

# Numerical Programming Final Project (Task 3)

## Dynamic Tracking and Shape Preservation

Student: **(Andria Gvaramia)** Course: Numerical Programming KIU

January 18, 2026

### Task statement

**Input:** drone swarm at the New Year greeting (Task 2 final formation) and a video of choice.

**Goal:** move the swarm from the greeting to a moving object in the video and dynamically repeat the object's motion with shape preservation.

**Output:** trajectory of each drone and visualization whose input is the trajectory.

## 1 Overview

Task 3 is implemented in `task3/dynamic_tracking.py`. The key idea is:

- Extract the moving object's silhouette from each video frame (green-screen segmentation).
- Compute the silhouette **boundary/contour**.
- Sample  $N$  points on the contour at **equal arc-length spacing**.
- Maintain a **stable correspondence** so each drone stays attached to a particular contour point as the object moves.
- Visualize drones over the video (or on a blank background) as a GIF.

## 2 Inputs

### 2.1 Start formation

The start formation is the greeting formation (Task 2 targets):

```
task2/outputs/target_points.csv
```

The number of drones is  $N = \text{len}(\text{start})$  (default: 150 in our run).

### 2.2 Video

We use:

```
task3/video.mp4
```

The video contains a dark silhouette on a bright green background, which makes segmentation robust.

## 3 Mathematical model

### 3.1 State

We work in 2D pixel coordinates. Drone  $i$  has position:

$$x_i(t) \in \mathbb{R}^2, \quad i = 1, \dots, N.$$

### 3.2 Contour targets (shape preservation)

For each frame  $k$  we compute a closed contour curve  $C_k$  of the moving object. We define  $N$  target points by equal arc-length sampling:

$$T_i^{(k)} = C_k(s_i), \quad s_i = \frac{i}{N} L(C_k), \quad i = 0, \dots, N - 1,$$

where  $L(C_k)$  is the contour perimeter length. This produces a point set that lies **only on the boundary** (required condition).

### 3.3 Stable “attachment” across frames

Raw contour sampling has an arbitrary start index and orientation. To keep drone  $i$  attached to a consistent boundary point, we align the sequence by choosing a stable start and by minimizing a cyclic shift mismatch:

$$\min_{\text{shift } s} \sum_{i=0}^{N-1} \left\| T_i^{(k-1)} - T_{(i+s) \bmod N}^{(k)} \right\|^2,$$

also checking the reversed order (to handle orientation flips).

## 4 Numerical methods

### 4.1 Segmentation

We segment the silhouette using HSV thresholding for the green background:

- `mask_green = inRange(HSV, green_low, green_high)`
- `silhouette = NOT(mask_green)`
- Morphological open/close removes small artifacts.

We also ignore a small bottom strip (`--ignore-bottom-fraction`) to avoid spurious contours.

### 4.2 High-detail contour extraction

To increase boundary precision, we extract contours on an upscaled mask:

```
--contour-upscale 3.0 --contour-smooth 9
```

Upscaling yields subpixel-level detail once scaled back, and smoothing removes jagged contour noise.

### 4.3 Time discretization

Frames define a natural time step:

$$\Delta t = \frac{\text{video\_step}}{\text{fps}}.$$

Using `--video-step 1` uses every frame and maximizes temporal accuracy.

## 5 Drone motion controller

Two controllers are supported:

- **Direct (perfect tracking):**  $x_i^{(k)} = T_i^{(k)}$ . This produces **zero tracking error** and is ideal for visualization quality.
- **Dynamics (2nd-order):** a damped point-mass model integrated with RK4 to follow moving targets, which introduces lag.

The best result used direct tracking:

```
--controller direct
```

## 6 Reproducibility (commands)

All commands run from the project root.

### 6.1 Video + drones (best result)

```
python3 task3/dynamic_tracking.py \
--tracking-mode contour \
--controller direct \
--video-step 1 \
--contour-upscale 3.0 --contour-smooth 9 \
--hide-targets \
--save-gif --gif-fps 30 \
--output-gif task3/outputs/task3_contour_direct_hi_with_video.gif
```

### 6.2 Drones-only GIF (blank background)

```
python3 task3/dynamic_tracking.py \
--tracking-mode contour \
--controller direct \
--video-step 1 \
--contour-upscale 3.0 --contour-smooth 9 \
--no-bg --drone-color blue \
--save-gif --gif-fps 30 \
--output-gif task3/outputs/task3_contour_direct_hi_drones_only_blue.gif
```

## 7 Outputs

Generated files (reproducible):

- task3/outputs/task3\_contour\_direct\_hi\_with\_video.gif
- task3/outputs/task3\_contour\_direct\_hi\_drones\_only.gif
- task3/outputs/task3\_contour\_direct\_hi\_drones\_only\_blue.gif

## 8 Validation and limitations

### 8.1 Validation

The primary validation criterion is the constraint:

$$x_i^{(k)} \in \partial\Omega_k$$

where  $\partial\Omega_k$  is the object boundary at frame  $k$ . In `direct` mode, this holds by construction because drones are set to the contour samples.

### 8.2 Limitations

- Contour tracking assumes a clean silhouette; complex backgrounds would require a stronger segmentation model.
- Using dynamics (instead of direct tracking) introduces lag and can violate the strict boundary constraint.
- With fixed  $N$ , the maximum geometric detail is limited by the number of sampled points on the contour.

## 9 AI usage disclosure

This project was developed with AI assistance (ChatGPT) used for:

- explaining tracking / contour correspondence ideas,
- proposing numerical pipeline improvements and parameter choices,
- helping implement and debug Python code and visualization.