J Intell Robot Syst (2016) 84:259–276
DOI 10.1007/s10846-015-0292-1

Autonomous Navigation of UAV in Foliage Environment

Jin Q. Cui · Shupeng Lai · Xiangxu Dong · Ben M. Chen

Received: 16 December 2014 / Accepted: 8 October 2015 / Published online: 29 October 2015

Abstract This paper presents a navigation system that enables small-scale unmanned aerial vehicles to navigate autonomously using a 2D laser range finder in foliage environment without GPS. The navigation framework consists of real-time dual layer control, navigation state estimation and online path planning. In particular, the inner loop of a quadrotor is stabilized using a commercial autopilot while the outer loop control is implemented using robust perfect tracking. The navigation state estimation consists of real-time onboard motion estimation and trajectory smoothing using the GraphSLAM technique. The onboard real-time motion estimation is achieved by a Kalman filter, fusing the planar velocity measurement from matching the consecutive scans of a laser range finder and the acceleration measurement of an inertial measurement unit. The trajectory histories from the real-time autonomous navigation together with the observed features are fed into a sliding-window based pose-graph optimization framework. The online path planning module finds an obstacle-free trajectory based the local measurement of the laser range finder. The performance of the proposed navigation system is demonstrated successfully on the autonomous navigation of a small-scale UAV in foliage environment.

Keywords Unmanned aerial vehicle · Simultaneous localization and mapping · Path planning · GPS-less navigation

J. Q. Cui (✉) · X. Dong
Temasek Laboratories, National University of Singapore, Singapore, Singapore
e-mail: jinqiang@u.nus.edu

X. Dong
e-mail: tsldngx@nus.edu.sg

S. Lai
NUS Graduate School for Integrative Sciences, Engineering, National University of Singapore, Singapore, Singapore
e-mail: shupenglai@nus.edu.sg

B. M. Chen
Department of Electrical, Computer Engineering, National University of Singapore, Singapore, Singapore
e-mail: bmchen@nus.edu.sg

1 Introduction

Navigation of mobile robotics platforms in GPS-denied environments is being intensively studied in the research community, such as indoor offices [22, 25], underwater [19] and urban canyons [9]. This paper presents the autonomous navigation of a small-scale unmanned aerial vehicle (UAV) in foliage environment. Navigation of ground vehicles in foliage environment has been addressed in [10, 11] where a car equipped with a laser range finder drove through Victoria park in Sydney, Australia. The steep terrain, thick understorey vegetation and abundant debris characteristic of many forests prohibit the deployment of an autonomous ground vehicle in such scenario. A UAV

with autonomous navigation capability would be of paramount importance in forest survey, exploration and reconnaissance [3, 5].

The idea of autonomous flight of UAV in forest has been attempted using a low-cost inertial measurement unit (IMU) and a monocular camera [14], in which an unscented Kalman filter (UKF) was used to estimate the locations of obstacles and the state of UAV. Experiment verification was carried out with a remote control car running in synthetic outdoor environment. More recently, Ross et al. [21] realized autonomous flight through forest by mimicking the behavior of human pilots using a novel imitation learning technique. The application of learning technique is innovative but the system suffers from relatively high failure rate which the UAV can not afford.

To the best of our knowledge, this paper presents the first successful autonomous navigation of a small-scale UAV in unknown foliage environment. The navigation framework consists of real-time dual layer control, onboard motion estimation and pose-graph optimization based on GraphSLAM [23] and online path planning [12]. The whole navigation system is implemented on a quadrotor UAV equipped with a laser range finder and an IMU. The modular design of the framework makes it readily applicable to other mobile robotic navigation system in obstacle-strewn environment without GPS.

The remaining part of the manuscript is organized as follows: Section 2 presents the system structure, including the hardware and software structure respectively. Section 3 introduces the dynamics model structure of the UAV and presents the design of a robust perfect tracking control law. Section 4 presents the state estimation framework consisting of real-time motion estimation for autonomous control and the pose-graph optimization based on GraphSLAM. Section 5 presents the online path planning with only the local measurement of the laser range finder. Section 6 presents the experimental results regarding the autonomous flight and pose-graph optimization in outdoor environment. Finally, Section 7 concludes the paper and gives direction for future research.

2 Unmanned System Structure

The system structure of the UAV includes the hardware platform configuration, the software structure and the navigation system structure. Each of the three parts is indispensable for realizing a successful autonomous flight in GPS-denied environment. We will present these structures in this section and highlight the motivation for using such configurations.

2.1 Navigation System Structure

The navigation system of UAV is depicted in Fig. 1, including functional modules such as the UAV dynamics, autonomous control, state estimation and path planning. We first report the techniques used in each module in order to realize the autonomous flight of UAV in forest. More details of each part will be covered in latter sections.

To maneuver in forests consisting of a large number of obstacles, the platform has to be compact in size and is capable of hovering in the air. Platforms like quadrotors, helicopters and coaxial rotors all fulfill such requirements. In this paper, we use a quadrotor due to its unique advantages, such as the symmetric structure, the easy assembly of payload and the vast availability of commercial off-the-shelf (COTS) autopilot. The dynamics of a quadrotor consists of the inner loop dynamics and the outer loop dynamics. The inner loop dynamics relates the control input to the

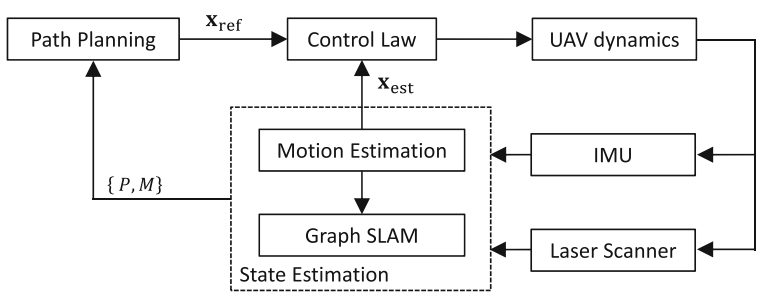


Fig. 1 System diagram of UAV navigation system

angular motion while the outer loop dynamics relates the angular motion to the linear velocity and position. We utilize a COTS autopilot to stabilize the inner loop dynamics and identify the outer loop dynamics in the frequency domain.

After the inner loop dynamics is stabilized, we design a Robust and Perfect Tracking (RPT) control law [16] to control the UAV outer loop to track a trajectory reference generated from the path planning module. The RPT problem is to design a controller such that the resulting closed-loop system is asymptotically stable and the controlled output almost perfectly tracks a given reference signal in the presence of any initial conditions and external disturbances. The almost perfect tracking means the ability of a controller to track a given reference signal with arbitrarily fast settling time despite of external disturbances and initial conditions.

The foliage environment renders the GPS signal unreliable, making the state estimation in such environment a challenging problem. We design the state estimation to include two parts: motion estimation and GraphSLAM [23]. The motion estimation is essentially a laser odometry technique, which matches the consecutive laser scans to generate the 2D velocity estimation. Combined with the acceleration measurement of IMU, a Kalman filter is designed to estimate the position and velocity of the UAV, which are used directly for the closed-loop control. On the other hand, the position estimate from the motion estimation is prone to drift, thus GraphSLAM is used as a post optimization process to achieve the optimal trajectory and consistent mapping.

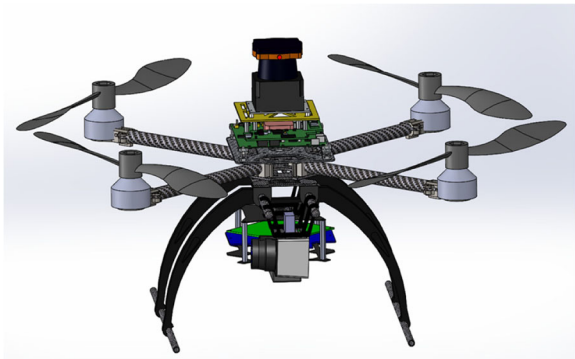


Fig. 2 The virtual design of the quadrotor UAV

In unknown obstacle-strewn environment, the path planning part plays a critical role in providing a trajectory reference which is both obstacle-free and meets the dynamics constraints of the UAV. Without a global map, the path planning needs to rely on only the local measurement of each scan to extract such a feasible path. The measurement of the laser scanner is inherently realized in polar coordinate, thus a polar grid map is first built based on the current laser scan measurement. In the polar grid map, by setting a starting point and a goal point, we use A* searching algorithm to find a series of line segments. The first sharp turning point in the line segments is selected as the local target. With the current starting point and the local target, the trajectory generation problem is treated as a two point boundary value problem for a triple integrator with constrained states and input. Details of the path planning will be covered in Section 5.

2.2 Hardware System Structure

The quadrotor is a fully customized platform (Figs. 2–3) designed by the NUS UAV Team. The platform is configured to be applicable in both indoor and outdoor environments, such as modern offices and forests. The platform is composed of carbon fiber plates and rods with a durable acrylonitrile butadiene styrene (ABS) landing gear to reduce the bare platform weight. The overall dimensions are 35 cm in

Fig. 3 NUS quadrotor platform with two onboard laser range finders

height and 86 cm from tip-to-tip. Different configurations of the rotor blade and the motor are compared before an optimal design is achieved. The motors used for the platform are 740 KV T-Motors with Turnigy Plush - 25 A Bulletproof electronic speed controllers. The propellers are APC 12×3.8 clockwise and anticlockwise fixed pitch propellers. Each motor and propeller setup could generate 15 kN static thrust. The final bare platform's main body weighs 1 kg. Its maximum total take-off weight reaches 3.3 kg with a 4 cell 4300 mAh lithium polymer battery. We have tested that the platform was able to fly at 8 m/s for a period of 10 to 15 minutes depending on the total weight and the battery volume.

The platform is also fully customizable in terms of sensor arrangement and is scalable such that additional computational boards could be mounted with a stack-based design. As shown in Fig. 3, the platform is equipped with two laser range finders. The above one is Hokuyo UTM-30LX, used to detect the environment in the horizontal plane. The bottom one is Hokuyo URG-04LX, used to scan the vertical plane to measure the height of the UAV in complex terrain conditions. One noteworthy thing is that the whole avionics system is mounted on the platform through four mechanical isolators (CR1-100 from ENIDINE). Experiment results show that the noise of acceleration measurements in x, y, z axis of the IMU decreases by 5 times compared with that without any vibration isolation. The vibration isolation is also beneficial for the laser range finder which can only withstand 20 g shock impact for 10 times.

2.3 Software System Structure

Considering the comprehensive functions and logics implemented on UAV onboard system, it is further structured into two main modules given the hierarchical property of navigation and control. As shown in Fig. 4, two onboard processors are adopted exclusively for each modules: *Mission plan processor* and *Flight control processor*. As mission plan tasks normally involve computationally intensive algorithms such as path planning, obstacle avoidance and SLAM, a high-end powerful Intel Core i7 based processor called Mastermind (from Ascending Technologies Germany) with Ubuntu 12.04 is deployed as the *mission plan processor*. The Ubuntu operating system has mature development environment with rich libraries for robotics applications, which can facilitate the overall development. For the critical flight control, a lightweight yet powerful OMAP3530 based Computer-On-Module (COM) called Gumstix Overo Fire is adopted. The flight control system

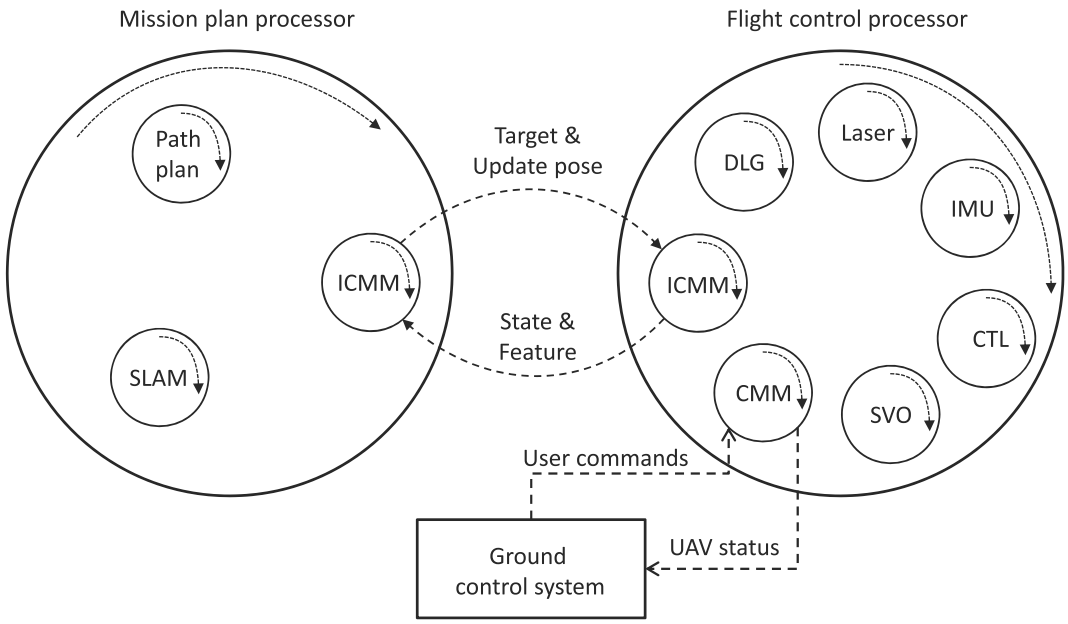


Fig. 4 Software structure of the UAV navigation system. Robust perfect tracking control is implemented in *CTL*, and scan matching in *Laser*, Kalman filter in *IMU*, GraphSLAM in *SLAM* and obstacle avoidance in *Path plan*

is implemented based on QNX Neutrino real-time operating system (RTOS). QNX RTOS is developed with a true microkernel architecture which integrates only the fundamental services including CPU scheduling, interprocess communication, interrupt and timers. Drivers and user applications are all executed as user processes. This architecture can provide a quite small yet fully customizable and manageable user application suits with necessary drivers and libraries.

Based on the specifications from the system structure, tasks to realize the flight missions are examined. The tasks are assigned, from the high level navigation to the low level flight control, into *Mission plan processor* and *Flight control processor* respectively. Since the Mastermind processor possesses powerful processing capabilities, high level tasks such as *SLAM* and *Path planning* are scheduled. For the flight control subsystem, its subtasks are scheduled into the following order to achieve the closed-loop control system. Navigation sensors are retrieved first with *Laser* and *IMU*. With the laser data, the scan matching is performed on the two consecutive scan data, generating incremental rotation and translation estimates. After being fused with the acceleration measurement of the IMU in a Kalman filter, the incremental translation and rotation produce the navigation state estimates which are further used for the control task *CTL*. With the generated automatic control signal, motor driving signals are sent to the UAV from the *SVO* task to realize 6 degree of freedom (DOF) movement. Other auxiliary tasks are also implemented: the communication task *CMM* is used to send status data back to Ground Control System (GCS) for user monitoring and receiving user commands, the data logging task *DLG* is used to record flight status data for post flight analysis. Finally, to pass high level navigation data to *Flight control processor* and share UAV status with *Mission plan processor*, the inter-processor communication task *ICMM* is implemented on both processors.

All the tasks are scheduled in a periodic fashion, whose executions follow the order in Fig. 4. On *Flight control processor*, all the tasks are scheduled in 50 Hz. As high level data is only for navigation purpose, a relative low scheduling frequency of 10 Hz is implemented on *Mission plan processor*. *CMM* and *DLG* are executed every one second to fully utilize the efficiency of the processor.

3 Modeling and Control

3.1 Model Structure

Following the routines of traditional aircraft model [2, 20], the model structure of the quadrotor platform is separated into the inner loop model and the outer loop model, as illustrated in Fig. 5. The 6-DOF dynamics of quadrotor consists of the 3-DOF translation dynamics and the 3-DOF angular dynamics. Since the angular dynamics of quadrotor is much faster than its translation dynamics, it is practical to extract the angular dynamic model and the translational dynamic model. The separation of the model structure facilitates the design of separate controllers for the inner-loop model and the outer-loop model.

In the current system hardware configuration, the inner-loop attitude dynamics is stabilized using a commercial autopilot 'NAZA-M', which is an all-in-one stability controller for multi-rotor platforms. The control inputs $(\delta_{\text{lat}}, \delta_{\text{lon}}, \delta_{\text{col}}, \delta_{\text{ped}})$ from the remote transmitter are fed into the onboard autopilot. Then 'NAZA-M' controller generates pulse width modulation (PWM) signals to drive the four rotors to generate the thrust forces. The four combined thrusts lift the platform and maintain the attitude stability at the same

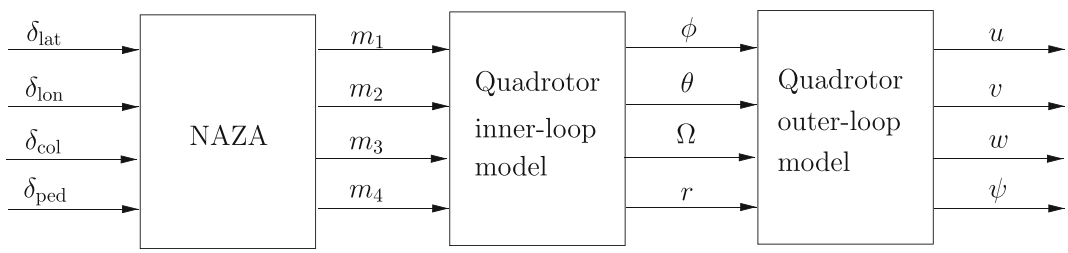


Fig. 5 Model structure of the quadrotor UAV

time. From the perspective of 'NAZA-M', the four inputs from the remote transmitter correspond to the control references for the roll angle (ϕ), the pitch angle (θ), the yaw angular rate (r), and the average motor speed (Ω) respectively.

In the outer loop aspect, the states regarding linear velocity (u, v, w) and heading angle ψ are of concern. In the lateral and longitudinal dynamics, there is a dynamics model which maps the roll and pitch angle to the lateral and longitudinal velocity respectively, which is governed by the rigid body kinematics with some damping effect from the air resistance. The heave velocity w and heading angle ψ are governed directly by the inner loop controller.

Without knowledge of the internal structure of firmware in 'NAZA-M', we treat the inner loop as a black box and identify the inner loop model. This is essential since we need to know the bandwidth of the inner loop model to facilitate the design of the outer loop controller. The inner model can be decoupled into four input/output pairs. Due to the symmetric structure of the platform, we note that the dynamics model in the roll and pitch directions share the same set of equations. Referring to Fig. 5, the inner loop model can be separated into three groups: the roll/pitch dynamics, the yaw dynamics and the heave dynamics.

The inner loop model identification is performed in the frequency domain. First, the platform with the inner loop controller is perturbed in all directions during which the corresponding input/output are logged. Then the logged data are fed into a software called CIFER [1] to derive a transfer function which matches the best with the logged data [25]. The identified sub system models are listed as follows:

– Roll/Pitch dynamics: input $\delta_{\text{lat}}/\delta_{\text{lon}}$, output ϕ/θ
Transfer function:

$$H_1(s) = \frac{9688}{s^4 + 27.68\,s^3 + 485.9\,s^2 + 5691\,s + 15750}. \tag{1}$$

– Yaw dynamics: input δ_{ped}, output ψ
Transfer function:

$$H_2(s) = \frac{3.372}{s}. \tag{2}$$

– Heave dynamics: input δ_{col}, output w
Transfer function:

$$H_3(s) = \frac{-13.35}{s + 2.32}. \tag{3}$$

3.2 Outer Loop Control

In obstacle-strewn environments, fast and accurate maneuvering of the UAV is required to avoid any possible collision. This poses challenges for the design of outer loop controller. We need to control the UAV to follow external references including the linear position and the heading angle. In order to perform fast and precise tracking of the given references, the RPT controller is adopted from [4, 16]. The procedures of designing an RPT controller for the state feedback case has been addressed in [16] and a case study is exemplified in [24].

For precision control, it's desirable to include an integrator to ensure zero steady state error in case of step input. We propose the RPT controller which considers the integration of position tracking error as an augmented state. The system with state-augmentation is formulated as:

$$\Sigma^{\text{xy}}_{\text{AUG}}: \begin{cases} \dot{\widetilde{\mathbf{x}}}_{\text{xy}} = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \widetilde{\mathbf{x}}_{\text{xy}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u}_{\text{xy}} \\ \widetilde{\mathbf{y}}_{\text{xy}} = \widetilde{\mathbf{x}}_{\text{xy}} \\ \widetilde{\mathbf{h}}_{\text{xy}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \widetilde{\mathbf{x}}_{\text{xy}} \end{cases} \tag{4}$$

where $\widetilde{\mathbf{x}}_{\text{xy}} = [\int e \quad r_p \quad r_v \quad r_a \quad p \quad v]^T$; r_p, r_v, r_a are the position, velocity and acceleration references; p, v are the actual position and velocity; $e = p - r_p$ is the tracking error of position. Since there is error integration $\int e$ in the augmented states, the feedback control law would contain a term of $K_i \int e$. Following the steps in [16], a linear state feedback control law of the form (5) is acquired,

$$\mathbf{u}_{\text{xy}} = \mathbf{F}_{\text{xy}}(\varepsilon)\, \widetilde{\mathbf{x}}_{\text{xy}}, \tag{5}$$

where

$$\mathbf{F}_{\text{xy}}(\varepsilon) = \begin{bmatrix} \dfrac{-k_{\text{i}}\omega_{\text{n}}^2}{\varepsilon^3} & \dfrac{\omega_{\text{n}}^2 + 2\zeta\omega_{\text{n}}k_{\text{i}}}{\varepsilon^2} & \dfrac{2\zeta\omega_{\text{n}} + k_{\text{i}}}{\varepsilon} \\ 1 & -\dfrac{\omega_{\text{n}}^2 + 2\zeta\omega_{\text{n}}k_{\text{i}}}{\varepsilon^2} & -\dfrac{2\zeta\omega_{\text{n}} + k_{\text{i}}}{\varepsilon} \end{bmatrix}, \tag{6}$$

where ε is a design parameter to adjust the settling time, ω_n, ζ, k_i are the parameters that determine the desired pole locations of the infinite zero structure of $\Sigma_{\text{AUG}}^{\text{xy}}$ through:

$$p(s) = (s + k_{\text{i}})(s^2 + 2\zeta\,\omega_{\text{n}} s + \omega_{\text{n}}^2)\,. \tag{7}$$

In principle, when the design parameter ε is small enough, the RPT controller gives arbitrarily fast response. However, in practice, due to the constraints of physical system and inner loop dynamics, we would like to limit the bandwidth of the outer loop to be at least one third of the inner loop system bandwidth. The roll/pitch dynamics H_1 has a bandwidth of 3.82 rad/s. For roll/pitch outer loop controller, we select the parameters in (8) to have a bandwidth of 0.83 rad/s:

$$\omega_{\text{n}} = 0.4, \quad \zeta = 1.2, \quad \varepsilon = 1, \quad k_{\text{i}} = 0.8\,. \tag{8}$$

For the outer loop controller in heave dynamics and yaw dynamics, the inner loop controller already controls the heave velocity w and yaw angular velocity r. We only need to design a controller to control the height z and yaw angle ψ. Similarly, the integral of tracking error is augmented to the original system and forms another augmented system $\Sigma_{\text{AUG}}^{\text{hy}}$:

$$\Sigma_{\text{AUG}}^{\text{hy}}: \begin{cases} \dot{\widetilde{\mathbf{x}}}_{\text{hy}} = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \widetilde{\mathbf{x}}_{\text{hy}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u}_{\text{hy}} \\ \widetilde{\mathbf{y}}_{\text{hy}} = \widetilde{\mathbf{x}}_{\text{hy}} \\ \widetilde{\mathbf{h}}_{\text{hy}} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \widetilde{\mathbf{x}}_{\text{hy}} \end{cases} \tag{9}$$

where $\widetilde{\mathbf{x}}_{\text{hy}} = [\int e \quad r_p \quad r_v \quad p]^T$; r_p, r_v are the position, velocity references; p is the actual height or yaw angle; $e = p - r_p$ is the tracking error of height or yaw angle.

A linear state feedback control law of the form (10) is acquired,

$$\mathbf{u}_{\text{hy}} = \mathbf{F}_{\text{hy}}(\varepsilon)\,\widetilde{\mathbf{x}}_{\text{hy}}\,, \tag{10}$$

where

$$\mathbf{F}_{\text{hy}}(\varepsilon) = \begin{bmatrix} -\dfrac{\omega_n^2}{\varepsilon} & \dfrac{2\zeta\omega_n}{\varepsilon^2} & 1 & -\dfrac{2\zeta\omega_n}{\varepsilon^2} \end{bmatrix}. \tag{11}$$

For height controller:

$$\omega_{\text{n}} = 0.5, \quad \zeta = 1.1, \quad \varepsilon = 1\,. \tag{12}$$

For yaw angle controller:

$$\omega_{\text{n}} = 1, \quad \zeta = 1, \quad \varepsilon = 1\,. \tag{13}$$

4 Navigation State Estimation

4.1 Estimation Framework

The state of UAV includes the position, the orientation and the velocity. The orientation could be estimated by the mechanization of angular measurement from IMU. In the ideal case, the translation velocity could be derived by integrating the acceleration measurement once and the position for another integration. Since the accelerometer output is subject to bias, the double integration of acceleration will result in prohibitively large position drift, rendering it inapplicable for long time navigation of UAV. To facilitate successful autonomous control of UAV, we proposed a state estimation framework as shown in Fig. 6, consisting of two sub-systems: front-end and back-end.

The front-end provides the essential estimate for the onboard autonomous control and the back-end optimizes the UAV trajectory for pose correction and consistent mapping of the environment. However, the GraphSLAM is in essence an off-line optimization technique which can not be ran in parallel with the onboard control loop. In practice, a sliding time window is applied to the trajectory history and only those poses inside the sliding window are optimized. After

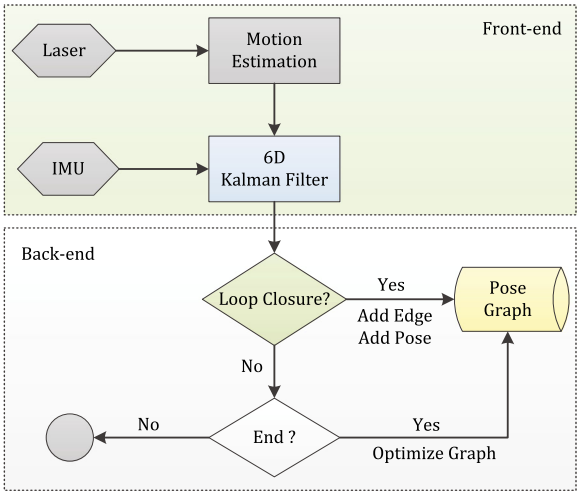


Fig. 6 System schematics illustrating front-end and back-end

the whole trajectory is obtained, a global optimization in the back-end is performed to obtain the optimal trajectory and the global consistent map.

In Fig. 6, we have put the loop closure detection block to the back-end. This makes sense because we utilize the position and velocity estimates from Kalman filter as the input into the the real-time autonomous control. By putting the loop detection into the back-end, the back-end module operates in a self-contained manner which will not influence the real time performance of the UAV control.

4.2 Front-End Estimation

The front-end block provides high frequency state estimate for onboard autonomous control and the initial pose estimate for the back-end block. The block diagram of the front-end state estimation is shown Fig. 7, which consists of three main blocks: feature extraction, scan matching and Kalman Filter.

Feature extraction is the first step toward accurate motion estimation. Considering the operation environment to be forests, we choose the center of trees at flight height to be the features. A laser range finder scans the environment continuously to provide range information in 30 meter range of 270 degree field of view. To extract the validated trees, the laser scan range are processed in three steps: preprocessing, segmentation and extraction. Preprocessing the scan range is necessary since the raw laser data is noisy and corrupted with outliers. Segmentation is performed to generate clusters of range points corresponding to candidate tree stems. The extraction process validates these clusters through a series of geometric descriptors, producing the estimated centers of clusters as the landmarks [7].

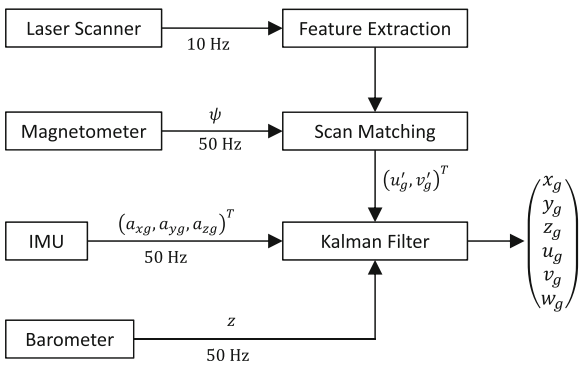


Fig. 7 The front-end state estimation diagram

Scan matching of 2D range scans has been is a mature technique to estimate robot pose in unknown environment. But it suffers from local minima and large rotation errors when using only raw measurements. To overcome these problem, we use feature based scan matching and incorporate the heading angle measurement from IMU in the scan matching process. The extracted features in each scan maintain a certain spatial distance to each other, making it less possible for ambiguous data association. With the heading measurement from IMU, the latter scan is first projected to the previous scan using the heading angle difference. This further reduces the wrong data association induced by possible large rotation movement. Once the correct data association is established, the incremental translation and rotation could be solved in closed-form [17].

A Kalman filter is designed to fuse the sensor information: the translation measurement (u'_g, v'_g) from scan matching, the height measurement z from barometer or mirrored laser beam and the three axis acceleration from IMU.

The process model is described by,

$$\begin{bmatrix} \dot{\mathbf{P}}_\mathbf{n} \\ \dot{\mathbf{v}}_\mathbf{n} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_\mathbf{n} \\ \mathbf{v}_\mathbf{n} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{R}_{\mathbf{n/b}} (\mathbf{a_b} + \mathbf{w_a}) , \quad (14)$$

where $\mathbf{R_{n/b}}$ is the rotation matrix from the body frame to the local north-east-down (NED) frame, $\mathbf{a_b}$ is the acceleration measurement without noise, $\mathbf{w_a}$ is the acceleration measurement noise vector with normal distribution, $\mathbf{a_b} + \mathbf{w_a}$ is the IMU acceleration measurement in body-fixed frame, $\mathbf{P_n}$ is the local NED position vector, $\mathbf{v_n}$ is the local NED velocity vector. $\mathbf{I}$ and $\mathbf{0}$ are identity matrix and zero matrix of proper dimension.

The measurement model is

$$\mathbf{y} = \begin{bmatrix} \mathbf{0}_{3\times 2} & \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 1} \end{bmatrix} \begin{bmatrix} \mathbf{P_n} \\ \mathbf{v_n} \end{bmatrix} + \mathbf{R} , \quad (15)$$

where $\mathbf{R}$ is the measurement covariance matrix with normal distribution.

Implementing the KF requires discretizing the continuous process model (14) and the measurement model (15) using zero-order-hold method. The following procedure is a standard KF process with interleaved time update and measurement update. One

noteworthy point is that the laser motion estimation update rate is 10 Hz while the acceleration measurement refreshes at 50 Hz. When the motion estimation measurement is not available, the state is updated using only the process model (14).

4.3 Back-End Estimation

The position estimate from the Kalman filter is only suitable for short time navigation. This is due to the fact that the planar position is not observable in the measurement equation and thus suffers from long term drift. The back-end estimation use the GraphSLAM technique to bound the position drift to facilitate long range UAV navigation.

GraphSLAM requires the availability of all measurements along the trajectory before they are used to build up a graph and optimize it afterwards. This makes GraphSLAM an off-line method, rendering it inapplicable for real-time navigation. We have reported our results about using GraphSLAM as an off-line optimization technique to generate consistent maps in our previous work [6]. Here we present an on-line GraphSLAM technique using a sliding-window technique.

The main idea of the on-line GraphSLAM is to set a sliding window along the trajectory, limiting the search range only to those poses lying in the time window prior to the current pose. As illustrated in Fig. 8, the first pose and it's measurement is denoted as $\mathbf{x}_0$. It also serves as a reference origin to which all the future poses will be referred. As the UAV collects more data, a series of new poses and measurements are added, including $\{\mathbf{x}_2 \cdots \mathbf{x}_t\}$. The sliding window is initialized

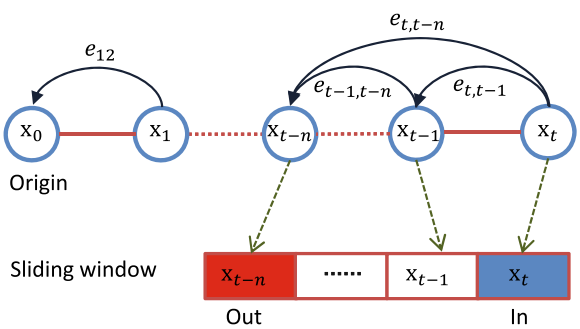


Fig. 8 Sliding window diagram with poses being pushed in and popped out

with a capacity of n as the first pose $\mathbf{x}_0$ is pushed into the window.

The online GraphSLAM based on the sliding window significantly decreases the drift of the position estimate compared with that of the Kalman filter. However, the introduction of sliding window is indeed a sacrifice of the optimization performance by limiting the searching range only in the 5 seconds sliding window. For long time navigation, the UAV position is still prone to drift without global optimization. Therefore, a two-layer back-end framework is presented as shown in Fig. 9. Poses and features from the front-end are pushed into the sliding window at each time step. After the local optimization, the optimized pose estimate is transfered back to the front-end for real-time control. At the same time, the locally optimized pose is pushed into a larger container to store all the poses and measurements, forming a global graph to be optimized after the mission. This two-layer graph setup makes sure the UAV achieves slow drift in the pose estimation during flight and eventually obtains a globally consistent trajectory and map afterwards. This configuration is justified by the fact that the UAV does not need perfect pose estimate during flight and the slow drift caused by the local sliding window optimization is acceptable for UAV operations lasting up to 20 minutes.

Due to hardware constraints, the front-end and back-end algorithms run in two onboard computers. The front-end algorithms, including the scan matching and the Kalman filter, run on the Gumstix Overo Fire. While the back-end algorithms, including the sliding window online GrpahSLAM and the global pose graph construction, run on the Mastermind. The two computers are connected through a serial port. Figure 9 shows the message interaction between the two computers. Initially, the state $\mathbf{P}_0$ is directly fed to the autonomous control. To optimize the initial state $\mathbf{P}_0$, it is sent to Mastermind with its measurement. The time delay caused by the local optimization, the global pose construction and the serial communication make it impossible to use $\mathbf{P}_0$ directly for flight control. In particular, experiments show that the delay Δt between the initial $\mathbf{P}_0$ and the optimized pose $\mathbf{P}_\mathbf{0}^\mathbf{n}$ is 300 ms. Recalling that the main loop in the front-end is 50 Hz, the delay of 15 loops is not negligible for real-time operation. To deal with the delay, we propose to design a state update scheme to take into account the

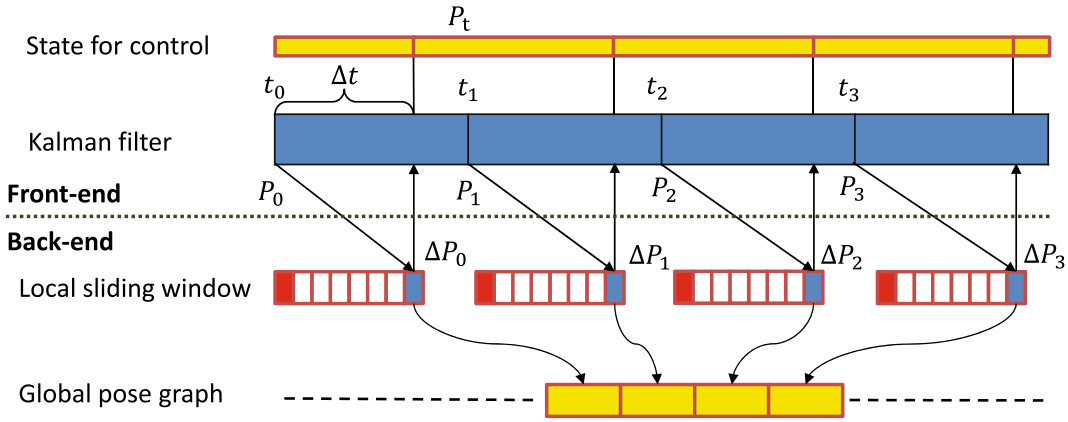


Fig. 9 A timing graph showing the interaction between the front-end, the sliding window and the global back-end

delay which works as follows: At time t_0, we have an initial pose $\mathbf{P}_0$ from the Kalman filter. After time Δt we received pose correction $\Delta\mathbf{P}_0$ of $\mathbf{P}_0$, then we have a new pose for time t_0,

$$\mathbf{P}_\mathbf{0}^\mathbf{n} = \mathbf{P}_0 \Delta \mathbf{P}_0. \tag{16}$$

At any time $t > t_0 + \Delta t$, the initial pose is,

$$\mathbf{P}_t = \mathbf{P}_0 \Delta \mathbf{P}_\mathbf{0}^\mathbf{t}, \tag{17}$$

where $\Delta\mathbf{P}_\mathbf{0}^\mathbf{t}$ is the initial pose difference between $\mathbf{P}_t$ and $\mathbf{P}_0$. The update pose $\mathbf{P}_\mathbf{t}^\mathbf{n}$ of time t is,

$$\mathbf{P}_\mathbf{t}^\mathbf{n} = \mathbf{P}_\mathbf{0}^\mathbf{n} \Delta \mathbf{P}_\mathbf{0}^\mathbf{t} = \mathbf{P}_0 \Delta \mathbf{P}_0 \Delta \mathbf{P}_\mathbf{0}^\mathbf{t}. \tag{18}$$

At time t_1 a new initial pose is sent to the back-end for optimization and after time t the update pose $\Delta\mathbf{P}_2$ is returned. The update state is again updated using Eq. 18, except that the time index is changed from t_0 to t_1.

5 Path Planning in Unknown Clustered Environment

For the vehicle to navigate safely in the foliage environment, it is necessary to have real-time path planning and obstacle avoidance capability. Due to the low computational power of the on-board computer and the general unknown environment, the path planning and obstacle algorithm needs to be fast enough to run in a mobile computer. Conventionally, a path planning algorithm is treated as an optimization problem. But it is generally difficult to solve due to the high dimension of optimization vector, which includes the state of the vehicle and the non-linear constraints introduced by the obstacles. Even using state-of-art numerical algorithms like the CPLEX, the convergence of the solution is not guaranteed and the real-time capability is hard to achieve. Therefore, in practical engineering applications, a suboptimal solution is usually considered by separating the optimization problem into two or more steps.

In our solution, we consider the planning in the configuration space and the state space of the vehicle separately. This separation reduces the dimension of the optimization problem and makes it possible to achieve a real-time path planning algorithm in obstacle-strewn environment. We propose a path planning system with global path planner using A* searching [15] and a local smoother using efficient boundary value problem solver [13]. The A* searching is performed in a discrete grid map built from the laser scan measurement. The vehicle in the grid map is considered as a point-mass particle which could move from one grid to all its connected grids. This does not violate the actual vehicle model since the quadrotor could be simply considered as a double or triple integrator when it operates at low speed [18].

The A* searching provides candidate target points for the vehicle to reach. Due to the dynamic of the UAV, we would like to gain a trajectory that is able to be limited on jerk, acceleration and velocity. We use an bang-zero-bang based time optimal solution to generate references meeting the vehicle's dynamic constraints [13]. Collision checking is also performed to make sure that the smoothed trajectory does not collide with any obstacles. This two-step path planning

algorithm can provide dynamically feasible trajectories to lead the vehicle from any initial position to any reachable final position. The path planning structure is given in Algorithm 1.

Algorithm 1: Online path planning framework

Input: Current pose $\mathbf{x}$, obstacle position $\{m_i\}$ in local body frame, $i = 1, \cdots, n$.
Output: Trajectory reference $\mathbf{x}_{\text{ref}}$

1 Search in the configuration space using A*;
2 Connect the grids using split and merge, generating a series of line segments;
3 $R_{\text{g}} \leftarrow$ take the first turning point of the line segment as the line segments;
4 $\{\mathbf{r}_j\} \leftarrow$ sample multiple local targets around the current vehicle state $\mathbf{x}$ and order them in descending order based on their distance to the global target R_{g};
5 **for** *all local targets in* $\{\mathbf{r}_j\}$ **do**
6 $\quad \mathbf{x}_{\text{ref}} \leftarrow$ Solve the boundary value problem between $\mathbf{x}$ and $\mathbf{r}_j$;
7 $\quad$ collision $\leftarrow$ check if collision happens between $\{m_i\}$ and $\mathbf{x}_{\text{ref}}$;
8 $\quad$ **if** *collision* **then**
9 $\quad\quad$ Delete the current local target and choose the second best local target ;
10 $\quad$ **else**
11 $\quad\quad$ Break ;
12 **return** $\mathbf{x}_{\text{ref}}$;

The algorithm consists of several main blocks: the global configuration space search using A* (step 1) and the boundary value problem (step 6) [8]. The boundary value problem seeks to generate a reference trajectory given two sets of conditions on the boundaries. Reflexxes Motion Libraries [13] provides a general solution to this problem. The global configuration space search is to give a rough plan that ignores the complex dynamics of the vehicle but considers as much topological information as possible. Since there is no prior map of the environment, a local map based on the current laser range finder measurement is built up in polar coordinate. The A* path planning algorithm is actually a graph search algorithm. To run the A* searching algorithm, a polar coordinate map is first built from the input of a 30 m laser scanner. For each obstacle point returned from the 30 m laser scanner's measurement, a Gaussian-based cost filed is added around it. A preprocessed polar coordinate map is shown in Fig. 10.

During the A* searching, the algorithm would generate a path with the lowest cost from the current position of the vehicle to the target point. A typical path is shown in Fig. 10 as the green line. The resulted path normally consists of a series of waypoints located in each grid in the polar coordinate. In order to find the best direction the vehicle should aim for, a split and merge algorithm is used to transfer these waypoints

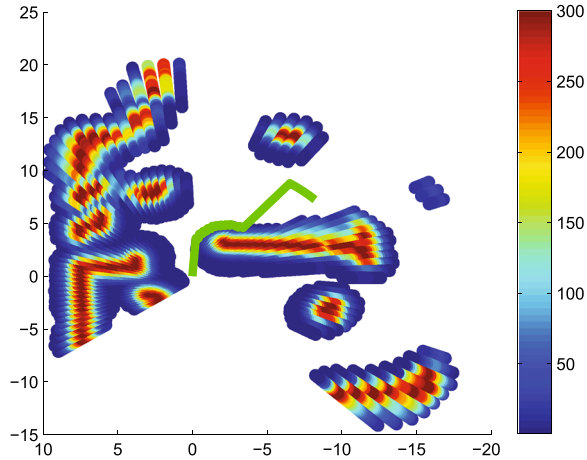


Fig. 10 A Gaussian cost map in polar coordinate. The color ranging from *red* to *blue* indicates the closeness of the grid to the detected obstacle

into a series of line segments. The split and merge process finds the first turning point R_{g}, which is used to determine the best target point to go. The direction and the distance of the first turning point are then passed to the local path planner to search for a collision free path from the current vehicle position to the turning point.

For the local planner, a similar idea close to the vector field histogram (VFH) method is adopted. In VFH, the vehicle always turns to the direction that is both obstacle free and also towards the target R_{g} from the global planner. The behavior is realized by forming an optimal function that consists of different objectives, such as distance to the global target and the angle difference compared to the last direction. We sample multiple local target points around the vehicle and calculate the resulted trajectory based on the current vehicle states and the local target points. The trajectory that is both collision free and closest to R_{g} is chosen. Each trajectory starts at the current vehicle states and ends at the local target points with zero velocity and zero acceleration. Therefore, the vehicle is always at a safe state so that it could stop safely when following the current trajectory.

6 Experimental Results

All the function blocks, including the RPT control law, the onboard motion estimation and the path planning,

are implemented in the UAV onboard processors to form a comprehensive navigation system. The developed navigation system enables our customized UAV to perform autonomous flight through a small area of forest. The front-end state estimation is verified by the flight test, and the back-end state estimation is validated using the onboard data in an off-line processing manner.

6.1 Autonomous Flight in Forest

The test field is set in a small forest with sparse trees as shown in Fig. 11. The distances among trees are sparse enough to provide an obstacle-free trajectory for the UAV to fly. At the flight height, some trees have thick branches instead of a single tree trunk. This poses challenges for the onboard feature extraction and data association. The obstacle-free trajectory is predefined by assuming knowledge of the tree positions in the environment. This simulates the case where the onboard obstacle avoidance is properly functional. It should be noted that the map information is not used during the process of motion estimation and autonomous control.

Figure 11 shows the test platform flying in the test field. The UAV platform is being autonomously controlled while following a predefined trajectory. The predefined trajectory is loaded into the system during the system power up. The front-end state estimation provides the position and velocity estimates which are fed back to the RPT control law to control the UAV.

Figure 12 shows the position tracking performance together with the tracking error. In the top two sub-figures, the red dashed lines are the position references and the blue solid lines are the state estimates from the UAV. As can be seen from the top two sub-figures, the RPT controller controls the UAV to track the position reference in both x and y direction. The tracking error in x direction is 0.2 meter and 0.5 meter in y direction. The maximum tracking errors occur at the beginning of the trajectory when there is step acceleration reference.

The flight test demonstrates the accuracy of the UAV dynamics model and the efficiency of onboard motion estimation in a computation-limited processing unit. It also shows that the front-end state estimate can provide reliable state estimate for the RPT controller.

6.2 Back-End Optimization

The back-end state estimation seeks to optimize the initial trajectory to produce a consistent map of the environment. To better verify the effect of trajectory optimization, we test the GraphSLAM algorithm using data collected in two environments: the indoor forest environment (Fig. 13) and the outdoor real forest (Fig. 11).

The indoor forest scenario consists synthetic trees with perfect cylindrical shapes. Besides the poles, square concrete pillars and the interior of the wall will also be measured by the laser range finder. We

Fig. 11 The outdoor testing scenario with the flying quadrotor

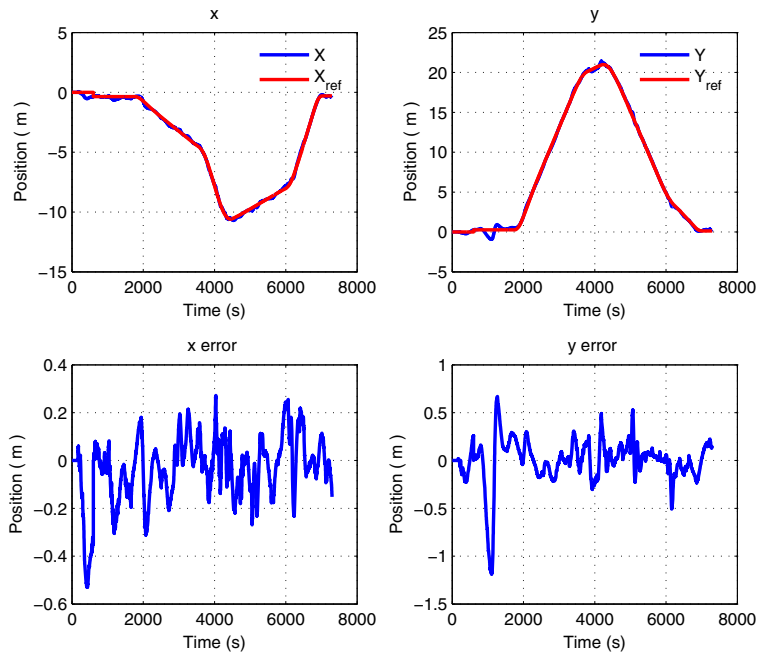


Fig. 12 Position and velocity tracking in X-Y direction

manipulate the distance among the trees to make sure there are enough number of features in each scan. An autonomous flight is performed to collect all the onboard data including the trajectory history and the laser range data.

Figure 14 compares the map projected on the initial trajectory and the optimized trajectory respectively. Since there is no ground truth available in the indoor forest dataset, we can not quantitatively analyze the effect of trajectory optimization. We consider the consistency of the projected map as the criteria to evaluate the back-end state estimation. The initial map and trajectory are the results of motion estimation based on scan matching, marked by green dots plot. It can be seen that when all the measurements are projected on the initial trajectory, the overall map is not consistent.

Fig. 13 Indoor test scenario for SLAM verification

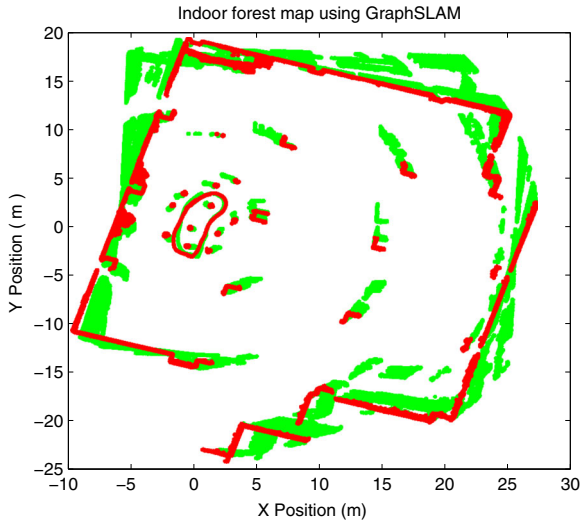


Fig. 14 Map comparison between the initial trajectory and the optimized trajectory

The interior walls and position of the square pillars drift away. The red-dot plot is the optimized map and trajectory. By visual checking of the map we could see the optimized map is more consistent than the initial map. Figure 15 shows close-view of one part of the map, indicating that the red pillar retains its rectangular shape while the green pillars deform to an incorrect way. Figure 15 also clearly manifests the perfect circular contour of the landmarks while the initial green contours scatter around.

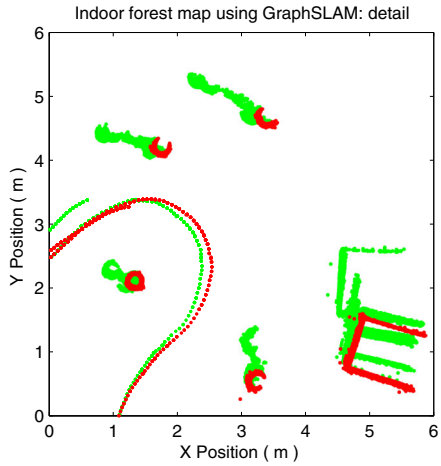


Fig. 15 Optimized map and trajectory detail

After the evaluation of the GraphSLAM algorithm in indoor environment, we apply the algorithm to the dataset from in a real small forest. Figure 16 shows the comparison between the initial map and the optimized map for the forest. The green plot is the initial map from the Kalman filter while the red one is the optimized trajectory and map. It can be seen that there are two neighboring clusters of green plots while only one cluster of red points. Due to the complexity of the environment, there is no hollow tree contours extracted. There are still large clusters of objects which do not correspond to trees in the environment.

The effect of trajectory optimization is less optimal than the indoor forest dataset. This is expected since the real environment exhibits several challenges: first, the uneven terrain produces undesired ground strikes of the laser range finder, making the feature extraction a challenging task. Second, the real trees in the environment are relatively smaller than the ones in indoor forest and have thick branches at the flight height. Wrong data association may happen at a certain time step. Third, the distance among trees are very large, leaving each segment in the clustered range scan with limited number of points. Estimating the tree centers with less points produces large position error. Last,

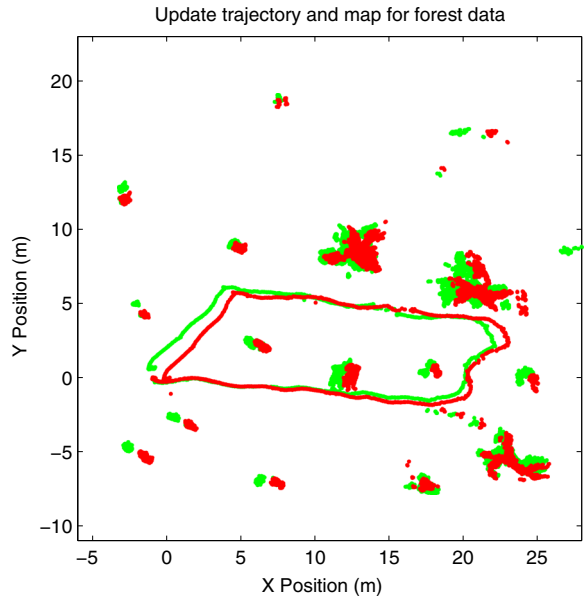


Fig. 16 Optimized map and trajectory in the small forest test

the cross section of tree trunk at the scanned plane may not follow a circular shape. The estimated position of tree centers may be different when the trees are scanned at different view angle. All these factors make the optimized map less consistent than the one in indoor forest. But with proper feature extraction and data association, the back-end optimization demonstrates its positive effect in correcting the trajectory and generating more consistent map.

6.3 Autonomous Flight with Online GraphSLAM and Online Path Planning

We have validated the performance of the UAV navigation system using online consistent state estimation and robust perfect tracking. The previous two tests have used predefined trajectory references during flights. In practice, the flight area is unknown prior to the take-off, requiring the UAV to be able to avoid any obstacles in its vicinity during the flight period. Online path planning is demanded to provide real-time trajectory references which avoid the obstacles and maintain the original trajectory route as much as possible.

We designed a flight test to verify the UAV navigation system including the online GraphSLAM and online path planning. The UAV is required to travel to five waypoints (w1-w5) which fall on a rectangle shape, which are labeled as black circle in Fig. 17. Once reaching a waypoint, a hover of 10 seconds is performed. At each corner of the rectangle, the UAV's heading is shifted 90 degrees clock wise after the hovering. Without obstacles, the designed trajectory is a rectangle shape. However, as shown in Fig. 17, there are two obstacles (T1, T2) lying on the connected line of the waypoints. To reach the waypoints, the UAV must find other feasible path instead of the direct connection between the waypoints.

The whole mission was fully autonomous, including take-off, waypoint navigation, obstacle avoidance,

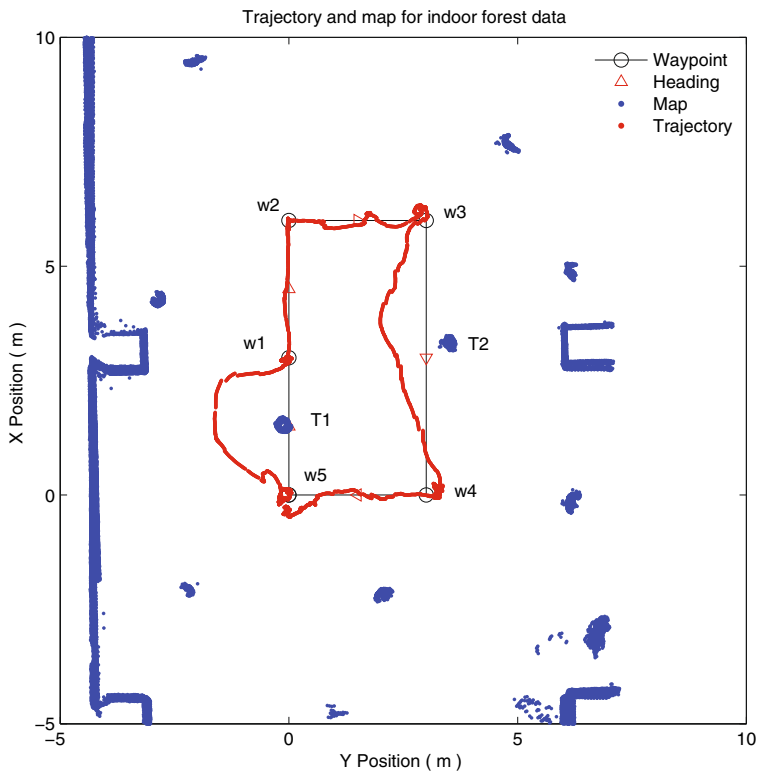
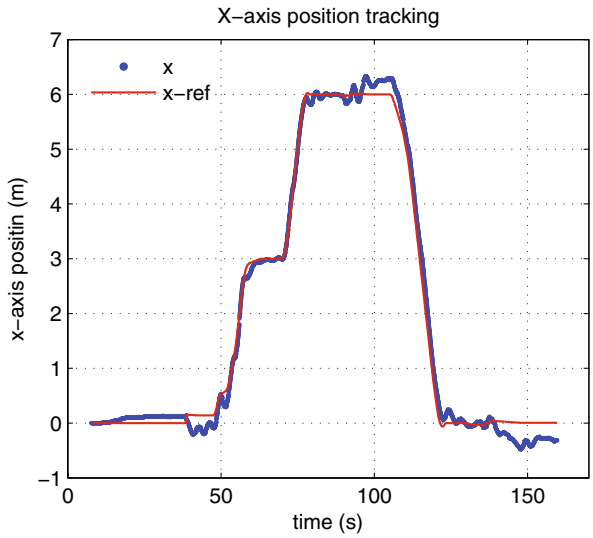
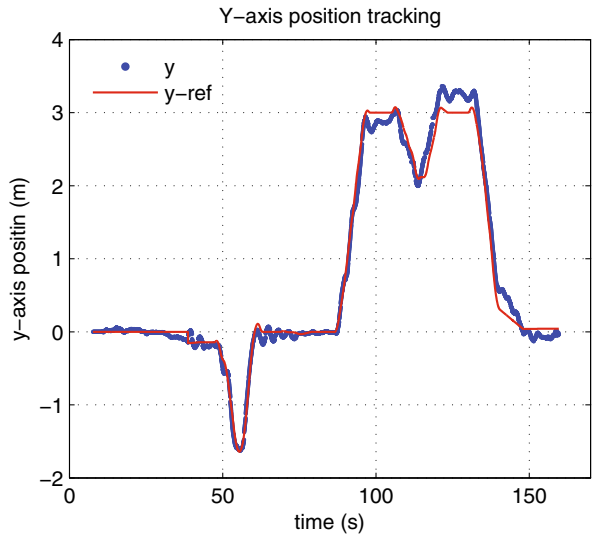


Fig. 17 Consistent map with obstacle avoidance trajectory. The UAV is required to follow a rectangle shape trajectory with five waypoints (*black circle* w1-w5) along the way. At each corner of the rectangle, the UAV's heading is shifted 90 degrees clock wise. The *red dot* trajectory is the real flight path which avoided obstacles (T1, T2) on the rectangle

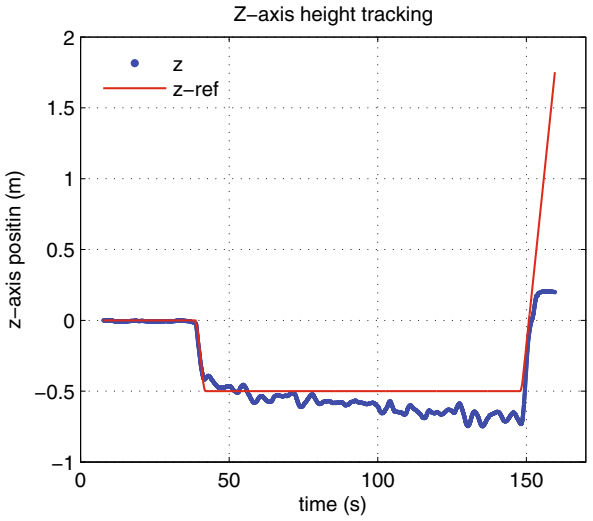
online GraphSLAM and landing. The flight data were recoded onboard and plotted in Fig. 17. The red dot trajectory is the real flight path which avoided obstacles nearby the rectangle. The blue dots plot is the accumulative plot of the environment, including the trees, rectangle pillars and walls. The blue good shape of walls and pillars in the map demonstrates the consistency of the trajectory estimate of the entire mission. Figure 18a–d show the good tracking performance in x, y, z and heading ψ directions, validating all the algorithm developed in this paper: the consistent state estimation algorithm, the obstacle avoidance and the robust and perfect control technique.



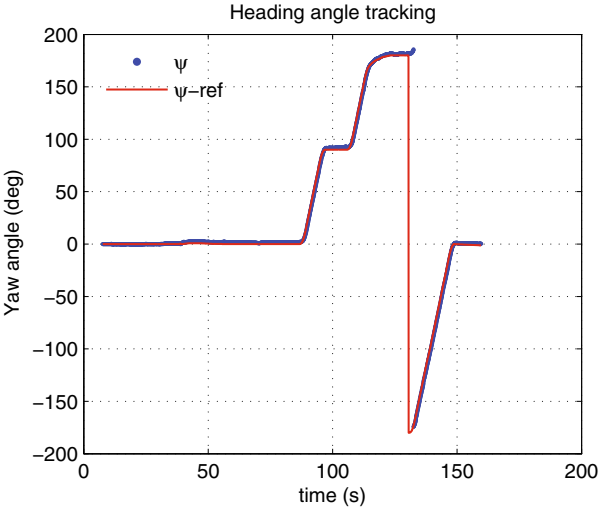
(a) x position tracking performance



(b) y position tracking performance



(c) z height tracking performance



(d) Yaw angle tracking performance

Fig. 18 Onboard trajectory tracking performance with obstacle avoidance. The whole mission is fully autonomous, including take-off, waypoint navigation with online GraphSLAM and obstacle avoidance, and landing. Figure 18a–d show the good tracking performance in x, y, z, and yaw directions, validating all the algorithms developed in this study: the consistent state estimation algorithm, the obstacle avoidance and the robust and perfect control

7 Conclusion

We have presented the navigation system for UAV in foliage environment and verified the system in real flight tests. Various essential parts of the navigation system have been presented, including model and control, state estimation and path planning. The model of a quadrotor UAV is first structured as inner loop and outer loop models. The inner loop model is stabilized with a commercial autopilot. To track any external reference with disturbance rejection capability, we have designed a robust perfect tracking outer loop control law. The outer loop control law demonstrates perfect tracking of given position reference in the autonomous flight of UAV in forest.

The state estimation framework of UAVs in foliage environments is the focus of this manuscript. The framework consists of a front-end and a back-end. The front-end is a real-time motion estimation by matching scans from the onboard laser range finder. The velocity measurement from the scan matching is fused with the acceleration of the IMU in a Kalman filter to provide the high update rate state estimation for the outer loop control. The back-end optimization is a customized GraphSLAM algorithm. To make optimization constant in time, a sliding window technique is introduced to optimize those poses falling in the time window. All initial poses from the Kalman filter in the time window form a local pose graph. By optimizing the local graph, the initial pose from the Kalman filter is updated and fed back to the controller to correct the position drift of the front-end. If necessary, the locally optimized poses could also form a global pose graph which is optimized at the end of the mission to produce a globally consistent trajectory and map.

We also developed the real-time path planning using the measurement of the horizontal laser range finder. An A* searching is performed in the local grid map to produce a candidate target point. Considering the dynamics constraints of the UAV, the trajectory generation is formulated as a boundary value problem and solved in analytic form, achieving real-time performance in obstacle-strewn environment, especially forests.

To verify all the developed algorithms, we performed autonomous flight tests in indoor simulating forests and outdoor practical forests. Given a mission plan with five waypoints, the UAV can navigate precisely using the proposed motion estimation technique and avoid obstacles along the trajectory. Replaying the recorded laser scans on the estimated poses generates consistent map of the environment, demonstrating the accuracy of the navigation system.

We are now working on the improve the navigation system in the following aspects. First, the laser-based motion estimation will be fused with other measurement, such as visual odometry from optical flow sensors. This is due to the fact that laser range finders have limited measurement range and may return little measurement points in sparse environments. Visual odometry can serve as a complimentary odometry measurement in such scenarios. Second, we are working to increase the flight speed of the UAV form 0.5 m/s to 5 m/s. This is essential to increase the operation range of the UAV in the constraint of limited flight duration. Fast flight of UAVs in obstacle-strewn environment poses great challenges for motion estimation and path planning. Last but not least, we are trying to develop real-time 3D motion estimation and path planning. The current navigation system relies on a 2D laser range finder, which is based on an assumption that the environment does not change drastically in the vertical direction. We are investigating the possibility of using stereo cameras or RGB-D cameras for 3D navigation of UAVs in complex environments.

References

1. Cifer. http://uarc.ucsc.edu/flight-control/cifer/
2. Alvarenga, J., Vitzilaios, N.I., Valavanis, K.P., Rutherford, M.J.: Survey of unmanned helicopter model-based navigation and control techniques. J. Intell. Robot. Syst., 1–52 (2014)
3. Cai, G., Dias, J., Seneviratne, L.: A survey of small-scale unmanned aerial vehicles: Recent advances and future development trends. Unmanned Syst. **02**(02), 175–199 (2014)
4. Chen, B.M.: Robust and H∞ Control. Communications and Control Engineering Series. Springer (2000)
5. Chisholm, R.A., Cui, J.Q., Lum, S.K.Y., Chen, B.M.: UAV LiDAR for below-canopy forest surveys. J. Unmanned Veh. Syst. **01**(01), 67–68 (2013)
6. Cui, J.Q., Lai, S., Dong, X., Liu, P., Chen, B.M., Lee, T.H.: Autonomous navigation of UAV in forest. In: 2014 International Conference on Unmanned Aircraft Systems, pp. 726–733 (2014)
7. Cui, J.Q., Wang, F., Dong, X., Ang Zong Yao, K., Chen, B.M., Lee, T.H.: Landmark extraction and state estimation for UAV operation in forest. In: 32nd Chinese Control Conference, pp. 5210–5215. Xi'an (2013)

8. Dong, W., Gu, G.Y., Zhu, X., Ding, H.: Solving the boundary value problem of an under-actuated quadrotor with subspace stabilization approach. J. Intell. Robot. Syst., 1–13 (2014)
9. Georgiev, A., Allen, P.: Localization methods for a mobile robot in urban environments. IEEE Trans. Robot. **20**(5), 851–864 (2004)
10. Guivant, J., Masson, F., Nebot, E.: Simultaneous localization and map building using natural features and absolute information. Robot. Auton. Syst. **40**(2–3), 79–90 (2002)
11. Guivant, J., Nebot, E., Baiker, S.: Localization and map building using laser range sensors in outdoor applications. J. Robot. Syst. **17**(10), 565–583 (2000)
12. Kang, K., Prasad, J.V.R.: Development and flight test evaluations of an autonomous obstacle avoidance system for a rotary-wing UAV. Unmanned Syst. **01**(01), 3–19 (2013)
13. Kröger, T.: On-Line Trajectory Generation in Robotic Systems. Springer Tracts in Advanced Robotics, vol. 58. Springer, Berlin (2010)
14. Langelaan, J.W.: State estimation for autonomous flight in cluttered environments. Ph.D. thesis, Stanford University (2006)
15. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
16. Liu, K., Chen, B.M., Lin, Z.: On the problem of robust and perfect tracking for linear systems with external disturbances. Int. J. Control **74**(2), 158–174 (2001)
17. Lu, F., Milios, E.: Robot pose estimation in unknown environments by matching 2D range scans. J. Int. Robot. Syst. **18**, 249–275 (1997)
18. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: IEEE International Conference on Robotics and Automation, pp. 2520–2525 (2011)
19. Paull, L., Saeedi, S., Seto, M., Li, H.: AUV navigation and localization: A review. IEEE J. Ocean. Eng. **39**(1), 131–149 (2014)
20. Phang, S.K., Li, K., Yu, K.H., Chen, B.M., Lee, T.H.: Systematic design and implementation of a micro unmanned quadrotor system. Unmanned Syst. **02**(02), 121–141 (2014)
21. Ross, S., Melik-Barkhudarov, N., Shankar, K., Wendel, A., Dey, D., Bagnell, J., Hebert, M.: Learning monocular reactive UAV control in cluttered natural environments. In: IEEE International Conference on Robotics and Automation, pp. 1765–1772 (2013)
22. Shen, S., Michael, N., Kumar, V.: Obtaining liftoff indoors: Autonomous navigation in confined indoor environments. Robot. Autom. Mag. IEEE **20**(4), 40–48 (2013)
23. Thrun, S., Montemerlo, M.: The GraphSLAM algorithm with applications to large-scale mapping of urban structures. Int. J. Robot. Res. **25**(5–6), 403–429 (2006)
24. Wang, B., Chen, B.M., Lee, T.H.: An RPT approach to time-critical path following of an unmanned helicopter. In: 8th Asian Control Conference, pp. 211–216 (2011)
25. Wang, F., Cui, J.Q., Phang, S.K., Chen, B.M., Lee, T.H.: A mono-camera and scanning laser range finder based UAV indoor navigation system. In: International Conference on Unmanned Aircraft Systems, pp. 694–701 (2013)

Jin Q. Cui received the B.S. and M.S degree in Mechatronic Engineering from Northwestern Polytechnical University, Xi'an, China, in 2005 and 2008 respectively. He has finished his Ph.D. in Electrical and Computer Engineering in National University of Singapore (NUS) in 2015. In the Ph.D. study, his research direction is navigation of unmanned aerial vehicles (UAV) in GPS-denied environments, especially forest. He is now a research scientist in the Control Science Group at Temasek Laboratories in NUS. His research interests are GPS-less navigation using LIDAR and vision sensing technologies.

Shupeng Lai received the B.S. degree from Nanyang Technological University, Singapore, in 2012. He now pursuing his Ph.D. in National University of Singapore, Singapore. His research interests include path planning for small-scale UAVs and multi-agent systems.

Xiangxu Dong received the B.S. degree from Xiamen University, Xiamen, China, in 2006. In 2012, he received his Ph.D. in National University of Singapore, Singapore. He is now a Research Scientist with the Temasek Laboratories, National University of Singapore. His research interests include real-time software systems, formation flight control, and unmanned aerial vehicles.

Ben M. Chen (S'89–M'92–SM'00–F'07) received the B.S. degree in mathematics from Xiamen University, Xiamen, China, in 1983, the M.S. degree in electrical engineering from Gonzaga University, Spokane, WA, USA, in 1988, and the Ph.D. degree in electrical and computer engineering from Washington State University, Pullman, WA, 1991. He is currently a Professor and the Director of the Control, Intelligent Systems and Robotics Area of the Department of Electrical and Computer Engineering with the National University of Singapore, Singapore, where he is also the Head of the Control Science Group, Temasek Laboratories. He has authored/coauthored ten research monographs, including H2 Optimal Control (Prentice-Hall, 1995), Robust and H∞ Control (Springer, 2000), Hard Disk Drive Servo Systems (Springer, 1st Edition, 2002; 2nd Edition, 2006), Linear Systems Theory (Birkhäuser, 2004), Unmanned Rotorcraft Systems (Springer, 2011), and Stock Market Modeling and Forecasting (Springer, 2013). His current research interests are in systems and control, unmanned aerial systems, and financial market modeling. Dr. Chen currently serves as Editor-in-Chief of Unmanned Systems and Deputy Editor-in-Chief of Control Theory and Technology. He had served on the editorial boards of a number of journals, including the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, Systems and Control Letters, and Automatica.