IMRL: An Improved Inertial-Aided KLT Feature Tracker

Meixiang Quan[1,2], Beipeng Mu[1], and Zheng Chai[1]

Abstract— **In recent years, visual simultaneous localization and mapping (SLAM) is widely used in robotic applications. Feature tracking is a fundamental problem in visual SLAM. Kanade-Lucas Tomasi (KLT) feature tracker is the most popular intensity-based feature tracking algorithm for its fast speed and easiness of use. However, it is vulnerable to large optical flow and accumulates error over time. To overcome the drawbacks, we propose a novel inertial-aided multi-reference and multi-level patch based feature tracking approach called IMRL feature tracker. A probabilistic approach is used to estimate the feature's depth, and it is combined with the inertial measurements to provide a good initial feature position, which improves the robustness of our feature tracker to both fast camera rotation and translation. Furthermore, we propose a novel multi-reference and multi-level patch (MRL) based feature alignment method to improve the tracking accuracy. Thorough experiments were carried on open source datasets EuRoC and KITTI. The results show that comparing to the original KLT feature tracker, the proposed IMRL feature tracker achieves better robustness and accuracy with lower computational cost.**

I. Introduction and Motivation

Visual simultaneous localization and mapping (SLAM) is a fundamental and key technology for applications like augmented and virtual reality, robotics, and autonomous driving. Most visual SLAM uses point features to construct map and locate in the map, thus feature tracking is a fundamental component in visual SLAM. In order to ensure the good performance of visual SLAM, accurate and robust feature tracking is required.

Feature tracking algorithms can be divided into two categories: intensity-based [1][2][3] and descriptor-based [4][5][6][7] approaches. Descriptor-based feature trackers achieve better accuracy and robustness, at the cost of high computational burden. In comparison, intensity-based feature trackers have significantly lower computational requirements, thus it is more widely used on real-time low power devices. Based on the above analysis, this paper focuses on the intensity-based feature trackers.

From the works of [1], [3], and [8], Kanade-Lucas Tomasi (KLT) feature tracker is the most representative and widely used intensity-based frame by frame feature tracking algorithm. KLT feature tracking is a nonlinear optimization problem, which requires a good initial guess to ensure it can converge to the global minimum. The original KLT feature tracker set the initial guess to the last feature position, therefore when the feature experiences a large optical flow across image, the initial guess is not in the convergence region for the global minimum, which makes the nonlinear optimization converge to a local minimum or fail to converge. In addition, KLT feature tracker updates the template patches, which accumulates the feature position error, thus the tracked features may drift over time. The shortcomings of KLT feature tracker will greatly impact the accuracy and robustness of visual SLAM. In this paper, to solve these problems, we present a novel inertial-aided multi-reference and multi-level patch based feature tracking approach called IMRL feature tracker, which tracks the feautres accurately, robustly, and efficiently. The IMRL feature tracker is an independent module and can be combined to any visual-inertial SLAM system.

In paper [9], pyramidal KLT feature tracking was presented to increase its robustness to larger optical flow, which is achieved by recursively optimizing the feature position from deepest pyramid level to the original image. Based on this work, paper [10] further improves the robustness of KLT feature tracker to fast camera rotation by providing a better starting point with the aid of gyroscope. In this paper, we aim to provide a good initial feature position for both fast camera rotation and translation, thereby improving the robustness of our feature tracker to all kinds of camera motion. Optical flow induced by camera translation depends on not only the relative transformation, but also the depth of feature. Therefore, with the aid of inertial measurements, we use a probabilistic depth estimation method [11] to estimate the feature's depth. Then, the estimated depth is combined with the transformation computed from inertial measurements to provide a good initial guess for feature position, which increases the possibility that the feature position converges to the global minimum.

In addition, to achieve better feature tracking accuracy with lower computational cost, we propose a novel multi-reference and multi-level patch (MRL) based feature alignment approach. Previous works [1][10] just use one reference frame to construct template patches. Although the template patches extracted from the image pyramid of reference frame contain different resolution information, all the patches are corresponding to same feature position. Therefore, once there is a feature position error in the updated template patches, the feature tracking error will accumulate. In order to eliminate the accumulation of feature position error, we construct template patches from the image pyramids of two reference frames. Furthermore, the feature alignment is a nonlinear optimization problem. More factors the factor graph constains, more accurate the result is. Therefore, to improve the feature tracking accuracy, different from the recursive optimization method of KLT, we use all the template patches at once to optimize the feature position. In this way, when the template patches of a feature are all drift-free, our MRL feature alignment

[1]Momenta, Beijing 100083, China
[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
B. Mu is the corresponding author, email:mubeipeng@momenta.ai

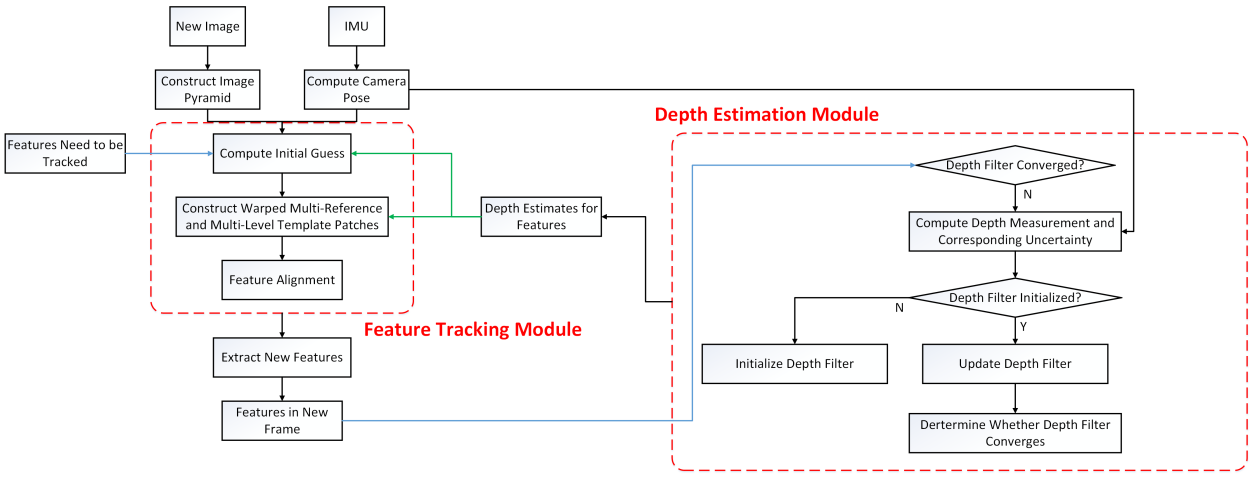


Fig. 1. The flow chart of IMRL feature tracker. When the new image arrives, with the aid of depth estimate (green line) from depth estimation module and the predicted pose from inertial measurements, we align each feature that needs to be tracked (blue line) in feature tracking module. In depth estimation module, for each feature successfully tracked in new frame (blue line), we use its observation and the predicted pose from inertial measurements to initialize or update its depth filter. The features in new frame are the output of our system, and it is used as the input of next feature tracking.

can provide a good feature position. However, once there is a template patch with feature position drift, in order to satisfy both accurate and false measurements simultaneously, our MRL feature alignment will fail to converge. Therefore, our MRL feature alignment approach can always provide the reliable feature tracks. In addition, optimizing the feature position in this manner with a good initial guess also decreases the computational burden.

In experiments, we show that compared to the traditional KLT feature tracker, the proposed IMRL feature tracker achieves better accuracy and robustness with lower computational cost.

II. SYSTEM OVERVIEW

For new frame k, we construct four level image pyramid from intensity image $I_k : \mathbb{R}^2 \to \mathbb{R}$ with down-sampling scale $s = 1.5$. Then with the aid of rigid body transformation $\mathbf{T}_{C_k}^W = \begin{bmatrix} \mathbf{R}_{C_k}^W & \mathbf{p}_{C_k}^W \\ \mathbf{0} & 1 \end{bmatrix} \in \mathrm{SE}(3)$ predicted from the IMU measurements, we aim to track the features in the last frame. Fig. 1 shows an overview of the system. The algorithm consists of two modules, feature tracking module for tracking features, and depth estimation module for estimating the depth of features. Depth estimates of features from depth estimation module are used to compute the good initial estimates of feature positions in feature tracking module, and features that are successfully tracked in feature tracking module provide the new measurements for depth estimation module.

As described in Section III, for each feature u in the last frame $k-1$, the feature tracking module searches for its optimal position $\mathbf{u}_k^*$ in new frame k. In the first step, by using its depth estimate from depth estimation module and the predicted pose from inertial measurements, we compute an initial guess $\breve{\mathbf{u}}_k$ of feature u as Section III-A. Then as described in Section III-B, we construct the multi-reference and multi-level template patches $\mathcal{T}$ for feature u. Each template patch $T_n \in \mathcal{T}$ is constructed by using the corresponding affine matrix $\mathbf{A}_{kn}$ and patch search level ξ_n. Finally, starting with the initial guess $\breve{\mathbf{u}}_k$, we minimize the objective function formulated in Section III-C to find its best estimate $\mathbf{u}_k^*$.

Then in depth estimation module, we use all the available feature measurements $\mathbf{u}_r, ..., \mathbf{u}_k$ and the corresponding transformations $\mathbf{T}_{C_r}^W, ..., \mathbf{T}_{C_k}^W$ to estimate the depth of each feature u, in which the feature is extracted in reference frame r. For each new observation $\mathbf{u}_k (k \neq r)$, if the depth filter of feature u is not converged, we firstly use the new feature match $\mathbf{u}_r$ and $\mathbf{u}_k$ to compute a new depth measurement d_k and the corresponding variance τ_k^2 as Section IV-B. Then the new depth measurement d_k and its variance τ_k^2 are used to initialize or update the depth filter of feature u as described in Section IV-A. We think the depth filter of feature u is converged if and only if the parallax of reference measurement $\mathbf{u}_r$ and current measurement $\mathbf{u}_k$ is greater than a threshold γ_{th} and the current variance of depth estimate σ_k^2 is less than a threshold σ_{th}^2. The depth filter is implemented on the inverse depth coordinates to deal with far scene depths.

III. IMRL FEATURE TRACKING MODULE

A. Computing Initial Feature Position

For each feature u in the last frame $k-1$, by combining its reference feature position $\mathbf{u}_r$, its depth estimate $\hat{d}$ with the relative transformation $\mathbf{T}_{C_r}^{C_k}$ predicted from inertial measurements, we can compute its initial position $\breve{\mathbf{u}}_k$ in the new frame k:

$$\begin{aligned} \mathbf{f}^{C_k} &= \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}} = \mathbf{T}_{C_r}^{C_k} \hat{d} D^{-1}(\pi^{-1}(\mathbf{u}_r)) \\ \breve{\mathbf{u}}_k &= \pi\left(D\left(\frac{\mathbf{f}^{C_k}}{z}\right)\right) \end{aligned} \tag{1}$$

where $\pi(\cdot)$ is the camera projection model and $D(\cdot)$ is the camera distortion model, which is determined by the camera intrinsic parameters and known from the calibration. In addition, according to whether the depth filter of feature u is initialized, we set the depth estimate $\hat{d}$ in different ways. If the depth filter is not initialized, we set $\hat{d}$ as the average scene depth of last frame $k-1$, else we set $\hat{d}$ as the current mean of depth estimate μ_{k-1} that is computed from the depth estimation module.

B. Constructing Multi-Reference and Multi-Level Template Patches

When the depth filter of feature u is initialized, the feature has certain parallax to its initial observation. Therefore, we construct the template patches of feature u from the image pyramids of two reference frames i and j, in which the reference frame i is several frames away from the new frame k to provide the drift-free feature observation, and the reference frame j is selected as the last frame $k-1$ to provide the similar view and texture. In this case, benefiting from the accurate depth estimate, we can get a good initial feature position in the new image. In addition, higher resolution image can achieve better feature tracking accuracy. So the template patches of feature u are just constructed from the pyramid level 0 and 1 of both frames i and j to ensure the computational efficiency. The template patches are denoted as $\mathcal{T}^{ini} = \{T_i^{(0)}, T_i^{(1)}, T_j^{(0)}, T_j^{(1)}\}$, which is constructed by extracting 8×8 image patches from the template images $\mathcal{I}^{ini} = \{I_i^{(0)}, I_i^{(1)}, I_j^{(0)}, I_j^{(1)}\}$ at the corresponding feature positions $\Upsilon^{ini} = \{\mathbf{u}_i, \frac{\mathbf{u}_i}{s}, \mathbf{u}_j, \frac{\mathbf{u}_j}{s}\}$, in which $\{\cdot\}_a^{(b)}$ denotes a certain variable at the pyramid level b of frame a. In the following, I_a denotes $I_a^{(0)}$.

Then when the depth filter of feature u is not initialized, the feature is just extracted in the last frame or the feature has almost no parallax with respect to its first observation. Therefore, the template patches of feature u are only constructed from the image pyramid of reference frame r to align the feature. In this case, since the depth estimate is set to the average scene depth of last frame $k-1$, the computed initial feature position in new frame is not as good as the case of depth filter being initialized. So we construct two template patches $\mathcal{T}_1^{nini} = \{T_r^{(2)}, T_r^{(3)}\}$ and $\mathcal{T}_2^{nini} = \{T_r^{(0)}, T_r^{(1)}\}$, which is constructed in same way as template patches $\mathcal{T}^{ini}$. Template patches $\mathcal{T}_1^{nini}$ are used to preliminarily refine the initial feature position, and the template patches $\mathcal{T}_2^{nini}$ are used to further refine the feature position.

For each feature u, in order to take into account the perspective change, scale change, and variations in lens distortion between the template patch $T_n \in \mathcal{T}$ and the corresponding patch in the new image I_k, we compute a warping matrix $\mathbf{A}_{kn}$. $\mathbf{A}_{kn}$ denotes the horizontal and vertical pixel displacements in new image I_k caused by the unit horizontal and vertical pixel displacements at template feature position $\mathbf{u}_n \in \Upsilon$ in the template image $I_n \in \mathcal{I}$.

Furthermore, for each template patch $T_n \in \mathcal{T}$ of feature u, we use the determinant of matrix $\mathbf{A}_{kn}$ to find a patch search level ξ_n that makes the patch of feature u in image $I_k^{\xi_n}$ most closely matches the scale of T_n, i.e., we choose ξ_n that makes $\frac{\det(\mathbf{A}_{kn})}{(s^{2\xi_n})}$ closest to 1.

Finally, each template patch $T_n \in \mathcal{T}$ is constructed as follows. We firstly use $\mathbf{A}_{kn}^{-1}$ to map all the pixel displacements $\mathcal{W} = \{\mathbf{h} \mid -4 \le h_x \le 3,\ -4 \le h_y \le 3\}$ in image $I_k^{\xi_n}$ to the template image $I_n \in \mathcal{I}$. Then the warped displacements are added to the template feature position $\mathbf{u}_n \in \Upsilon$, where we get the corresponding pixel intensities to construct the template patch T_n. Mathematically, the element of template patch T_n that corresponds to pixel displacement $\mathbf{h}_m \in \mathcal{W}$ in image $I_k^{\xi_n}$ is:

$$T_n(\mathbf{h}_m) = I_n\left(\mathbf{u}_n + \mathbf{A}_{kn}^{-1} s^{\xi_n} \mathbf{h}_m\right) \tag{2}$$

C. Feature Alignment

For the case where the depth filter of feature u is initialized, we optimize the feature position $\mathbf{u}_k$ by minimizing the sum of photometric errors between all the template patches $\mathcal{T}^{ini}$ and the corresponding patches in new frame k:

$$\mathbf{u}_k^* = \arg\min_{\mathbf{u}_k} \sum_{T_n \in \mathcal{T}^{ini}} \sum_{\mathbf{h}_m \in \mathcal{W}} \left\| T_n(\mathbf{h}_m) - I_k^{\xi_n}\left(\frac{\mathbf{u}_k}{s^{\xi_n}} + \mathbf{h}_m\right) \right\|^2 \tag{3}$$

Then for the case where the depth filter of feature u is not initialized, we firstly use template patches $\mathcal{T}_1^{nini}$ to preliminarily optimize the feature position $\mathbf{u}_k$, the optimization is equal to (3) except that $\mathcal{T}^{ini}$ is replaced by $\mathcal{T}_1^{nini}$. Then with the optimized feature position as initial guess, we further optimize the feature position $\mathbf{u}_k$ by using the higher resolution template patches $\mathcal{T}_2^{nini}$. The least squares problem (3) is solved by the inverse compositional algorithm [3].

D. Feature Detection

For new image I_k, the new features are detected by FAST corner detector [12]. Among the candidate features, we firstly remove the features whose position is near to the existing features or image boundary. Then we select out features with high Shi-Tomasi score [8] to enable the high precision feature alignment. Furthermore, we use the bucketing technique to ensure the homogeneous distribution of features.

IV. DEPTH FILTER

A. Bayesian Inference and Parametric Posterior Update

Given the correct depth $\tilde{d}$ and the inlier probability ρ, the probability distribution of depth measurement d_k is a Gaussian + Uniform mixture model, which mixes a good measurement that is normally distributed around the true depth $\tilde{d}$ and a bad measurement that is uniformly distributed in the interval $[d_{min}, d_{max}]$ with the probability ρ and $1-\rho$ respectively:

$$p(d_k|\tilde{d}, \rho) = \rho \mathcal{N}(d_k|\tilde{d}, \tau_k^2) + (1-\rho)\mathcal{U}(d_k|d_{min}, d_{max}) \tag{4}$$

where τ_k^2 is the variance of a good measurement.

Assuming the depth measurements $d_{r+1}, ..., d_k$ are independent, then the estimate for $\tilde{d}$ can be computed in the framework of Bayesian approach:

$$p(\tilde{d}, \rho|d_{r+1}, ..., d_k) \propto p(\tilde{d}, \rho) \prod_{l=r+1}^{k} p(d_l|\tilde{d}, \rho) \tag{5}$$

where $p(\tilde{d}, \rho)$ is a prior on the depth and the inlier ratio. In [13], it is demonstrated that the posterior (5) can be approximated as a parametric Gaussian $\times$ Beta distribution, which is the product of a Gaussian distribution for depth and a Beta distribution for inlier ratio:

$$q(\tilde{d}, \rho|a_k, b_k, \mu_k, \sigma_k^2) = Beta(\rho|a_k, b_k)\mathcal{N}(\tilde{d}|\mu_k, \sigma_k^2) \tag{6}$$

where a_k and b_k control the Beta distribution for ρ, which is all initialized to 10. In addition, μ_k and σ_k^2 represent the mean and variance of depth estimate, which is initialized with the first depth measurement d_{r+1} and its variance τ_{r+1}^2. Then

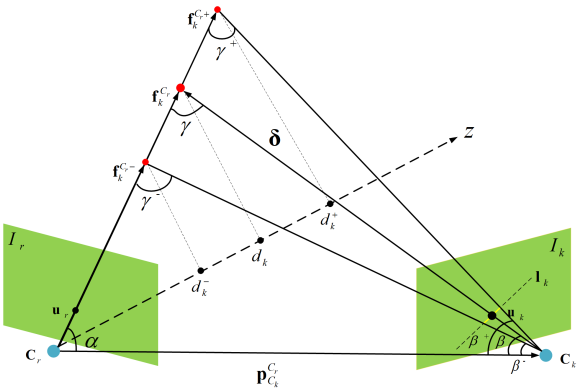


Fig. 2. Calculation of depth measurement uncertainty. Yellow line that passes through $\mathbf{u}_k$ and lies on the epipolar line l_k indicates the one pixel variance of feature measurement $\mathbf{u}_k$. Then the depth measurement uncertainty corresponding to the one pixel variance is $\tau_k^2 = (d_k^+ - d_k^-)^2$.

whenever the new depth measurement d_k comes, the posterior is updated as:

$$p(\tilde{d},\rho|d_{r+1},...,d_k) \propto p(d_k|\tilde{d},\rho) q(\tilde{d},\rho|a_{k-1},b_{k-1},\mu_{k-1},\sigma_{k-1}^2) \quad (7)$$

Although the true posterior (7) is no longer the Gaussian $\times$ Beta distribution, we can still approximate it as a parametric posterior $q(\tilde{d},\rho|a_k,b_k,\mu_k,\sigma_k^2)$ by matching the first and second moments of true and parametric posterior w.r.t $\tilde{d}$ and ρ. The new posterior parameters a_k, b_k, μ_k, σ_k^2 are updated as [13].

B. Depth Measurement Uncertainty

The calculation of depth measurement uncertainty is motivated by the method of [14]. As shown in Fig. 2, $\mathbf{f}_k^{C_r} = d_k \mathbf{n}_r$ is the new 3D coordinate of feature u corresponding to the new depth measurement d_k, in which d_k is obtained by triangulating the feature match $\mathbf{u}_r$ and $\mathbf{u}_k$, and $\mathbf{n}_r$ is the normalized coordinate of $\mathbf{u}_r$. We assume that the variance of depth measurement d_k is corresponding to a fixed one pixel variance of feature measurement $\mathbf{u}_k$ in the new image I_k. Therefore, the variance of depth is computed by back-projecting the one pixel variance along the epipolar line $\mathbf{l}_k$ to 3D space. Given the translation $\mathbf{p}_{C_k}^{C_r}$ and the 3D position $\mathbf{f}_k^{C_r}$, we can compute:

$$\alpha = \arccos\left(\frac{\mathbf{f}_k^{C_r} \cdot \mathbf{p}_{C_k}^{C_r}}{\| \mathbf{f}_k^{C_r} \| \| \mathbf{p}_{C_k}^{C_r} \|}\right)$$
$$\boldsymbol{\delta} = \mathbf{f}_k^{C_r} - \mathbf{p}_{C_k}^{C_r}, \quad \beta = \arccos\left(-\frac{\boldsymbol{\delta} \cdot \mathbf{p}_{C_k}^{C_r}}{\| \boldsymbol{\delta} \| \| \mathbf{p}_{C_k}^{C_r} \|}\right) \quad (8)$$

Each 0.5 pixel on both sides of $\mathbf{u}_k$ all leads to the angle change in the epipolar plane, thus we can compute β^+ and γ^+ angles as:

$$\beta^+ = \beta + \arctan\left(\frac{1}{2f}\right), \quad \gamma^+ = \pi - \alpha - \beta^+ \quad (9)$$

where f is the focal length of camera. Then by the law of sines, we can compute the depth d_k^+ as:

$$\| \mathbf{f}_k^{C_r +} \| = \| \mathbf{p}_{C_k}^{C_r} \| \frac{\sin(\beta^+)}{\sin(\gamma^+)}, \quad d_k^+ = \frac{\| \mathbf{f}_k^{C_r +} \|}{\| \mathbf{n}_r \|} \quad (10)$$

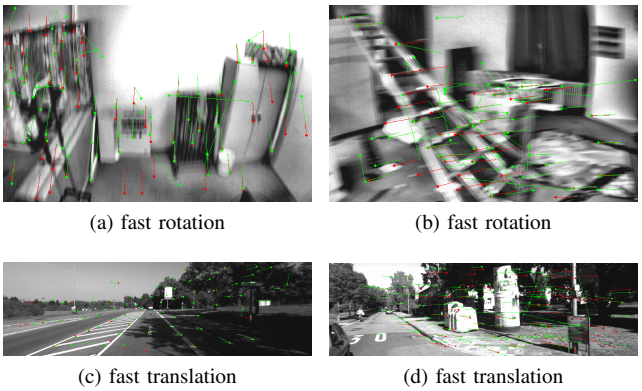
The depth d_k^- can be computed in same way. Therefore, the variance of depth measurement d_k in (4) is :

$$\tau_k^2 = (d_k^+ - d_k^-)^2 \quad (11)$$

V. EXPERIMENTS

We evaluate the proposed IMRL feature tracker by performing experiments on two public datasets: EuRoC [15] and KITTI [16]. Our feature tracker is independent and can be used in any visual-inertial SLAM estimator. In the experiments, we choose VINS-MONO system [17], which is because VINS-MONO uses the classical KLT feature tracker + GFTT [8] feature detection approach. Then we use the inertial-aided KLT + GFTT and the IMRL + FAST to replace the original KLT + GFTT used in VINS-MONO, thereby evaluating the performance of motion compensation and MRL feature alignment of our feature tracker respectively. Since the default patch sizes of KLT and IMRL feature trackers are 21×21 and 8×8 respectively, we use both sizes to evaluate the algorithms. The performance of these feature trackers is quantified by evaluating the estimated trajectories from VINS-MONO with different feature trackers. In the following, we call VINS-MONO with different KLT, inertial-aided KLT, and IMRL feature trackers as KLT-VINS, IKLT-VINS, and IMRL-VINS. Since the feature trackers of IKLT-VINS and IMRL-VINS depend on the poses of frames, when the systems are not initialized, we still use the KLT feature tracker to track features, then after the systems are initialized, we begin to use those new feature trackers. We maintain 150 features in each frame, and for each dataset, we use the same parameters for all the feature trackers to ensure a fair comparison. In addition, we turn off the loop closure function in VINS-MONO system to only consider the effect of feature tracking on performance. The corresponding videos are available at: https://youtu.be/3N2W7-VO368.

A. Robustness Evaluation



(a) fast rotation (b) fast rotation

(c) fast translation (d) fast translation

Fig. 3. Feature tracking results from default KLT and IMRL feature trackers. Red points and lines are feature positions and optical flows tracked by the IMRL feature tracker, and the greend one corresponds to KLT.

Fig. 3 shows the feature tracking results of default KLT and IMRL feature trackers for frames undergoing fast camera rotation and translation. Frames in Fig. 3a and Fig. 3b are from the EuRoC dataset, which experiences the fast camera rotation, and frames in Fig. 3c and Fig. 3d are from the KITTI dataset, which experiences the fast camera translation. From the results, we can know that IMRL can more reliably track

TABLE I

TRANSLATION RMSE ON EUROC DATASET. THE BEST RESULTS ARE GIVEN IN BOLD.

sequence	KLT_8(m)	IKLT_8(m)	IMRL_8(m)	KLT_21(m)	IKLT_21(m)	IMRL_21(m)
V1_03_difficult	0.173	0.167	0.114	0.162	0.132	**0.113**
V2_03_difficult	0.382	0.219	0.181	0.309	0.242	**0.156**
MH_04_difficult	0.463	0.450	**0.381**	0.487	0.472	0.420
MH_05_difficult	0.349	0.285	0.262	0.306	0.268	**0.257**

the features experiencing large optical flow, thereby validating the robustness of our feature tracker to fast camera motion. The robustness improvement is achieved by using the inertial measurements and the depth estimate from depth estimation module to give a good initial feature position.

B. Indoor Experiment on EuRoC Dataset

We use EuRoC dataset to evaluate the performance of our feature tracker in indoor environment. We only use the images from the left camera.

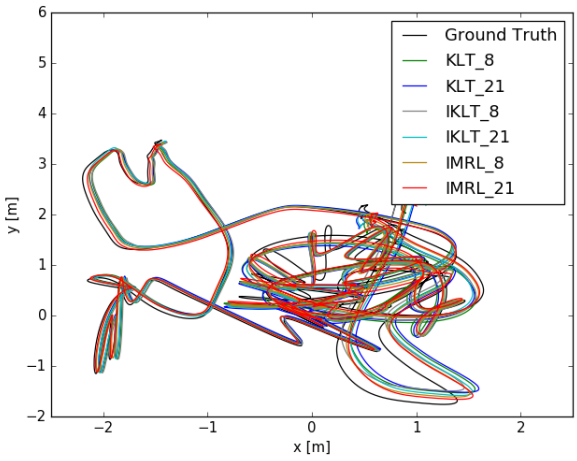


Fig. 4. Estimated trajectories on V1_03_difficult sequence.

For V1_03_difficult sequence, the intuitive comparison of trajectories estimated from KLT-VINS, IKLT-VINS, and IMRL-VINS using the patch size of both 8×8 and 21×21 is shown in Fig. 4. The trajectories are aligned with the ground truth in closed form by using the method of Horn [18]. As evident, our feature tracker gives the trajectory closest to the ground truth.

For quantitative evaluation, Table I shows the translation Root Mean Square Error (RMSE) [19] of all the estimated trajectories for difficult sequences of EuRoC dataset. From the result, we can know that compared with the original KLT-VINS, IKLT-VINS improves the average translation RMSE of 14.9%, and IMRL-VINS improves 28.3%. The performance improvement of feature tracking is achieved by 1) using inertial measurements and depth estimate from depth estimation module to give a good initial feature position; 2) using the proposed MRL feature alignment approach to obtain an accurate feature position.

In addition, we can also find that for V1_03_difficult, V2_03_difficult, and MH_05_difficult sequences, feature trackers with 21×21 patch size achieve better accuracy. However for MH_04_difficult sequence, feature trackers with 8×8 patch size achieve better accuracy. The phenomenon is caused by different degree of occlusion in each scene. If a scene contains more occluding objects, since different motion of occluding objects, the patch information of features is less consistent at different times. Therefore for scenes with less occluding objects, using bigger patch achieves better accuracy by providing more valid measurements for optimizing the feature position. However for scenes with more occluding objects, small patch contains less inconsistent details, thus using small patch can achieve better accuracy.

TABLE II

TIMING STATISTICS ON EUROC DATASET.

Sequence	Module	KLT_8(ms)	IMRL_8(ms)
V	pyramid construction	20.67	6.53
	feature tracking		7.99
	depth estimation	0.0	0.56
	total time	20.67	15.08
MH	pyramid construction	22.34	7.78
	feature tracking		7.94
	depth estimation	0.0	0.44
	total time	22.34	16.16

The experiment was carried out on a laptop with Intel Core i7-8550U 1.80GHz CPU and 8GB RAM. In Table II, we show the timing statistics of KLT and IMRL feature trackers using 8×8 patch size. IMRL also tracks each feature independently, so it can also be accelerated by simultaneously tracking the features in multiple threads. However, since our feature tracker hasn't done any acceleration, we have run the feature trackers in one core to fairly compare the computational cost. Comparing to KLT feature tracker, although the depth estimation module of our feature tracker takes extra average computing time of 0.5ms for each frame, benefiting from the use of the proposed MRL feature alignment approach with a good initial guess, IMRL feature tracker can still decrease the computational cost of 27%.

C. Outdoor Experiment in KITTI Dataset

The outdoor experiment is performed on the odometry benchmark of KITTI dataset [16]. The dataset does not provide the inertial measurements, so we simulated the real IMU measurements from the ground truth poses. In addition, VINS-MONO does a particularly poor job of initialization for the sequences due to its special motion. Therefore in order to make the system works, we use ground truth poses to initialize the system. Table III shows the translation RMSE of estimated trajectories for even number sequences. From the results, we can know that IMRL-VINS achieves the average translation

TABLE III

TRANSLATION RMSE ON KITTI DATASET. THE BEST RESULTS ARE GIVEN IN BOLD.

sequence	KLT_8(m)	IKLT_8(m)	IMRL_8(m)	KLT_21(m)	IKLT_21(m)	IMRL_21(m)
00	11.826	9.795	**9.493**	10.360	11.710	11.607
02	33.179	21.010	**18.730**	43.039	36.148	29.064
04	32.361	18.414	18.414	15.574	10.109	**8.843**
06	11.456	10.425	9.337	9.627	8.194	**7.633**
08	28.381	20.516	**18.844**	32.454	21.850	21.561
10	5.164	2.334	**2.333**	6.520	3.768	3.731

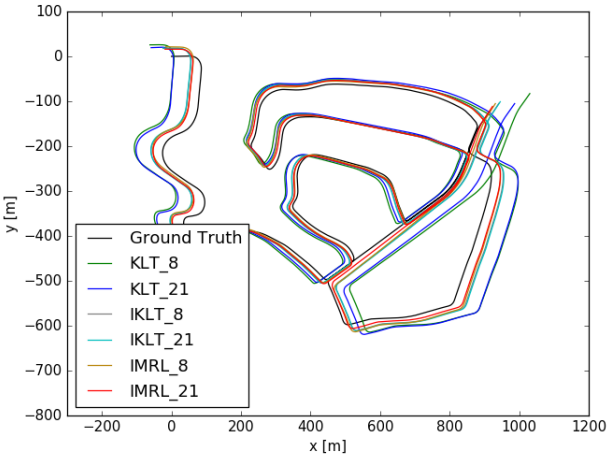


Fig. 5. Estimated trajectories on sequence 02.

RMSE of 13.299m with respect to 19.995m of KLT-VINS and 14.522m of IKLT-VINS, which demonstrates again that IMRL feature tracker outperforms the KLT and IKLT feature trackers. To intuitively demonstrate the superiority of IMRL feature tracker, the comparison of all the estimated trajectories for sequence 02 is shown in Fig. 5. In addition, we also find that IKLT-VINS and IMRL-VINS give very close error in some sequences (e.g. sequence 10), which indicates that given a good initial value, feature alignment accuracy of KLT can achieve the accuracy of MRL feature alignment in some cases.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an improved inertial-aided KLT feature tracking approach called IMRL feature tracker. The robustness of IMRL feature tracker to both fast camera rotation and translation is improved by using inertial measurements and depth estimate from depth estimation module to compute a good initial feature position. In addition, the accuracy and computational efficiency of IMRL feature tracker are improved by proposing a novel MRL feature alignment method. In the experiments, we have validated the superiority of the proposed IMRL feature tracker in terms of accuracy, robustness, and computational cost. As part of future work, we plan to take photometric factors into account, as well as to deal with dynamic objects in environment.

REFERENCES

[1] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 24–28, 1981.

[2] H. Y. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 16(1):63–84, 2000.

[3] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2001.

[4] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[5] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, 2006.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European Conference on Computer Vision*, 2010.

[7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *IEEE Int. Conf. on Computer Vision*, pages 2564–2571, 2011.

[8] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[9] J. Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000.

[10] M. Hwangbo, J. S. Kim, and T. Kanade. Inertial-aided klt feature tracking for a moving camera. In *Int. Conf. on Intelligent Robots and Systems*, 2009.

[11] G. Vogiatzis and C. Hernandez. Video-based, real-time multi view stereo. *Image and Vision Computing*, 29(7), 2011.

[12] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Computer Vision - ECCV*, pages 430–443, 2006.

[13] G. Vogiatzis and C. Hernandez. Video-based, real-time multi view stereo. *Image and Vision Computing*, 29(7), 2011.

[14] M. Pizzoli, C. Forster, and D.Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2014.

[15] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 28(5):595–599, 2009.

[17] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *arXiv*, abs/1708.03852, 2017.

[18] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[19] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 573–580, 2012.