Artificial Life and Robotics (2018) 23:428–433
https://doi.org/10.1007/s10015-018-0432-3

ORIGINAL ARTICLE

Mission planning of iOS application for a quadrotor UAV

Zeming Lu[1] · **Fusaomi Nagata**[1] · **Keigo Watanabe**[2]

Received: 4 February 2018 / Accepted: 26 March 2018 / Published online: 5 April 2018

Abstract
Development of civilian UAV (unmanned aerial vehicle) applications has become possible with the progress of electronics and information technologies. In addition, smartphones have rapidly gained popularity and become very important due to the simple operability and mobility. Under such a background, there is a need to have an easy and flexible way to control a UAV using such a smartphone. The authors already developed basic handlers to enable an operator to remotely control a quadrotor and monitor its surroundings using an iOS device. The basic handlers were implemented for obtaining compass information, controlling a gimbal, autopilot function for return. In this paper, following and circling around functions while gazing a moving object are first developed. Then, another promising function called the mission planning is additionally designed and implemented to allow the quadrotor to execute a self-flight task using global positioning system (GPS) information. As a result, the iOS application enables the quadrotor to achieve complex tasks. The functionality of the developed software is evaluated through experiments using a quadrotor and an iOS device.

Keywords Unmanned aerial vehicle · Quadrotor · iOS · iPhone · Remote control · Following mode · Circling around mode · GPS

1 Introduction

This paper is centered on the proposal of an original quadrotor UAV controller design based on widely spread personal smartphones. In recent years, if a UAV is operated through wireless communication, a PC is generally used for controlling it [1–3]. Smartphones have rapidly gained popularity and become very important due to the simple operability and mobility, and hence the flexibility and ease of use to control a UAV will be provided with portable technologies such as iPhone. It is also expected that the smartphone will enable us to get real-time information from various kinds of sensors built in a quadrotor. However, at the present stage, it seems that actual applications using smartphones are not many.

In our research, to cope with the need, a quadrotor UAV equipped with multiple monochrome binocular cameras, ultrasonic distance sensors and a RGB camera was first considered as a controlled object. Then, a basic system was developed to monitor the surrounding environment while remotely controlling the quadrotor using an iOS device. iOS formerly means iPhone OS which is a mobile operating system created and developed by Apple Inc. Basic handlers for obtaining compass information, controlling a gimbal, autopilot function for return were already implemented [4–6].

In this paper, following and circling around functions while gazing a moving object are first developed [7, 8]. Then, another promising function called the mission planning is additionally designed and implemented to allow the quadrotor to execute a self-flight task using GPS information. As a result, the iOS application enables the quadrotor to achieve complex tasks. The functionalities of the programs are evaluated through the software development and actual experiments using a quadrotor and an iOS device.

This work was presented in part at the 23rd International Symposium on Artificial Life and Robotics, Beppu, Oita, January 18–20, 2018.

✉ Fusaomi Nagata
nagata@rs.tusy.ac.jp

[1] Graduate School of Science and Technology, Tokyo University of Science, Yamaguchi, 1-1-1 Daigaku-Dori, Sanyo-Onoda 756-0884, Japan

[2] Okayama University, Okayama, Japan

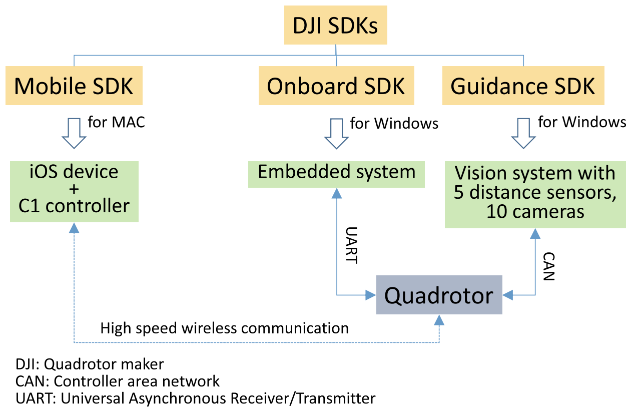

DJI SDKs
Mobile SDK
Onboard SDK
Guidance SDK
for MAC
for Windows
for Windows
iOS device + C1 controller
Embedded system
Vision system with 5 distance sensors, 10 cameras
UART
CAN
Quadrotor
High speed wireless communication
DJI: Quadrotor maker
CAN: Controller area network
UART: Universal Asynchronous Receiver/Transmitter

Fig. 1 Hardware block diagram and SDKs of the quadrotor system

2 Experimental system

2.1 Hardware of quadrotor

Figure 1 shows the overview of hardware structure of the quadrotor system. The main body is a quadrotor platform (Hardware name: Matrice 100) provided by DJI Co., Ltd., in which a CPU controls four DC motors through electronic speed control (ESC) port to drive four rotors. A GPS, compass, and micro-inertial measurement unit (MIMU) are also equipped. The MIMU can measure angles, their velocities, and accelerations concerning the attitude of the quadrotor.

Besides these, an onboard embedded system, an RGB camera fixed to a gimbal, and a vision system called guidance are mounted on the main body. The embedded system allows to handle flight control, vehicle telemetry, camera, and gimbal control. The flight control functions include, e.g., real-time attitude control, velocity control, and position control. The embedded system makes state information available in real time, e.g., inertia, attitude, heading, velocity, position, battery-remaining capacity sensors and a barometric pressure gauge. The height of the quadrotor is estimated from the value of the pressure gauge. In addition, the vision system is composed of five ultrasonic distance sensors and ten monochrome cameras. Figure 2 shows the quadrotor's overview used in this study. Table 1 tabulates the main specifications of the quadrotor.

2.2 Software development environment

Figure 3 presents the software development environment that constitutes three software development kits (SDKs). The mobile SDK for Xcode on macOS allows us to build a customized mobile application for iOS device [9]. In

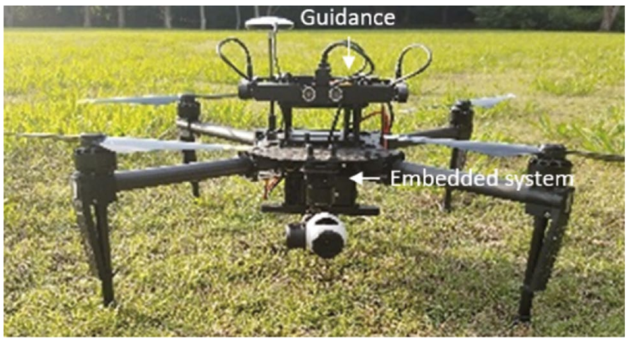


Fig. 2 Overview of the quadrotor used in experiment

Table 1 The main specifications of the quadrotor

Items	Specifications
Diagonal wheelbase	650 mm
Weight	2355 g
Max. takeoff weight	3600 g
Max. speed of ascent	5 m/s
Max. speed of descent	4 m/s
Max. wind resistance	10 m/s
Max. speed	22 m/s
Hovering time	22 min
Operating temperature	− 10 to 40 °C
Operating frequency	922.7–927.7 MHz (Japan)
Transmission distance	3.5 km

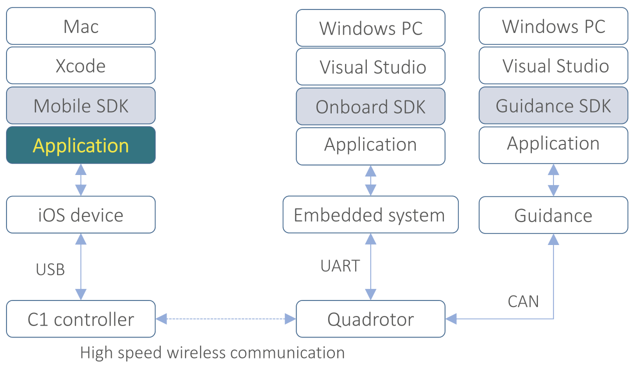


Fig. 3 Three SDKs provided by DJI and their software development environment

addition, the onboard embedded system (Windows OS) mounted on the quadrotor can monitor and control the flight behavior of the body using API functions included in DJI onboard SDK for Windows while utilizing the built-in intelligent navigation (IN) mode to create autonomous flight paths and maneuvers. The IN mode is built in the board on the quadrotor. When an iOS device is not available, the operator can manually control the quadrotor using a handy-sized controller called C1. The onboard embedded system communicates with the DJI flight controller built in the quadrotor via a direct serial connection (UART). Furthermore, the DJI guidance SDK enables to customize the application and extend to the functions using vision and distance sensors according to the needs of developers.

Xcode is an integrated development environment (IDE) containing a suite of software development tools for macOS, iOS, WatchOS, and tvOS provided by Apple. Xcode uses Model View Controller called MVC for development. The MVC is a software architectural pattern for implementing a user interface on Mac. It divides a given software application into three interconnected parts, i.e., model, view, and controller, so that users can view the status of the quadrotor and give suitable commands to it.

One of the relative merits is that this application can combine a quadrotor with a smartphone's Internet access capability so that the quadrotor can obtain more information and data processing capability such as Apple map. In addition, the software development environment for iOS applications is technically open, so that it is easier for engineers to develop, improve and extend the functions in each application using the three SDKs shown in Figs. 1 and 3.

3 Developed software and experiment

3.1 Circling and following functions

It is difficult for an operator to manipulate the quadrotor along a circular flight path, and it is more difficult to make the quadrotor to keep gazing on a moving target. The controller designed for the quadrotor has two function modes. They are the circling mode and following mode as shown in Figs. 4 and 5, respectively. The following and circling modes are switchable. The following mode is set by default. In addition to the altitude, the radius in the circling mode and the distance in the following mode can be set in the operating interface of the application. iOS devices such as iPhone,

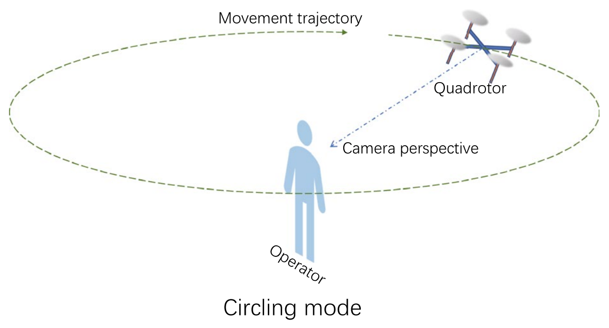


Fig. 4 In circling mode, the quadrotor circles around the operator while gazing on him

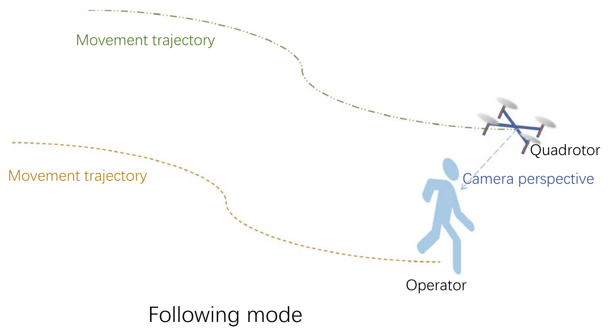


Fig. 5 In following mode, the quadrotor follows the operator

iPad and iPod have a GPS module, so that the quadrotor can basically follow the positions in GPS data transmitted from the mobile phone while retaining the distance. In the circling mode, the quadrotor in real time calculates a circular trajectory with the radius while centering the GPS position, and tries to follow it.

Let us explain a little bit more in detail using an example. In the circling mode, the quadrotor flies around the operator while retaining a constant radius and gazing on him, in which the location information of the iPhone is used to guide the quadrotor tracking the flight path. The positions of the operator and the quadrotor are displayed on the iPhone screen. The quadrotor can fly around the operator with a set radius, e.g., 20 m, while facing the front direction to him at all times. When the operator moves, the quadrotor also can follow the movement. In this case, a camera mounted under the quadrotor can automatically follow the operator to accurately monitor him.

When the quadrotor is flying in a circling mode, the operator can always stop the mode and change to the following mode using the iPhone. In the following mode, which is set by default, the quadrotor follows the operator keeping a constant distance. In this case also, the location information of the iPhone is used. Figures 6 and 7 show the successful experimental scenes of the circling and following modes.

In future work, some image recognition technology should be implemented to the quadrotor to cope with the automatic identification and tracking of the target without using GPS information [10, 11].

3.2 Mission planning function

To enable the quadrotor to achieve more complex tasks based on the above developed functions, a mission planning function is further developed. In this function, the Apple map available on iOS is used for an operator to input desired task points. The desired task points are set in the map in accordance with the operator's planning, so that the quadrotor can automatically fly along the task points one by one. First, the initial operation screen is displayed as shown in

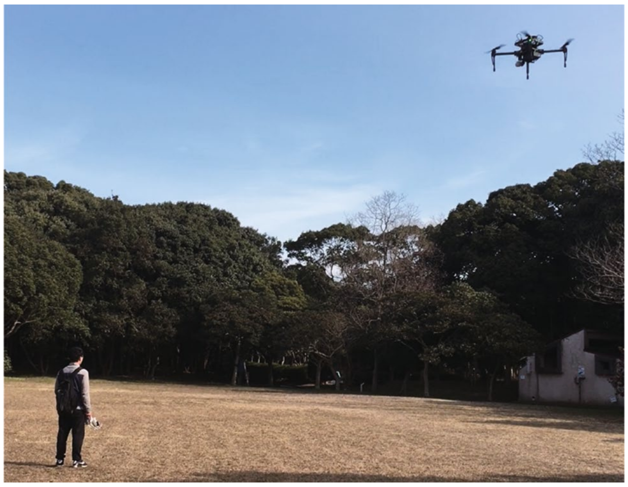


Fig. 6 Experiment of circling function while gazing on the operator

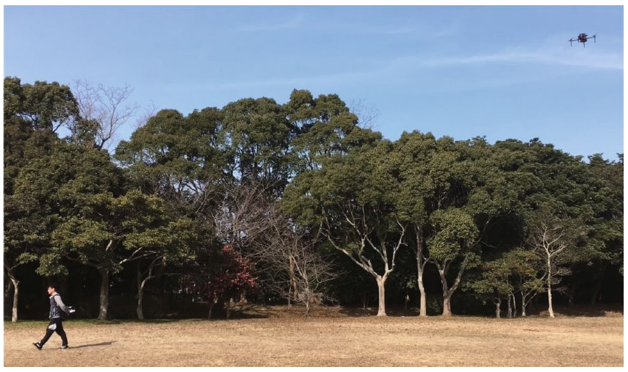


Fig. 7 Experiment of following function

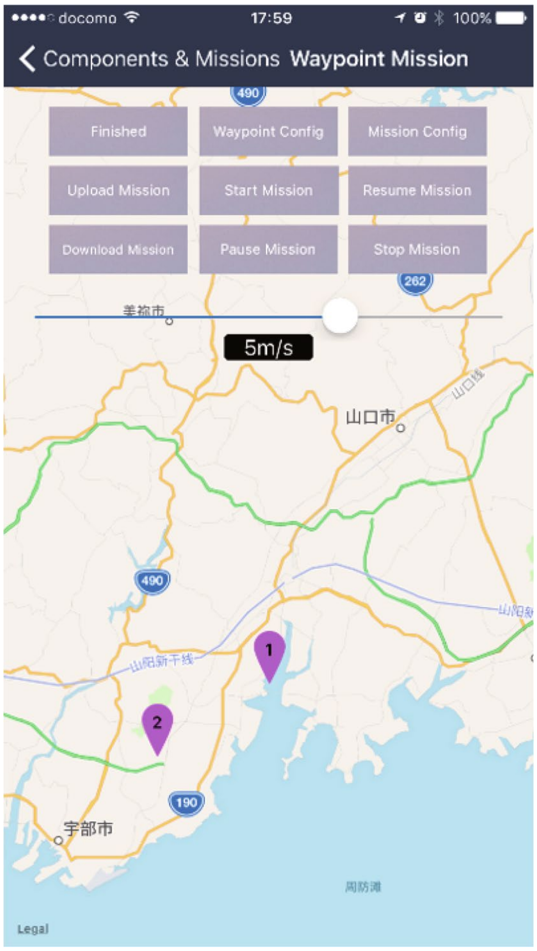


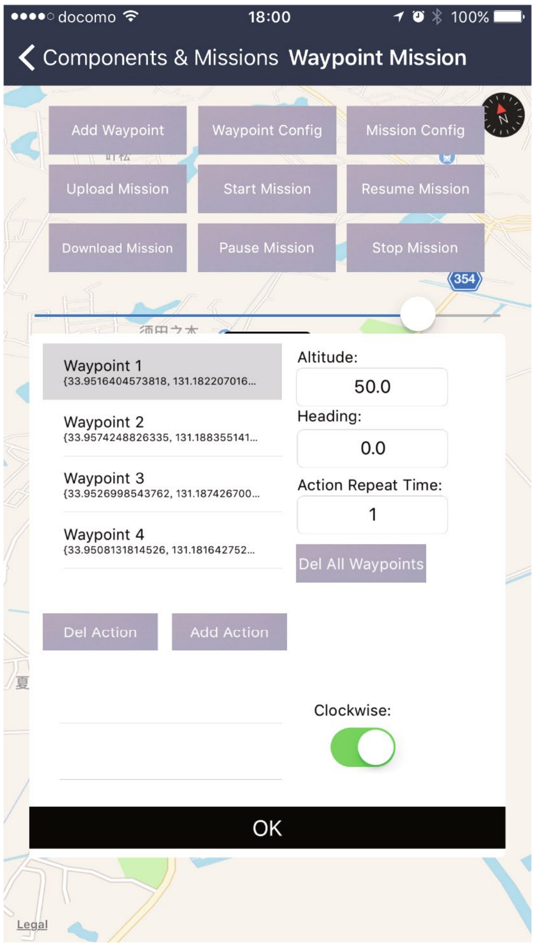
Fig. 8 Task point setting interface in mission planning function

Fig. 8, in which the operator does the initialization of the mission, then has only to set desired task points on the map and give the orders to the points according to the mission. In Fig. 8, the task points 1 and 2 are given as examples. This process is called the mission planing.

The specification of operation steps is described in detail below. In the developed interface, the operator can move the map as he wants to set, and enlarge or reduce the size. When a target point on the map is clicked once, it is set to a task point. Clicking again will add the next task point. The numbers of task points are automatically given in the order. After recording the task points as shown in Fig. 8, the operator can edit them while deleting and/or adding as shown in Fig. 9, and set flight altitude, flight angle and repeat times for each point. The corresponding latitude and longitude information at each point obtained by GPS can be monitored in the task point editing menu as shown in Fig. 9. The format of GPS information every sampling period includes latitude and longitude components whose respective lengths are 13 digits. The quadrotor can automatically hover on a task point by referring the GPS information within the error of 1.5 [m] at worst. In the current system, desired altitude values need to be set considering obstacles and reducing the likelihood of accidents. The flight angle setting allows the control of the quadrotor's perspective so that the surrounding can be observed in real time. Obstacle recognition and avoidance capabilities will be realized with the binocular vision system and the ultrasound system to accommodate the quadrotor to complex terrain missions.

In addition to editing task points, the task plan can be further refined as shown in Fig. 10. For example, the operator can add an action such as hovering, landing at a task point, returning to the starting point or the first task point, as an additional task point. Not only the maximum flight speed can be limited between two adjacent task points but also the flight direction to the starting point, the operator or the previous task point can be set through the developed interface.

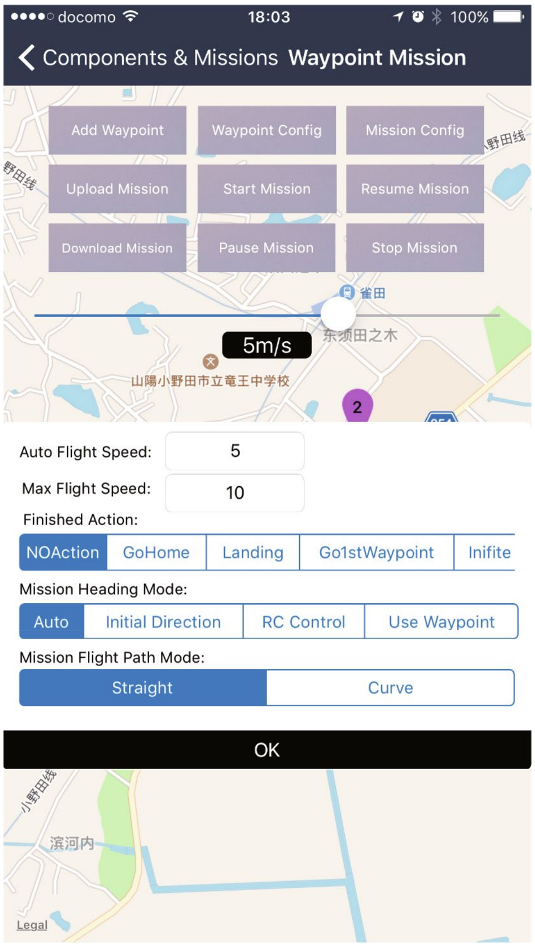
After the mission plan is completed through the three steps as shown in Figs. 8, 9 and 10, the plan is uploaded to the quadrotor. The mission plan can be easily executed by clicking a "Start Mission" button shown in Fig. 8. Of course, the operator



Waypoint 1 {33.9516404573818, 131.182207016...
Waypoint 2 {33.9574248826335, 131.188355141...
Waypoint 3 {33.9526998543762, 131.187426700...
Waypoint 4 {33.9508131814526, 131.181642752...

Altitude: 50.0
Heading: 0.0
Action Repeat Time: 1
Clockwise: on

Fig. 9 Task point editing interface



Auto Flight Speed: 5
Max Flight Speed: 10
Finished Action: [✓ NOAction] GoHome Landing Go1stWaypoint Inifite
Mission Heading Mode: [✓ Auto] Initial Direction RC Control Use Waypoint
Mission Flight Path Mode: [✓ Straight] Curve

Fig. 10 Task point action editing interface

can suspend and resume the task. A speed control slider is also designed in the interface, so that the flight speed of the quadrotor can be adjusted for better control.

Finally, an actual self-flight experiment using GPS information was conducted using a mission plan as shown in Fig. 11, in which three task points are given. The quadrotor could successfully fly from the start point to the last point, i.e., task point 3 via task points 1 and 2, so that the effectiveness and validity of the proposed system were verified.

In the development of mission-planning function described in Sect. 3.2, a core routine provided in Mobile SDK, which certainly leads the quadrotor from a task point to another one, is mainly used. Other operating functions developed on iOS shown in Figs. 8, 9, 10 and 11 are our actual contributions.

4 Conclusions

In our research, a remote control and monitoring system using an iOS device is focused on and designed for a quadrotor to be able to monitor its surrounding environment. Three basic handlers for obtaining compass information, controlling a gimbal and autopilot function for return were developed for remote control. The operator was able to send control commands, while remotely checking current quadrotor's state and watching its surrounding flight environment only using an iOS device.

In this paper, the following and circling around functions while gazing a moving object were further developed. Then, an iOS application was designed and implemented to allow the quadrotor to execute a self-flight task using GPS information, that was called the mission planning function. As a result, the iOS application enabled the quadrotor to achieve complex mission planning consisting of multiple task points. The functionality and effectiveness were evaluated and confirmed through actual experiments carried out outdoors.

In future work, the quadrotor will be able to have abilities to automatically detect and avoid obstacles during the flight, to arrive at a preset destination for complete remote autopilot function, and to identify and count moving objects using both GPS and image processing technologies.

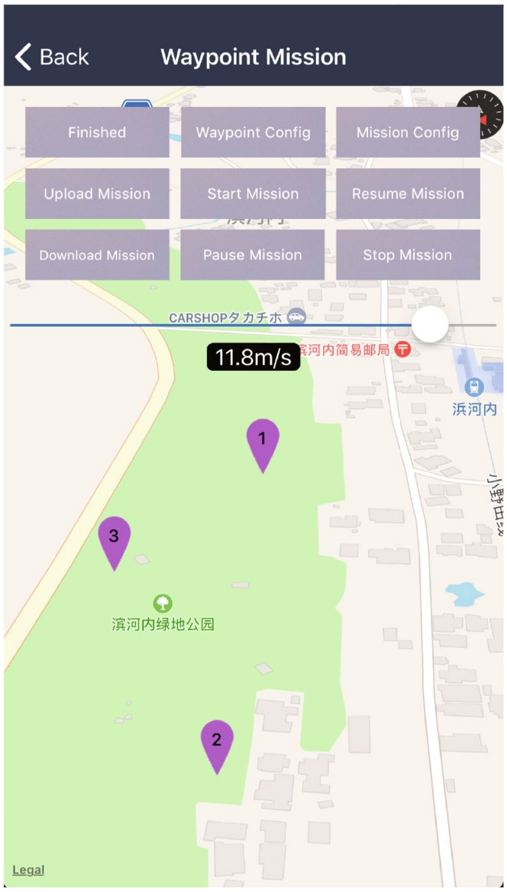


Fig. 11 An actual flight experiment could be successfully conducted using a mission plan

References

1. Iscold P, Pereira S, Torres A (2010) Development of a hand-launched small UAV for ground reconnaissance. IEEE Trans Aerosp Electron Syst 46(1):335–348
2. Mahony R, Kumar V, Corke P (2012) Multirotor aerial vehicles, modeling, estimation, and control of quadrotor. Robot Autom Mag 19(3):20–32
3. Lim H, Park J, Lee D, Kim HJ (2012) Build your own quadrotor. Robot Autom Mag 19(3):33–44
4. Lu Z, Nagata F (2017) Proposal of iPhone application for quadrotor UAV remote control—implementation of basic functions with iPhone. In: Proceedings of 22nd international symposium on artificial life and robotics (AROB2017), pp 571–575
5. Lu Z, Nagata F, Watanabe K, Maki K (2017) Habib, iOS application for quadrotor UAV remote control—implementation of basic functions with iPhone. Artif Life Robot 22(3):374–379
6. Lu Z, Nagata F, Watanabe K (2017) Development of iOS application handlers for quadrotor UAV remote control and monitoring. In: Proceedings of the 2017 IEEE international conference on mechatronics and automation (ICMA 2017), pp 513–518
7. Lu Z, Nagata F, Watanabe K (2017) iOS application for remotely controlling a quadrotor UAV and monitoring a moving object. In: Proceedings of the 27th fuzzy, artificial intelligence, neural networks and computational intelligence (FAN2017), pp 104–108
8. Lu Z, Nagata F, Watanabe K (2018) Remote control iOS application for a quadrotor UAV. In: Proceedings of 23rd international symposium on artificial life and robotics (AROB2018), pp 506–511
9. Anderson F (2014) Xcode 5 start to finish: iOS and OS X development (Developer's Library). Addison-Wesley Professional, Boston
10. Alenya G, Dellen B, Foix S, Torras C (2013) Robotized plant probing. Robot Autom Mag 20(3):50–59
11. Shuying Y (2015) Image recognition and project practice. Publishing House of Electronics Industry, Beijing **(in Chinese)**