Semantic-Syntactic Integration for Generative Drone Light Show Design

Nico Posner

Richard Huang

Sarthak Dhanke

Huanlin Dai

Binting Xia

Advisor: Stephen Barry

University of Chicago

Master of Science in Applied Data Science

Research Capstone Project

November 6, 2025

Abstract

This work presents a structured pipeline for co-creative drone light show design that integrates large language models (LLMs) with analytical validation and deployment systems. Rather than treating LLMs as autonomous choreographers, the study reframes them as semantic front ends that generate interpretable creative intent, which is then translated into physically valid and executable drone formations. The proposed framework couples language-based generative interfaces with analytic solvers and the **Skybrush Studio API**, which performs safety verification, trajectory optimization, and compilation into deployable show formats. Through this integration, we demonstrate how semantic creativity and syntactic rigor can coexist within a single workflow, enabling intuitive yet verifiable design of drone swarm performances. The findings clarify where language-driven generation adds value, where it requires analytic reinforcement, and how modular architectures can support collaboration between human designers, generative models, and production-grade control software.

Keywords: drone light shows, large language models, semantic-syntactic integration, Skybrush Studio API, generative design pipeline, creative robotics, human-AI collaboration, formation sampling, trajectory optimization

Executive Summary

Designing drone light shows requires the reconciliation of two fundamentally different modes of reasoning: artistic ideation and analytical precision. While commercial tools such as SPH Engineering's Drone Show Software, Verge Aero's Design Studio, and Vimdrones Designer provide robust production environments for professionals, they remain heavily procedural. Designers must manually specify formations, transitions, and safety constraints within rigid graphical interfaces. These systems guarantee operational safety but offer limited support for the exploratory, iterative, and semantically rich stages of creative design.

At the same time, advances in large language models (LLMs) have made it possible to express complex spatial and temporal ideas in natural language. Recent academic work — including CLIPSwarm, SwarmGPT-Primitive, Swarm-GPT, FlockGPT, and LLM-Flock — has attempted to harness this capability to generate drone swarm behaviors directly from prompts. Yet these approaches largely treat LLMs as translators rather than collaborators. They produce symbolic or geometric intermediates that must still be processed by traditional solvers, without engaging with the workflows, constraints, or interpretive needs of professional show designers.

This project addresses that disconnect by proposing and implementing an integrated design pipeline that unites the semantic flexibility of LLMs with the syntactic rigor of analytical systems. Instead of attempting to have a language model produce executable trajectories directly, we introduce a structured framework in which generative models act as semantic front ends. They produce interpretable spatial representations—images, meshes, or symbolic formations—that are subsequently sampled, optimized, validated, and compiled through analytical methods. The final stage of this process integrates with the **Skybrush Studio API**, which performs automated safety checks, trajectory optimization, and binary compilation into deployable show files (`.csv` or `.skyc` formats).

Through this architecture, the project reframes LLM-assisted design not as an automation problem, but as a problem of *semantic–syntactic integration*. The pipeline demonstrates that

human creativity, machine interpretation, and analytical enforcement can coexist in a modular, interoperable workflow. The language model proposes; the analytical system verifies; the human designer iterates.

The research proceeded through four principal phases:

1. **Workflow Analysis:** Mapping the tools, cognitive processes, and constraints of existing professional pipelines through documentation review and, where possible, practitioner input.

2. **Pipeline Development:** Implementing a modular end-to-end system encompassing language interpretation, coordinate generation, temporal optimization, and validation through the Skybrush Studio API.

3. **Evaluation:** Assessing the semantic coherence, syntactic validity, and iterative usability of generated formations across multiple model architectures and prompting strategies.

4. **Design Synthesis:** Deriving practical guidelines for structuring prompts, incorporating analytic feedback, and supporting human–AI co-creation within production workflows.

The findings suggest that LLMs are most valuable not as autonomous generators of executable drone paths, but as accelerators of creative ideation. When coupled with analytical solvers and validation APIs, they enable rapid exploration of conceptual designs that can be safely transformed into physical performances. In this way, the project contributes both a working prototype and a conceptual model for integrating semantic and syntactic intelligence in creative robotics.

Ultimately, this work demonstrates that the path toward accessible and interpretable drone show design lies not in replacing human expertise, but in structuring collaboration—between human designers, generative models, and the analytical infrastructures that ensure safe and executable outcomes.

Table of Contents

List of Figures

List of Figures

List of Tables

List of Tables

List of Appendices

Introduction

Problem Statement

Drone light shows represent a unique synthesis of creative design and algorithmic precision, transforming abstract artistic concepts into tightly coordinated aerial performances. While their visual and cultural impact has grown rapidly, the process of designing such shows remains complex, highly specialized, and constrained by strict physical and regulatory boundaries. Translating human imagination into executable drone trajectories requires both aesthetic sensitivity and formal guarantees of safety, spacing, and feasibility.

Recent proposals have explored using Large Language Models (LLMs) to bridge the gap between natural language expression and swarm behavior generation. In principle, this approach offers an intuitive design interface: a user describes the desired effect — "form a spiral that blossoms into a sphere" — and an LLM translates that prompt into flight paths or formation parameters. However, existing systems such as CLIPSwarm, SwarmGPT, and LLM-Flock have largely remained proof-of-concept demonstrations. These architectures use language models as semantic translators, producing intermediate representations such as geometric outlines or waypoints, which are then refined through separate analytic solvers. As a result, the LLM occupies a peripheral role: it initiates the design process but does not ensure executable, safe, or optimized outcomes.

The practical consequence of this separation is that current research prototypes have not achieved the level of reliability or workflow integration required for real-world deployment. The artistic and semantic potential of language-based co-design remains underutilized, while the technical constraints that define feasible drone motion are handled independently through manual or traditional optimization methods. This disconnect mirrors broader challenges in LLM-assisted content generation, where models excel at producing semantically rich ideas but struggle when constrained by formal syntactic or physical rules.

Our work begins from this recognition and seeks to explore whether these two modalities — the semantic flexibility of language models and the syntactic precision of analytical systems —

can be combined more effectively. Specifically, we propose an integrated pipeline in which the LLM functions not as a direct controller but as a generative front-end for a structured, verifiable production system. The final stages of this pipeline, including trajectory validation and compilation into deployable show files, are handled through the **Skybrush Studio API**, a professional toolchain for drone show design and execution. This framework grounds the generative capabilities of LLMs within the constraints of industry-standard safety and performance requirements.

Analysis Goals

The central objective of this research is to evaluate how language-based systems can meaningfully contribute to the workflow of drone show design, rather than to demonstrate isolated technical feasibility. We aim to examine how semantic generative processes can be coupled with formal analytic stages to produce syntactically valid and operationally safe results.

Accordingly, we define the following guiding questions:

1. What are the real creative, technical, and operational constraints that define professional drone show design?

2. How can the expressive power of LLMs be harnessed to produce meaningful inputs for downstream analytical solvers and simulation systems?

3. Can we formalize a pipeline in which semantic generation and syntactic enforcement coexist — preserving creative flexibility while ensuring feasibility?

4. What role can established production systems such as Skybrush Studio play in this integration, and how might such a hybrid approach inform future design interfaces?

This investigation treats LLMs not as autonomous choreographers but as components within a larger cybernetic structure. By analyzing their performance in tandem with analytical solvers, we seek to clarify their true contribution to the creative process and identify where the boundary between generative flexibility and formal verification should lie.

Scope

This project focuses on the design phase of drone light show creation, emphasizing the transition from conceptual imagery or verbal description to a validated, executable formation plan. It does not involve real-time flight control, deployment logistics, or novel control theory. Rather, it seeks to bridge the semantic and syntactic stages of the production pipeline.

The scope of this work includes:

- Developing and testing a modular pipeline that combines generative LLM outputs with analytic transformation and validation stages.
- Implementing 2D and 3D formation generation from images and meshes using sampling algorithms inspired by computer graphics.
- Integrating the final stage of compilation and verification through the Skybrush Studio API, enabling compatibility with professional-grade drone control systems.
- Exploring the potential of Model Context Protocol (MCP)-style interaction to provide real-time, human-in-the-loop editing within design environments such as Blender.

We do not aim to develop a self-contained generative model for drone trajectories, nor to produce a fully autonomous end-to-end system. The purpose of this work is to formalize and test a *structured interface* between generative language systems and the analytic frameworks that ensure physical validity. In doing so, it provides both a practical and conceptual foundation for future tools that support co-creative, interpretable, and safe design of drone swarm performances.

Background

Commercial Tools

The design and execution of professional drone light shows are supported by a small ecosystem of proprietary software suites, including SPH Engineering's Drone Show Software,

Verge Aero's Design Studio, and Vimdrones Designer. These platforms provide integrated environments for defining 3D formations, scripting transitions, synchronizing lighting, and validating physical constraints. Their primary users are production companies and technical directors who work directly with proprietary drone fleets, often under hardware-specific licenses.

While these systems are robust and production-ready, they are not designed for exploratory or co-creative use. Their interfaces are graphical and timeline-based, emphasizing precision editing over conceptual ideation. Designers must specify individual formations, transitions, and timing parameters manually, with little algorithmic assistance during the early creative phase. Moreover, these environments assume substantial domain expertise: familiarity with aeronautical constraints, spatial reasoning in three dimensions, and the structure of executable show files. This combination of technical and artistic knowledge creates a high barrier to entry and limits the accessibility of drone choreography as a creative medium.

In recent years, **Skybrush Studio** has emerged as an influential hybrid system, bridging production-grade deployment with modular, API-accessible design tools. Its architecture separates front-end interfaces such as Blender extensions from a back-end engine that performs safety validation, trajectory optimization, and compilation into binary formats (e.g., `.skyc`). The **Skybrush Studio API** allows external systems to interact with this back-end directly through HTTP requests, enabling procedural or automated generation of drone flight plans. This capacity positions Skybrush not only as a professional editing suite but also as a potential integration layer for research pipelines that combine semantic generation with analytical verification.

From the standpoint of creative ideation, however, all these tools — including Skybrush in its commercial form — remain fundamentally syntactic. They excel at enforcing constraints and ensuring safety but offer no mechanisms for generating or interpreting creative intent expressed in natural language or visual form. The "semantic front end" of drone choreography remains largely undeveloped.

Academic Work

Academic research into LLM-assisted swarm design has sought to fill this semantic gap by introducing generative, language-driven approaches to formation and trajectory planning. Across systems such as *CLIPSwarm*, *SwarmGPT-Primitive*, *Swarm-GPT*, *LLM-Flock*, and *FlockGPT*, the central objective is consistent: to make swarm design more intuitive by allowing human operators to describe desired behaviors in natural language.

While conceptually aligned, these systems differ in scope and architecture. **CLIPSwarm** uses CLIP embeddings to associate textual prompts with 2D geometric outlines, producing silhouettes through iterative refinement of contour shapes. Its outputs are visually legible but limited to planar designs without dynamic motion or 3D extension. **SwarmGPT-Primitive** adopts a library-based approach, mapping language to predefined motion primitives that are synchronized with musical beats; it provides consistency but constrains expressiveness. **Swarm-GPT** attempts a more flexible waypoint generation process, where LLMs output raw coordinates subsequently filtered for safety, but the method suffers from verbosity, limited editability, and fragile temporal coherence.

LLM-Flock departs from centralized control by embedding LLMs directly on each agent, coordinating flight patterns through decentralized consensus. Though theoretically elegant, the approach introduces unpredictability and remains confined to simple geometric formations. Finally, **FlockGPT** introduces a dialogue-based interaction model, using signed distance functions to represent target shapes and allowing the user to iteratively refine geometry through conversation. Yet even this interaction model operates on static spatial data and does not address the temporal or safety constraints inherent in executable drone shows.

Across this literature, the primary role of the language model remains interpretive or symbolic: it translates natural language into an intermediate representation but defers all physical validation, optimization, and compilation to analytic solvers. The separation between semantic expression and syntactic enforcement persists, mirroring the same division seen in professional production tools.

Identified Gaps

Taken together, commercial and academic approaches outline two complementary but disconnected strengths. Commercial software provides the formal infrastructure — robust safety guarantees, validated trajectories, and executable output formats — but offers little support for creative ideation. Academic systems, conversely, provide generative flexibility through natural language or image-based interfaces, yet lack integration with the analytical frameworks that ensure real-world feasibility.

This research seeks to reconcile these domains through a structured, end-to-end pipeline that unites semantic generation and syntactic validation. The core insight is that LLMs should not attempt to directly produce executable flight paths, but rather to generate interpretable intermediate representations — such as 2D or 3D spatial configurations — which are then transformed, optimized, and verified through established analytical systems. The **Skybrush Studio API** provides a natural anchor for this integration, offering a programmatic interface where creative outputs from generative models can be converted into validated trajectories and compiled into deployable show formats.

In summary, the key methodological gaps motivating this work are:

- The absence of an integrated pipeline linking semantic generation to formal verification.
- The lack of human-centered design frameworks that combine creative exploration with operational feasibility.
- The need for modular interfaces — such as those provided by Skybrush — that can mediate between generative AI systems and existing production infrastructures.

This project positions itself at that intersection: developing a structured, hybrid workflow that unifies language-driven ideation with professional-grade safety and execution frameworks.

Methodology

This work formalizes and implements a structured pipeline for co-creative drone light show design that integrates semantic generation by large language models (LLMs) with the syntactic validation and compilation capabilities of analytical solvers. The methodology proceeds from a mapping of existing workflows to the construction of a modular design pipeline, culminating in the export of validated trajectories through the Skybrush Studio API.

Workflow Analysis

The first stage of this research involves a systematic analysis of professional drone show design workflows. While the conceptual focus of this study lies in AI-assisted co-creation, it is essential that the proposed pipeline reflects the actual practices, constraints, and cognitive demands of current production environments.

- **Primary Research:** Interviews and correspondence with professional drone show choreographers, if accessible, to identify the stages of ideation, formation design, sequencing, and safety validation that define standard production workflows.
- **Secondary Research:** A review of technical manuals, training materials, and user documentation from commercial tools such as SPH Engineering's Drone Show Software, Verge Aero's Design Studio, and Skybrush Studio. This analysis reconstructs the implicit logic of these platforms—the constraints they enforce, their data structures, and the order in which creative and technical steps are performed.

This workflow mapping establishes the empirical grounding for the pipeline: it clarifies which design functions can be meaningfully automated, which require human oversight, and how a semantic generation process can integrate with the syntactic verification systems already used in practice.

Pipeline Architecture

Building on this foundation, we designed a modular end-to-end pipeline that mirrors the conceptual separation between *semantic generation* and *syntactic enforcement*. The system consists of four primary stages: input interpretation, coordinate generation, optimization and validation, and export.

1. **Semantic Front End:** User intent is captured through natural language prompts or visual references. The LLM interprets these inputs to produce intermediate representations—textual descriptions, geometric specifications, or symbolic formation data. This stage is analogous to high-level code generation in agentic programming: the model proposes structures without guaranteeing their syntactic validity.

2. **Coordinate Generation:** The interpreted data are transformed into discrete drone coordinates. For 2D imagery, this involves sampling from binary or thresholded masks using point-selection algorithms such as greedy furthest-point or Poisson blue-noise sampling. For 3D models, polygonal meshes serve as sampling surfaces, with point distributions constrained by inter-drone distance thresholds to avoid collisions. These sampling methods formalize the "glyph rasterization" of visual data into spatial coordinates.

3. **Optimization and Temporal Linking:** When extended to animations or multi-frame sequences, a constrained optimization solver minimizes the distance between drone positions across frames while maintaining spatial separation constraints. This ensures smooth trajectories and continuous motion without collisions or visual stuttering.

4. **Validation and Export:** The resulting coordinates and flight paths are serialized into a format compatible with the **Skybrush Studio API**. The API backend performs safety verification, interpolates feasible trajectories, and compiles the validated data into executable show formats such as `.csv` or `.skyc`. This stage enforces all syntactic requirements of drone flight execution.

By distributing the generative and analytic responsibilities across these stages, the system ensures that creative exploration remains flexible while operational validity is guaranteed by established solvers.

Integration with Skybrush Studio API

The Skybrush Studio API serves as the backbone of the syntactic enforcement stage. It exposes HTTP endpoints that accept structured drone show descriptions in JSON or CSV form and return verified or compiled outputs ready for execution in Skybrush Live or equivalent control software.

In our implementation, this API functions as the *compiler* within the pipeline. It processes semantic output from the generative stage, applies formal safety checks—such as maximum velocity limits, altitude boundaries, and minimum spacing—and returns an optimized trajectory plan. This modular integration allows the pipeline to remain model-agnostic: any LLM or image generator can serve as the semantic source, provided that its output conforms to the schema required by the API.

The choice to use an external, production-grade API rather than a custom solver reflects a broader methodological commitment: to demonstrate not only the feasibility of semantic–syntactic coupling in principle, but also its practical interoperability with existing industrial infrastructures.

Evaluation

Evaluation of the system focuses on qualitative and structural criteria rather than numerical performance benchmarks. The objective is to determine whether the proposed pipeline enhances interpretability, usability, and alignment with professional workflows.

- **Semantic Coherence:** Assess whether LLM-generated formations correspond consistently to the user's described intent across prompts and iterations.
- **Syntactic Validity:** Verify that exported formations pass Skybrush API validation without requiring manual correction.

- **Editability:** Measure the ease with which users can adjust intermediate outputs—either through re-prompting or direct parameter modification—without destabilizing the pipeline.
- **Iterative Efficiency:** Compare the time required to produce an acceptable formation or sequence against baseline manual methods within commercial software.
- **Transparency:** Evaluate whether the system's modular structure makes its transformations comprehensible to users, facilitating trust and learning.

Where feasible, these evaluations will be supplemented by qualitative feedback from domain professionals and by visual inspection of generated trajectories within Skybrush Studio.

Limitations

Several limitations are inherent in this methodology. First, the pipeline depends on the accuracy and interpretability of LLM outputs, which may exhibit semantic drift or ambiguity. Second, while the Skybrush Studio API enforces operational safety, it does not optimize for aesthetic quality, leaving subjective refinement to human users. Third, the optimization process for temporal transitions is computationally intensive and may not scale linearly with the number of drones or frames.

Finally, the current implementation assumes modular interoperability among independently developed systems—LLMs, sampling algorithms, and industrial solvers—that were not designed to function together. While this architecture demonstrates conceptual viability, further engineering would be required to achieve real-time performance or deployment readiness.

Despite these constraints, the methodology provides a practical framework for bridging semantic generation and syntactic enforcement in the context of creative robotics. It establishes a reproducible structure that future work can refine, extend, and generalize to other domains where expressive design must coexist with formal constraints.

Findings

[Finding Content Here]

Discussion

[Discussion content here]

Conclusion

[Conclusion Content Here]

Additional Data Analysis

[Additional analysis content]

Code Documentation

[Code documentation]

LLM Prompt Examples and Outputs

[LLM prompts and outputs]