Artif Life Robotics (2017) 22:374–379
DOI 10.1007/s10015-017-0372-3

ORIGINAL ARTICLE

iOS application for quadrotor remote control

Implementation of basic functions with iphone

Zeming Lu[1] · Fusaomi Nagata[1] · Keigo Watanabe[2] · Maki K. Habib[3]

Received: 17 February 2017 / Accepted: 12 June 2017 / Published online: 20 June 2017

Abstract With the progress of electronics technology, the development of civilian UAV (unmanned aerial vehicle) applications becomes possible. In addition, smartphones have rapidly gained popularity and become very important due to the simple operability and mobility. Hence, there is a need to have an easy and flexible way to control a UAV using such technology. In this study, a remote controller using an iOS device is developed for a quadrotor to enable remote control with easy operations. Four basic programs for obtaining compass information, controlling a gimbal, autopilot function for return, and video preview function are developed and implemented for an iOS device. The basic functionalities of the programs are evaluated and confirmed through experiments using a quadrotor and an iOS device.

Keywords Unmanned aerial vehicle · Quadrotor · iOS · iPhone · Remote control · User interface for UAV

This work was presented in part at the 22nd International Symposium on Artificial Life and Robotics, Beppu, Oita, January 19–21, 2017.

✉ Fusaomi Nagata
nagata@rs.tusy.ac.jp

[1] Department of Mechanical Engineering, Tokyo University of Science, Yamaguchi, 1-1-1, Daigaku-Dori, Sanyo-Onoda 756-0884, Japan

[2] Okayama University, Okayama, Japan

[3] American University in Cairo, Cairo, Egypt

1 Introduction

Over the last decade, there has been increasing interest in UAV systems. Various types and objectives of UAV systems have been proposed and developed from both aspects of theory and practice. For example, Iscold et al. presented the engineering design and implementation of a low cost, portable, and hand-launched small UAV-platform integrated with real data from practical flight tests. The required number of sensors and actuators was reduced without losing the feasibility and the required functionality [1]. Mahony et al. provided a tutorial introduction to modeling, estimation, and control for multi-rotor aerial vehicles such as common quadrotors. The results were useful for engineers concerned in related developments [2]. In addition, Lim et al. presented eight quadrotor OSPs (open source projects) with descriptions of their avionics, sensor composition, analysis of attitude estimation and control algorithms, and features comparison. Several research projects that used OSPs as a main flight controller were described, in which it was expected that sharing the same platform became easier with such services [3]. Sanna et al. proposed natural user interfaces and visual computing methods using an RGB-depth sensor to control the navigation of a quadrotor in indoor environments, where GPS information is not available. The proposed visual odometry algorithm allowed not only the quadrotor to autonomously navigate the environment but also the user to control complex maneuvers by gestures and body postures [4]. Furthermore, Luxman and Liu reported an implementation of back-stepping integral controller for a quadcopter which was commanded by human gestures. A novel technology using computer vision was introduced for easy operation of the quadcopter, in which methods detecting and tracking human were developed [5].

As for the utilization of smartphones, Kim et al. reported the feasibility of using a smartphone as the payload for a photogrammetric UAV system due to the availability of 3G network environment at any time or location, high-resolution images, and 3D location and attitude data were measured by built-in sensors [6]. Loianno et al. also used a smartphone as a payload for a quadrotor, in which the integration of a robot, processor, and smartphone through simple software architecture was described to build autonomous functions for navigating and mapping unknown, indoor environments [7]. Furthermore, Allen et al. introduced smartdrones as the next step-change in technology via PCs and smartphones, whose features were defined from the aspects of affordability, lightweight structure, standardization of hardware and software, autonomy, and so on [8].

However, papers adequately applying a smartphone to users' remote controller could not be seen well. If a UAV system is operated with wireless communication, a PC or specialized remote controller is generally used for controlling it . Actually, in almost conventional commercially-provided UAVs, operators have to not only use each specialized remote controller but also continue giving needed commands manually. In such cases, it seems that there are still several problems for the users to handle, for example, the complex operability of the controller; the difficulty of manual flight control when exceeding the visible range; and the difficulty of planning (programming) for designing new functions such as automatic return function and navigation referring GPS information, because easy programming environments are not well provided from quadrotors' makers. Those are the reasons why the current user interface and functionality seem to be insufficient in terms of operability and extendability at the users' side.

This paper is centered on the proposal of an original quadrotor UAV controller design based on widely spread personal smartphones. Smartphones have rapidly gained popularity and become very important due to the simple operability and mobility. It is also expected that smartphones enable us to get real-time information from various kinds of sensors built in robots and mechatronics systems including UAVs. The flexibility and ease of use to control a UAV with the portable technologies such as iPhone are enhanced with the developed techniques.

In this paper, to cope with the needs, a quadrotor UAV equipped with multiple monochrome binocular cameras, ultrasonic distance sensors, and an RGB camera is first considered as a controlled object. Then, a basic control system is developed to monitor 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. Four basic programs for obtaining compass information, controlling a gimbal, autopilot function for return and video preview function are designed and implemented for future application development. The functionalities of the programs are evaluated through the software development and actual experiments using a quadrotor and an iOS device.

2 Experimental system

2.1 Hardware of quadrotor

Figure 1 shows the hardware structure overview of the quadrotor system. Figure 2 illustrates the details of the CPU unit built in the quadrotor body. 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. For example, a GPS, compass, and micro-inertial measurement unit (MIMU) are included in Fig. 2. The MIMU can measure angles, their velocities, and accelerations concerning the attitude of the quadrotor.

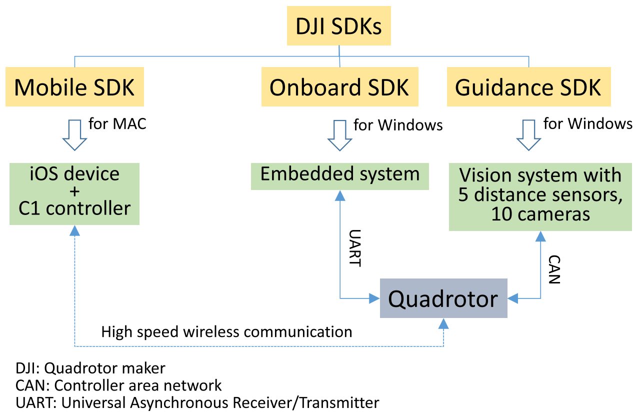
DJI SDKs
Mobile SDK
Onboard SDK
Guidance SDK
for MAC
for Windows
for Windows
iOS device + C1 controller
Embedded system
Vision system with 5 distance sensors, 10 cameras
UART
CAN
Quadrotor
High speed wireless communication
DJI: Quadrotor maker
CAN: Controller area network
UART: Universal Asynchronous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

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

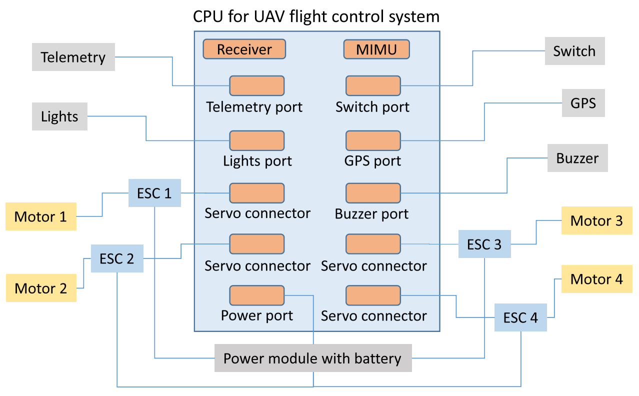

Fig. 2 Hardware details of the quadrotor

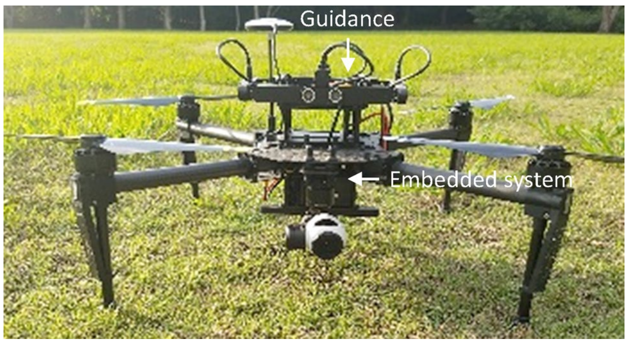

Fig. 3 Overview of the main body of the quadrotor with a guidance and an onboard embedded system

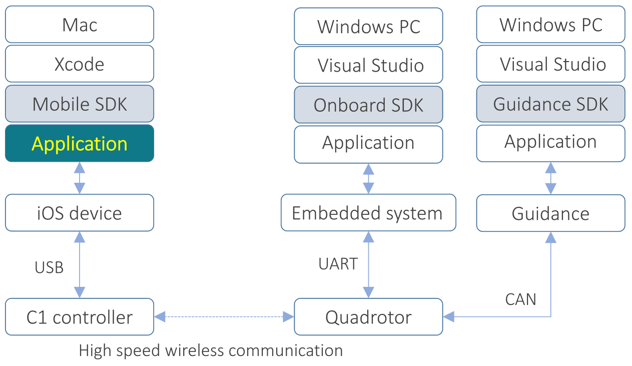

Fig. 4 Three SDKs provided by DJI and software development environment

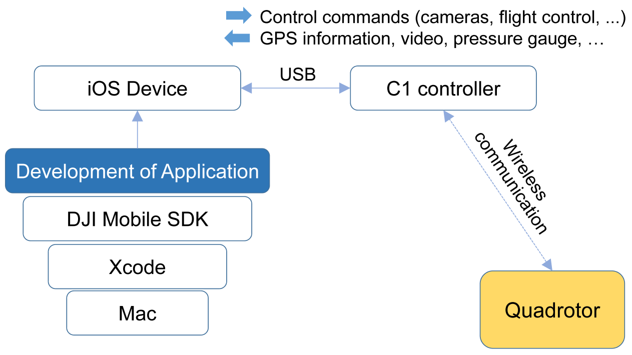
Control commands (cameras, flight control, ...)
GPS information, video, pressure gauge, …
iOS Device
USB
C1 controller
Development of Application
DJI Mobile SDK
Xcode
Mac
Wireless communication
Quadrotor

Fig. 5 Detailed data flows among iOS Device, C1 controller, and quadrotor included in Fig. 4

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, position control. The embedded system makes state information available in real time, e.g., inertial sensors, attitude, heading, velocity, position, battery remaining capacity, and 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 3 shows the hardware of the quadrotor used in this study. The maximum ground speed and the radius of controllable area are about 17 m/s and 5 km, respectiv ely.

2.2 Software development environment

Figure 4 presents the used software development environment that constitutes three software development kits (SDKs). The mobile SDK for Xcode as a MacOS allows us to build a customized mobile application for iOS device [9], as shown in Fig. 5. In addition, the onboard embedded system 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 modes 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 the C1 controller.

The onboard embedded system (Windows OS) communicates with the DJI flight controller built-in the quadrotor via a direct serial connection (UART), as shown in Fig. 6. Furthermore, DJI guidance SDK enables to customize the application and extend the functions of vision and distance sensors according to the needs of developers. Furthermore, 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, as shown in Fig. 7. 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.

3 Developed software and experiment

3.1 Monitoring compass information of quadrotor

To remotely control the quadrotor, the operator must know the current moving direction. To cope with the need, compass information is effective and obtained from the MIMU built in the CPU of the quadrotor, as shown in Fig. 2. The compass information has not only the deviation from the north direction, i.e., the data of yaw angle, but also horizontal and vertical states to both

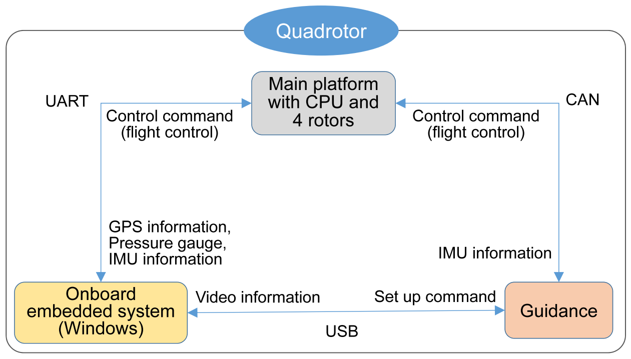
Quadrotor
Main platform with CPU and 4 rotors
UART
Control command (flight control)
CAN
Control command (flight control)
GPS information, Pressure gauge, IMU information
IMU information
Onboard embedded system (Windows)
Video information
Set up command
USB
Guidance

Fig. 6 Detailed data flows among the main platform of quadrotor, embedded system, and guidance included, as shown in Fig. 4

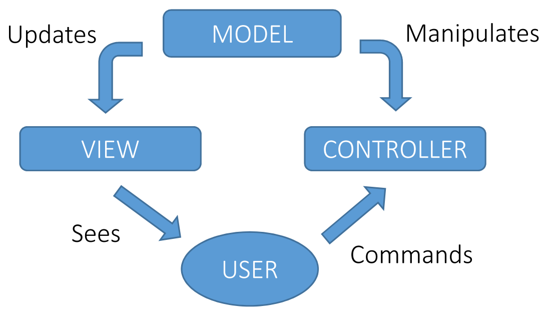

Fig. 7 Model view controller (MVC) is used for developing an iOS device application under the software development environment called Xcode

roll and pitch axes. Accordingly, the compass information enables to correct the desired moving direction by compensating roll, pitch, and yaw operations of the quadrotor. The MIMU further has three-axis acceleration sensor and gyroscope. It was confirmed from an experiment that the compass information of the quadrotor could be well monitored on iOS device.

3.2 Control function of gimbal

A gimbal with a pan and tilt structures is fixed under the quadrotor to keep the quality and stability of camera images even in a severe flight situation with disturbance. An RGB camera is attached to the quadrotor through the gimbal, as shown in Fig. 8. This feature allows to easily adjust the pan and tilt angles using iOS device, so that scenes in various direction, i.e., pictures and movies, can be obtained and viewed. It was confirmed from an experiment that the pan and tilt angles could be well adjusted using an iOS device.

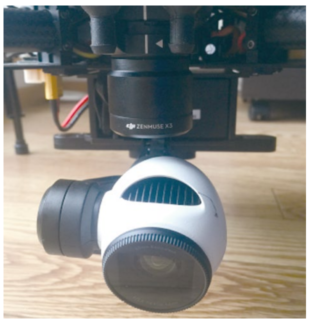

Fig. 8 Gimbal and RGB camera fixed under the quadrotor

Fig. 9 Experimental scene of automatic return function a little bit before landing

3.3 Takeoff, hovering, and automatic return function

Beginners are not familiar with the operation of a quadrotor, so that they are more prone to have misoperation or accidents. To cope with this problem, functions to automatically act takeoff, landing, and hovering were developed and implemented. These functions bring simpler and easier operation for beginners. In addition, an autopilot function for return was implemented to ensure that the quadrotor is safe and it will not get lost due to an unpredictable trouble, e.g., flying beyond the plan. For example, a quadrotor could not be directly controlled and consequently went out of view, i.e., if some undesirable signal interference or signal loss occurred.

To deal with such problems, the implemented automatic return function reduces such a serious risk that the quadrotor is out of control and consequently is lost. Figure 9 shows a scene on the way returning to the initial preset point in the automatic return function. Furthermore, Fig. 10 shows the detailed flight record, in which it was observed that the quadrotor successfully flew from the point ① to the initial point ④ via points ② and ③. Furthermore, the time variation of the distance to the initial point ④ in automatic return function is shown in Fig. 11. It is verified from these

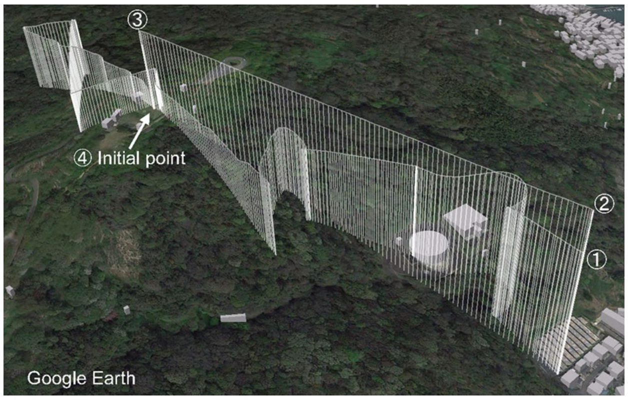

Fig. 10 Flight record of the quadrotor viewed using Google Earth

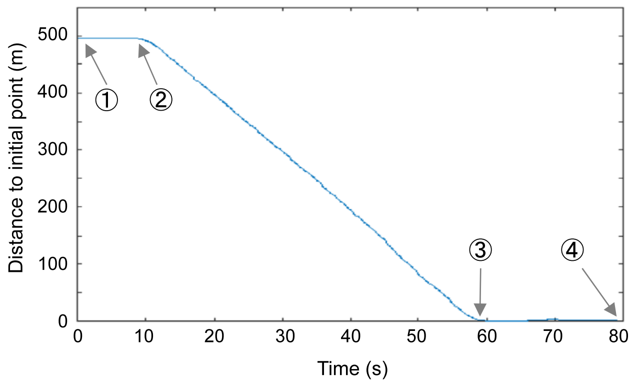

Fig. 11 Time variation of the distance to the initial point in automatic return function

figures that the quadrotor could linearly return to the initial point, where the operator set in advance using an iPhone. Although the horizontal accuracy of landing in automatic return was around ±1.5 m due to a single GPS receiver, it will be reduced to ±2 cm using three GPS receivers.

3.4 Video preview function

In this subsection, basic principles and outline of video preview function are described. This function was developed for remote monitoring of surrounding environment viewed under the quadrotor, by which a streaming movie (H.264) [10] captured by an RGB camera fixed under the quadrotor can be displayed on the iOS terminal via a remote controller C1, as shown in Fig. 5. The framework of H.264 video transfer system used in experiment is shown in Fig. 12. iOS Device can get live H.264 video data from the RGB camera shown in Fig. 8 using the Mobile SDK. The H.264 is a video coding format that is currently one of the most commonly spread formats for compression and distribution of video content.

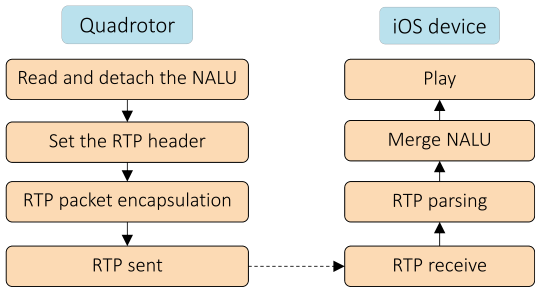

Fig. 12 Framework of image (H.264 video) transfer system, in which NALU means network abstraction layer unit

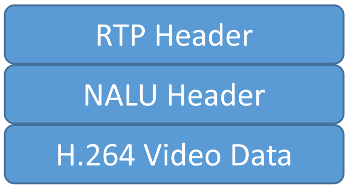

Fig. 13 Structure of the RTP file

The video previewer function for iOS devices was developed with “VideoPreviewer( )” included in Mobile SDK. “VideoPreviewer( )” is a real-time preview function of H.264 video data developed based on the RTP (real-time transport protocol). The RTP was proposed by the Audio–Video Transport Working Group of the Internet Engineering Task Force (IETF) and was described in RFC 3550 published in 2003 [11]. The RTP is an effective network protocol for delivering audio and video data through IP networks [12, 13]. The structure of the RTP file consists of RTP header, NALU one, and H.264 video data, as shown in Fig. 13 [11–13].

The software flow chart in the quadrotor-side algorithm for sending H.264 video data to an iOS device is shown in Fig. 14. First, the RGB camera’s video data are transmitted to the image transmission system, as shown in Fig. 12. The image transmission system decomposes the video data into several small data packets called NALU (network abstraction layer unit) [10] to facilitate the fast transmission. The image transmission system uses a broadcast mode similar to one-way broadcast systems. This can guarantee the stability of transmission and its distance to improve flight safety. Due to the process explained above, the image transmission system can send a complete real-time video data to the iOS device. The iOS device continuously receives, synthesizes, and displays the video data for monitoring.

In the iOS device side shown in Fig. 12, several important functions included in Mobile SDK shown in Fig. 4 were used to develop the video preview function. For example, “self.fetchCamera( )” can access the camera data; “super.didReceiveMemoryWarning( )” temporally

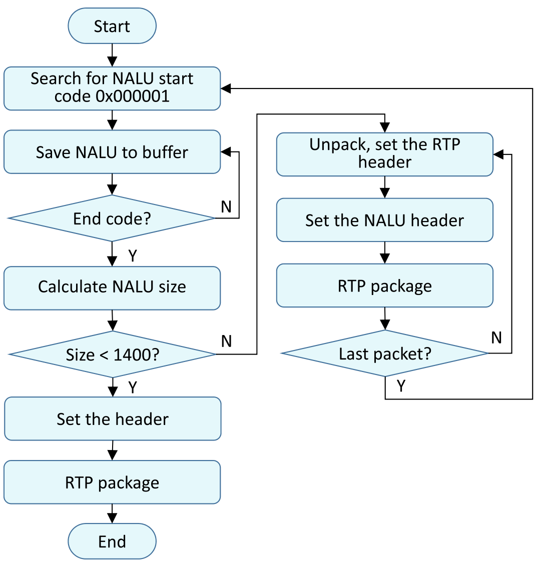

Fig. 14 Software flow chart of "RTP sent" in Fig. 12 for sending H.264 video data used in quadrotor-side algorithm

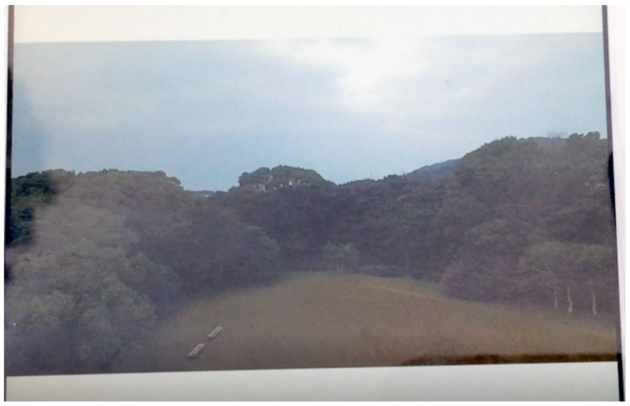

Fig. 15 Experimental scene of video preview function on iPhone's display

stores the received data in the cache; "VideoPreviewer. instance. start( )" can read the cached data; and "VideoPreviewer. instance.set( )" can display the cached data on the screen. Figure 15 shows successful experimental scene of the developed complete real-time video preview function. Of course, video capture function will be able to be further achieved in the future work.

4 Conclusions

In this study, a remote control system was focused and designed for a quadrotor to remotely control it using an iOS device. At first, three basic programs for obtaining compass information, controlling a gimbal, autopilot function for return were developed. Then, another function for remote monitoring of surrounding environment under the quadrotor was developed, by which a streaming movie (H.264) captured by an RGB camera fixed under the quadrotor could be displayed on a remote iPhone terminal. The functionality and effectiveness were evaluated and confirmed through actual experiments carried out outdoors.

In future work, the operator will be able to send control commands, while remotely checking current quadrotor's states and watching surrounding flight environment only using an iOS device. In addition, 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 GPS and image processing technologies.

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. Sanna A, Lamberti F, Paravati G, Manuri F (2013) A kinect-based natural interface for quadrotor control. Entertain Comput 4(3):179–186
5. Luxman R, Liu X (2015) Implementation of back-stepping integral controller for a gesture driven quad-copter with human detection and auto follow feature. In: Proceedings of 2015 second international conference on computer science, computer engineering, and social media (CSCESM), pp 134–138
6. Kim J, Lee S, Ahn H et al (2013) Feasibility of employing a smartphone as the payload in a photogrammetric UAV system. ISPRS J Photogramm Remote Sens 79:1–18
7. Loianno G, Cross G, Qu C et al (2015) Flying smartphones: automated flight enabled by consumer electronics. IEEE Robot Autom Mag 22(2):24–32
8. Allen R, Pavone M, Schwager M (2016) Flying smartphones: when portable computing sprouts wings. IEEE Pervas Comput 15(3):83–88
9. Anderson F (2014) Xcode 5 start to finish: iOS and OS X development (Developer's Library). Addison-Wesley Professional, New York
10. Wenger S, Hannuksela MM, Stockhammer T et al (2005) RTP payload format for H.264 video. Network Working Group, RFC 3984
11. Schulzrinne H (2003) A transport protocol or real-time applications. Audio-Video Transport Working Group of the Internet Engineering Task Force (IETF), RFC 3550
12. Durresi A, Jain R (2005) RTP, RTCP, and RTSP—internet protocols for real-time multimedia communication. The Industrial Information Technology Handbook. CRC Press, Los Angeles
13. Perkins C (2012) RTP: audio and video for the internet. Addison-Wesley, New York