2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

On-board Inertial-assisted Visual Odometer on an Embedded System

Guyue Zhou[1], Jiaxin Ye[1], Wei Ren[2], Tao Wang[2] and Zexiang Li[1]
1: Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology
2: Hong Kong Dajiang Innovation Technology Co, Ltd.
{guyue.zhou, jiaxin.ye, wei.ren, frank.wang}@dji.com, eezxli@ust.hk

***Abstract*— In this paper, we propose a novel inertial-assisted visual odometry system intended for low-cost micro aerial vehicles (MAVs). The system sensor assembly consists of two downward-facing cameras and an inertial measurement unit (IMU) with three-axis accelerometers/gyroscopes. Real-time implementation of the system is enabled by a low-cost embedded system via two important features: firstly, simple pixel-level algorithms are integrated in a low-end FPGA and accelerated via pipeline and combinational logic techniques; secondly, a fast yaw-and-translation estimation algorithm works well with a novel outlier rejection scheme based on probabilistic predetermined operations rather than hypothesis testing iterations. We illustrate the performance of our system by hovering a MAV in a GPS-denied environment. Its feasibility and robustness is also illustrated in complex outdoor environments.**

Fig. 1. An 200Hz IMU and two 20Hz downward-facing cameras with baseline of 18cm are mounted on the rotorcraft platform we use – DJI F550. Our real-time visual odometry algorithms can be implemented on board based on a low-cost embedded system.

I. Introduction

The task of building an accurate on-board localization system for low-cost MAVs operating in complex outdoor environments is inherently limited by the low power and payload capability of typical MAVs. Among commonly proposed localization systems, visual odometry (VO) based system stands out as a capable and compact solution, since it offers rich environmental information and should also work well in a GPS-denied environment. An early attempt was conducted by Nister *et al.* [1], which estimated the motion of a stereo head or a single moving camera based on video input alone. Since then, there have been quite a few successful implementations of real-time visual odometry system using different camera configurations (e.g. the monocular case [2], the stereo case [3] and the omnidirectional case [4]) on either wheeled or legged vehicles. However, VO is usually computationally expensive, and the robustness of a pure vision-based localization system is usually weakened in a poor-textured environment.

Recently, the fusion of vision and inertial sensing received great attention from the aerial robotics community, since the IMU is available on most MAVs and is highly complementary to visual sensors in state estimation and 3-D mapping [5]. Available work can be categorized into either loosely coupled or tightly coupled approaches [6]. The tightly coupled approach, for example, uses a single high-order optimal filter to combine the data of visual and inertial sensors for motion estimation. However, it is rarely used in practical MAV applications due to the difficulty of on-board real-time implementation and its high sensitivity to external noise [7].

In comparison, there were many successful attempts at building real-time on-board inertial-assisted visual localization systems using loosely coupled solutions. A low-cost open source and open hardware design of an optical flow method with gyroscope compensation was recently developed in [8], which worked well on a flat ground area. However, it was prone to failure in undulating terrain due to the assumption that the distance between the scene and the camera was approximately constant. Weiss *et al.* [9] proposed a functional inertial-assisted monocular pose estimator based on the parallel tracking and mapping (PTAM) scheme. However, it only worked well in small augmented reality workspace but not in a large-scale complex environment. Shen *et al.* [10] presented a novel system solution by integrating inertial measurement into monocular simultaneous localization and mapping (SLAM) and determine the metric scale using the secondary camera which gave promising results in hovering and autonomous navigation. Nevertheless, it was likely to give a poor localization performance in a relatively high-altitude outdoor environment where frontward-facing cameras might fail to capture the ground features.

The main contribution of this paper is the proposal of an inertial-assisted stereo VO system which can be implemented on a low-cost embedded system in real-time. On the one hand, it enjoys a computational efficiency in terms of fundamental image processing. More specifically, pixel-level algorithms (e.g. anti-distortion, stereo rectification, feature detection and feature description) are pipelined and run on a low-end FPGA; feature scoring session is accelerated via combinational logic techniques. On the other hand, based on our 2-point algorithm which can estimate translational and yaw motion simultaneously (with given 2D-3D feature

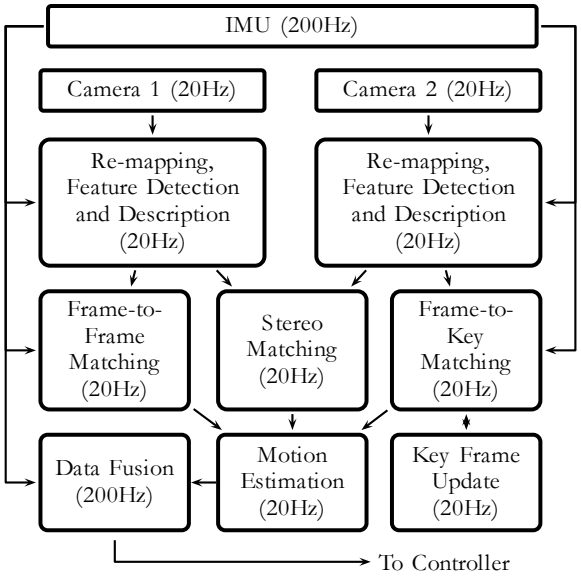


Fig. 2. Software architecture of the proposed system

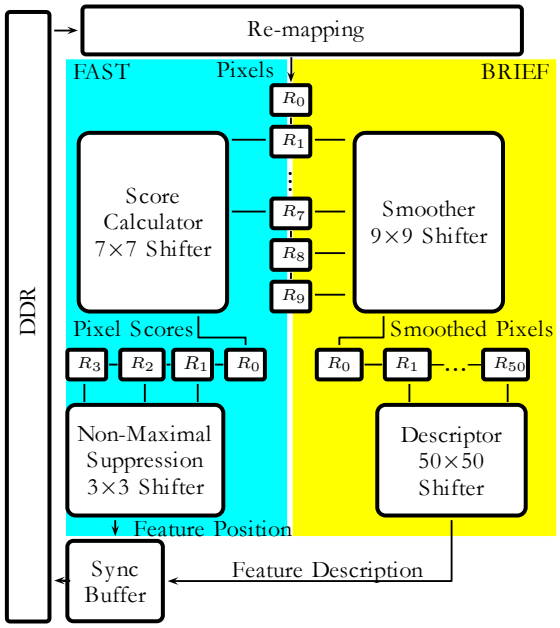


Fig. 3. The pipeline architecture. Notation $R_0, R_1, .., R_k$ are line-buffers where R_0 is a second-level buffer to deal with row changing conditions. It is possible to move a whole group of pixels from one line-buffer to the next within one clock cycle. The blue area indicates the FAST feature detection algorithm while the yellow area represents the BRIEF feature description algorithm.

correspondences and reliable pitch and roll behavior from IMU), we design a simple inlier selection scheme by picking the longest sequence of successive correspondences with motion consistency. It is equally accurate but computationally more efficient than traditional RANSAC [11] method. In the following Section II and Section III, the detailed system description and experimental results based on real flight data will be given respectively.

II. Proposed System

The proposed system hardware architecture consists of two subsystems. The first is a sensing subsystem which includes two downward-facing cameras with VGA(640 × 480) resolution and an external IMU. The second is a processing subsystem based on a low-cost (around $15) Altera's SoC FPGA with a 600MHz ARM Cortex processor inside. The system software architecture is shown in Fig. 2 and will be explained as follows.

A. System Calibration

A total of four parameter sets are calibrated upon system initialization: lens distortion parameters, camera intrinsic parameters, stereo extrinsic parameters and IMU-camera extrinsic parameters. The first three sets are calibrated using traditional methods (OpenCV) [12]. The last set, which is a typical hand-eye calibration problem, is calibrated using Daniilidis' method [13], which is based on simple motion estimation results from the image sequence.

B. Pixel-level Pipeline

We successfully combine IMU, a FAST [14] feature detector and a BRIEF [15] feature descriptor to approximate the rotation-invariant ORB [16] features, used in large angular velocity estimation. These feature-related algorithms are pipelined with other pixel-level algorithms (e.g. anti-distortion and rectification) on a low-cost FPGA. The pipeline mainly consists of five modules: re-mapping, FAST feature scoring, image smoothing, non-maximal suppression and BRIEF descriptor generation (see Fig. 3).

In the re-mapping module, both distortion recovery and rectification is carried out using a pre-computed mapping table (generated according to the camera-related parameters) which corrects the captured images to undistorted and rectified images. The output is a calibrated pixel sequence that drives the other four modules in later process. These pixels are then stored in line-buffers, with a size of the image width, for further pipelining. As shown in Fig. 3, pixels stored in the line-buffers are processed by the score calculator shifter and the smoothing shifter simultaneously, and sent to the next level line-buffers. The next level line-buffers are linked to the suppression shifter and descriptor shifter. Finally, the suppression module generates the feature position while the descriptor module outputs the descriptions for all the pixels. Before being pushed back to the DDR memory, every feature position generated from the FAST side is coupled with the corresponding descriptor from the BRIEF side using a sync buffer, which offsets the fixed time difference between the two sides.

C. Feature Matching

At two successive sampling time, we combine the 3D position of a given feature point in the previous image with the motion from IMU to predict its 2D projection in the next image. Then we use a circle-window to search for the real corresponding feature point. For another matching

scheme considering the rectified stereo frames at the same sampling time, we are able to search the corresponding features directly on the same row of the second image.

D. Motion Estimation

The 2D-3D correspondences are used for motion estimation, since 3D information can be recovered from a pre-calibrated stereo rig directly. Each correspondence can be represented by $c_i = \{\boldsymbol{p}_i, \boldsymbol{u}_i\}$, where $\boldsymbol{p}_i = [x_i, y_i, z_i]^T$ is the 3D coordinate in the previous frame and $\boldsymbol{u}_i = [u_i, v_i, 1]^T$ is the 2D image (undistorted and rectified) pixel index in the current frame. Then we have

$$\lambda_i \boldsymbol{u}_i = \boldsymbol{K}(\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t}) \tag{1}$$

$$\boldsymbol{K} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where $\boldsymbol{K}$ is the camera intrinsic parameters, $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation and translation between two successive camera frames, and λ_i is the unknown scale factor.

Considering the requirement of outlier rejection, we focus on two types of 2-point solvers (the minimal solutions) in this part. For the multiple feature cases, both of them can be easily extended by finding the least square solutions to over-determined linear equations.

1) Translation solver: During the short period between two successive visual samplings, it is reasonable to assume that the exact rotation $\boldsymbol{R}$ can be recovered from IMU readings. For the 2-feature case, let $\boldsymbol{R} = [\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3]^T$ and $\boldsymbol{t} = [t_x, t_y, t_z]^T$ respectively. Then we have $\lambda_i = \boldsymbol{r}_3^T \boldsymbol{p}_i + t_z$. Replace λ_i with $\boldsymbol{t}$ in (1) and we can estimate $\boldsymbol{t}$ with the least square solution to the following linear equations.

$$\begin{bmatrix} (u_1 - u_c)\boldsymbol{r}_3^T\boldsymbol{p}_1 - f_u\boldsymbol{r}_1^T\boldsymbol{p}_1 \\ (v_1 - v_c)\boldsymbol{r}_3^T\boldsymbol{p}_1 - f_v\boldsymbol{r}_2^T\boldsymbol{p}_1 \\ (u_2 - u_c)\boldsymbol{r}_3^T\boldsymbol{p}_2 - f_u\boldsymbol{r}_1^T\boldsymbol{p}_2 \\ (v_2 - v_c)\boldsymbol{r}_3^T\boldsymbol{p}_2 - f_v\boldsymbol{r}_2^T\boldsymbol{p}_2 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c - u_1 \\ 0 & f_v & v_c - v_1 \\ f_u & 0 & u_c - u_2 \\ 0 & f_v & v_c - v_2 \end{bmatrix} \boldsymbol{t} \tag{3}$$

2) Translation and yaw solver: For long-term applications (e.g. keyframe-based hovering), the yaw angle observed from IMU is not reliable. Therefore the rotation matrix should be decomposed into $\boldsymbol{R} = \boldsymbol{R}(\theta)\hat{\boldsymbol{R}}$, thanks to the proximity between the optical axis of the downward-facing camera and the direction of gravity.

Specifically, we need to estimate the yaw component

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

with a given $\hat{\boldsymbol{R}}$ representing the pitch and roll information from IMU.

By simply extending (1), we can obtain

$$f_u(x_i\cos\theta - y_i\sin\theta + t_x) = (u_i - u_c)(z_i + t_z) \tag{5}$$

$$f_v(x_i\sin\theta + y_i\cos\theta + t_y) = (v_i - v_c)(z_i + t_z) \tag{6}$$

where $\hat{\boldsymbol{R}}\boldsymbol{p}_i = [x_i, y_i, z_i]^T$.

With two given correspondences, we firstly use them to eliminate t_x and t_y as below

$$(x_1 - x_2)\cos\theta - (y_1 - y_2)\sin\theta = F_1(t_z) \tag{7}$$

$$(x_1 - x_2)\sin\theta + (y_1 - y_2)\cos\theta = F_2(t_z) \tag{8}$$

where F_1 and F_2 are linear functions with regard to t_z.

Then we can obtain a quadratic equation of t_z by adding the squared (7) and (8) together, implying a close-form solution to t_z. Moreover, the spurious solution to this equation can be easily rejected by checking the reprojection error of the known correspondences with estimated $\cos\theta$, $\sin\theta$ and $\boldsymbol{t}$, which can be further solved from linear equations (7), (8) and (3).

E. Outlier Rejection

We shall now introduce our novel outlier rejection algorithm – LONgest Successive Consistency (LONSC). We will explain the details of this algorithm and compare it to the traditional well-known outlier rejection method – RANSAC.

The basic idea of LONSC is to sweep all correspondences stored in a 1D array and regard the longest successive motion-consistent sequence (SMS) as an inliers set. This inliers set will be used to estimate a motion model which can gather most of other inliers. The LONSC algorithm architecture is shown in Algorithm 1.

Algorithm 1 Motion estimation based on LONSC

Require:
The sequence of correspondences $\{c_1, c_2, ..., c_{n_c}\}$;
The rotation $\boldsymbol{R}$ provided by IMU;
Ensure:
The set of inliers $\boldsymbol{\Phi}$;
The estimated translation $\boldsymbol{t}$;
$MLen = MTail = CLen = 0, \boldsymbol{T} = \boldsymbol{0}, \boldsymbol{\Phi} = \varnothing$;
for $i = 2$ to n_c **do**
  if $\boldsymbol{R}, \boldsymbol{T}$ and c_i are consistent **then**
    $CLen = CLen + 1$;
    if $CLen > MLen$ **then**
      $MLen = CLen, MTail = i$;
    end if
  else
    Use c_{i-1} and c_i to estimate $\boldsymbol{T}$;
    $CLen = 1$;
  end if
end for
Use $\{c_{MTail-MLen+1}, ..., c_{MTail}\}$ to estimate $\hat{\boldsymbol{t}}$;
for $i = 1$ to n_c **do**
  if $\boldsymbol{R}, \hat{\boldsymbol{t}}$ and c_i are consistent **then**
    Add c_i to $\boldsymbol{\Phi}$;
  end if
end for
Use $\boldsymbol{\Phi}$ to estimate $\boldsymbol{t}$;
return $\boldsymbol{\Phi}, \boldsymbol{t}$;

To demonstrate the feasibility of LONSC, we compare its performance to RANSAC's in terms of both computational

efficiency and failure rate under the same inlier ratio. Denote the inlier ratio as $r_i = n_i/n_c$, where n_c is the total number of matched 2D-3D correspondences and n_i is the amount of inliers. The number of outliers n_o equals $n_o = n_c - n_i$.

1) 2-point RANSAC: The probability that traditional 2-point RANSAC fails is approximately:

$$P_{RF} = (1 - r_i^2)^m \tag{9}$$

where m is the amount of hypothesis samplings. For instance, in a typical situation where $n_c = 100$ and $r_i = 0.7$, the minimal m satisfying $P_{RF} < 10^{-4}$ is 14.

2) LONSC: To study the failure rate of LONSC, we will firstly study the failure cases. Since LONSC will fail when the longest SMS is an outliers set, we can calculate the failure rate by finding the number of cases where outliers form a longer SMS than inliers do. We firstly consider the following problem: if both inliers and outliers are stored randomly in a 1D array, compute the number $G^k(n_o, n_i)$ of cases that no k successive inliers exist with known n_o and n_i (considering only the combination of inliers and outliers).

$G^k(n_o, n_i)$ can be solved by inserting outliers alternatively to divide inliers into different groups with a group size less than k. Define $g^k(i, j)$ as the number of cases of inserting the i th outlier behind the j th inlier satisfying the assumption that no k successive inliers exist before the i th outlier. Then we can generate g iteratively using

$$g^k(i, j) = \sum_{l=0}^{\min(k-1, j)} g^k(i-1, j-l) \tag{10}$$

with initial values $g^k(0, j)$ for every possible k and j.

$$g^k(0, j) = \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{if } j \neq 0 \end{cases} \tag{11}$$

Then we can get the number of cases where no k successive inliers exist from the equation below

$$G^k(n_o, n_i) = g^k(n_o + 1, n_i) \tag{12}$$

To identify the failure rate of LONSC, we first find the total number n_f of cases where the longest SMS consists of outliers (considering only the combination of inliers and outliers). To ensure the authenticity of the failure rate, we would like to find an upper-bound (some duplicates included) of the expectation value of $n_g(k, n_o, n_i)$ – the total number of failure cases for a fixed k representing the length of the longest SMS consisted of outliers with known n_o and n_i. By treating these k successive motion-consistent outliers as one special 'big' outlier, we can get the equation below

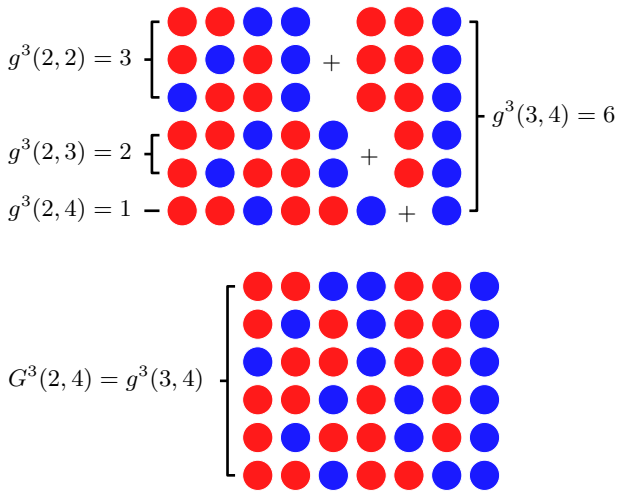
$$n_g(k, n_o, n_i) \leq (n_o - k + 1){p_m}^{k-1} G^k(n_o - k + 1, n_i) \tag{13}$$

where we assume the probability for k successive outliers to be motion consistent is ${p_m}^{k-1}$. A numerical example may clarify the mechanics of g, G and n_g as shown in Fig. 4.

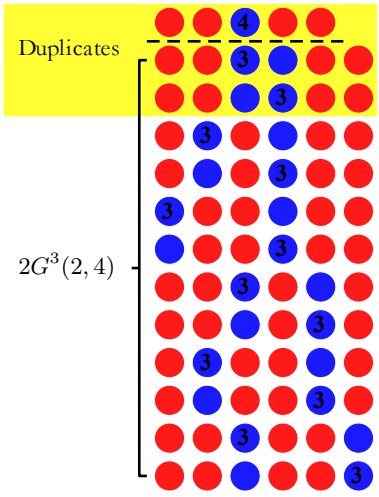
When finding the failure rate of LONSC, we make another assumption. Since we have no prior knowledge about the arrangement of outliers when a new sequence of correspondences is given, we assume that the probability of every possible permutation is equal to each other. Then we can represent LONSC's failure rate by P_{LF}:

$$P_{LF} = \frac{n_o! n_i! n_f}{n_c!} \leq \frac{n_o! n_i! \sum_{k=1}^{n_o} n_g(k, n_o, n_i)}{n_c!} \tag{14}$$

Fig. 5 shows the failure rate (its upper-band in fact) of LONSC on a logarithmic scale under a different number of correspondences and inliers ratios, with $p_m = 0.6$ which is much larger than the case in practice. From the figure,



(a) An example of g and G



(b) An example of n_g

Fig. 4. (a) An example of $g^3(3,4)$ and $G^3(2,4)$: all possible sequential combinations of inliers (red balls) and outliers (blue balls) are listed, the left part shows how to use $g^3(2,*)$ to obtain $g^3(3,4)$ and the right part illustrates the relation between g and G where the dashed ball should be thought as a virtual outlier; and (b) an example of $n_g(3,4,4)$: the blue ball with an arabic figure n inside stands for a sequence consisted of n outliers and two different kinds of duplicates are highlighted with yellow background.

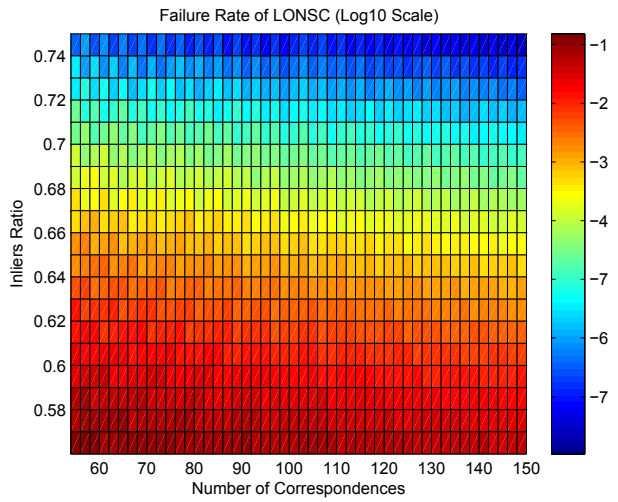


Fig. 5. Failure rate of LONSC on logarithmic scale

the failure rate of LONSC is at least as low as the one of RANSAC for the same inlier ratio $r_i = 0.7$ while the RANSAC obviously has a higher computational complexity (recall that we need to run 14 iterations to make the expectative failure rate of RANSAC to be less than 10^{-4}). In particular, when the number of correspondences is 100 and $r_i = 0.7$, the failure rate of LONSC is less than 2.228×10^{-5}. Note that the value of p_m is overestimated here, the failure rate of LONSC can be even lower in practical cases.

Meanwhile, we can also realize that the upper band of the expectative failure rate of LONSC is quite sensitive to the inlier ratio r_i therefore we need to make the criteria stricter when selecting correspondences. Based on our experience, LONSC always has a very robust performance with the cases $r_i > 0.5$. On the other hand, for a fixed inlier ratio, LONSC is more reliable when there are more correspondences. That is because the majority (inliers when $r_i > 0.5$) will take advantage of the fact that the expectative length of the longest SMS becomes longer.

F. Keyframe-based Hovering and Data Fusion

As is the case with most VO, we can reduce the accumulated errors by computing pose from a previous keyframe and only update the keyframe where there are few detected correspondences between the keyframe and recent frames. This will ensure the positive performance of our system when the MAV is in the hovering mode. For the data fusion part, we simply implement an extended kalman filter (EKF) to fuse the inertial and visual data for velocity estimation.

III. Experimental Results

A. Experiment Configurations

The UAV platform used in the experiments was a hexcopter provided by Dajiang Innovation Technology Co, Ltd. (DJI), with an IMU, a precise GPS positioning system and an autopilot board. On this platform, we had a downward-facing CMOS global shutter camera pair with a 18cm baseline. These cameras captured and transferred images with VGA (640×480) resolution to a processing board consisting of a low-end SoC FPGA (with a 600MHz ARM Cortex hardcore inside). The cameras' frame-rates were set to be 20Hz whereas the IMU provided measurements at 200Hz.

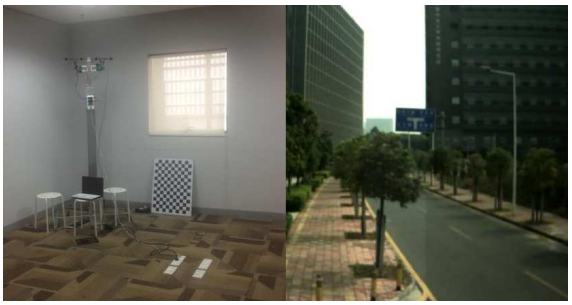


Fig. 6. Experimental environments. The left and right parts show (a) an indoor LAB area, and (b) an urban street with a complex environment.

The experimental environments included an indoor LAB area and a complex urban street (Figs. 6(a) and 6(b)). We conducted three experiments: (1) performance evaluation of our algorithm compared to traditional algorithm on real data, (2) autonomous navigation and hovering using the positioning data provided by the on-board estimator, and (3) real-time localization in complex environment.

B. Comparison to Traditional Methods

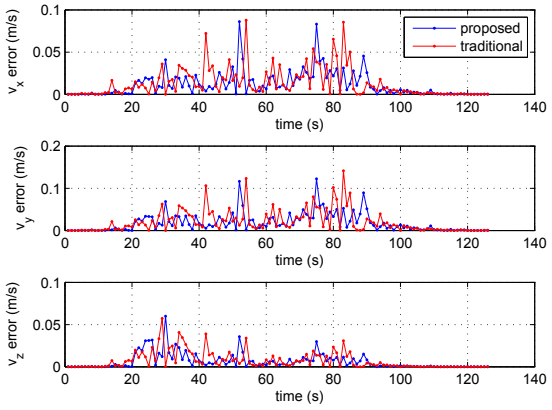


Fig. 7. Velocity estimation errors of the proposed (blue) and traditional (red) algorithms

To evaluate the performance of our proposed algorithm, we compared it with another stereo visual odometer which differed from ours in two major aspects: (1) using 2-point RANSAC (maximum hypothesis iterations: 14) to reject outliers, and (2) implementing the pixel-level algorithms fully on the ARM Cortex hardcore. Then we ran these two algorithms on the same random flight data collected in our indoor environment with the maximum velocity 5m/s.

The velocity estimation errors, referred to the ground truth provided by a group of Vicon-like global cameras, are shown

TABLE I

RUNTIME OF THE PROPOSED AND TRADITIONAL ALGORITHMS

Module	Proposed Algorithm	Traditional Algorithm
Re-mapping	N/A	77.28ms
FAST	N/A	184.68ms
BRIEF	N/A	195.39ms
Pixel-wise Pipeline	16.68ms	N/A
Feature Matching	22.95ms	22.95ms
Motion Estimation (with Outlier Rejection)	8.61ms	57.14ms
Filtering	0.35ms	0.35ms
Total Runtime	48.59ms	537.79ms

in Fig. 7, where we can empirically verify that the accuracy of the two algorithms are quite close. The expectation and standard deviation of the velocity estimation errors using our algorithm and traditional algorithm are respectively $\boldsymbol{m}_{e_1} = [0.010, 0.016, 0.006]^T$m/s, $\boldsymbol{\sigma}_{e_1} = [0.015, 0.022, 0.009]^T$m/s and $\boldsymbol{m}_{e_2} = [0.011, 0.018, 0.006]^T$ m/s, $\boldsymbol{\sigma}_{e_2} = [0.017, 0.026, 0.010]^T$m/s.

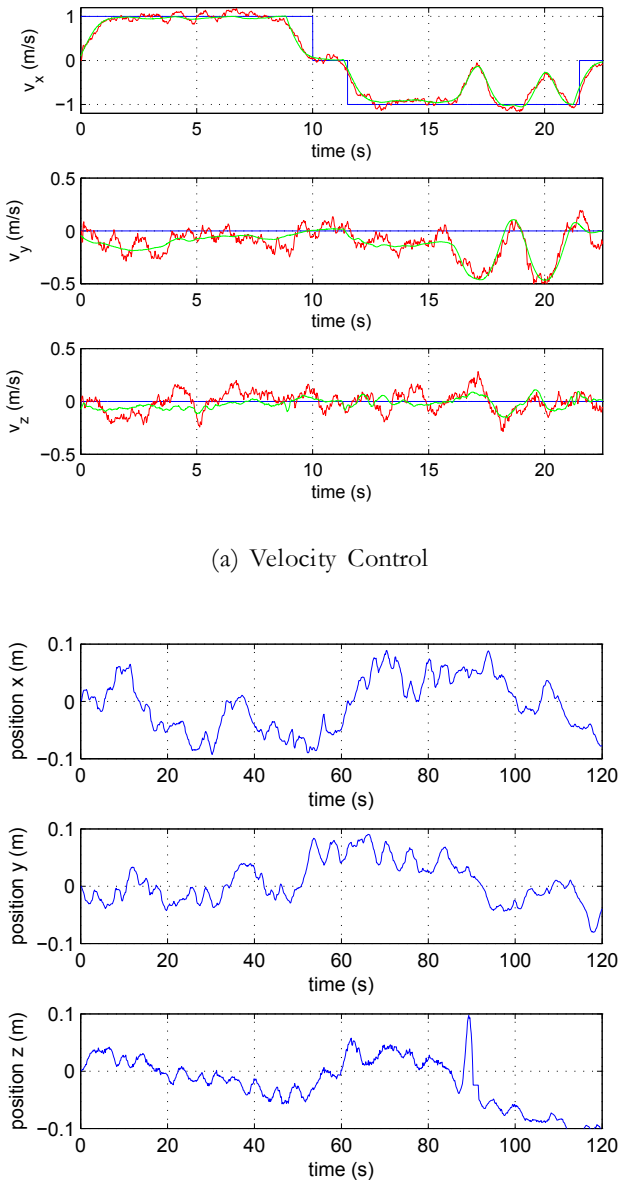
Under similar performance in accuracy, the computational efficiency of these two algorithms are compared in TABLE. I, showing the average runtime of every module in both algorithms. In comparison, our system is greatly accelerated (more than 10 times faster than the traditional method) by the FPGA techniques and the new outlier rejection scheme with $O(2n)$ computational complexity (n stands for the number of feature correspondences).

C. Autonomous Navigation and Hovering

We designed a straight-line navigation task and a hovering task in indoor environment to demonstrate the capability of real flight control using our system. During the experiment, the estimation results of our system were transferred to a simple PD velocity controller based on an H-infinity attitude controller embedded in the DJI's autopilot. We loaded the control command sequence in advance therefore both tasks were finished autonomously. In the first task, an impulse response with the maximum magnitude $1m/s$ was desired and Fig. 8(a) shows a relatively stable velocity control result with the average error less than $0.137m/s$ and standard deviation less than $0.132m/s$. Note that these results are based on observations (red curve) instead of ground truth (green curve) from GPS. In the second task, the hovering lasted for a two-minute-long period and Fig. 8(b) demonstrates a perfect hovering performance given by a simple PD position controller which successfully eliminates the drift thanks to the existence of keyframes. The average positional error is $0.073m$ and the corresponding standard deviation is $0.027m$.

D. Localization in Complex Environment

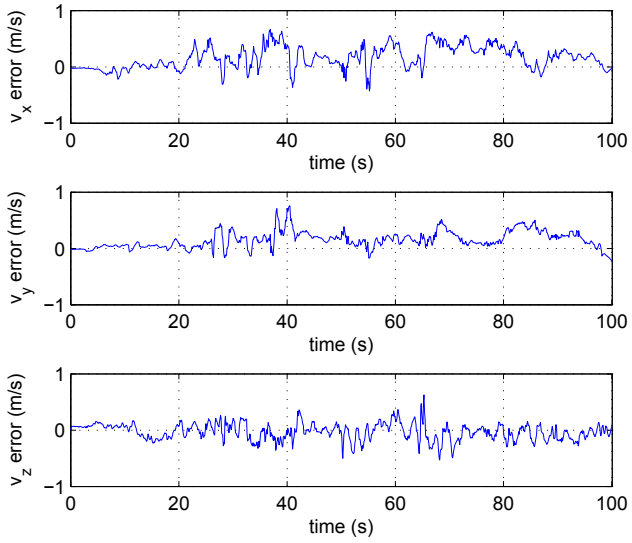
This experiment considered the accuracy of the on board localization system in an urban street environment which contained a mixture of buildings, streets, trees, grass and vehicles. We collected 12 groups of data and only a typical one of them is shown here due to space limitations. In this case, the MAV was commanded to fly through the street with an average speed of approximately $4m/s$ (the maximum speed is around $7m/s$) and eventually landed next to the starting point via a remote controller. The total duration of this experiment was 120 seconds and the flight trajectory of the MAV is shown in Fig. 9(b). Meanwhile, Fig. 9(a) demonstrates that our result is quite close to the ground truth provided by GPS with an expectation of $\boldsymbol{m}_e = [0.020, 0.016, 0.012]^T\ m/s$ a standard deviation of $\boldsymbol{\sigma}_e = [0.016, 0.013, 0.010]^T\ m/s$ in term of the velocity estimation errors.



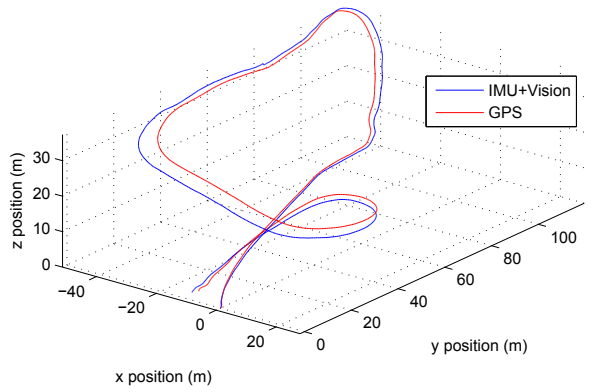
(a) Velocity Control

(b) Hovering

Fig. 8. Experimental results on (a) straight-line navigation: the blue, red and green curves stand for desired velocity, measured velocity and reference velocity respectively, and (b) hovering.



(a) Velocity Errors



(b) Flight Trajectory

Fig. 9. (a) Velocity estimation errors, and (b) flight trajectory: the distance of this flight is around $400m$, the average positional error between GPS and integrated odometry results is $2.696m$ and the ultimate landing positional error is $2.118m$.

IV. Conclusion and Future Work

In this paper, we present an inertial-assisted visual odometry system, which can be implemented on a low-cost embedded system. It features a high-efficiency FPGA-based pipeline and a fast motion estimation scheme. Several experiments have been carried out to demonstrate the capability of our VO system for real flight control in both indoor and outdoor complex environments.

Such a low-cost and accurate VO system will serve as the first step toward a fully autonomous MAV platform in complex GPS-denied environments in the future. We will also try to extend our hardware architecture to solve the problem of autonomous navigation and obstacle avoidance as well and target at an all-in-one compact intelligent flying platform in the next step.

Acknowledgment

The research project concerned in this paper is jointly granted by Hong Kong Innovation and Technology Commission (HKITC) and DJI with the HKITC project reference number UIT/117. Additionally, the authors would like to thank all engineers from DJI vision group for their kindly support and assistance.

References

[1] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–652, IEEE, 2004.

[2] P. Hansen, H. Alismail, P. Rander, and B. Browning, "Monocular visual odometry for robot localization in lng pipes," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3111–3116, IEEE, 2011.

[3] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3946–3952, IEEE, 2008.

[4] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1015–1026, 2008.

[5] A. Martinelli, "Closed-Form Solutions for Attitude, Speed, Absolute Scale and Bias Determination by Fusing Vision and Inertial Measurements," Rapport de recherche RR-7530, INRIA, Jan. 2011.

[6] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," *The International Journal of Robotics*, vol. 26, pp. 519–535, 2007.

[7] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart, "Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4546–4553, May.

[8] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *Intl. Conf. Robotics and Automation*, 2013.

[9] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 957–964, IEEE, 2012.

[10] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation for autonomous rotorcraft mavs in complex environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[12] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.

[13] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[14] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, pp. 430–443, Springer, 2006.

[15] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010*, pp. 778–792, Springer, 2010.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011.