VINS-Multi: A Robust Asynchronous Multi-camera-IMU State Estimator

Luqi Wang, Yang Xu and Shaojie Shen

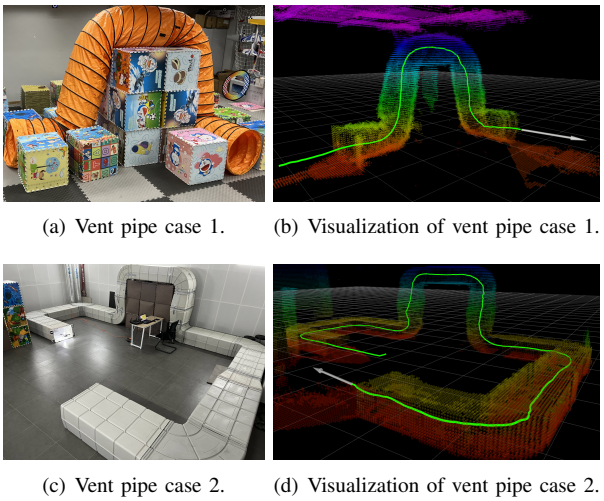
***Abstract*— State estimation is a critical foundational module in robotics applications, where robustness and performance are paramount. Although in recent years, many works have been focusing on improving one of the most widely adopted state estimation methods, visual inertial odometry (VIO), by incorporating multiple cameras, these efforts predominantly address synchronous camera systems. Asynchronous cameras, which offer simpler hardware configurations and enhanced resilience, have been largely overlooked. To fill this gap, this paper presents VINS-Multi, a novel multi-camera-IMU state estimator for asynchronous cameras. The estimator comprises parallel front ends, a front end coordinator, and a back end optimization module capable of handling asynchronous input frames. It utilizes the frames effectively through a dynamic feature number allocation and a frame priority coordination strategy. The proposed estimator is integrated into a customized quadrotor platform and tested in multiple realistic and challenging scenarios to validate its practicality. Additionally, comprehensive benchmark results are provided to showcase the robustness and superior performance of the proposed estimator.**

I. Introduction

State estimation serves as a fundamental component within robotic applications, underpinning higher-level tasks with essential support. Therefore, the robustness and concision of the system are required during practice. A lot of recent research has concentrated on enhancing the performance and resilience of visual-inertial odometry (VIO), one of the predominant methods utilized for state estimation [1]–[3].

An intuitive and effective scheme for enhancing VIO involves the incorporation of additional cameras to cover different directions, thereby acquiring more information from the surrounding environment [4]–[6]. Most of the works primarily investigated the use of synchronous cameras, which necessitate auxiliary hardware triggering mechanisms. However, practically, numerous cameras lack the hardware synchronization capability and systems reliant on synchronous cameras often succumb to malfunctions induced by faulty trigger signals or partial camera failures, which stem from design or technical issues. Hence, during some real applications in demanding environments, for instance wilderness and construction sites, deploying multiple asynchronous cameras can be less cumbersome and offer superior robustness against failures. Although [7] has explored the use of multiple asynchronous cameras on a driving dataset, the approach combines asynchronous frames as multi-frame batches and approximates the states by a B-spline trajectory for interpolation, which typically loses degrees of freedom. To circumvent this limitation, the incorporation of an inertial measurement unit (IMU), a commonly adopted sensor in robotics, can be a simple and effective solution. As a result, we propose the VINS-Multi, an asynchronous multi-camera-IMU state estimator developed from our previous work [3, 8].

All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. {lwangax, yxuew}@connect.ust.hk, eeshaojie@ust.hk.



(a) Vent pipe case 1. (b) Visualization of vent pipe case 1.

(c) Vent pipe case 2. (d) Visualization of vent pipe case 2.

Fig. 1. The deployment of VINS-Multi in aerial vent pipe inspection scenarios. The green lines are the estimated trajectory and the white arrows indicate the estimated odometries. The maps are constructed during the flights and the color code indicates the height.

The contributions of this paper are the following:

1) We design a novel dynamic feature number allocation alongside a frame priority coordination strategy to efficiently handle the asynchronous frame inputs.
2) We combine a parallel front end and a front end coordinator based on the proposed strategy, as well as a sliding window optimization module into a robust multi-camera-IMU state estimator that can accommodate multiple asynchronous cameras of mixed types.
3) The estimator is integrated into a quadrotor platform for extensive experiments in realistic and challenging scenarios, and the benchmark results are presented.

II. Methodology

The entire system architecture is depicted in Fig. 2. The images, accompanied by depth if using RGBD cameras, are sent to the visual front ends and the processed measurements are subsequently coordinated by the front end coordinator. The coordinator dynamically redistributes the number of features processed by each front end based on the collected features, as well as determines whether the measurements are

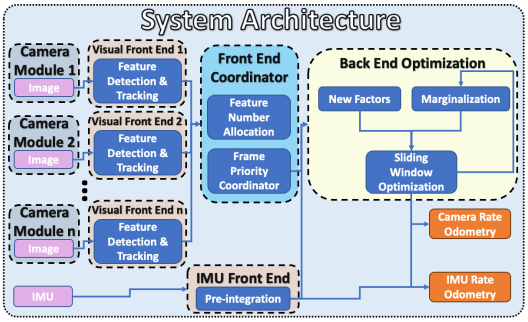


Fig. 2. The system architecture of the proposed state estimator. Note that the camera modules can be of mixed types.

forwarded to the sliding window in the back end optimization module according to the priority. The back end obtains the visual and depth measurements, as well as the pre-integration results from the raw IMU input to output the camera and IMU rate odometries at 30 and 500 Hz after optimization.

A. Front End

The visual front end adheres to a standard procedure which includes feature detection, tracking and outlier rejection, similar to the methodologies established by VINS-Mono [3] and VINS-Fusion [8], while the front ends are running in parallel with each front end thread handling a separate camera module. The extracted visual measurements, along with the corresponding depth measurements (when using RGBD cameras), are firstly sent to the front end coordinator preceding the back end optimization. The IMU front end performs pre-integration and directly outputs the high rate odometry based on the latest optimization results.

B. Front End Coordinator

1) Feature Number Allocation: Considering the different feature qualities caused by the varying captured scenes from cameras facing distinct directions , it is inefficient to treat the cameras equally. To make effective use of the computation resources, the maximum extracted feature number from images of each camera is dynamically allocated according to the scene. In particular, given the total maximum feature number FN we allocate fn_i features for camera i according to the tracked feature number from the last frame tfn_i:

$$fn_i = \frac{tfn_i}{\sum_{i=0}^{N} tfn_i} FN. \tag{1}$$

Note that by dynamically adjusting the maximum feature number of each camera, the feature tracking rate will converge to the same across the cameras, meaning that when a front end has high feature tracking rate, larger maximum feature number will be allocated to it.

2) Frame Priority Coordinator: Since in some scenarios, not all the frames are necessary to be inserted into the sliding window for optimization, for instance when a camera is obstructed, or the time interval between two frames is excessive compared with others, a frame priority coordinator is required to handle the priority among the cameras so as to make efficient use of the computation resources. The priority

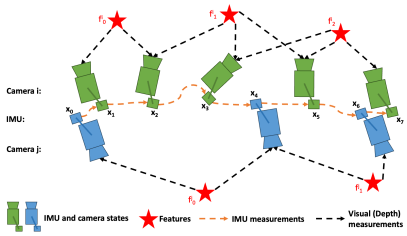


Fig. 3. A schematic of the sliding window incorporating IMU data and visual (including depth) data for optimization.

of a frame of camera i depends on both the feature and time interval δt_i since the last frame from camera i was received. The feature priority P_{fi} is determined by the ratio between the current maximum feature number of camera i fn_i and the total maximum feature number FN :

$$P_{fi} = \frac{fn_i}{FN}, \tag{2}$$

and the frame time interval priority P_{ti} is formulated as:

$$P_{ti} = e^{-k\delta t_i}, \tag{3}$$

where k is a positive constant. The coordinator waits for the frame with either the highest feature priority to maintain consistent feature tracking quality or the highest time priority to preclude camera failures and forwards the corresponding measurements to the back end for optimization.

C. Back End Optimization

In the back end optimization, we extend the formulation from our previous work [3, 8] to accommodate multiple asynchronous cameras. As depicted in Fig. 3, the accepted frames from the cameras are chronologically ordered inside the sliding window according to their time stamps for optimization with a full state vector $\mathcal{X}$ defined as:

$$\begin{aligned}
\mathcal{X} &= \left[\mathbf{x}_0, \mathbf{x}_1, \cdots \mathbf{x}_n, \lambda_0, \lambda_1, \cdots \lambda_l, \mathbf{x}_{c_0}, \mathbf{x}_{c_1}, \cdots, \mathbf{x}_{c_{N-1}}\right] \\
\mathbf{x}_i &= \left[\mathbf{p}_i^w, \mathbf{v}_i^w, \mathbf{R}_i^w, \mathbf{b}_a, \mathbf{b}_g\right], i \in [0, n] \\
\mathbf{x}_k^c &= \left[\mathbf{p}_{c_k}^b, \mathbf{R}_{c_k}^b, t_{d_k}\right], k \in [0, N-1],
\end{aligned} \tag{4}$$

where $\mathbf{x}_i$ is the i-th IMU and camera states consists of position, velocity, rotation as well as acceleration and gyroscope biases, λ_j denotes the inverse depth of the j-th feature upon its initial observation, and $\mathbf{x}_k^c$ encompasses of the extrinsic translation, rotation and the time offset relative to the IMU of the k-th camera. The optimization objective is formulated similarly as the combination of the prior factor, the IMU propagation factor and the visual (depth) factor:

$$\begin{aligned}
\min_{\mathcal{X}} \{ & \underbrace{\left\| \mathbf{e}_p - \mathbf{H}_p \mathcal{X} \right\|^2}_{\text{prior factor}} + \underbrace{\sum_{k \in \mathcal{B}} \left\| \mathbf{e}_{\mathcal{B}} \left(\mathbf{z}_{k+1}^k, \mathcal{X} \right) \right\|_{\mathbf{P}_{k+1}^k}^2}_{\text{IMU propagation factor}} \\
& + \underbrace{\sum_{(l,j) \in \mathcal{C}} \left\| \mathbf{e}_{\mathcal{C}} \left(\mathbf{z}_l^j, \mathcal{X} \right) \right\|_{\mathbf{P}_l^j}^2}_{\text{visual (depth) factor}} \}.
\end{aligned} \tag{5}$$

The detailed formulation can be found in [3], while a depth (re-projection) error is integrated to facilitate RGBD cameras. The prior factor originates from the marginalization of

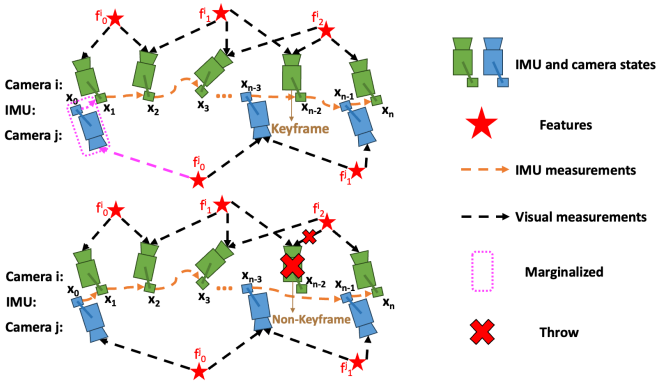
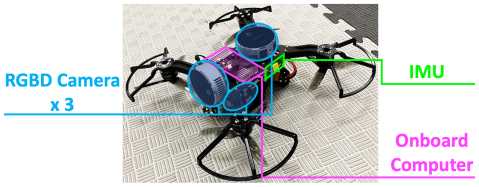


Fig. 4. An illustration of the marginalization strategy.



(a) The custom-built quadrotor platform equipped with three RGBD cameras.

(b) Replace the top RGBD camera by a stereo camera.

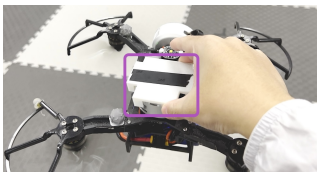
Fig. 5. The quadrotor deployed in experiments. The system is compatible with mixed types of monocular, RGBD, and stereo camera modules.

the frames, which follows a different strategy from preceding research, as illustrated in Fig. 4. Upon the arrival of a new frame from camera i, the last frame from camera i is checked. If it is a keyframe, the most outdated frame in the sliding window is marginalized, regardless of the originating camera. Otherwise, the last frame from camera i is thrown. This strategy is able to handle uneven frames from multiple cameras and cope with the malfunction of certain cameras.

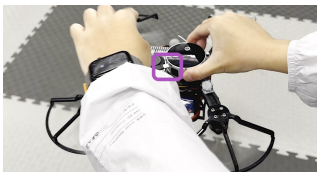
III. Experiment and Results

A. Experiment Setup

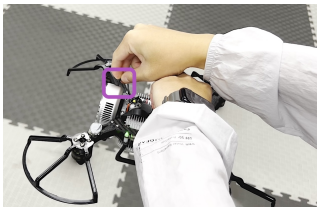
The evaluation of the proposed state estimator is conducted using a custom-built quadrotor platform depicted in Fig. 5. The quadrotor is equipped with three Intel Realsense L515 RGBD cameras, while in particular experiments, the top RGBD camera is replaced with an Intel Realsense D435 stereo camera to verify the performance on mixed types of cameras. The data from a BMI088 IMU embedded in the NxtPX4 flight controller is adopted throughout the test flights. All the computations are performed on an Nvidia Jetson Orin NX onboard computer. The experiments are conducted in four scenarios: camera failure and recovery shown in Fig. 6, wall inspection shown in Fig. 7(a), flights employing mixed types of camera modules on the quadrotor in Fig. 5(b), and vent pipe inspection shown in Fig. 1.



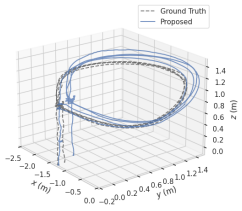
(a) Cover the top camera with a lid.



(b) Unplug the top camera.



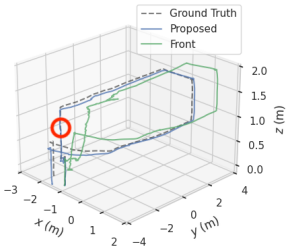
(c) Plug the front camera.



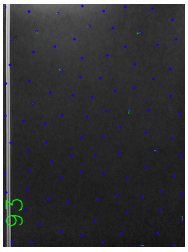
(d) Trajectory result.

Fig. 6. The procedure and comparison of the trajectory results against the ground truth during the camera failure and recovery scenario.

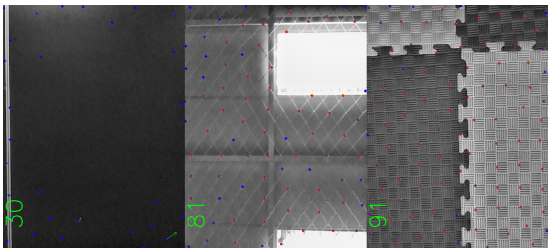
(a) Snapshot of the scenario with scarce features for stable tracking in the marked areas.



(b) Trajectory result.



(c) Features from the front camera.



(d) Features from three cameras using the proposed method.

Fig. 7. The scene and the comparison of the trajectory results of the proposed method using three cameras and using solely the front camera with the ground truth and the features extracted at the circled position from the front, top, and bottom images in the wall inspection scenario. The quantity of allocated features is indicated on the lower left of each image, with the feature point color gradient representing tracking duration from blue (shortest) to red (longest).

B. Result and Analysis

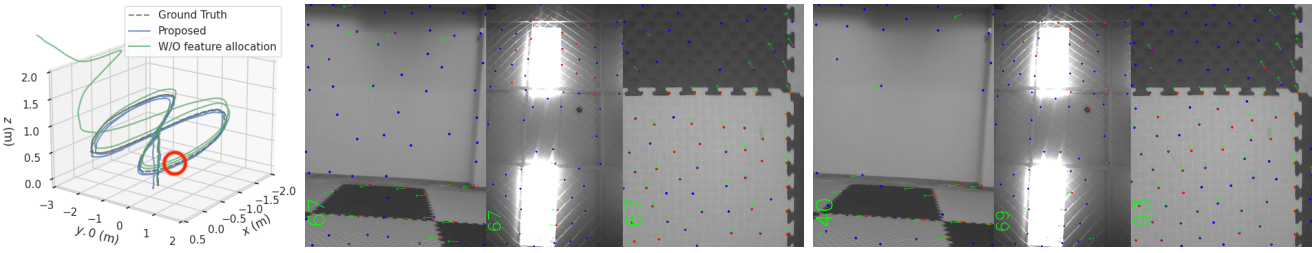
During the failure and recovery scenario, the quadrotor first takes off with only its top and bottom cameras. The robustness of the system is tested through a sequence of procedures consisting of covering the top camera with a lid, removing the lid, unplugging the top camera, and finally

TABLE I

BENCHMARK RESULTS ON THE ESTIMATED TRAJECTORIES

Seq.	Multiple cameras				Single camera					
	Proposed		W/O feature allocation		Front		Top		Down	
	ATE(m)	RPE(m)	ATE(m)	RPE(m)	APE(m)	RPE(m)	ATE(m)	RPE(m)	ATE(m)	RPE(m)
failure & recovery	**0.07711**	**0.00042**	2.45813	0.00160	×	×	×	×	×	×
wall inspection	**0.08488**	**0.00070**	0.28015	0.00077	0.44229	0.00284	3.57276	0.00065	×	×
mixed camera types	**0.07596**	**0.00076**	2.02272	0.00306	0.17287	0.00094	0.11617	0.00102	×	×

×: fail. **Bold**: best results.



(a) Trajectory results. (b) The features without feature allocation. (c) The features using the proposed method.

Fig. 8. The comparison of the trajectory results of the proposed method and the method without feature allocation with the ground truth and the features extracted at the circled position on the front, top, and bottom images in the mixed camera types configuration scenario. The quantity of allocated features is indicated on the lower left of each image, with the feature point color gradient representing tracking duration from blue (shortest) to red (longest).

plugging the front camera during the flight. As evidenced by Fig. 6(d), the proposed system maintains functionality amid camera failures, exhibiting decent estimation errors, and the recovered front camera can be successfully added into the estimation, demonstrating the superior robustness, which is unattainable by adopting single cameras.

The trajectory result and comparison with a solitary front camera during the wall inspection scenario is shown in Fig. 7(b). When the single front camera faces the black wall during take-off, we notice a conspicuous trajectory drift of the estimated trajectory due to the lack of stable tracking features (see Fig. 7(c)), while the proposed method using multiple cameras is capable of handling this case with stable feature tracking using the top and bottom cameras (as shown in Fig. 7(d)).

The trajectory result and comparison with the ablation of dynamic feature allocation in flights with mixed camera types configuration is shown in Fig. 8(a). The method lacking dynamic feature allocation is prone to substantial drift during rapid yaw movements, particularly apparent when wasting unnecessary features on feature-poor walls (see Fig. 8(b)). Conversely, the proposed method copes effectively with this challenging scenario buttressed by its dynamic feature allocation strategy, as shown in Fig. 8(c).

The efficacy and robustness of the proposed method are further corroborated during the complex task of aerial vent pipe inspection, as shown in Fig. 1, reiterating its potential for various practical applications.

The benchmark trajectory estimation results for scenarios involving failure and recovery, wall inspection, and mixed camera types are provided in Tab. I. Examination of the table reveals that the proposed method outperforms single-camera approaches in terms of accuracy across all three scenarios, illustrating the advantage of employing multiple cameras. Additionally, the proposed method surpasses the performance of the ablation of dynamic feature allocation version in terms of accuracy, further proving the efficacy of the strategy.

IV. CONCLUSION

In this paper, we propose a robust feature-aware multi-camera-IMU state estimator for asynchronous camera modules. The estimator encompasses parallel front ends, a front end coordinator and a back end optimization. It makes efficient use of input frames by implementing a dynamic feature number allocation and a frame priority coordination strategy. The superior robustness and performance of the estimator is validated through real-flight experiments in various challenging scenarios.

REFERENCES

[1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Trans. Robot. (TRO)*, vol. 37, no. 6, pp. 1874–1890, 2021.

[2] 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.

[3] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot. (TRO)*, vol. 34, no. 4, pp. 1004–1020, 2018.

[4] L. Zhang, D. Wisth, M. Camurri, and M. Fallon, "Balancing the budget: Feature selection and tracking for multi-camera visual-inertial odometry," *IEEE Robot. Autom. Ltr. (RA-L)*, vol. 7, no. 2, pp. 1182–1189, 2021.

[5] J. Jaekel, J. G. Mangelson, S. Scherer, and M. Kaess, "A robust multi-stereo visual-inertial odometry pipeline," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS)*. IEEE, 2020, pp. 4623–4630.

[6] Y. He, H. Yu, W. Yang, and S. Scherer, "Towards robust visual-inertial odometry with multiple non-overlapping monocular cameras," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS)*. IEEE, 2022, pp. 9452–9458.

[7] A. J. Yang, C. Cui, I. A. Bârsan, R. Urtasun, and S. Wang, "Asynchronous multi-view slam," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA)*. IEEE, 2021, pp. 5669–5676.

[8] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," 2019.