Onboard Hover Control of a Quadrotor using Template Matching and Optic Flow

Ping Li[1]**, Matthew Garratt**[2]**, Andrew Lambert**[3]**, Mark Pickering**[4] **and James Mitchell**[4]
School of Engineering and Information Technology, University of New South Wales, Canberra, Australia

Abstract— *Autonomous hover control of a low-cost Micro Air Vehicle (MAV) is considered in this paper. To avoid the long-term drift during hover, the 'snapshot' idea is practiced, where an image of the ground under the MAV is stored as the reference image, and the following images are directly compared with this reference image for estimating horizontal position. For hover control, the measured position is used in conjunction with the speed estimated from frame-to-frame image motion. All computations are performed onboard the vehicle and controller parameters are roughly tuned in the experiments. Flight tests carried out both indoors and outdoors prove the effectiveness of the proposed method for the hover control of a MAV.*

Keywords: Visual hover control, Micro aerial vehicle, snapshot, long-term drift.

1. Introduction

Much work has been devoted to the control of Micro Aerial Vehicles (MAV) using vision. Visual sensors are small, light-weight and have a large field of view and low power consumption, making them an ideal choice for platforms with limited payload. Visual means can act as a good complement to, if not replace, other navigation sensors for improving their positioning accuracy and reliability.

The relative movement of a MAV to the environment can be inferred from the frame-to-frame image motion known as optic flow (OF). Using a ventral camera and assuming altitude is available, horizontal velocity can be computed from OF and fed back to provide a speed damping effect during hover [1], [2]. If horizontal speed is provided by other sensors, OF can be utilized to estimate height for maintaining terrain-clearance [3]. OF is also used in other aspects like landing [2] and obstacle avoidance [4], [5], by exploiting the divergent flow pattern.

A big challenge for MAVs is that altitude (scale) information of the vehicle should be determined during flight. GPS signal coverage is easily lost in a confined space. A laser range finder is too heavy and power-demanding for MAV. Stereo vision can be used [6], however a minimum baseline is required that makes miniaturization difficult. Besides, the computational cost using two cameras is expected to be much more than that using just one camera. An approach in [7] makes an attempt to estimate height by doing texture analysis on a downward-looking camera. This method is proven to work only at low altitude environments with rich texture. With a monocular camera installed on their Pelican quadrotor, altitude is estimated [8] by combining Simultaneous Localization and Mapping (SLAM) algorithm with Inertial Measurement Unit (IMU). It is found in their experiments that the map is lost from time to time, and it takes a long time for the algorithm to recover. A short hovering period has to be inserted every few seconds to adjust the map with the gravity vector.

With only speed control using OF, the vehicle still drifts away over time in hovering because there is no absolute position feedback. A simple and unique pattern of known geometry is painted on the ground in [9] so that the MAV can identify the pattern to estimate altitude and horizontal position at the same time for hover control. This technique limits the operation of the vehicle to artificial environments while we target natural landscapes. Using a global map, the SLAM algorithm has the ability to correct for long-term drift [8], [10]. However, to store and update the map, SLAM is very time and memory consuming, usually requiring a powerful processing unit that is not available on a low-cost MAV.

The idea of visual snapshot is proposed in [11], where during hover, an image of the ground is captured and stored as the reference (snapshot) image, against which the following images are compared to calculate the absolute snapshot displacement for providing position feedback. In this paper, onboard hover control using both optic flow and snapshot algorithms is successfully implemented on an AR Drone version 1.0 quadrotor, which is a very cost-effective platform. Actual height is measured by the onboard ultrasonic sensor. A number of flight tests in both indoor and outdoor environments shows that the proposed approach can effectively prevent long-term drift against external disturbances.

2. Quadrotor Platform

2.1 Hardware and Software

The AR Drone 1.0 shown in Fig. 1 is a small battery-powered quadrotor. The four rotors, driven through brushless motors, control thrust by changing the Revolutions Per Minute (RPM) of each rotor. Onboard sensors are: (a) an ultrasonic sensor which has a range up to 6 m updating at 25Hz; (b) Bosch BMA150 3-axis accelerometers; (c) a

2-axis IDG500 gyroscope measuring pitch and roll rate, a single-axis EPSON XV3700 gyroscope for yaw rate [12]. As one of the mainboards, the navigation board has a 16-bit 40MHz PIC micro-controller that samples the inertial sensors at 200Hz. Another mainboard is the motherboard which features a 468MHz ARM9 processor, a 128MB RAM running at 200MHz and also a Wi-Fi chip. Multiple threads are managed by a BusyBox v1.14.0 version of the Linux operating system.

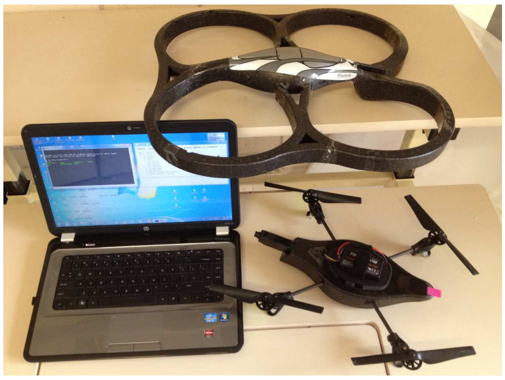

Fig. 1: Parrot ARDrone 1.0 with indoor hull and a laptop as the ground station.

The drone has two cameras: one is downward-looking with a 45°×35° field of view providing a color image of 176×144 pixels; the other is forward-looking with a 75°×60° field of view providing a color image of 640×480 pixels. The two cameras are connected to the ARM9 processor which encodes and sends the data from the cameras to a ground station through Wi-Fi link.

The downward-looking camera is of particular interest to us for the hover control. We have previously tried to use a laptop to process images transmitted from the drone and then send back command to control the hover. It is found that the hover performance is not so satisfactory, especially in outdoor environments where there is wind disturbance. Part of the reason may be the low frame rate available. The ventral camera is able to capture images at 60 fps but due to the imposed Wi-Fi limitation, a client can only receive images at 15 fps, which will bring more latency into the control system. The control signal also has to be sent through Wi-Fi, causing a delay in the drone's response. Therefore, we have chosen to perform the visual control onboard. This is achieved by building on an open source C program found in the personal blog of Hugo Perquin[1] that enables developers to have direct access to the onboard sensors and motors. One can modify the code, cross compile it into ARM executable files, open a *telnet* session to the drone and FTP those files to the drone's */data/video* folder as other folders ask for *sudo* access. Then, one can enter that directory and run the program onboard containing their own algorithms instead of the original Parrot AR Drone program. A user interface called *Open Flight* written in $C\#$ is also included in the custom program that can send commands to the drone and log navigation data to a ground station. Starting from Perquin's program, image processing, sensor fusion and control algorithms have been added for our project.

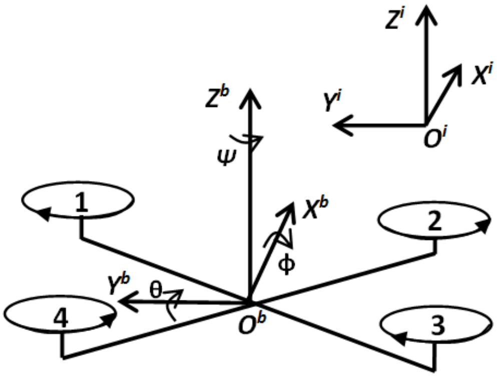

Fig. 2: The defined coordinate system for AR Drone.

2.2 Vehicle Dynamics and Control Structure

The definition of the coordinate system for the Drone is shown in Fig. 2. Generally, two coordinate systems are made use of to describe the motion of a MAV. One is the body coordinate system ($O^b - Z^bX^bY^b$) and the other is the inertial (world) coordinate system ($O^i - Z^iX^iY^i$). A quadrotor is an under-actuated system in that it has four independent rotors and six degree of freedom. The vertical motion is controlled by letting the four rotors change the rotating speed at the same time. Pitch (θ) angle can be adjusted through increasing (decreasing) the speed of rotor 3 and rotor 4 while decreasing (increasing) the speed of rotor 1 and rotor 2 by the same amount. Rolling (ϕ) and yawing (ψ) are regulated in a similar manner. The horizontal motion is realized by making the vehicle change its roll angle or pitch angle first. Therefore, a cascaded (inner-outer loop) structure [17] is often adopted, where the outer loop regulates the speed and position by sending attitude command to the inner loop. The inner loop will seek to manipulate a difference in the speed of the rotors for tracking the desired angle.

After the discussion above, the control commands sent to the four motors can be simply defined as:

$$r_1 = T_{total}/(cos(\phi) \cdot cos(\theta)) + \tau_\phi - \tau_\theta + \tau_\psi \qquad (1)$$

$$r_2 = T_{total}/(cos(\phi) \cdot cos(\theta)) - \tau_\phi - \tau_\theta - \tau_\psi \qquad (2)$$

$$r_3 = T_{total}/(cos(\phi) \cdot cos(\theta)) - \tau_\phi + \tau_\theta + \tau_\psi \qquad (3)$$

$$r_4 = T_{total}/(cos(\phi) \cdot cos(\theta)) + \tau_\phi + \tau_\theta - \tau_\psi \qquad (4)$$

[1] http://blog.perquin.com/blog/ar-drone-program-elf-replacement

where the commands τ_ϕ and τ_θ are the output of PID controller in the inner loop for the drone to reach the desired attitude. Note that AR Drone 1.0 does not have a magnetometer. Perquin's program uses a PI controller to control yaw. The yawing motion appears to be small in flight, so at the moment, we have not tried to make any change. T_{total} is composed of two parts: one is the trim value (T_{trim}) during hover and the other is output of a PD controller regulating height. A PI controller is utilized in the control of the horizontal speed to provide speed damping effect and bound the speed in hovering to a small value. A P controller is then used to control horizontal position based on the position estimation from snapshot displacement. The controller structure is shown in Fig. 3 with the control parameters roughly tuned in the experiments. In the figure, p, q, r are respectively the pitch rate, roll rate and yaw rate, $V_x, V_y, V_z, P_x, P_y, P_z$ are the estimated speed and position. θ_d, ϕ_d are the desired attitude command set by the outer loop.

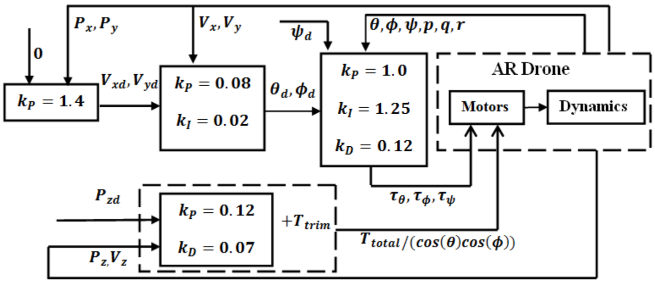

Fig. 3: Controller structure for hover control of the quadrotor with the controller parameters roughly tuned in the experiments.

3. Image Processing

For the frame-to-frame optical flow calculation, the Image Interpolation Algorithm (I^2A) [13] was used because it is robust to noise, fast to implement and able to give sub-pixel accuracy. For a better accuracy, an average filter is usually applied to images before using I^2A. Because only part of the image is needed in the motion estimation, filtering is thus only performed on the region of interest to save time. However, I^2A is not chosen for calculating snapshot displacement, as it is found to be very sensitive to illumination change. This creates a problem for the snapshot computation since lighting conditions may vary over time due to moving cloud interfering with the sunlight or self-shadowing of the vehicle. The Incremental Sign Correlation (ISC) [14] was demonstrated to be robust against illumination variation and chosen for the computation of snapshot displacement in this work. ISC is essentially a binary template matching algorithm that derives a binary image from the intensity image:

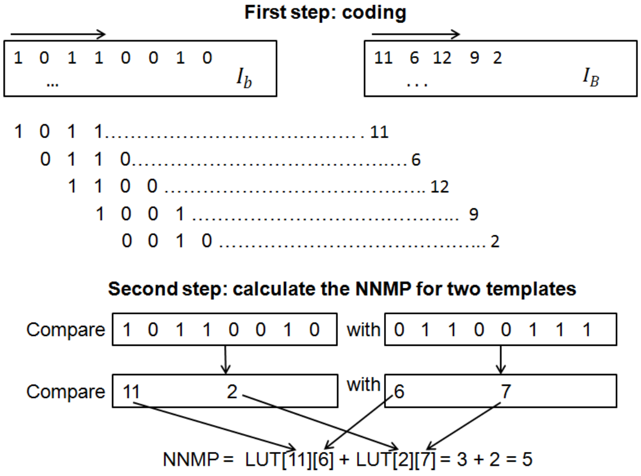

Fig. 4: The binary image is processed in a way that 4 successive bits are coded to an integer and stored in a new image. When calculating the NNMP for two binary templates, a look up table can help reduce the computation.

$$I_b(i,j) = \begin{cases} 1 & \text{if } I(i,j+1) > I(i,j), \\ 0 & \text{otherwise.} \end{cases} \qquad (5)$$

where $I(i,j)$ is the intensity value, $I_b(i,j)$ is the binary value. When comparing two templates after the binary transformation, the Number of Non-Matching Points (NNMP) is used as the similarity measure. For binary template matching, logical operations can replace arithmetic operations, making the algorithm much faster than normal template matching, especially in hardware implementation. In order to further speed up the computation, the binary image was preprocessed in this way [15] that a location (i,j) in the new image I_B stores an integer, which encodes the bits from (i,j) to $(i,j+n)$ in the original binary image. This technique is explained in Fig. 4, for example, $I_B(i,j)$ is 11 if the bits from (i,j) to $(i,j+n)$ in the binary image is '1011' with n being 4. A Look up Table (LUT) with a size of $2^n \times 2^n$ can be constructed beforehand, from which one can directly know the number of different bits that two integers have. In this way, when computing the NNMP for two templates of 8 bits (see Fig. 4) with n being 4, only 2 look-up-table operations and 1 addition are required for this technique while direct comparison needs 8 XOR operations and 7 additions. With the template size unchanged, a choice of a large n seems to reduce the number of operations afterwards, but also consumes more memory and time in storing the coded image and a large LUT size. n is set to be 8 in the paper. Partial Distortion Search (PDS) [16] is also employed to reduce execution time. The idea is to terminate the calculation for a search point if the accumulated NNMP is larger than the minimum NNMP computed at that moment.

In our implementation, optic flow is calculated at 60

fps. The search window for template matching can not be too small for good tracking performance, but larger search window means more computational cost. Search window is set to be [-10,10] and the previous snapshot displacement is used to provide an initial guess for the next search. Given the limited processing power onboard, the calculations for ISC were spread over several frame intervals. Snapshot displacement can be updated by accumulating optic flow during intermediate frames and corrected once the calculation for ISC is finished. It is not guaranteed that every update from ISC is correct and so a confidence measure ($conf$) should be introduced to reject those false measurements. Image motion is calculated for several templates in the snapshot image and assuming yaw angle is small during hover, the standard deviation of the motion vectors for those templates can indicate the reliability of that measurement. The confidence measure is computed as:

$$conf = \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{m}} + \sqrt{\frac{\sum_i (y_i - \bar{y})^2}{m}} \tag{6}$$

where m represents number of templates, x_i, y_i are the image displacement calculated with ISC in the X and Y direction for each template, and $\bar{x}, \bar{y}$ are the mean displacement for these templates. If the confidence measure is below a threshold, the result is used to correct for snapshot displacement and predict the next search, otherwise, the accumulation of optic flow is trusted. Sometimes with repeated pattern such as bricks or tiles, the algorithm may falsely track a similar region. This can be avoided by ruling out a sharp jump in the estimation. If the confidence measure remains beyond the threshold for a certain period of time, another snapshot image is taken as the new reference image.

4. Pose Estimation

4.1 Attitude

A simple complementary filter was used to estimate pitch and roll angle from the inertial sensors output:

$$\theta_n^- = \theta_{n-1}^+ + p \cdot dt \tag{7}$$

$$\theta_n^+ = k \cdot \theta_a + (1-k) \cdot \theta_{n-1}^- \tag{8}$$

$$\phi_n^- = \phi_{n-1}^+ + q \cdot dt \tag{9}$$

$$\phi_n^+ = k \cdot \phi_a + (1-k) \cdot \phi_{n-1}^- \tag{10}$$

where equation (7) and (9) predict pitch angle and roll angle with pitch rate p and roll rate q while equation (8) and (10) corrects the estimation by accelerometer measurement ($\phi_a = atan(a_y/a_z)$, $\theta_a = atan(-a_x/a_z)$, a_x, a_y, a_z are the accelerometer output with the unit being g). k is set at 0.015 in the subsequent experiments.

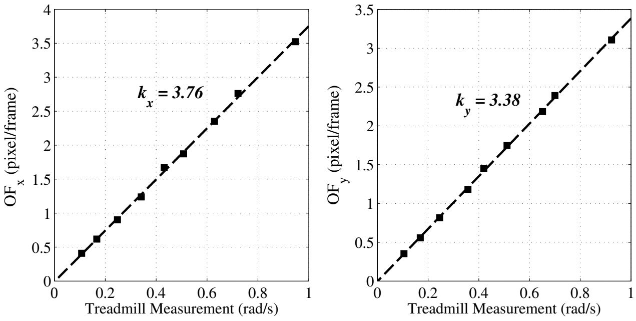

Fig. 5: Calibration for optic flow scale factor

4.2 Speed and Position

Optic flow in the X and Y direction, denoted as OF_x and OF_y, is measured in $pixel/frame$, before it can be used in the estimation of horizontal speed, the scale factor (k_x, k_y) that convert it to $radian/s$ should be found. The scale factor can be extracted according to the camera geometry (field of view and resolution), but it is prudent to re-calibrate it due to optic distortion and other errors. A treadmill was used for this calibration. Please refer to [1] for the detail of this experiment. Fig. 5 gives the calibration result. After that, horizontal speed (v_x, v_y) and position (p_x, p_y) using OF and snapshot displacement (S_x, S_y) is computed as:

$$v_x = (OF_x/k_x + p) \cdot P_z \tag{11}$$

$$v_y = (OF_y/k_y - q) \cdot P_z \tag{12}$$

$$p_x = (S_x/k_x + \theta) \cdot P_z \tag{13}$$

$$p_y = (S_y/k_y - \phi) \cdot P_z \tag{14}$$

where pitch rate and roll rate should be subtracted from the measured OF, and pitch angle and roll angle are subtracted from the snapshot displacement. Speed calculated from optic flow in equation (11) and (12) does not suffer from long-term drift but is very noisy. Speed integrated from acceleration is smooth but drifts over time. So once again these two are combined in the same way that pitch and roll angle are estimated, as in equation (7) to (10). Likewise, horizontal position during hover can be predicted using speed measurement and corrected with the position estimation from snapshot displacement. Perquin's program estimated vertical speed by linear regression based on height measurement from the ultrasonic sensor. Similar to the horizontal speed and position estimation, vertical acceleration can be incorporated to have a smoother vertical speed and height estimation. When pitch angle and roll angle are small, the actual accelerations (A_x, A_y, A_z) in the X^i, Y^i and Z^i direction can be approximated by [17]:

$$A_x = g \cdot (\theta + a_x) \tag{15}$$

$$A_y = g \cdot (-\phi + a_y) \tag{16}$$

$$A_z = g \cdot (a_z - 1) \tag{17}$$

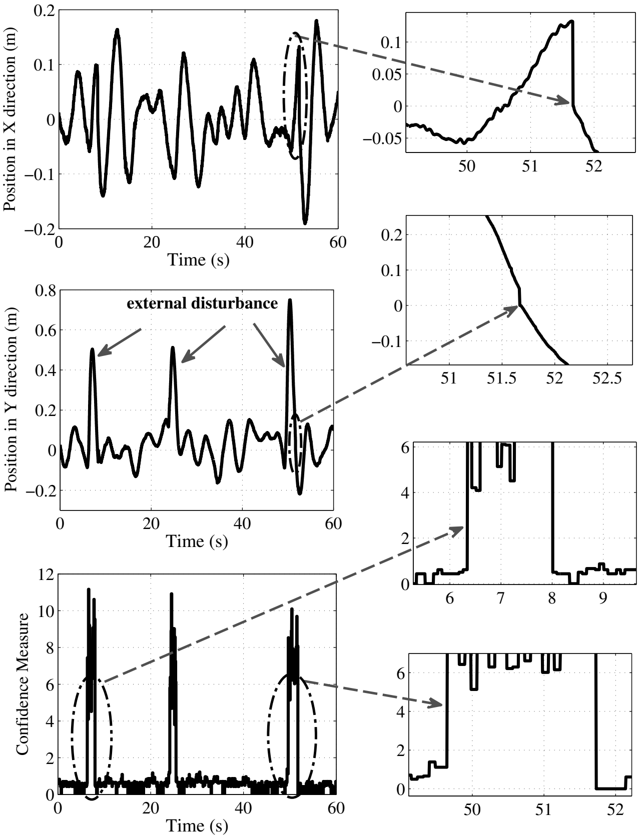

Fig. 6: Horizontal position and confidence measure during an indoor hover under external disturbance.

5. Flight Tests

A number of flight tests were conducted both indoors and outdoors. During hovering in an indoor environment, the drone was disturbed (mainly in the Y direction) by hand three times. Once it is pushed away from the visual anchor point, the confidence measure becomes very large (Fig. 6), and during this time the snapshot displacement is only updated using accumulation of OF. It is noted that for the first two disturbances, the drone is able to come back and lock onto the reference image. For the third case with larger perturbation, the drone failed to make its way back to the original hovering point but remains very close. In the experiment, if the confidence measure stays larger than 2 for 2 seconds, a new snapshot image will be taken. As seen from the zoomed figures pointed to by the dashed arrows in Fig. 6, for the first two disturbances the confidence measure remains larger than 2 for less than 2 seconds, but for the third case, this period is longer than 2 seconds, thus another reference image is captured and position estimation is reset to 0 for another round of tracking. If the tolerance period is raised (for example 3 seconds) for the confidence measure larger than 2, the vehicle may be able to find 'home' again. It seems that the use of a longer tolerance period is able to handle stronger perturbation, however, the integration of OF may have drifted so much that the vehicle can not go back. In fact, this tolerance period reflects how much the integration of OF is trusted. It should be mentioned that we have not tried to determine the optimal value for this tolerance period. The horizontal speed estimation and attitude response are displayed in Fig. 7. The desired attitude angle set by the outer loop are tracked very well by the inner loop.

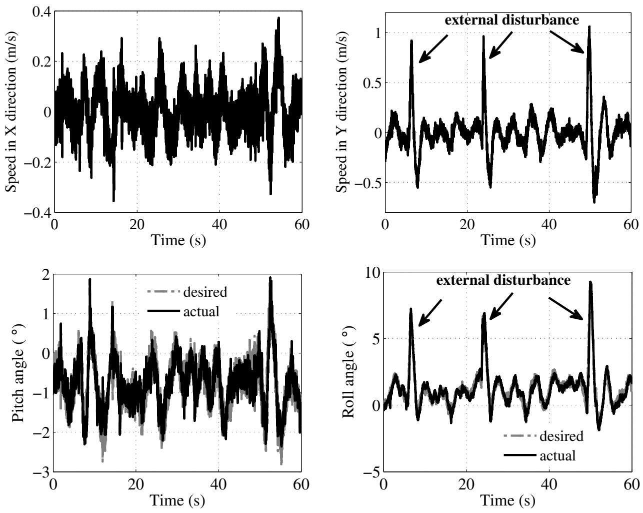

Fig. 7: Horizontal speed, pitch and roll response during an indoor hover under external disturbance

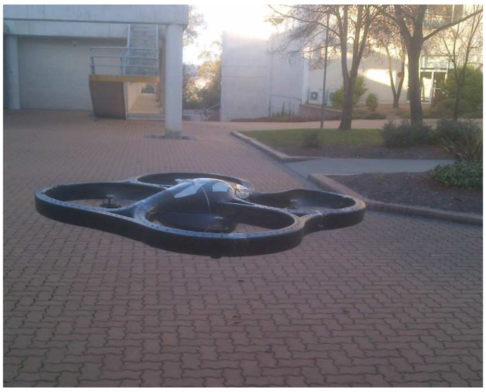

Fig. 8: AR Drone flying outdoors over a repeated pattern.

Fig. 9 gives the horizontal position estimation and confidence measure during hovering outdoors as shown in Fig. 8. The confidence measure goes beyond the threshold more frequently than in the indoor environments. That means new snapshot images are taken more frequently. Despite that, the drone is observed to stay in the vicinity of the original hovering point. The vehicle is flying over an repetitive pattern (Fig. 8) in this experiment, when blown away by wind gust, it is possible to track a new region having similar templates to those in the snapshot image. Because for our case (image resolution, size of the templates, search window), if the drone flies within the boundary of the snapshot image, the snapshot displacement should be less than 50 pixels. Note that in Fig. 10, a small value in confidence measure is reported even when the displacement is more than 100 pixels. This usually results in a sharp jump in the displacement estimation and can be prevented by imposing an upper limit on the variation of current update with respect to previous update. Only when this variation and the confidence measure are both smaller than the predefined threshold will the calculation from ISC be trusted. The horizontal speed and attitude response are shown in Fig. 11.

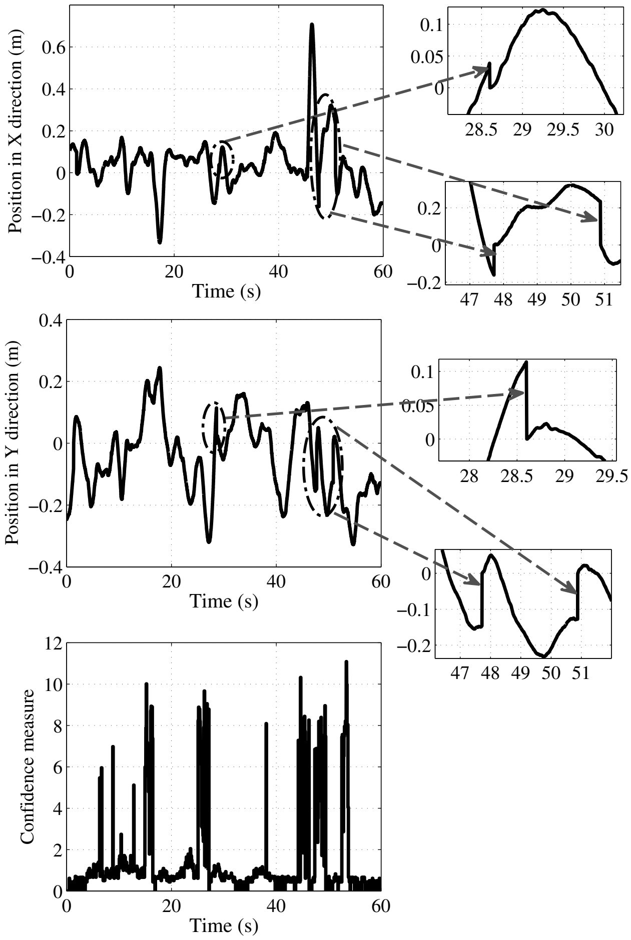

Fig. 9: Horizontal position and confidence measure during an outdoor hover with wind disturbance.

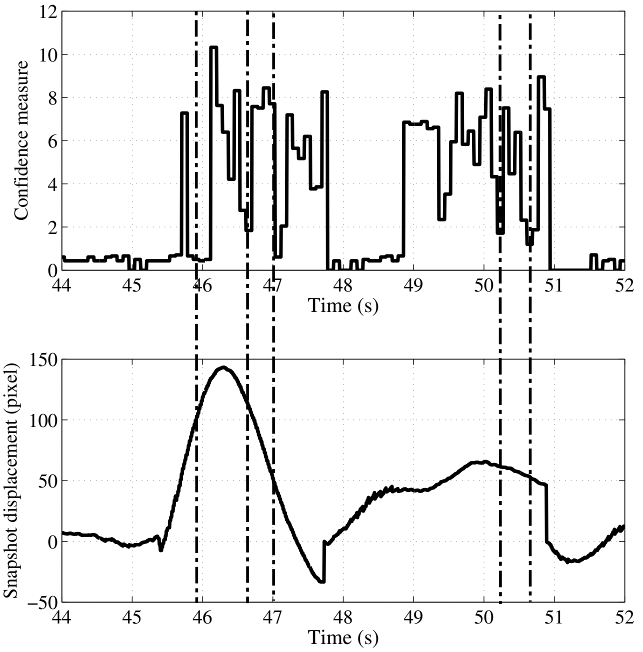

Fig. 10: Snapshot displacement in the X direction versus the confidence measure.

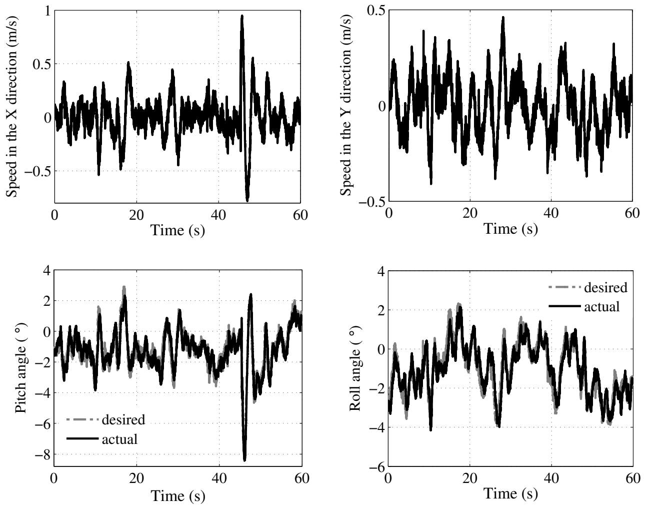

Fig. 11: Horizontal speed, pitch and roll response during an outdoor hover with wind disturbance.

6. Conclusion

In this paper, the snapshot idea [11] is proven to be very effective in eliminating the long-term drift of a rotorcraft in hover. Based on an open source program, onboard implementation of visual algorithms, sensor fusion and controller design are successfully achieved. Flight tests demonstrate the proposed method work satisfactorily both indoors and outdoors even with repeated pattern and light wind disturbance. Although not proven in flight, the ISC algorithm has the ability to deal with illumination change. However, the limitation of the platform should also be mentioned. A strong wind gust can easily drive the drone away from the scene where snapshot image is captured. In fact, even moderate but non-constant wind is difficult enough to deal with. A varying wind condition will lead to a big variation in the attitude, which is the only way that an AR Drone can resist external disturbance. This is very likely to make the drone lose the visual anchor point (may be the reason why the confidence measure goes over the threshold more frequently outdoors than indoors) due to the small field of view of the downward-looking camera. In a dim-light condition, the image quality will get worse and degrades the results. For a good hovering performance under a wider range of circumstances, a better camera with larger field of view is preferred, or the rotors themselves can tilt [19] so that external disturbance can be counteracted without causing much change in the attitude of the vehicle.

Future work on this platform will focus on: (a) optimizing the controller parameters; (b) optimizing the combination of snapshot measurement with the integration of OF; (c) exploiting the distribution of motion vectors [18] for the purely visual control of height [11] and yaw angle as well as horizontal positions.

References

[1] M. Garratt and J. Chahl, "An Optic Flow Damped Hover Controller for an Autonomous Helicopter," in *22nd International UAV Systems Conference*, Bristol, April 2007.

[2] B. Hérissé, F. Russotto, T. Hamel and R. Mahony, "Hovering flight and vertical landing control of a VTOL Unmanned Aerial Vehicle using Optical Flow," in *International Conference on Intelligent Robots and Systems*, France, pp. 801–806, 2008.

[3] M. Garratt and J. Chahl, "Vision-Based Terrain Following for an Unmanned Rotorcraft," *Journal of Field Robotics.*, vol. 25, no. 4–5, pp. 284–301, 2008.

[4] S. Hrabar, G. Sukhatme, P. Corke and K. Usher, "Combined optic-flow and stereo-based navigation of urban canyons for a UAV," in *International Conference on Intelligent Robots and System*, Canada, pp. 3309–3316, 2005.

[5] J. Zufferey and D. Floreano, "Fly-Inspired Visual Steering of an Ultralight Indoor Aircraft," *IEEE Transactions on Robotics.*, vol. 22, no. 1, pp. 137–146, 2006.

[6] P. Corke, "An Inertial and Visual Sensing System for a Small Autonomous Helicopter," *Journal of Robotic Systems.*, vol. 21, no. 2, pp. 43–51, 2004.

[7] A. Cherian, J. Andersh, V. Morellas and N. Papanikolopoulos, "Autonomous altitude estimation of a UAV using a single onboard camera," in *International Conference on Intelligent Robots and System*, pp. 3900–3905, 2009.

[8] S. Weiss, D. Scaramuzzaand and R. Siegwart, "Monocular SLAM Based Navigation for Autonomous Micro Helicopters in GPS-Denied Environments," *Journal of Field Robotics.*, vol. 28, no. 6, pp. 854–874, 2011.

[9] S. Lange, N. Sunderhauf and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments," in *International Conference on Advanced Robotics*, 2009.

[10] J. Engel, J. Sturm and D. Cremers, "Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing," in *Proc. of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, 2012.

[11] M. Garratt, A. Lambert and H. Teimoori, "Design of a 3D snapshot based visual flight control system using a single camera in hover," *Auton Robot.*, September 2012.

[12] P. J. Bristeau, F. Callou, D. Vissière and N. Petit, "The Navigation and Control technology inside the AR. Drone micro-UAV," in *18th IFAC World Congress*, vol. 18, no. 1, Milano, Italy, pp. 1477–1484, 2011.

[13] Srinivasan. M, "An image interpolation technique for the computation of optic flow and egomotion," *Biological Cybernetics.*, vol. 71, no. 5, pp. 401–415, 1994.

[14] S. Kanekoa, I. Muraseb and S. Igarashia, "Robust image registration by increment sign correlation," *Pattern Recognition.*, vol. 35, pp. 2223–2234, 2002.

[15] Y. K. Wang and G. F. Tu, "Fast Binary Block Matching Motion Estimation using Efficient One-Bit Transform," *Department of Electrical Engineering, the Graduate School of Academia Sinica*, Beijing, China.

[16] O. Urhan, "Constrained one-bit transform-based motion estimation using predictive hexagonal pattern," *Journal of Electronic Imaging.*, vol. 16, no. 3, p. 033019, 2007.

[17] Y. Sun, "Modeling, identification and control of a quad-rotor drone using low-resolution sensing," *Master thesis, University of Illinois at Urbana-Champaign*, 2012.

[18] A. Hung, M. Pickering and M. A. Garratt, "Fast image registration using a multi-pass image interpolation approach," in *7th International Conference on Information Technology and Applications*, Sydney, Australia, pp. 21–24, Nov. 2011.

[19] S. K. Phang, C. X. Cai, B. M. Chen and T. H. Lee, "Design and Mathematical Modeling of a 4-Standard-Propeller (4SP) Quadrotor," in *Proceedings of the 10th World Congress on Intelligent Control and Automation*, Beijing, China, pp. 3270–3275, July. 2012.