Autonomous Drone Swarm Simulation

Modeling Coupled Multi-Agent Systems via Potential Fields

Andria Beridze

Computer Science Department

January 27, 2026

1. Project Scope & Objectives

Goal: Design a physics-based simulation for a swarm of **1,000 autonomous drones** capable of performing coordinated tasks without centralized control.

Core Requirements:

- **Scale:** The system must handle $N = 1000$ agents in real-time.
- **Physics:** Must simulate inertia, air resistance, and actuation limits.
- **Safety:** Zero collisions allowed during high-speed maneuvers.
- **Tasks:**
  1. Static Formation (Image Reconstruction)
  2. Dynamic Transition (Morphing text)
  3. Video Tracking (Pursuit of moving target)

2. Mathematical Formulation: System of ODEs

We model the swarm as a system of N coupled Ordinary Differential Equations (ODEs).

The Initial Value Problem (IVP): Given the state vector $\mathbf{S}(t) = [\vec{r}_1, \ldots, \vec{r}_N, \vec{v}_1, \ldots, \vec{v}_N]^T$, we solve for $t > 0$ subject to:

$$\begin{cases} \frac{d\vec{r}_i}{dt} = \vec{v}_i \\ m\frac{d\vec{v}_i}{dt} = \vec{F}_{net} = \vec{F}_{att}(\vec{r}_i) + \sum_{j \neq i} \vec{F}_{rep}(\vec{r}_{ij}) - k_d \vec{v}_i \\ \vec{r}_i(0) = \vec{r}_{i,0}, \quad \vec{v}_i(0) = \mathbf{0} \end{cases} \tag{1}$$

This represents a second-order coupled system where every agent's motion depends on the relative positions of its neighbors.

3. Physical Variables: State Quantities

These variables describe the instantaneous physical state of each drone.

Symbol	**Name**	**Physical Interpretation**
$\vec{r}_i$	Position Vector	3D position (x, y, z) of drone i in world coordinates.
$\vec{v}_i$	Velocity Vector	Time derivative of position; determines direction and speed of motion.
$\vec{r}_{ij}$	Relative Position	Vector from drone j to drone i, used for collision avoidance.
$\|\|\vec{r}_{ij}\|\|$	Distance	Euclidean distance between drones i and j.

These variables evolve continuously through numerical integration.

4. Physical Variables: Constants & Gains

These parameters control the physical behavior and stability of the swarm.

Symbol	**Name**	**Physical Interpretation**
m	Mass	Determines inertia; higher mass resists acceleration ($F = ma$).
k_p	Attraction Gain	Controls how aggressively drones move toward their assigned targets.
k_d	Damping Coefficient	Velocity-dependent drag force that removes energy and prevents oscillations.
k_{rep}	Repulsion Gain	Strength of the collision avoidance force between nearby drones.
R_{safe}	Safety Radius	Distance threshold at which repulsion forces become active.

Proper tuning of these constants is essential for stability and safety.

5. Force Logic: Attraction & Damping

The net force $\vec{F}_{net}$ is composed of three components.

1. Goal Seeking (Attraction)

Modeled after **Hooke's Law** (Spring Force). It pulls the drone toward the target $\vec{r}_{target}$.

$$\vec{F}_{att} = -k_p(\vec{r}_i - \vec{r}_{target})$$

If the drone is far away, this force is strong. As it arrives, the force drops to zero.

3. Stabilization (Damping)

A friction force opposite to velocity. Essential for stability, otherwise the drones would orbit the target forever.

$$\vec{F}_{damp} = -k_d \cdot \vec{v}_i$$

6. Force Logic: Collision Avoidance

The most critical component for swarm safety is the Repulsion Force.

2. Collision Avoidance (Repulsion)

An **Inverse-Square Force** that acts like a magnetic field. It approaches infinity as distance $d \to 0$.

$$\vec{F}_{rep} = \sum_{j \neq i} k_{rep} \left(\frac{1}{||\vec{r}_{ij}||} - \frac{1}{R_{safe}} \right) \frac{\hat{r}_{ij}}{||\vec{r}_{ij}||^2}$$

Note: This force is only calculated if $||\vec{r}_{ij}|| < R_{safe}$. This creates a "local" interaction model, where drones only care about their immediate neighbors.

7. Numerical Integration: RK4

Since the repulsion force creates sharp "spikes" when drones get close, standard Euler integration ($x_{t+1} = x_t + v \cdot dt$) causes "energy explosions" (instability).

Solution: Runge-Kutta 4 (RK4) I implemented a custom RK4 solver. It samples the slope at 4 different points within one time-step (dt) to predict the next position with $O(dt^4)$ accuracy.

```
# RK4 Integration Logic
def rk4_step(pos, vel, targets, dt):
    # k1: Slope at start
    k1_v = compute_forces(pos, vel, targets)
    # k2: Slope at midpoint (using k1)
    k2_v = compute_forces(pos + k1_p*0.5*dt, ...)
    # ...
    # Weighted Average
    new_vel = vel + (dt / 6.0) * (k1_v + 2*k2_v + 2*k3_v +
    k4_v)

```

8. Task 1: Static Formation (Name Generation)

Objective: Arrange 1,000 drones to form the name "Andria Beridze" in 3D space.

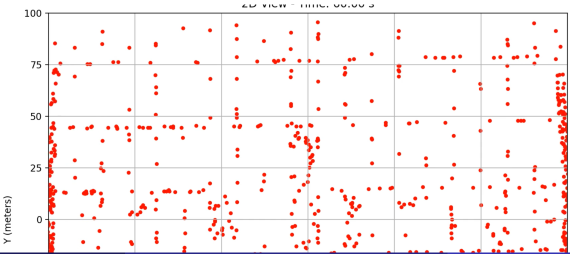
- **Input Processing:** The script reads `name2.jpg`, detects black pixels, and converts them into (x, y, z) target coordinates.
- **Assignment:** Each drone is assigned one target point.
- **Challenge:** The drones start randomly on the ground. They must rise and converge without hitting each other.

8b. Task 1: Failure Analysis

The Problem: Initial attempts to process the handwritten name failed to produce a clear formation.

Root Cause:

- **Low Contrast:** The ink-to-background contrast in the original image was insufficient for proper thresholding.
- **Background Separation:** Simple binary thresholding could not distinguish between the handwriting and paper texture/shadows.
- **Result:** The algorithm extracted incorrect or incomplete pixel coordinates, preventing proper formation.



9. Task 2: Dynamic Morphing

Objective: Transition the swarm from "Name Formation" to "Greeting Formation" (Happy New Year!).

- **The Morphing Process:** At $t = 0$, the target array **T** switches instantly from Set A (Name) to Set B (Greeting).
- **Physics Response:** The Attraction Force $\vec{F}_{att}$ creates a new vector for every drone pointing to the new location.
- **Collision Management:** As 1,000 drones cross paths in the center, the Repulsion Force $\vec{F}_{rep}$ ensures they "slide" past each other rather than crashing.
- **Note:** This task could not fail as the drones were already in a stable formation and had clear paths to their new targets.

10. Task 3: Video Target Tracking

Objective: The swarm must track a moving object extracted from a video file.

- **Computer Vision Pipeline:** We process ‘video.mp4‘ frame-by-frame using OpenCV. The centroid of the moving object becomes the swarm's focus point.
- **Time-Varying Targets:** Unlike Task 1 (Static), here $\vec{r}_{target}(t)$ changes every 0.03 seconds.
- **Success Case:** When tested on the moving video where the object moved slower than $V_{max} = 5.0$ m/s, the drones successfully tracked and converged to the moving target, maintaining formation throughout.
- **Failure Test Validation:** When the moving object's speed was increased to 20 m/s (exceeding $V_{max} = 5.0$), the drones could not keep up. The swarm correctly exhibited "Phase Lag" and broke formation, proving the inertia model and velocity limits are realistic.

11. Engineering Challenges: The Crash Fix

The Problem: Initial runs showed **486 collisions** at Step 0. **Why?** Drones were spawned in a $10m \times 10m$ box (High Density).
The Solution (Spatial Initialization): Instead of slowing the drones down (which makes the show boring), I increased the initialization area to $150m \times 150m$. This reduced initial density, allowing for a crash-free start at high speed.

Parameter	Initial (Crash)	Final (Stable)
Start Area	$10m \times 10m$	**150m** $\times$ **150m**
Start Density	High (Clumped)	**Low (Distributed)**
Max Velocity	5.0 m/s	**5.0 m/s** (High Speed Kept)
Collisions	**486** (Step 0)	**0** (All Steps)

12. Optimization: Parameter Tuning

In addition to spatial distribution, I rigorously tuned the PID and Physics coefficients to ensure stability in "Safe Mode".

Key Parameter Adjustments

1. **Gentle Pull (K_P):** $2.0 \rightarrow 1.0$
   Reduced attraction gain to prevent drones from "rushing" too aggressively toward targets.
2. **Heavy Braking (K_D):** $1.5 \rightarrow 4.0$
   Significantly increased damping. This acts like "air brakes," stopping oscillations instantly when they reach the goal.
3. **Strong Shield (K_{REP}):** $10.0 \rightarrow 200.0$
   Massive increase in repulsion strength. This ensures that even at high speeds, the "invisible shield" is strong enough to deflect neighbors.
4. **Early Warning (R_{SAFE}):** $0.8 \rightarrow 1.2$
   Increased the detection radius. Drones now start reacting to neighbors sooner, allowing smoother avoidance curves.

13. System Limitations

While functional, the simulation has specific constraints:

- **Computational Complexity ($O(N^2)$):** Collision detection requires checking every drone against every other drone. For $N = 1000$, this is $1,000,000$ checks per step. This prevents scaling to 10,000+ drones without spatial partitioning (e.g., Quadtrees).
- **Point-Mass Approximation:** The physics model ignores rotational dynamics (torque), rotor aerodynamics, and battery weight distribution. It treats drones as floating points with mass.
- **Idealized Environment:** The simulation assumes perfect sensor accuracy and zero communication delay, which is unattainable in physical hardware.

14. AI Usage Statement

In compliance with project guidelines, I declare the following use of AI tools:

- **Syntax Support:** AI was used to generate LaTeX templates and Matplotlib 3D configuration snippets.
- **Code Optimization:** AI assisted in vectorizing the pairwise distance calculation (‘numpy.linalg.norm‘) to improve performance.
- **Logic Verification:** All core algorithms (IVP formulation, RK4 Solver, Force Laws) were manually implemented and tuned by me.

Project Successful

The simulation demonstrates that large-scale autonomous swarms can be controlled using simple local interaction rules.
By optimizing the **initial spatial distribution** and tuning the **physics coefficients**, we achieved a collision-free system that maintains high-performance flight characteristics.

Thank You.