Real-time Drone Surveillance Integration:

This system will have the following components:

1--Drone Control (autonomous movement) - A drone platform like DJI SDK or PX4 can be used to control the drone autonomously.

2--Video Streaming - Using RTSP (Real-Time Streaming Protocol) or RTMP to stream video from the drone to a server or front-end.

3--Real-Time Data Processing - For traffic, pollution, and infrastructure monitoring.

4--Web Interface - Display live video feeds and sensor data.

**DRONE CONTROL—**

import dji\_sdk

class DroneControl:

def \_\_init\_\_(self):

self.drone = dji\_sdk.Drone()

def takeoff(self):

self.drone.takeoff()

print("Drone is taking off.")

def land(self):

self.drone.land()

print("Drone is landing.")

def move\_forward(self, distance):

self.drone.move\_forward(distance)

print(f"Drone is moving forward {distance} meters.")

def hover(self):

self.drone.hover()

print("Drone is hovering.")

# Create an instance and control the drone

drone = DroneControl()

drone.takeoff()

drone.move\_forward(20) # Move 20 meters forward

drone.hover()

drone.land()

**VIDEO STREAMING**—

import cv2

# Drone Video Feed URL (RTSP Stream URL from DJI drone)

rtsp\_url = "rtsp://<drone-ip>:554/live"

# Open the RTSP stream

cap = cv2.VideoCapture(rtsp\_url)

if not cap.isOpened():

print("Error: Couldn't connect to the video feed.")

else:

print("Connected to the video feed.")

while True:

ret, frame = cap.read()

if not ret:

break

# Process the video frame (e.g., detect traffic or pollution)

# Here, we display it in a window

cv2.imshow('Drone Video Feed', frame)

# Exit the loop if the user presses 'q'

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Release resources

cap.release()

cv2.destroyAllWindows()

**REAL TIME DATA PROCESSING—**

import cv2

import numpy as np

# Load YOLO model (pre-trained)

net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i-1] for i in net.getUnconnectedOutLayers()]

# Capture the video feed (drone stream)

cap = cv2.VideoCapture(rtsp\_url)

while True:

ret, frame = cap.read()

if not ret:

break

# Prepare image for YOLO object detection

blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

net.setInput(blob)

outs = net.forward(output\_layers)

# Process the outputs (detect objects like cars)

class\_ids = []

confidences = []

boxes = []

height, width, channels = frame.shape

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.5:

# Detected object is car, traffic-related

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

boxes.append([center\_x, center\_y, w, h])

confidences.append(float(confidence))

# Show video with detected traffic objects (cars)

for box in boxes:

x, y, w, h = box

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.imshow('Drone Video with Traffic Detection', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**REAL TIME MONITORING—**

1— BACKEND—

const WebSocket = require('ws');

const express = require('express');

const app = express();

const wss = new WebSocket.Server({ noServer: true });

app.use(express.static('public'));

wss.on('connection', ws => {

console.log('New WebSocket connection');

// Simulate sending drone video frames (processed as base64 encoded string)

setInterval(() => {

let videoFrameBase64 = getVideoFrameBase64(); // Capture frame from drone

ws.send(videoFrameBase64); // Send to frontend

}, 100); // Send frame every 100 ms

});

const server = app.listen(3000, () => {

console.log('Server running on port 3000');

});

server.on('upgrade', (request, socket, head) => {

wss.handleUpgrade(request, socket, head, ws => {

wss.emit('connection', ws, request);

});

});

function getVideoFrameBase64() {

// Simulate video frame encoding as base64

return 'base64EncodedVideoFrameData';

}

**2—FRONTEND—**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Drone Surveillance Feed</title>

</head>

<body>

<h1>Live Drone Surveillance</h1>

<video id="video" width="640" height="480" autoplay></video>

<script>

const videoElement = document.getElementById('video');

const socket = new WebSocket('ws://localhost:3000');

socket.onmessage = function(event) {

// Convert base64 video data to Blob and play it in the video element

const binaryData = atob(event.data);

const buffer = new ArrayBuffer(binaryData.length);

const view = new Uint8Array(buffer);

for (let i = 0; i < binaryData.length; i++) {

view[i] = binaryData.charCodeAt(i);

}

const videoBlob = new Blob([view], { type: 'video/mp4' });

const videoURL = URL.createObjectURL(videoBlob);

videoElement.src = videoURL;

};

</script>

</body>

</html>