BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2023
http://excel.fit.vutbr.cz

DroneMissions: A Visual Communication System for Emergency Drone Operations

Martin Rakús

Abstract

Nowadays, the primary communication channel for firefighters and mountain rescue units is voice transferred over the radio. However, radio communication might not be sufficient enough with the increasing use of drones in rescue operations, especially for communication between drone pilots and mission commanders or operators. When relying solely on radio communication, the ground team and the commander have to depend on verbal communication alone, which can lead to restricted communication and the possibility of missed or misinterpreted instructions. To address this, I developed two applications – one for the drone pilot and one for the unit commander. The pilot app includes the interface necessary for flying the drone, while the commander app connects to the pilot app and provides access to the drone video feed and other important information. The commander can send visual feedback to the pilot by drawing into the video feed, placing annotation icons, or sending flight commands for more precise cooperation between the drone and the ground team. As the solution is a stand-alone system, it is not susceptible to radio communication overload, which guarantees that commands are not lost and that it can be used even when the radio fails.

*xrakus04@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

In recent years, drones have become increasingly common in rescue operations, providing rescuers with many previously unavailable options. This project aims to implement an application allowing two-way visual communication between drone pilots and rescue commanders, enabling real-time image and drone information transmission and visual communication.

In currently existing applications, visual communication mostly works in one-way mode only. That is sharing the drone video feed with the commander or ground units. However, using this approach, the commander can communicate with the pilot using voice only. Two of the most advanced systems available are DroneSense and DJI FlightHub 2. They are both working beta applications, but one of them is restricted to one-way communication, while the other has limited streaming minutes and mapping images per month and is said to be very costly when it is released.

My solution aims to provide reliable local communication and a platform for two-way visual communication. I want to give commanders the option to intervene in the drone pilot through drawings, annotations, or image commands and, if necessary, allow a remote takeover of the drone for more precise maneuvering.

2. Pilot application

Instead of reinventing the wheel, this application is based on the established and successful UI design of drone control apps such as DJI Fly or Litchi. The main focus is on the video feed from the drone, with drone information like battery level, speed, distance from the takeoff point and others displayed in the foreground of the interface (see Figure 1).

In addition to drone-related information, this application also includes elements related to remote collaboration, such as control of the connection, the ability to change the user ID, a switch for allowing the visibility for connection in the commander's application, displaying connected applications and the option to disconnect individual devices.

From a communication standpoint, this application displays drawings and annotation icons received from any connected commander application right on top of the drone video feed, along with notifications con-

taining images of commands from the commander. In the event of an emergency where the pilot needs to notify the commander, the pilot application provides a red button that, when pressed, immediately sends an alert to all connected commander applications with the drone's current location and altitude.

3. Commander app

An application designed for the commander contains a menu displaying currently available pilot applications to which a connection request can be sent. Once connected, the application switches to a user interface that includes a live stream of the drone's video, important drone status data, and a map (see Figure 2).

The basic and most important functionality is the ability to draw directly on this video, with the drawings being automatically transmitted in real-time to the pilot application. Those drawings are automatically erased after 5 seconds. The app has a panel that allows changing the color of the drawing and switching to permanent drawing mode, where the automatic erasing is disabled. Erasing any drawing is possible with an eraser tool (see Figure 3).

The next panel contains image commands with basic drone and camera control options, which can be immediately sent to the pilot as a warning (see Figure 4). Another menu includes the selection of annotation icons for marking fire, people, objects, flammable, toxic, or otherwise dangerous materials, hydrants, and others. These icons can be easily grabbed and dragged to any part of the screen where they stay until the drone moves (see Figure 5).

In cases when radio communication could be overloaded, messaging is included as an additional form of communication. Therefore, the commander's application contains a menu with a text input for entering any text up to a maximum of 50 characters, which can be sent as a notification to the pilot application. If necessary, the commander can request remote control of the drone and, after the pilot's confirmation, control its movement and gimbal with simple virtual joysticks.

This application also includes a map that displays the location of the commander, drone location, and home location of the drone, as well as the flight path of the drone and its surroundings.

4. Technical details

I developed these applications using the Swift programming language, and they are compatible with DJI drones and Apple devices. I leveraged the DJI development kit to access all the necessary information from the drone, including a live video feed. To establish a local connection, I used the MultipeerConnectivity framework[1], which supports transmitting up to 4 frames per second when connected to the mobile hotspot provided by the pilot device and up to 8 frames per second when connected to a Wi-Fi network or hotspot from another device.

MultipeerConnectivity also allows multiple commander applications to connect to a single pilot, enabling them to view the drone's footage from multiple devices. However, as the number of connected devices increases, the transferred information also increases, which is countered by a decrease in the frame rate. The maximum number of devices that can be connected at once depends on network speeds and reliability, but by default, it can connect up to 8 devices.

The pilot application sends drone footage as an image and a JSON data structure containing all the relevant drone information. The commander application receives this data and processes it in real-time.

When drawing, the commander application sends a JSON data structure containing the drawing ID, peer ID, and coordinates of the last point of the drawing on the screen, and when placing annotation icons, the name and position of the icon. During remote control of the drone, the commander application sends the joystick position every second to the pilot application, but only when the joystick is not in the zero position and continues until either the commander or the pilot stops the remote control.

5. Conclusions

Following extensive testing with various Apple devices and DJI Air 2 and DJI Mini 2 SE drones, two fully functional applications have been developed as part of this project, offering a broad range of use cases. The real-time and stable communication between them enables users to view live drone video from a secondary device and command the pilot without requiring any external communication.

During the development, the application and its final design were consulted with the firefighters from HZS Brno, who provided valuable feedback and recommendations for further development, for example, the implementation of draggable annotation icons, which have since been fully implemented.

[1] https://developer.apple.com/documentation/multipeerconnectivity