Dronacharya: Enabling guided collaboration in a disconnected environment

18843: Mobile and Pervasive Computing
Carnegie Mellon University

Akshunna Vaishnav
Mentored by Jim Blakley

Background
There are often situations in the real world that require technological operation in a disconnected environment, i.e. one that does not have access to the internet at large or any form of cloud. There are three necessary components in such a situation:
• An infrastructure for network access (JIT Cloudlet [1])
• Computing resources (JIT Cloudlet)
• Useful applications and services (this project)

![Diagram of Dronacharya components: OpenScout, Signaling Server, Server Container, etc.]

Fig. 1: An overview of Just-in-Time Cloudlet components.

In this project, I have created an application for communication, collaboration, and exploration in a disconnected scenario. Purpose: unlock communication and collaboration in use-cases for the Just-In-Time Cloudlet.

Dronacharya Use Case: Search and Rescue

Search and Rescue: allowing first responders to
1) collaborate with ease
2) locate people quickly, through the use of a drone, when they have no pre-existing network connectivity

The Application
The application provides a familiar interface to allow users to view drone footage and communicate with each other. Drone snapshots are taken to preserve a moment that was captured.

The application allows for:
• Asynchronous connect/disconnect
• Shared drone footage viewing
• Shared snapshot functionality

The application uses these technologies:
• OpenScout [2], the pre-existing object detection engine was used, changes were needed to transmit detected output to clients
• Industry-standard WebRTC [3] facilitates peer-to-peer collaboration, and requires NO cloud infrastructure, unlike Zoom, Skype, Discord, etc.

System Architecture

The application comprises of four main components:
• OpenScout, drone footage extraction + analysis
• OpenScout-to-WebRTC adapter, to send analyzed footage
• A signaling server, to facilitate WebRTC peer connections
• Clients, who see analyzed drone footage and talk to each other

The front-end is a JavaScript-based WebApp that prioritizes user-friendliness. Web sockets [4] are used for signaling.

Demo
A simple demo which uses image streams instead of live camera footage.

Note that the Web Application is accessible from any device through a web browser, and join/leave is completely automatic.

Drone footage can be paused, and a single screenshot of the drone footage can be taken for analysis.

Challenges + Learnings
There were a few challenges along the way:
• Limited WebRTC API/resources for Python or Android
• Moving data across Python-JavaScript abstraction layer
• Limited exposure to JavaScript, WebSockets, Docker
• Allow asynchronous join/disconnect, clients and drone

Through this project, I learned:
• OS-agnostic often simpler than multiple native apps
• Cloud-native at the edge JIT Cloudlet applications are worth exploring

Future Work
Future directions for development include:
• Drone image buffer can be replaced by a service which takes OpenScout output and conveys it to WebRTC Server container.
• Video recording of the drone footage
• Containerized deployment and real-world disconnected test

References
[3] [WebRTC, WebRTC API], https://webrtc.org/docs/