2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Visual Inertial Odometry using Coupled Nonlinear Optimization

Euntae Hong and Jongwoo Lim*

***Abstract*— Visual inertial odometry (VIO) gained lots of interest recently for efficient and accurate ego-motion estimation of robots and automobiles. With a monocular camera and an inertial measurement unit (IMU) rigidly attached, VIO aims to estimate the 3D pose trajectory of the device in a global metric space. We propose a novel visual inertial odometry algorithm which directly optimizes the camera poses with noisy IMU data and visual feature locations. Instead of running separate filters for IMU and visual data, we put them into a unified non-linear optimization framework in which the perspective reprojection costs of visual features and the motion costs on the acceleration and angular velocity from the IMU and pose trajectory are jointly optimized. The proposed system is tested on the EuRoC dataset for quantitative comparison with the state-of-the-art in visual-inertial odometry and on the mobile phone data as a real-world application. The proposed algorithm is conceptually very clear and simple, achieves good accuracy, and can be easily implemented using publicly available non-linear optimization toolkits.**

I. INTRODUCTION

In robot and automobile applications estimating ego-motion is of utmost importance. Up to now many sensors and systems are proposed to this end, including wheel odometer, inertial measurement unit (IMU), GPS, and visual odometry (VO). Recently visual inertial odometer (VIO) gets more and more interest since it can overcome the shortcomings of other systems. IMU-based ego-motion estimation works well for a short period of time, but suffers severe drift due to sensor noise. GPS-based systems can provide global absolute locations (latitude and longitude), but only work outdoor and cannot estimate small-scale (sub-meter) motions. Visual odometery is more robust to drift since it can observe far visual features for an extended period of time, but it is susceptible to inaccurate or insufficient visual features due to lack of scene textures, motion blur by fast motion, or low illumination. Moreover monocular visual odometry can only measure the motion up to scale unless other additional information on the sensor or environment is provided.

By fusing the IMU and visual information together, VIO can operate in extreme conditions where VO fails, and achieve higher accuracy with less drift than IMU-based approaches. Also the metric scale of the motion can be recovered from the inertial measurements.

VIO has been studied mainly in two directions: filtering-based and optimization-based. In filtering-based approaches [1], [2], [3], [4], [5], extended Kalman filters or similar are designed to estimate the current state (e.g., 3D pose and covariance) from the acceleration and angular velocity from IMU and the visual feature locations in the image. They can run very fast enough for real-time applications but they are less accurate than optimization-based approaches due to the approximations in the state update step. Optimization-based approaches [6], [7], [8], [9], [10] can be very accurate (e.g., the batch optimization solution in this paper) since it can use more information and more computation, but making them fast while keeping the accuracy in an acceptable level is a challenging and interesting research problem.

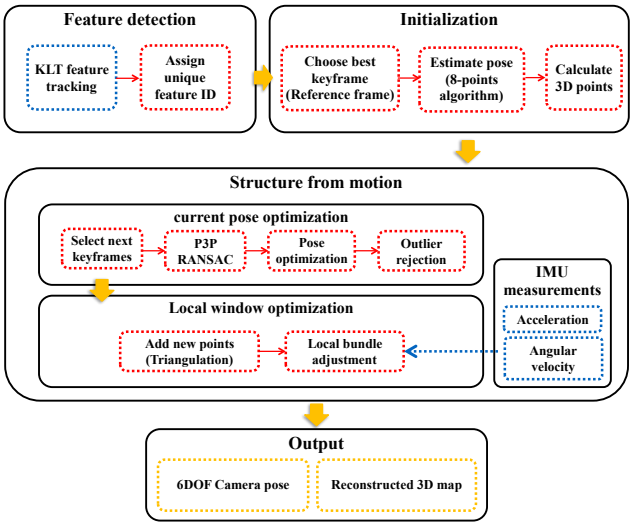


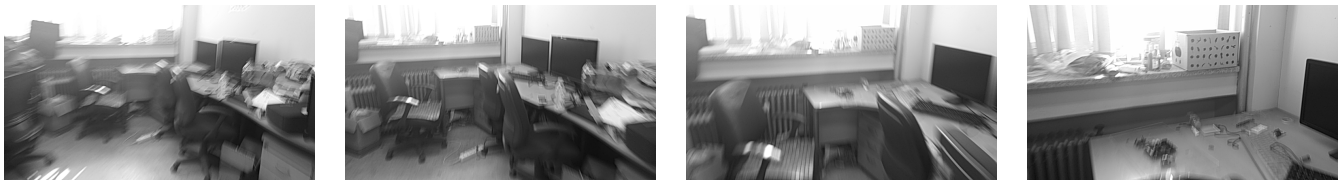
Fig. 1: A flowchart of the proposed algorithm, showing the steps performed by feature detection, initialization and Structure from motion. We proposed directly couples inertial and visual observations on nonlinear optimization.

In this paper, we propose a novel VIO algorithm which uses coupled nonlinear optimization of the camera poses on the inertial measurements and visual feature locations. First we derive a concise model that relates the camera's 6-DOF poses and the 3D visual landmark positions with the IMU readings and tracked visual feature locations. The cost in the optimization framework is simply the inconsistency in the visual and inertial relations, and the parameters (camera poses, landmark positions, and IMU biases) which minimizes the cost are incrementally updated as more observations become available.

To verify the correctness of the proposed approach, we implemented the batch optimization on all feature tracks and IMU readings. The batch results are the most accurate ones we can achieve using the model and data, and we show that they are much better than those of the state-of-the-art

*This work was supported by some organizations.

Euntae Hong and Jongwoo Lim are with Division of Computer Science and Engineering, Hanyang University, Seoul, 133-791, Korea. {hongeuntae,jlim}@hanyang.ac.kr



(a) (b) (c) (d)

Fig. 2: A captured smart phone (LG G3) dataset involving forward-backward fast motion. (a)-(d) shows fast motion makes global/local image blur and it causes visual measurements(feature tracking or matching performance) unstable. Our proposed method used KLT [11] feature tracker instead of using descriptor approaches like SIFT [12], SURF [13], BRIEF [14] and ORB [15] to get robust observation in dynamic environments.

algorithms. Inherently VIO needs to operate online and real-time. We propose an online VIO algorithm by limiting the optimization window to include only recent observations and parameters. As the window size varies the trade-off between estimation accuracy and computational cost takes place. One of our contributions is the design of the local optimization cost that is efficient but keeps the accuracy as much as possible. The details of the proposed online optimization algorithm is presented in Section III.

Our main contributions are summarized as follows.

- We propose a simple and unified framework that directly couples inertial and visual observations into one non-linear optimization problem for the VIO task.
- The batch and online versions of VIO is presented and we show that the accuracy-speed tradeoff according to the active window size.
- The experimental results show that the proposed algorithm can handle real-world data captured by a mobile phone, and achieves higher accuracy compared to the prior art in a well-known benchmark dataset.

In Section II, recent VIO algorithms and other related works are discussed. The proposed algorithm is described in detail in Section III, and the qualitative and quantitative experimental results are presented in Section IV. In Section V we conclude with discussions on the proposed algorithms and future work.

II. RELATED WORK

VIO has been studied actively for recent decades due to the demands from robotics, drones, and automotive applications. The core of VIO is the fusion of inertial and visual data, and there are two categories of the approaches: *f*iltering-based and *o*ptimization-based. The filtering-based approaches builds filters on measured inertial motions and estimated visual motions, whereas the optimization-based algorithms estimate camera poses using optimization using the captured data. Since filtering-based approaches only update the state once per frame, they are in general much faster but less accurate than optimization-based algorithms.

Most filtering based approach use by Extended Kalman Filter (EKF) or Unscented Kalman Filter [1] for fusing different sensor data by operating on a probabilistic state representation with the mean and covariance. EKF-based approaches are less computationally expensive and less accurate than optimization-based approaches [16]. Pinies *et al*. [2] proposed a loosely coupled SLAM system using IMU observations to get the real scale of the estimated map. Kleinert and Schleith [3] developed a camera-IMU system for real-time autonomous navigation by building the map using both inertial and image observations. In the multi-state constraint Kalman filter (MSCKF) [4], the visual and IMU data are coupled in the filter and the body poses are updated with triangulated keypoints and performs high accuracy than other EKF based approaches. Li and Mourikis [5] improved the performance of MSCKF by using a novel closed-form expression for the IMU error-state transition matrix and modeled the online camera-to-IMU extrinsic calibration.

Compared to the filtering approaches, the keyframe-based approaches use extracted image features and IMU data to solve a large nonlinear optimization problem on the motion estimation. The measurements from an IMU sensor is converted to 3D rigid motions and they are fused with feature observations. In general, thanks to powerful non-linear optimizers, keyframe-based methods accomplish higher accuracy but suffers from higher computational cost [6]. To mitigate the drawback it is common to only optimize the poses in a small local window [7] or to use incremental smoothing techniques [17]. Another way to make it online is to design a probabilistic cost function that combines visual and IMU terms [8], [9]. [8] is able to estimate 6DOF poses, velocities, and IMU biases with 3D map of sparse landmarks. However, since these approaches use a method of integrating IMU sensor reading, they suffer from the need for re-evaluate summation to be performed again according to the changed rotation at time of optimization. Recently, Foster *et al*. [10], [18] proposed a factor graph optimization framework using the pre-integrated IMU model that improved this disadvantage. The proposed method can be classified as a keyframe-based method using the optimization framework, but does not marginalize previous states and optimizes all pose in the local window, as opposed to previous methods. Therefore, our proposed method can be extended as an algorithm including loop closing as well as VIO. It also distinguishes it from other methods in a high-precision, simple, integrated framework that tightly combines visual and inertial measurements without preprocessors or smoothness.

III. PROPOSED ALGORITHM

Conventional visual odometry systems consist of feature tracking, pose estimation, and filtering or optimization steps. The feature points are detected and tracked throughout the input frames, then the camera motions between frames (or

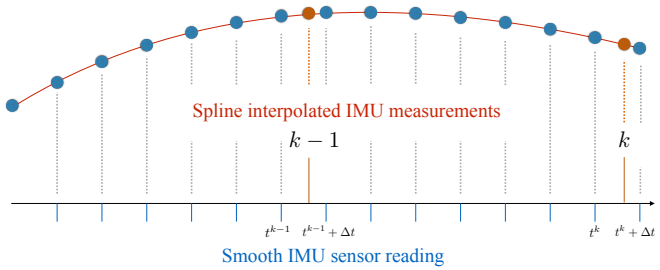


Fig. 3: A spline to interpolate IMU data on the camera frames. The estimated IMU sensor measurement at frame-k is calculated using immediate oncoming IMU data.

keyframes) are computed from the feature motions. The visible features are kept as 3D landmarks, and they are used to compute or improve the camera trajectory. As shown in Figure 1, the proposed algorithm follows the basic steps of the visual odometry, and incorporate the measurements of acceleration and angular velocity from the IMU.

There are several issues in using the IMU data in this framework. First the acquisition frequency of the IMU data is not consistent with and is much higher than the video frame rate.

In the proposed system we use the camera frame number (k) as the reference time index for device motion. Since the IMU measurement frequency is much higher than the frame rate, we assume that the additional IMU measurement $(t_k+\Delta t)$ immediately after the frame k is available. Separate Kalman filters for acceleration and angular velocity are used to mitigate the noise in the sensor readings, and we use spline interpolation to calculate the synchronized IMU estimates at the camera frames as shown in Figure 3. One could run a separate filter on IMU data and use the estimated motion by integrating at each frame as in [8], however, the integration of noisy raw IMU data in the filter is in general less accurate than optimizing the states with the IMU data. The IMU reading have biases and it is very important to model and estimate them accurately to prevent drift. Instead of modeling the bias in the filtering framework, the bias parameters are estimated in the optimization together with the camera poses and landmark positions by fully utilizing the visual feature observations.

Monocular visual odometer can only estimate the camera motion up to scale, and in VIO the acceleration data from IMU can be used to determine the metric scale. However the IMU only provides the noisy acceleration measurement of the device, and to recover the travel distance it has to be doubly integrated.

Also the raw IMU acceleration data contains the gravity component, and it must be canceled to compute the motion correctly which requires accurate vertical direction estimation. It is one of the merits of the proposed approach that the gravity cancellation is seamlessly incorporated in the process.

A. Mathematical Formulation

The 6-DOF camera pose $\mathbf{p}_t$ at time t is denoted as the rotation and translation vectors $(\mathbf{r}_t^\top, \mathbf{t}_t^\top)^\top$. The 3×3 rotation matrix R_t can be converted to/from $\mathbf{r}_t$: $R_t = \mathrm{R}(\mathbf{r}_t)$ and

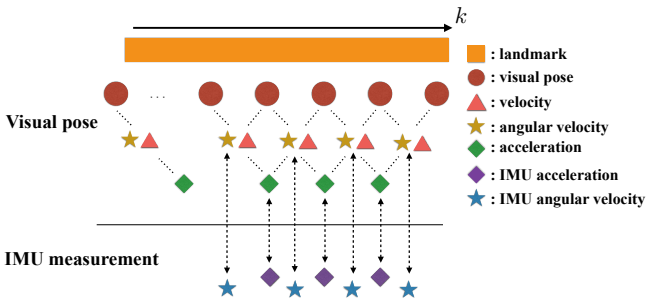


Fig. 4: Relations between camera poses and IMU measurements in the proposed optimization framework. Robust estimated pose from visual observation can be optimized directly with IMU sensor measurements.

$\mathbf{r}_t = \mathrm{rot}(R_t)$. The camera poses and landmark positions are in the ground coordinate system where z-axis is parallel to the gravity direction.

The acceleration $\hat{\mathbf{a}}_t$ measured by the IMU contains the gravity and some bias. It is in the device coordinate system, and we use $\hat{}$ to denote the coordinate system difference. To compute the acceleration $\mathbf{a}_t$ in the ground coordinate system, the accelerometer bias $\hat{\mathbf{b}}_a$ and the gravity $\mathbf{g}$ must be subtracted properly

$$\mathbf{a}_t = R_t^{-1}(\hat{\mathbf{a}}_t - \hat{\mathbf{b}}_a) - \mathbf{g}.$$

From three camera poses computed using visual features, we can also compute the acceleration $\bar{\mathbf{a}}_t$ as follows:

$$\bar{\mathbf{v}}_t = (\mathbf{t}_t - \mathbf{t}_{t-1})/\Delta t, \text{ and}$$
$$\bar{\mathbf{a}}_t = (\bar{\mathbf{v}}_t - \bar{\mathbf{v}}_{t-1})/\Delta t,$$

where Δt is the time between two consecutive frames. The acceleration error $\mathbf{e}_t^{acc}$ in our optimization framework penalizes the difference between the acceleration from visual features and that from the IMU

$$\mathbf{e}_t^{\mathrm{acc}} = \mathbf{a}_t - \bar{\mathbf{a}}_t. \tag{1}$$

The angular velocity $\hat{\omega}_t$ is also measured by the IMU, and the angular velocity in the ground coordinate system ω_t is

$$\omega_t = \mathrm{rot}(R_t^{-1}\,\mathrm{R}(\hat{\omega}_t - \hat{\mathbf{b}}_\omega)),$$

where $\hat{\mathbf{b}}_\omega$ is the gyro bias. The angular velocity $\bar{\omega}_t$ from the visual features is

$$\bar{\omega}_t = \mathrm{rot}(R_t\,R_{t-1}^{-1})/\Delta t,$$

and the angular velocity error $\mathbf{e}_t^\omega$ can be defined as

$$\mathbf{e}_t^\omega = \omega_t - \bar{\omega}_t. \tag{2}$$

The acceleration and angular velocity errors are visualized in Figure 4.

Finally the reprojection error $\mathbf{e}_{i,j}^{\mathrm{proj}}$ is computed using the camera pose $\mathbf{p}_j$ and the landmark position $\mathbf{x}_i$ as in the standard visual odometry,

$$\mathbf{e}_{i,j}^{\mathrm{proj}} = \mathrm{proj}(\mathbf{x}_i;\mathbf{p}_j) - \mathbf{f}_{i,j}, \tag{3}$$

where $\mathrm{proj}(\mathbf{x};\mathbf{p})$ is the projected image coordinate of a 3D point $\mathbf{x}$ at the camera $\mathbf{p}$, and $\mathbf{f}_{i,j}$ is the feature location of the landmark i in the image j.

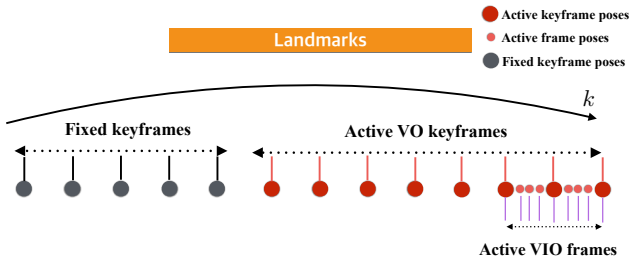


Fig. 5: Keyframes and frames in online optimization. We defined 2 different active VO keyframes/VIO frames for local optimization. Active VIO frames are included in the optimization to constrain the motion between consecutive frames.

B. Batch Optimization

When the video frames and the IMU measurements are given, we can compute the camera trajectory that minimizes all of the above errors. For the IMU readings $\{\hat{\mathbf{a}}_t, \hat{\boldsymbol{\omega}}_t\}_T$ and the tracked feature set $F = \{\mathbf{f}_{i,j}\}$, the cost function for the batch optimization

$$E_{batch} = \sum_{\mathbf{f}_{i,j} \in F} \left\| \mathbf{e}_{i,j}^{\text{proj}} \right\|_{\Sigma_f}^2 + \sum_{t \in T} \|\mathbf{e}_t^{\text{acc}}\|_{\Sigma_a}^2 + \sum_{t \in T} \|\mathbf{e}_t^{\omega}\|_{\Sigma_\omega}^2$$

is optimized on the cameras $\mathbf{p}_1, \ldots, \mathbf{p}_T$, the landmarks $\mathbf{x}_1, \ldots, \mathbf{x}_L$, the biases $\mathbf{b}_a$ and $\mathbf{b}_\omega$. $\|\cdot\|_\Sigma$ is the Mahalanobis distance with the covarience Σ. In this work we use diagonal covariance matrices with fixed variances σ_f^2, σ_a^2, and σ_ω^2. Note that the above formulation is the best we can do with the aforementioned model, and we verify that the batch-optimized results are (obviously) much better than any online algorithms. Batch optimization can generate most accurate results, but it requires all data to be ready before processing, and it takes long time to converge. We only use this batch optimization for model verification and performance evaluation, and the more efficient online version is presented in the next section.

C. Online Optimization

Considering its target applications, VIO must be able to output the current camera pose in real-time. To this end we propose a simplified version of the batch optimization. We focus how to reduce the number of parameters without sacrificing the performance significantly.

In most VO systems keyframes are chosen when there exist significant changes in motion or visual contents. The reprojection error terms are only built upon the keyframes and the landmarks. Keyframes are used to integrate the information for a longer period of time without exploding the number of parameters from the intermediate frames. We adopt the convention and use keyframes for landmark generation and pose optimization. At every frame the camera pose is computed using the image location of the 3D landmarks, and when the camera is moved enough the frame is initialized as a keyframe. The feature trajectories in the keyframes provide enough constraint on camera poses and landmark positions in the optimization (Eqn. 3).

However it can be problematic if the IMU measurements are averaged between keyframes, since the keyframes are much sparser than the frames. As shown in Figure 5 the frames between recent a few keyframes are included in the optimization to constrain the motion between the consecutive frames as in Eqn. 1 and 2. These active VIO frames links the keyframes with inertial constraints. The feature locations of these frames are not used in the optimization since it increases the computational cost and the feature motions between consecutive frames are not significant. When a keyframe is added, the online optimization is performed on the active VO keyframes K_a and the active landmarks L_a, as well as the active VIO frames T_a

$$E_{online} = \sum_{j \in K_a \cup K_f,\, i \in L_a} \left\| \mathbf{e}_{i,j}^{\text{proj}} \right\|_{\Sigma_f}^2 + \sum_{t \in T_a} \|\mathbf{e}_t^{\text{acc}}\|_{\Sigma_a}^2 + \sum_{t \in T_a} \|\mathbf{e}_t^{\omega}\|_{\Sigma_\omega}^2, \tag{4}$$

where K_f are the fixed VO keyframes which are needed to enforce the landmark positions to be consistent with the observations outside the local optimization window. Algorithm 1 summarizes the online optimization process for VIO.

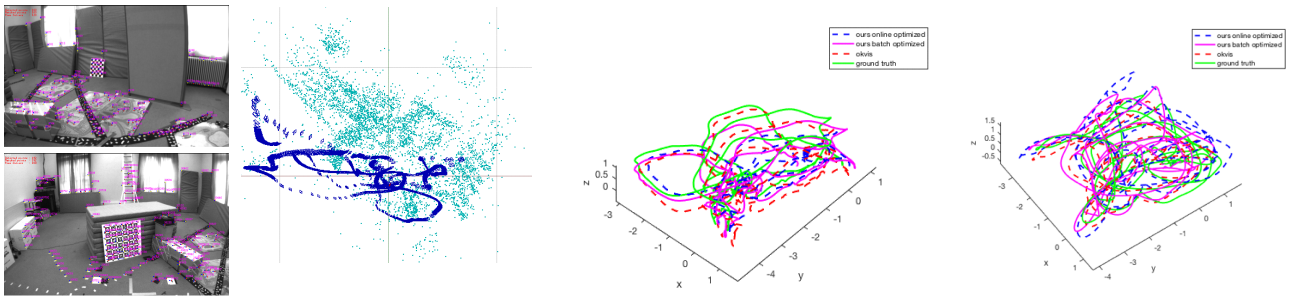
Algorithm 1: Proposed online VIO algorithm

Data: Images, accelerations and gyro
Result: 6DOF poses and landmarks
Initialization : Select 2 keyframes to initialize landmarks. Then, calculate the real scale using integrated IMU sensor measurements.;
for *k = 1 to K* **do**
  Extract and tracking keypoints using KLT [11];
  Estimate k^{th} frame's pose using p3p;
  Performs pose optimization;
  if *k^{th} frame is keyframe* **then**
    Add new landmarks;
    Perform online optimization minimizing cost function Eqn. 4.
  end
end
return optimized 6DOF pose and landmarks involving real scale

D. Implementation Details

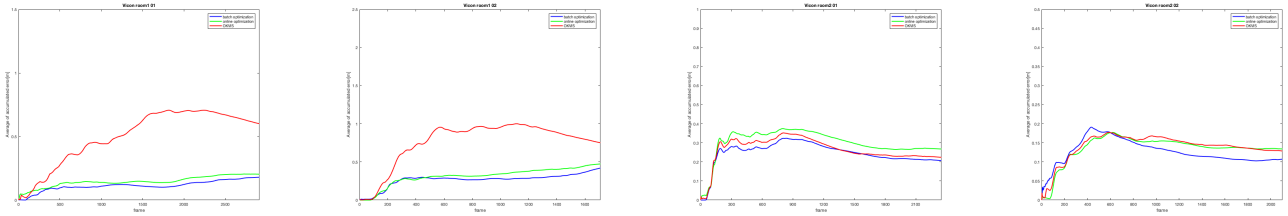
For robust VIO operation, visual feature detection and tracking is very important. In VSLAM (visual simultaneous localization and mapping) or image retrieval, more sophisticated features such as SIFT [12] or SURF [13] are used for their matching performance, but they are not suitable for real-time application due to high computational cost. For VO or VIO systems either fast binary descriptors like BRIEF [14] or ORB [15], or feature tracker like KLT [11] are commonly used. In our work we use Harris corner detector and KLT for feature tracking, but it can be replaced with other methods without affecting the core optimization part.

Initialization of the world model is another important issue in monocular systems, since the metric scale of the camera motion cannot be recovered from the visual features. As discussed earlier the absolute scale can be recovered from IMU data, but during initialization there exist only a limited amount of noisy IMU data. We choose the strategy of initializing the model roughly correct using the available data (visual information), and then update the model using



(a) (b)

Fig. 6: (a) Left two images from the dataset Room1.2 shows the tracked 2d features. Red points represent pose inliers, yellow shows outliers, and magenta numbers are landmark IDs. The right image is the estimated camera trajectory(blue) and landmarks (teal). (b) The ground-truth and estimated trajectories of Room1.1 and Room1.2 sequences.



Room1.1 Room1.2 Room2.1 Room2.2

Fig. 7: The average location error at each frame, for comparison of the proposed methods and OKVIS [8]. Green and blue line represent our online/batch process results and red line shows OKVIS results.

incoming measurements during the online optimization process. To determine the initial scale factor s we compute the actual travel distance d_{IMU} between the first two keyframes by integrating the IMU data:

$$d_{IMU} = \sum_t \left\| \mathbf{v}_t \Delta t + \frac{1}{2} \tilde{\mathbf{a}}_t \Delta t^2 \right\|, \quad \text{where } \mathbf{v}_t = \mathbf{v}_{t-1} + \tilde{\mathbf{a}}_t \Delta t,$$
$$d_v = \sum_t \|\mathbf{t}_t - \mathbf{t}_{t-1}\|, \text{ and}$$
$$s = \frac{d_{IMU}}{d_v}.$$

The computed scale s is then multiplied to the camera and landmark locations, and the online optimization is performed on the two keyframes and all frames in-between for more accurate initialization.

IV. Experimental Results

We have evaluated our proposed method on various datasets, and the experimental results are presented in this section. The system is implemented in C++ without multi-threading or GPU optimization, and ran on an Intel Core i7 3.0G CPU laptop with 16GB RAM.

A. Comparison Experiments

For quantitative performance evaluation we used the EuRoC micro aerial vehicle datasets [19], which is collected from the 'Firefly' hex-roter helicopter with stereo camera and inertial measurement unit in slow and fast flying speed. This dataset offers the 6D ground truth pose of the vehicle acquired from a Vicon motion capture system at 100 Hz.

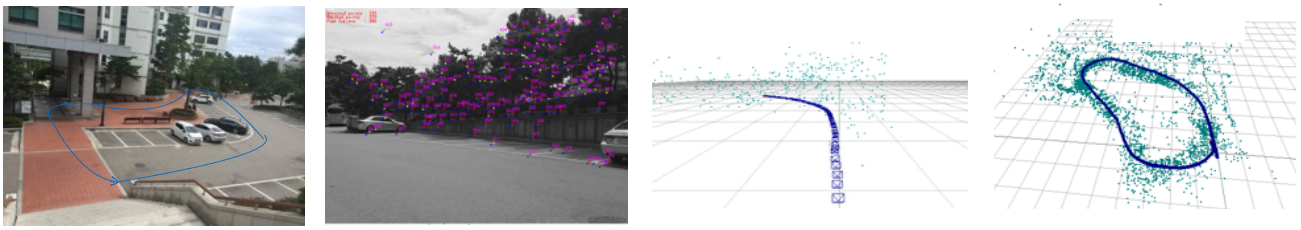
EuRoC seq.	Ours (batch)	Ours (online) w=15	w=7	w=3	OKVIS [8]
Room1.1	0.2164	0.2292	0.2384	**0.2457**	0.5317
Room1.2	0.3064	0.3213	0.3540	**0.3789**	0.7895
Room2.1	0.2116	0.2651	0.2941	0.3410	**0.2792**
Room2.2	0.1344	0.1420	0.1499	**0.1516**	0.1579
Overall	0.2172	0.2394	0.2591	**0.2793**	0.4395

TABLE I: Average distance error on EuRoC dataset (unit: meters).

In our experiments we only use the left images and the IMU data in Vicon room 1 01-02 (referred as Room1.1, Room1.2) and room 2 01-02 (Room2.1, Room2.2) sequences for evaluation (Figure 6).

We ran four experiments on each sequence; three online optimization with different window sizes, and one batch optimization. For comparison we ran OKVIS [8] on the same sequence with the configuration parameters provided by the author. The relative error metric for 3d position proposed in [20] is used for all quantitative evaluation.

As shown in Table I the batch optimization gives the best accuracy for all sequences, which clearly outperforms the OKVIS result. As the window size decreases from 15 to 3 the error increases slightly but still better than the OKVIS system overall. However the processing time (Table II) decreases significantly as the window size shrinks. Feature detection and tracking takes about 17.1 ms on average for 130-150 features at every frame, and by replacing this with a more



(a) (b) (c) (d)

Fig. 8: (a) Outdoor trajectory of ˜100m travel with identical start and end positions. The start to end error of proposed method is 1.8m. (b) An example of tracked features (red: inliers, yellow: outliers, blue: feature tracks, magenta: landmark id). (c,d) Snapshots of the estimated trajectory(blue pyramids) and reconstructed landmarks(green dots).

	Ours (online)		
EuRoC seq.	w=3	w=7	w=15
Room1.1	45	62	78
Room1.2	44	58	72
Room2.1	43	55	71
Room2.2	49	61	78

TABLE II: Average processing time per frame on EuRoC dataset

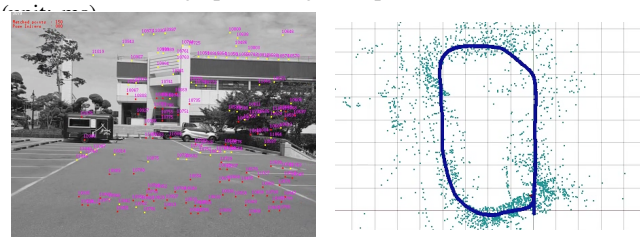


Fig. 9: The left image shows an image of the test sequence collected by a mobile phone (Samsung Galaxy Note4). The person walked around 70 meters and returned to the starting position. The start to end error of proposed method is 1.3m. The right image shows the camera trajectory(blue) and reconstructed landmarks(green).

efficient method (e.g., FAST-ORB) it is possible to achieve the real-time performance. For challenging sequences (e.g., Room1.2 for fast motion) the error increased much more than other easier sequences, which shows that considering more frames in optimization helps achieving better accuracy in difficult situations. Figure 7 shows the average location errors of the proposed algorithm and OKVIS at each frame in the sequences.

The average location errors of the proposed methods are 0.2172 (batch optimization) and 0.2793 (online optimization), where OKVIS records 0.4395 for all four sequences. The experimental results show that our method shows more stable and better performance compared to the OKVIS system.

B. Mobile phone Experiments

In addition to the public dataset, we tested the dataset captured using mobile phones (Samsung Galaxy Note4 and LG G3) which equip with 30fps camera and 200Hz 3-axis gyroscope and a 3-axis accelerometer. The datasets are

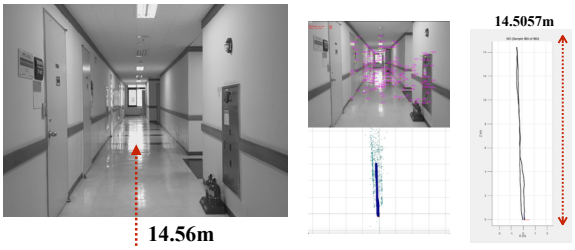


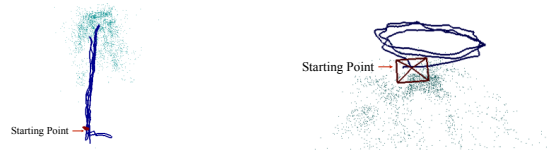
Fig. 10: An experiment for real scale estimation. The left image shows real space distance measured my manually. The right image shows estimated traveled trajectory. The estimated travel distance is 14.50m, and the difference from the real distance is only 0.06m.

collected by the app developed by ourselves, and processed on the same laptop for evaluation. We have collected several datasets indoor and outdoor in usual walking speed, and we present four results in this paper (Figure 8 - 11).

Figure 10 is recorded using Samsung Glaxy Note4 in a corridor to test the accuracy of the proposed algorithm. The phone is walked for 14.56m forward and then backward to the original position. The estimated distance of travel by our system was 14.51m, and the difference of the starting and end position was 0.03m. The estimated trajectory was also qualitatively close to the real one.

In Figure 8 we show the result of a longer travel in outdoor from Samsung Galxy Note4. The person walked around 100m in a parking lot and came back to the starting position. The location error of the starting and end points were 1.8m, which shows the accuracy of the proposed algorithm. Figure 9 shows another outdoor sequence for 70m travel and 1.3m location error between the start and end.

Figure 11 is recorded using LG G3 to show the results of experiencing fast linear and circular motion in the room illustrated on Figure 2. On this dataset we evaluate the difference between start and end positions as the device is moved on a loop. The start-to-end error of proposed approach is 0.42m for fast motion (loop length: 68m) dataset and 0.12m for circulation motion (loop length: 18m) dataset. These results show that our approach can performs well even if the visual measurements are unstable cased by blur image. The outdoor experiment is performed on a street and Figure 12 shows the estimated trajectory overlayed on the Google map. The position error between end points of estimated trajectory and real map points which is 1.2m. These experimental results



(a) (b)

Fig. 11: The 3D trajectories of two experiments performed with the dataset from LG G3 (left image : fast motion, right image : circular motion) are presented. The estimated trajectory is shown as blue line and red square represents current camera attitude.

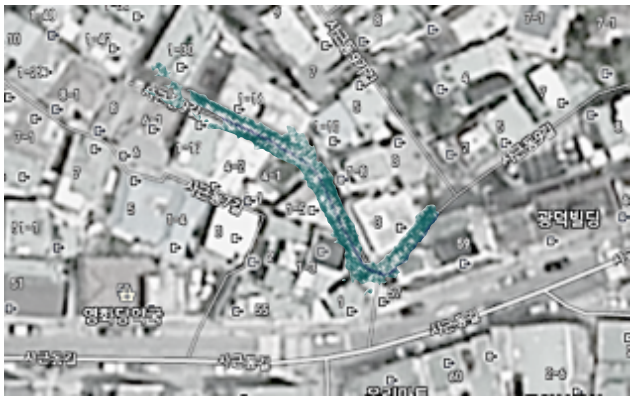


Fig. 12: An outdoor trajectory (length : 85m) on the street is overlayed on the real map. The difference between the estimated end points and the points on the map is 1.2m. The green dots represent the 3D landmarks reconstructed by the prosed algorithm.

show that the proposed algorithm can operate robustly and minimize drift in outdoor environments as well as various benchmark data sets.

V. Conclusion

We propose a novel visual inertial odometry algorithm which directly optimizes the camera poses using the visual and inertial measurements. The proposed framework is conceptually simple and straightforward, and can be implemented easily with publicly available optimization libraries. The batch version of our method shows that the proposed approach can handle noisy real data and achieves the most accurate results. The online version can process the incoming data online by using sliding-windowed optimization with inertial and reprojection costs for the keyframes and VIO frames. We experimentally show that the proposed algorithms outperform the state-of-the-art algorithm, and demonstrate that it works with noisy mobile phone data.

For future work we plan to make the system run faster to achieve real-time performance, and make it more robust to external disturbances such as extreme illumination changes and abrupt motions.

ACKNOWLEDGMENT

This research was supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2017(R2015040004), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1A2B4011928).

References

[1] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. Ieee, 2000, pp. 153–158.

[2] P. Piniés, T. Lupton, S. Sukkarieh, and J. D. Tardós, "Inertial aiding of inverse depth slam using a monocular camera," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 2797–2802.

[3] M. Kleinert and S. Schleith, "Inertial aided monocular slam for gps-denied navigation," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*. IEEE, 2010, pp. 20–25.

[4] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[5] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual–inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[6] S.-H. Jung and C. J. Taylor, "Camera trajectory estimation using inertial sensor measurements and structure from motion results," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2. IEEE, 2001, pp. II–732.

[7] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Generic and real-time structure from motion using local bundle adjustment," *Image and Vision Computing*, vol. 27, no. 8, pp. 1178–1193, 2009.

[8] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[9] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and imu," *Robotics*, vol. 17, pp. 17–24, 2013.

[10] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration theory for fast and accurate visual-inertial navigation," *arXiv preprint arXiv:1512.02363*, 2015.

[11] S. Birchfield, "Klt: An implementation of the kanade-lucas-tomasi feature tracker," 2007.

[12] P. C. Ng and S. Henikoff, "Sift: Predicting amino acid changes that affect protein function," *Nucleic acids research*, vol. 31, no. 13, pp. 3812–3814, 2003.

[13] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[14] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. IEEE, 2011, pp. 2564–2571.

[16] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," in *Robotics research*. Springer, 2010, pp. 201–212.

[17] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, p. 0278364911430419, 2011.

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, 2016.

[19] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, p. 0278364915620033, 2016.

[20] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.