**Drone Programming Simulation for Crime Prevention**

**1. Overview / Objective**

**Drones are increasingly proposed for crime prevention and urban monitoring. The objective of this simulation is to demonstrate how a drone can systematically visit crime hotspots identified in our analysis, providing real-time data for authorities.**

**The simulation focuses on:**

* **Mapping hotspots in a grid-based 3D frame.**
* **Generating waypoints for efficient path planning.**
* **Simulating drone flight using simple algorithms.**

**2. 3D Grid and Hotspot Mapping**

**To monitor a city or area, we can represent the area as a 1 km × 1 km grid. Each grid cell represents a spatial zone the drone can visit.**

* **Hotspots identified from crime analysis are assigned as points of interest (POIs) within the grid.**
* **Each hotspot has coordinates (x, y) within the grid, representing its central location.**
* **Example Python representation:**

**# Define grid size (1km x 1km) and hotspots**

**grid\_size = 1000 # meters**

**hotspots = {**

**'City Center': (200, 800),**

**'Industrial Area': (600, 400),**

**'Residential Zone': (800, 900)**

**}**

**3. Waypoint Generation & Path Planning**

**Once hotspots are mapped, the drone needs an efficient path. Common approaches include:**

**a) Lawn Mower Pattern**

* **Covers the entire area in parallel sweeps.**
* **Ensures the drone surveys all grid cells, not just hotspots.**
* **Example:**

**# Simple lawnmower pattern across grid**

**waypoints = []**

**step = 200 # meters between sweeps**

**for y in range(0, grid\_size + step, step):**

**row = range(0, grid\_size + step, step)**

**if y // step % 2: # zigzag for efficiency**

**row = reversed(row)**

**for x in row:**

**waypoints.append((x, y))**

**b) Nearest-Neighbour Traversal**

* **Visits all hotspots efficiently by always going to the closest next point.**
* **Reduces travel time and battery usage.**
* **Example:**

**import math**

**def distance(a, b):**

**return math.sqrt((a[0]-b[0])\*\*2 + (a[1]-b[1])\*\*2)**

**# Greedy nearest-neighbour path**

**current = (0, 0)**

**path = [current]**

**remaining = list(hotspots.values())**

**while remaining:**

**nearest = min(remaining, key=lambda p: distance(current, p))**

**path.append(nearest)**

**current = nearest**

**remaining.remove(nearest)**

**4. Flight Methods**

**To simulate or operate the drone:**

1. **Autonomous Flight – drone follows waypoints automatically using onboard GPS.**
2. **Manual Override – operator can take control in emergencies.**
3. **Altitude Control – maintain safe height to avoid obstacles.**
4. **Obstacle Avoidance – simulated using virtual barriers in Python or through sensors in real drones.**
5. **PID Control – ensures smooth navigation and accurate positioning.**

**5. Simulation & Visualization**

**We can plot the drone path using matplotlib to visualize the trajectory:**

**import matplotlib.pyplot as plt**

**x\_coords, y\_coords = zip(\*path)**

**plt.figure(figsize=(6,6))**

**plt.plot(x\_coords, y\_coords, marker='o', linestyle='-')**

**for label, coord in hotspots.items():**

**plt.text(coord[0]+10, coord[1]+10, label)**

**plt.title('Drone Path Visiting Hotspots')**

**plt.xlabel('X coordinate (meters)')**

**plt.ylabel('Y coordinate (meters)')**

**plt.grid(True)**

**plt.show()**

***This figure shows the drone visiting all hotspots efficiently.***

**6. Assumptions and Limitations**

* **Battery life is sufficient to complete the planned path.**
* **No dynamic obstacles (vehicles, people) are considered in this simulation.**
* **GPS is accurate for positioning in the grid.**
* **Simulation simplifies real drone physics (e.g., wind, acceleration).**

**7. Summary**

* **Hotspot Mapping: 3D grid-based representation with points of interest.**
* **Path Planning: Lawn mower for complete coverage; nearest-neighbour for hotspot efficiency.**
* **Flight Methods: Autonomous navigation, obstacle avoidance, altitude control, PID-based stability.**
* **Visualization: Python matplotlib plots allow verification of drone paths.**

**This simulation demonstrates how drones can enhance urban monitoring and crime prevention, providing actionable insights for authorities.**

**How I would simulate a drone visiting hotspots:**  
First I create a simple 3D frame of the patrol area — a 1 km × 1 km square — and overlay a grid (for example 20 × 20 cells at 50 m resolution). I map the hotspots found in my crime analysis to coordinates in that frame, and mark them as points of interest with a fixed surveillance altitude (e.g., 50 m).  
For planning, I use two complementary approaches. If I need full area coverage (for mapping or systematic patrolling) I perform a lawnmower sweep — a back-and-forth pattern that guarantees all cells are observed. If I only need to visit a handful of hotspots quickly I run a nearest-neighbour greedy tour that visits the nearest unvisited hotspot until all are covered. Both planners output ordered waypoints (x, y, z) which are then fed to an autopilot simulation. Low-level flight control is handled by PID controllers for stability and altitude hold, while obstacle detection triggers local re-planning. The simulation logs path length, mission time, and whether each hotspot was visited — useful metrics to judge effectiveness.