2018 24th International Conference on Pattern Recognition (ICPR)
Beijing, China, August 20-24, 2018

Online Temporal Calibration of Camera and IMU using Nonlinear Optimization

Jinxu Liu, Wei Gao*, Zhanyi Hu
{jinxu.liu, wgao, huzy}@nlpr.ia.ac.cn
NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
University of Chinese Academy of Sciences, Beijing 100049, China

***Abstract*—In this paper, we aim to calibrate the time delay of timestamps of cameras and IMU measurements provided by Android smart phones or other low-cost devices whose camera and IMU are not temporally aligned. The time delay is estimated online in an iterative way through nonlinear optimization in sliding windows. We add new terms that are relative to time delay to the pre-integration results of IMU measurements instead of feature observations in order to improve the precision of temporal calibration. The experimental results indicate that our calibration result is closer to the real value compared with the state-of-the-art system and that our method appears to converge faster. By using our temporal calibration, the visual inertial odometry algorithm is less likely to suffer from fast turning or sudden stop.** [1]

I. Introduction

Recent years have seen many great visual inertial odometry algorithms where camera and IMU data are combined to use because they are complementary. MSCKF [1] utilizes Extended Kalman Filter (EKF) that propagates IMU measurements to update state variables and regards features from pictures as observations. OKVIS [2] makes a tightly coupled fusion of camera and IMU data by nonlinear optimization. From then on, more visual inertial odometry or SLAM algorithms resorting to nonlinear optimization appear, such as VINS [3], which adds well-designed initialization and loop closure modules to OKVIS, and the visual-inertial version of ORB-SLAM [4], which is adapted from vision only monocular ORB-SLAM [5] and also jointly optimizes all sensor states in a local window. They are widely used for localization of micro air vehicles (MAV), as well as AR development and location based services on smart phones.

Every of the algorithms expects the timestamps of cameras and IMU measurements to be precise and temporally aligned. But on many low-cost systems, such as Android smart phones, the timestamps provided by cameras and IMUs are in different time references. They differ from each other by a time delay, typically the delay between the time of first exposure of the camera and the time of entering the callback function of the first picture. Such time delay is a fixed value or drifts very slowly with respect to time during a program run, but differs every time the program runs. Thus the time delay, gotten from an off-line temporal calibration using off-the-shelf tools such as Kalibr [6] before program runs, is not the one needed in the visual inertial odometry program. Hence online temporal calibration is required to estimate the time delay between the timestamps of pictures and those of IMU measurements when program runs.

Different approaches on temporal calibration of camera-IMU systems have been proposed. Some of them first solve camera poses using visual information only and then align them with gyroscope data acquired from IMU [7], [8]. In [7], the angular velocity of camera is derived from the image based motion estimation. Then temporal alignment is made by computing cross-correlation ratio or phase congruency between camera angular velocities and angular velocities measured by IMU. [8] proposes a TD-ICP method where camera and IMU orientations are regarded as points in three dimensional space, and the time delay and spatial transformation between the two sets are computed iteratively like Iterative Closest Points (ICP). Both of the above methods require computing camera poses before temporal alignment, where the precision of every camera pose is regarded as equal, thus an inaccurate estimation of one camera pose may affect the alignment seriously. Another approach [9] models the IMU-camera calibration problem in a nonlinear optimization framework but is not real-time capable. [10] is an online calibration method that includes the time delay t_d in the IMU state vector of Extended Kalman Filter (EKF) which cannot be directly adapted to be used in visual inertial odometry algorithms that use nonlinear optimization such as [2] and [3].

The latest version of the open source code of VINS [3] by the time this paper is written includes online calibration module using nonlinear optimization. The velocity at which every feature point moves on image is calculated by dividing the displacement of the feature point between two successive frames by the time interval between them. Then a correction term, which is the product of feature velocity and time delay, is added to feature observations. In this way, the time delay to be estimated is included as a parameter in nonlinear optimization. However, visual measurements are not only at a much lower frequency than IMU measurements, but also suffer from image blur and matching errors. The velocity calculated may not be the real velocity at which the feature point moves at the instant the picture is taken because the time interval between frames is big. Therefore, our approach is based on [3] but instead add new terms that are relative to time delay to the pre-integration

[1]This work was supported in part by the National Key R&D Program of China (2016YFB0502002), and in part by the Natural Science Foundation of China (61472419, 61333015, 61772444). *Corresponding Author: Wei Gao{wgao@nlpr.ia.ac.cn}.

results of IMU measurements. The calculation is a bit more complicated but our approach appears to converge faster and gets a more precise estimation of the time delay than [3].

II. Our Approach

A. The Definition of Time Delay

Taking our Android smart phone for example, the timestamps from camera and IMU measurements are on different time basis. The timestamps from IMU are the time since system boot, that is, they are on the same time base as the time we get from Java Virtual Machine (JVM). The timestamps of images are precise but on a totally unknown time base. To roughly align the two sets of timestamps, we substitute the time of first exposure of camera with the time of executing the callback function of the first image, because the time of callback function execution is the time from JVM, so that it is on the same time base as the time base of IMU measurements. We define $t^I_{e,0}$ and $t^I_{e,k}$ as the time at the first and the k^{th} frame exposure respectively, on IMU time base. In fact, $t^I_{e,k}$ is what we need exactly, and it can be written as:

$$t^I_{e,k} = t^I_{e,0} + (t^I_{e,k} - t^I_{e,0}) \tag{1}$$

which is an identical equation. Let $t^I_{f,0}$ denote the time at the execution of the callback function of the first image, and let $t^C_{e,0}$ and $t^C_{e,k}$ denote the time at the first and the k^{th} frame exposure respectively, on camera's own time base. Given the fact that neither $t^I_{e,k}$ nor $t^I_{e,0}$ is known, we substitute $t^I_{e,0}$ with $t^I_{f,0}$ to compute $t^I_{e,k}$. Because $t^I_{e,0} - t^I_{e,k}$ and $t^C_{e,0} - t^C_{e,k}$ are equal, the timestamp of the k^{th} frame that we compute is:

$$\hat{t}^I_{e,k} = t^I_{f,0} + (t^C_{e,k} - t^C_{e,0}) \tag{2}$$

and the time delay t_d is:

$$t_d = \hat{t}^I_{e,k} - t^I_{e,k} = t^I_{f,0} - t^I_{e,0} \tag{3}$$

which is the time interval between the exposure moment of the first frame and the moment of executing the callback function of the first image.

B. IMU Model with Time Delay

Both OKVIS [2] and VINS [3] jointly optimize a cost function that contains reprojection error terms, IMU error terms, and the marginalization error term which represents prior information. The reader may refer to [3] for details. Among them only the IMU error terms will be relevant to the time delay. The IMU error terms can be expressed as:

$$C_{IMU}(\mathbf{x}) = \sum_{k=1}^{K-1} \mathbf{e}_s^{k^T}(\mathbf{x}_k, \mathbf{x}_{k+1}) {\mathbf{P}_s^{k,k+1}}^{-1} \mathbf{e}_s^k(\mathbf{x}_k, \mathbf{x}_{k+1}) \tag{4}$$

where k denotes the camera frame index, $\mathbf{x}_k$ is the IMU state vector of the k^{th} camera frame, and $\mathbf{x}$ is composed of $\mathbf{x}_1 \dots \mathbf{x}_K$. The covariance matrix $\mathbf{P}_s^{k,k+1}$ is computed through IMU pre-integration [3]. The residual vector $\mathbf{e}_s^k$ is relevant to the state vectors of the k^{th} and $k+1^{th}$ frame as well as the IMU pre-integration between them:

$$\begin{aligned}
&\mathbf{e}_s^k(\mathbf{x}_k, \mathbf{x}_{k+1}) \\
&= \begin{bmatrix}
\mathbf{R}_w^{b_k}(\mathbf{p}^w_{b_{k+1}} - \mathbf{p}^w_{b_k} + \frac{1}{2}\mathbf{g}^w \Delta t_k^2 - \mathbf{v}^w_{b_k} \Delta t_k) - \hat{\alpha}^{b_k}_{b_{k+1}} \\
\mathbf{R}_w^{b_k}(\mathbf{v}^w_{b_{k+1}} + \mathbf{g}^w \Delta t_k - \mathbf{v}^w_{b_k}) - \hat{\beta}^{b_k}_{b_{k+1}} \\
2\left[\mathbf{q}_w^{b_k} \otimes \mathbf{q}^w_{b_{k+1}} \otimes (\hat{\gamma}^{b_k}_{b_{k+1}})^{-1}\right]_{xyz} \\
\mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k} \\
\mathbf{b}_{wb_{k+1}} - \mathbf{b}_{wb_k}
\end{bmatrix}
\end{aligned} \tag{5}$$

where $\mathbf{p}^w_{b_k}$ and $\mathbf{v}^w_{b_k}$ are the camera position and velocity at the moment of the exposure of the k^{th} frame in the world frame respectively, while $\mathbf{b}_{ab_k}$ and $\mathbf{b}_{wb_k}$ are the biases of accelerometer and gyroscope respectively, in IMU body frame. $\mathbf{R}_w^{b_k}$ is the rotation matrix from world frame to IMU body frame at the time of the exposure of k^{th} frame, and $\mathbf{q}_w^{b_k}$ is its quaternion form. $\hat{\alpha}^{b_k}_{b_{k+1}}$, $\hat{\beta}^{b_k}_{b_{k+1}}$ and $\hat{\gamma}^{b_k}_{b_{k+1}}$ are the results of IMU pre-integration between the time t_k and t_{k+1}, which denote the time at the exposure of k^{th} and $k+1^{th}$ frame respectively.

If there is no time delay between image timestamps and IMU timestamps, $\hat{\alpha}^{b_k}_{b_{k+1}}$, $\hat{\beta}^{b_k}_{b_{k+1}}$ and $\hat{\gamma}^{b_k}_{b_{k+1}}$ are directly computed through pre-integration, as is what [3] does. Given that there is a time delay, let's say ΔT, between camera and IMU, what we compute through pre-integration are in fact the variations of displacement, velocity and rotation between time $t_k + \Delta T$ and $t_{k+1} + \Delta T$. We define the time $t_k + \Delta T$ and $t_{k+1} + \Delta T$ as t_b and t_e respectively, which represent the time at the beginning and the ending of the IMU pre-integration we actually performed. Let $\hat{\alpha}^b_e$, $\hat{\beta}^b_e$ and $\hat{\gamma}^b_e$ denote the variations of displacement, velocity and rotation between time t_b and t_e, and those are directly computed through IMU pre-integration. We are going to represent $\hat{\alpha}^{b_k}_{b_{k+1}}$, $\hat{\beta}^{b_k}_{b_{k+1}}$ and $\hat{\gamma}^{b_k}_{b_{k+1}}$ using $\hat{\alpha}^b_e$, $\hat{\beta}^b_e$, $\hat{\gamma}^b_e$ and ΔT, along with other IMU measurements.

Since $\hat{\gamma}^{b_k}_{b_{k+1}}$ is the rotation quaternion of IMU body frame from the time t_{k+1} to time t_k, it can be decomposed as the rotation from time t_{k+1} to t_e, the rotation from time t_e to t_b and the rotation from time t_b to t_k. The rotation quaternion from t_e to t_{k+1} is computed as:

$$\gamma_e^{b_{k+1}} = \begin{bmatrix} 1 \\ \frac{1}{2}(\hat{\boldsymbol{\omega}}_e - \mathbf{b}_{wb_k})\Delta T \end{bmatrix} \tag{6}$$

where $\hat{\boldsymbol{\omega}}_e$ is the angular velocity measured by gyroscope at time t_e, and $\mathbf{b}_{wb_k}$ is the gyroscope bias. Similarly, the rotation quaternion from t_b to t_k is computed as:

$$\gamma_b^{b_k} = \begin{bmatrix} 1 \\ \frac{1}{2}(\hat{\boldsymbol{\omega}}_b - \mathbf{b}_{wb_k})\Delta T \end{bmatrix} \tag{7}$$

where $\hat{\boldsymbol{\omega}}_b$ is the angular velocity measured by gyroscope at time t_b. Finally, the rotation quaternion from t_{k+1} to t_k can be expressed as:

$$\hat{\gamma}^{b_k}_{b_{k+1}} = \gamma_b^{b_k} \otimes \hat{\gamma}^b_e \otimes (\gamma_e^{b_{k+1}})^{-1} \tag{8}$$

The relation among time t_k, t_{k+1}, t_b, t_e and ΔT is shown in Figure 1.

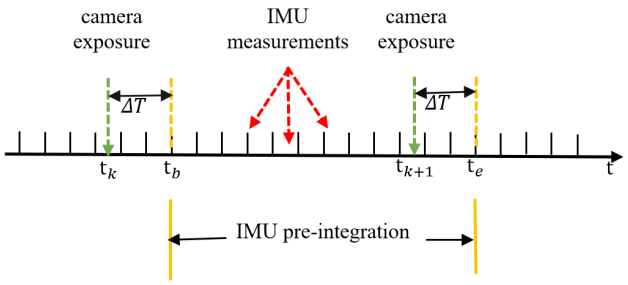


Fig. 1. t_k,t_{k+1},t_b,t_e and ΔT are on the time axis. t_k and t_{k+1} are at the moment of camera exposure. t_b and t_e are the beginning and the ending of IMU pre-integration respectively.

As shown in Figure 1, there are many IMU measurements between time t_b and t_e. According to the IMU pre-integration process in [3], the velocity change $\hat{\beta}^{b_k}_{b_{k+1}}$ is expressed in the discrete form:

$$\hat{\beta}^{b_k}_{b_{k+1}} = \sum_{t_i=t_k}^{t_{k+1}} \mathbf{R}(\hat{\gamma}^{b_k}_i)\mathbf{a}_i\delta t_i \tag{9}$$

where $\mathbf{a}_i$ represents $\hat{\mathbf{a}}_i - \mathbf{b}_{ab_k}$. t_i represents the moment of the i^{th} IMU measurement between t_k and t_{k+1}. $\hat{\gamma}^{b_k}_i$ is the rotation quaternion from the IMU body frame at the time of the i^{th} IMU measurement to the body frame at time t_k, which is computed beforehand only using the gyroscope measurements and the time intervals among them. $\mathbf{R}(\hat{\gamma}^{b_k}_i)$ is the rotation matrix form of $\hat{\gamma}^{b_k}_i$. $\hat{\mathbf{a}}_i$ is the i^{th} accelerometer measurement and $\mathbf{b}_{ab_k}$ is the accelerometer bias. δt_i is the time interval between two IMU measurements i and $i+1$. As mentioned before, what we compute through IMU pre-integration is in fact $\hat{\beta}^b_e$, which is:

$$\hat{\beta}^b_e = \sum_{t_i=t_b}^{t_e} \mathbf{R}(\hat{\gamma}^b_i)\mathbf{a}_i\delta t_i \tag{10}$$

From (9) and (10), it is derived that

$$\hat{\beta}^{b_k}_{b_{k+1}} = \mathbf{R}(\gamma^{b_k}_b)\hat{\beta}^b_e + \sum_{t_i=t_k}^{t_b} \mathbf{R}(\hat{\gamma}^{b_k}_i)\mathbf{a}_i\delta t_i - \sum_{t_j=t_{k+1}}^{t_e} \mathbf{R}(\hat{\gamma}^{b_k}_j)\mathbf{a}_j\delta t_j \tag{11}$$

Assuming ΔT is small, $\hat{\mathbf{a}}_i$ from t_k to t_b is approximated by $\hat{\mathbf{a}}_b$ and $\hat{\mathbf{a}}_j$ from t_{k+1} to t_e is approximated by $\hat{\mathbf{a}}_e$. Meanwhile, we substitute $\mathbf{R}(\hat{\gamma}^{b_k}_i)$ with $\mathbf{R}(\gamma^{b_k}_b)$ and substitute $\mathbf{R}(\hat{\gamma}^{b_k}_j)$ with $\mathbf{R}(\gamma^{b_k}_b)\mathbf{R}(\hat{\gamma}^b_e)$. $\hat{\gamma}^b_e$ has been computed through IMU pre-integration and $\gamma^{b_k}_b$ is expressed in (7). Thus $\hat{\beta}^{b_k}_{b_{k+1}}$ can be approximated as:

$$\hat{\beta}^{b_k}_{b_{k+1}} = \mathbf{R}(\gamma^{b_k}_b)(\hat{\beta}^b_e + \mathbf{a}_b\Delta T - \mathbf{R}(\hat{\gamma}^b_e)\mathbf{a}_e\Delta T) \tag{12}$$

where $\mathbf{a}_b$ represents $\hat{\mathbf{a}}_b - \mathbf{b}_{ab_k}$ and $\mathbf{a}_e$ represents $\hat{\mathbf{a}}_e - \mathbf{b}_{ab_k}$. Similarly, $\hat{\alpha}^{b_k}_{b_{k+1}}$ can also be expressed as the formula:

$$\hat{\alpha}^{b_k}_{b_{k+1}} = \sum_{t_i=t_k}^{t_{k+1}} \delta t_i \sum_{t_j=t_k}^{t_i} \mathbf{R}(\hat{\gamma}^{b_k}_j)\mathbf{a}_j\delta t_j + \frac{1}{2}\sum_{t_i=t_k}^{t_{k+1}} \mathbf{R}(\hat{\gamma}^{b_k}_i)\mathbf{a}_i\delta {t_i}^2 \tag{13}$$

while what we actually compute through IMU pre-integration is

$$\hat{\alpha}^b_e = \sum_{t_i=t_b}^{t_e} \delta t_i \sum_{t_j=t_b}^{t_i} \mathbf{R}(\hat{\gamma}^b_j)\mathbf{a}_j\delta t_j + \frac{1}{2}\sum_{t_i=t_b}^{t_e} \mathbf{R}(\hat{\gamma}^b_i)\mathbf{a}_i\delta {t_i}^2 \tag{14}$$

Comparing (13) and (14) we can derive that

$$\begin{aligned}\hat{\alpha}^{b_k}_{b_{k+1}} =& \mathbf{R}(\gamma^{b_k}_b)\hat{\alpha}^b_e + \sum_{t_i=t_k}^{t_{k+1}} \delta t_i \sum_{t_j=t_k}^{t_b} \mathbf{R}(\hat{\gamma}^{b_k}_j)\mathbf{a}_j\delta t_j \\ &- \sum_{t_i=t_{k+1}}^{t_e} \delta t_i \sum_{t_j=t_b}^{t_e} \mathbf{R}(\hat{\gamma}^{b_k}_j)\mathbf{a}_j\delta t_j\end{aligned} \tag{15}$$

By approximating $\mathbf{R}(\hat{\gamma}^{b_k}_j)$ from t_k to t_b by $\mathbf{R}(\gamma^{b_k}_b)$ and approximating $\mathbf{a}_j$ from t_k to t_b by $\mathbf{a}_b$, we can reduce (15) to:

$$\hat{\alpha}^{b_k}_{b_{k+1}} = \mathbf{R}(\gamma^{b_k}_b)(\hat{\alpha}^e_b + \mathbf{a}_b(t_e - t_b)\Delta T - \hat{\beta}^b_e\Delta T) \tag{16}$$

Substitute (8), (12) and (16) into (5), and then we get the IMU residual vector $\mathbf{e}^k_s$ that contains the time delay ΔT between the camera and IMU time base as a parameter. For deriving the Jacobian matrix which is composed of the partial derivatives of $\mathbf{e}^k_s$ with respect to all the parameters the reader may refer to [11] for details. Although in this paper (8), (12) and (16) are derived assuming $\Delta T \geq 0$, they still hold when $\Delta T < 0$. There is some approximation in the Jacobian matrix, but the approximation error approaches 0 as ΔT approaches 0. However, in the latest open source code of [3], the approximation error caused by using the average velocity between two successive frames does not approach to 0 however small ΔT is. This is one of the reasons why our approach converges faster than the latest open source code of [3], and another reason is that IMU measurements come at a much higher frequency than images.

C. Iterative Alignment

We adopt the iterative alignment strategy adopted by the latest open source code of [3]. The time interval ΔT is estimated every time nonlinear optimization is performed in the sliding window. When ΔT is computed, we subtract ΔT from the succeeding image timestamps, to make the image timestamps more close to the real camera exposure moments. In this way, the time delay between camera time base and IMU time base becomes smaller and smaller as nonlinear optimization is performed once and once again. As the time delay decreases the approximation error also decreases. At last the time delay converges to zero while ΔT computed every time in the sliding window remains almost zero. Figure 2 explains the change on image timestamps of succeeding frames after ΔT is estimated.

III. Experimental Results

We experimentally evaluated our work on the Android smart phone Huawei Honor Note8, which provides an 2.5GHz $\times$ 4 + 1.8GHz $\times$ 4 CPU and 4GB memory. First we make quantitative comparison with the temporal calibration work

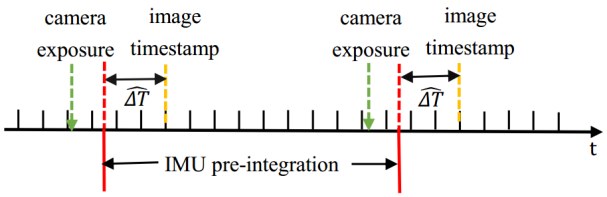


Fig. 2. $\widehat{\Delta T}$ is the estimated time interval between image timestamp (the yellow bar) and the moment of camera exposure (the green arrow). After subtracting $\widehat{\Delta T}$ the updated image timestamp (the red bar) is more close to the camera exposure moment. For succeeding frames IMU pre-integration is performed using IMU measurements between the updated camera timestamps.

adopted by the latest open source code of [3] on the EuRoc MAV Dataset [12], and then test our work on the data acquired by our smart phone in our lab. Ceres Solver [13] is used for nonlinear optimization. Although the derivation of our approach is a bit more complicated, the experiments prove that it is not computationally more expensive.

A. Experiments on Datasets

On the Euroc MAV Dataset where the camera and IMU data are temporally aligned, we manually shift the camera timestamps for 60ms, 30ms, 0ms and -30ms in turn, and compare the estimated time delay by our method and by the latest open source code of VINS [3] against the shift we add on camera timestamps. The experiments where camera timestamps are shifted for 0ms are aimed to test whether the online temporal calibration have negative effect on the results when the camera and IMU has been temporally aligned actually. Furthermore, we compare the camera trajectory computed by the two methods against both the ground truth data and the computed trajectory when camera timestamps are not shifted and the time delay is not estimated. The latter comparison is based on the assumption that if the time delay is well estimated (and meanwhile the camera timestamps are accurately restored) by online temporal calibration, the camera trajectory should be close to the original trajectory when no time shift is added on camera timestamps.

Figure 3 displays the temporal calibration error. Because of space limitations, only the results on five datasets are shown, which include results on two easy datasets, one medium dataset and two difficult datasets. Generally speaking, our method outperforms the latest open source code of VINS in terms of temporal calibration error on the datasets and neither of the calibration methods has bad effects when the time delay is actually zero.

Table I shows the Mean Square Error (MSE) of the trajectory compared with groundtruth. The results prove that both our method and the method adopted by the latest open source code of VINS can dramatically reduce the MSE compared with ground truth. However, a noteworthy phenomenon appears that no time delay does not guarantee the least MSE even when no temporal calibration is performed. For example, the MSE of MH_01_easy when $t_d = -30ms$ is smaller than that when

TABLE III
COMPARISON OF TIME DELAY AT 25 SECONDS AFTER INITIALIZATION

	Calibrated Time Delay (ms)							
	seq1	seq2	seq3	seq4	seq5	seq6	seq7	seq8
Ours	32.9	23.6	82.4	56.6	52.0	59.1	35.7	37.1
VINS	10.3	11.4	49.5	10.2	9.1	18.7	10.2	26.3
Align	42	30	45	42	45	48	42	39

The *Ours* and *VINS* rows show the calibrated time delay by our method and the latest open source code of VINS respectively. The *Align* row shows the calibrated time delay by the third method.

$t_d = 0ms$. The similar thing happens on MH_05_difficult. Thus by comparing the MSE of the two methods in Table I we can't tell which trajectory is better. Thus we go on to compare against the trajectory when no time shift is added on the timestamps and no temporal calibration is performed, whose results are shown in table II. In general, the MSE of our trajectory is smaller. In other words, our trajectory is more close to the trajectory of temporally aligned data.

B. Experiments on Our Data

We collected 8 image sequences in our lab using an Android smart phone held by a pedestrian. The moving pattern of pedestrians is different from MAVs in that pedestrians like to walk in a straight line with fast turnings and sudden stops occasionally. And both image data and IMU measurements are more noisy. Our calibration method appears to converge faster than the latest open source code of VINS and it performs well on fast turnings or sudden stops.

1) Comparison on convergence speed: We compare the online calibration results at 25 seconds after initialization between our method and VINS. Since we don't know the true time delay, we perform a third online calibration method to see which result it *approves*. In the third method, like [7] and [8], we solve camera orientations using image data only and then find the time delay that make IMU measurements most consistent with the camera orientations. To avoid incorrect results, the third method is only performed when a list of strict thresholds are satisfied. The calibrated time delays are shown in Table III. Our method appears to converge faster than VINS.

2) Results on fast turning and sudden stop: Among the 8 image sequences, Sequence 4 suffers from a sudden stop and Sequence 5 suffers from a fast turning. In Sequence 4, we walked around a rectangular area whose size is about $11m \times 21m$ and the points in Figure 4 A, B_1, B_2 and B_3 should be approximately coincide. The trajectories computed from VINS and from the method without online temporal calibration both suffer from a shrunken scale after the sudden stop. But our trajectory still maintain a right scale after the sudden stop to avoid a pedestrian. In sequence 5, the camera turned suddenly not long after initialization. The camera poses from the method without temporal calibration drifted away and finally failed. The poses from VINS also drifted away because of its incorrectly estimated time delay at the turning moment.

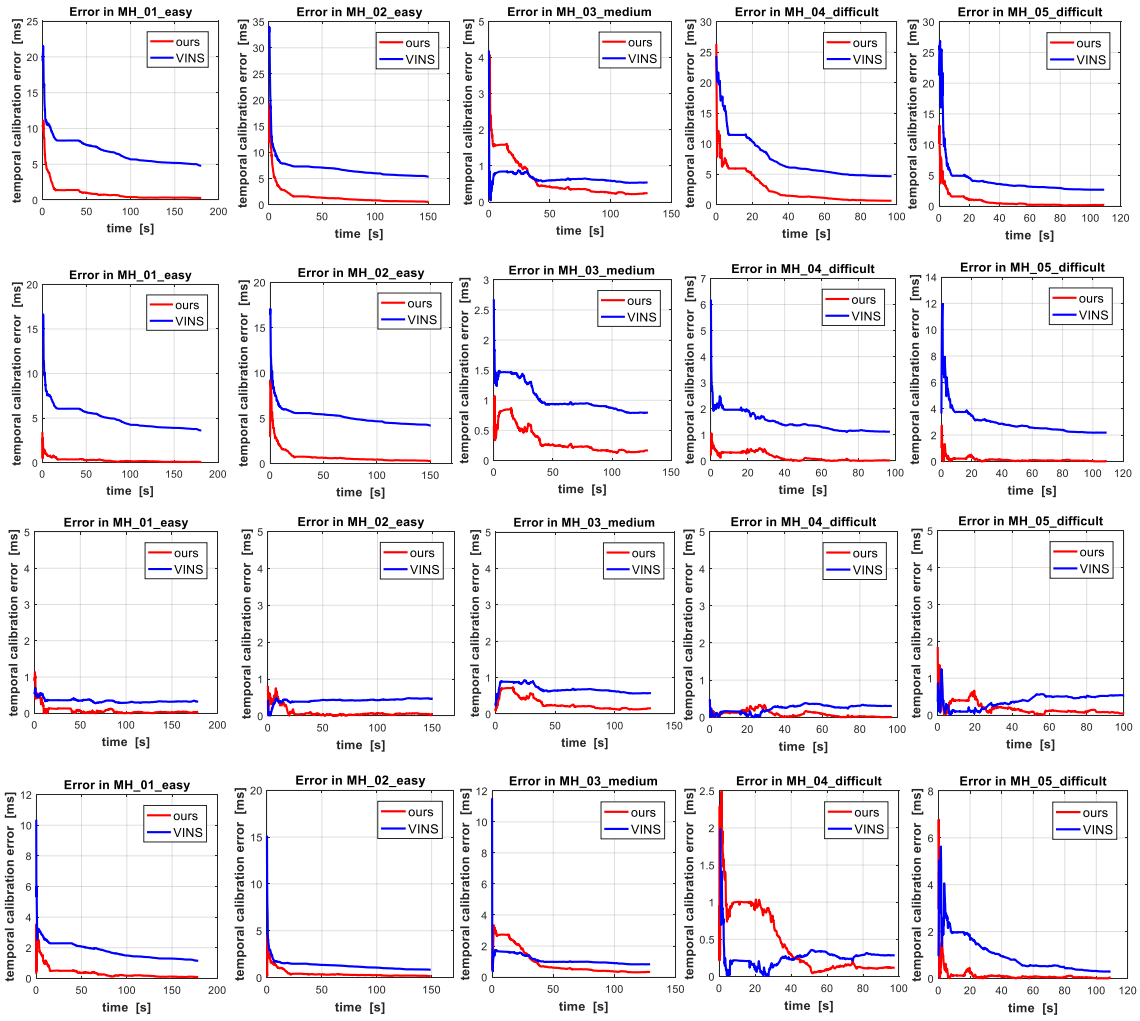
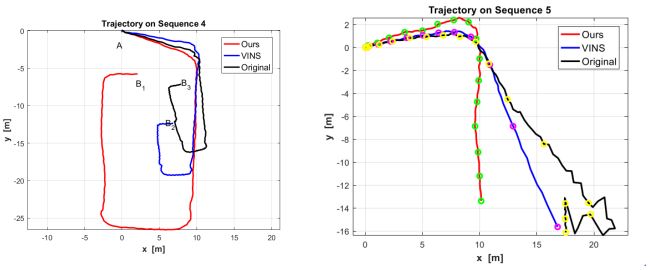


Fig. 3. Calibration error along time. The calibration results when camera timestamps are shifted for 60ms, 30ms, 0ms and -30ms corresponds to the first row to the last row of the figures.

Our trajectory did not drift away and the camera moved at an approximately constant speed as it actually did. The results are shown in Figure 4. The dots in Figure 4 are selected camera positions at a constant time interval, which shows that the camera moves at a constant speed in our trajectory but they move faster after the turning in the other two trajectories.

IV. Conclusion

This paper demonstrates an online temporal calibration approach which can be applied on visual inertial odometry algorithms based on nonlinear optimization and sliding windows. Comparing with the nonlinear optimization approach which adds correction terms on the coordinates of feature points, our approach performs better on EuRoc MAV Dataset in terms of calibrated time delay and the resulting trajectory.



(a) (b)

Fig. 4. (a) and (b) are respectively the trajectories of Sequence 4 and Sequence 5. The dots on the trajectory in (b) are selected poses at a constant time interval.

TABLE I
MEAN SQUARE ERROR OF TRAJECTORY COMPARED WITH GROUND TRUTH

	Trajectory MSE(m)											
	d_t =60ms			d_t =30ms			d_t =0ms			d_t =-30ms		
	Ours	VINS	Original	Ours	VINS	Original	Ours	VINS	Original	Ours	VINS	Original
MH_01_easy	**0.272**	0.330	Fail	**0.255**	0.306	0.436	0.266	**0.253**	0.262	0.265	**0.241**	0.228
MH_02_easy	0.284	**0.252**	0.720	0.286	**0.263**	0.387	0.289	**0.287**	0.288	**0.289**	0.295	0.501
MH_03_medium	0.234	**0.227**	Fail	0.229	**0.227**	0.582	0.226	**0.226**	0.226	**0.225**	0.229	0.452
MH_04_difficult	0.305	**0.298**	1.297	0.281	**0.279**	0.545	**0.276**	0.276	0.277	**0.27**	0.284	0.421
MH_05_difficult	0.431	**0.419**	1.545	0.426	**0.397**	0.374	0.43	**0.426**	0.427	**0.428**	0.442	0.583
V1_01_easy	**0.113**	0.114	0.65	**0.114**	0.114	0.230	0.113	**0.112**	0.112	0.114	**0.113**	0.175
V1_02_medium	**0.299**	0.302	1.701	**0.301**	0.319	0.558	0.292	**0.29**	0.314	0.301	**0.292**	0.449
V1_03_difficult	**0.283**	0.417	3.937	**0.251**	0.271	1.348	0.257	**0.256**	0.257	**0.249**	0.250	1.433

The *Ours* and *VINS* columns show MSE of trajectory by our method and the latest open source code of VINS respectively. the *Original* columns show the MSE when online temporal calibration is not performed. d_t is the time shift added on camera timestamps.

TABLE II
MEAN SQUARE ERROR OF TRAJECTORY COMPARED WITH TRAJECTORY ON TEMPORALLY ALIGNED DATA

	Trajectory MSE(m)											
	d_t =60ms			d_t =30ms			d_t =0ms			d_t =-30ms		
	Ours	VINS	Original	Ours	VINS	Original	Ours	VINS	Original	Ours	VINS	Original
MH_01_easy	**0.013**	0.085	Fail	**0.019**	0.055	0.212	**0.007**	0.011	0	**0.006**	0.029	0.313
MH_02_easy	**0.016**	0.064	0.467	**0.011**	0.046	0.467	**0.005**	0.013	0	**0.008**	0.013	0.271
MH_03_medium	0.025	**0.016**	Fail	**0.007**	0.014	0.465	**0.004**	0.011	0	0.032	**0.014**	0.417
MH_04_difficult	0.103	**0.069**	1.237	0.028	**0.022**	0.412	**0.005**	0.009	0	**0.035**	0.037	0.448
MH_05_difficult	**0.011**	0.049	1.374	**0.005**	0.045	0.421	**0.005**	0.006	0	**0.012**	0.031	0.295
V1_01_easy	**0.007**	0.020	0.579	0.006	**0.005**	0.144	**0.001**	0.001	0	**0.003**	0.003	0.126
V1_02_medium	**0.048**	0.081	1.634	**0.035**	0.040	0.365	**0.072**	0.073	0	0.080	**0.069**	0.388
V1_03_difficult	**0.089**	0.283	3.893	**0.039**	0.109	1.315	**0.003**	0.010	0	0.045	**0.032**	1.387

The *Ours* and *VINS* columns show the MSE of trajectory by our method and the latest open source code of VINS respectively. The *Original* columns show the MSE when online temporal calibration is not performed. d_t is the time shift added on camera timestamps.

Our approach seems to converge faster on the data captured by our smart phone and helps dealing with the tough data containing fast turnings and sudden stops. But it might be prone to overestimate the time delay on our own data. Our approach may be applied on Android smart phones to calibrate the time delay between camera and IMU online.

REFERENCES

[1] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and automation, 2007 IEEE international conference on*, pages 3565–3572. IEEE, 2007.

[2] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[3] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *arXiv preprint arXiv:1708.03852*, 2017.

[4] Raúl Mur-Artal and Juan D Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[5] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[6] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1280–1286. IEEE, 2013.

[7] Elmar Mair, Michael Fleps, Michael Suppa, and Darius Burschka. Spatio-temporal initialization for imu to camera registration. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 557–564. IEEE, 2011.

[8] Jonathan Kelly and Gaurav S Sukhatme. A general framework for temporal calibration of multiple proprioceptive and exteroceptive sensors. In *Experimental Robotics*, pages 195–209. Springer, 2014.

[9] Michael Fleps, Elmar Mair, Oliver Ruepp, Michael Suppa, and Darius Burschka. Optimization based imu camera calibration. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3297–3304. IEEE, 2011.

[10] Mingyang Li and Anastasios I Mourikis. Online temporal calibration for camera–imu systems: Theory and algorithms. *The International Journal of Robotics Research*, 33(7):947–964, 2014.

[11] Joan Sola. Quaternion kinematics for the error-state kf. 2015.

[12] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016.

[13] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.