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].

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...
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 odometries 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 $tfni$:

$$fn_i = \frac{tfni}{\sum_{i=0}^{N} tfni} FN. \quad (1)$$

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 of a frame of camera $i$ depends on both the feature and time interval $\delta t_i$ 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 $FN$:

$$P_{fi} = \frac{fn_i}{FN}, \quad (2)$$

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

$$P_{ti} = e^{-k\delta t_i}, \quad (3)$$

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 $\mathbf{X}$ defined as:

$$\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_N, \lambda_0, \lambda_1, \cdots, \lambda_l, \mathbf{x}_{c0}, \mathbf{x}_{c1}, \cdots, \mathbf{x}_{cN-1}]$$

$$\mathbf{x}_i = [\mathbf{p}_i^w, \mathbf{v}_i^w, \mathbf{R}_i^w, \mathbf{b}_a, \mathbf{b}_g], i \in [0, n]$$

$$\mathbf{x}_k = [\mathbf{p}_k^b, \mathbf{R}_k^b, \mathbf{t}_k], k \in [0, N - 1], \quad (4)$$

where $\mathbf{x}_i$ is the $i$-th IMU and camera states consists of position, velocity, rotation as well as accelerometer and gyroscope biases, $\lambda_j$ 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:

$$\min_{\mathbf{X}} \left\{ \| \mathbf{e}_p - \mathbf{H}_p \mathbf{X} \|^2 + \sum_{k \in B} \| \mathbf{e}_B (\mathbf{z}_{k+1}, \mathbf{X}) \|^2 \mathbf{P}_{k+1} \right\} + \sum_{(l,j) \in C} \| \mathbf{e}_C (\mathbf{z}_l^j, \mathbf{X}) \|^2 \mathbf{P}_l \right\}. \quad (5)$$

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
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.

#### 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
Additionally, the proposed method surpasses the performance approaches in terms of accuracy across all three scenarios, revealing that the proposed method outperforms single-camera camera types are provided in Tab. I. Examination of the table involving failure and recovery, wall inspection, and mixed camera types configuration is shown in Fig. 8(a). The method lacking of dynamic feature allocation in flights with mixed camera types is indicated on the lower left of each image, with the feature point color gradient representing tracking duration from blue (shortest) to red (longest).

Conversely, the proposed method copes effectively with unnecessary features on feature-poor walls (see Fig. 8(b)). The trajectory result and comparison with the ablation of dynamic feature allocation version in terms of accuracy, further proving the efficacy of the strategy.

In this paper, we propose a robust feature-aware multicamera-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


