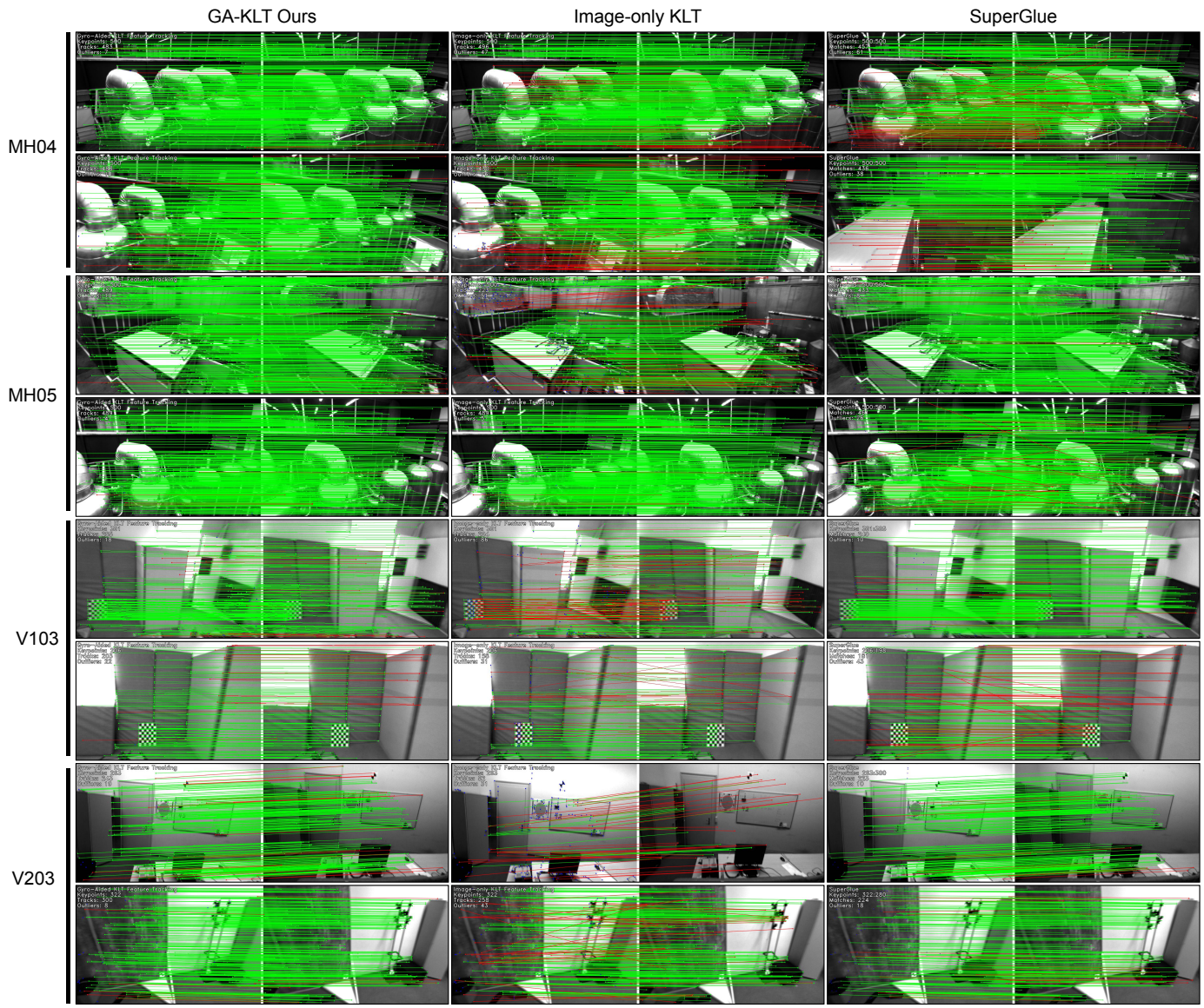


Fig. 7: **Qualitative results.** The proposed method is compared to "Image-only KLT" and SuperGlue [52] in the EuRoC dataset. The good tracks are colored in green, and the mistracks that filtered out by the introduced geometric validation are colored in red. **Best to zoom in and viewed in color.**

of "GA KLT - Ours" and "Image-only KLT" are both set as 11×11.

Fig. 7 plots the comparison of qualitative results on MH04, MH05, V103, and V203 sequences. It is obvious that "Image-only KLT" and SuperGlue have a lot of mistracks, while most of the features we track are good tracking. Table III shows the quantitative tracking performance on the whole EuRoC dataset. The average RGT and RGP of our method are respectively 89.61% and 93.65%, which are both higher than the results of "Image-only KLT" (84.94% and 90.99%) and SuperGlue (78.75% and 92.07%). It's worth noting that the RGT of "Image-only KLT" is higher than that of SuperGlue, while the RGP of the former is lower than the latter. The reason is that the "Image-only KLT" outputs the potential track for each feature point, while the SuperGlue only outputs strong matches. This strategy leads to a higher RGP of SuperGlue than "Image-only KLT". In comparison, our method tracks most features and achieves high RGP on most of the sequences with the help of gyroscope measurements.

C. *Real-World Experiments*

The proposed method is tested in real-world experiments, using the Intel RealSense Depth Camera D435i. This sensor contains a three-axis accelerometer with a sample rate of 250 Hz, a three-axis gyroscope with sample rates of 200 or 400 Hz, two global infrared cameras, and one rolling shutter RGB camera. Following the process in [49], the infrared camera was launched at 15 Hz with a resolution of 640×480. To synchronize with the accelerometer, the gyroscope was launched at 400 Hz and then down-sampled and linearly interpolated to 250 Hz. In this experiment, we hand-held the

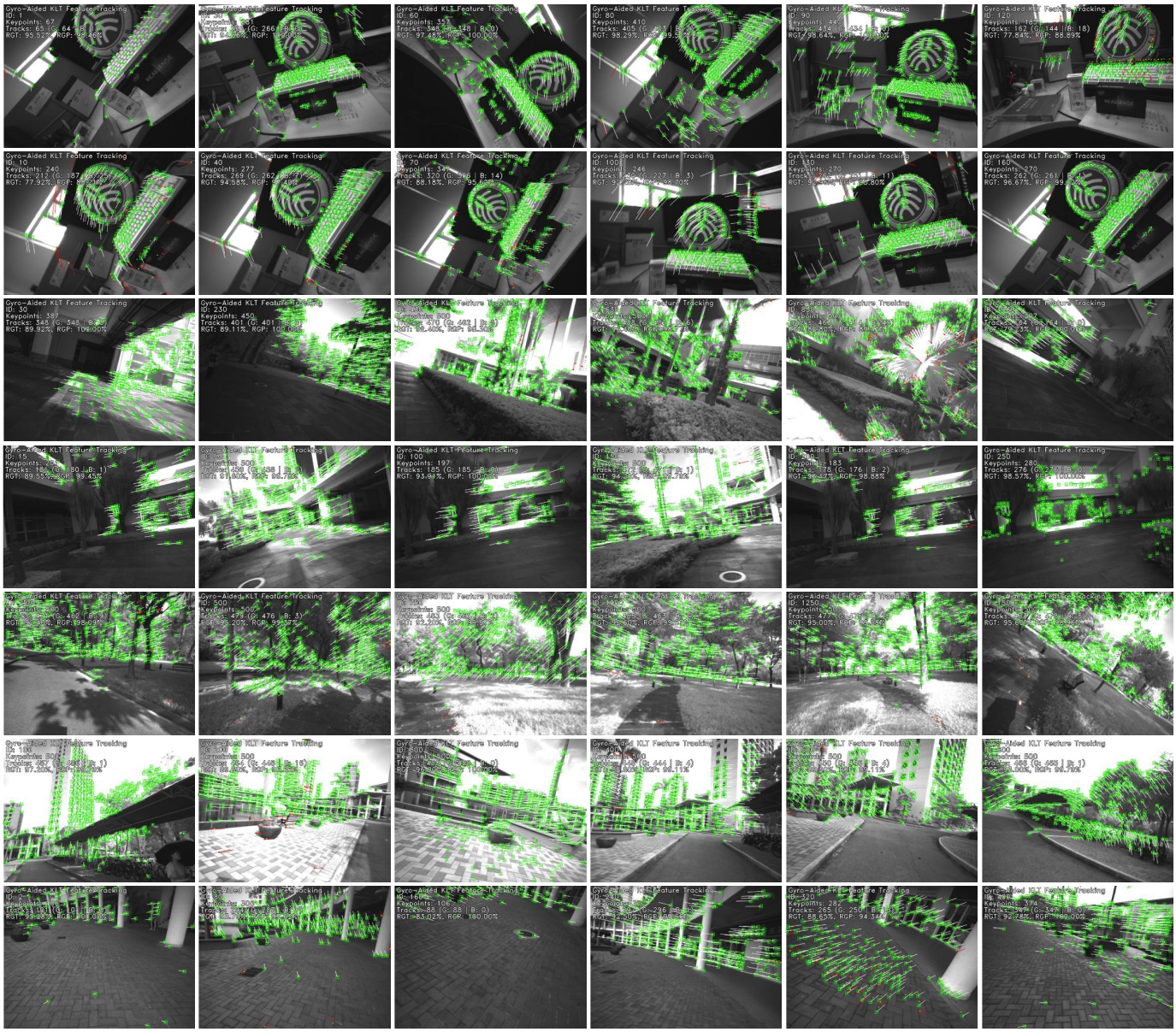


Fig. 8: Qualitative results of the proposed method on the real-world sequences. **Best viewed in color.**

device and recorded seven sequences[4]. Two of them were collected in front of an indoor desk with fast camera rotations, and the other five were collected on an outdoor campus with random motions.

Fig. 8 shows the qualitative feature tracking results of the proposed method with patch size as 11×11. It shows that the features can be successfully tracked even though these sequences are challenging as there exist fast camera rotation, motion blur, illumination change, weak and repeats texture, etc. The quantitative tracking performance compared with "Image-only KLT" and SuperGlue [52] is provided in Table IV. The average RGT and RGP of our method are 90.87% and 97.17% respectively, indicating that most of the features can be accurately predicted and successfully tracked by our method. The two values are both higher than that of "Image-only KLT" (RGT: 66.03%, RGP: 87.98%) and SuperGlue (RGT: 76.50%, RGP: 90.97%), indicating the 24.84% and 12.12% improvements of RGT compared with the "Image-only KLT" and SuperGlue, respectively.

D. *Parameter Sensitivity*

In KLT-based trackers, the patch size influences the accuracy, robustness, and computational complexity of the feature tracking. In the following, the impact of the patch size parameter on the feature tracking performance is analyzed using the DeskScene sequence with an interval-sampled rate as 2.

The number of good tracks w.r.t various patch sizes are shown in Fig. 9, in which Fig. 9 (a) is the result when the camera undergoes tilting motions and Fig. 9 (b) the moving forward with rolling motions. For tilting motions, Fig. 9 (a)

[4]The sequences are available at: https://drive.google.com/drive/folders/1oBaiijQvzDb9SezgaVPm1ABosvTcztJ7?usp=sharing

TABLE III: Quantitative Tracking Performance on the EuRoC Dataset.[1]

	GA KLT - Ours		Image-only KLT		SuperGlue [52]	
	RGT	RGP	RGT	RGP	RGT	RGP
MH01	93.56	96.17	**94.21**	**96.50**	82.01	94.71
MH02	91.63	94.69	**92.68**	**95.70**	79.98	93.24
MH03	**91.75**	**94.82**	89.59	94.39	79.08	92.26
MH04	**91.53**	**95.03**	88.75	94.43	81.01	93.64
MH05	**91.26**	**94.82**	89.66	94.73	81.49	94.01
V101	**93.64**	**96.33**	91.71	94.54	83.76	94.87
V102	**85.17**	**90.98**	75.86	85.23	76.41	90.97
V103	**87.52**	**92.67**	72.28	82.40	73.99	90.50
V201	**92.97**	**95.75**	92.14	94.46	84.11	94.76
V202	**85.19**	**90.65**	78.77	87.66	74.79	87.70
V203	**81.44**	**88.25**	68.70	80.89	69.66	86.12
Avg.	**89.61**	**93.65**	84.94	90.99	78.75	92.07

[1] The best and the second best of RGT (RGP) are colored in **bold red** (**bold blue**) and green (cyan) respectively.

TABLE IV: Quantitative Tracking Performance on the Real-World Experiments.[1]

	GA KLT - Ours		Image-only KLT		SuperGlue [52]	
	RGT	RGP	RGT	RGP	RGT	RGP
seq_1	**91.89**	**96.92**	68.88	85.46	77.54	89.38
seq_2	**90.72**	**95.68**	67.21	85.13	74.05	86.00
seq_3	**91.21**	**97.13**	76.55	94.80	75.85	90.00
seq_4	**92.72**	**99.66**	74.87	94.33	81.41	95.21
seq_5	**92.68**	**97.99**	74.04	96.78	72.22	88.21
seq_6	**90.91**	**97.54**	52.67	86.37	75.14	91.84
seq_7	**85.99**	95.31	47.98	72.97	79.27	**96.14**
Avg.	**90.87**	**97.17**	66.03	87.98	76.50	90.97

[1] The best of RGT and RGP are colored in **bold red** and **bold blue** respectively.

shows that the number of good tracks of "Image-only KLT" grows as the half-patch width increases from 5 to 30. One example is illustrated in Fig. 4 (c) and (d), where the half-patch widths are set as 10 and 20 respectively and the corresponding number of good tracks increases from 135 to 219. These results indicate that the "Image-only KLT" tracker requires a large patch size to handle fast camera tilting motions. In contrast, our method can achieve good results using all patch size settings. Especially, our method can successfully track 253 features using a small patch with the half-patch width as 5, which is even higher than the best result of "Image-only KLT" (228 features when setting the half-patch width as 35). The number of good tracks of our method grows significantly when the half-patch width increases from 5 to 10. However, the growth becomes gentle when the half-patch width is larger than 10. The reason is that the initial features predicted by our method are close to their true locations, thus a small patch is sufficient for covering most of the target features. The patch cover rates, i.e., the probability that target features are located within the patches centered on the predicted points, are plotted in Fig. 10. Note that the results in Fig. 6 and Table II show that the mean and standard deviation of the initial prediction

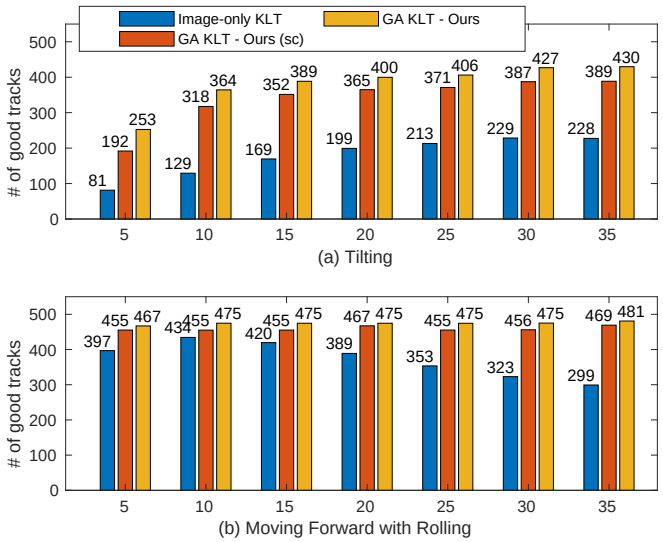


Fig. 9: The number of good tracks w.r.t various half-patch widths over the (a) tilting motions and (b) moving forward with rolling motions of the DeskScene sequence when the interval-sampled rate is set as 2. The x-axis enumerates the half-patch width (i.e., w) ranging from 5 to 35 with a step of 5. The y-axis is the number of good tracks.

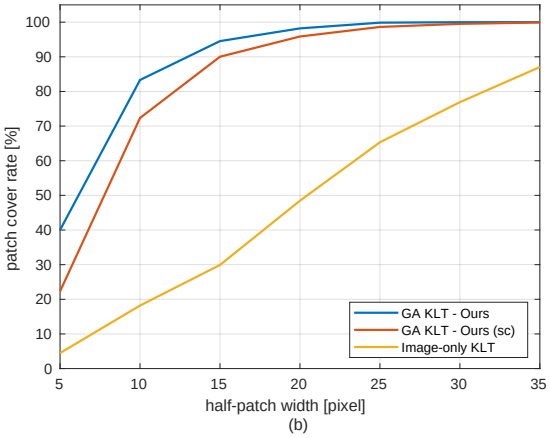


Fig. 10: Patch cover rates w.r.t various half-patch widths when the camera undergoes tilting motions.

errors of "Image-only KLT" are respectively 22.57 and 14.17 pixels, therefore the patch cover rate of "Image-only KLT" grows slowly with the half-patch width increases. In contrast, our method achieves 6.67 and 4.27 pixels of mean and standard deviation of the initial prediction errors, so that the patch cover rate grows quickly from 40.03% to 83.31% when the half-patch width increases from 5 to 10, and it grows gently when the half-patch width above 10. The high patch cover rate helps our method to track more features.

For moving forward with rolling motions, Fig. 9 (b) shows that the number of good tracks of "Image-only KLT" first increases slightly and then decreases as the patch size grows. The results might be because that the "Image-only KLT" tracker implemented based on OpenCV [51] does not consider the patch affine deformation problem, thus large patches

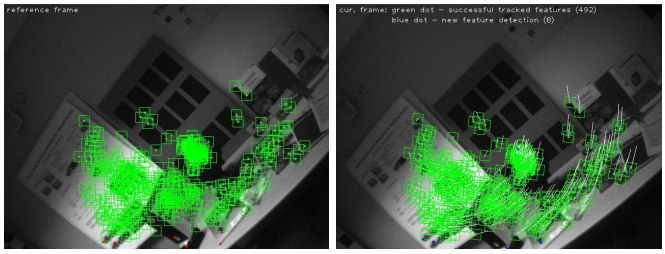


Fig. 11: Feature tracking when the camera undergoes rolling motions. Left: the reference frame; Right: the current frame. The patches are denoted as green squares with a size of 21×21. In the current frame, the patches are deformed to compensate for camera rotations.

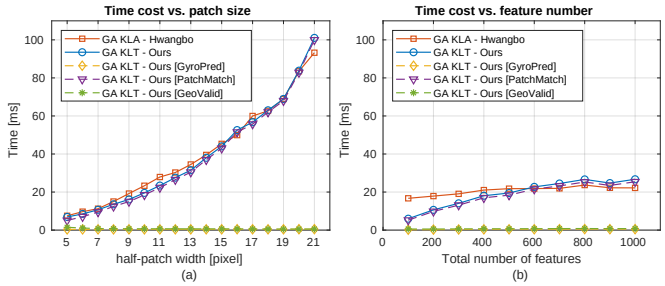


Fig. 12: Time cost comparison between "GA KLT - Hwangbo" [39], [40] and each tracking step of the proposed method as the patch size and the number of features increase (21×21 patch and 500 features when they were fixed in the comparison). (a) Time cost against the half-patch width. (b) Time cost against the number of features.

violate the data conservation assumption of conventional KLT tracker. In contrast, the patch affine deformation matrix for each feature is estimated by our pixel-aware gyro-aided feature prediction algorithm. As a consequence, our method can track almost all the features and remain consistent performance in all patch size settings. An example of feature tracking of our method when the camera undergoes rolling motions is shown in Fig. 11. The patches in the reference frame are depicted in green squares. They are deformed to compensate for camera rolling motions, as shown in the current frame. Note that the performance of the simplified configuration of the proposed method ("GA KLT - Ours (sc)") w.r.t various patch sizes is also shown in Fig. 9. The number of good tracks of "GA KLT - Ours" is higher than that of "GA KLT - Ours (sc)", especially when the camera undergoes tilting motions. The results demonstrate the necessity of considering the pixel coordinates in the feature prediction process.

Fig. 12 shows the time cost comparison between "GA KLT - Hwangbo" [39], [40] and each tracking step of the proposed method as the path size and the number of features increase. The curves with legends of "GA KLT - Hwangbo" and "GA KLT - Ours" are the total time cost of the feature tracking without including the new feature detection and registration. The comparison of time cost against patch size is shown in Fig. 12 (a), where the maximum number of features to be tracked is limited to 500. The results show that the time cost of the proposed method is similar to "GA KLT - Hwangbo", as both the time costs quadratically increase with the half-path width grows. The curves in Fig. 12 (b) show that the time cost consumed by our method is less than "GA KLT - Hwangbo" when the maximum number of features is smaller than 570. When the number of features is larger than the threshold, our method consumes slightly more time than "GA KLT - Hwangbo". Note that "GA KLT - Hwangbo" relies on a GPU device to track features in parallel, while our method performs on pure CPU. These results demonstrate the computational efficiency of the proposed method. The comparison between the time cost of feature prediction, patch matching, and geometric validation shows that the time-consuming of patch matching occupies most of the time cost of the proposed method, while the time costs of feature prediction and geometric validation are negligible. The time cost of the proposed method ranges from 6.92 ms for a half-patch width of 5 pixels to 101.03 ms for a half-patch width of 21 pixels. Note that a large patch size does not significantly improve the tracking performance of our method as analyzed in Fig. 9, it is recommended to select a half-patch size smaller than 10 pixels by balancing the tracking performance and computational complexity.

VII. Conclusions and Future Work

In this paper, the benefits of combining pixel coordinates and gyroscope measurements to solve the feature tracking problem are demonstrated. In particular, a pixel-aware gyro-aided KLT feature tracker is proposed. It introduces a pixel-aware gyro-aided feature prediction algorithm to predict the initial optical flow and the patch affine deformation matrix for each feature, and adopts a geometric validation mechanism based on the score-ratio-based homography/fundamental model selection to filter out the inevitable mistracks. The effectiveness of the proposed method is verified under various scenarios where rapid camera motions and changing illuminations violating the standard Lucas-Kanade assumptions. The experiments on the DeskScene sequence show that by considering the pixel coordinates in the prediction process, the mean and standard deviation of initial prediction errors can be respectively reduced from 8.56 and 5.04 pixels of the similar "GA KLT - Hwangbo" tracker to 6.67 and 4.37 pixels when the camera undergoes tilting motions. For the rate of good tracks, the experimental results on the real-world sequences show that the proposed method achieves 24.84% and 12.12% better than the traditional "Image-only KLT" tracker and the learning-based SuperGlue feature matcher respectively. One drawback of the proposed method is that the computational complexity of the patch-match optimization process is quadratic to the patch size. In the future, we will use the Intel Streaming SIMD Extensions (SSE) to accelerate the computation. In addition, we will apply the proposed method to some challenging tasks like visual-inertial simultaneous localization and mapping, and study the self-calibration problem during the feature tracking.

References

[1] S. Shirmohammadi and A. Ferrero. Camera as the instrument: The rising trend of vision based measurement. *IEEE Instrum. Meas. Mag.*, 17(3):41–47, 2014.

[2] S. Yao, H. Li, S. Pang, B. Zhu, X. Zhang, and S. Fatikow. A review of computer microvision-based precision motion measurement: Principles, characteristics, and applications. *IEEE Trans. Instrum. Meas.*, 70:1–28, 2021.

[3] N. H. Dardas and N. D. Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Trans. Instrum. Meas.*, 60(11):3592–3607, 2011.

[4] S. Zhu and Y. Gao. Noncontact 3-d coordinate measurement of cross-cutting feature points on the surface of a large-scale workpiece based on the machine vision method. *IEEE Trans. Instrum. Meas.*, 59(7):1874–1887, 2009.

[5] K. D. Sharma, A. Chatterjee, and A. Rakshit. A PSO–Lyapunov hybrid stable adaptive fuzzy tracking control approach for vision-based robot navigation. *IEEE Trans. Instrum. Meas.*, 61(7):1908–1914, 2012.

[6] Y. Hu, R. Song, and Y. Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 5704–5712, 2016.

[7] B. Lucas and Takeo T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Int. Joint Conf. Artif. Intell.* Vancouver, British Columbia, 1981.

[8] C. Tomasi and T. Kanade. Detection and tracking of point. *Int J Comput Vis*, 9:137–154, 1991.

[9] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 593–600, 1994.

[10] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255, 2004.

[11] J. Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm. *Intel Corporation, Microprocessor Res. Labs, Tech. Rep.*, 1999.

[12] J. Y. Bouguet. Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm. *Intel Corporation, Microprocessor Res. Labs, Tech. Rep.*, 2001.

[13] J. S. Kim, M. Hwangbo, and T. Kanade. Realtime affine-photometric KLT feature tracker on GPU in CUDA framework. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 886–893, 2009.

[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2564–2571, 2011.

[16] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *Proc. IEEE Eur. Conf. Comput. Vis.*, pages 467–483, 2016.

[17] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. LF-Net: Learning local features from images. In *NeurIPS*, 2018.

[18] D. DeTone, T. Malisiewicz, and A. Rabinovich. Toward geometric deep SLAM. *arXiv preprint arXiv:1707.07410*, 2017.

[19] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 224–236, 2018.

[20] R. Mur-Artal, J. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. Robot.*, 31(5):1147–1163, 2015.

[21] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.*, 33(5):1255–1262, 2017.

[22] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, P. Pajdla, and J. Sivic. Neighbourhood consensus networks. In *NeurIPS*, 2018.

[23] I. Rocco, R. Arandjelović, and J. Sivic. Efficient neighbourhood consensus networks via submanifold sparse convolutions. In *Proc. IEEE Eur. Conf. Comput. Vis.*, pages 605–621, 2020.

[24] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. LoFTR: Detector-free local feature matching with transformers. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 8922–8931, 2021.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[26] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(3):611–625, 2017.

[27] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans. Robot.*, 33(2):249–265, 2016.

[28] T. Qin, J. Pan, S. Cao, and S. Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint arXiv:1901.03638*, 2019.

[29] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *Int. J. Comput. Vision*, 12(1):43–77, 1994.

[30] H. Liu, T. H. Hong, M. Herman, T. Camus, and R. Chellappa. Accuracy vs efficiency trade-offs in optical flow algorithms. *Comput. Vision Image Understand.*, 72(3):271–286, 1998.

[31] T. Senst, V. Eiselein, and T. Sikora. II-LK–A real-time implementation for sparse optical flow. In *Proc. Int. Conf. Image Anal. Recog.*, pages 240–249, 2010.

[32] T. Senst, V. Eiselein, and T. Sikora. Robust local optical flow for feature tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 22(9):1377–1387, 2012.

[33] N. Ramakrishnan, T. Srikanthan, S. K. Lam, and G. R. Tulsulkar. Adaptive window strategy for high-speed and robust klt feature tracker. In *Image and Video Technology*, pages 355–367, 2015.

[34] S. N. Sinha, J. M. Frahm, M. Pollefeys, and Y. Genc. GPU-based video feature tracking and matching. In *EDGE, workshop on edge computing using new commodity architectures*, volume 278, page 4321, 2006.

[35] C. Zach, D. Gallup, and J. M. Frahm. Fast gain-adaptive KLT tracking on the GPU. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1–7, 2008.

[36] H. Fassold, J. Rosner, P. Schallauer, and W. Bailer. Realtime KLT feature point tracking for high definition video. In *Proc. Comput. Graphics, Comput. Vision Math*, 2009.

[37] H. Li, K. Luo, and S. Liu. GyroFlow: Gyroscope-guided unsupervised optical flow learning. *arXiv preprint arXiv:2103.13725*, 2021.

[38] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Proc. IEEE Virtual Reality*, pages 260–267, 1999.

[39] M. Hwangbo, J.S. Kim, and T. Kanade. Inertial-aided KLT feature tracking for a moving camera. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1909–1916, 2009.

[40] M. Hwangbo, J.S. Kim, and T. Kanade. Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation. *Int. J. Robot. Res.*, 30(14):1755–1774, 2011.

[41] Y.G. Ryu, H.C. Roh, and M. J. Chung. Video stabilization for robot eye using IMU-aided feature tracker. In *Proc. IEEE Int. Conf. Control Automat. Syst.*, pages 1875–1878, 2010.

[42] H. C. Ravichandar and A. Dani. Gyro-aided image-based tracking using mutual information optimization and user inputs. In *Proc. IEEE Int. Conf. Syst. Man Cybern.*, pages 858–863, 2014.

[43] G. Yao, M. Williams, and A. Dani. Gyro-aided visual tracking using iterative earth mover's distance. In *Proc. IEEE Int. Conf. Inform. Fusion*, pages 2317–2323, 2016.

[44] B. Poling and G. Lerman. Enhancing feature tracking with gyro regularization. *Image and Vision Computing*, 50:42–58, 2016.

[45] L. Chermak, N. Aouf, and M.A. Richardson. Scale robust IMU-assisted KLT for stereo visual odometry solution. *Robotica*, 35(9):1864–1887, 2017.

[46] L. Chermak, N. Aouf, M. Richardson, and G. Visentin. Real-time smart and standalone vision/IMU navigation sensor. *J. Real-Time Image Pr.*, 16(4):1189–1205, 2019.

[47] O.J. Woodman. An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007.

[48] W. Huang, H. Liu, and W. Wan. An online initialization and self-calibration method for stereo visual-inertial odometry. *IEEE Trans. Robot.*, 36(4):1153–1170, 2020.

[49] W. Huang, W. Wan, and H. Liu. Optimization-based online initialization and calibration of monocular visual-inertial odometry considering spatial-temporal constraints. *Sensors*, 21(8):2673, 2021.

[50] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004, pp: 153-177.

[51] L. Robert. *OpenCV 3 Computer Vision Application Programming Cookbook*. Packt Publishing Ltd, 2017.

[52] P. E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 4938–4947, 2020.

[53] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets. *Int. J. Robot. Res.*, 35(10):1157–1163, 2016.

[54] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 431–437, 2014.