2019 19th International Conference on Control, Automation and Systems (ICCAS 2019)
Oct. 15~18, 2019; ICC Jeju, Jeju, Korea

Comparison of Visual Inertial Odometry using FlightGoggles Simulator for UAV

Eungchang Mason Lee[1], Inhwan Wee[2], Taeyeon Kim[3], and David Hyunchul Shim[4*]

[1-4] School of Electrical Engineering,
Korea Advanced Institute of Science and Technology, Daejeon, 34141, Korea
(eungchang_mason@kaist.ac.kr[1], finani@kaist.ac.kr[2], taeyeon.k@kaist.ac.kr[3], and hcshim@kaist.ac.kr[3]) * Corresponding author

Abstract: Recently, application of Unmanned Aerial Vehicle (UAV) attract much attention of not only academic but also industrial community. To control and make the most of UAV, state estimation is mandatory and it should work accurately on Real-time on computational-limited onboard PC. Especially, for the cases that GNSS-disabled environments or indoor situation, Visual Inertial Odometry using camera and IMU sensor has been actively studied.

Generally, to research and develop the indoor navigation algorithm, high-performanced and super-expensive facilities like motion capture cameras are needed to measure the ground-truth.

In this paper, we apply state-of-art Visual Inertial Odometry algorithms for state estimation of UAVs to FlightGoggles simulator which is photorealistic and which reflects UAV's real dynamics. Moreover, we compare the performance and resource utilization of those algorithms.

Keywords: Visual Inertial Odometry, State Estimation, Indoor Navigation, Unmanned Aerial Vehicle

1. INTRODUCTION

Visual Odometry (VO) is one way of indoor or GNSS-disabled navigation for state estimation of UAV, which is essential to control it and VO can be classified into feature-based and direct method by means of application of camera. In addition, it can be specified to Visual Inertial Odometry (VIO) by usage of IMU and then VIO can be divided into loosely-coupled and tightly-coupled VIO [1].

To analyze real-time operation of VIO on resource-limited embedded board, we apply the most widely known, Robot Operating System (ROS) based, and open-source algorithms. Additionally, we compare the CPU, Memory usage, and Frame per Second (FPS) of the algorithms like ORB-SLAM2 [2], VINS-Mono [3], and ROVIO [4].

We test algorithms on photorealistic and real-like FlightGoggles Simulator [5] which is equipped with noisy sensor and is modeled with real dynamics of UAV.

Fig. 1 FlightGoggles Simulator

From section 2, we briefly explain and compare the main methods of VO and VIO. And from section 3, we introduce the environments of simulation and experiment setup. After that, section 4 covers the test result and comparison of algorithms.

2. RELATED WORK

2-1. Visual Odometry

VO algorithms, which are not using IMU sensor, are divided into feature-based and direct method.

Feature based VO extracts feature points on the image and matches them to estimate the movement of camera under Special Euclidean Group (SE(3)) which minimizes the re-projection error of the aligned feature points [6]. SE(3) consists of Rotational matrix, R and Translational vector, t that meet the optimization of following equation (1) where π represents the camera intrinsic matrix which projects 3-dimentional world coordination onto 2-dimentional image-plane pixel positions of corresponding feature points. Σ means variance here.

$$\{R,t\} = \underset{R,t}{argmin} \sum_i \left\| x^i - \pi(RX^i + t) \right\|_\Sigma^2 \qquad (1)$$

Direct method directly uses pixel intensity to estimate displacement of camera under SE(3) that minimum-optimizes the photometric error of the matching pixel of 2-dimentional pixels according to equation (2) [6]. Here

$$\{R,t\} = \underset{R,t}{argmin} \sum_i \left\| I_k(x_k^i) - I_{k-1}(x_{k-1}^i) \right\|_\sigma^2 \qquad (2)$$

I represents pixel intensity values of pixel position at x. σ means the weights for weighted sum of equation (2).

For comparison, we use ORB-SLAM2 [2], which is one of the most popular VO algorithms and great-performanced and compatible with mono, stereo, and RGB-D camera.

It is based on Bundle Adjustment (BA) to optimize re-projection and provides Loop-closure and 3D Reconstruction which are the key point of Simultaneous Localization and Mapping (SLAM).

2-2. Visual Inertial Odometry

VIO, which is using IMU sensor and camera together to estimate the states of camera, is classified into Loosely-coupled and Tightly-coupled VIO by the optimization method of sensors. Recently, tightly-coupled VIO is more popular and widely developed due to its better performance [1].

Extended Kalman Filter (EKF) based and Non-linear optimization based methods are representative among many algorithms.

Consequently, we test Direct method & Extended Kalman Filter based ROVIO [4], and Feature based & non-linear optimization based VINS-Mono [3] which are the most representative of each method.

ROVIO is extended to ROVIOLI [7], which is for map building and localization. And VINS-Mono provides online temporal calibration which solves synchronization problem of sensors.

3. SIMULATION

3.1 Environments

Selected VO and VIO algorithms and the FlightGoggles simulator are fully compatible with ROS which is great middle-ware easy to handle.

FlightGoggles simulator [5] provides stereo camera at 60 FPS, 0.32m baseline, and changeable resolution 1024x768 pixels for default. The camera pair has zero distortion and 70 degree Field of View. It also has high-rated at 960 Hz IMU sensor with accelerometer variance 0.005 and gyroscope variance 0.003. In addition, ground-truth data is available at around 1000Hz, which is essential for analysis of navigation algorithm.

Coordination of camera and IMU sensor of simulator is described at Fig.2 and the relative transformation can be simply expressed under SE(3).

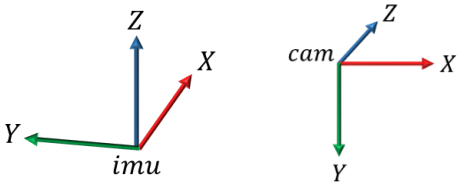


Fig. 2 3D Coordinates of IMU and Camera

3.2 Experiments

As we are using fully ROS compatible packages, we tested our algorithms on ROS using *rosbag* function which handles record and playing of data easily.

Firstly, we manually controlled aerial vehicle on the simulator to record input and sensors data as dataset. We used it as ground-truth reference dataset. Recorded dataset's specification is detailed at Table 1. The dataset is available from the shared link at conclusion section.

Table 1. Reference dataset specification

Trajectory Length	540.887 meters
Bag file duration	134.3 seconds
Mean of Linear velocity	4.120 m/s
Maximum of Linear velocity	14.810 m/s
Mean of Angular velocity	0.242 rad/s
Maximum of Angular velocity	2.333 rad/s

Total length of flight is 540.887 meters long and the maximum velocities are 14.810 m/s and 2.333 rad/s for linear and angular each. Dataset's trajectory is visualized at Fig. 3.

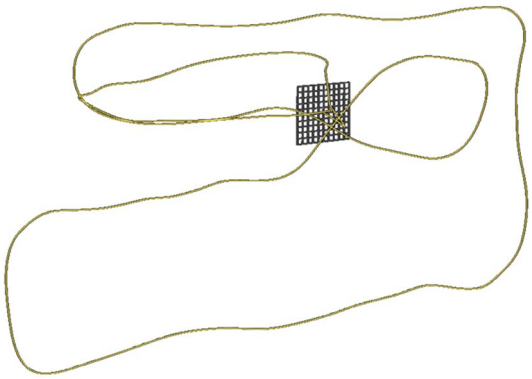


Fig. 3 Trajectory of Manual controlled UAV dataset

After recording data, we played back dataset using *rosbag* function and applied selected state estimation algorithms for comparison. Detailed configuration parameters are specified at Table 2.

Table 2. Main parameters of selected algorithms

Algorithm	Main parameters
ROVIO	30 features 6x6 Patch NLevel 4
VINS-Mono	130 features 8 threads enabled Time offset enabled
ORB-SLAM2	1200 features ThDepth 40

4. RESULT AND COMPARISON

4.1 Estimated States and Resource Usages

While running each selected VO and VIO algorithms, we measured CPU and Memory usage and checked FPS of output of algorithms. Moreover, we calculated Root Mean Square Error (RMSE) between ground-truth and estimated states as following equation (3).

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{{e_{i,x}}^2 + {e_{i,y}}^2 + {e_{i,z}}^2}\ [m] \qquad (3)$$

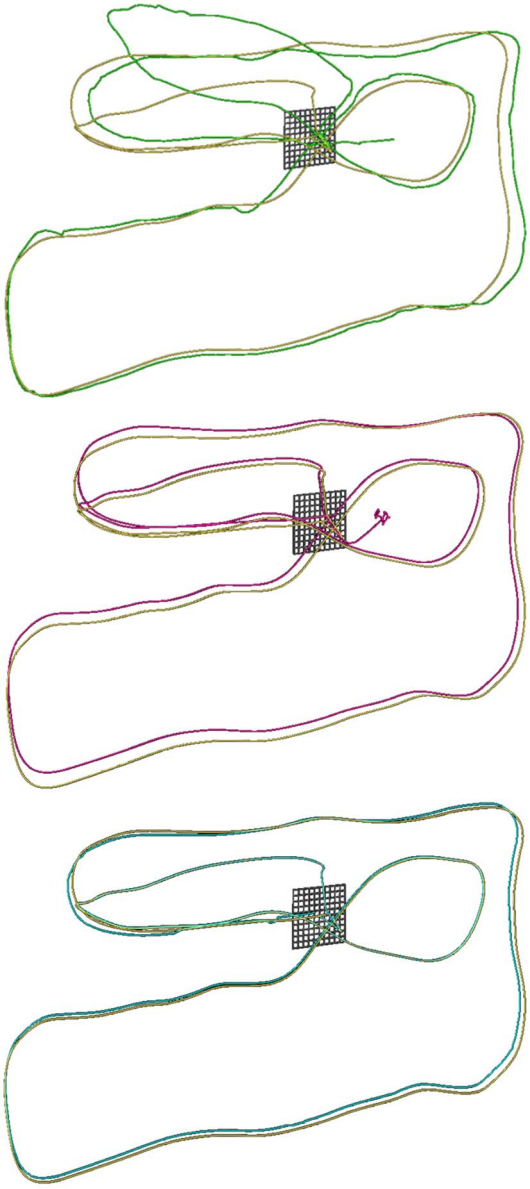


Fig. 4-a. Trajectory of ground-truth (**Fixed**, yellow) versus trajectory of estimated states of UAV (green : ROVIO, purple : VINS-Mono, turquoise : ORB-SLAM2)

Table 3 The Main results of experiment

	ROVIO	VINS-Mono	ORB-SLAM2
RMSE	6.0585m	1.7215m	0.6158m
Avg. FPS	29.893	14.9	13.834
Avg. CPU Usage	46.3%	142.7%	194.2%
Max CPU Usage	71.0%	183.0%	315.6%
Min CPU Usage	34.7%	116.8%	142.0%

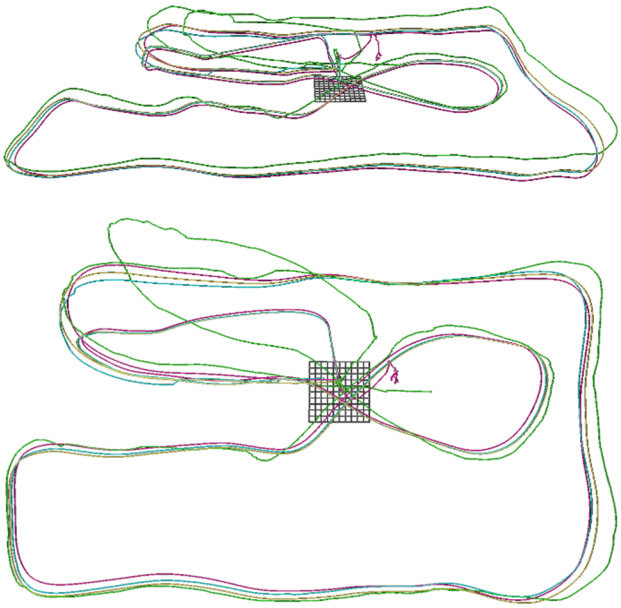
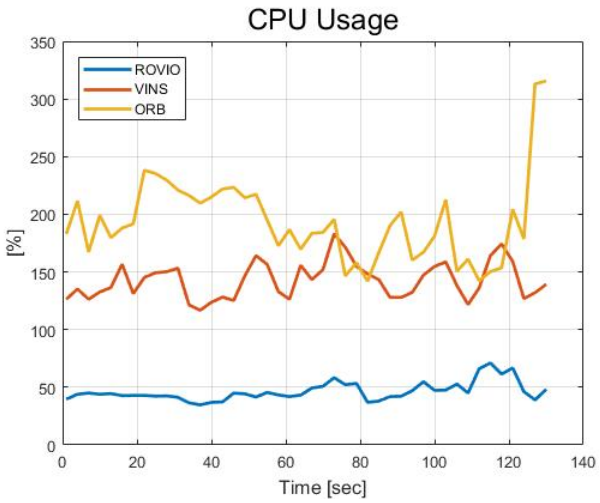
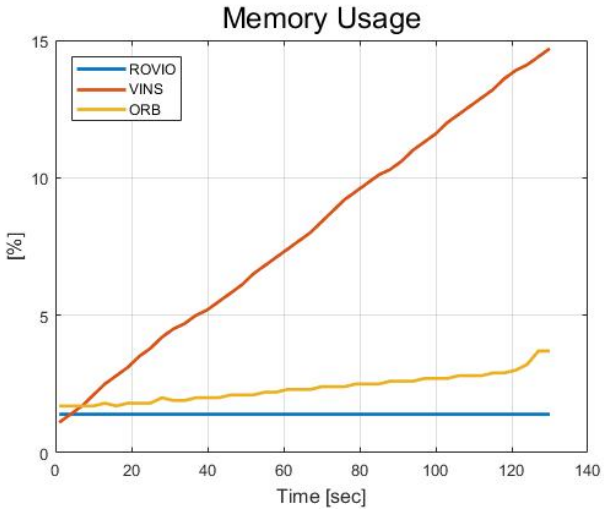


Fig. 4-b. All trajectories for comparison together – same color set as Fig. 4-a.





. Fig. 5. Resource Usage of selected algorithms

4.2 Analysis

The main results of comparison can be seen from Table.3 and Fig. 4, and Fig. 5.

As shown above, ORB-SLAM2 uses CPU the most but showed the best accuracy even without IMU sensor.

However, due to tracking too many feature points, the FPS was too low to use it as real-time onboard with limited computational resource given CPU usage. Additionally, it sometimes diverges when purely rotates. Even without the much memory, it guarantees loop closure thanks to its bundle adjustment method though.

VINS-Mono solves non-linear optimization to estimate states so it uses memory the most but it was much accurate enough. Even it provides online temporal calibration so that poor calibration can be ignored and hardware synchronization is not necessary. In addition, it supports loop-closure and multi-sensor fusion. However, FPS was too low and CPU usage was also too high to use onboard.

ROVIO showed the worst RMSE but, it uses the CPU and memory the least. Thanks to its light computational complexity, it guarantees real-time. However due to its filter-based method, it diverges quite frequently.

5. CONCLUSION

In this paper, we reviewed the state-of-art Visual Inertial Odometry algorithms which are the essential for UAV to control it. Besides, we compared the resource utilization and performances of them.

As shown above, due to the situation that uses limited onboard but needs accurate, real-time states estimation, we must set the priority well.

Thanks to real-like simulator, FlightGoggles, we demonstrated that even without super-expensive facilities, we can test the state estimation algorithms and that eventually we can utilize it to develop new navigation algorithm for the future works.

Results of experiment and comparison can be seen at video clips here https://youtu.be/XMyiNlIbDXU

Manual control input and sensors dataset is shared at https://drive.google.com/open?id=15G6Gl2Qtxc0-FhU9UgSW6v-cCzuZopaT

ACKNOWLEDGEMENTS

This research was supported by a grant(19ATRP-C108186-05) from UAV Safety Technology Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

APPENDIX

Computer specifications are detailed at Table 3 below.

Table 4 Simulated Computer Specifications

CPU	Intel(R) Core(TM) i7-6700K @ 4.00GHz (8 CPUs)
GPU	NVIDIA TITAN X (Pascal)
Memory	Samsung DDR4 8GBytes RAM * 4
OS	Ubuntu 16.04 LTS 4.15.0-47-Generic Kernel
ROS	Kinetic 1.12.14

REFERENCES

[1] Y. Wu, F. Tang, and H. Li, “Image-based camera localization: an overview,” *Visual Computing for Industry, Biomedicine, and Art*, Vol. 1, No. 1, pp. 8, 2018

[2] R. Mur-Artal, and J. D. Tardós., "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, Vol. 33, No. 5, pp. 1255-1262, 2017

[3] 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

[4] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust visual inertial odometry using a direct EKF-based approach,” *IEEE/RSJ international conference on intelligent robots and systems*, pp. 298-304, 2015.

[5] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, “FlightGoggles: Photorealistic Sensor Simulation for Perception-driven Robotics using Photogrammetry and Virtual Reality,” *arXiv preprint arXiv:1905.11377*, 2019

[6] D. Scaramuzza, and F. Fraundorfer, “Visual odometry [tutorial],” *IEEE robotics & automation magazine*, Vol. 18, No. 4, 80-92, 2011

[7] M. Bloesch, M. Burri, S. 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