Commercially Suitable Hybrid Multi-Visual, Inertial and Satellite Navigation Unit for Small Unmanned Aerial Vehicles

N.A. Klyuchnikov,* A.A. Uryasheva, N.S. Rodichenko
Skolkovo Institute of Science and Technology, Moscow

Abstract

Despite extensive research background and availability of components, navigation systems for small Unmanned Aerial Vehicles (UAVs) that incorporate all the three methods (vision-, inertia-, and satellite-based) in a single module are not available in the market. To solve this problem we develop a commercially suitable hybrid navigation module, incorporating two state-of-the-art visual navigation algorithms, namely: Fast Semi-Direct Monocular Visual Odometry and Large-Scale Direct Monocular SLAM, while a framework called Multi-Sensor-Fusion Extended Kalman Filter is used for fusing readings from the visual sensor with the other ones. The software is implemented on top of the Robot Operating System. After prototyping the system, we test it for robustness to different motion and environment types to ensure that it is viable for commercial applications.

1 Introduction

Over the past two decades there has been a lot of advances in research on visual and hybrid navigation for robots [1], however, most of the works were merely aimed at proving the concept, so there has been much engineering work left to be conducted. Nowadays, the standard navigation kit of small UAVs remains consisting only of Inertial Navigation System (INS), some secondary sensors (e.g. an atmosphere pressure sensor) and a Global Navigation Satellite System (GNSS) receiver [2]. Despite some attempts to endow small UAVs with visual sensing [3, 4, 5], there is still no ubiquitous vision-based GNSS backup system.

2 Materials and Methods

Two state-of-the-art monocular visual navigation algorithms, which can be run online on modern microcomputers, were used:

1. Fast Semi-Direct Monocular Visual Odometry (SVO) [6] extracts feature-points from the image-frame only when the observed scene significantly differs from the reference one. For all other interim cases the algorithm applies a sparse direct method for matching patches around the highlighted features. To work correctly SVO requires a high frequency image input stream (approximately 50Hz and up).

2. Large-Scale Direct Monocular SLAM (LSD) [7] is a novel direct monocular visual navigation algorithm, which incorporates frame depth estimation errors into photometric error of their alignment. This approach enables correction of scale drifting by geometry optimization of the scale along with rigid body transformation.

Multi-Sensor-Fusion Extended Kalman Filter (MSF) [8] is an algorithm for fusing navigation data with automatic calibration of absolute and relative measurements.

Robot Operating System (ROS) [9] is an open source platform with a set of tools, libraries and drivers designed to ease the process of creating and sharing robotic software. ROS project constitutes a set of small program modules called nodes, which communicate with each other through publishing messages to specific topics and subscribing to them respectively.

Pixhawk [10] is a low-cost open hardware and software flight control module, which includes an autopilot, processing unit, sensors and input/output system.

PX4 Pro [11] is a running on Pixhawk flight control stack, which is a state-of-the-art software platform. PX4 Pro enables seamless integration of visual navigation data into position estimation by means of an inertial navigation filter INAV [12] or a Kalman filter.

Intel NUC i5 [13] is a mini, but complete computer with Intel Core i5 processor. Used in the work as an additional computational module.

PlayStation 3 Eye [14] is a low-cost digital usb camera which allows high-frequency frame capturing (up to 120 Hz).

To test the system robustness we used environments with different texture parameters. Two examples of test scenes are provided in figure 1. We also used different system motion types, which included rotations, horizontal and vertical translations.

*Email address: fmsnew@gmail.com

Figure 1: Examples of test environments.

3 Results

We tested how the combination of the two visual navigation algorithms improves system robustness in terms of duration of critical moments (i.e. moments within which both algorithms fail to track the motion). Figure 2 represents the statistics of gain in robustness over 34 tests. The value of robustness_gain for each test was calculated as a ratio of the total duration of critical moments to the minimum total tracking fail durations of both algorithms. The mean value of the robustness_gain is approximately 0.46.

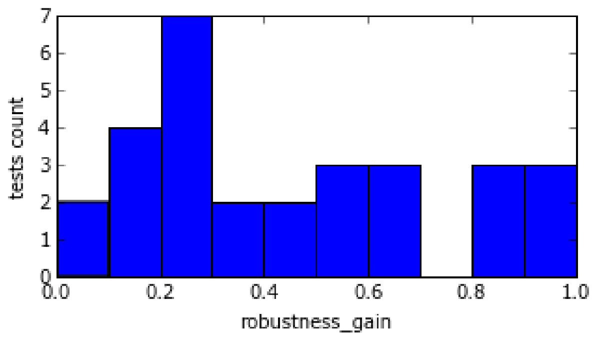

Figure 2: Tests statistics on gain in robustness when combination of visual navigation algorithms is used.

Duration of each single fail period is also an important indicator of robustness (see Figure 3), since the system has to rely only on an extremely unstable INS within critical moments. The mean LSD fail duration is 0.21s, whereas mean hybrid is 0.18. SVO has huge outliers, so it is better described by median, which is 0.39s.

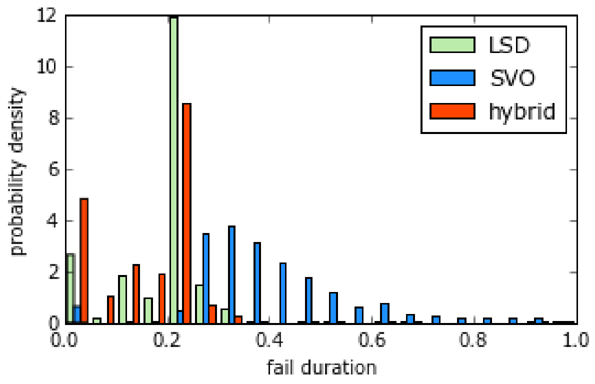

Figure 3: The duration statistics in seconds of each single fail period. More than 98% cases for LSD and hybrid, and 86% cases for SVO are under 1s. The rest cases for LSD and hybrid are under 2s, whereas SVO fail time is unlimited (for some tests this algorithm was even unable to initialize because of lack of features).

4 System Development

4.1 System Architecture

The algorithms reviewed above are quite resource demanding, thus in order to run them online, an additional computationally powerful hardware is needed. On the other hand, to insure against failures of the computation module it is required to provide the autopilot with inertial and satellite data independently. Such considerations led us to a system design, where the autopilot receives these data first and then translates it to the computational module. A Pixhawk control module fits this design and it already includes INS and altimeter. Moreover, PX4 Flight Stack can also estimate states based on these data and fuse it with the one, calculated by the external module, in the role of the visual state estimation. Here it is worth being noticed that the computation module also fuses inertial and satellite data with the visual information, but passes the result in the figure of visual state estimation without any repercussions, just in order to fit the standard interface with the Pixhawk autopilot. ROS is helpful for organizing internal software modules, which run different algorithms (*Core Nodes*), as well as communications between the computation module and the autopilot (*mavros* interface) and camera (*usb_cam* interface). The described system design is represented in the Figure 4.

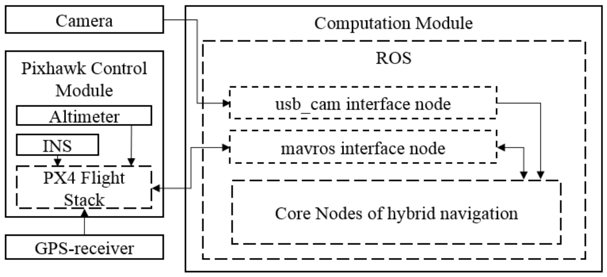

Figure 4: Software and hardware architecture of the hybrid navigation module.

4.2 Core Nodes Framework

Now let us look at the architecture of hybrid navigation core nodes in more detail (Figure 5). They have three input data flows: the first one is raw image shots by the camera (*usb_cam/image_raw*), the second contains processed data from INS and altimeter (*mavros/imu*), and the third data flow represents the final fusion by PX4 Flight Stack of all state estimations incorporating the satellite data as well (*mavros/local_position*).

Visual navigation algorithms are run in the separate nodes and require monochromic undistorted images, which are rectified by *imag_proc* node. In case visual tracking is lost, the basic SVO and LSD implementation moves into the re-localization stage, trying to match incoming images with the explored scenes. The disadvantage of such an approach is that re-localization disables further tracking until the UAV returns in the previously visited area. We modified frameworks of these algorithms to force re-initialization instead of re-localization and notify MSF about it. However, such modification brought two new problems: firstly, the system resets the state estimation, so after re-initialization it works in a new coordinate system, and secondly, the system is losing information about motion during re-initialization, which could be critical if there are tiniest little changes in orientation or scale. We resolved these problems in the supervisor node. State estimations from SVO and LSD are fused with the inertial state estimation and atmosphere pressure in MSF, which is instantiated in two separate nodes (one for each visual navigation algorithm). The implementation of the MSF was also modified to do re-initialization once the corresponding notification is received.

The output data flow is formed by the Supervisor node in the figure of visual state estimation as it was mentioned in the previous section.

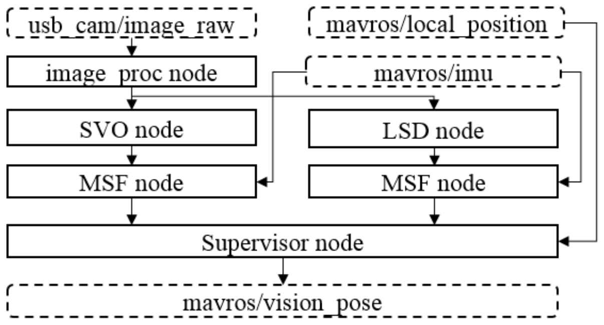

Figure 5: Hybrid navigation core nodes and their framework.

4.3 *Supervisor Design*

The supervisor design and framework are depicted in the Figure 6. Three callback functions (two of them process refined by MSF state estimations from SVO and LSD, the third one processes *local_position* via *mavros* interface) and the main thread are run in parallel. The callback functions receive messages from the corresponding nodes and write their info to the target buffers. The main thread firstly resolves the problems connected with re-initializations of visual navigation algorithms (Reinforced Navigation, elaborated in the next section). Here we emphasize that after this stage, state estimations originated from SVO and LSD are transformed in one consistent coordinate system. A wide class of algorithms now can be applied to fuse these states, from a simple states averaging smoothing through a Kalman filter to more advanced ones with, for example, applying machine learning techniques. The result of the fusion is advertised in *vision_pose* output data flow.

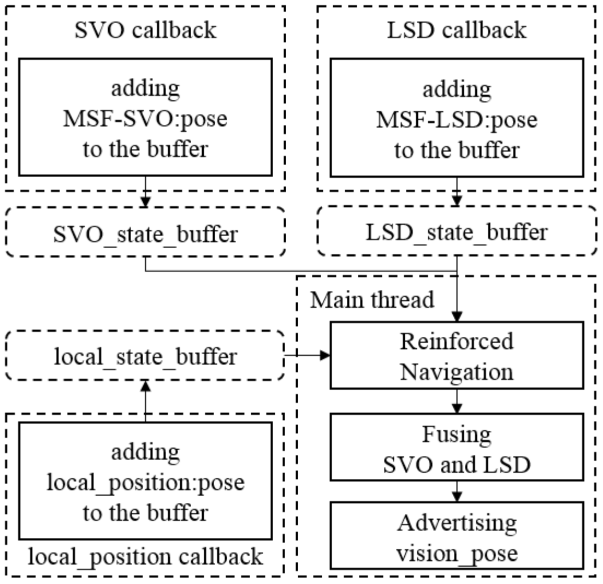

Figure 6: Supervisor node design and framework.

4.4 *Reinforced Navigation*

The Reinforced Navigation algorithm is based on the following idea: when one visual navigation node finishes re-initialization, state estimations from the other one, if it tracks motion correctly, can be used to supplement the lost information. In case both visual navigation algorithms are failed tracking simultaneously (assume that satellite data is not available too), the system can rely only on inertial and pressure sensors. However, it should be taken into account that commercial grade INS are reliable only for a few fractions of a second [15], thus we call such cases critical. Actually, the maximum duration of a critical moment depends on the scale of the mission, but generally, they shall be shorter than approximately 0.5–2 seconds.

5 DISCUSSION

The prototyped navigation unit has the total weight (approximately 500g) and components cost (approximately $600). These values can be considered as an upper bound estimation of the corresponding parameters of the off-the-shelf version, so here we imply that the system can be technically and economically viable for commercial small UAVs, which can carry payload from a few hundred grams and cost from a few thousand dollars.

We examined the system on its robustness by a number of tests with different environments and motion types and the demonstrated level is appropriate for large-scale outdoor applications. There are also several limitations connected with monocular implementation of visual navigation, the most important of them are: the observed scene should be almost static, sufficiently optically transparent and well-lit. However, despite these limits, the system is still suitable for the majority commercial applications, like agriculture, real estate, and infrastructure monitoring, since they use UAVs in good weather conditions during the daylight.

For the prototyping purposes we used only two visual navigation algorithms, however, the developed framework allows for adding other algorithms implemented as ROS node. Moreover the system robustness might be improved if different configurations of SVO or LSD are incorporated, which is also possible within the framework.

6 Conclusion

We developed a commercially suitable hybrid navigation unit, which includes three components; the first two are classical GNSS and INS, whereas the third one is vision-based, which is implemented via a monocular video camera. Visual information is processed in a separate powerful mini computer by two state-of-the-art algorithms for visual navigation, which are integrated and complemented to increase system robustness. Conducted tests showed an average two-fold increase in robustness in terms of visual tracking failure total durations. We also saw a slight gain in terms of average single failure duration time when the algorithms are combined. In addition, the developed framework allows using more algorithms limited only by the computational power.

References

[1] Alberto Ortiz Francisco Bonin-Font and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53:263–296, 2008.

[2] Farid Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29:315–378, 2012.

[3] Dji guidance. Website. Available at http://www.dji.com/product/guidance.

[4] Smart camera px4flow. Website. Available at https://pixhawk.org/modules/px4flow.

[5] Skybotix vi-sensor. Website. Available at http://www.ros.org/news/2014/05/skybotix-opens-vi-sensor-early-adopter-program.html.

[6] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[7] Jakob Engel, Thomas Sch ops, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision (ECCV)*, 2014.

[8] Stephan Weiss Margarita Chli Simon Lynen, Markus W. Achtelik and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[9] Robot operating system. Website. Available at http://www.ros.org.

[10] Pixhawk. Website. Available at https://pixhawk.org.

[11] Px4 pro stack. Website. Available at http://px4.io.

[12] Inertial position estimator inav. Website. Available at https://github.com/iNavFlight/inav/wiki/Inertial-position-estimator-(INAV).

[13] Intel nuc i5. Website. Available at http://www.intel.com/content/www/us/en/nuc/nuc-kit-nuc6i5syk.html.

[14] Playstation 3 eye. Website. Available at https://en.wikipedia.org/wiki/PlayStation_Eye.

[15] Ins and imu grades. Website. Available at http://www.vectornav.com/support/library/imu-and-ins.