SwarmGPT: Combining Large Language Models With Safe Motion Planning for Drone Swarm Choreography

Martin Schuck, *Graduate Student Member, IEEE*, Dinushka Orrin Dahanaggamaarachchi, Ben Sprenger, Vedant Vyas, Siqi Zhou, *Member, IEEE*, and Angela P. Schoellig, *Member, IEEE*

***Abstract*—Drone swarm performances—synchronized, expressive aerial displays set to music—have emerged as a captivating application of modern robotics. Yet designing smooth, safe choreographies remains a complex task requiring expert knowledge. We present SwarmGPT, a language-based choreographer that leverages the reasoning power of large language models (LLMs) to streamline drone performance design. The LLM is augmented by a safety filter that ensures deployability by making minimal corrections when safety or feasibility constraints are violated. By decoupling high-level choreographic design from low-level motion planning, our system enables non-experts to iteratively refine choreographies using natural language without worrying about collisions or actuator limits. We validate our approach through simulations with swarms up to 200 drones and real-world experiments with up to 20 drones performing choreographies to diverse types of songs, demonstrating scalable, synchronized, and safe performances. Beyond entertainment, this work offers a blueprint for integrating foundation models into safety-critical swarm robotics applications.**

***Index Terms*—Swarm robotics, AI-enabled robotics, robot safety, aerial systems: applications, art and entertainment robotics.**

I. Introduction

Autonomous drones are increasingly featured in large-scale events—from concerts and sports championships to Olympic opening ceremonies [1], [2], [3]. Their agility and ability to move in tightly synchronized formations make them powerful tools for creating striking visual effects and conveying artistic expression alongside human performers.

Despite their potential, designing choreographed movements for drone swarms remains a complex challenge. Performances must not only achieve visual and artistic goals but also satisfy system constraints, such as smoothness, collision avoidance, feasibility, and downwash effects [4], [5], [6]. Balancing expressiveness with safety and hardware feasibility requires significant expert knowledge, making the design process labour-intensive and inaccessible to non-experts.

Recent advances in foundation models, driven by large-scale datasets, have established language and vision-language models as intuitive interfaces for human–robot interaction [7]. Their semantic understanding and reasoning capabilities have been increasingly applied in robotics—for example, in contextual navigation [8], [9], affordance-driven grasping and manipulation [10], [11], and guiding reward design or data collection for skill learning [12]. Language-based control of robot swarms holds similar promise, with potential applications extending beyond performances to domains such as disaster response or search-and-rescue—enabling non-experts to safely and effectively interact with complex systems.

However, integrating foundation models into real-world robotics remains challenging. Even small errors in perception or reasoning can propagate through control loops, leading to irreversible outcomes [7], [13]. These risks are amplified in swarm robotics, and particularly in drone swarms, where any physical contact between agents can cause immediate and catastrophic failure.

Building on prior work in drone performances [14], [15], we present SwarmGPT—a language-based choreographer that uses the reasoning capabilities of large language models (LLMs) to design synchronized, rhythmic drone swarm performances (Fig. 1). To ensure real-world deployability, we introduce a safety filter that preserves the intended choreography while enforcing collision-free and physically feasible trajectories. SwarmGPT offers an intuitive interface that enables non-experts to create and iteratively refine drone swarm behaviours using natural language—without needing to account for safety or low-level feasibility constraints.

The key contributions of this work are:

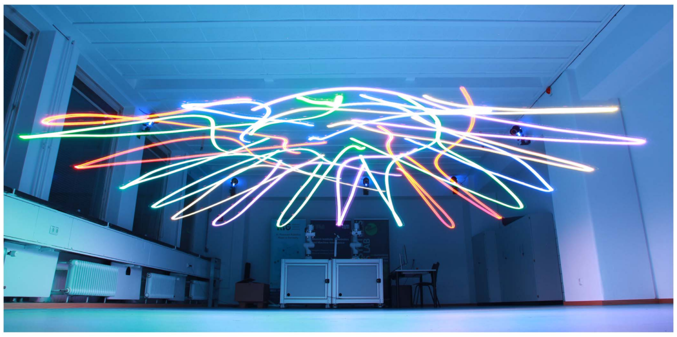
1) *SwarmGPT:* A language-based choreographer that combines the reasoning abilities of LLMs with optimization-based safety filters to generate drone swarm performances.
2) *Scalability:* Support for large swarms via motion primitives, self-correction mechanisms, and a distributed safety

Received 15 April 2025; accepted 25 September 2025. Date of publication 9 October 2025; date of current version 20 October 2025. This article was recommended for publication by Associate Editor J.R. Martinez-de Dios and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported in part by the Robotics Institute Germany under BMBF Grant 16ME0997K, in part by Marie Skłodowska-Curie European Union's Horizon Europe Research and Innovation Programme under Grant 101155035, and in part by the Humboldt Professorship for Robotics and Artificial Intelligence. *(Corresponding author: Martin Schuck.)*

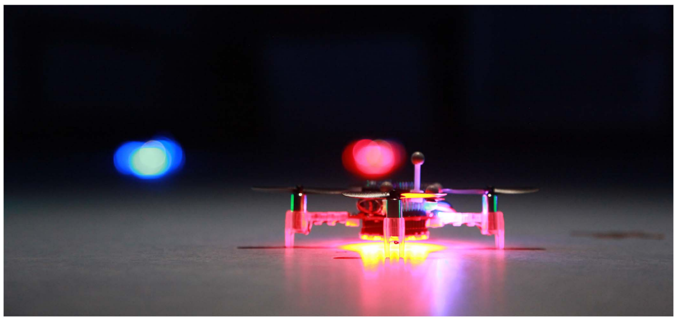
The authors are with the Learning Systems and Robotics Lab, Technical University of Munich, 80333 Munich, Germany, and also with the Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, 80333 Munich, Germany (e-mail: martin.schuck@tum.de; dinushka.dahanaggamaarachchi@tum.de; ben.sprenger@tum.de; vedant.vyas@tum.de; siqi.zhou@tum.de; angela.schoellig@tum.de).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3619745, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3619745



(a) Long-exposure image of 12 drones transitioning from a helix to a spiral motion at a musical beat.



(b) A close-up view of one of the performing drones.

Fig. 1. SwarmGPT generates drone swarm choreographies from natural language, synchronized with music. Demonstration videos are available at https://tiny.cc/swarmgpt and in the supplementary materials (S1–S7). The project website is https://tiny.cc/swarmgpt_project.

filter optimizer that preserves choreographic intent while ensuring safety and feasibility.

3) *Validation:* Extensive simulations (up to 200 drones) and real-world performances (up to 20 drones) across multiple musical genres, demonstrating the music interpretation capabilities of the LLM, the safe execution, and the effectiveness of language reprompting.

II. Related Work

A. Robot Choreography

Robot choreography design to music has emerged as a compelling form of robotic performance, explored across various platforms including humanoids [16], [17], [18], [19], quadrupeds [16], [17], robotic arms [20], and drone swarms [21], [22], [23]. Designing expressive and synchronized motions typically involves extensive manual tuning and domain expertise to ensure feasibility, safety, and alignment with musical or visual intent, making the process labour-intensive and non-intuitive.

In aerial swarm choreography, prior work has primarily relied on manually constructed sequences of motion primitives with hand-tuned parameters [23], [24]. These approaches require expert involvement to align choreography with musical timing and expression.

In contrast, we propose a language-based choreographer that generates synchronized drone swarm performances from natural language descriptions in minutes. Our system lowers the barrier to creating complex, deployable swarm behaviours by providing an intuitive interface for non-experts.

B. LLMs for Robotics

Recent work has explored the use of LLMs for robotic decision-making [7], with language guiding movement in tasks such as visual-language navigation and trajectory refinement [25], [26]. Beyond navigation, LLMs have been applied to joint navigation and manipulation in unstructured environments [27], as well as high-level task planning—whether through code generation [28], translating instructions into action sequences [29], or selecting action templates to control robots [9], [30].

Building on these directions, we apply LLMs to drone swarm coordination, focusing on language-driven choreography and its safe real-world deployment. A central challenge in robotics is grounding language in the robot's embodiment and capabilities. Prior work has approached grounding by predicting platform-specific affordances [11], using self-correcting inner monologues based on feedback [31], or fine-tuning on physical interactions to improve physical reasoning [32].

In our approach, grounding is achieved by converting high-level LLM-generated actions into deployable choreographies using a low-level, model-based safety filter that enforces collision avoidance and physical feasibility.

C. Safe Robot Swarm Decision-Making

Deploying drone swarm performances in the real world requires planned trajectories to be collision-free and dynamically feasible, respecting actuation and safety constraints. Recent work has increasingly combined learning-based models with model-based control to ensure safety and performance in real-world deployments [33]. While these methods have shown strong results in single-agent systems, their application to multi-agent swarms is still emerging.

For drone swarms, optimization-based motion planning is a natural choice to enforce constraints explicitly. Coordination strategies are typically classified as centralized, where a global optimization problem resolves conflicts among agents [24], [34], [35], or distributed, where each agent solves a local problem based on partial knowledge [36], [37], [38]. While both can handle safety and feasibility, distributed approaches scale better with swarm size [39]. Note that a distributed solution is often still run on a central computer, yielding significant scaling benefits from parallelization compared to centralized methods. However, existing distributed methods [36], [37], [38] are generally limited to point-to-point transitions without fixed timing constraints.

In this work, we propose a distributed safety filter that extends existing approaches by considering multi-target trajectories with time synchronization, ensuring that drones reach specific positions at fixed times. This formulation enables efficient, real-time enforcement of safety and feasibility during language-driven drone swarm performances.

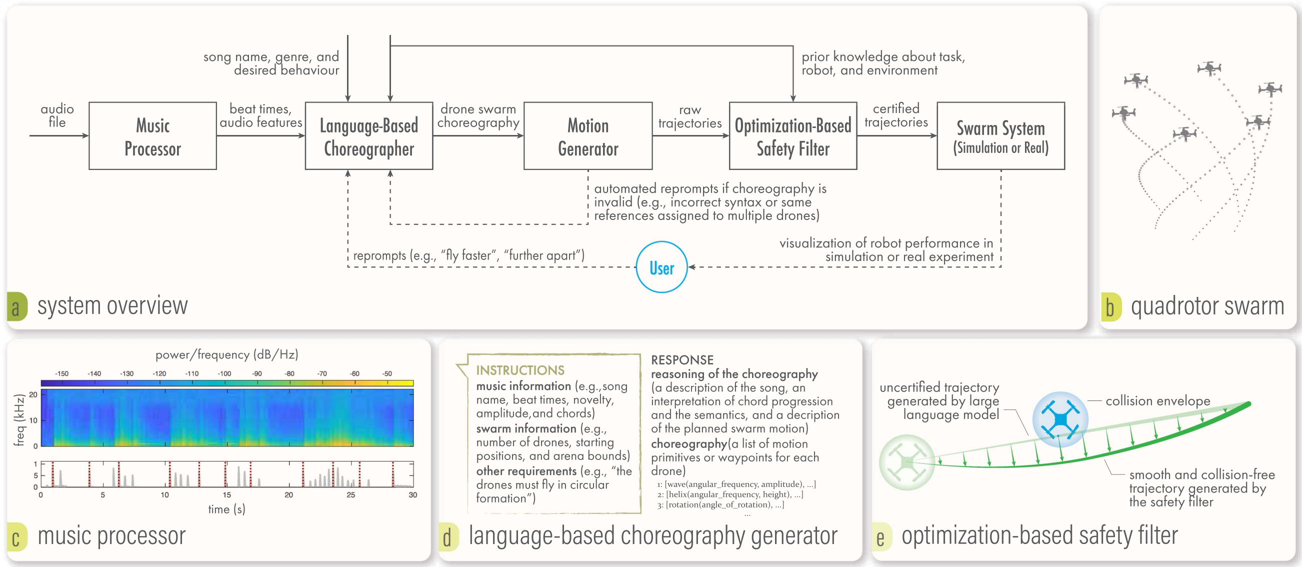

song name, genre, and desired behaviour
prior knowledge about task, robot, and environment
audio file
Music Processor
beat times, audio features
Language-Based Choreographer
drone swarm choreography
Motion Generator
raw trajectories
Optimization-Based Safety Filter
certified trajectories
Swarm System (Simulation or Real)
automated reprompts if choreography is invalid (e.g., incorrect syntax or same references assigned to multiple drones)
reprompts (e.g., "fly faster", "further apart")
User
visualization of robot performance in simulation or real experiment
a system overview
b quadrotor swarm
power/frequency (dB/Hz)
freq (kHz)
time (s)
c music processor
INSTRUCTIONS
music information (e.g.,song name, beat times, novelty, amplitude,and chords)
swarm information (e.g., number of drones, starting positions, and arena bounds)
other requirements (e.g., "the drones must fly in circular formation")
RESPONSE
reasoning of the choreography (a description of the song, an interpretation of chord progression and the semantics, and a decription of the planned swarm motion)
choreography(a list of motion primitives or waypoints for each drone)
1: [wave(angular_frequency, amplitude), ...]
2: [helix(angular_frequency, height), ...]
3: [rotation(angle_of_rotation), ...]
...
d language-based choreography generator
uncertified trajectory generated by large language model
collision envelope
smooth and collision-free trajectory generated by the safety filter
e optimization-based safety filter

Fig. 2. Overview of the SwarmGPT system. **(a)** A high-level LLM designs unique choreographies, while a low-level optimization-based safety filter ensures feasible, collision-free deployment. **(b)** The drone performers are quadrotor vehicles. **(c)** A music processor extracts beat times and audio features from the song. **(d)** The LLM generates choreographies from the audio features and music beat times, producing raw drone trajectories. **(e)** The safety filter outputs safe, feasible trajectories for simulation or hardware deployment, enabling non-experts to iteratively refine via natural language.

III. The SwarmGPT Framework

In this section, we present our proposed SwarmGPT framework for generating safe swarm performances from language. An overview of the proposed framework is shown in Fig. 2(a), which consists of the following core modules: *(i)* a music processor, *(ii)* a language-based choreographer, *(iii)* a motion generator, and *(iv)* an optimization-based safety filter. Details of the individual modules are presented in the respective subsections below. The choreography is executed by a swarm of quadrotors, see Fig. 2(b).

A. Music Processor

In the music processor, we analyze the waveform of the raw audio signal to determine a set of beat times to which the performances are synchronized. The beat times are computed based on the spectral-based novelty function of the waveform, following the approach outlined in [40]. Intuitively, the novelty function exemplified in Fig. 2(c) characterizes the distance between successive signal points of the audio, and gives us a measure of how much tones stand out. We define beat times

$$\mathbb{B} = \{t_0, t_1, \ldots, t_T\} \tag{1}$$

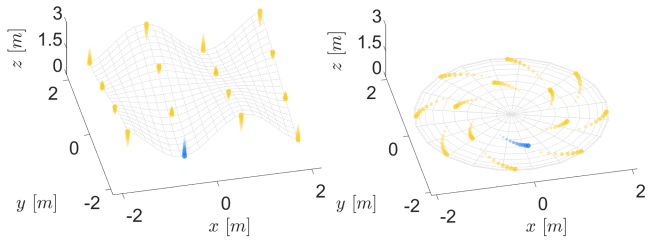
as the peaks of the novelty function, where $T \in \mathbb{N}$ corresponds to the total number of beats. To extract the beats, we use standard peak detection tools from [41]. These extracted timestamps serve as discrete synchronization points for the drone choreography. Specifically, the LLM is permitted to specify motions only at these beat times, ensuring that drone actions are temporally aligned with the musical structure. The beat times are further annotated with their novelty, decibels relative to full scale (i.e., their loudness) and their respective chords. To estimate chords from the audio track, we make use of the hidden Markov model-based approach described in [40]. Including this information enables planning performances that are linked more closely to the original audio track. This annotation enables the planning process to incorporate both rhythmic and expressive elements of the audio, resulting in performances that are closely linked to the audio track.

B. Language-Based Choreographer

Given the extracted beat times and audio features, each performance is designed by a LLM (specifically GPT-4 [42] in our case) that defines reference positions for the swarm. The initial prompt introduces the task, provides the music information, and specifies output actions as shown in Fig. 2(d). Grounding LLMs in physical embodiments is challenging and can be greatly aided by selecting suitable action modalities [43]. Therefore, we investigate the performance of our framework with two distinct action modalities.

1) Waypoints: We let the LLM generate a series of target coordinates for the beat times for each drone in the swarm. While this allows for a large variety of possible configurations, the LLM needs to reason over multiple coordinates to interpret the current swarm configuration. As a consequence, generating coherent geometrical swarm shapes becomes difficult with growing swarm size.

2) Motion Primitives: Alternatively, we let the LLM compose performances based on a library of 12 parameterized motion primitives. We augment the initial prompt with a list of available primitives such as `rotate`, `helix`, `spiral` and `wave` with usage examples. A selection of these primitives can be seen in Fig. 3. The LLM selects primitives, their parameters, and respective motion start beats $t_i \in \mathbb{B}$ and end beats $t_f \in \mathbb{B}$.



(a) Wave Primitive (b) Spiral Primitive

Fig. 3. Visualizations of two example motion primitives in the form of (4). The largest solid markers show drones' current positions with trailing tails indicating motion over time; a single blue trajectory highlights one drone's path. **(a)** Wave primitive mimicking a surface wave on a grid-configured swarm, with frequency and amplitude parameters adjustable by the LLM [14]. **(b)** Spiral primitive rotating a circular swarm about the z-axis while gradually expanding its radius, configurable by the LLM.

We present a detailed discussion of the motion primitives in the following subsection.

In case of planning failures such as omitting beat times, missing waypoints for drones, using non-existent primitives or when parameters selected by the LLM result in trajectories violating physical limits, the LLM receives a failure report and triggers a corresponding self-correction. In addition to self-corrections, after visualizing the performance in either simulation or experiments, users can modify the swarm choreography directly using natural language via additional reprompting. A full example of the prompts is included in supplementary material S8.

C. Motion Generator

Generating motions for waypoint actions is straightforward by specifying all desired coordinates and relying on the safety filter to generate physically feasible, collision-free trajectories between points. Our motion generators for primitives for a swarm of N drones are defined as tuples [14]:

$$\mathcal{MP}_m = \left(t_{m,i}, t_{m,f}, \{\mathbf{c}_{m,n}\}_{n=1}^{N}, \mathcal{T}_m(\mathbf{c}, t)\right), \tag{2}$$

where $m \in \mathbb{N}_{[0,M]}$ is the motion primitive index, $t_{m,i} \in \mathbb{B}$ and $t_{m,f} \in \mathbb{B}$ are the start time and the end time of the motion primitive, respectively, $\mathbf{c}_{m,n} \in \mathbb{R}^3$ is a unique configuration vector associated with a particular drone n for the motion primitive m, and $\mathcal{T}_m(\mathbf{c}, t) : \mathbb{R}^3 \times [t_{m,i}, t_{m,f}] \to \mathbb{R}^3$ is the position reference trajectory generator. The position reference trajectory for each drone is

$$\mathbf{r}_{m,n}(t) = \mathcal{T}_m(\mathbf{c}_{m,n}, t),\ \forall t \in [t_{m,i}, t_{m,f}]. \tag{3}$$

There are different ways to define trajectory generators. One general formulation has the following form (with the trajectory index m dropped for clarity):

$$\mathcal{T}(\mathbf{c}_n, t) = \mathcal{M}(\mathbf{c}_n) + \mathcal{T}_{\text{per}}(\mathbf{c}_n, t) + \mathcal{T}_{\text{poly}}(\mathbf{c}_n, t), \tag{4}$$

where $\mathcal{M}(\mathbf{c}_n)$ is constant with respect to time, and $\mathcal{T}_{\text{per}}(\mathbf{c}_n, t)$ and $\mathcal{T}_{\text{poly}}(\mathbf{c}_n, t)$ are periodic and polynomial terms defined as follows:

$$\mathcal{T}_{\text{per}}(\mathbf{c}_n, t) = \sum_{p=1}^{P} \left(\mathcal{A}_p(\mathbf{c}_n)\sin(\omega_p t) + \mathcal{B}_p(\mathbf{c}_n)\cos(\omega_p t)\right), \tag{5}$$

$$\mathcal{T}_{\text{poly}}(\mathbf{c}_n, t) = \sum_{q=1}^{Q} \mathcal{C}_q(\mathbf{c}_n)\, t^q. \tag{6}$$

In the general formulation (4), the terms $\mathcal{M}(\mathbf{c}_n)$, $\mathcal{A}_p(\mathbf{c}_n)$, $\mathcal{B}_p(\mathbf{c}_n)$, $\mathcal{C}_q(\mathbf{c}_n)$, ω_p, P, and Q are parameters defining the motion primitives. The parameters $\mathcal{A}_p(\mathbf{c}_n)$, $\mathcal{B}_p(\mathbf{c}_n)$, $\mathcal{C}_q(\mathbf{c}_n)$ are drone-dependent amplitude functions for each term. Motion components for some or all drones can be omitted by setting the corresponding amplitudes to zero. Note that, when $\mathcal{C}_q(\mathbf{c}_n)$ is zero for all q and n, we recover the motion primitive generator defined in [14]. As shown in [14], with the first two terms alone, diverse periodic behaviours such as rigid body rotation and wave patterns can be achieved for rhythmic drone performances. With the addition of $\mathcal{T}_{\text{poly}}(\mathbf{c}_n, t)$, we can further incorporate non-periodic components into the choreography. Two example motion primitives are shown in Fig. 3.

Since the equations are difficult to interpret semantically, we generate a library of motion primitives with physically interpretable parameters that the LLM can use to capture the audio features or incorporate human feedback. For example, we generate the `rotate` primitive by setting the polynomial amplitudes to zero and keeping the height fixed. The LLM can specify the desired angular displacement, which we then convert to ω_p. Importantly, users do not specify these equations directly but instead issue high-level prompts such as "move faster," which are translated into primitives and parameter changes by the LLM. Our motion primitives are similar in spirit to action template design in language-based task planners [9].

D. Optimization-Based Safety Filter

The language-based choreographer outputs the performance as a set of position reference trajectories $\mathbf{r}_n(t)$ for each drone n parametrized either by waypoints or through composition of motion primitives. Waypoint references offer no safety guarantees since they do not restrict the LLM's output to collision-free and feasible swarm configurations in any way. Similarly, while individual motion primitives can be designed such that the trajectories $\mathcal{T}_m(\mathbf{c}_{m,n}, t)$ are smooth and collision-free for the intervals $t \in [t_{m,i}, t_{m,f}]$, this does not guarantee safe performances. In particular, in between two consecutive motion primitives, the swarm needs to transition dynamically based on their current positions, and these transitions can result in collisions and non-smooth behaviours.

To certify that trajectories are feasible and safe for deploying to the real-world system, we employ an underlying safety filter that optimizes swarm trajectories for smoothness while adhering to safety constraints. Intuitively, the safety filter allows the drones to follow the designed trajectories where possible and deviate from their trajectory where the proximity to another swarm member necessitates collision avoidance as depicted in

Fig. 2(e). The incorporation of the safety filter allows us to decouple choreography design and the satisfaction of safety constraints for deployment.

We formulate the safety filter as a distributed multi-agent trajectory optimization problem that is solved in a receding horizon manner. As compared to centralized methods [14], [24], distributed methods can be implemented efficiently in real-time and scale up to larger swarms [36], [38]. However, prior works have focused on point-to-point transitions of swarms and do not include timing constraints. Since we require scalable, synchronized motions to multiple swarm configurations in succession, we formulate a distributed safety filter that can handle multiple reference points and respects timing constraints.

We discretize the reference trajectories at fixed time intervals δt based on a pre-specified control frequency and, at each sampling time $t_k = t_0 + k\delta t$, each drone n solves an independent optimization problem over a finite prediction horizon $[t_k, t_{k+K}]$:

$$\mathbf{r}^*_{n,k:k+K-1|k} = \pi_{\text{safe}}\left(\mathbf{x}_{n,k}, \mathbf{r}_n, \{\mathbf{p}^*_{j,k:k+K|k-1}\}_{j\in\mathbb{J}}\right), \qquad (7)$$

where the notation "$i|k$" means the variable at time step i predicted at time step k, ":" abbreviates consecutive time steps, $K \in \mathbb{N}$ is the prediction horizon, $\mathbf{x}_{n,k} = [\mathbf{p}^T_{n,k}, \mathbf{v}^T_{n,k}]^T$ is a vector containing the current position and velocity of the drone, $\mathbb{J}$ is the set of drone indices with which drone i needs to avoid collision, and $\mathbf{p}^*_{j,k:k+K|k-1} \in \mathbb{R}^3$ is the predicted position of neighbouring drone j in the swarm based on the optimization from the previous time step. The problem in (7) is solved for every time step $t_k \in \{t_0, t_0+\delta t, t_0+2\delta t, \ldots, t_T\}$ for each drone n, and the certified position reference for drone n at t_k is given by the first element of the optimized variable obtained at each time step: $\mathbf{r}^*_n(t_k) = \mathbf{r}^*_{n,k|k}$.

The safety filter optimization problem π_{safe} solved for individual drones at time step k is defined as follows (with the drone index n dropped for brevity):

$$\min_{\mathbf{u}_{\cdot|k}} \sum_{i\in\mathbb{K}} \alpha \left\|\mathbf{p}^{(D)}_{i+1|k}\right\|^2 + \beta \left\|\mathbf{u}^{(D)}_{i|k}\right\|^2 \qquad (8a)$$

$$\text{s.t.} \quad \mathbf{x}_{k|k} = \mathbf{x}_k \qquad (8b)$$

$$\mathbf{u}^{(d)}_{k|k} = \mathbf{u}^{(d)}_{k|k-1},\ \forall d \in \mathbb{D} \qquad (8c)$$

$$\mathbf{x}_{i+1|k} = \mathbf{A}\mathbf{x}_{i|k} + \mathbf{B}\mathbf{u}_{i|k},\ \forall i \in \mathbb{K} \qquad (8d)$$

$$\mathbf{p}^{(d)}_{s|k} = \mathbf{r}^{(d)}(t_s),\ \forall s \in \mathbb{S}_k,\ \forall d \in \mathbb{D} \qquad (8e)$$

$$\underline{\mathbf{p}} \preceq \mathbf{p}_{i|k} \preceq \overline{\mathbf{p}},\ \forall i \in \mathbb{K} \qquad (8f)$$

$$\left\|\mathbf{p}^{(1)}_{i|k}\right\|^2 \leq \overline{v}^2,\ \forall i \in \mathbb{K} \qquad (8g)$$

$$\underline{f}^2 \leq \left\|\mathbf{p}^{(2)}_{i|k} + \mathbf{g}\right\|^2 \leq \overline{f}^2,\ \forall i \in \mathbb{K} \qquad (8h)$$

$$\left\|\mathbf{p}_{i|k} - \mathbf{p}^*_{j,i|k-1}\right\|^2_{\Theta_j^{-1}} \geq 1,\ \forall i \in \mathbb{K}, \forall j \in \mathbb{J}, \qquad (8i)$$

where $\mathbb{K} = \{k, k+1, \ldots, k+K-1\}$ is the set of time indices over the horizon, $\alpha, \beta \in \mathbb{R}_{\geq 0}$ are cost function parameters, $\mathbf{u}_{\cdot|k}$ is the optimization variable parameterized as Bernstein polynomials [38], $\mathbf{u}_{k:k+K-1|k}$ are values of $\mathbf{u}_{\cdot|k}$ sampled at discrete time steps, the superscript $(\cdot)^{(d)}$ denotes the d-th order derivative with respect to time, $\mathbb{D} = \{0, 1, \ldots, D\}$ with D being the desired level of continuity enforced, $\mathbb{S}_k$ is a set of time indices for sampling the reference trajectory, $\mathbf{x}_{i|k} = [\mathbf{p}^T_{i|k}, \mathbf{v}^T_{i|k}]^T$ with $\mathbf{v}_{i|k} = \mathbf{p}^{(1)}_{i|k}$, $(\underline{\mathbf{p}}, \overline{\mathbf{p}})$ are the position bounds characterizing the size of the flying arena, $\overline{v}$ is the maximum allowable speed, $(\underline{f}, \overline{f})$ are the minimum and maximum mass-normalized collective thrusts generated by the motors, and Θ_j is a diagonal matrix with the diagonal elements characterizing the dimensions of ellipsoidal envelopes for collision and downwash avoidance [6]. The matrices, $\mathbf{A}$ and $\mathbf{B}$, are identified system matrices representing the closed-loop dynamics of the quadrotor system [39]. Intuitively, the safety filter optimization (8) encourages the certified trajectory $\mathbf{r}^*(t)$ to match the trajectory generated by the language-based choreographer $\mathbf{r}(t)$, while ensuring the trajectory is smooth, feasible, and collision-free for real-world deployment. Note that (8e) ensures that drone motions are time synchronized with the trajectory and is enforced for all waypoints within the current horizon.

The optimization problem in (8) is a non-convex, quadratically constrained quadratic program (QCQP), which can be computationally expensive to solve, especially for large swarms. In this work, similar to AMSwarm [38], we reparametrize the quadratic constraints in polar form and solve the distributed optimization problem using an alternating minimization scheme. This approach allows us to efficiently solve the problem in (8) in real-time, without relying on conservative approximations (e.g., linearizing the quadratic constraints as in [36], [37], [39]). As compared to [38], our method additionally accounts for the drones' tracking dynamics and enforces continuity in the reference trajectories, which enables more faithful realization of the intended choreography.

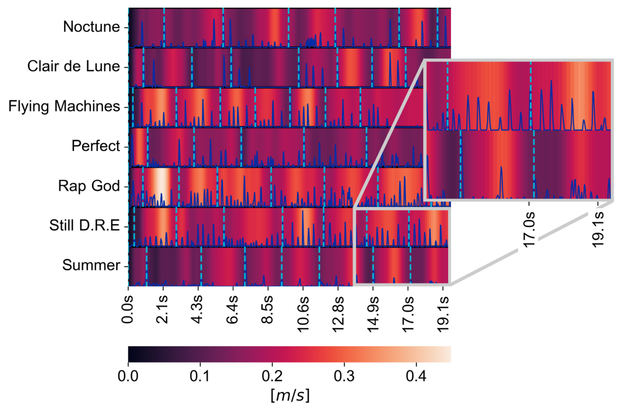
IV. Experimental Results

Our proposed approach is demonstrated using the Crazyflies 2.1 as shown in Fig. 1. Each Crazyflie is equipped with a controller that converts high-level position references to low-level motor commands. Simulated experiments are conducted with `crazyflow` [44], a drone swarm simulator modeled after the Crazyflies and successor to `gym-pybullet-drones` [45]. Our hardware setup for real deployment is further described in Sec. IV-D.

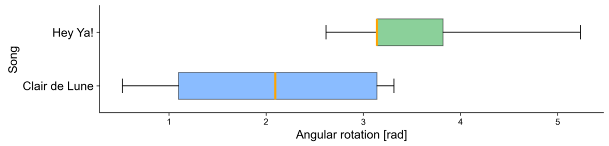
A. LLMs as Choreographers

In our first experiments, we show that SwarmGPT generates drone performances that are synchronized to the given songs and further incorporates human feedback to alter the behaviour of the swarm.

Performances Match the Music: To assess that SwarmGPT produces distinct and synchronized performances for different songs, we select seven songs across different genres and create 10 performances each using waypoint actions. Fig. 4(a) shows a plot of the average drone speed alongside the beat times and the novelty function derived from music analysis. The plot



(a) Swarm velocity heatmap over several different songs.



(b) Speed parameter distribution for a fast (green) and a slow (blue) song.

Fig. 4. Matching music to choreography characteristics. **(a)** Average speed of all drones in the swarm over 10 performances per song. The dark blue line depicts the songs' novelty function, and the dashed light-blue lines denote the beat times. The speed profile is distinct for each song and correlates with the beat times. **(b)** Average value of the spiral speed parameter over 25 performances for a slow and a fast song. Boxes denote the 10th and 90th quantiles, with the median in orange. The LLM uses faster speeds for the upbeat song, showing that the music analysis influences the design.

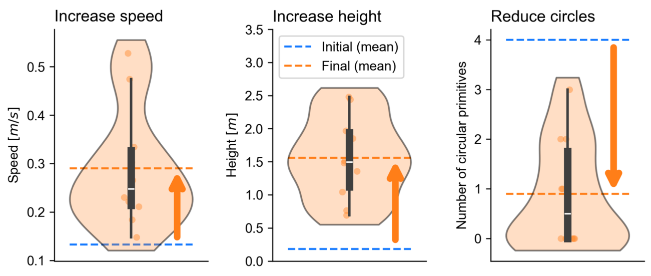


Fig. 5. Examples of LLM swarm behaviour modification. In the user study, 10 participants modified an initial performance (blue) with the goal to increase speed, increase height, or reduce circular motion primitives. The choreography statistics (orange) show participants consistently achieved their goal, with white being the median, the dashed lines the respective mean values, and the arrows indicate the desired change.

reveals that the swarm speed extrema (highlighted in red) are synchronized with the beat times (indicated by dashed light-blue lines), with a distinct pattern for each song. We also observe that performances for classic songs are slower on average than, e.g., the faster rap songs.

To further emphasize the connection between music and swarm motion, we design 25 performances using a motion primitive with an interpretable speed parameter for an upbeat song and a slow, classic song each. The distribution of the speed parameter displayed in Fig. 4(b) assumes significantly higher values for the lively song, showing that the music analysis is reflected in the design of the LLM.

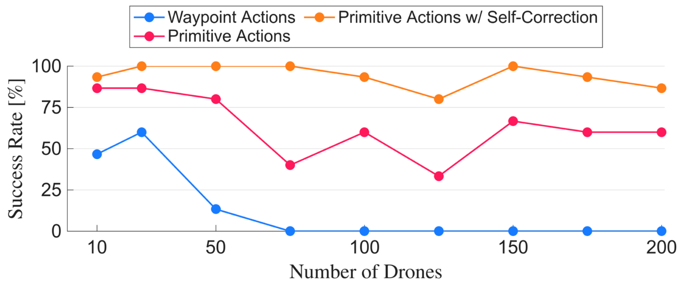


Fig. 6. A comparison of success rates for waypoint actions and motion primitives with and without self-correction. We limit self-correction reprompting to two times for each performance. The primitive-based approach leads to significantly higher success rates, with further improvements through self-correction. Mean response times remain at around 7.5 seconds despite self-corrections, enabling an interactive design process.

User Interaction Through Reprompting: The LLM can also be reprompted to generate revised versions of the current performance. We demonstrate this in a user study with 10 participants. Using three pre-designed performances, users are tasked to a) increase the speed, b) increase the height and c) reduce the number of circular motion primitives of the performance. The resulting trajectory statistics in Fig. 5 show that participants consistently achieve their goals. Participants required approximately two minutes to be satisfied with their task. Notably, users were using a wide variety of styles to prompt the choreographer. We thus conclude that SwarmGPT does not only capture the intent of users with its choreographies, but also demonstrates robustness to different prompt formulations.

B. *Scaling Language-Based Choreographies*

When scaling up the swarm size, the choice of action modalities—waypoints or motion primitives—becomes increasingly significant for successful planning and execution. Specifically, we compare the design success rates of the primitive-based approach with and without self-correction as well as that of the waypoint-based formulation for different swarm sizes. Performances that contain incorrect syntax after sanitizing responses or have the same reference simultaneously assigned to two or more drones are considered failures. Each method is evaluated three times on five different songs, which corresponds to a total of 15 performances for each drone swarm size case. As shown in Fig. 6, the waypoint-based approach yields approximately a 50% success rate for a swarm of 10 drones, but performance degrades to below 20% for a swarm of 50 drones and eventually drops to 0%. In contrast, the motion primitive method, which restricts the LLM's outputs to a set of semantically interpretable parameters, achieves success rates around 85% that decline initially and then remain around 65% even for large swarms of 200 drones. Including a self-correction mechanism that automatically reprompts the LLM in case of failures with additional information on the failure type further increases success rates to 85-100%. We attribute the rapid deterioration of success rates for waypoint actions to verbose outputs that are prone to syntax and geometric reasoning errors as the number of drones grows, whereas motion primitives remain easier to reason about. This

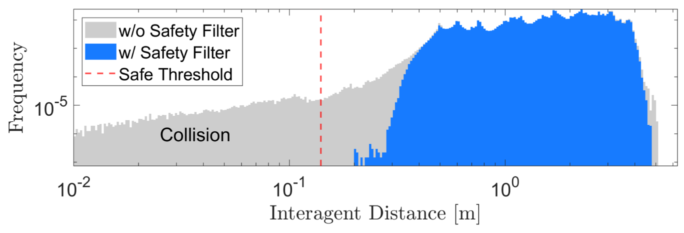


Fig. 7. A comparison of interagent distances with and without the safety filter applied. The red dashed line shows the minimum interagent distance to be considered collision-free. The safety filter effectively mitigates collisions that would otherwise occur in language-based choreography designs.

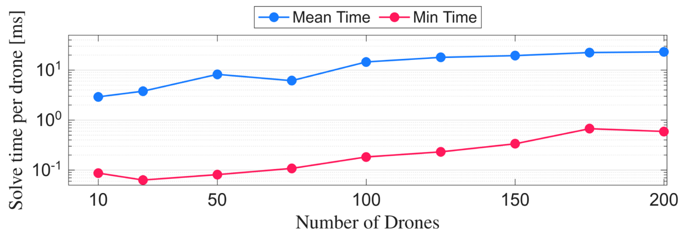


Fig. 8. Per-drone filter solve times for up to 200 drones in milliseconds. As we scale towards larger swarms, the optimization problem becomes more complex and thus per-drone solve times increase despite the distributed formulation. We achieve similar performance to other approaches despite the increased complexity of tracking multiple positions at specific timings.

highlights the importance of using structured responses when scaling towards larger swarms.

C. Guaranteeing Safety Through Filters

LLM-generated reference trajectories are inherently unsafe. While motion primitives may be designed for individual feasibility, transitions between them may still be unsafe. Figure 7 compares inter-agent distances with and without our safety filter. The results clearly show that the safety filter successfully prevents collisions by resolving unsafe configurations. Evaluations with waypoint actions show similar results.

As the swarm size increases, the safety filter's optimization becomes more complex. Our distributed formulation solves each drone's problem independently on a central computer, enabling efficient parallelization and avoiding the high cost of joint optimization. This allows fast solve times, supporting interactive choreography design. Note that "distributed" refers to the optimization structure—not to inter-drone communication.

Computation times of our safety filter are around the same magnitude as other approaches in the literature [36], [38]. Results of simulating 10 performances each on up to 200 drones are shown in Fig. 8. At 10 drones, the filter on average requires approximately 2.9 ms per drone to solve the problem. This number increases to 23.1 ms for 200 drones. Note that we are solving a more difficult problem compared to prior works, including multiple positions with timing constraints and an additional dynamics model. Our safety filter is implemented in JAX [46] for ease of integration and scalability and is recomputed at 8 Hz.

Feasible motion primitives reduce the need for safety filter intervention. Across 15 performances with 20 drones, filtered trajectories deviate on average by only 8.49 cm for primitives, compared to 23.02 cm for waypoint actions, allowing the swarm to more faithfully follow the LLM-designed choreography. This fidelity is especially important for larger swarms, where infeasible plans may prevent filter convergence.

D. Real-World Deployment

We demonstrate the capabilities of SwarmGPT by deploying performances for swarms of up to 20 drones on the Crazyflies in real-world experiments. Our setup uses a central base station with two antennas and the Crazyswarm platform [47] to broadcast 10 Hz position commands to drones, which are tracked via Vicon and onboard controllers. Poses are estimated using motion capture and sent from the base station to each drone. Although our filter runs in real time for the deployed swarm sizes, we pre-compute trajectories to eliminate delays and ensure convergence, with optional simulation previews. No inter-drone communication is required, and scaling to larger swarms is possible by adding base stations.

A long-exposure capture of one performance is included in Fig. 1, and videos of the drone performances can be found at http://tiny.cc/swarmgpt and in the supplementary materials (S3-S7). In the videos, we give a complete walkthrough of SwarmGPT and showcase designs based on various music genres and the capabilities of the system for modifying swarm behaviours through language instructions (see S2). The performances are accurately tracked by the drones' controllers, with an average tracking error of approximately 4.8 ± 2.77 cm.

V. Conclusion

We introduced SwarmGPT, a framework for generating drone swarm choreographies from natural language. By leveraging the reasoning capabilities of LLMs and an intuitive interface, SwarmGPT enables non-experts to design and refine performances to music. An optimization-based safety filter ensures feasibility by minimally adjusting unsafe trajectories. We validated our approach in simulation with up to 200 drones and in real-world experiments with up to 20 drones, demonstrating scalability across different swarm sizes and music styles. While motion primitives help with spatial reasoning and reduce planning failures, they limit flexibility in practice. Future work will explore how to balance structured outputs for scalability with the freedom needed to expand the design space. Overall, SwarmGPT offers a blueprint for safely integrating LLMs into complex robotic systems and opens the door to broader applications, such as non-expert swarm control in disaster response and search-and-rescue.

Supplementary Materials

Supplementary materials include (a) S1, an explainer video of our work, (b) S2, a walk-through video of SwarmGPT, (c) S3-S7, recordings of SwarmGPT performances, and (d) S8 example prompts used. Supplementary materials are accessible at https://tiny.cc/swarmgpt.

References

[1] E. Ackerman, "Flying LampshadeBots come alive in cirque du soleil performance: Cirque du soleil and eth Zurich turn quadrotors into magical floating lampshades," Sep. 2014. Accessed: Sep. 15, 2024. [Online]. Available: https://spectrum.ieee.org/lampshades-come-alive-eth-zurich-and-cirque-du-soleil

[2] The 2024 International Olympic Committee, "Spectacular intel drone light show helps bring Tokyo 2020 to life," Jul. 2021. Accessed: Sep. 15, 2024. [Online]. Available: https://olympics.com/ioc/news/spectacular-intel-drone-light-show-helps-bring-tokyo-2020-to-life-1

[3] P. O'Mahony, "The art of creative storytelling," May 2020. Accessed: Sep. 15, 2024. [Online]. Available: https://skymagic.show/the-art-of-creative-storytelling/

[4] A. Schöllig, M. Hehn, S. Lupashin, and R. D'Andrea, "Feasiblity of motion primitives for choreographed quadrocopter flight," in *Proc. Amer. Control Conf.*, 2011, pp. 3843–3849.

[5] A. Desai, E. A. Cappo, and N. Michael, "Dynamically feasible and safe shape transitions for teams of aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 5489–5494.

[6] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme, "Downwash-aware trajectory planning for large quadrotor teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 250–257.

[7] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.

[8] Q. Gu et al., "ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 5021–5028.

[9] I. Singh et al., "ProgPrompt: Generating situated robot task plans using large language models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 11523–11530.

[10] A. Rashid et al., "Language embedded radiance fields for zero-shot task-oriented grasping," in *Proc. Conf. Robot Learn.*, 2023, pp. 178–200.

[11] A. Brohan et al., "Do as i can, not as i say: Grounding language in robotic affordances," in *Proc. Conf. Robot Learn.*, 2022, pp. 287–318.

[12] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Proc. Conf. Robot Learn.*, 2023, pp. 3766–3777.

[13] R. Firoozi et al., "Foundation models in robotics: Applications, challenges, and the future," in *Proc. Int. J. Robot. Res.*, vol. 44, no. 5, 2025, pp. 701–739.

[14] X. Du, C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Fast and in sync: Periodic swarm patterns for quadrotors," in *Proc. Int. Conf. Robot. Automat.*, 2018, pp. 9143–9149.

[15] A. P. Schoellig, H. Siegel, F. Augugliaro, and R. D'Andrea, "So you think you can dance? Rhythmic flight performances with quadrocopters," in *Controls and Art: Inquiries at the Intersection of the Subjective and the Objective*, A. LaViers and M. Egerstedt, Eds. Berlin, Germany: Springer, 2014, pp. 73–105.

[16] Boston Dynamics, "Do you love me?," Dec. 2020. [Online] Available: https://www.youtube.com/watch?v=fn3KWM1kuAw

[17] E. Guizzo, "By leaps and bounds: An exclusive look at how Boston dynamics is redefining robot agility," *IEEE Spectr.*, vol. 56, no. 12, pp. 34–39, Dec. 2019.

[18] G. Xia, J. Tay, R. Dannenberg, and M. Veloso, "Autonomous robot dancing driven by beats and emotions of music," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, Richland, SC, USA, 2012, vol. 1, pp. 205–212.

[19] M. Boukheddimi et al., "Robot dance generation with music based trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 3069–3076.

[20] A. Rogel, R. Savery, N. Yang, and G. Weinberg, "RoboGroove: Creating fluid motion for dancing robotic arms," in *Proc. Int. Conf. Movement Comput.*, 2022, pp. 1–9.

[21] C. du Soleil, "SPARKED: Behind the technology," Sep. 2014, [Online] Available: https://youtu.be/7YqUocVcyrE?si=GRWwC_bbIisFsuMu

[22] A. Schoellig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 3355–3360.

[23] X. Du, C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Fast and in sync: Periodic swarm patterns for quadrotors," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 9143–9149.

[24] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Dance of the flying machines: Methods for designing and executing an aerial dance choreography," *IEEE Robot. Automat. Mag.*, vol. 20, no. 4, pp. 96–104, Dec. 2013.

[25] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 10608–10615.

[26] A. Bucker et al., "LATTE: Language trajectory transformer," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7287–7294.

[27] B. Zitkovich et al., "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proc. Conf. Robot Learn.*, 2023, vol. 229, pp. 2165–2183.

[28] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada, "SayTap: Language to quadrupedal locomotion," in *Proc. Conf. Robot Learn.*, 2023, vol. 229, pp. 3556–3570.

[29] D. Driess et al., "PaLM-e: An embodied multimodal language model," in *Proc. Int. Conf. Mach. Learn.*, vol. 202, 2023, pp. 8469–8488.

[30] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "ChatGPT for robotics: Design principles and model abilities," *IEEE Access*, vol. 12, pp. 55682–55696, 2024.

[31] W. Huang et al., "Inner monologue: Embodied reasoning through planning with language models," in *Proc. Conf. Robot Learn.*, 2023, vol. 205, pp. 1769–1782.

[32] J. Gao et al., "Physically grounded vision-language models for robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 12462–12469.

[33] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Ann. Rev. Control, Robot., Auton. Syst.*, vol. 5, pp. 411–444, 2022.

[34] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1917–1922.

[35] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. IEEE Eur. Control Conf.*, 2001, pp. 2603–2608.

[36] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 604–611, Apr. 2020.

[37] E. Soria, F. Schiano, and D. Floreano, "Distributed predictive drone swarms in cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 73–80, Jan. 2022.

[38] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "AMSwarm: An alternating minimization approach for safe motion planning of quadrotor swarms in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1421–1427.

[39] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 375–382, Apr. 2019.

[40] M. Müller, *Fundamentals of Music Processing - Audio, Analysis, Algorithms, Applications*. Berlin, Germany: Springer, 2015.

[41] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[42] J. Achiam et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.

[43] Y. Jiang et al., "VIMA: Robot manipulation with multimodal prompts," in *Proc. Int. Conf. Mach. Learn.*, vol. 202, 2023, pp. 14975–15022.

[44] "Crazyflow: Scalable crazyflie simulation using JAX and mujoco," 2025. Accessed: Jul. 29, 2025. [Online]. Available: https://github.com/utiasDSL/crazyflow

[45] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—A gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 7512–7519.

[46] J. Bradbury et al., "JAX: Composable transformations of Python NumPy programs," 2018. [Online]. Available: http://github.com/jax-ml/jax

[47] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3299–3304.