VINS-Stereo for the FPV Drone Racing VIO Competition 2020

He Zhang and Cang Ye

I. System Overview

We developed a new visual-inertial SLAM method, called VINS-Stereo, which extends the VINS-Mono [1] to utilize the stereo input in the state estimation module. Specifically, we rewrote the initialization function and adjusted the optimization process. With stereo input, the visual scale can be accurately computed, so the initialization process is simplified and yet improved—estimating fewer state variables with an accurate prior scale. Before the optimization, the stereo measurements of a visual feature are first preprocessed by the geometric correction approach [2], and then used to compute the feature's depth by stereo triangulation. The geometric correction step computes a small perturbation to update the visual features' locations on the stereo images. The updated locations will satisfy the epipolar geometry between the left and the right images of the stereo camera. After that, the updated features are used to construct the visual constraints in the optimization process for state estimation.

Besides the state estimation, we also rewrote the feature tracking module to handle the stereo input. Specifically, we use a two-way tracking strategy to track features between stereo frames. We denote the left image and the right image of the k^{th} stereo frame as F_k^l and F_k^r, respectively. For any visual feature p on F_{k-1}^l, it is tracked to p' on F_k^l by using the KLT algorithm [3]. Next, we reverse this process by tracking p' back to p'' on F_{k-1}^l. If the distance between p'' and p is smaller than a threshold (0.5 pixels), (p, p') is a potential feature match. (p, p') becomes a good match if it also passes the outlier rejection, which is conducted by using RANSAC with a fundamental matrix model. If the number of good matches is smaller than a threshold (300), we extract new visual features [4] from F_k^l for future feature tracking. By using the above process—two-way tracking strategy and outlier rejection, we also find the good feature correspondences (q, q') between F_k^l and F_k^r as well. The visual features (p') on F_k^l and (q') on F_k^r in good matches are sent to the state estimation module.

The other two modules in VINS-Mono, pose graph optimization and loop closure detection, are included in VINS-Stereo without any modification.

In summary, VINS-Stereo is an optimization-based method, which leverages the stereo-inertial data and applies a sliding window based bundle adjustment (BA) to fuse these sensor data for state estimation. It has a loop closure module, which adds the loop constraint and the pose of the loop-closure frame into the BA process to reduce pose error. In addition, it does not use information from the future to predict the pose at a given time.

II. Timing

We used a desktop, Dell Inspiron 5680, to run the VINS-Stereo with the FPV data sequences. It has 16G RAM and an Intel Core i5-8400 CPU with 6 Cores operating at 2.8 GHz. The program runs in Ubuntu 16.04 without GPU acceleration. The data are processed in four parallel threads: feature tracking, state estimation, pose graph optimization, and loop closure detection. These are implemented in each core of the CPU. We measured the per-frame and the total time cost by each module for each data sequence, and their results are tabulated in Table I and II, respectively. For per-frame processing, the state estimation is the most time-consuming module (~50.76 ms) followed by feature tracking (~33.94 ms), loop detection (~23.34), and pose graph optimization (~10.02 ms). Since the four modules are run in parallel, the VINS-Stereo can run at a speed of about 20 Hz. For total processing, feature tracking costs the most computation time because it processes each stereo frame by detecting and tracking features on the left and the right images. To enable real-time implementation, about every third of the feature tracking's outputs is sent to the state estimation module. Therefore, the total time for state estimation is less than that

Table I: Per-frame processing time in milliseconds (ms) cost by different modules for each data sequence

Data Sequence	State Estimation	Feature Tracking	Loop Detection	Pose Optimization
indoor_45_3	52.79	34.65	11.84	14.92
indoor_45_16	53.69	32.07	8.9	0.00
indoor_forward_11	57.46	35.15	24.2	14.34
indoor_forward_12	42.14	35.56	20.5	2.6
outdoor_forward_9	49.47	32.78	35.11	16.74
outdoor_forward_10	48.99	33.42	39.49	11.54
Mean	**50.76**	**33.94**	**23.34**	**10.02**

Table II: Total computation time in seconds (s) cost by different modules for each data sequence

Data Sequence	Data Duration [s]	State Estimation	Feature Tracking	Loop Detection	Pose Optimization	**Total Computation**
indoor_45_3	76.14	40.23	68.62	7.04	0.03	**115.92**
indoor_45_16	46.36	15.71	41.49	2.76	0.00	**59.95**
indoor_forward_11	78.63	44.06	75.24	14.86	0.01	**134.17**
indoor_forward_12	58.50	23.79	58.37	7.21	0.01	**89.68**
outdoor_forward_9	88.16	43.68	77.47	21.2	0.12	**142.47**
outdoor_forward_10	111.07	54.48	97.43	29.45	0.06	**181.42**

H. Zhang and C. Ye are with the Dept. of Computer Science, Virginia Commonwealth University, Richmond, VA, USA (hzhang8@vcu.edu)

of feature tracking. The total computation time for each data sequence is computed and displayed in the last column of Table II.

III. Parameters

We used the same parameters to process all the test sequences. The intrinsic parameters of both snapdragon cameras are fixed, as well as the extrinsic parameters between the cameras and the IMU. The rolling shutter effect is ignored whereas the temporal delays between the two cameras and the IMU are online estimated. The parameters of the IMU noise are shown in Table II.

Table III: IMU Noise and Random Walk Bias

	Noise density	Random walk
accelerometer	$0.12\ \frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	$0.002\ \frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
gyroscope	$0.02\ \frac{rad}{s}\frac{1}{\sqrt{Hz}}$	$0.00004\ \frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$

The other important parameters are listed below in Table IV.

Table IV: Parameters and Settings

Parameters	Value
Max tracking feature number	300
Min distance between two features (pixel)	25
Frequency (Hz) to publish feature tracking result	10
RANSAC threshold (pixel)	1.0
Histogram equalization	No
Optical flow search window size (pixels)	(21,21)
Reverse optical flow	Yes
Max solver time (ms)	40
Max solver iteration	8
Keyframe parallax (pixel)	10.0
Window size for BA	10

IV. Qualitative Results

We show some qualitative results to illustrate the accuracy of the proposed method. We compare the trajectories generated by VINS-Stereo with ground truth using the data sequences *indoor_forward_7* and *indoor_45_13*, which are provided by the UZH-FPV dataset. The comparison between the estimated trajectories between the ground truth are plotted in Fig. 1 and Fig. 2, respectively.

V. Conclusions and Future Work

In this report, we propose VINS-Stereo, which extends VINS-Mono by exploiting the stereo visual information. We test its performance using the data sequences for IROS 2020 UZH-FPV VIO competition. To better contribute to the community, we plan to open-source our code which will be released at https://github.com/rising-turtle/VINS-Stereo.

Due to time restrictions, many functions in the current version were developed in harsh and they can be improved in various ways. Some but not limited include 1) using the stereo visual features with a better initialization strategy [5]; 2) utilizing deep learning to detect and track visual features on stereo images, etc.

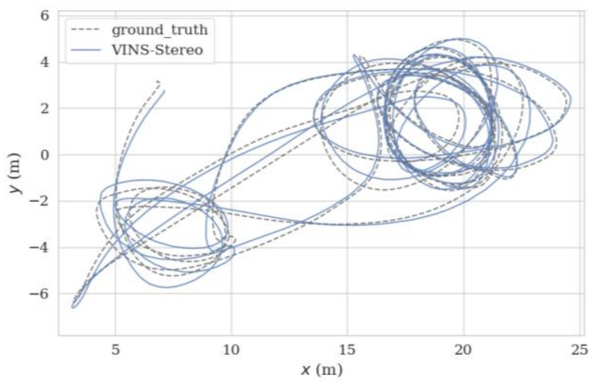

Fig.1 Trajectory estimation vs ground truth using *indoor_forward_7*

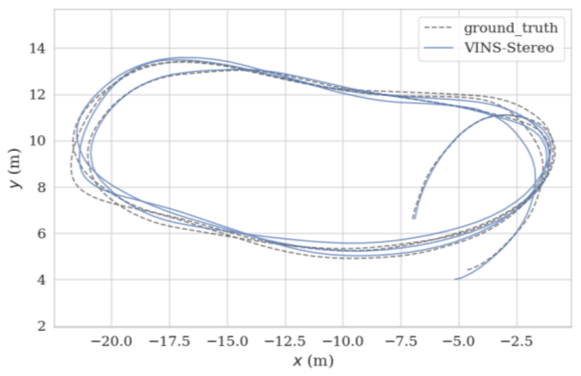

Fig.2 Trajectory estimation vs ground truth using *indoor_45_13*

References

[1] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics* vol. 34, no. 4, pp. 1004-1020, 2018.

[2] K. Kanatani, *Statistical optimization for geometric computation: theory and practice*. Courier Corporation, 2005.

[3] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in P*roc. Int. Joint Conf. Artif. Intell*., Vancouver, Canada, Aug. 1981, pp. 24–28

[4] J. Shi and C. Tomasi, “Good features to track,” in *Proc. IEEE Int. Conf. Pattern Recog*., 1994, pp. 593–600

[5] C. Campos, José MM Montiel, and Juan D. Tardós, “Inertial-Only Optimization for Visual-Inertial Initialization,” *arXiv preprint arXiv:2003.05766*, 2020.