A Tightly-Coupled Visual-Inertial System with Synchronized Time for Indoor Localization

Nguyen H. K. Tran and Vinh-Hao Nguyen
Ho Chi Minh City University of Technology, VNU-HCM, Ho Chi Minh City, Vietnam
Email: {thknguyen, vinhhao}@hcmut.edu.vn

***Abstract*—Indoor localization remains a difficlut problem due to the lack of global positioning facilities. In this paper, we present an indoor localization system based on tightly coupling the stereo camera and the inertial measurement unit (IMU). To achieve the correct fusion, we strictly synchronized the sensors by compensating for all of system delays, including the varied exposure time of camera. The calibration datasets were carefully recorded to obtain the exact model of the stereo camera as well as the relationship between the camera and the IMU. After preparing, we performed the localization by utilizing the keyframe-based visual-inertial odometry. This algorithm works by minimizing the IMU error jointly with the camera reprojection, to get an optimal estimation of robot states. The visual landmarks are calculated by keypoint matching and triangulation between current frames and keyframes. The keyframes are selected to view keypoints from different angles to improve the depth uncertainty of landmarks. The algorithm was put into practice by an embedded computer which has enough processing ability, while presented in a small size for convenient deployment on mobile robots. The experimental results indicated that the developed system could achieve sufficient accuracy and robustness under real-world conditions.**

***Index Terms*—visual-inertial odometry, stereo camera, IMU, time synchronization, indoor localization, mobile robot**

I. Introduction

Many mobile robots rely on their location information to operate autonomously. In indoor environments without any global positioning systems, a solution of Simultaneous Localization and Mapping (SLAM) [1]-[3] is required. In recent years, vision-based SLAM algorithms, or Visual Odometry (VO), are widely researched due to the popularity and cost-efficiency of cameras [4]-[6]. However, the low update rate caused by high computational cost usually makes it difficult for VO systems to track fast and dynamic motions. Integrating the inertial measurement unit (IMU) into the system could solve this drawback. By propagating the states in between two update cycles using IMU, the system could produce an approximate prediction that helps the estimation converge with higher precision and robustness. This methodology is known as Visual-Inertial Odometry (VIO).

Consider the level of integration, we can divide the VIO systems into loosely and tightly coupled. In loosely coupled systems, each sensor separately estimated its states and the fusion only happens at final stage. The advantage of this method is the simple design where sensors could be combined as modules without changing the inside operation. For instance, the work in [7] fused the output of the SVO algorithm [6] with the IMU estimator by using the Multi-Sensor Fusion approach [8]. The quadrotor low-level controller PIXHAWK [9] loosely integrated vision and IMU using the Extended Kalman Filter (EKF). However, the lack of consideration of internal relationships between sensors makes the loosely integrated systems not achieve high accuracy. Recent studies focused on tightly coupled systems, where the estimation is performed based on constraints between sensors to achieve the optimal result. The ROVIO framework [10] tightly fused the sensors by means of an Iterated Extended Kalman Filter (IEKF). In contrast, the optimization-based OKVIS method [11] jointly estimated the IMU and camera in a combined cost function. The comparison in [12] shown that tightly integrated visual-inertial systems exhibit superior accuracy compared to the loosely ones under a same condition.

Challenges of building a tightly coupled VIO systems comes from its complexity. First, all sensors must be strictly synchronized in time. When time delay exists, a sensor state would be wrongly estimated because it relates to the other sensor data. Similarly, intrinsic and extrinsic parameters of sensors should be accurately calibrated to avoid estimation error. The algorithm needs to be correctly configured to achieve good performance. Finally, the system hardware should be built with enough processing power to run the algorithm, but still stay in a compact form to deploy on mobile robots.

In this work, we gradually handle those challenges. Particularly, we proposed a time synchronization method that explicitly derives each time delay in the system and compensates for them. The datasets for calibration were recorded according to specific criterions. We utilized the OKVIS [11] method to be the VIO algorithm. The method is based on batch optimization the reprojection and IMU errors to update the robot states. Its front-end employed the keyframe approach, keeping keyframes and current frames in a bounded optimization window.

The remaining of this paper is organized as follows. Section II describes the method of time synchronization

Manuscript received February 17, 2020; revised March 13, 2020.

doi: 10.18178/ijmerr.10.4.202-208

and sensor calibration. Section III presents the visual-inertial odometry algorithm. Experimental setup and results are shown in Section IV. Finally, we give the research conclusion in Section V.

II. System Synchronization and Calibration

A. Time Synchronization

Time synchronization is a crucial demand in tightly coupled systems where all sensors are jointly estimated. To start with, we program the IMU board to generate a periodic pulse signal for external triggering the stereo camera. The trigger signal uses the same clock source with the IMU polling, but is prescaled to match the camera framerate. Sensor data is timestamped at the received time on the computer. However, these timestamps have been delayed from the true timestamps due to camera shuttering, IMU filtering and data transmission. Consequently, the IMU and camera timestamps are misaligned. To solve this problem, we proposed a method that calculates the system delays, shifts the trigger pulse and corrects the timestamps to the synchronized position, as shown in Fig. 1.

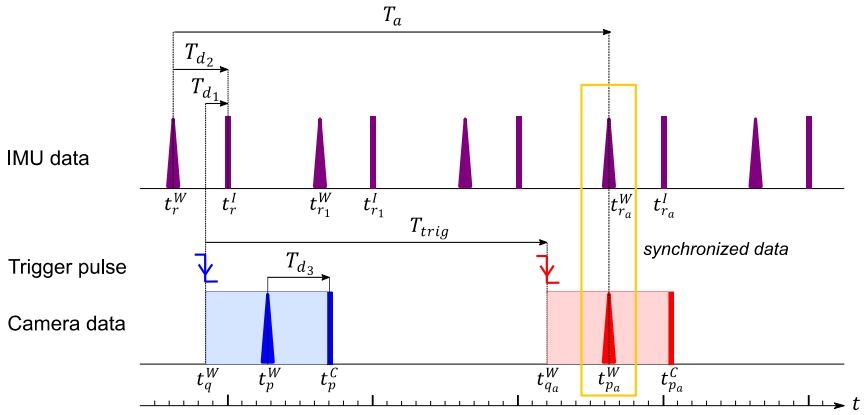


Figure 1. Time synchronization scheme between IMU and camera. Vertical bars mark the timestamps when data is received. Triangles show the true timestamps when the sensors capture the world. Trigger pulses are visualized by falling edges. Blue indicates the original trigger signal and camera data with exposure delay, while red shows the synchronized ones. IMU data is colored purple. The yellow rectangle marks a synchronized IMU-camera data pair.

On the symbol of timestamp t, let the subscripts r, q, p stand for the IMU data, the trigger pulse and the camera data, respectively. Additionally, superscripts represent the timeline where t is expressed, including the IMU received time I, the camera received time C and the true world time W. We define the system delays with respect to the world time as:

$$T_{d_1} \coloneqq t_r^I - t_q^W, \qquad T_{d_2} \coloneqq t^I - t^W = t_r^I - t_r^W, \quad \text{(1a)}$$

$$T_{d_3} \coloneqq t^C - t^W = t_p^C - t_p^W = \frac{T_{shutter}}{2}, \quad \text{(1b)}$$

where T_{d_1} denotes the difference in transmission time between IMU and camera to the computer, which also is the difference from the trigger pulse to its contemporary IMU data; T_{d_2} denotes the internal filter delay of IMU; and T_{d_3} indicates the image delay which is a half of camera shutter time $T_{shutter}$. Naturally, timestamping an image at its middle exposure will best represent the average motion captured.

To find the system delays, we need to calculate the time differences:

$$d_L \coloneqq t^I - t^C = \frac{-T_{shutter}}{2} + T_{d_2}, \quad \text{(2a)}$$

$$d_S \coloneqq t_r^I - t_p^C = -T_{shutter} + T_{d_1}, \quad \text{(2b)}$$

where d_L is the difference between the IMU and camera timeline, d_S is the difference between the IMU and camera received timestamp. We simply obtain d_S from the value of t_r^I and t_p^C. $T_{shutter}$ is read from the camera because the light condition changes continuously. Moreover, d_L are estimated by employing the temporal calibration function of the Kalibr toolbox [13]. From the above values, we found the system delays T_{d_1}, T_{d_2} and T_{d_3} as well as the true timestamps t_r^W and t_p^W. Then, we can align the camera and IMU timestamps by shifting the trigger pulse for an appropriate period T_{trig} as:

$$T_{trig} \coloneqq t_{q_a}^W - t_q^W = T_a - d_L + d_S, \quad (3)$$

where t_q^W and $t_{q_a}^W$ denotes the original and modified trigger timestamps. Instead of synchronizing at the first IMU timestamp t_r^W, we choose a later timestamp $t_{r_a}^W$ to keep T_{trig} always positive for simple implementation. The alignment offset, T_a, is a constant multiple of the IMU cycle but not larger than the camera sampling time.

B. Sensor Calibration

Fig. 2 illustrates the calibration setup of the visual inertial sensor. $\mathcal{F}_{\mathcal{A}}$ denotes the coordinate frame of calibration pattern. $\mathcal{F}_{\mathcal{C}_1}$, $\mathcal{F}_{\mathcal{C}_2}$ mark the coordinate frame of stereo cameras, and $\mathcal{F}_{\mathcal{I}}$ stands for the IMU coordinate frame. Since the sensors are rigidly mounted, the transformations among $\mathcal{F}_{\mathcal{C}_1}$, $\mathcal{F}_{\mathcal{C}_2}$ and $\mathcal{F}_{\mathcal{I}}$ are considered invariants. The calibration is divided into two stages: camera-camera calibration that estimates the camera

parameters, and camera-IMU calibration that estimates the camera-IMU transformation.

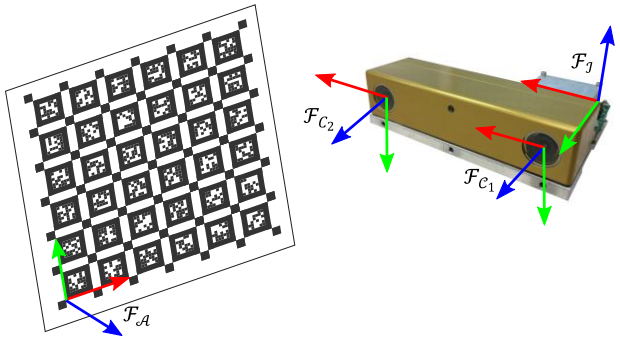


Figure 2. Coordinate frames of visual inertial sensors and calibration pattern. The calibration pattern is formed by AprilTag [14].

In the camera-camera stage, we record a dataset of stereo image capturing the motion of calibration pattern in front of the standing-still camera. To obtain a good calibration, we move the pattern to different positions and orientations while passing it through the entire image area. Once again, we employ the Kalibr toolbox [13] to compute the calibration. The toolbox will estimate the intrinsic camera model, $\mathbf{h}(.)$, and the stereo transformation, $\mathbf{T}_{\mathcal{C}_1\mathcal{C}_2}$, based on the pinhole camera model combined with equidistant distortion model.

For the camera-IMU calibration, a dataset is collected by waving the sensor pair in front of the fixed pattern. The motion should be smooth and cover all the IMU linear and angular axes with sufficient speed to ensure that the IMU measurement is well observed. Besides, the camera needs good light condition and low shutter time to avoid blurring images. To provide the IMU noise characteristics for the toolbox, we conduct an experiment placing the IMU in standstill for 5 minutes and inspecting the data, as the real noises are usually larger than those specified in the datasheet. Finally, the Kalibr toolbox [13] estimates the camera-IMU transformation $\mathbf{T}_{\mathcal{C}_1\mathcal{J}}$ together with the time-varying IMU pose, accelerometer and gyroscope biases in a batch optimization, where the IMU pose is represented by B-splines and biases are modeled as random walks.

III. Visual-Inertial Odometry Algorithm

From the synchronized and calibrated system, we estimate the robot motion by utilizing the state-of-the-art Keyframe-based Visual-Inertial SLAM (OKVIS) algorithm [11]. This method is based on the tightly integration between IMU and camera by optimizing a joint cost function. Keyframes from different views are maintained in the process window to improve the system robustness. Furthermore, the window is bounded by marginalization to keep the problem computable in real-time. The cost function is minimized using the Google Ceres nonlinear least-square solver [15]. The method is summarized as below.

First, let $\mathbf{x}$ denotes the state vector of robot as:

$$\mathbf{x} \coloneqq [\mathbf{r}^{\mathcal{W}} \quad \mathbf{q}_{\mathcal{W}\mathcal{J}} \quad \mathbf{v}^{\mathcal{J}} \quad \mathbf{b}_{\omega} \quad \mathbf{b}_{\mathrm{a}}]^T, \tag{4}$$

where $\mathbf{r}^{\mathcal{W}}$ and $\mathbf{q}_{\mathcal{W}\mathcal{J}}$ stand for the position and orientation of IMU in the world coordinate frame $\mathcal{F}_{\mathcal{W}}$, $\mathbf{v}^{\mathcal{J}}$ denotes the IMU body velocity in the IMU coordinate frame $\mathcal{F}_{\mathcal{J}}$, $\mathbf{b}_{\omega}$ and $\mathbf{b}_{\mathrm{a}}$ mark the gyroscope and accelerometer biases.

The states are propagated by using the nonlinear IMU model:

$$\begin{aligned}
\dot{\mathbf{r}}^{\mathcal{W}} &= \mathbf{C}_{\mathcal{W}\mathcal{J}}\mathbf{v}^{\mathcal{J}}, \\
\dot{\mathbf{q}}_{\mathcal{W}\mathcal{J}} &= -\boldsymbol{\Omega}(\boldsymbol{\omega}^{\mathcal{J}})\mathbf{q}_{\mathcal{W}\mathcal{J}}, \\
\dot{\mathbf{v}}^{\mathcal{J}} &= \mathbf{a}^{\mathcal{J}} + \mathbf{C}_{\mathcal{J}\mathcal{W}}\mathbf{g}^{\mathcal{W}} - \boldsymbol{\omega}^{\mathcal{J}} \times \mathbf{v}^{\mathcal{J}}, \\
\dot{\mathbf{b}}_{\omega} &= \mathbf{w}_{\mathrm{b}_{\omega}}, \\
\dot{\mathbf{b}}_{\mathrm{a}} &= \mathbf{w}_{\mathrm{b}_{\mathrm{a}}},
\end{aligned} \tag{5}$$

where $\mathbf{C}_{\mathcal{W}\mathcal{J}}$ denotes the rotation matrix from $\mathcal{F}_{\mathcal{W}}$ to $\mathcal{F}_{\mathcal{J}}$, $\boldsymbol{\Omega}(.)$ indicates the matrix that maps the angular velocity from Euler space to quaternion space, $\mathbf{g}^{\mathcal{W}}$ stand for the gravitational acceleration, $\mathbf{w}_{\mathrm{b}_{\omega}}$ and $\mathbf{w}_{\mathrm{b}_{\mathrm{a}}}$ are the Gaussian random walk of the gyroscope and accelerometer biases. The actual IMU acceleration $\mathbf{a}^{\mathcal{J}}$ and angular velocity $\boldsymbol{\omega}^{\mathcal{J}}$ are related to their measurement value, $\tilde{\mathbf{a}}$ and $\tilde{\boldsymbol{\omega}}$, as following:

$$\begin{aligned}
\mathbf{a}^{\mathcal{J}} &= \tilde{\mathbf{a}} - \mathbf{b}_{\mathrm{a}} + \mathbf{w}_{\mathrm{a}}, \\
\boldsymbol{\omega}^{\mathcal{J}} &= \tilde{\boldsymbol{\omega}} - \mathbf{b}_{\omega} + \mathbf{w}_{\omega},
\end{aligned} \tag{6}$$

where $\mathbf{w}_{\omega}$ and $\mathbf{w}_{\mathrm{a}}$ stand for the Gaussian white noise of measurements.

Next, the update step is conducted by minimizing a cost function that combines the reprojection error $\mathbf{e}_{\mathrm{repj}}$ and the IMU error $\mathbf{e}_{\mathrm{imu}}$:

$$J \coloneqq \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j\in\mathcal{J}(i,k)} \mathbf{e}_{\mathrm{repj}}^{i,j,k\,T}\mathbf{W}_{\mathrm{repj}}^{i,j,k}\mathbf{e}_{\mathrm{repj}}^{i,j,k} + \sum_{k=1}^{K-1} \mathbf{e}_{\mathrm{imu}}^{k\,T}\mathbf{W}_{\mathrm{imu}}^{k}\mathbf{e}_{\mathrm{imu}}^{k}, \tag{7}$$

where i is the camera index, k is the image frame index, and j is the landmark index in the set $\mathcal{J}(i,k)$. $\mathbf{W}_{\mathrm{repj}}$ and $\mathbf{W}_{\mathrm{imu}}$ denote the information matrices derived from system uncertainty. The IMU error is the difference between the predicted state $\hat{\mathbf{x}}^{k+1}$ and the actual state $\mathbf{x}^{k+1}$:

$$\mathbf{e}_{\mathrm{imu}}^{k} = \hat{\mathbf{x}}^{k+1} - \mathbf{x}^{k+1}, \tag{8}$$

Moreover, the reprojection error is established as:

$$\mathbf{e}_{\mathrm{repj}}^{i,j,k} = \mathbf{z}^{i,j,k} - \mathbf{h}_i\left(\mathbf{T}_{\mathcal{C}_i\mathcal{J}}^{k}\mathbf{T}_{\mathcal{W}\mathcal{J}}^{k}\mathbf{l}^{\mathcal{W},j}\right), \tag{9}$$

where $\mathbf{z}^{i,j,k}$ is the 2D keypoint location in the image and $\mathbf{l}^{\mathcal{W},j}$ is the 3D landmark position. The landmarks are reprojected to 2D image by the calibrated camera model $\mathbf{h}_i(.)$, the calibrated camera-IMU transformation $\mathbf{T}_{\mathcal{C}_i\mathcal{J}}^{k}$, and the current IMU pose $\mathbf{T}_{\mathcal{W}\mathcal{J}}^{k}$.

Before the optimization, landmark's locations must be initialized by the front-end, which is an image processing program. First, image keypoints are extracted using the BRISK feature detector [16]. Given the propagated state from IMU measurement, the front-end considers the previous landmarks that could be visible in the current

frame for 3D-2D matching with current keypoints. Next, the 2D-2D matching and triangulation is performed across all images in the process window. Potential landmarks will go through a RANSAC step [17] to remove outliers and only qualified landmarks are transferred to the optimization. New keyframe is selected by a heuristic rule that helps the landmarks to be viewed from different angles. Finally, the process window is marginalized out to ensure a limited number of most current frames and previous keyframes. The keypoint matching step is demonstrated in Fig. 3.

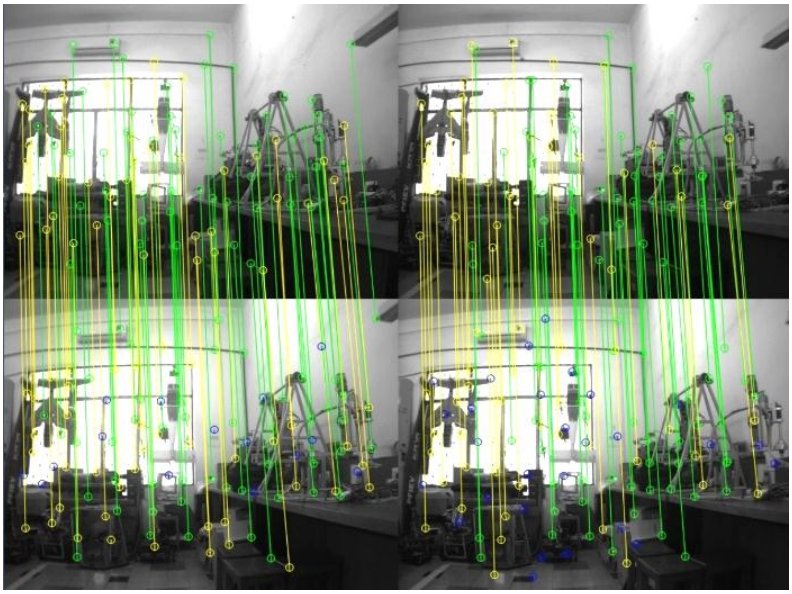


Figure 3. Example of the keypoint matching between the most recent stereo keyframe (above) and current stereo frame (below). Green indicates 3D-2D matching, yellow show 2D-2D matching, and blue marks the stereo matching only.

IV. Experiments

A. System Setup

The 3D model of our experimental system is visualized in Fig. 4. The system consists of a Point Grey Bumblebee2 stereo camera, an ADIS16488 factory-calibrated IMU, and an IEI NANO-HM650 Core-i5 embedded computer. The camera streams 2x640x480 pixels grayscale images to the computer at 20 Hz. With a rate of 500 Hz, the IMU broadcasts raw inertial measurements which has noise characteristics as specified in Table I. The processing board of IMU also sends a set of well-estimated rotation angles, which will be used for evaluating the developed system. The IMU and camera are synchronized by a custom-made device, which adjusts the camera trigger pulse (emitted by IMU) using the proposed synchronization method. An embedded Wi-Fi router is installed for remote monitoring. The system can work standalone since the embedded computer is able to process the VIO algorithm in real-time. Moreover, all devices are compacted in an 18x16.5x12 cm box, making it convenient to install on mobile robots.

TABLE I. IMU Noise Characteristics

Accelerometers			Gyroscopes		
σ_{a}	2×10^{-3}	$m/s^2/\sqrt{Hz}$	σ_{ω}	5×10^{-3}	$rad/s/\sqrt{Hz}$
$\sigma_{b_{\mathrm{a}}}$	2×10^{-5}	$m/s^3/\sqrt{Hz}$	$\sigma_{b_{\omega}}$	5×10^{-5}	$rad/s^2/\sqrt{Hz}$

The computer software was designed with 3 layers: the driver layer that communicated with IMU, camera and other devices, the VIO layer that runs the VIO algorithm, and the graphic layer that displays the estimated robot motion in 3D. We programmed the software on Robot Operating System (ROS) for a concrete platform while using C++ language for high speed performance. The ROS-node feature was exploited for efficient data transfer among layers, saving the developing time.

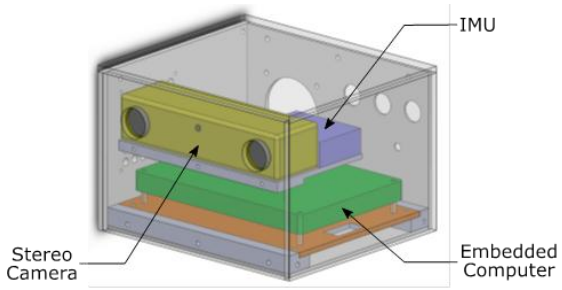


Figure 4. Visualization of the developed visual-inertial system.

In all of the experiments below, we configured the algorithm to keep 4 keyframes and 3 current frames in the optimization window, as well as detect maximum 150 keypoints per image. This light-weight configuration was chosen to ensure the real-time operation on our embedded computer, while keeping the result as good as possible.

B. Slider Experiment

In this experiment, we tested the VIO system on a single forward-backward motion generated by a custom-

made linear slider, as depicted in Fig. 5. The system was mounted on top of the slider moving part which was pulled by a servo motor. The motion was created in form of a sine wave at 0.3 Hz on the trajectory of 0.6 m with a maximum speed of 0.56 m/s. The motor's encoder was read synchronously with the VIO system and would be used as ground-truth position to evaluate the estimation result. The slider was placed indoor under artificial light sources. To examine the system ability to reject motions happened in the environment, one person has stayed in the camera view and performed casual actions across the dataset. The dataset lasted about 6 minutes and crossed the distance of 120 m.

Figure 5. The slider used for the experiment.

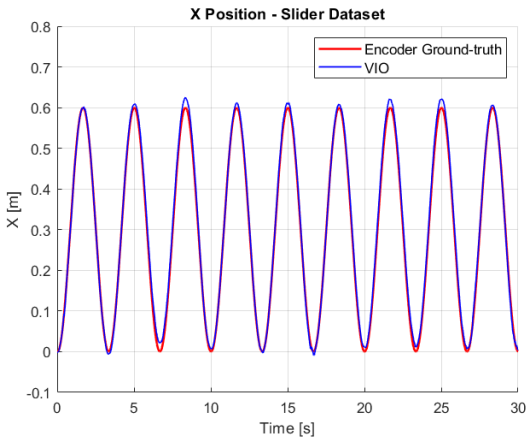


Figure 6. Estimated X position in comparison with encoder ground-truth for the Slider Dataset.

We show the system estimated position in Fig. 6 and Fig. 7. Note that the system coordinates had been aligned with the slider motion on the X axis, so that the Y and Z axes only contained vibrations. The estimated result was consistent with real motion, although slight differences can be observed on transition points of direction. The translation and orientation errors were measured statistically and expressed in relation to the traveled distance. To obtain the error statistics of each traveled distance, we first determined a set of sub-paths by sliding the respective distance window throughout the entire path, then calculated the accumulated error between each estimated and true sub-path. The number of samples used for each distance was 101. In particular, Fig. 8 and Fig. 9 visualize the statistic errors of VIO estimation by means of boxplots, which contain 5 quantities: maximum, minimum, median, 25th and 75th percentiles. The translation error increased with respect to the distance, but its median was maintained around 5 cm after 110 m traveled. Besides, the yaw error was mostly kept below 0.3 degree on the entire path. Note that the yaw error was not enlarged because the motion in this experiment was a pure translation.

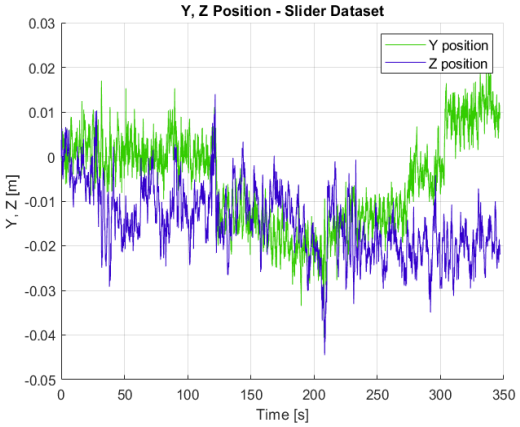


Figure 7. Estimated Y and Z position maintained around zero for the Slider Dataset.

Figure 8. Error statistics of translation error for the Slider Dataset.

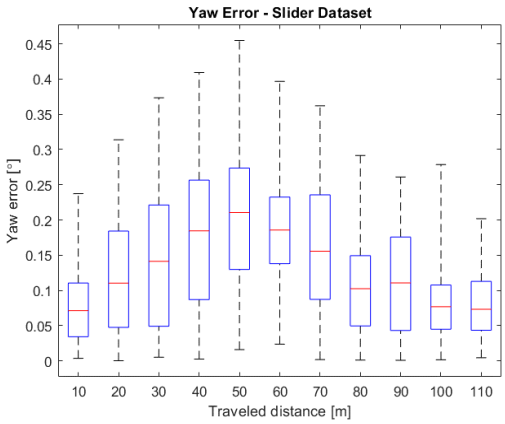


Figure 9. Error statistics of yaw error for the Slider Dataset.

C. Walking Loop Experiment

In order to evaluate the system on real motions with translation and rotation combined, we conducted the

experiment of carrying the system and walking around the room in loops. The trajectory was drawn on the floor as a 6x3 m rectangle with 0.6 m turning radius, which were tracked by the carrying person. The total path length was about 220 m with 11 loops passed in 5 minutes. Since there was no available position ground-truth data for indoors, we manually picked a starting point and calculated repeating error each time the system went back to this point. On the other hand, the orientation ground-truth was provided by the processing board of IMU with just 0.1 degrees on yaw drift over 5 minutes. We could not apply the sub-path scheme to calculate errors statistics as in the previous experiment due to the lack of position ground-truth. Instead of that, we collected the error sets by running the VIO algorithm again for 51 times on the same dataset.

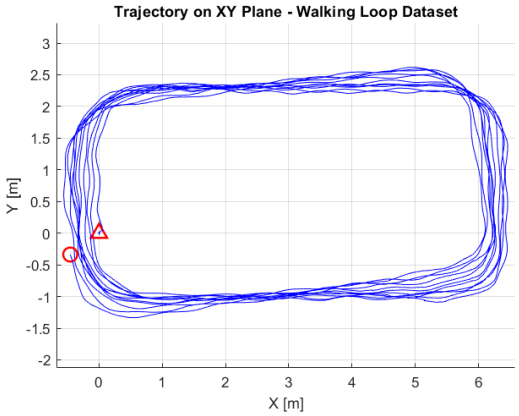


Figure 10. The trajectory of VIO system plotted on XY plane, for the Walking Loop Dataset. The red triangle and circle mark the beginning and end point of the trajectory, respectively.

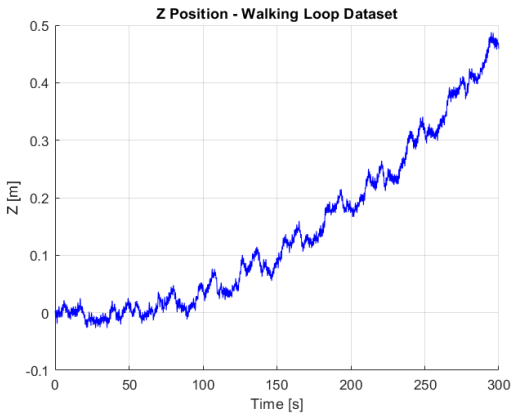


Figure 11. The Z position for the Walking Loop Dataset.

Fig. 10 shows that the estimated trajectory was very similar to the desired at the beginning, but then became further and skewer due to the accumulated error, which is clearly observed in Fig. 11. In spite of that, the errors per traveled distance were consistent, with a translation error about 0.5 % and an orientation error smaller than 0.1 °/m, starting from the 4th loop, as depicted in Fig. 12 and Fig. 13. We can further improve the system repeating error by applying a loop-closure method which will realign the robot location again whenever a closed-loop is detected. However, this approach is computational costly and might require a more powerful embedded computer to handle the current update rate.

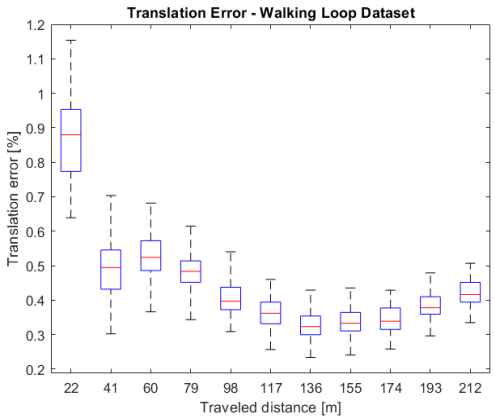


Figure 12. Translation error for the Walking Loop Dataset.

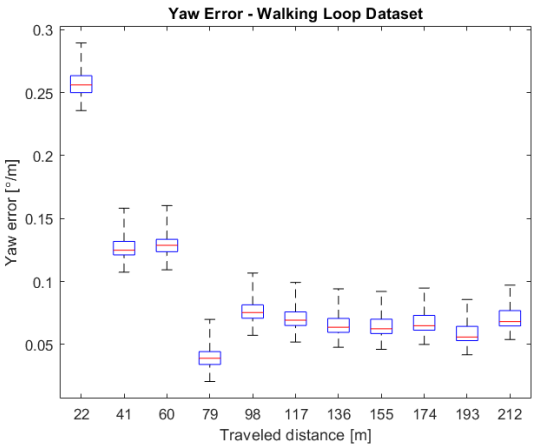


Figure 13. Yaw error for the Walking Loop Dataset.

V. Conclusion

In this paper, we have presented a visual-inertial system that tightly coupled the stereo camera and the IMU. We have proposed a timing method that is used for strict synchronization between sensors, despite the variation of the environment illumination. The system calibration has been performed through camera-camera and camera-IMU stages. The utilized algorithm has been summarized into two steps of states propagation and update. We have built a real system to examine our method. Experimental results have shown that the system performance was accurate and robust in practice. The developed VIO system can be applied on mobile robots for indoor localization.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Nguyen H. K. Tran developed the presented method, conducted the experiment, analyzed the result and wrote the manuscript. Vinh-Hao Nguyen devised the original

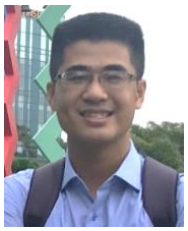
idea, supervised the project and gave critical feedback. All authors had approved the final version.

ACKNOWLEDGMENT

This research is funded by Ho Chi Minh University of Technology, VNU-HCM, under grant number BK-SDH-2020-1870473.

REFERENCES

[1] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108-117, 2006.
[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052-1067, June 2007.
[3] Z. An, L. Hao, Y. Liu, and L. Dai, "Development of mobile robot SLAM based on ROS," *International Journal of Mechanical Engineering and Robotics Research*, vol. 5, no. 1, pp. 47-51, January 2016.
[4] A Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3d reconstruction in real-time," in *Proc. of IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany, 2011.
[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 611-625, 2018.
[6] C. Forster, Z. Zhang, M. Gassner, M. Werlberger and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249-265, 2017.
[7] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *Proc. of 2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, 2015.
[8] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *Proc. of 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, 2013.
[9] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A system for autonomous flight using onboard computer vision," in *Proc. of 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011.
[10] M. Bloesch, M. Burri, Sammy Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053-1072, 2017.
[11] 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. 4, pp. 314-334, 2014.
[12] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *Proc. of 2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia , 2018.
[13] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.
[14] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
[15] S. Agarwal, K. Mierle and et. al., "Ceres Solver," 2015. [Online]. Available: ceres-solver.org. [Accessed 1 June 2019].
[16] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. of IEEE International Conference on Computer Vision*, Barcelona, Spain, 2011.
[17] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartograhpy," *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.



Nguyen H. K. Tran received B.Eng. degree in control engineering & automation from Ho Chi Minh City University of Technology, Vietnam, in 2018. He currently works as a researcher in Ho Chi Minh City University of Technology. His research interests are Vision-INS fusion and robot teleoperation.

Vinh-Hao Nguyen is presently a lecturer of electrical and electronics engineering at Ho Chi Minh City University of Technology, Vietnam. He received B.S. and M.S. degrees in electrical and electronics engineering from Ho Chi Minh City University of Technology, Vietnam, in 2001 and 2003, respectively. He received Ph.D. degree in Electrical Engineering from the University of Ulsan, Korea, in 2009. His research interests are INS/GPS positioning system, autonomous robot and adaptive control systems.