A Comprehensive UAV Indoor Navigation System Based on Vision Optical Flow and Laser FastSLAM

WANG Fei[1] CUI Jin-Qiang[1] CHEN Ben-Mei[2] LEE Tong H[2]

Abstract This paper presents a comprehensive control, navigation, localization and mapping solution for an indoor quadrotor unmanned aerial vehicle (UAV) system. Three main sensors are used onboard the quadrotor platform, namely an inertial measurement unit, a downward-looking camera and a scanning laser range finder. With this setup, the UAV is able to estimate its own velocity and position robustly, while flying along the internal walls of a room without collisions. After one complete flight, with the collected data the historic UAV path and the indoor environment can be well estimated. The autonomous navigation part of the system does not require any remote sensory information or off-line computational power, while the mapping is done off-line. Complete flight tests have been carried out to verify fidelity and performance the navigation solution.

Key words Unmanned aerial vehicle (UAV) flight control, indoor navigation, simultaneous localization and mapping (SLAM), optical flow, laser FastSLAM

Citation Wang Fei, Cui Jin-Qiang, Chen Ben-Mei, Lee Tong H. A comprehensive UAV indoor navigation system based on vision optical flow and laser FastSLAM. *Acta Automatica Sinica*, 2013, **39**(11): 1889−1900

DOI 10.3724/SP.J.1004.2013.01889

The development of advanced indoor navigation systems for miniature unmanned aerial vehicles (UAVs) has aroused extensive interest recently because of its great potential in military and civil applications. If such a system can be pragmatically implemented, it can be used for exploration and mapping, search and rescue, and other strategic missions. However, this system has to be intelligent enough to overcome problems caused by complicated indoor environments, such as scattered obstacles and denied reception of GPS signals, and constraints of UAV platforms, for example, unconventional mechanical design and limited payload.

It is noted that the topics of UAV modeling and control[1−2], GPS-denied navigation[3−4], path planning[5−7] and simultaneous localization and mapping (SLAM)[8−9] are separately addressed by researchers in areas including robotics, mechatronics, computer vision, computer science, etc. Each of these topics covers a breadth of literature. With the knowledge of these existing theoretical works, this paper, however, focuses on the overall system design, integration and implementation. Moreover, a lot of the existing algorithms in literature have not been realized onboard the UAV in real time. In most cases, only simulations, off-line processing, or part of the whole system have been tested. This largely limits the possible real-life applications of the indoor UAVs, especially when the control or path planning algorithms highly relies on some remote and immobile sensors or computers. As such, this paper selects and customizes the existing algorithms and methods and aims to realize a robust indoor UAV system with real-time processing capability solely onboard. In addition, there are also new ideas about system design, control law implementation, UAV state estimation, and path planning which can distinguish this paper from other complete indoor navigation solutions, such as [10−11].

The content of this paper is organized as follows: Section 1 gives an overall structure of the proposed indoor navigation strategy, followed by Sections $2 \sim 5$, which respectively expand the UAV modeling and control, state estimation, path plan, and SLAM. Section 6 will provide actual implementation results to verify the overall system. Lastly, concluding remarks will be made in Section 7.

1 Platform and system structure

Being mechanically simple and robust, quadrotor helicopters have been widely used as UAV platforms for research purposes these days. There are ready products, such as AscTec′s Pelican, Parrot′s ARDrone, and MikroKopter′s Quadro, which are all capable of performing stable semi-autonomous flight. However, the quadrotor platform used for the work described in this paper is completely self-customized with special considerations about weight reduction and power efficiency (see Fig. 1). Its frame is made of carbon fiber plates and rods with a durable acrylonitrile butadiene styrene (ABS) landing gear. Its dimensions are 35 cm in height and 86 cm in diagonal width (tip-to-tip). This custom-made quadrotor has a maximum take-off weight of 2.9 kg (conservatively tested) and can fly at 8 m/s for about 10 to 15 mins, depending on sensor configuration and environmental factors. The platform onboard system is also fully customizable in terms of sensor arrangement and scalability, meaning that additional electronic boards can be added with a stack-based design.

For the onboard avionics, the IG-500N inertial measurement unit (IMU) from SBG systems is used to provide the

Manuscript received July 10, 2013; accepted August 29, 2013
Invited Articles for the Special Issue for the 50th Anniversary of Acta Automatica Sinica

1. Graduate School for Integrative Sciences and Engineering, National University of Singapore (NUS), Singapore, Singapore 2. Department of Electrical and Computer Engineering, National University of Singapore, Singapore, Singapore

UAV linear accelerations, angular rates and Euler angles. There is also a built-in barometer which can provide air pressure measurement for UAV height estimation. URG-30LX scanning laser range finder from Hokuyo is used to measure the distance from the surrounding objects in a frontal 270° and 30 m′s range. The vision sensor used on this platform is a PointGrey FireFly camera. Two separate onboard processors are used; one is the Gumstix Overo Fire for control and navigation purposes, while the other one, Fit-PC2, is dedicated to vision processing. Fig. 2 shows the full onboard configuration and how the signals flow among all the components.

Fig. 1 The custom-made quadrotor platform

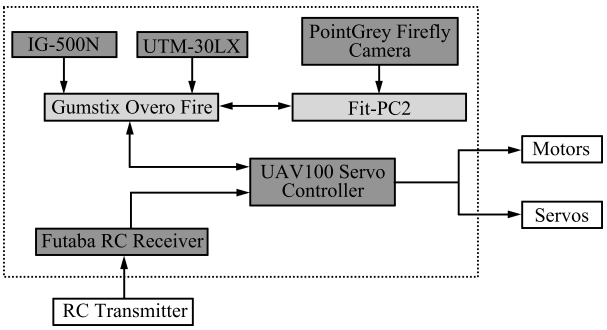


Fig. 2 Onboard avionics configuration of the quadrotor platform

Zooming out, Fig. 3 shows the top-level structure of the proposed indoor navigation solution. As the UAV moves, the IMU sensor and vision sensor′s measurements will be fed to the "State estimator" block to obtain an estimation of all UAV state variables, and then acquired by the "Control law" block for feedback control. The other input to the "Control law" block is the UAV trajectory reference generated by the "Path plan" block. For path planning, the main sensor used is the scanning laser range finder. For the following sections, the "Control law", "State estimator" and "Path plan" blocks will be expanded and detailed, respectively.

2 Modeling and control

The model structure of the quadrotor platform being used is illustrated in Fig. 4. The four-channel control inputs $\delta_{\rm lat}$, $\delta_{\rm lon}$, $\delta_{\rm col}$, $\delta_{\rm ped}$ are fed to the onboard NAZA controller (NAZA is an off-the-shelf all-in-one stability controller specially designed for multi-rotor flying platforms). NAZA controller outputs PWM signals to drive the four motors to generate the thrust forces, which not only lift the platform but also maintain its attitude stability. From the perspective of NAZA, the four inputs from the remote transmitter correspond to the control references for the roll angle (ϕ), pitch angle (θ), yaw angular rate (r), and average motor speed (Ω), respectively. There exists a dynamic process which maps the roll and pitch angles to the lateral and longitudinal velocities (u, v). It is mainly caused by rigid body kinematics with air resistance damping effect.

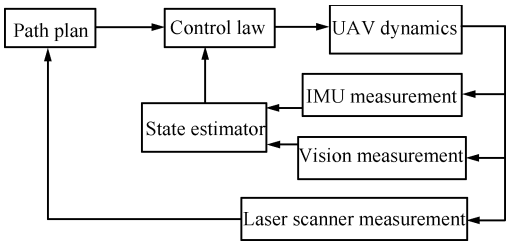


Fig. 3 Overall structure of the indoor navigation system

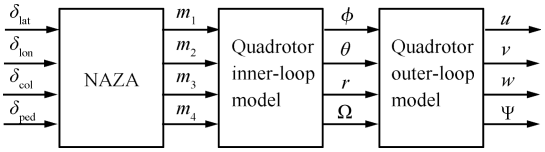


Fig. 4 Overview of quadrotor model structure

The platform operates in the so-called "X" mode as shown in Fig. 5, where there are two frontal rotors and two rear rotors. The UAV body frame is defined as x pointing forward, y pointing rightward, and z pointing downwards, following the right-hand rule. Since the structure configuration of the platform and the design of the onboard system are highly symmetric, it is reasonable to assume that the longitudinal and lateral dynamics of this platform are exactly the same. We also assume that the model is completely decoupled among all four channels. That means the dynamic model of each channel can be identified independently.

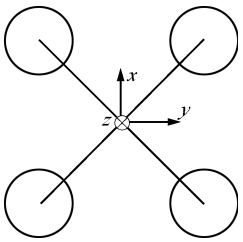


Fig. 5 Quadrotor body frame definition

The model identification process is done with a software toolbox called CIFER. It was developed by the NASA Ames Research Center for military-based rotorcraft systems. It

converts the collected input-output data to frequency-domain responses and then fit to a low-order transfer function using advanced numerical optimization.

Due to the symmetric structure of the quadrotor platform, the roll and pitch dynamics share the same model structure as well as parameters. When the platform is perturbed in the roll (or pitch) direction, the onboard avionics system logs the roll (or pitch) angle and linear velocity′s responses with the corresponding synchronized control input, δ_{lat} (or δ_{lon}). The ultimate goal is to identify the model from the control input δ_{lon} (or δ_{lat}) to the linear velocity u (or v). However, we can divide this task into two sub-tasks, i.e., identify the model from control input to attitude angle and identify the model from angle to velocity. Then the two models can be cascaded to produce an overall longitudinal (or lateral) model.

Using CIFER, the dynamics from the longitudinal (or lateral) control input to the pitch (or roll) angle can be well fitted by the following 4th-order transfer function:

$$H_1 = \frac{9\,688}{s^4 + 27.68\,s^3 + 485.9\,s^2 + 5\,691\,s + 15\,750} \tag{1}$$

The frequency response comparison between the identified model and the real data is shown in Fig. 6. The third subplot in the figure shows the coherence value of the frequency domain matching. At frequencies below 20 rad/s, the coherence value remains above 0.8, indicating the system can be well characterized by a linear process.

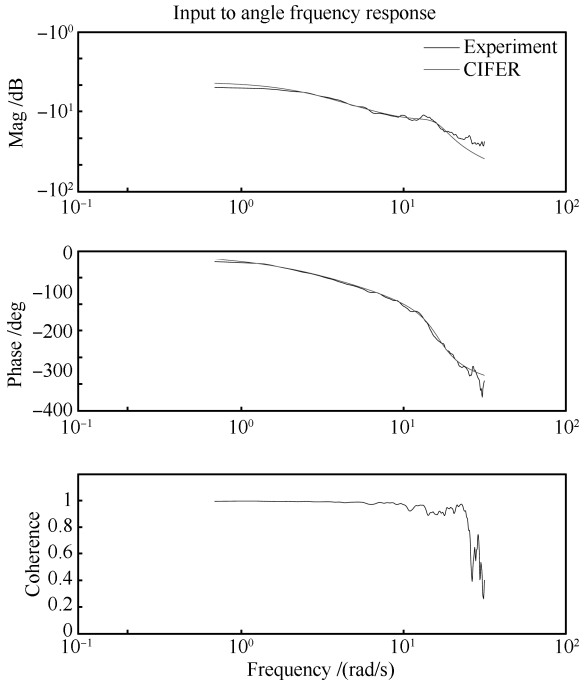


Fig. 6 Input-angle frequency response matching

Similarly, the transfer function from roll (or pitch) angle to the lateral (or longitudinal) velocity can be identified as a first-order transfer function shown below:

$$H_2 = \frac{8.661}{s + 0.09508} \tag{2}$$

As a result, the cascaded dynamics from the control input to the UAV′s linear velocity is a 5th-order system whose transfer function is the product of H_1 and H_2. It is observed that H_1 has its poles at $-4.0977 + 15.8272\,\text{i}$, $-4.0977 - 15.8272\,\text{i}$, -15.7414, -3.7433, while H_2 has its pole at -0.0958. Obviously, the pole of H_2 is much closer to the imaginary axis than any poles of H_1. Hence, the dynamics of the cascaded system from input to velocity is dominated by the pole of H_2. To simplify the control law design, we remove the poles from H_1 and keep the overall steady state gain the same. As a result, the simplified model from the control input to the linear velocity is

$$H_3 = \frac{5.3275}{s + 0.09508} \tag{3}$$

Note that this simplified model will only be used for control law design purpose. The full 5th-order complete model will still be used for flight simulation and control law verification.

For the yaw channel, since the inner-loop dynamics controlled by NAZA is very fast, we can safely assume that the control input directly corresponds to the yaw angular rate with a proportional gain. Thus, the transfer function from yaw control input to the yaw angle is an integration of a constant, shown as below:

$$H_4 = \frac{3.37025}{s} \tag{4}$$

The transfer function from the collective input δ_{col} to the z-axis velocity w is identified to be

$$H_5 = \frac{13.3506}{s + 2.32033} \tag{5}$$

The negative coefficient on the numerator of H_5 is due to the opposite definition of the positive control direction and the positive z-axis velocity direction.

The above sub-system models are identified with the presence of the inner-loop NAZA controller. The next step is to design a model based outer-loop control law to track a 3D position and heading reference. Again, since all channels′ dynamics are assumed to be decoupled, the outer-loop control laws can be designed separately. The control structure for each channel can be found in Figs. 7 ∼ 9. Except for the conventional state feedback and reference feedforward components, one special component of this control structure is that the derivative of the reference is also used. In this way, the UAV responds faster to reference changes and the time factor is emphasized. This is obviously an advantage for flying vehicle′s mission of trajectory tracking, but the position reference needs to be generated smooth enough for pragmatic implementation.

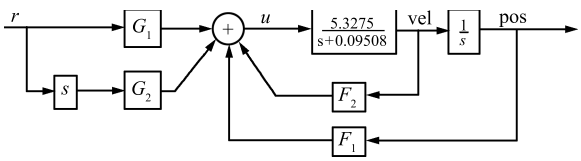


Fig. 7 x, y channel control structure

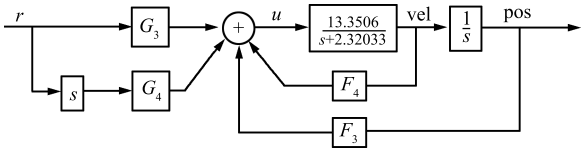


Fig. 8 z channel control structure

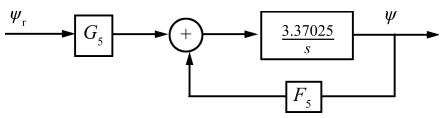


Fig. 9 Yaw channel control structure

3 State estimation with sensor fusion

While angle and angular rate measurements can be directly provided by the onboard IMU, the UAV's position and velocity need to be estimated indirectly. Since dead reckoning using IMU acceleration will face severe drifting problems, information from other sensors must be fused in to cure the divergence and at the same time, to reduce noises. The solution goes to the onboard camera and barometer. Moreover, a Kalman filter is formulated to fuse the information from all sensors and provide a smooth UAV state estimation for control purposes.

3.1 Velocity estimation by visual optical flow

For many times, optical flow based vision algorithms have been proposed by researchers to solve UAV GPS-denied navigation problems. Some researchers mount the UAV onboard camera pointing forward for obstacle avoidance, while some others mount the camera(s) sideward to navigate in indoor corridors or outdoor canyons. For our case, since there is already information from the onboard laser scanner which can detect frontal and sideward obstacles, it is preferable to mount the camera facing downward, so that both the x-axis and y-axis components of the UAV body frame velocity can be better estimated. This configuration also takes great advantage of the fact that the floor of an indoor environment is usually flat and vision algorithms involving feature points on the same plane can be largely simplified. In the computer vision society, there is the so-called structure from motion (SFM) problem, in which the camera translation and rotation can be calculated by given consecutive views of a scene. There are two fundamental sub-problems from SFM:

1) Correspondence (2D motion in the image);

2) Reconstruction (3D motion of the camera).

The first problem is commonly known as the 2D optical flow problem. It requires the knowledge of spatial and temporal image intensity derivatives. Let the grey-level intensity of a pixel on the image be $E(x, y, t)$. It is a continuous and differentiable function of space and time. Suppose the brightness pattern is locally displaced by a distance $\mathrm{d}x$, $\mathrm{d}y$ over time period $\mathrm{d}t$, the intensity of the displaced pixel should be the same as the original pixel:

$$E(x, y, t) = E(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t) \tag{6}$$

In other words, the total derivative of E w.r.t. time is zero.

$$\frac{\delta E}{\delta x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\delta E}{\delta y}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{\delta E}{\delta t} = 0 \tag{7}$$

If we denote

$$E_x = \frac{\delta E}{\delta x},\ \ E_y = \frac{\delta E}{\delta y},\ \ E_t = \frac{\delta E}{\delta t},\ \ v_x = \frac{\mathrm{d}x}{\mathrm{d}t},\ \ v_y = \frac{\mathrm{d}y}{\mathrm{d}t}$$

then the equation can be rewritten as

$$E_x v_x + E_y v_y + E_t = 0 \tag{8}$$

The above is known as the brightness constancy equation (BCE). In BCE, E_x, E_y and E_t are measurable, while v_x and v_y are the unknown 2D flows. One individual BCE is not enough to estimate the 2D flows. Solutions to this normally involve considering a small window of adjacent pixels. Among all these solutions, the Lucas & Kanade algorithm is the most classical and elegant one. It assumes that the motion field at a given time is constant over a block of pixels. For that particular block of n number of pixels, there are n BCEs. They form a linear over-determined equation set. Least square estimation can be used to obtain reliable v_x and v_y over that region.

To implement this 2D optical flow algorithm, the well-known OpenCV library from Intel can be used. OpenCV library also provides the "goodFeaturesToTrack" function which is able to find predefined number of feature points on an image and these feature points are selected particularly suitable for the later "calcOpticalFlowPyrLK" function, which calculates the optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. The implementation result can be found in Figs. 10 (a) ∼ 10 (d).

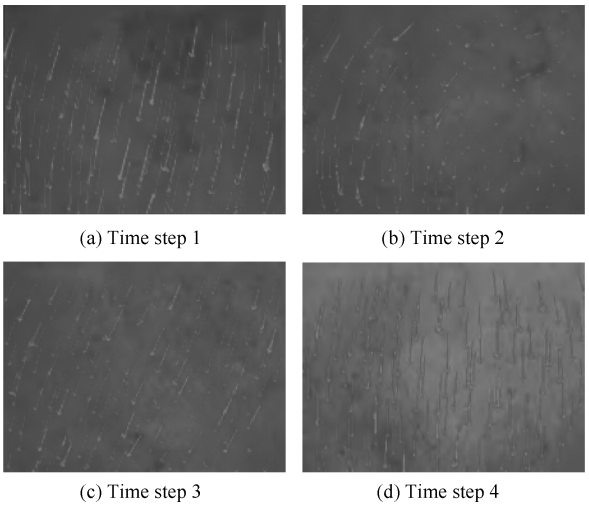


Fig. 10 The 2D optical flow implementation results

The next problem is to estimate the camera 3D motion from the 2D optical flow mentioned above. One of the most

elegant methods relies on an important concept called "homography". "Homography" is a well-known term in the computer vision society in describing the linear position relationship between feature points on two different (image) planes. Suppose a downward-looking camera on the UAV takes images of the ground during flight. Given two consecutive images taken at time t_1 and t_2, the corresponding visual features′ pixel positions in the 1st and 2nd images are related by a 3 by 3 matrix H, provided that the ground scene is within a level plane[12]. This H is called the "homography" matrix. The level plane assumption is quite reasonable for any indoor environment as the floor is usually manmade, thus flat. H carries useful information about the UAV motion from t_1 to t_2. If R and T are the inter-frame rotation and translation of the UAV from t_1 to t_2, N is the unit-length normal vector of the ground plane resolved in the camera frame at t_1, and d is the UAV altitude with respect to the ground plane, then the homography matrix H can be expressed as[13]

$$H = R + \frac{1}{d} T N^{\mathrm{T}} \tag{9}$$

The decomposition of H into R, T and N is possible but very complicated and it usually results in large numerical errors in practice. Fortunately, R and N are known in our case since the IMU will give Euler angle information at every moment. If we have UAV attitude angles ϕ_1, θ_1, ψ_1 at t_1 and ϕ_2, θ_2, ψ_2 at t_2, then

$$N = \begin{bmatrix} -\sin\theta_1 \\ \sin\phi_1 \cos\theta_1 \\ \cos\phi_1 \cos\theta_1 \end{bmatrix} \tag{10}$$

and

$$R = R_{\mathrm{b/n}}(t_2) R_{\mathrm{n/b}}(t_1) \tag{11}$$

where $R_{\mathrm{b/n}}$ is the rotational matrix to convert 3D points from the inertia frame to the UAV body frame and $R_{\mathrm{n/b}}$ is vice versa and they are transpose of each other. Hence, with H, R, d and N known, the translational motion, T can be calculated as

$$T = d(H - R)N \tag{12}$$

3.2 Height measurement from barometer

As mentioned previously, indoor UAV height measurement can be from various sensors. Using range sensor is one of the most direct ways to measure the distance from the UAV body to the ground. However, to obtain the UAV absolute height with respect to its initial position, this range sensor approach has to assume that the ground is without any altitude variations. Moreover, if the UAV is required to go through windows, this measurement will be inaccurate when the UAV is directly above the windowsill. In view of this, a more robust choice would be to use the barometer, as its measurement is independent of the environment′s physical shape. If a reference pressure (usually the pressure right before taking off), P_0, is selected, then the z direction position of the UAV at any later moment can be calculated by the following equation:

$$z_g = -44\,307 \left[1 - \left(\frac{P}{P_0} \right)^{0.1902} \right] \tag{13}$$

with P being the present pressure sensor output in Pascal.

3.3 Data fusion by Kalman filter

The Kalman filter framework describes a discrete-time linear system as follows

$$\begin{aligned} \boldsymbol{x}(k+1) &= A\boldsymbol{x}(k) + B(\boldsymbol{u}(k) + \boldsymbol{w}(k)) \\ \boldsymbol{y}(k) &= C\boldsymbol{x}(k) + \boldsymbol{v}(k) \end{aligned} \tag{14}$$

where $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{y}$ are the state, input and measurement vectors, respectively. A, B, C are system matrices with appropriate dimensions. $\boldsymbol{w}$ and $\boldsymbol{v}$ are input and measurement noises, which are assumed to be Gaussian with zero means. The main objective of Kalman filter is to estimate $\hat{\boldsymbol{x}}(k|k)$ at the time step k with the measurement $\boldsymbol{y}(k)$, input $\boldsymbol{u}(k-1)$ and the previously estimated state $\hat{\boldsymbol{x}}(k|k-1)$. If system 14 is observable, then the statistically optimal estimator is given as follows.

1) Time update:

$$\begin{aligned} \hat{\boldsymbol{x}}(k|k-1) &= A\hat{\boldsymbol{x}}(k-1) + B\boldsymbol{u}(k-1) \\ P(k|k-1) &= AP(k-1)A^{\mathrm{T}} + BQB^{\mathrm{T}} \end{aligned}$$

2) Measurement update:

$$\begin{aligned} H(k) &= P(k|k-1)C^{\mathrm{T}}(CP(k|k-1)C^{\mathrm{T}} + R)^{-1} \\ \hat{\boldsymbol{x}}(k) &= \hat{\boldsymbol{x}}(k|k-1) + H(k)(\boldsymbol{y}(k) - C\hat{\boldsymbol{x}}(k|k-1) \\ P(k) &= (I - H(k)C)P(k|k-1) \end{aligned}$$

To apply Kalman filter to this indoor UAV state estimation problem, the motion model and the measurement model need to be first defined. For the motion model, the simple point mass kinematics model can be used, that is, in the NED frame, position is integral of velocity and velocity is the integral of acceleration. The discrete representation of this motion model with a sampling frequency of 50 Hz is shown below:

$$\begin{aligned} \boldsymbol{x} &= \begin{pmatrix} x_g & y_g & z_g & u_g & v_g & w_g \end{pmatrix}^{\mathrm{T}} \\ \boldsymbol{u} &= \begin{pmatrix} ax_g & ay_g & az_g \end{pmatrix}^{\mathrm{T}} \\ \boldsymbol{y} &= \begin{pmatrix} z_g & u_g & v_g \end{pmatrix}^{\mathrm{T}} \end{aligned}$$

$$A=\begin{bmatrix} 1 & 0 & 0 & 0.02 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.02 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.02 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B=\begin{bmatrix} 0.0002 & 0 & 0 \\ 0 & 0.0002 & 0 \\ 0 & 0 & 0.0002 \\ 0.02 & 0 & 0 \\ 0 & 0.02 & 0 \\ 0 & 0 & 0.02 \end{bmatrix}$$

$$C=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

While the above system matrices have been numerically defined, the input noise and measurement noise matrices Q and R, being both diagonal, can be selected by logging real flight test data; the non-zero elements in Q represent the acceleration measurement noises and the non-zero elements in R represent the noise of z direction position measurement from barometer and velocity measurements from vision optical flow.

4 Wall following path plan

Most UAV path planning algorithms require the UAV to know the global map, i.e., its own position, the target′s location, and all obstacles′ positions presented in the global frame. However, the environment is usually unknown in practice and the exploration mission may not involve a well defined target. In these cases, two approaches are usually used. One is to build the unknown map through SLAM, so that all the conventional path planning algorithms can be applied afterwards. However, this kind of methods suffer from the problem of large computational burden, and it is very difficult to be implemented onboard and executed in real time on miniature indoor UAVs. The second approach is to use UAV local information only and dynamically calculate the tracking reference of the UAV, that means the reference generation is contemporary and only one step ahead. In this paper, the path planning strategy proposed follows the second approach, i.e., no global map or self-location information is needed. It is a universal strategy for any indoor enclosed environment with regular walls. First of all, this algorithm utilizes the concept of artificial potential field. All measurements from the laser scanner are treated as obstacles and they exert repulsive forces on the UAV body. Besides, to let the UAV keep moving forward, there is a constant attractive force coming from a virtual target two meters ahead from the UAV body. If the UAV is ordered to follow the left-side wall, then this virtual target is placed at the left-front of the UAV heading. If the UAV is order to follow the right-side wall, then the target is placed at the right-front of the UAV heading. The following formulation describes the aforementioned logic:

$$F=F_{\text{att}}-F_{\text{rep}} \tag{15}$$

where

$$F_{\text{att}}=\mathrm{e}^{-\frac{|T|^2}{2\sigma_1^2}}T \tag{16}$$

and

$$F_{\text{rep}}=\sum_{k=n_1}^{n_2}\mathrm{e}^{-\frac{|W(k)|^2}{2\sigma_2^2}}\frac{W(k)}{\sigma_2 K} \tag{17}$$

where F is the resultant force of the artificial potential field, F_{att} is the attractive force coming from the virtual target and F_{rep} is the repulsive force generated by the wall obstacles. T and $W(k)$ are the vectors pointing towards the virtual target and the points measured by the laser scanner. σ_1 and σ_2 represent the stiffness of the Gaussian-like potential fields which can be tuned for different indoor situations. In (17), all laser scanner measurements indexed from n_1 to n_2 will be examined one by one. Invalid measurements (in case of out of range) will be dropped, and K is the total number of valid measurements being used.

Although we call F a "force", it can be interpreted as other physical entities in practice. In this implementation, we let the UAV 2D velocity reference be proportional to F, and by integration, it also forms the 2D position reference. For the z direction, the UAV is ordered to maintain a predefined height with respect to the flat indoor floor. In addition, to determine the UAV heading reference, another algorithm is running at the same time to determine the UAV yaw angle reference at every scan moment, and it runs as follows:

1) If the UAV is to fly along the wall on its left, then omit the scanned points on the right side. If the UAV is to fly along the right side wall, then omit the scanned points on the left side.

2) For all remaining scanned points, calculate the best straight line fit using the least square optimization method.

3) The UAV heading reference deviates from the UAV current heading direction by a fraction (a%) of the difference between the fitted line gradient and the current UAV heading (a can be tuned for different indoor environments).

By combining the potential field algorithm with the line fitting algorithm, the UAV outer-loop references can be completely set up.

5 Customized FastSLAM algorithm

Many UAV indoor navigation applications need the unknown environment around the UAV path to be mapped. This makes the SLAM technology indispensable as a key component of the whole navigation system. Moreover, if detailed map information can be obtained, more intelligent path planning algorithm can be implemented. The dominant approach to the SLAM problem in literature uses the

extended Kalman filter (EKF) framework. However, EKF SLAM has disadvantages such as its quadratic complexity and sensitivity to data association failures. To overcome these problems, a lot of SLAM variants have been proposed by researchers. Some of them tried to exploit the sparsity of the matrix updating step in EKF[14−15]. Some have proposed more robust methods of data association[16−17]. Among them, the FastSLAM[9] is one of the most promising methods to improve both the robustness and efficiency of the algorithm. Unlike many other methods which factorize the SLAM problem spatially, FastSLAM factorizes the SLAM posterior over time using the path of the robot. The resulting algorithm scales logarithmically with the number of features in the map. In addition, FastSLAM originates from the particle filter, retaining different data association hypotheses to different particle solutions. The particles with wrong data associations can be completely forgotten in the long run.

In this work, the FastSLAM framework is adopted while the type of map features has been extended from the classical corner-only features to both corner and line features. The detailed algorithm can be found in [18]. The probability distribution of the UAV′s pose at time t is denoted as s_t. The covariance of the UAV pose is represented by a distribution of M particles, and it is assumed that the mth particle $P^{[m]}$ knows exactly where the vehicle′s position and orientation are, without uncertainty. Hence, instead of maintaining a single huge covariance matrix, such as that of the EKF SLAM, this algorithm has many small covariance matrices for each combination of the map features and possible UAV poses. This avoids the large matrix inversions, which is typically the most computationally expensive step in those conventional SLAM algorithms. Each particle carries its own map. Similar to the EKF SLAM approaches, each feature in the map is assumed to have a Gaussian distribution, which can be described by its mean $\mu_{n,t}^{[m]}$ and variance $\Sigma_{n,t}^{[m]}$, where n is the associated index of the map features. The following context will expand the customized FastSLAM algorithm by sequencing its main steps.

5.1 Feature extraction

The objective of this step is to convert a frame of raw laser scanner measurement points into a set of measurement features, $\{z_{1,t}, z_{2,t}, \cdots, z_{n,t}\} \in z_t$, and at the same time, to establish the probability distribution of the extracted features, $P(z_{i,t})$. As mentioned before, the features to be extracted are line segments and corners, and each of them can be represented by a vector mean and a square matrix covariance with the following notations with Fig. 11 providing a graphical illustration to all the parameter variables:

1) Lines:

$$z_l = \mathcal{N}(r, \theta; \mu_l^z, \Sigma_l^z) \tag{18}$$

$$\mu_l^z = \{\mu_r, \mu_\theta\} \tag{19}$$

$$\Sigma_l^z = \begin{bmatrix} \sigma_{rr} & \sigma_{r\theta} \\ \sigma_{\theta r} & \sigma_{\theta\theta} \end{bmatrix} \tag{20}$$

2) Corners:

$$z_c = \mathcal{N}(x, y, \alpha, \beta; \mu_c^z, \Sigma_c^z) \tag{21}$$

$$\mu_c^z = \{\mu_x, \mu_y, \mu_\alpha, \mu_\beta\} \tag{22}$$

$$\sigma_c^z = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\alpha} & \sigma_{x\beta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\alpha} & \sigma_{y\beta} \\ \sigma_{\alpha x} & \sigma_{\alpha y} & \sigma_{\alpha\alpha} & \sigma_{\alpha\beta} \\ \sigma_{\beta x} & \sigma_{\beta y} & \sigma_{\beta\alpha} & \sigma_{\beta\beta} \end{bmatrix} \tag{23}$$

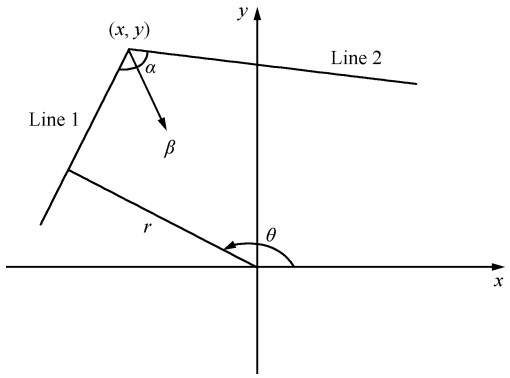


Fig. 11 Parameters to describe line and corner features

The process of converting a frame of raw scan points into a set of corners and line segments involves three steps, namely point clustering, line fitting and corner extraction. The clustering task is to group the raw data points in such a way that points within the same group belong to the same line segment as reasonable as possible. A recent review of line extraction algorithm[19] concludes that the split-and-merge and the Incremental are the two preferred methods, with split-and-merge being faster and Incremental being more robust. The split-and-merge method is more than sufficient for a clean indoor environment such as our test case and is therefore used. It is a recursive algorithm with the following steps:

Step 1. Start with all input points.

Step 2. Connect the first point and the last point with a line.

Step 3. Calculate the perpendicular distances of all other intermediate points with respect to the line segment obtained in the previous step.

Step 4. Search for the point that has the largest distance and compare this distance with a defined threshold.

Step 5. If the maximum distance is less than the threshold, all points between the first point and the last point belong to the same line; else, recursively call Step 1 with (first point, max point) and (max point, last point).

The next step, line fitting, aims to extract accurate line parameters and covariances. For this step, since the uncertainties of the points belong to the same line are different,

as a result of projecting elliptical shaped uncertainty (as the radial and angular components are different) of varied magnitude (measurements are more noisy for points at longer distance from the sensor), points have to be weighted when fitting to the line. Furthermore, as the weight of the point is dependent on the heading of the line, no close form formula has been found so far. Instead, an iterative method that maximizes the likelihood of the line has been derived in [20] and it is adopted in our work:

Step 1. Assume an initial heading of the line by connecting the first and the last points.

Step 2. Find the radial position and its variance based on the weight obtained by projecting uncertainties of the point onto the perpendicular direction of the line.

Step 3. Calculate the iterative incensement for heading.

Step 4. Repeat Step 1 to Step 3 until heading converges.

Step 5. Calculate the remaining terms in the covariance matrix for line parameters.

The detailed formulation of the iterative line parameter extraction and covariance estimation can be found in [20] and it will not be repeated here. However, the results can be appreciated in Figs. 12 and 13, where the extracted lines have dotted boundaries at both sides of the line, representing the 3-sigma uncertainty region.

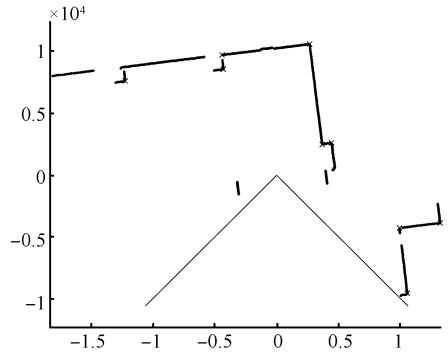


Fig. 12 Lines and corners extracted from laser scanner data

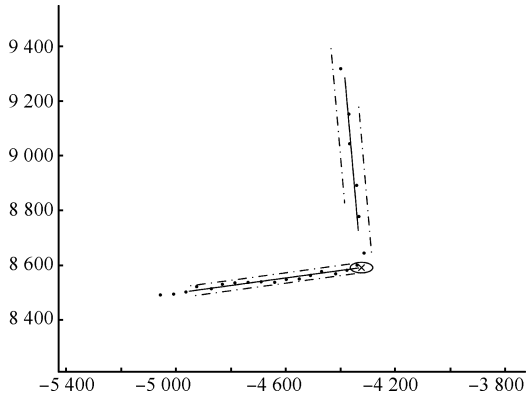


Fig. 13 Line and corner features with 3-sigma uncertainty

For corner extraction, adjacent lines are extended and their intersecting point is taken to be the position of the corners. Direction (β) and angle (α) of all corners are also computed from the line parameters. Covariance of the corner can be obtained from line covariances. Similar to line extraction, formulas to calculate the covariance of the corner from the information of the lines have been well derived in [21] and again will not be repeated here.

5.2 Motion estimation and proposal generation

This step gives a rough prediction about the UAV motion from the previous frame to the current frame. Concurrently, the uncertainty caused by this motion is propagated. The predicted body-frame displacement of the UAV position (Δx_t, Δy_t) and heading (Δc_t) are assumed to be random variables on their own and distributed normally around their expected values. So,

$$\Delta c_t = \mathcal{N}(\Delta \bar{c}_t, \sigma_{\Delta c}) \tag{24}$$

$$\Delta x_t = \mathcal{N}(\Delta \bar{x}_t, \sigma_{\Delta x}) \tag{25}$$

$$\Delta y_t = \mathcal{N}(\Delta \bar{y}_t, \sigma_{\Delta y}) \tag{26}$$

where $\Delta\bar{x}_t$, $\Delta\bar{y}_t$, $\Delta\bar{c}_t$ are the expected values and $\sigma_{\Delta x}$, $\sigma_{\Delta y}$, $\sigma_{\Delta c}$ are their respective standard deviations which can be defined by the following equations:

$$\sigma_{\Delta c} = \alpha_1 |\Delta \bar{c}_t| + \alpha_2 \tag{27}$$

$$\sigma_{\Delta x} = \alpha_3 |\Delta \bar{x}_t| + \alpha_4 \tag{28}$$

$$\sigma_{\Delta y} = \alpha_3 |\Delta \bar{y}_t| + \alpha_4 \tag{29}$$

where α_1, α_2, α_3, α_4 denote proportional and additive noise parameters that can be tuned for practical implementations.

There are various ways to obtain the average motion estimation, i.e., to calculate $\Delta\bar{x}_t$, $\Delta\bar{y}_t$, $\Delta\bar{c}_t$. Our solution is to use scan matching to estimate the displacement of the UAV by comparing position of corner features extracted in two adjacent scans. This rigid motion involves a rotation R and a translation T and they can be obtained via a closed-form solution as follows:

Step 1. Check corner feature correspondences based on their pair-wise Mahalanobis distances.

Step 2. Organize corner features in such a way that p_i or $[px_i \;\; py_i]^{\mathrm{T}}$ in the previous frame corresponds to q_i or $[qx_i \;\; qy_i]^{\mathrm{T}}$ in the current frame.

Step 3. Calculate the centroid of the feature points for both frames, denoted by $\bar{p}$ and $\bar{q}$.

Step 4. Form matrix $P = [p_1 - \bar{p}, \;\; p_2 - \bar{p}, \;\; \ldots \;\; , \;\; p_{\max} - \bar{p}]$.

Step 5. Form matrix $Q = [q_1 - \bar{q}, \;\; q_2 - \bar{q}, \;\; \ldots \;\; , \;\; q_{\max} - \bar{q}]$.

Step 6. Let $[U, S, V] = \mathrm{SVD}(PQ^{\mathrm{T}})$ and $d = \mathrm{sgn}(\det(PQ^{\mathrm{T}}))$.

Step 7. Then $R = V[1 \;\; 0; \;\; 0 \;\; d]U^{\mathrm{T}}$ and $T = R\bar{p} - \bar{q}$, leads to $\Delta\bar{c}_t = -\mathrm{atan2}(R(1), R(2))$, $\Delta\bar{x}_t = T(1)$, $\Delta\bar{y}_t = T(2)$.

The motion estimation result will be applied to all particles with random additive and multiplicative noises. So for particle $P^{[m]}$,

$$\Delta c_t^{[m]} = \Delta \bar{c}_t(1 + \alpha_1 \text{randN}(1)) + \alpha_2 \text{randN}(1) \tag{30}$$

$$\Delta x_t^{[m]} = \Delta \bar{x}_t(1 + \alpha_3 \text{randN}(1)) + \alpha_4 \text{randN}(1) \tag{31}$$

$$\Delta y_t^{[m]} = \Delta \bar{y}_t(1 + \alpha_3 \text{randN}(1)) + \alpha_4 \text{randN}(1) \tag{32}$$

where randN(1) represents a function that can generate a random value from a standard normal distribution, and the updating equations are as follows:

$$P_t^{[m]}.c = P_t^{[m]}.c + \Delta c_t^{[m]} \tag{33}$$

$$P_t^{[m]}.x = P_t^{[m]}.x + \cos(P_t^{[m]}.c)\Delta x_t^{[m]} - \sin(P_t^{[m]}.c)\Delta y_t^{[m]} \tag{34}$$

$$P_t^{[m]}.y = P_t^{[m]}.y + \sin(P_t^{[m]}.c)\Delta x_t^{[m]} + \cos(P_t^{[m]}.c)\Delta y_t^{[m]} \tag{35}$$

5.3 Per-particle data association

The objective of this step is to find data pairs that associate contemporary measurement features with the existing features in the map. There is a wide range of data association algorithms available, among which the following three classes are the most popular.

The first class considers each feature independently. It aims to maximize the probability of associating map features with measured features without excluding repeated matches. The individual compatibility nearest neighbor (ICNN) algorithm is one of the famous examples. On the other hand, the second class makes sure the associations are consistent in a sense that no duplicated associations can be possibly made for the same feature. Such algorithms include sequential compatibility nearest neighbor (SCNN) and joint compatibility branch and bound (JCBB). The third class of data association algorithms takes one step further. It considers data associations in the previous iteration. In other words, it tries to maximize the whole probability history. When making data associations, previous decisions are examined and modified. Examples of such algorithms include the tree-structured searching algorithm developed in [22].

Not surprisingly, algorithms of lower complexity do not produce as robust results as compared to that by complex algorithms. Hence, choosing an algorithm that balances well between computational load and robustness is a critical task in the entire implementation of the SLAM algorithm. At the moment, the JCBB method is used for data association in this work, as it provides a robust yet efficient solution.

5.4 Per-particle measurement update

Since every particle has its own map and the estimations of landmark features in each map are conditioned on the corresponding particle′s path, there are $(N_c + N_l)$ low-dimensional EKFs attached to each particle, where N_c and N_l are the number of corner features and number of line features in the map, respectively. Moreover, we need to do map update for all M particles, that means there are in total $(N_c+N_l)\times M$ EKFs, and for each of them, the updating rule runs as follows (line segment features and corner features are updated with the same formula but with different dimensions):

$$Z_{n,t} = \Sigma_{n,t-1}^{[m]} + \Sigma_{n,t}^{z} \tag{36}$$

$$K_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]} Z_{n,t}^{-1} \tag{37}$$

$$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]} + K_{n,t}^{[m]}(z_{n,t} - \hat{z}_{n,t}) \tag{38}$$

$$\Sigma_{n,t}^{[m]} = (I - K_{n,t}^{[m]})\Sigma_{n,t-1} \tag{39}$$

5.5 Importance weighting and resampling

Samples from the proposal distribution, i.e., particles after motion update, are distributed according to $p(s^t|z^{t-1}, u^t)$, where x^t means all the time history of x: $\{x_1, x_2, \cdots, x_t\}$. This distribution most likely does not match the posterior probability $p(s^t|z^t, u^t)$. Importance weighting is to correct this difference by giving each particle a weight according to its probability of observing z_t at the time step t. So in a Gaussian distribution case,

$$w_t^{[m]} = (2\pi Z_{n,t})^{-\frac{1}{2}} e^{-\frac{1}{2}(z_{n,t}-\hat{z}_{n,t})^{\mathrm{T}}[Z_{n,t}]^{-1}(z_{n,t}-\hat{z}_{n,t})} \tag{40}$$

After the particles have been assigned with their corresponding weights, a new set of samples can be drawn from the original set with probabilities in proportion to their weights.

6 Implementation results

Actual flight tests have been carried out to verify the complete UAV indoor navigation strategy. The quadrotor platform, equipped with IG500N IMU, PointGrey FireFly camera and UTM-30LX laser scanner, performed an autonomous wall following flight in an indoor hall successfully. Fig. 14 sequentially shows the eight instances of the flight, with the left photo showing the actual flying condition and the right sub-figure showing the laser scanner data at that particular moment.

Another flight test was carried out in the same indoor hall, but remotely controlled by a human pilot. After one complete flight, the onboard laser scanner logged data were ported to a desktop computer and off-line processing was done to verify the customized FastSLAM algorithm. Fig. 15 (a) ∼ 15 (f) show six moments of the whole journey. For each moment, the left sub-figure shows the UAV body-frame laser scanner raw data and the extracted line and corner features. The right sub-figure shows the UAV pose and the map built in progress. The UAV pose is indicated by a cloud of crosses, representing the particles. It can be seen that the line segments and corners naturally form a map of the indoor environment with straight walls and sparsely distributed pillars. The result is accurate as compared to the real environment.

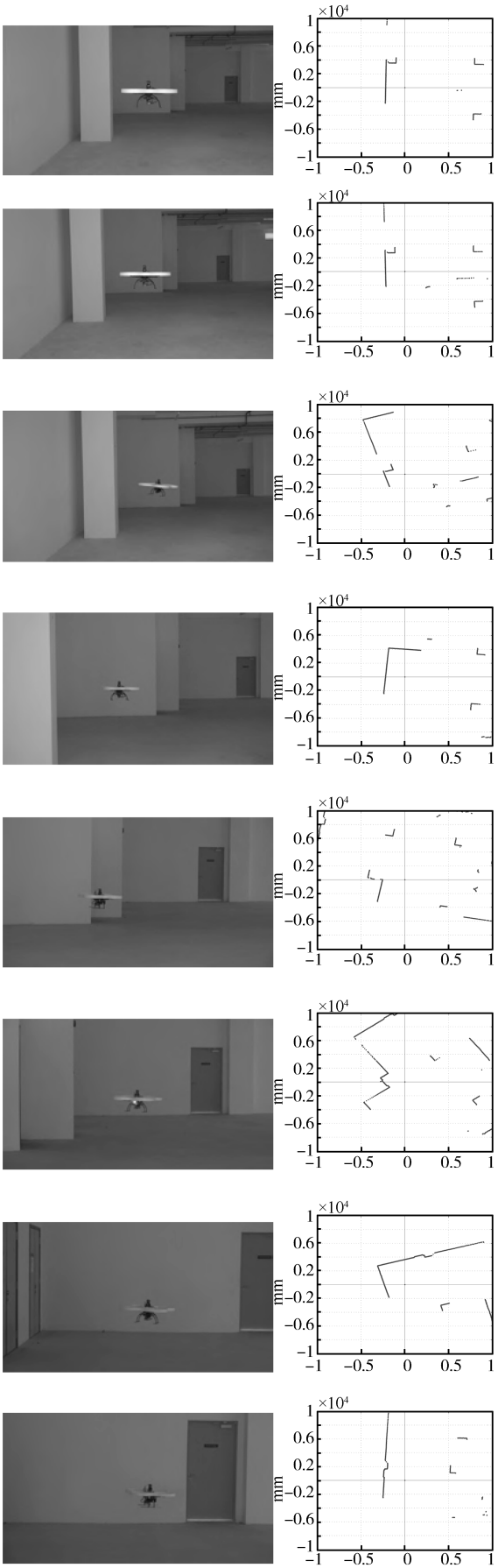


Fig. 14 Instances of the wall following flight test

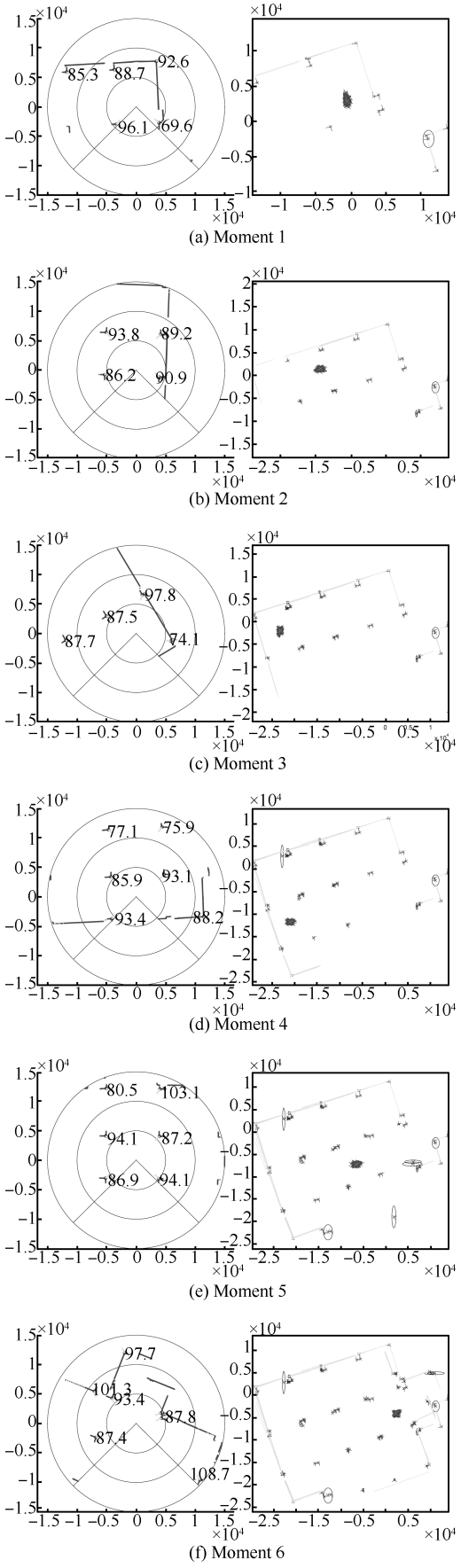


Fig. 15 Customized FastSLAM result in the indoor hall with pillars

7 Conclusions

In conclusion, this paper has proposed a complete navigation scheme for an indoor quadrotor UAV system by selecting, customizing, and combining suitable existing algorithms. Innovations have been focussing on improving algorithm efficiency and system integration. Real-life flight tests have been carried out to verify the fidelity and performance of the overall system. The greatest advantage of this scheme is its minimal requirement on the onboard computational power. The UAV, after being issued with the main navigation command, does not need to maintain any wireless link to the ground control station. Furthermore, although the testbed platform is a quadrotor UAV, the proposed scheme is also applicable to other miniature UAV platforms as long as they have IMU, mono-camera and scanning laser range finder mounted onboard.

References

1 Pounds P, Mahony R, Corke P. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 2010. **18**(7): 691−699

2 Tayebi A, McGilvray S. Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 2006, **14**(3): 562−571

3 Kim J, Kweon I S. Vision-based autonomous navigation based on motion estimation. In: Proceedings of the 2008 International Conference on Control, Automation and Systems. Seoul: IEEE, 2008. 1738−1743

4 Liu Y C, Dai Q H. Vision aided unmanned aerial vehicle autonomy: an overview. In: Proceedings of the 3rd International Congress on Image and Signal Processing. Yantai, China: IEEE, 2010. 417−421

5 Chen Y C, Zhao Y, Wang H K. Real time path planning for UAV based on focused D. In: Proceedings of the 4th International Workshop on Advanced Computational Intelligence. Wuhan, China: IEEE, 2011. 80−85

6 Kang K, Prasad J V R. Development and flight test evaluations of an autonomous obstacle avoidance system for a rotary-wing UAV. *Unmanned Systems*, 2013, **1**(1): 3−19

7 Keller J, Thakur D, Dobrokhodov V, Jones K, Pivtoraiko M, Gallier J, et al. A computationally efficient approach to trajectory management for coordinated aerial surveillance. *Unmanned Systems*, 2013, **1**(1): 59−74

8 Moutarlier P, Chatila R. Stochastic multisensory data fusion for mobile robot location and environment modeling. In: Proceedings of the 5th International Symposium on Robotics Research. Cambridge, MA: MIT Press, 1989. 207−216

9 Montemerlo M, Thrun S. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics (Springer Tracts in Advanced Robotics)*. New York: Springer, 2007

10 Bachrach A G. Autonomous Flight in Unstructured and Unknown Indoor Environments [Master dissertation], MIT, Cambridge, MA, 2009

11 Shen S J, Michael N, Kumar V. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation. Shanghai, China: IEEE, 2011. 20−25

12 Ma Y, Soatto S, Kosecka J, Sastry S S. *An Invitation to 3-D Vision*. New York: Springer, 2004

13 Hartley R, Zisserman A. *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press, 2004

14 Guivant J E, Nebot E M. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 2001, **17**(3): 242−257

15 Thrun S, Liu Y, Koller D, Ng A Y, Ghahramani Z, Durrant-Whyte H. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 2002. **23**(7-8): 693−716

16 Bar-Shalom Y, Fortmann T. *Tracking and Data Association*. San Diego, CA, USA: Academic Press, 1988

17 Neira J, Tardos J D. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 2001, **17**(6): 890−897

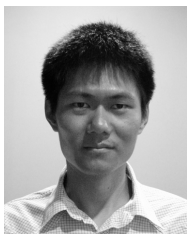
18 Hon B X, Tian H, Wang F, Chen B M, Lee T H. A customized fastslam algorithm using scanning laser range finder in structured indoor environments. In: Proceedings of the 10th IEEE International Conference on Control and Automation. Hangzhou, China: IEEE, 2013. 640−645

19 Nguyen V, Martinelli A, Tomatis N. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In: Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robotics and Systems. Edmonton, Alta: IEEE, 2005. 1929−1934

20 Pfister S T, Roumeliotis S I, Burdick J W. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation. Taipei, China: IEEE, 2003. 1304−1311

21 Núñez P, Vázquez-Martín R, del Toro J C, Bandera A, Sandoval F. Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation. *Robotics and Autonomous Systems*, 2008, **56**(3): 247−264

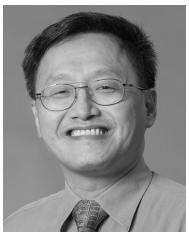
22 Hähnel D, Burgard W, Wegbreit B, Thrun S. Towards lazy data association in SLAM. In: Proceedings of the 11th International Symposium of Robotics Research. Sienna, Italy: Springer, 2003. 421−431



WANG Fei Received his bachelor degree with first class honors in the Department of Electrical and Computer Engineering, National University of Singapore (NUS), Singapore, in 2009. He has joined the NUS unmanned aircraft system (UAS) team from his 4th year undergraduate study while doing his final year project. He is currently pursuing his Ph. D. degree at the NUS Graduate School for integrative sciences and engineering. He is now working on a project related to UAV 3D indoor navigation system. His research interest covers unmanned aerial vehicles, simultaneous localization and mapping, computer vision and UAV indoor navigation systems.
E-mail: wangfei@nus.edu.sg

CUI Jin-Qiang Received his bachelor and master degrees in mechatronic engineering from Northwestern Polytechnical University, Xi′an, China, in 2005 and 2008 respectively. He has been working on MEMS capacitive and piezoresistive sensor design and gyroscope readout circuit design before joining NUS, Singapore. Currently he is pursuing his Ph. D. degree in electrical and computer engineering at the National University of Singapore, Singapore. His current research interest covers UAV navigation in foliage environment using LiDAR and vision sensing technologies. E-mail: jinqiang@nus.edu.sg

CHEN Ben-Mei Received his bachelor degree in mathematics and computer science from Xiamen University, China, in 1983, master degree in electrical engineering from Gonzaga University, USA, in 1988, and Ph. D. degree in electrical and computer engineering from Washington State University, USA, in 1991. He was a software engineer from 1983 to 1986 in South-China Computer Corporation, Guangzhou, China, and was an assistant professor from 1992 to 1993 at the State University of New York at Stony Brook, USA. Since August 1993, he has been with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, where he is currently a professor. His current research interest covers systems theory, robust control, unmanned aerial systems, and financial market modeling. Corresponding author of this paper.
E-mail: bmchen@nus.edu.sg



LEE Tong H Received his bachelor degree with first class honors in the engineering tripos from Cambridge University, England, in 1980; and the Ph. D. degree from Yale University, USA, in 1987. He is a professor in the Department of Electrical and Computer Engineering at the National University of Singapore (NUS), Singapore, and also a professor in the Graduate School for Integrative Sciences and Engineering, NUS, Singapore. His research interest covers adaptive systems, knowledge-based control, intelligent mechatronics and computational intelligence.
E-mail: eleleeth@nus.edu.sg