Subtask 1: Static Formation from Grid to Handwritten Name

Student Name

1 Introduction and Problem Definition

The objective of Subtask 1 is to simulate the motion of a synchronized drone swarm transitioning from an initial uniform configuration to a static handwritten target shape. The handwritten shape is provided as a raster image and represents the desired final formation of the swarm. This task corresponds to a *static target tracking problem* and is naturally formulated as an *Initial Value Problem (IVP)* for a system of ordinary differential equations.

Inputs

- Image of a handwritten name
- Number of drones N
- Initial positions $\{x_i(0)\}_{i=1}^{N}$ arranged on a uniform grid

Outputs

- Time-continuous trajectories $x_i(t)$ of all drones
- Visualization of the swarm converging to the handwritten shape

The key requirements are smooth motion, collision avoidance, and convergence to the target without oscillations.

2 Mathematical Model

2.1 State Variables

Each drone $i \in \{1, \dots, N\}$ is described by:

- Position: $x_i(t) \in \mathbb{R}^2$
- Velocity: $v_i(t) \in \mathbb{R}^2$

The full system state is:

$$Y(t) = (x_1, v_1, x_2, v_2, \dots, x_N, v_N) \in \mathbb{R}^{4N}.$$

2.2 Governing Equations (IVP)

The swarm dynamics are governed by the following system:

$$\dot{x}_i = v_i \cdot \min\left(1, \frac{v_{\max}}{\|v_i\|}\right), \tag{1}$$

$$\dot{v}_i = \frac{1}{m}\Big[k_p(T_i - x_i) + \sum_{j \neq i} f_{\text{rep}}(x_i, x_j) - k_d v_i\Big], \tag{2}$$

with initial conditions:

$$x_i(0) = x_{i,0}, \qquad v_i(0) = 0.$$

Here $T_i \in \mathbb{R}^2$ denotes the static target point assigned to drone i.

2.3 Velocity Saturation

To respect physical speed limits of real drones, velocity saturation is applied:

$$\dot{x}_i = v_i \cdot \min\left(1, \frac{v_{\max}}{\|v_i\|}\right).$$

This ensures $\|\dot{x}_i\| \leq v_{\max}$ for all time.

2.4 Collision Avoidance Model

Collision avoidance is enforced using a pairwise repulsive force:

$$f_{\text{rep}}(x_i, x_j) = \begin{cases} k_{\text{rep}} \dfrac{x_i - x_j}{\|x_i - x_j\|^3}, & \|x_i - x_j\| < R_{\text{safe}}, \\ 0, & \text{otherwise.} \end{cases}$$

This inverse-square-type force becomes significant only within a safety radius R_{safe} and guarantees collision-free motion.

3 Target Extraction and Assignment

3.1 Target Point Extraction

The handwritten image is converted to grayscale and processed using Canny edge detection. Let $\mathcal{E}$ denote the set of detected edge pixels. From $\mathcal{E}$, exactly N target points are sampled uniformly:

$$\{T_i\}_{i=1}^{N} \subset \mathcal{E}.$$

3.2 Assignment Problem

Each drone must be assigned to exactly one target point. The cost of assigning drone i to target j is defined as the Euclidean distance:

$$D_{ij} = \|x_i(0) - T_j\|.$$

A greedy nearest-neighbor algorithm is used to compute a one-to-one assignment that approximately minimizes the total travel distance. While not globally optimal, this approach has $\mathcal{O}(N^2)$ complexity and is computationally efficient for large swarms ($N \approx 2000$).

4 Numerical Method

4.1 Time Integration

The system of ODEs is solved numerically using the classical fourth-order Runge–Kutta (RK4) method, which provides:

- Fourth-order accuracy in time ($\mathcal{O}(h^4)$)
- Good stability properties for damped second-order systems

The time step Δt is fixed, and multiple integration steps are performed per animation frame.

4.2 Justification of RK4

The drone dynamics include stiff components due to strong spring and repulsive forces. Compared to Euler or RK2 methods, RK4 achieves higher accuracy without excessively small time steps, making it suitable for real-time visualization.

5 Stability and Convergence Analysis

5.1 Linearized Single-Drone Model

Neglecting repulsion and considering a single drone with target at the origin, the dynamics reduce to:

$$\ddot{x} + k_d\dot{x} + k_p x = 0.$$

The characteristic equation is:

$$\lambda^2 + k_d\lambda + k_p = 0.$$

For the chosen parameters ($m = 1, k_p = 25, k_d = 12$), both eigenvalues are real and negative, implying asymptotic stability.

5.2 Damping Regime

The damping ratio is:

$$\zeta = \frac{k_d}{2\sqrt{mk_p}} > 1,$$

which corresponds to an overdamped system, guaranteeing monotonic convergence to the target without oscillations.

6 Selected Code Segments and Their Mathematical Meaning

This section presents representative parts of the Python implementation and explains how they directly correspond to the mathematical model in (1)–(2). This explicit mapping demonstrates consistency between theory and implementation.

6.1 State Derivative Computation

The acceleration used in (2) is implemented in code by combining attraction, damping, and repulsion:

```
acc = kp * (target - pos) - kd * vel + rep_force
```

Here:

- `kp * (target - pos)` corresponds to the spring attraction term $k_p(T_i - x_i)$
- `-kd * vel` corresponds to viscous damping $-k_d v_i$
- `rep_force` accumulates collision avoidance forces $\sum_{j \neq i} f_{\text{rep}}(x_i, x_j)$

6.2 Velocity Saturation

To enforce the constraint $\|\dot{x}_i\| \leq v_{\max}$, saturation is applied:

```
speed = np.linalg.norm(vel)
if speed > V_MAX:
    vel = vel * (V_MAX / speed)
```

This guarantees physically feasible velocities throughout the simulation.

6.3 Collision Avoidance

The repulsive force model

$$f_{\text{rep}}(x_i, x_j) = k_{\text{rep}} \frac{x_i - x_j}{\|x_i - x_j\|^3}$$

is applied only for drones within the safety radius:

```
if dist < R_SAFE:
    force += K_REP * diff / (dist**3)
```

This prevents collisions while remaining inactive at safe separations.

7 Task–Solution Mapping Table

Project Task	**Implemented Solution**	**Mathematical Basis**	**Code Reference**
Swarm initialization	Uniform grid	Initial condition $x_i(0)$	`generate_grid()`
Target extraction	Edge detection & sampling	$T_i \in \mathcal{E}$	`extract_points()`
Drone–target assignment	Greedy matching	$\min \sum \|x_i - T_j\|$	`greedy_assignment()`
Target attraction	Spring force	$k_p(T_i - x_i)$	`kp * (target - pos)`
Damping	Velocity feedback	$-k_d v_i$	`-kd * vel`
Collision avoidance	Repulsion	f_{rep}	`repulsion()`
Speed limit	Saturation	$\|v_i\| \leq v_{\max}$	velocity scaling
Time integration	RK4	4th-order IVP solver	`Swarm.step()`

8 Practical Considerations and Limitations

- The greedy assignment does not guarantee global optimality, but performs well in practice and is efficient for large N.
- Repulsive force computation is accelerated using spatial hashing to avoid $\mathcal{O}(N^2)$ scaling in dense swarms.
- Very dense edge regions may cause local clustering, mitigated by damping and repulsion.

9 Conclusion

Subtask 1 formulates the static formation problem as a well-posed IVP with physically interpretable forces. The combination of spring attraction, damping, collision avoidance, and velocity saturation ensures stable, smooth, and collision-free convergence to the handwritten target. This subtask establishes the mathematical and numerical foundation for the subsequent dynamic extensions in Subtasks 2 and 3.