Semantic Trajectory Generation for Goal-Oriented Spacecraft Rendezvous

Yuji Takubo *, Arpit Dwivedi †, Sukeerth Ramkumar ‡, Luis A. Pabon §, Daniele Gammelli ¶,
Marco Pavone ‖, Simone D'Amico **
Stanford University, Stanford, CA, 94305

Reliable real-time trajectory generation is essential for future autonomous spacecraft. While recent progress in nonconvex guidance and control is paving the way for onboard autonomous trajectory optimization, these methods still rely on extensive expert input (e.g., waypoints, constraints, mission timelines, etc.), which limits the operational scalability in real rendezvous missions. This paper introduces SAGES (Semantic Autonomous Guidance Engine for Space), a trajectory-generation framework that translates natural-language commands into spacecraft trajectories that reflect high-level intent while respecting nonconvex constraints. Experiments in two settings—fault-tolerant proximity operations with continuous-time constraint enforcement and a free-flying robotic platform—demonstrate that SAGES reliably produces trajectories aligned with human commands, achieving over 90% semantic-behavioral consistency across diverse behavior modes. Ultimately, this work marks an initial step toward language-conditioned, constraint-aware spacecraft trajectory generation, enabling operators to interactively guide both safety and behavior through intuitive natural-language commands with reduced expert burden.
Project Website: `https://semantic-guidance4space.github.io/`

I. Introduction

Aᴜᴛᴏɴᴏᴍᴏᴜs decision-making for spacecraft is essential for the emerging era of the future space ecosystem. In particular, automated rendezvous, proximity operations, and docking technology enable frequent space logistics operations, including on-orbit servicing, crewed docking, and in-space manufacturing, which could realize a sustainable economy in orbit and beyond. A central requirement for these rendezvous missions is real-time trajectory generation that can jointly optimize mission objectives while satisfying complex operational constraints.

Motivated by these ambitious goals, recent advances in nonlinear and nonconvex trajectory optimization have significantly enhanced the autonomy and reliability of spacecraft guidance and control systems. There exist numerous algorithms that solve optimal control numerically, including direct methods [1], indirect methods [2–4], and differential dynamic programming [5]. Among these approaches, Sequential Convex Programming (SCP) [6] has emerged as one of the most promising nonconvex guidance algorithms for practical deployment. By iteratively solving a convexified problem until convergence, SCP provides a computationally efficient and reliable strategy for spacecraft trajectory optimization. Its effectiveness has been demonstrated in a range of aerospace applications, from powered descent and landing [7, 8] to spacecraft rendezvous [9, 10]. This practical success is supported by theoretical works establishing convergence guarantees to local optima [11–13], as well as research broadening the class of nonconvex constraints that SCP can handle. Notable extensions include mixed-integer formulations [14–16], uncertainty-aware methods—both chance-constrained [17] and moment-based approaches such as covariance steering [18–20]—and techniques for enforcing continuous-time constraints [21].

Despite these advances, the use of nonconvex trajectory optimization in complex rendezvous operations remains limited by two key challenges. First, human experts must still manually specify problem parameters, such as waypoints, timelines, operational constraints, and objectives. The ability to formulate such problems quickly is essential for autonomous decision-making, responsive operations, and agile mission and navigation analysis. Yet achieving this level

---

*Ph.D. Candidate, Department of Aeronautics and Astronautics, 496 Lomita Mall, AIAA Student Member.
†Master Student, Department of Aeronautics and Astronautics, 496 Lomita Mall.
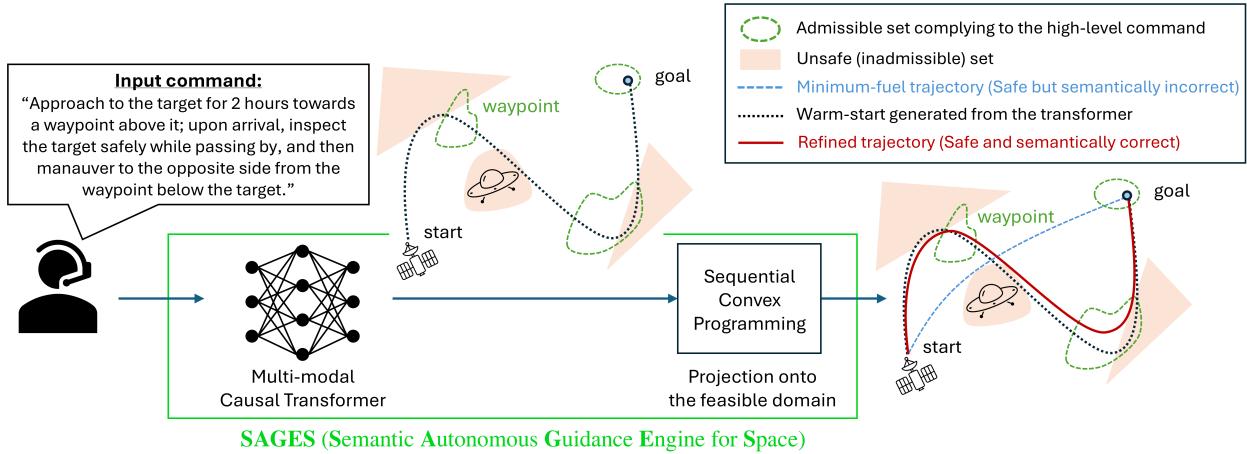‡Master Student, Department of Mechanical Engineering, 440 Escondido Mall Building 530.
§Ph.D. Candidate, Department of Aeronautics and Astronautics, 496 Lomita Mall.
¶Postdoctoral Fellow, Department of Aeronautics and Astronautics, 496 Lomita Mall.
‖Associate Professor, Department of Aeronautics and Astronautics, 496 Lomita Mall. AIAA Associate Fellow.
**Associate Professor, Department of Aeronautics and Astronautics, 496 Lomita Mall. AIAA Associate Fellow.

Fig. 1 Overview: The proposed framework SAGES (Semantic Autonomous Guidance Engine for Space) takes a natural-language command as input and produces a semantically correct, constraint-satisfying trajectory. A multimodal encoder–decoder model (a transformer conditioned on text, constraint specifications, and the current system state) autoregressively predicts the optimal next action at each step, generating an initial trajectory that aligns with the requested behavior. This trajectory is then refined using SCP, which projects it onto the feasible set.

of flexibility requires deep expertise in optimal control, making large-scale deployment of nonconvex guidance algorithms difficult. Second, classical objective functions such as fuel or time minimization often lack the expressive power needed to capture the full range of desired behaviors. As a result, operators rely on the detailed waypoint construction to shape the intended rendezvous trajectory. However, for space rendezvous, the geometry and kinematics of relative motion [22, 23] introduce subtleties that require significant astrodynamics knowledge, making it challenging to design waypoints that reliably yield the intended behavior. Moreover, with the advent of high-capacity propulsion systems for multi-spacecraft servicing and orbit-transfer vehicles, control effort during high-precision proximity operations has become comparatively inexpensive in a wide range of orbital regimes [24–26]. This shift unlocks a design space where slight increases in fuel usage can be traded for diverse behaviors and more complex maneuvers—echoing goal-oriented autonomy in terrestrial robotics [10, 27]. Nevertheless, current methods based on optimal control formulations that focus solely on cost minimization are inherently limited in their ability to leverage this opportunity.

Given these challenges, a language-conditioned interface offers a promising new direction for trajectory generation in space. Such an interface would allow non-expert spacecraft operators and systems engineers to specify high-level mission objectives in natural language, facilitating rapid synthesis of dynamically feasible trajectories that align with task semantics.

Concurrently, the emergence of internet-scale, broadly capable Foundation Models (FMs) offers an opportunity to fundamentally rethink how autonomous systems are designed, deployed, and operated. Trained on vast and diverse datasets, these models capture broad priors about the world and have achieved breakthroughs in vision, language, and multi-modal reasoning. Building on this momentum, a substantial body of work has explored how to adapt pretrained FMs for robot control [28–31]. Within the space domain, Large Language Model (LLM)-driven spacecraft autonomy is beginning to emerge—for instance, in systems that directly map textual commands to control inputs [32–34] or perform semantic reasoning over multimodal inputs [35]. However, these approaches typically assume closed-loop operation and relegate most of the dynamics and constraint handling to lower-level controllers. This paradigm is poorly suited to aerospace applications, which require trajectory-level safety guarantees and operate in environments with virtually zero tolerance for error. Moreover, although spacecraft dynamics are known with far greater precision than those of terrestrial robots (e.g., locomotion involving contact forces), current FM-based methods do not make intrinsic use of this analytical structure during policy generation. Instead, they rely on data-driven approximations that lack formal verification, limiting their suitability for safety-critical space missions.

As a first step toward addressing these limitations, this paper introduces **SAGES** (**S**emantic **A**utonomous **G**uidance **E**ngine for **S**pace, Fig. 1), a text-conditioned spacecraft trajectory generation framework that translates high-level language commands into feasible trajectories under nonconvex constraints. The proposed approach bridges semantic

intent and continuous dynamics through a two-stage architecture: (1) a multi-modal encoder-decoder model embeds language descriptions and constraint specification into a shared latent space and autoregressively generates a semantically meaningful trajectory sequence, and (2) a SCP layer then refines this trajectory to enforce principled constraint satisfaction with respect to dynamics and operational constraints. This framework builds upon the Autonomous Rendezvous Transformer (ART) [36], a transformer-based trajectory generation model originally designed to provide an initial guess (warm-start) to an SCP. Unlike previous ART formulations [36–40], SAGES treats SCP not as a local optimizer but as a projector to the feasible domain: any converged SCP solution strictly satisfies the encoded constraints while preserving semantic and temporal coherence. By integrating high-level language conditioning with a dynamics-aware decision model, the proposed framework enables coherent, constraint-satisfying trajectory generation that unifies semantic intent and physics-based optimization.

The contributions of this paper are threefold:
- A new paradigm for semantic spacecraft trajectory generation that explicitly incorporates both hard safety constraints and soft, high-level behavioral specifications through FMs.
- The proposed **SAGES** framework integrates a multimodal, transformer-based semantic trajectory generator with an SCP-based feasible-set projector, enabling the computation of safe trajectories that satisfy quantitative task requirements while remaining aligned with high-level semantic intent.
- The effectiveness of SAGES is demonstrated in two domains: (i) a fault-tolerant spacecraft proximity operation scenario with continuous-time safety constraints, and (ii) hardware experiments on a free-flyer robotic testbed with an embedded GPU system. Across both settings, SAGES successfully demonstrates semantically guided safe autonomy, improving SCP's algorithmic performance while generating trajectories that execute the intended behavior.

The remainder of the paper is organized as follows. Section II reviews ART, a transformer-based trajectory generation framework that is foundational for SAGES. Section III introduces SAGES, a novel method that incorporates constraint-aware semantic trajectory generation. Section IV presents two experimental setups: a free-flying robotic testbed and a spacecraft proximity operation. Section V reports the results of the proposed framework, and Section VI concludes the paper.

II. Background: Autonomous Rendezvous Transformer

The Autonomous Rendezvous Transformer (ART) [36] is an approach that combines optimization-based and learning-based methods for spacecraft trajectory optimization. The method entails leveraging high-capacity (namely, transformer-based) neural network models for the solution of discrete-time optimal control problems of the form:

$$\min_{\{\boldsymbol{x}_k\}_{k=1}^{N}, \{\boldsymbol{u}_k\}_{k=1}^{N}} \quad \mathcal{J} = \sum_{k=1}^{N} j_k(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) \tag{1a}$$

$$\text{subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{F}_k(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad \forall k = 1, ..., N-1, \tag{1b}$$

$$(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{S}_k, \quad \forall k = 1, ..., N, \tag{1c}$$

where $\boldsymbol{x}_k = \boldsymbol{x}(t_k) \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k = \boldsymbol{u}(t_k) \in \mathbb{R}^{n_u}$ denote the system state and control input at discrete timestep k. The objective function $\mathcal{J}$ aggregates the stage costs j_k over a fixed horizon $t \in [0, t_f]$ discretized by N steps with the timestep $\Delta t = t_f/(N-1)$, and $\boldsymbol{F}_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ defines the discrete-time system dynamics. The admissible state–control pairs at each step are constrained within the feasible set $\mathcal{S}_k \subseteq \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$, which encodes all operational and physical constraints. More generally, the feasible region may consist of multiple disjoint constraint sets, expressed as a finite union $\mathcal{S}_k = \mathcal{S}_{k,1} \cup \mathcal{S}_{k,2} \cup \cdots \cup \mathcal{S}_{k,n_c}$, where each $\mathcal{S}_{k,i}$ represents a distinct admissible subset.

The central idea of ART is to train a transformer that predicts future states and control inputs conditioned on past states, actions, and mission context. These predictions provide a high-quality initial guess for the SCP, thereby accelerating convergence and enabling the solver to obtain more optimal solutions. A key enabler of ART for robust trajectory generation is the state representation (i.e., tokenization) of the continuous trajectory information to a sequence of states, controls, and corresponding performance metrics. The general tokenized trajectory representation is defined as:

$$\tau_{1:N} = \{\mathcal{P}_1, \boldsymbol{x}_1, \boldsymbol{u}_1, ..., \mathcal{P}_N, \boldsymbol{x}_N, \boldsymbol{u}_N\}, \tag{2}$$

where $\mathcal{P}_k$ is a performance metric vector. In particular, two performance metrics are considered in this work as $\mathcal{P}_k = \{r_k, c_k\}$, where $r_k \in \mathbb{R}$ is the *reward-to-go* and $c_k \in \mathbb{N}$ is the *constraint-to-go*, defined as:

$$r_k = -\sum_{j=k}^{N} \mathcal{J}(\boldsymbol{x}_j, \boldsymbol{u}_j, t_j) \tag{3a}$$

$$c_k = \sum_{j=k}^{N} \mathbf{1}_{\mathcal{S}_j}(\boldsymbol{x}_j, \boldsymbol{u}_j), \quad \mathbf{1}_{\mathcal{S}_k}(\boldsymbol{x}_k, \boldsymbol{u}_k) := \begin{cases} 1 & \text{if } (\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{S}_k \\ 0 & \text{otherwise.} \end{cases} \tag{3b}$$

Together, these metrics quantify the cumulative optimality and feasibility of the remaining trajectory beginning at step k.

III. SAGES: Semantic Autonomous Guidance Engine for Space

Building on ART, this paper introduces SAGES, a framework for language-conditioned trajectory generation. The objective is to compute trajectories that satisfy the following problem:

$$\text{find} \quad \{\boldsymbol{x}_k\}_{k=1}^{N}, \{\boldsymbol{u}_k\}_{k=1}^{N} \tag{4a}$$

$$\text{subject to} \quad (\{\boldsymbol{x}_k\}_{k=1}^{N}, \{\boldsymbol{u}_k\}_{k=1}^{N}) \in (\text{Set of trajectories consistent with a high-level command}). \tag{4b}$$

$$(1\text{b}), (1\text{c}) \tag{4c}$$

In other words, the generated trajectories must satisfy the hard constraints defined in the optimal control formulation (cf. Eq. (1)) while also adhering to a soft, semantic constraint that enforces consistency with the behavior specified by the natural-language command. This formulation emphasizes feasibility under semantic guidance rather than explicit cost minimization.

To enable text-conditioned safe trajectory generation, SAGES adopts a trajectory tokenization scheme that extends ART's original representation by incorporating text information into a shared latent space [41]. The resulting token sequence is defined as:

$$\tau_{1:N} = \{\boldsymbol{e}, c_1, \boldsymbol{x}_1, \boldsymbol{u}_1, c_2, \boldsymbol{x}_2, \boldsymbol{u}_2, ..., c_N, \boldsymbol{x}_N, \boldsymbol{u}_N\}, \tag{5}$$

where $\boldsymbol{e}$ denotes the text embedding vector, given by:

$$\boldsymbol{e} = [\boldsymbol{e}_1, ..., \boldsymbol{e}_{N_e}] = f_{\text{enc}}(z; \theta_{\text{enc}}). \tag{6}$$

Here, f_{enc} denotes a text encoder with frozen weights θ_{enc}, and z is the natural-language input specifying the high-level (trajectory-level) command. Each encoder output token $\boldsymbol{e}_k \in \mathbb{R}^h$ represents the embedding of a (sub-)word and shares the same dimensionality as the hidden space used for ART's trajectory tokens. The full text embedding $\boldsymbol{e}$ has a fixed length of N_e. Commands containing fewer than N_e (sub-)tokens are zero-padded, whereas longer commands are truncated to match the predefined embedding dimension.*"
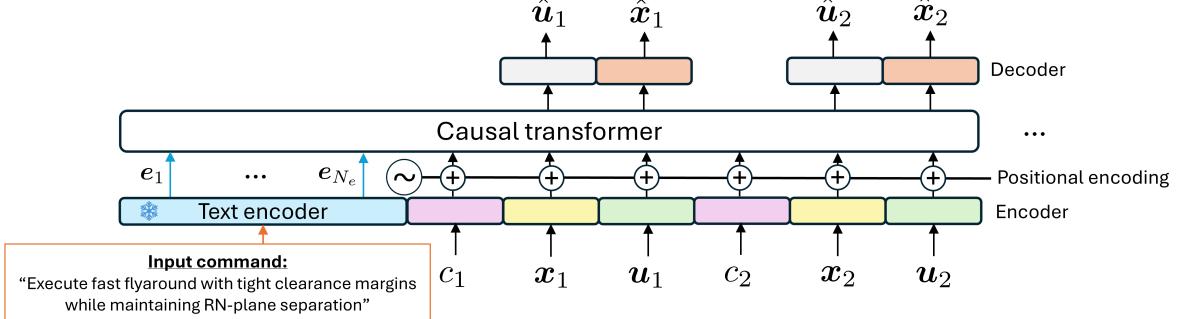
To process a sequence of tokens in Eq. (5) and generate trajectories, a causal transformer (cf. Fig. 2) that generates trajectories in the joint space of textual information and constraint signals is introduced.

The workflow of SAGES comprises four phases: (i) dataset generation, (ii) transformer training, (iii) test-time inference, and (iv) trajectory refinement via SCP.

Dataset Generation: A paired dataset of trajectories and high-level text commands must first be constructed to train the model. Each dataset element consists of a spacecraft trajectory and a natural-language description of the associated behavior. The goal is to achieve broad diversity across operating conditions, problem parameters, and linguistic expressions. This diversity is critical for the learning model to learn across a wide solution space and generalize in both control performance and semantic interpretation. In this work, we propose a scalable language-annotation pipeline for spacecraft trajectories enabled by LLMs. A finite set of high-level behavior modes is first defined, each characterized by an ordered sequence of waypoints and the corresponding times at which they are reached. For each behavior mode, a large pool of trajectories satisfying these waypoint constraints is generated using a nonconvex optimization framework (SCP). In parallel, a separate and typically smaller pool of text commands is produced using an LLM, which greatly accelerates the generation of diverse natural-language descriptions. Because the trajectories and commands

---

*An end-of-sentence (EOS) token may be appended easily to indicate termination of variable-length commands.

Fig. 2 Neural network architecture for semantic trajectory generation in SAGES. Natural language task descriptions, constraint signals, states, and controls are encoded into a shared embedding space, which is used to predict next-step control and state variables.

are created independently, their pool sizes may differ. After generation, the two pools are randomly shuffled, and command-trajectory pairs associated with the same behavior mode are drawn to form the final dataset. This decoupled procedure allows large-scale trajectory synthesis while keeping the language-generation burden manageable. The detailed dataset-generation procedure adopted in this paper is provided in Appendix A.

Training: After generating the dataset, the transformer is trained using standard teacher forcing for next-token prediction. Given a text command describing the high-level intent of the maneuver, the text encoder produces a semantic embedding (cf. Eq. (6)). Because the command corresponds to a trajectory-level instruction rather than a token-ordered sequence, no positional encoding is applied to the text embedding [41]. This embedding is then concatenated with the constraint signal and the state–action history and fed into a causal transformer. The transformer's output layer splits into two task-specific heads that predict the next state and the next control input, respectively, thereby enabling joint autoregressive rollout conditioned on both semantics and dynamics. The loss function over a batch of B trajectories is defined as:

$$\mathcal{L}(\boldsymbol{\tau}_{1:N}) = \sum_{b=1}^{B} \sum_{k=1}^{N} \left(\left\| \boldsymbol{x}_k^{(b)} - \hat{\boldsymbol{x}}_k^{(b)} \right\|_2^2 + \left\| \boldsymbol{u}_k^{(b)} - \hat{\boldsymbol{u}}_k^{(b)} \right\|_2^2 \right), \tag{7}$$

where $\hat{\boldsymbol{x}}_k$ and $\hat{\boldsymbol{u}}_k$ denote the model's predicted state and control input, and $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are their corresponding ground-truth values sampled from the dataset.

Test-time inference: After training, the transformer can be used to autoregressively generate spacecraft trajectories for novel input commands. During the test-time inference, only the control input prediction head is utilized to generate the entire trajectory in an autoregressive manner. At each step, the predicted control input is propagated through a known dynamics model $\boldsymbol{F}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$, which is the same model employed within the subsequent trajectory optimization problem. This model-in-the-loop rollout ensures that the generated trajectories remain dynamically feasible. The constraint-to-go is updated at every step based on the applied control input and the propagated state.

Trajectory Refinement via SCP: The generated trajectories are then used to provide a warm-start for the SCP. Therefore, the SCP is used to map the initial guess onto the feasible domain. In particular, SAGES solves the following problem as a particular form of the feasibility problem in Eq. (4a):
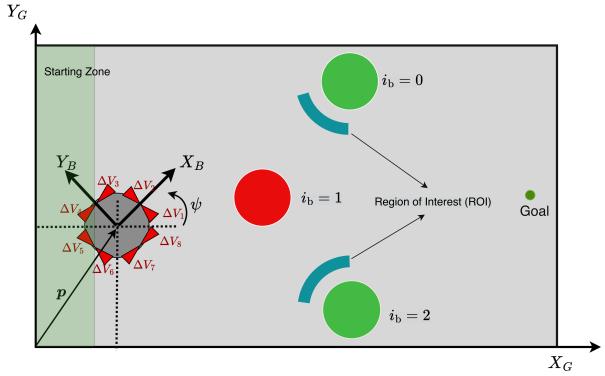
$$\min_{\{\boldsymbol{x}_k\}_{k=1}^{N}, \{\boldsymbol{u}_k\}_{k=1}^{N}} \tilde{\mathcal{J}} = \sum_{k=1}^{N} \left(\|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k\|_2^2 + \lambda \|\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k\|_2^2 \right) \tag{8a}$$

$$\text{subject to} \quad (1b), (1c) \tag{8b}$$

This design of the objective function ensures that the refined trajectory remains close to the warm-start solution, thereby preserving its semantic behavior, where $\lambda > 0$ is a hyperparameter. This is opposed to more traditional objectives—such as minimum-fuel optimization or strict projection onto the feasible domain (cf. Algorithm 1 in Ref. 42)—which can yield feasible but semantically inconsistent trajectories, as the solution may deviate substantially from the original warm-start. The main advantage of this framework is that, although SAGES can generate semantically aligned and safe trajectories, the refinement stage always prioritizes the feasibility of the trajectory. This hierarchy, where safety is prioritized over semantic alignment, is essential in safety-critical applications such as on-orbit maneuvering, where constraint violations cannot be tolerated.

5

(a) Hardware testbed on a quasi-frictionless table.

(b) Reference frames, thruster configurations, and obstacle locations.

Fig. 3 Overview of the free-flyer robotic testbed.

IV. Problem Scenarios

Two case studies are conducted to evaluate the performance of the proposed semantic trajectory generation framework: onboard a free-flyer robotic testbed and a fault-tolerant spacraft proximity operation scenario. This section describes the problem formulation for each scenario and the detailed procedure for dataset generation.

A. Free-flyer Robotic Testbed

The free-flyer system (cf. Fig. 3) is a planar, microgravity robotic testbed that provides two translational and one rotational degree of freedom on a quasi-frictionless granite table. It is widely used as a ground-based platform for validating control algorithms for space applications. The vehicle is actuated by eight on–off compressed CO_2 thrusters that generate an impulsive velocity change ΔV to achieve roto-translational motion. The workspace contains a central obstacle and two additional static bodies placed on the left and right sides of the table. The free-flyer initiates its maneuver from a predefined start region and approaches a fixed goal state, where these bodies create two flanking corridors to reach the goal. Figure 3a shows the physical hardware, whereas Fig. 3b illustrates the coordinate systems that are used in the following trajectory optimization: the global Cartesian reference frame O_G and the free-flyer's body-fixed reference frame O_B. The state of the free-flyer is represented as $\boldsymbol{x} := [\boldsymbol{p}, \boldsymbol{v}, \psi, \omega] \in \mathbb{R}^6$, where $\boldsymbol{p}, \boldsymbol{v} \in \mathbb{R}^2$ are the two-dimensional position and velocity of its center of mass in the global frame, and $\psi, \omega \in \mathbb{R}$ are the bearing angle from the X_G-axis and its angular velocity, respectively.

The following subsections outline the scenario design, covering (1) the definition of high-level behavior patterns, (2) the nonconvex trajectory optimization used for dataset generation, and (3) the subsequent nonconvex refinement stage.

1. Behavior Patterns and Model Conditioning

Table 1 summarizes the six distinct behavior modes considered in this scenario and provides representative example commands for each. These modes are defined based on (i) whether the vehicle approaches the goal via left-side passage, right-side passage, or a central traverse, and (ii) whether the traverse is fast or slow, which is dictated by the arrival time to the goal. In particular, the left and right passages are induced by requiring the trajectory to pass through a designated region of interest (ROI) before reaching the goal. The position of the waypoint, denoted as $\boldsymbol{p}_{\text{wyp}}$, is selected from the domain:

$$\boldsymbol{p}_{\text{wyp}} = \boldsymbol{p}_O^{(i_b^*)} + (R^{\text{KOZ}} + \rho) \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \quad \theta \in [\theta_0 - \Delta\theta_0, \theta_0 + \Delta\theta_0], \ \rho \in [\rho_{\min}, \rho_{\max}], \ i_b^* = \begin{cases} 0, & b \in \{0, 1\}, \\ 2, & b \in \{2, 3\}, \end{cases} \quad (9)$$

where $\boldsymbol{p}_O^{(i_b)} \in \mathbb{R}^2$ denotes the center position of the i_b-th body; $R^{\text{KOZ}} = \gamma(R_O^{(i_b)} + R_{\text{FF}})$ denotes the Keep-out Zone (KOZ) radius, which is the sum of the radius of the free-flyer and the body, denoted as R_{FF} and $R_O^{(i_b)}$, respectively, with a safety margin γ. Additionally, θ_0 denotes the bearing angle from the body of interest i_b^* to the obstacle ($i_b = 1$), and $\Delta\theta_0$ denotes the maximum displacement half-angle. The offsets in the polar coordinates (ρ, θ) are sampled from the

Behavior	Example Command
0: Left passage (fast)	"Execute a rapid left-side bypass, maintaining tight KOZ clearance with an agile maneuver profile."
1: Left passage (slow)	"Adopt a broad left-side arc, ensuring KOZ compliance with extended loiter in the central corridor."
2: Right passage (fast)	"Clearance maintained; sharp right-side arc executed with compressed schedule, ensuring KOZ compliance."
3: Right passage (slow)	"The trajectory follows a broad right-side arc, widening clearance for extended low-ΔV standoff."
4: Central traverse (fast)	"With precise KOZ compliance, a rapid central corridor sprint is executed via high RCS cadence."
5: Central traverse (slow)	"Prioritize broad central corridor transit, ensuring KOZ compliance with minimal Δv and cautious velocity."

Table 1   Behavior modes and sample text commands defined in the free-flyer scenario.

uniform distributions. The ROIs defined based on this formulation are illustrated in Fig. 3b. During annotation with text commands (cf. Table 1), each sentence is restricted to have fewer than 15 words, whereas the maximum text-embedding length is set to $N_e = 30$.

Based on the definition of the behavior modes, the semantic correctness is assessed by the following two criteria:

1) **ROI Passage:** The trajectory must pass through the designated ROI if specified (i.e., left and right passages).
2) **Terminal-Time Consistency:** For fast traversal modes, the free-flyer must reach the goal position within the error of ϵ_g m in less than 30 s. For slow traversal modes, the arrival epoch is required to fall between 35 and the nominal 40 s.

These criteria ensure alignment between high-level natural-language intent and the generated motion profile to evaluate semantic compliance of the generated trajectories.

2. Trajectory Optimization for Dataset Generation

The trajectory optimization problem of the free-flyer system used for the dataset generation is formulated as follows:

$$\min_{\{\boldsymbol{x}_k\}_{k=1}^N, \{\boldsymbol{u}_k\}_{k=1}^N} \quad \sum_{k=1}^N \|\boldsymbol{u}_k\|_1 + w \sum_{k \in \mathcal{I}_{\text{wyp}}} \left[\|\boldsymbol{p}_k - \boldsymbol{p}_{\text{wyp}}\|_2 - r_{\text{wyp}} \right]^2, \tag{10a}$$

$$\text{subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + [\boldsymbol{I}_3 \cdot \Delta t, \boldsymbol{I}_3]^\top \boldsymbol{u}_k, \qquad \forall k = 1, ..., N-1, \tag{10b}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_i, \quad \boldsymbol{x}_N = \boldsymbol{x}_f, \tag{10c}$$

$$\boldsymbol{x}_k \in \mathcal{X}_{\text{table}}, \qquad \forall k = 1, ..., N, \tag{10d}$$

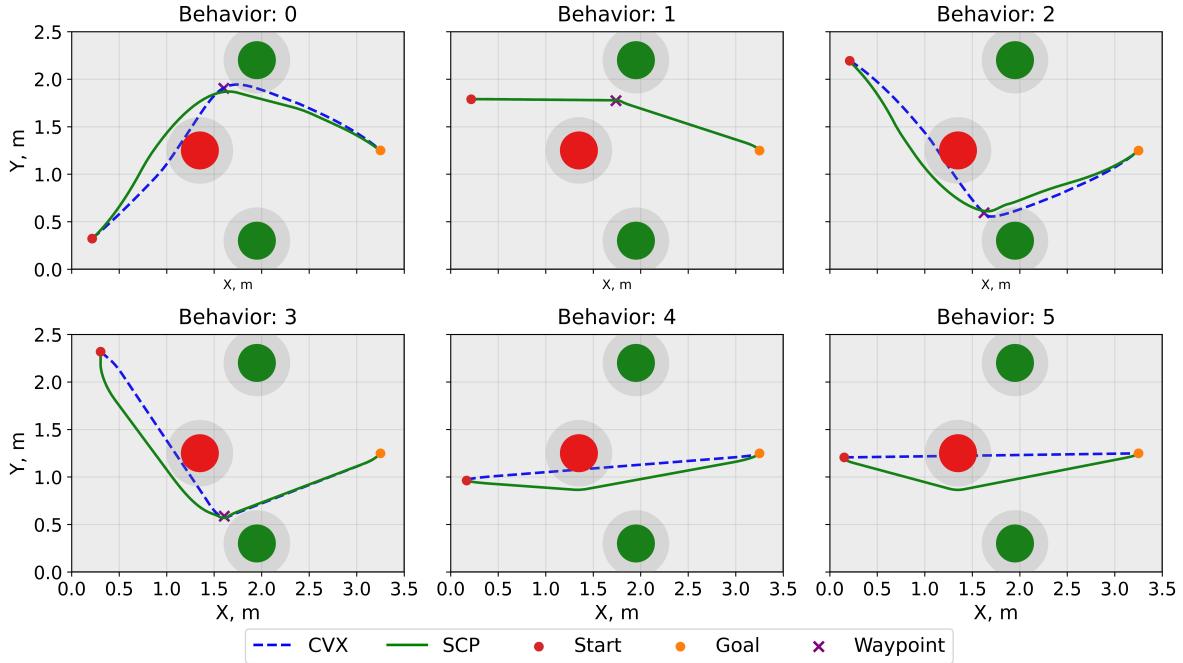$$\|\boldsymbol{p}_{k_{\text{wyp}}} - \boldsymbol{p}_{\text{wyp}}\|_2 \leq r_{\text{wyp}}, \tag{10e}$$

$$0 \leq \Lambda^{-1} \mathbf{R}_{GB}^{-1}(\psi_k) \boldsymbol{u}_k \leq \Delta V_{\max}, \qquad \forall k = 1, ..., N, \tag{10f}$$

$$\left\| \boldsymbol{p}_k - \boldsymbol{p}_O^{(i_b)} \right\|_2 - R_O^{(i_b)} \geq 0, \qquad \forall k = 1, ..., N, \ \forall i_b \in \{0, 1, 2\}. \tag{10g}$$

First, Eq. (10a) specifies the objective based on the impulsive-thruster control. Eqs. (10b) and (10c) define the system dynamics and boundary conditions, and the testbed workspace limits are specified in Eq. (10d). For behavior modes involving left- or right-passage, waypoint passage is enforced through two complementary mechanisms. First, Eq. (10e) imposes a *hard*, pointwise waypoint constraint, requiring the free-flyer's position to lie within a radius r_{wyp} of the sampled waypoint location (cf. Eq. (9)) at timestep k_{wyp}. Second, Eq. (10a) includes a *soft* waypoint-loitering penalty, defined as the cumulative squared displacement from the waypoint over $\mathcal{I}_{\text{wyp}} = \{k_{\text{wyp}} - 3, \ldots, k_{\text{wyp}} + 3\} \cap \{1, \ldots, N\}$, and weight $w = 10^3$. Whereas the hard constraint enforces passage at a specific time, the soft penalty encourages the vehicle to remain near the waypoint over a short interval surrounding the passage, facilitating the smooth maneuver near the ROI. Furthermore, Eq. (10f) denotes the nonconvex control input constraint. The control input vector in the global frame is given by $\boldsymbol{u} = \mathbf{R}_{GB}(\psi) \Lambda \Delta V \in \mathbb{R}^3$, where $\Delta V \in \mathbb{R}^8$ denotes the impulsive ΔVs applied by each thruster; $\Lambda \in \mathbb{R}^{3 \times 8}$ denotes the thruster configuration matrix; and $\mathbf{R}_{GB} \in \mathbb{R}^{3 \times 3}$ denotes the rotation matrix from the body-fixed frame to the global frame. The ΔVs applied by each thruster are physically limited to $0 \leq \Delta V \leq \Delta V_{\max} = T \Delta t / m$, where T and m denote the maximum thrust level and the system mass, respectively. Finally, Eq. (10g) represents the nonconvex collision avoidance constraint with respect to the three obstacles, which ensures the exclusion from the circular KOZ. Appendix B presents the problem-specific parameters used in the free-flyer scenario. Note that to generate "fast" behaviors, the terminal time is shortened from the nominal $t_f = 40$ s to $t_f^{\text{fast}} \in [0.65, 0.75] t_f$, i.e., the problem is solved under a reduced horizon $N_{\text{fast}} = \lfloor t_f^{\text{fast}} / \Delta t \rfloor$.

The trajectory dataset of the free-flyer scenario is generated by successively solving SCPs. First, given an initial state, transfer time, and waypoint condition (position and timestep), a relaxed nonconvex problem in Eq. (10) without

Fig. 4   2D-planar trajectories of the free-flyer corresponding to each behavior mode (left/right/central traverse + slow/fast, cf. Table 1).

Eq. (10g) is solved by the SCP, which generates a dynamically feasible trajectory that satisfies the waypoint passage but the collision avoidance constraints. This trajectory is then used as a warm-start for the second SCP that solves the full problem in Eq. (10). This two-stage SCP structure reduces the infeasibility in the SCP and ensures that the final trajectories satisfy both the behavior-conditioned waypoint and collision avoidance.

Based on the above formulation, sample trajectories demonstrating the behavior patterns defined in Table 1 are shown in Fig. 4.

3. Trajectory Optimization for Refinement after the Test-time Inference

After the warm-start is generated, the following optimization problem is solved for the trajectory refinement:

$$\min_{\{\boldsymbol{x}_k\}_{k=1}^{N}, \{\boldsymbol{u}_k\}_{k=1}^{N}} \quad (8a) \quad \text{subject to} \quad (10b), (10c), (10d), (10f), (10g). \tag{11a}$$

Note that no waypoint-passage constraint is applied during refinement. This step aims to project the warm-start into the safe and dynamically feasible region without compromising its high-level semantic intent. Enforcing additional optimality objectives at this point may introduce feasibility conflicts; omitting them allows the solver to produce a safe, intent-consistent trajectory.

B. Spacecraft Proximity Operation

The second case study considered in this paper is the spacecraft proximity operation scenario, in which a servicer spacecraft executes a complex maneuver around a target spacecraft. Due to the inherent difficulty of this maneuver, a conservative continuous-time passive safety constraint is imposed in the trajectory optimization, making strict constraint satisfaction particularly challenging. Additionally, the high-level text commands specify the time durations of the sub-tasks within each maneuver using explicit numerical values. Although this mixture of natural language and numerical information reflects realistic spacecraft operations, it introduces additional learning challenges compared to the free-flyer scenario. In particular, the model must accurately capture the temporal structure and coherence of each behavior, while also extracting the numerical quantities embedded in the text and associating them with the corresponding maneuver pattern.

The state vector $\boldsymbol{x} \in \mathbb{R}^6$ is represented in the form of Relative Orbital Elements (ROE) [22]. Let the Keplerian elements be denoted by $\boldsymbol{\alpha} = [a, e, i, \Omega, \omega, M]$, where a is the semi-major axis, e the eccentricity, i the inclination, Ω the right ascension of the ascending node, ω the argument of periapsis, and M the mean anomaly. The ROE are defined as specific differences between the orbital elements of the servicer (deputy) and the target, denoted by $\boldsymbol{\alpha}_d$ and $\boldsymbol{\alpha}$, respectively. In this study, the quasi-nonsingular ROE (qnsROE) formulation scaled by the chief's semimajor axis is adopted, expressed as [22]:

$$\boldsymbol{x}_{\text{qns}} = a \begin{bmatrix} \delta a \\ \delta \lambda \\ \delta e_x \\ \delta e_y \\ \delta i_x \\ \delta i_y \end{bmatrix} = a \begin{bmatrix} (a_d - a)/a \\ (M_d + \omega_d) - (M + \omega) + (\Omega_d - \Omega) \cos i \\ e_d \cos \omega_d - e \cos \omega \\ e_d \sin \omega_d - e \sin \omega \\ i_d - i \\ (\Omega_d - \Omega) \sin i \end{bmatrix}. \tag{12}$$

In parallel with the previous section, the subsequent subsections describe the scenario design, covering (1) the definition of high-level behavior patterns, (2) the nonconvex trajectory optimization used for dataset generation, and (3) the feasibility problem employed for final nonconvex refinement.

1. Behavior Patterns and Model Conditioning

Six behavior modes are considered to describe the maneuver of the servicer. The servicer spacecraft begins its maneuver from an initial along-track offset of approximately $\delta \lambda = -120$ m (the -V-bar direction) with a relative orbit state that exhibits no Radial/Normal (RN)-plane separation. From this starting configuration, the servicer performs various behaviors such as circumnavigation or docking. Furthermore, this case study does not involve the conditioning based on the goal state; namely, the trained model is desired to define its (terminal) waypoints as the result of the rollout.

To realize these behaviors within the dataset, waypoints are strategically selected, as summarized in Table 2. The notions of fast and slow maneuvers are encoded directly through numerical parameters that appear in the command text. Since each command may describe a sequence of actions, interpreting the semantics and executing a safe maneuver becomes more challenging.

Inclusion of the numerical values in the command sentence increases additional complexity in the annotation process for the dataset. To efficiently characterize trajectories and label each with an appropriate quantitative description, a collection of pre-generated command templates is first prepared, each capable of accepting a set of behavior-dependent variables. After generating the dataset trajectories, a template corresponding to the desired behavior mode is randomly drawn from this pool, and quantitative values, such as the terminal along-track separation ($\delta \lambda$) or waypoint epochs, are substituted to form a trajectory-specific command. Since numerical values are inserted based on the specific trajectory, each resulting command becomes unique as a complete sentence. The variables associated with each behavior mode, along with illustrative command templates, are summarized in Table 3. To promote linguistic diversity, a given command is not required to include all available variables. During the auto-generation of text command, each sentence is set to have fewer than 23 words, and the maximum text-embedding length is set to $N_e = 50$.

Semantic correctness is assessed by verifying whether a generated trajectory reaches the prescribed set of waypoints in Table 2, once the quantitative parameters within the command are instantiated. Three classes of values are evaluated: (i) the waypoint-passing epoch, (ii) the commanded along-track separation, and (iii) the terminal state. For commands specifying a time index (e.g., `T_appr_orbits`), the value is converted to the closest discrete timestep, denoted by k_{wyp}. At this index, element-wise consistency with the waypoint across the qnsROE space is checked using:

$$|\boldsymbol{x}_{k_{\text{wyp}},j} - \boldsymbol{x}_{\text{wyp},j}| \le q |\Delta \boldsymbol{x}_j|, \quad j = 1, ..., 6, \tag{13}$$

where $\boldsymbol{x}_{\text{wyp}}$ is the nominal waypoint and $\Delta \boldsymbol{x}$ is the maximum admissible deviation used during waypoint generation, as summarized in Table 2. The hyperparameter $q > 0$ adjusts the tolerance of the criterion; $q = 1$ corresponds to the requirement that the generated trajectory falls within the same tolerance bounds used to construct the training dataset. In Table 2, some components of the waypoint are strictly fixed; for example, the relative semimajor-axis δa of all waypoints are set to zero to avoid the along-track drift. For these components, when evaluating the waypoint set inclusion, an additional margin of $\Delta \boldsymbol{x}_j = \pm 2$m is added to each component. The same condition as in Eq. (13) is applied to the terminal state to determine whether the trajectory converges to the designed terminal set, although this terminal requirement is not explicitly commanded in the text input. Finally, for commands that specify an along-track

Behavior	Waypoints	
	timestep, k	$\boldsymbol{x}_{\mathrm{qns}} \pm \Delta \boldsymbol{x}_{\mathrm{qns}}[\mathrm{m}]$
(initial state)	1	$[0, -120 \pm 20, 0 \pm 4, 5 \pm 4, 0 \pm 4, 5 \pm 4]$
0: Approach to the target and circumnavigate	[31, 49]	$[0, 0 \pm 5, 0 \pm 2, 32 \pm 2, 0 \pm 2, 32 \pm 2]$
	50	$[0, 0 \pm 5, 5 \pm 2, 32 \pm 2, 5 \pm 2, 32 \pm 2]$
1: Go to -35 m V-bar and hold (dock)	[31, 49]	$[0, -35 \pm 5, 0 \pm 2, 0 \pm 2, 0 \pm 2, 0 \pm 2]$
	50	$[0, -35 \pm 5, 0 \pm 2, 0 \pm 2, 0 \pm 2, 0 \pm 2]$
2: Fast flyby under KOZ, from -V-bar to +V-bar	[35, 49]	$[0, 150 \pm 2, 0 \pm 2, 5 \pm 2, 0 \pm 2, 5 \pm 2]$
	50	$[0, 150 \pm 2, 0 \pm 2, 5 \pm 4, 0 \pm 2, 5 \pm 2]$
3: Flyby with RN (Radial-Normal)-plane separation, from -V-bar to +V-bar	[6, 15]	$[0, -120 \pm 20, 0 \pm 2, 25 \pm 2, 0 \pm 2, 25 \pm 2]$
	[36, 45]	$[0, 120 \pm 20, 0, 25 \pm 2, 0, 25 \pm 2]$
	50	$[0, 120 \pm 10, 0, 25 \pm 2, 0, 25 \pm 2]$
4: Approach, circumnavigate, then escape along +V-bar	[21, 25]	$[0, 0, 0, 32 \pm 2, 0, 32 \pm 2]$
	[35, 39]	$[0, 0, 0, 32 \pm 2, 0, 32 \pm 2]$
	50	$[0, -120 \pm 10, 0 \pm 2, 35 \pm 2, 0 \pm 2, 35 \pm 2]$
5: Approach, circumnavigate, then escape along -V-bar	[21, 25]	$[0, 0, 0, 32 \pm 2, 0, 32 \pm 2]$
	[35, 39]	$[0, 0, 0, 32 \pm 2, 0, 32 \pm 2]$
	50	$[0, 120 \pm 10, 0 \pm 2, 35 \pm 2, 0 \pm 2, 35 \pm 2]$

Table 2 Definition of waypoints for each behavior mode in the spacecraft proximity operation scenario. The timestep of the waypoint passage is randomly sampled from the presented range. The waypoint value $z = x \pm \Delta x$ denotes random sampling in $x - \Delta x \le z \le x + \Delta x$, discretized into 21 values.

Behavior	Variables	Example Command Template
0	`T_appr_orbits`	Execute spiral approach to target for {T_appr_orbits} orbits; thereafter, circumnavigate while ensuring RN-plane separation.
1	`T_appr_orbits, d_lambda_meters`	Use {T_appr_orbits} orbits to close to {d_lambda_meters} m along -V-bar; remain there until instructed otherwise.
2	`T_appr_orbits, d_lambda_meters`	Execute a rapid underfly trajectory to +V-bar at {d_lambda_meters} m in {T_appr_orbits} orbits with focus on RT-plane safety.
3	`T_EI_sep_orbits, T_transfer_orbits`	Flyby operation; establish E/I-vector separation in {T_EI_sep_orbits} orbits; execute delta-a until {T_transfer_orbits} orbits; hold at -V-bar.
4	`T_appr_orbits, T_circ_orbits`	Commence -V-bar approach for {T_appr_orbits} orbits; circumnavigate until {T_circ_orbits} orbits; maneuver to +V-bar afterwards.
5	`T_appr_orbits, T_circ_orbits`	Initiate approach from -V-bar for {T_appr_orbits} orbits; circumnavigate until {T_circ_orbits} orbits; proceed back to -V-bar.

Table 3 Behavior modes and sample text templates defined in the spacecraft proximity operation scenario.

separation quantity (i.e., `d_lambda_meters`), expressed through the relative mean longitude $\delta\lambda$, the correctness is evaluated analogously:

$$\left| \delta\lambda_{k_{\mathrm{wyp}}} - \delta\lambda_{\mathrm{wyp}} \right| \le q \left| \Delta\delta\lambda \right|, \tag{14}$$

where $\delta\lambda_{\mathrm{wyp}}$ and $\Delta\delta\lambda$ denote the nominal relative mean longitude and its allowable tolerance, respectively.

In summary, a trajectory is deemed *semantically correct* if and only if both of the following conditions are satisfied:
1) **Terminal-state consistency**: the final qnsROE state lies within the admissible terminal waypoint set, i.e., it satisfies the tolerance condition in (13) when evaluated at the terminal index.
2) **Waypoint or along-track consistency**: at the commanded epoch k_{wyp}, the trajectory satisfies the corresponding inclusion condition for either (i) the prescribed waypoint or (ii) the commanded relative mean longitude $\delta\lambda$, depending on which quantity is explicitly referenced in the natural-language command.

These requirements ensure that semantic correctness accounts for both the long-horizon behavior of the optimized trajectory and the quantitative constraints specified in the text instruction.

2. Trajectory Optimization for Dataset Generation

To generate a set of trajectories that complies with the safety constraint, a discrete-time nonconvex optimal control problem for fault-tolerant spacecraft proximity operation is formulated as follows:

$$\min_{\{x_k\}_{k=1}^N, \{u_k\}_{k=1}^N} \sum_{k=1}^N \|u_k\|_2, \tag{15a}$$

$$\text{subject to} \quad x_{k+1} = \Phi(t_{k+1}, t_k)(x_k + \Gamma_k u_k), \qquad \forall k = 1, ..., N-1, \tag{15b}$$

$$g(t_k; \tau, \alpha) := 1 - x(t_k; \tau, \alpha)^\top S_{k\tau} x(t_k; \tau, \alpha) \leq 0 \qquad \forall k = 1, ..., N, \forall (\tau, \alpha) \in [0, \tau^s] \times [0, 1] \tag{15c}$$

$$x_1 = x_i, \quad x_N + \Gamma_N u_N = x_f, \tag{15d}$$

$$x_{k_{\text{wyp}}} = x_{\text{wyp}}, \qquad \forall k_{\text{wyp}} \in K_{\text{wyp}} \tag{15e}$$

$$\text{where} \quad x(t_k; \tau, \alpha) = x(t_k; 0, \alpha) + \int_0^\tau f(\gamma, x(t_k; \gamma), 0) d\gamma = \Phi(t_k + \tau, t_k) x(t_k; 0, \alpha), \tag{15f}$$

$$x(t_k; 0, \alpha) = x_k + \alpha \Gamma_k u_k, \tag{15g}$$

$$S_{k\tau} = \Psi(t_k + \tau)^\top P \Psi(t_k + \tau). \tag{15h}$$

The objective of the optimization problem is to minimize the total control cost, modeled as the sum of impulsive maneuver magnitudes. The discrete-time qnsROE dynamics, subject to impulsive velocity changes $u_k \in \mathbb{R}^3$ applied in the Radial–Tangential–Normal (RTN) frame, are presented in Eq. (15b). The State Transition Matrix (STM) $\Phi(t_{k+1}, t_k) \in \mathbb{R}^{6\times6}$ and the control input matrix $\Gamma_k = \Gamma(t_k) \in \mathbb{R}^{6\times3}$ are analytical functions of the target's orbital elements, independent of the relative state itself [22], allowing the dynamics to be expressed as a Linear Time-Varying (LTV) system. An STM model that incorporates the secular J_2 perturbation is adopted in this work [43], and the first-order control matrix Γ_k follows the formulation in D'Amico [22]. The boundary conditions and waypoint passage are imposed through Eqs. (15d) and (15e), respectively.

The most critical nonconvex constraint in this problem is the continuous-time passive safety under the contingency of partially imperfect burns, expressed in Eq. (15c). This constraint ensures that the deputy remains outside the ellipsoidal KOZ around the target, not only throughout the controlled interval $t \in [0, t_f]$ but also during a drift interval $\tau \in [0, \tau^s]$ following an unexpected loss of control authority at the post-burn state $x(t_k; 0, \alpha)$. Passive safety must be guaranteed (i) continuously in time, rather than at discrete nodes, and (ii) even under incomplete burn execution, characterized by the burn completion factor $\alpha \in [0, 1]$. Despite its conservatism, enforcing passive safety under imperfect burns is justified, as the objective is to enable complex maneuvers around the target where the geometric safety criteria [22] cannot always be guaranteed. The geometry of each KOZ is specified by a diagonal matrix $P \in \mathbb{R}^{6\times6}$, which defines the ellipsoidal semi-axes in the RTN frame, and matrix $\Psi_k = \Psi(t_k) \in \mathbb{R}^{6\times6}$ denotes the first-order linear mapping from qnsROE to the relative Cartesian state in the RTN frame, denoted as x_{RTN} [22]. Note that stochastic uncertainties due to actuation misalignment, navigation error, or unmodeled accelerations are not explicitly modeled in the chance-constrained framework; only the contingency arising from imperfect impulsive burns is considered in this problem. This simplification is justified because the high relative navigation accuracy achieved in close-proximity operations (<cm-level via differential GNSS [44]), and unmodeled accelerations are negligible over the few orbital revolutions considered. Instead, the emphasis is placed on ensuring continuous-time satisfaction of the passive safety constraint and robust exclusion from the KOZ under the presence of partially imperfect burns, which dominate the overall safety behavior.

The continuous-time passive safety constraint for an impulsive control at t_k is enforced by satisfying the following integral inequality [21]:

$$g(t_k; \tau, \alpha) \leq 0, \ \forall (\tau, \alpha) \in [0, \tau^s] \times [0, 1] \quad \Leftrightarrow \quad \tilde{g}(x_k, u_k) := \int_0^1 \int_0^{\tau^s} |g(t_k; \tau, \alpha)|_+^2 \, d\tau \, d\alpha \leq 0, \tag{16}$$

where $|x|_+ = \max(x, 0)$ denotes the hinge function. The convexified constraint of Eq. (16) using the reference variables
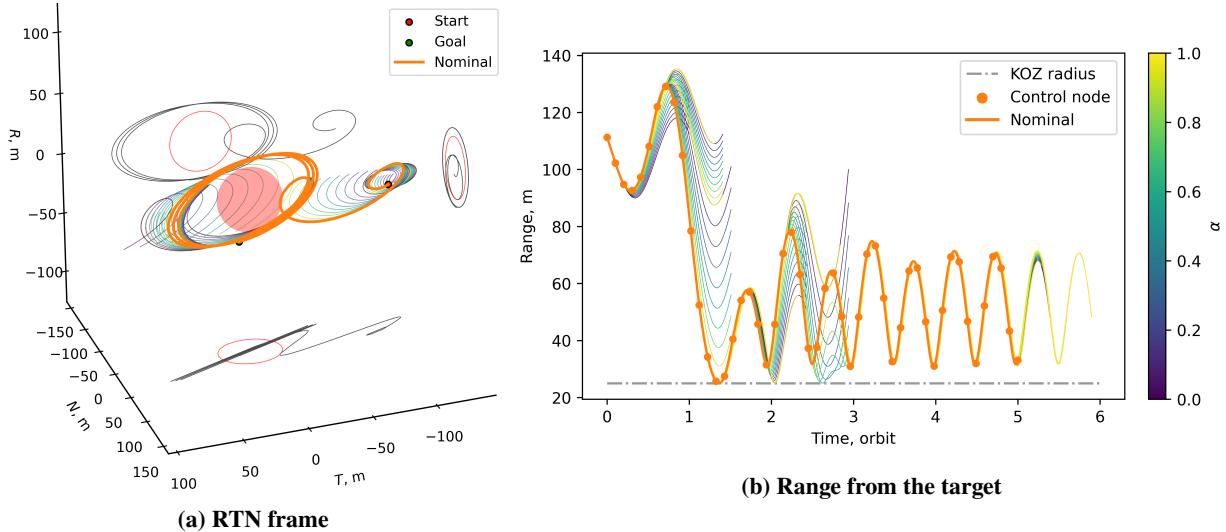
$(\bar{x}_k, \bar{u}_k)$ is obtained as:

$$G_k^x(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k) + G_k^u(\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k) + \tilde{g}_k(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) = 0, \tag{17a}$$

$$G_k^x := \left. \frac{\partial \tilde{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)}{\partial \boldsymbol{x}_k} \right|_{(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)} = -4 \int_0^1 \int_0^{\tau^s} |\bar{g}(t_k; \tau, \alpha)|_+ \Phi(t_k + \tau, t_k)^\top S_{k\tau} \Phi(t_k + \tau, t_k) \bar{\boldsymbol{x}}(t_k; 0, \alpha) \, d\tau \, d\alpha, \tag{17b}$$

$$G_k^u := \left. \frac{\partial \tilde{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)}{\partial \boldsymbol{u}_k} \right|_{(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)} = -4 \int_0^1 \int_0^{\tau^s} |\bar{g}(t_k; \tau, \alpha)|_+ \Gamma_k^\top \Phi(t_k + \tau, t_k)^\top S_{k\tau} \Phi(t_k + \tau; t_k) \bar{\boldsymbol{x}}(t_k; 0, \alpha) \, d\tau \, d\alpha, \tag{17c}$$

where $\bar{g}(t_k; \tau, \alpha)$ is evaluated using the reference variables. While the original theoretical framework enforces continuous-time safety via Ordinary Differential Equation (ODE) integration to evaluate Eq. (17a) [21], the specific case of impulsive control under LTV dynamics admits a more efficient solution. Leveraging the analytical STM, this paper replaces numerical propagation with a Gauss-Legendre quadrature rule to evaluate the constraint in Eq. (15c) directly. The details of the constraint formulation and its convexification are provided in Appendix C. An example optimized trajectory is shown in Fig. 5; particularly, Fig. 5b illustrates the range between the servicer and the target, where the exclusion from the spherical KOZ is satisfied not only node-wise but also at the level of both nominal and drift continuous trajectories.

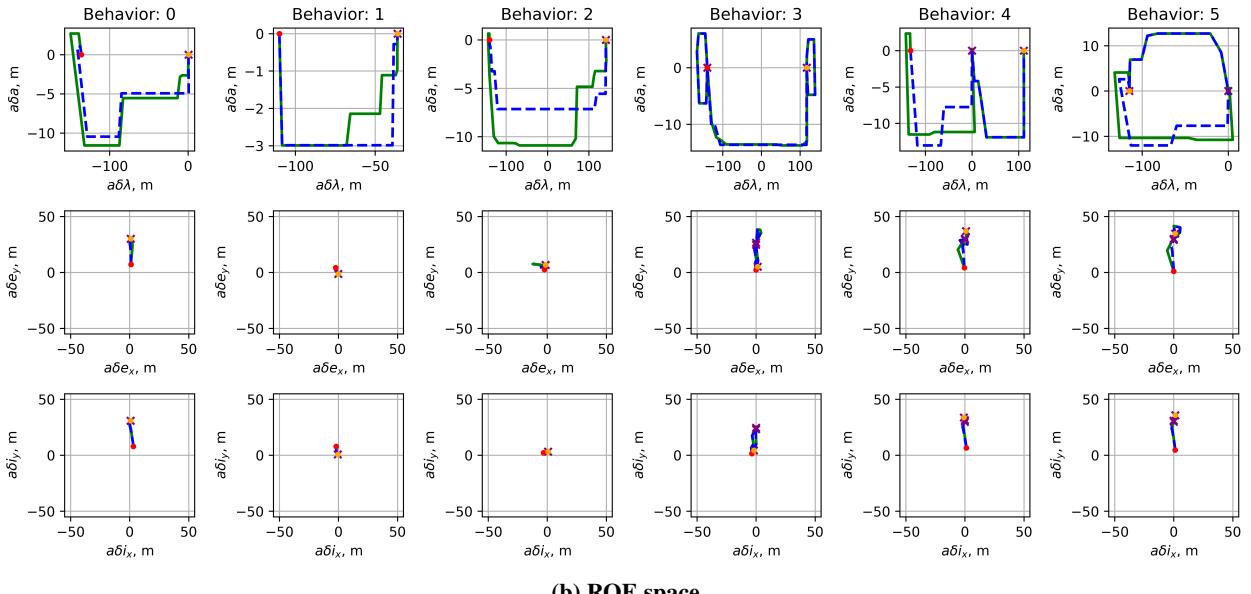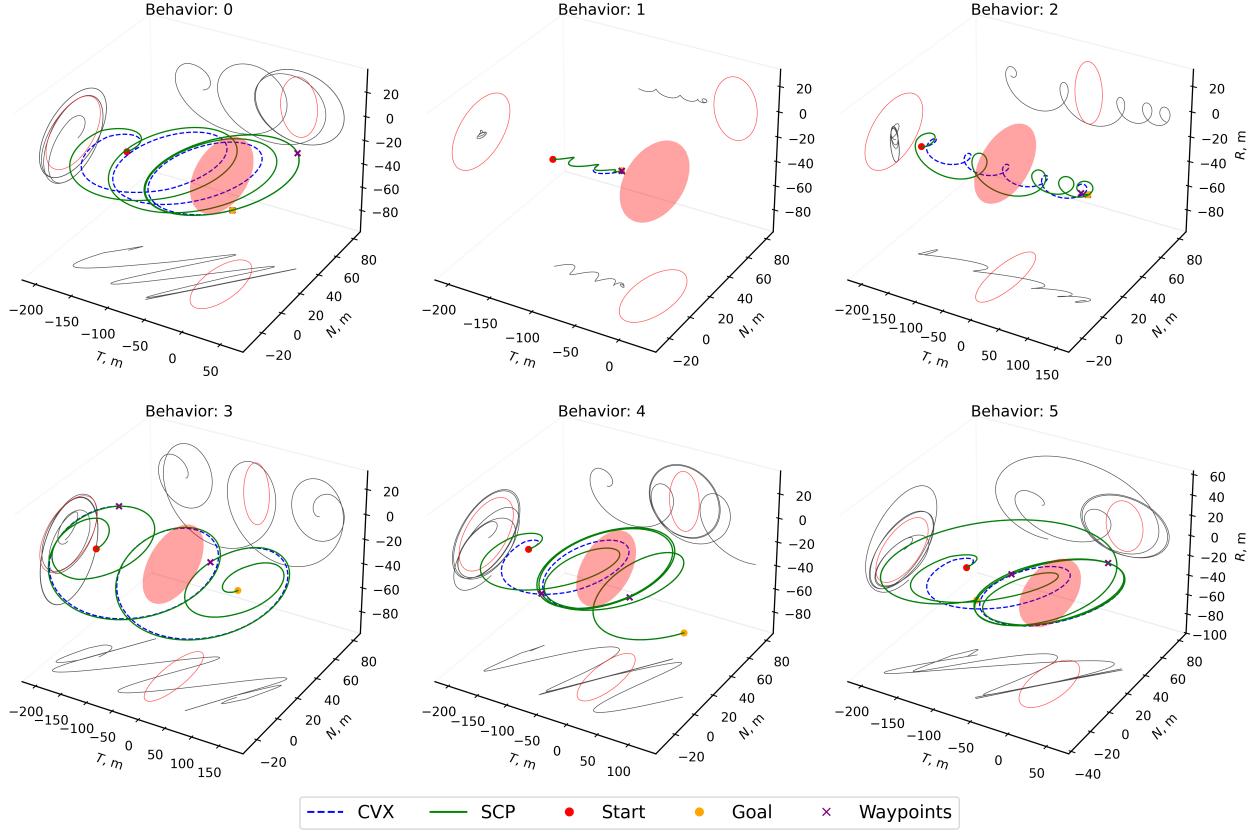(a) RTN frame

(b) Range from the target

Fig. 5 Example of a one-orbit continuous-time passive safety with respect to the KOZ (25m sphere) under partially imperfect burns. The color of the drift trajectories corresponds to the completion rate of the burn $\alpha \in [0, 1]$.

This case study examines a fixed-time trajectory optimization problem under a fixed orbital configuration. The transfer duration is set to $t_f = 5$ orbits, and the target spacecraft's initial orbital elements are fixed at $\boldsymbol{\alpha}(t_i) = [6738.14 \text{ km}, \ 5.58 \times 10^{-4}, \ 51.64°, \ 301.04°, \ 26.18°, \ 68.23°]$, corresponding to a near-circular low Earth orbit. The drift time for passive safety is set to $\tau^s = 1$ orbit. The KOZ is modeled as a sphere of 25 m radius, and the trajectory is uniformly discretized into $N = 50$ nodes. The maximum control magnitude constraint is omitted because the scenarios considered in the following subsection require only modest maneuvering effort (typically $\Delta V \leq 100 \text{ mm/s}$). With ten control nodes per orbit, the impulse per timestep remains well within standard thruster capability, as the required cumulative ΔV can be realized through a sequence of sufficiently small burns.

The trajectory optimization is solved using $\texttt{SCVx}^*$ [13], an augmented Lagrangian-based SCP algorithm with auto-tuning of the trust region and penalty weights. The hyperparameters are provided in Appendix D. During the SCP routine, all variables are normalized with respect to the initial guess, thereby scaling each element to the range $[-1, 1]$ to enhance numerical stability [6]. For the generation of training data sets, the solution of the convex relaxation problem (i.e., Eq. (15) without Eq. (15c)), which is the convex waypoying-hopping trajectory, is solved and used as a warm-start of the SCP.

With the boundary conditions defined in Table 2, representative trajectories provided by solving the above nonconvex trajectory optimization for each behavior mode are illustrated in both the RTN frame and the qnsROE space in Fig. 6.

12

(a) RTN frame

(b) ROE space

Fig. 6 Example trajectories corresponding to the behavior modes defined in the spacecraft proximity operations scenario (cf. Tables 2 and 3).

For comparison, the solution of convex relaxation that omits the continuous-time passive safety constraint in Eq. (15c) (denoted as convex waypoint-hopping, or CVX) is also shown in each subfigure.

3. Trajectory Optimization for Refinement after the Test-time Inference

Similar to the one in the free-flyer scenario, the following optimization problem is solved for the trajectory refinement after the warm-start trajectory is generated:

$$\min_{\{\boldsymbol{x}_k\}_{k=1}^N, \{\boldsymbol{u}_k\}_{k=1}^N} \text{(8a)} \quad \text{subject to} \quad \text{(15b), (15c), (15d),} \tag{18a}$$

where $\lambda = 0.1$ is defined as a weighting hyperparameter in the objective in Eq. (8a). Note again that the terminal waypoint is determined based on only a text command, so the terminal state constraint is defined using the one from the warm-start trajectory generated from the transformer. Furthermore, the waypoint passage constraint in Eq. (15e) is not enforced in the problem, as it is assumed that well-educated waypoint knowledge is not available during the deployment.

V. Results

This section evaluates the performance of the proposed SAGES framework through a series of numerical experiments. The proposed framework is designed to enable text-based commanding for future spacecraft autonomy. To assess its effectiveness, two Research Questions (RQs) are defined: **(RQ1) Do the trajectories generated by the transformer exhibit favorable performance in terms of safety and semantic correctness?**, **(RQ2) Given the high-quality warm-starts from the transformer, does SAGES generate safe and semantically-adherent trajectories with a high success rate?**. The remainder of this section addresses these research questions to examine the performance and generalization capability of the proposed language-driven safe trajectory generation pipeline.

For the frozen text encoder, `distilbert-base-uncased` model [45] is adopted. The hyperparameters of the transformer are summarized in Appendix E. A total of $N_{\text{data}} \approx 80,000$ and $90,000$ command-trajectory pairs are used for the training of the free-flyer and spacecraft proximity operation scenario, respectively (cf. see Appendix A for the details of the dataset generation). For the test data used in the follwoing sections, a separate set of (unseen) 1,000 command-trajectory pairs is randomly sampled.

A. Performance of Warm-Start Trajectories

The statistical properties of the warm-start trajectories produced by the transformer (SAGES-WS) are first analyzed. This subsection addresses two key questions related to (RQ1): **(i) To what extent do the generated warm-start trajectories satisfy the prescribed semantic specifications? (ii) Relative to the convex baseline, do the generated trajectories exhibit improved safety, reflecting a better understanding of the nonconvex constraint landscape?** To answer these questions, each generated trajectory is evaluated along semantic compliance and violations of nonconvex safety constraints. For comparison, the convex-based benchmark is also provided. In the free-flyer scenario, the CVX stage solves the SCP problem without enforcing the nonconvex collision avoidance constraint in Eq. (10g); instead, semantic compliance is induced through the designed waypoint, which shapes the resulting trajectory. In contrast, the spacecraft proximity operation scenario solves the convex waypoint-hopping constraint as CVX; trajectories generated by CVX are always semantically correct through the definition of the designed waypoint.

The robustness to different natural-language inputs is evaluated using two command sets:
- Seen command (template) with unseen scenario: the initial conditions and goals are in-distribution but unseen, while the commands match those used in training, each pair of text and the initial state still provides an unseen scenario to the model. When using the command templates, the sentence string itself also becomes unseen by the model with the added numerical values.
- Unseen command (template) with unseen scenario: both the initial conditions and the text inputs are unseen during training.

The results of the free-flyer and spacecraft proximity operation scenarios are summarized in Tables 4 and 5, respectively. In these tables, the reported success rate corresponds to the semantic compliance associated with each text command, as defined in the previous section. Furthermore, Fig. 7 presents the distribution of the cumulative constraint violation (cf. Eq. (3b)) of the generated trajectories based on the unseen commands and command templates, denoted as c_1.

The free-flyer scenario examines the trajectory generation both with seen and unseen commands in the dataset, along with unseen problem scenarios (i.e., initial state and waypoint position). The results show that trajectories generated by the transformer exhibit strong semantic adherence across all behavior modes. For several difficult initializations, particularly when the free-flyer's starting position lies along the Line-of-Sight (LOS) to the goal or to the added waypoint, the CVX warm-start often becomes infeasible, with the issue appearing most prominently in Behaviors 4 and

Method	Metric		Behavior					
	Command	Type	0	1	2	3	4	5
CVX		Safety [%]	46.15	46.99	37.67	48.55	36.57	40.34
		Semantic correctness [%]	100	100	100	100	100	100
SAGES-WS	Seen command	Safety [%]	69.93	83.61	66.44	82.08	52.0	59.66
		Semantic Correctness [%]	93.7	100	95.2	100	99.4	100
	Unseen command	Safety [%]	59.15	78.14	71.83	84.97	60.0	64.20
		Semantic Correctness [%]	94.4	100	95.8	100	99.4	100
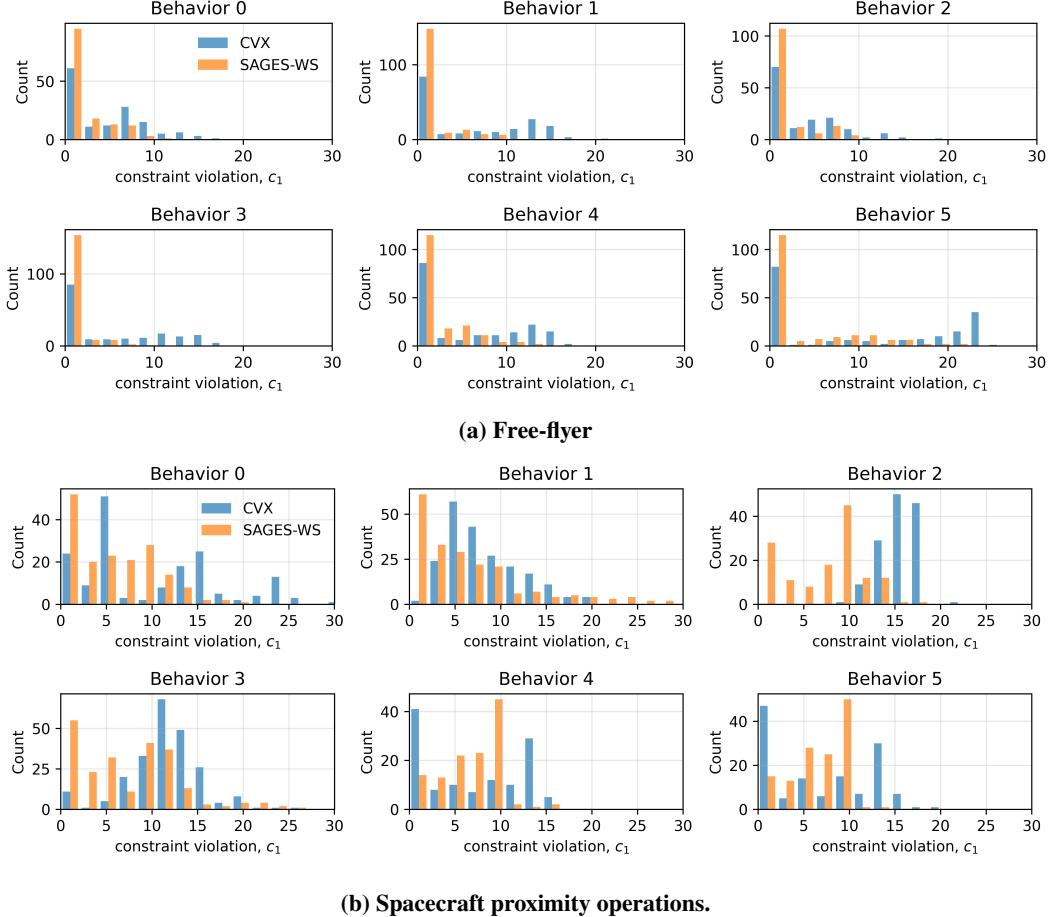
Table 4   Summary of the warm-start success rates and safety rates for free-flyers.

Method	Metric			Behavior					
	Command	Type	q	0	1	2	3	4	5
CVX		Safety [%]		0.0	0.5	0.6	2.3	0.0	0.0
		Semantic correctness [%]	1	100	100	100	100	100	100
SAGES-WS	Seen template	Safety [%]		25.9	23.2	13.0	21.9	7.1	6.9
		Semantic Correctness [%]	1	89.51	63.51	99.24	17.21	77.17	72.25
			2	100	92.42	99.24	42.33	91.34	88.44
			3	100	100	99.24	66.51	96.85	92.49
	Unseen template	Safety [%]		20.5	16.8	17.5	22.5	7.5	8.9
		Semantic Correctness [%]	1	89.74	63.27	99.35	12.44	50.93	73.39
			2	100	94.39	99.35	36.36	88.20	92.74
			3	100	100	99.35	62.68	91.93	95.97

Table 5   Summary of the success rates and safety rates (spacecraft proximity operations). The results of CVX use the terminal state of the trajectory generated by the transformer (SAGES-WS) under the text command with seen templates. q denotes the tolerance factors for the waypoint domain (cf. Eqs. (13) and (14).)

5 (cf. Fig. 7a). In contrast, the trajectories of SAGES-WS have a significantly higher likelihood of producing safe and semantically consistent trajectories without requiring additional refinement. Also, according to Fig. 7a, the transformer produces consistently lower violation magnitudes across all Behaviors, with a large mass concentrated at zero, indicating that the majority of warm-start trajectories are already achieving collision avoidance.

In the spacecraft proximity operations scenario, under challenging continuous-time passive safety conditions with imperfect burns, the convex waypoint-hopping solutions (CVX) rarely satisfy the safety constraint, even when the waypoints themselves are passively safe. In contrast, SAGES-WS trajectories generated by the transformer improve the likelihood of producing solutions that are already passively safe without additional refinement up to 25% (Behavior 0), while maintaining quantitative semantic correctness when a moderate waypoint tolerance (e.g., $q = 2$ or 3) is allowed. Table 5 further shows that performance is comparable under both seen and unseen command templates. This indicates that the model does not overfit to particular textual patterns; instead, it encodes the underlying semantic content, including numerical quantities, within the latent space of the text encoder. Figure 7b highlights this trend in the distribution of constraint violations. For Behaviors 0–2, SAGES-WS lowers the violation distribution while preserving semantic correctness, with most trajectories remaining within the waypoint set for $q \leq 2$. A trade-off appears for Behaviors 3–5, where the model shifts the violation distribution toward lower values but achieves reduced semantic-correctness rates. In particular, Behavior 3 yields lower semantic correctness for the SAGES-WS trajectory compared with the other behavior modes. This occurs because the qnsROE changes rapidly near this waypoint in the relative semi-major axis δa, as illustrated in Fig. 6b. Consequently, the waypoint set becomes too small at the desired passage epoch, increasing the likelihood that the resulting trajectory is classified as semantically incorrect. Conversely, for Behaviors 4 and 5, high-level commands are consistently satisfied, although the reduction in violation magnitude is less pronounced; nonetheless, the number of trajectories with large violations (e.g., $c_1 > 10$) is significantly reduced. This trade-off is consistent with observations in prior work, where tension often arises between optimizing a reward signal (here reflected in semantic correctness) and enforcing hard constraints [36], and it continues to appear even
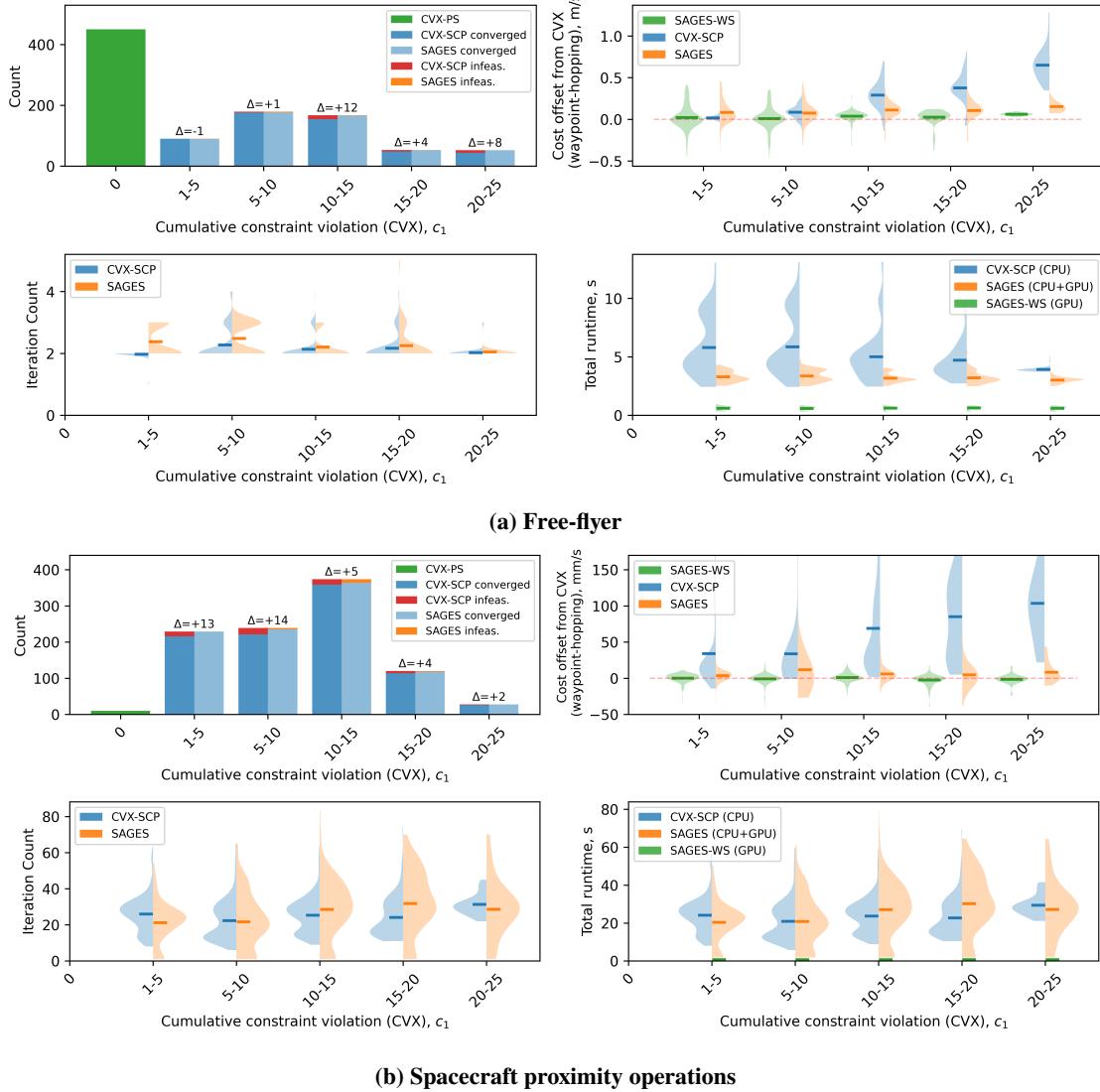
(a) Free-flyer

(b) Spacecraft proximity operations.

Fig. 7 **Constraint violations of the warm-starts across different behavior modes compared to the solutions of the convex relaxation.**

in multi-objective formulations [40]. Overall, the results show that the proposed framework captures the intended semantics of the command and generates trajectories that exhibit more favorable safety and correctness properties than benchmark waypoint-hopping trajectories.

B. Performance of Converged solutions in Sequential Convex Programming

In this subsection, the performance of SAGES is evaluated based on the refined trajectory after the SCP. To address (RQ2), two key questions of interest are: **(i) Does the usage of the trajectories generated by SAGES provide an advantageous property in the algorithmic performance of the SCP?**, and **(ii) Are the converged solutions in the SCP not only safe but also compliant with the high-level command?** The generated warm-starts are used as an initial guess of the SCP that converges to the solution that satisfies the safety constraints, as defined in Eqs. (11) and (18), respectively. The solutions are then compared against the benchmark solutions based on the initialization with the convex waypoint-hopping solutions.

First, the algorithmic performance of the SCP with a warm-start provided by SAGES is compared against the CVX initialization. Figure 8 summarizes four statistical metrics across: statistical probability of convergence in the SCP, converged objective value (as the cost offset from the convex waypoint-hopping solution), iteration counts of the SCP until the convergence, and the total runtime, which is the sum of the SCP routine and the warm-starting. These values for both the free-flyer and spacecraft proximity operations scenario are summarized as a histogram and a series of violin plots. Some problem scenarios satisfy the nonconvex constraint by solving a convex waypoint-hopping solution (i.e., $c_1=0$, shown as a green var in the top-left plot in Fig. 8), and these are excluded from the statistical analysis; the illustrated distribution of the objective's optimality, iteration count of SCP, and the total runtime is collected from

(a) Free-flyer

(b) Spacecraft proximity operations

Fig. 8 Performance analysis of the SCP warm-started with the SAGES-WS compared to CVX initialization, tested for 1,000 scenarios. The abscissa is binned across the cumulative constraint violation of the CVX solution.

the problem scenario in which the SCP converges in the respective warm-starting methods. Furthermore, the central question for SAGES is whether the SCP-refined trajectories remain semantically consistent with the high-level textual commands. To evaluate this, Tables 6 and 7 report both the SCP convergence rate (for warm-starting using CVX and SAGES-WS) and the semantic correctness of the resulting solutions. For comparison, the tables also include a fuel-optimal solution (cf. Eq. (15a)) initialized with the SAGES-WS.

The statistical performance of the Free-flyer scenario is evaluated on an NVIDIA GeForce RTX 3080[†]. All results use unseen text commands paired with unseen problem scenarios. Figure 8a shows that trajectories generated by SAGES consistently exhibit lower fuel consumption than the CVX-SCP baselines. The advantage is most pronounced in scenarios where the CVX initialization incurs significant constraint violations due to challenging waypoint geometries that make passive safety difficult to satisfy. In such cases, the transformer produces semantically consistent trajectories that are already close to passive safety, requiring only modest corrections during the subsequent SCP stage. As a result, the converged objective is typically lower for SAGES, since the CVX warm-start must undergo large corrective updates to achieve feasibility, which often leads to less favorable convergence behavior. Furthermore, SAGES exhibits a

---

[†]NVIDIA AGX Orin 64 GB is used for the hardware experiments on the testbed, as discussed in Sec. V.C.

Method	Metric		Behavior					
	Command	Type	0	1	2	3	4	5
CVX-SCP	Feasibility ($\lambda = 1$)	SCP Convergence (Safety) [%]	93.84	96.72	94.56	100	96.0	95.45
		Semantic correctness [%]	95.8	96.7	97.2	99.4	96.0	100
SAGES	Feasibility ($\lambda = 1$)	SCP Convergence (Safety) [%]	99.32	100	98.64	100	98.86	100
		Semantic Correctness [%]	95.1	100	97.9	100	98.9	100
	Fuel Optimality	SCP Convergence (Safety) [%]	99.32	100	98.64	100	98.86	100
		Semantic Correctness [%]	0.0	13.7	0.0	16.8	0.0	100

Table 6   Summary of the performance of the converged trajectories in SCP (unseen commands, free-flyers).

Method	Metric			Behavior					
	Command	Type	q	0	1	2	3	4	5
CVX-SCP	Feasibility ($\lambda = 0.1$)	SCP Convergence (Safety) [%]		90.38	91.84	94.16	85.17	95.03	85.48
		Semantic Correctness [%]	1	30.77	28.57	27.92	0.00	38.51	58.06
			2	62.18	60.20	58.44	5.74	66.46	70.16
			3	71.15	71.94	67.53	20.10	75.16	75.00
SAGES	Feasibility ($\lambda = 0.1$)	SCP convergence [%]		100	93.88	96.10	100	100	100
		Semantic Correctness [%]	1	89.74	62.76	95.45	12.44	50.93	73.39
			2	100	89.80	95.45	36.36	87.58	92.74
			3	100	92.35	95.45	62.20	91.93	95.97
	Fuel Optimality	SCP convergence (Safety) [%]		78.21	85.20	85.06	65.07	77.64	48.39
		Semantic Correctness [%]	1	26.28	35.20	51.95	0.00	0.00	0.00
			2	39.10	65.31	68.18	0.00	0.00	0.00
			3	49.36	78.57	80.52	0.00	0.00	0.00

Table 7   Summary of the performance of the converged trajectories in SCP (unseen command templates, spacecraft proximity operation). q **denotes the tolerance factors for the waypoint domain (cf. Eqs. (13) and (14)).**

significantly shorter total runtime compared to the CVX-SCP, despite the similar iteration counts in the SCP, which is almost always less than four iterations. This overhead in the total runtime arises because the CVX baseline must solve an additional SCP to construct its warm-start, adding roughly 5–7 s per instance. In contrast, the inference time by the transformer model is effectively negligible. Table 6 further shows that, with the feasibility-based projector, SAGES yields a higher rate of safe and convergent solutions while maintaining strong semantic alignment with the input text. The fuel-optimality variant also produces safe trajectories but places less emphasis on semantic fidelity, occasionally deviating from the commanded maneuver pattern and getting close to the central traverse behavior pattern near the straight line. Overall, SAGES provides safer and more semantically aligned trajectories, while simultaneously enabling more fuel-efficient solutions, compared to the standard CVX-SCP warm-start strategy.

For the spacecraft proximity operations scenario, all runtime measurements are obtained on the NVIDIA AGX Orin 64 GB platform, using unseen command templates paired with unseen problem instances. Overall, the SCP exhibits trends consistent with the free-flyer results. As shown in Fig. 8b, warm-starting SCP with the SAGES-WS improves convergence, even when the convex waypoint-hopping (CVX) initialization presents non-negligible constraint violations. The SAGES solutions also reliably recover the nominal CVX control cost, with most cases falling within 20 mm/s below to 50 mm/s above the CVX objective. By contrast, CVX-SCP often incurs significantly higher fuel usage—sometimes exceeding an additional 100 mm/s—particularly when the CVX warm-start is highly infeasible ($c_1 > 15$). Despite these differences, the iteration count and total runtime remain comparable to, or better than, those of CVX-SCP. The inference time of the transformer on the embedded GPU is negligible; instead, the runtime is dominated by the high-dimensional SCP, whose cost grows with the number of time discretization nodes. Although time-dilation methods [6] could reduce this burden, they increase sensitivity to initialization, especially for impulsive systems [46]. Table 7 further shows that feasibility-based refinement yields high convergence rates ($\simeq 95\%$ or more across all behavior modes) and strong post-refinement semantic correctness ($\geq 90\%$ for all modes except Behavior 3, which is already

challenging at warm-start; cf. Table 5). A clearer contrast emerges when comparing feasibility-based and fuel-optimal solves initialized with the same SAGES-WS solutions: fuel-optimal SCP suffers markedly lower convergence, as the semantically structured warm-starts are generally far from the minimum-fuel solution and require large corrective steps that often lead to nonconverging behaviors. Even when fuel-optimal SCP converges, semantic correctness deteriorates sharply, where Behaviors 3–5 often skip the intended waypoints entirely. These results indicate that feasibility-oriented refinement is substantially better aligned with preserving the command-driven behavioral structure, enabling reliable, semantically consistent, and safe trajectories.

Figure 9 qualitatively compares the representative trajectories generated by SAGES and CVX–SCP. In the free-flyer scenario, Figs. 9a and 9b illustrate two maneuvers initiated from the same state but commanded to execute a fast and a slow left passage. The former reaches its terminal waypoint in 24.8 s, while the latter requires 38.4 s; both remain collision-free and accurately reflect their respective high-level behaviors. By contrast, the CVX initialization produces an initially unsafe trajectory. Although the SCP refinement eventually recovers feasibility, the required corrections are substantial, leading to inflated fuel expenditure and a noticeable deviation from the intended motion.

A similar trend is observed in the spacecraft proximity operations scenario. Starting from $x_{\mathrm{qns},i} = [0, -110, -2.4, 5.8, 0.8, 4.2]$ m under the command "Approach the target over 4.7 orbits, then circumnavigate while maintaining a safe relative orbit" (Behavior 0), the trajectories produced by each initialization method and their SCP-refined counterparts are shown in Fig. 9c, together with the corresponding control profiles. The CVX warm-start enters the approach phase with inadequate RN-plane separation and only widens the relative orbit immediately before arrival. This yields a trajectory that is highly passively unsafe, inducing large corrective maneuvers during SCP and consequently increasing the overall control effort. In contrast, the SAGES-WS generates a safe RN-plane offset from the outset and transitions smoothly into the terminal circumnavigation. Only a small correction is required to satisfy the continuous-time passive safety constraints, resulting in a final SAGES trajectory that remains nearly indistinguishable from its warm-start (SAGES-WS).

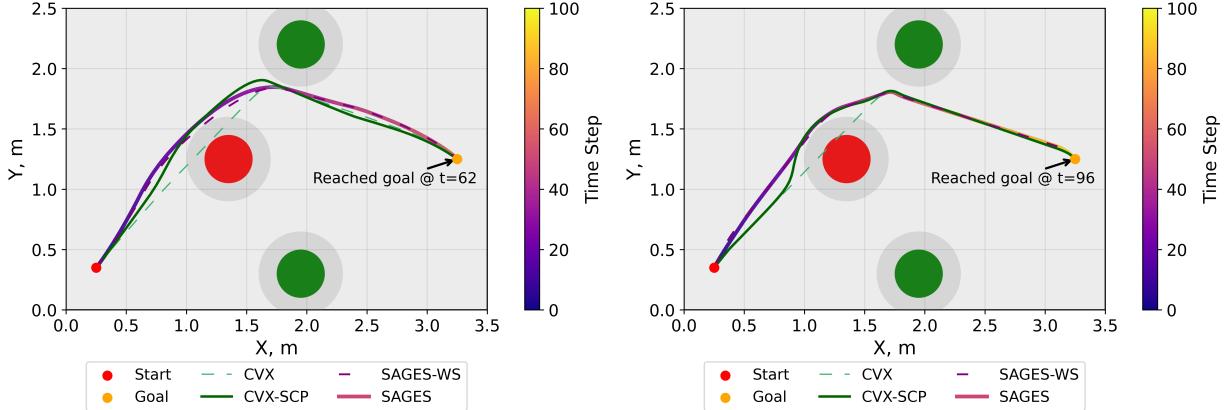### C. Hardware Demonstration on the Free-flyer Robotic Testbed

To validate the performance of SAGES in the physical world, hardware experiments are conducted using the free-flyer robotic testbed with an NVIDIA Jetson AGX Orin 64 GB. Once the open-loop guidance trajectory is generated by SAGES, a low-level PID controller operating at 10 Hz with $K_P = \mathrm{diag}[2.0, 2.0, 0.2]$, $K_D = \mathrm{diag}[45.0, 45.0, 0.4]$, and $K_I = \mathrm{diag}[0, 0, 0]$ was employed to track the guidance trajectory.

Figure 10 presents both qualitative and quantitative results from the free-flyer hardware experiments. The left panel of Fig. 10a illustrates the planned SAGES trajectory and its real-time execution under the PID controller for Behavior 1, corresponding to the command "For safe passage, skirt the left body while ensuring KOZ compliance and wide standoff." In contrast, the right panel shows the executed trajectory for the command "In this profile, a right-bias routing allows rapid transit near the KOZ." On the AGX Orin platform, the CVX warm-start pipeline requires 6–9 s, followed by 5-8 s of SCP refinement. In comparison, SAGES generates warm-starts in 2-3 s, with the SCP refinement stage taking a similar 5–8 s. While the refinement time is comparable or slightly better for SAGES relative to the CVX pipeline, the overall advantage is clearly observed in the warm-start phase, where the transformer dramatically outperforms the CVX-based warm-start, and SAGES remains competitive while providing significantly safer and more semantically aligned trajectories. This setup demonstrates the real-world applicability of the proposed approach under realistic computational constraints.
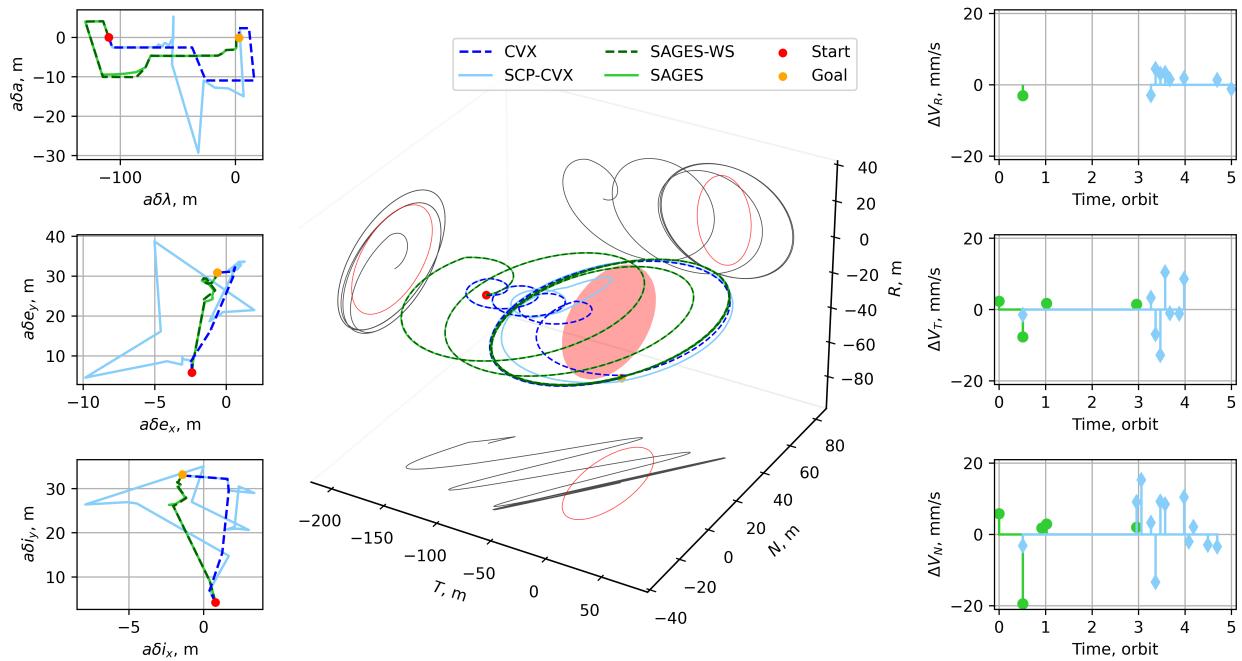
The distinction between fast and slow behavior is illustrated in Fig. 10b, which shows the time history of the vehicle's speed in the guidance (reference) trajectory and the realized (tracking) trajectory using the hardware. The slow behavior (Behavior 1) maintains a nonzero velocity until the final quarter of the horizon (30–40 s), whereas the fast behavior (Behavior 2) decelerates to near zero by approximately 33 s. Furthermore, the cumulative thruster firing times are 89.7 s for Behavior 1 and 131.0 s for Behavior 2, which confirms the increased fuel expenditure required to reach the goal more quickly. Although discrepancies arise between the reference and tracked velocity profiles—primarily due to modeling errors and the damping characteristics of the PID controller during reference overshoot—the hardware experiment confirms that the vehicle can reliably realize the intended velocity profile commanded through natural-language input with SAGES.

D. Towards Broadly Capable Semantic Behavior Models for Spacecraft

This work represents a first step toward a natural-language interface for spacecraft trajectory generation. The preceding sections show that SAGES can generate safe trajectories that remain semantically aligned with operator intent,

(a) Free-flyer Behavior 0: "To preserve clearance, a left-bias routing skirts the KOZ and rapid movement"

(b) Free-flyer Behavior 1: "Perform a broad left-side arc, expanding clearance for extended standoff with minimal Δv."
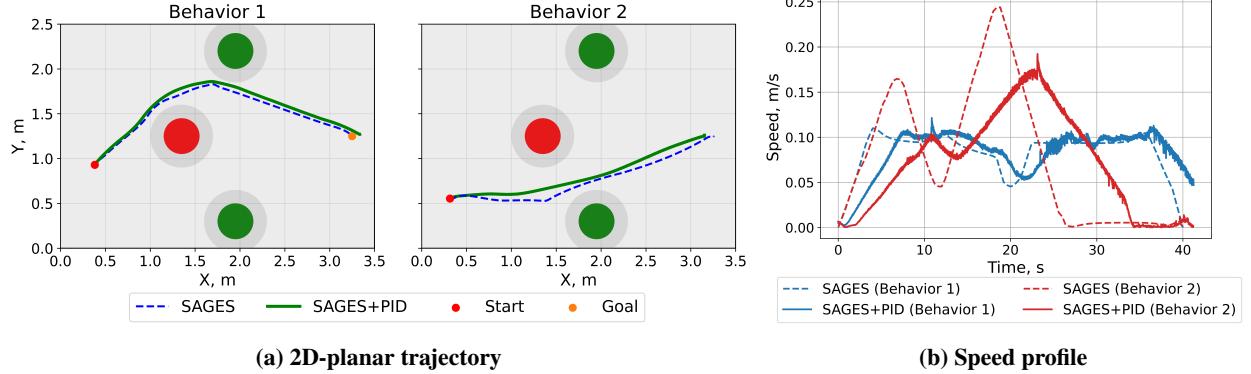

(c) Spacecraft proximity operations Behavior 0: "Approach target using 4.7 orbits; then execute a circumnavigation while ensuring a safe relative orbit." The projected trajectories in each plane are based on the SAGES output.

Fig. 9 Representative trajectories generated by SAGES for the free-flyer and spacecraft proximity operation scenario.

demonstrating the feasibility of text-conditioned and constraint-aware autonomy for proximity operations.

However, a central focus of future research must be on *improving generalization* and *providing formal performance guarantees* when such systems are queried outside their training distribution. While there is growing evidence that language can serve as a grounded representation to support combinatorial generalization of skills and concepts, spacecraft trajectory generation requires a joint understanding of linguistic intent and its corresponding physical realization under complex, often highly nonlinear dynamics. Closing this gap remains a major open challenge.

A central factor is the construction of the coupled text–trajectory dataset. In this study, the training corpus spans only a restricted set of behavior patterns and linguistic templates. Although this support can be expanded with greater computational resources and more extensive data generation, the current strategy is inherently bounded. Empirically, we found that performance degrades rapidly when the system is asked to synthesize trajectories corresponding to semantic behaviors outside the training modes. Future work should therefore examine the generalization properties of

(a) 2D-planar trajectory

(b) Speed profile

Fig. 10   Representative hardware executions of Behavior 1 and Behavior 2 on the free-flyer platform, comparing the planned motion (SAGES) and the realized motion with PID control (SAGES+PID). Videos: `https://semantic-guidance4space.github.io/`

SAGES-like frameworks in larger data regimes and develop principled methods to ensure reliable behavior beyond the training distribution.

At the same time, the structure of SAGES opens exciting opportunities for building safeguards and runtime monitors around FM-based trajectory generation. Figure 11 illustrates a two-dimensional Uniform Manifold Approximation and Projection (UMAP) [47] of the text embeddings associated with the spacecraft proximity operations scenario. The embeddings form clear clusters aligned with behavior modes across both seen (training) and unseen (validation) templates, indicating that the model captures meaningful distinctions in sentence-level semantics. This structure naturally supports out-of-distribution detection: a user command whose embedding falls far from the known clusters can be flagged as unsupported, preventing unintended or unsafe behaviors. Thus, while limited generalization presents challenges, it also provides a stable semantic backbone for real-time supervision.
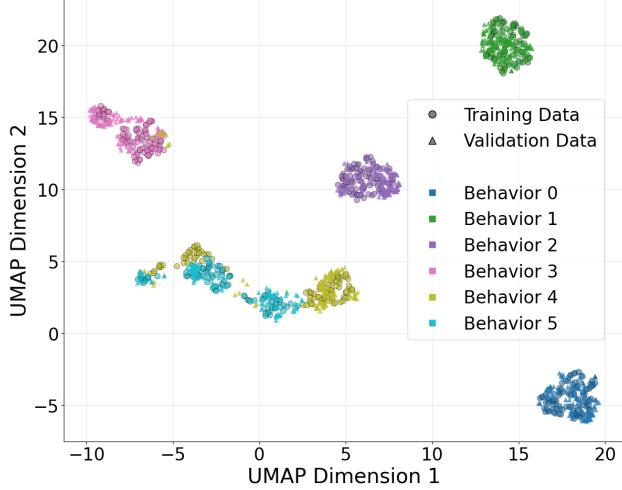
Beyond OOD detection, this clustering structure suggests a pathway for richer forms of runtime assurance. Embedding-space monitors could be used to (i) identify commands that ambiguously straddle multiple clusters, (ii) detect semantic drift during execution, or (iii) enforce guardrails on behavior modes by constraining the system to safe embedding regions—directly leveraging recent advances in safety filtering and anomaly detection in AI and robotics.

Another important challenge relates to compositional reasoning. Although the transformer processes text at the token level, supervision is provided only at the trajectory level, offering no explicit incentive to extract or recombine sub-behavior elements. As a result, the model tends to learn behavior modes as atomic units rather than developing a structured or compositional semantic space. Addressing this limitation will be crucial for enabling the synthesis of novel behavior sequences beyond the training distribution.

Several extensions may help broaden expressiveness while maintaining the safety benefits of a well-structured embedding space: (i) employing a task-adapted or partially trainable text encoder to capture finer semantic distinctions; (ii) incorporating a hierarchical policy capable of modeling subtask structure and temporal composition; and (iii) integrating SAGES with a higher-level multimodal reasoning model [35] to autonomously generate or refine behavior-level commands.

VI. Conclusion

This paper introduces a novel spacecraft trajectory generation framework capable of interpreting language-driven high-level behavioral commands while satisfying complex nonconvex constraints. The proposed two-stage architecture integrates a multi-modal transformer conditioned with the text and constraint information with a Sequential Convex Programming (SCP) refinement. The transformer generates a high-quality, text-conditioned warm-start trajectory that captures the intended behavior, whereas the SCP refines this trajectory to ensure compliance with nonconvex constraints, such as dynamics or safety, while preserving semantic consistency with the original command by solving a feasibility problem. Numerical experiments demonstrate that the proposed framework produces trajectories with significantly fewer constraint violations than traditional waypoint-based approaches, validated in two challenging scenarios: a free-flyer robotic system and a fault-tolerant spacecraft proximity operation with continuous-time constraint satisfaction. Overall,

Fig. 11 Distributions of the text commands used in the spacecraft proximity operation scenario projected onto the two-dimensional latent space via UMAP.

the proposed framework represents a foundational step toward human-centric spacecraft autonomy, enabling intuitive text-based commanding without requiring extensive domain expertise in constraint formulation or waypoint design.

Appendix

A. Command-Trajectory Dataset Generation

This subsection elaborates on the dataset generation process adopted in this paper, summarized in Fig. 12. To generate a semantically meaningful trajectory, the behavior modes that a spacecraft can take are first enumerated. Each behavior mode is represented as a sequence of waypoint regions with associated passage times. Using these waypoint specifications, a broad family of problem instances is created through domain randomization of key parameters such as boundary conditions and flight durations. For every randomized scenario, a nonconvex trajectory optimization problem is solved via SCP to produce a trajectory sample containing the full history of states, controls, constraint-to-go values, and other relevant parameters. In this work, to ensure that the dataset contains a sufficient number of feasible trajectories, only problem instances in which the SCP procedure successfully converged from a convex initialization are retained; when converged, both the convex trajectory and its SCP-refined counterpart are stored. In parallel, a large pool of text command templates is produced for each behavior mode. In this work, these templates are generated automatically using the ChatGPT-4o API, which synthesizes diverse, behavior-specific command forms based on curated keywords and descriptions with careful prompting. If the text command requires numerical values associated with the trajectory's property, numerical placeholders are placed in the command templates, which are later instantiated to produce concrete, scenario-consistent commands. Finally, trajectories and commands belonging to the same behavior mode are paired by random sampling and stored as elements of the text–trajectory dataset.
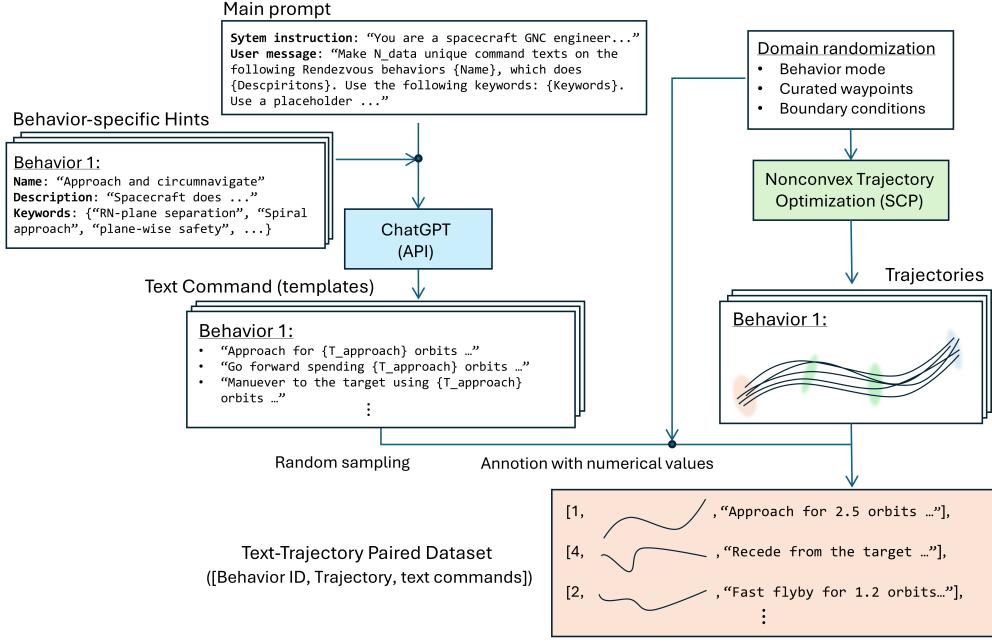
For each behavior mode, 120 distinct text commands (or command templates) are produced, where 100 are allocated to training, and 20 are reserved for testing, where the trajectory and the text command are randomly matched during the dataset generation.

The following sections present the prompt used to generate text commands for each behavior mode, along with an illustrative example of the associated keywords.

1. Free-flyer scenario
System instruction:

```
You are an expert GNC technical writer for proximity operations on a microgravity bench. Produce
    ONE sentence per input describing a goal-directed trajectory with KOZ compliance and left/right
    /central corridor behavior. Be concise (less than 15 words), precise, and varied in style.
```

Fig. 12 **Flowchart of the dataset generation pipeline: For each predefined behavior mode, trajectories are generated by solving batches of nonconvex SCP problems, while text commands are independently produced using the ChatGPT API to automate annotation. The two pools are then shuffled, and mode-consistent command–trajectory pairs are randomly sampled to form the final dataset.**

```
    Avoid jargon bloat.
```

User message template

```
Behavior description: {beh_text}
Task: Generate ONE sentence that characterizes this specific trajectory mode on a microgravity
testbed, capturing (i) path geometry, (ii) avoidance strategy, (iii) corridor selection,
and (iv) implied execution tempo.
Speed context: {speed_clause}
Style controls: Apply {style_voice}; {style_tone}; {style_structure}.
Guidance: Prefer concise proximity-operations phrasing (e.g., KOZ, standoff, LOS, RCS).
Do not reuse long phrases from the behavior description.
Strict constraints: <15 words; neutral and technically precise; no bullet points; no quotes;
avoid the following terms: {do_not_use}
Vocabulary hints: {geom_hints}; {spd_hints}
```

Example for Behavior 0

```
cmd = "Execute a rapid port-side arc that clears the KOZ while maintaining efficient transit."
```

2. Spacecraft proximity operation
System instruction:

```
You are an expert spacecraft GNC engineer and operator. Generate unique, short imperative templates
    with placeholders that will be filled later.
```

User message:

```
RPO Command: {cmd}
Generate {K} unique, short command templates (<={max_words} words) that ask the spacecraft to
    perform this behavior.
Behavior hints (optional vocabulary): {lexicon_hint}
```

```
Behavior structure and placeholder usage: {explanation_hint}
Hard constraints:
 0. Behavior structure: In each template, first state the terminal goal, then describe how it is
    achieved. Enumerate all phases in the order implied by the explanation; do not reorder phases.
 1. Uniqueness: Every template must be unique. Do not reuse the same sentence pattern.
 2. Length: Each template must be {max_words} words or fewer.
 3. Style: Use imperative commands directed at the spacecraft (e.g., Approach..., Hold..., Flyby
    ...).
 4. Placeholders: Use only the allowed placeholders {ph_str}, written exactly as {name}. Follow the
    phase order specified in the explanation.
 5. Units: Every placeholder denotes a time in orbits or a distance in meters. When used, append
    the unit (e.g., {T_appr_orbits} orbits, {d_lambda_meters} m).
 6. Placeholders {T_transfer_orbits} and {T_circ_orbits} denote the epoch at which a phase ends.
    They must be expressed using structures such as until {placeholder} orbits or ends at {
    placeholder} orbits. They must never be phrased as durations, such as for {placeholder} orbits.
```

Example for Behavior 0:

```
cmd = "Approach to the relative orbit around the target, and circumnavigate"
ph_str = "T_appr_orbits"
lexicon_hint = "E/I vector separation in relative orbit, spiral approach, plane-wise safety, RN-
    plane separation, after approach, skirt the keep-out zone"
explanation_hint = "Use {T_appr_orbits} orbits to make a spiral approach; upon arrival,
    circumnavigate for the remaining time with RN-plane separated safe relative orbit."
```

B. Free-flyer scenario Problem Parameters

Table 8 presents the problem-specific parameters used in the free-flyer scenario.

Table 8   Parameters in the free-flyer scenario.

R^{KOZ} [m]	r_{wyp} [m]	$\Delta\theta_0$ [°]	$(\rho_{\min}, \rho_{\max})$ [m]	T/m [N/kg]	ϵ_g [m]	N	Δt [s]
0.385	0.06	30	(0.04, 0.10)	0.005	0.06	100	0.4

C. Efficient Computation of Continuous-time Passive Safety under Imperfect burns

This subsection details the efficient computation of the continuous-time constraint satisfaction defined in Eq. (16). By exploiting the linear time-varying dynamics and the analytical State Transition Matrix (STM), the double integral over time τ and the impulse fraction α can be computed using a semi-analytical approach: exact integration over α followed by Gauss-Legendre quadrature over τ.

Let $\boldsymbol{x}(t_k; 0, \alpha) = \boldsymbol{x}_k + \alpha\Gamma_k\boldsymbol{u}_k$ denote the state immediately following a fractional impulse α. The drifted state at time $t_k + \tau$ along the reference state is given by $\bar{\boldsymbol{x}}(t_k; \tau, \alpha) = \Phi(t_k + \tau, t_k)\bar{\boldsymbol{x}}(t_k; 0, \alpha)$. Defining the auxiliary symmetric matrix $\Theta(t_k, \tau)$ and scalar coefficients a, b, c as

$$\Theta(t_k, \tau) := \Phi(t_k + \tau, t_k)^\top S_{k\tau} \Phi(t_k + \tau, t_k), \tag{19a}$$

$$a(\tau) := \bar{\boldsymbol{x}}_k^\top \Theta(t_k, \tau)\bar{\boldsymbol{x}}_k, \quad b(\tau) := \bar{\boldsymbol{x}}_k^\top \Theta(t_k, \tau)\Gamma_k\bar{\boldsymbol{u}}_k, \quad c(\tau) := (\Gamma_k\bar{\boldsymbol{u}}_k)^\top \Theta(t_k, \tau)\Gamma_k\bar{\boldsymbol{u}}_k, \tag{19b}$$

the function $\bar{g}(t_k; \tau, \alpha)$ defined in Eq. (16) becomes a concave quadratic in α:

$$\bar{g}(t_k; \tau, \alpha) := 1 - \bar{\boldsymbol{x}}(t_k; \tau, \alpha)^\top S_{k\tau} \bar{\boldsymbol{x}}(t_k; \tau, \alpha) = 1 - (1 + 2b(\tau))\alpha - c(\tau)\alpha^2. \tag{20}$$

For a fixed τ, the constraint violation occurs within the interval $[\alpha_1, \alpha_2] \subseteq [0, 1]$ where $\bar{g}(t_k; \tau, \alpha) \geq 0$, which is

simply a quadratic inequality. The fundamental moments of $\bar{g}(t_k; \tau, \alpha)$ over this active set admit closed-form solutions:

$$I_0(\tau) = \int_{\alpha_1}^{\alpha_2} \bar{g}(t_k; \tau, \alpha)d\alpha = (1-a)\Delta_1 - b\Delta_2 - \frac{c}{3}\Delta_3, \tag{21a}$$

$$I_1(\tau) = \int_{\alpha_1}^{\alpha_2} \alpha\bar{g}(t_k; \tau, \alpha)d\alpha = \frac{1-a}{2}\Delta_2 - \frac{2b}{3}\Delta_3 - \frac{c}{4}\Delta_4, \tag{21b}$$

$$I_2(\tau) = \int_{\alpha_1}^{\alpha_2} \alpha^2\bar{g}(t_k; \tau, \alpha)d\alpha = \frac{1-a}{3}\Delta_3 - \frac{b}{2}\Delta_4 - \frac{c}{5}\Delta_5, \tag{21c}$$

with $\Delta_n := \alpha_2^n - \alpha_1^n$. Utilizing the chain rule on Eq. (17a), the gradient terms G_k^x and G_k^u are derived as weighted sums of these moments:

$$G_k^x = -4\int_0^{\tau^s} \Theta(t_k, \tau)\left[\bar{x}_k I_0(\tau) + \Gamma_k\bar{u}_k I_1(\tau)\right]d\tau, \quad G_k^u = -4\int_0^{\tau^s} \Gamma_k^\top\Theta(t_k, \tau)\left[\bar{x}_k I_1(\tau) + \Gamma_k\bar{u}_k I_2(\tau)\right]d\tau. \tag{22a}$$

Similarly, the integral cost $\tilde{g}(\bar{x}_k, \bar{u}_k)$ is obtained by integrating the square of Eq. (20):

$$\tilde{g}(\bar{x}_k, \bar{u}_k) = \int_0^{\tau^s}\left[a^2\Delta_1 - 2ab\Delta_2 + \left(\frac{4}{3}b^2 - \frac{2}{3}ac\right)\Delta_3 + bc\Delta_4 + \frac{1}{5}c^2\Delta_5\right]d\tau. \tag{23}$$

Note that the coefficients a, b, c and the roots $\alpha_{1,2}$ are implicit functions of τ.

Since the STM is analytical, the remaining integrals over τ in Eqs. (22) and (23) are efficiently calculated using the Gauss–Legendre quadrature rule. The time interval $\tau \in [0, \tau^s]$ is mapped to the quadrature domain $\xi \in [-1, 1]$ via the transformation $\tau(\xi) = \frac{\tau^s}{2}(\xi + 1)$. The integral of a generic function $h(\tau)$ is approximated as

$$\int_0^{\tau^s} h(\tau)\,d\tau \approx \frac{\tau^s}{2}\sum_{j=1}^{N_q} w_j h\left(\tau(\xi_j)\right), \tag{24}$$

where N_q is the number of sample points, and (ξ_j, w_j) are the standard Legendre nodes and weights. Consequently, the convexified constraints are evaluated by summing the integrands of Eqs. (22) and (23) at the N_q evaluation points. This approach eliminates the need for dense ODE propagation while maintaining high accuracy for constraint satisfaction, making the constraint formulation more appealing to onboard applications. In the spacecraft proximity operation scenario, $N_q = 30$ is adopted.

D. SCP Parameters
Table 9 presents the hyperparameters of the SCP used for the spacecraft proximity operation scenario [13].

Table 9   Hyperparameters of SCVx* [13] used for spacecraft proximity operation scenario.

ϵ	$\{\rho_0, \rho_1, \rho_2\}$	$\{\alpha_1, \alpha_2, \beta, \gamma\}$	$\{r^{(1)}, r_{\min}, r_{\max}\}$	$\{w^{(1)}, w_{\max}\}$	# max. iter
10^{-3}	$\{0.0, 0.25, 0.7\}$	$\{2, 2, 1.5, 0.9\}$	$\{0.5, 10^{-6}, 10\}$	$\{10, 10^9\}$	100

E. Transformer Model
The presented transformer-based trajectory generation is implemented in PyTorch [48] and builds on Hugging Face's `transformers` library [‡]. Table 10 presents the hyperparameter settings used in this work.

Acknowledgements

[‡] https://huggingface.co/docs/transformers/index

Table 10 Hyperparameters of the causal transformer.

Hyperparameter	Value
Number of layers	6
Number of attention heads	6
Embedding dimension, h	384
Batch size	4
Context length K	50
Non-linearity	ReLU
Dropout	0.1
Learning rate	3e-5
Grad norm clip	1.0
Learning rate decay	None
Gradient accumulation iters	8

References

[1] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 41, No. 1, 2014, pp. 1–37.

[2] Lawden, D. F., *Optimal trajectories for space navigation*, 1963.

[3] Koenig, A. W., and D'Amico, S., "Fast algorithm for fuel-optimal impulsive control of linear systems with time-varying cost," *IEEE Transactions on Automatic Control*, Vol. 66, No. 9, 2020, pp. 4029–4042.

[4] Foss, E., and D'Amico, S., "Efficient Input-Constrained Impulsive Optimal Control of Linear Systems with Application to Spacecraft Relative Motion," *arXiv preprint arXiv:2510.03423*, 2025.

[5] Gershwin, S. B., and Jacobson, D. H., "A discrete-time differential dynamic programming algorithm with application to optimal orbit transfer," *AIAA Journal*, Vol. 8, No. 9, 1970, pp. 1616–1626.

[6] Malyuta, D., Reynolds, T. P., Szmuk, M., Lew, T., Bonalli, R., Pavone, M., and Açıkmeşe, B., "Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently," *IEEE Control Systems Magazine*, Vol. 42, No. 5, 2022, pp. 40–113. https://doi.org/10.1109/MCS.2022.3187542.

[7] Blackmore, L., "Autonomous precision landing of space rockets," *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*, Vol. 46, The Bridge Washington, DC, USA, 2016, pp. 15–20.

[8] Strohl, L., Doll, J., Fritz, M., Berning, A. W., White, S., Bieniawski, S. R., Carson, J. M., and Acikmese, B., "Implementation of a six degree of freedom precision lunar landing algorithm using dual quaternion representation," *AIAA SCITECH 2022 Forum*, 2022, p. 1831.

[9] Berning Jr, A. W., Burnett, E. R., and Bieniawski, S., "Chance-Constrained, Drift-Safe Guidance for Spacecraft Rendezvous," *arXiv preprint arXiv:2401.11077*, 2024.

[10] Rizza, A., D'Amico, S., and Topputo, F., "Goal-Oriented Trajectory Refinement for Asteroid Mapping Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, 2025, pp. 1–15.

[11] Mao, Y., Szmuk, M., and Açıkmeşe, B., "Successive convexification of non-convex optimal control problems and its convergence properties," *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 3636–3641.

[12] Bonalli, R., Cauligi, A., Bylard, A., and Pavone, M., "GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming," *IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, May, 2019.

[13] Oguri, K., "Successive convexification with feasibility guarantee via augmented lagrangian for non-convex optimal control problems," *Proc. IEEE Conf. on Decision and Control*, IEEE, 2023, pp. 3296–3302.

[14] Malyuta, D., and Açıkmeşe, B., "Fast homotopy for spacecraft rendezvous trajectory optimization with discrete logic," *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 7, 2023, pp. 1262–1279.

[15] Szmuk, M., Reynolds, T. P., and Açıkmeşe, B., "Successive convexification for real-time six-degree-of-freedom descent guidance with state-triggered constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 8, 2020, pp. 1399–1413.

[16] Mao, Y., Acikmese, B., Garoche, P.-L., and Chapoutot, A., "Successive convexification for optimal control with signal temporal logic specifications," *Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control*, 2022, pp. 1–7.

[17] Blackmore, L., Ono, M., and Williams, B. C., "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, Vol. 27, No. 6, 2011, pp. 1080–1094.

[18] Ridderhof, J., Okamoto, K., and Tsiotras, P., "Nonlinear uncertainty control with iterative covariance steering," *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 3484–3490.

[19] Kumagai, N., and Oguri, K., "Sequential chance-constrained covariance steering for robust cislunar trajectory design under uncertainties," *AAS/AIAA Astrodynamics Specialist Conference*, 2024.

[20] Takubo, Y., and D'Amico, S., "Multiplicative Approach to Constrained Stochastic Attitude Control with Application to Rendezvous and Proximity Operations," *AAS/AIAA Astrodynamics Specialist Conference, Broomfield, CO, USA*, 2024.

[21] Elango, P., Luo, D., Kamath, A. G., Uzun, S., Kim, T., and Açıkmeşe, B., "Continuous-time successive convexification for constrained trajectory optimization," *Automatica*, Vol. 180, 2025, p. 112464.

[22] D'Amico, S., "Autonomous Formation Flying in Low Earth Orbit." PhD Thesis, Delft University, 2010.

[23] Alfriend, K., Vadali, S. R., Gurfil, P., How, J., and Breger, L., *Spacecraft formation flying: Dynamics, control and navigation*, Vol. 2, Elsevier, 2009.

[24] Guffanti, T., Bell, T., Low, S. Y., Murray-Cooper, M., and D'Amico, S., "Autonomous Guidance Navigation and Control of the VISORS Formation-Flying Mission," *2023 AAS/AIAA Astrodynamics Specialist Conference, Big Sky, MT.*, 2023.

[25] Rizza, A., Foss, E., Monnier, J., Cutler, J. W., and D'Amico, S., "Space Interferometry Formation Design using Relative Orbital Elements: the STARI Mission," *2026 IEEE Aerosapce Conference (accepted)*, 2026.

[26] Takubo, Y., Manuel, W., Foss, E., and D'Amico, S., "Safe and Optimal N-Spacecraft Swarm Reconfiguration in Non-Keplerian Cislunar Orbits," *Journal of Guidance, Control, and Dynamics*, 2025. Accepted for publication.

[27] Ceballos, A., Bensalem, S., Cesta, A., De Silva, L., Fratini, S., Ingrand, F., Ocon, J., Orlandini, A., Py, F., Rajan, K., et al., "A goal-oriented autonomous controller for space exploration," *ASTRA*, Vol. 11, 2011.

[28] Lynch, C., and Sermanet, P., "Language conditioned imitation learning over unstructured data," *arXiv preprint arXiv:2005.07648*, 2020.

[29] Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T., "A survey of reinforcement learning informed by natural language," *arXiv preprint arXiv:1906.03926*, 2019.

[30] Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al., "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[31] Kawaharazuka, K., Oh, J., Yamada, J., Posner, I., and Zhu, Y., "Vision-language-action models for robotics: A review towards real-world applications," *IEEE Access*, 2025.

[32] Carrasco, A., Rodriguez-Fernandez, V., and Linares, R., "Large Language Models as Autonomous Spacecraft Operators in Kerbal Space Program," *Advances in Space Research*, 2025.

[33] Carrasco, A., Nedungadi, M., Rodriguez-Fernandez, V., and Linares, R., "Visual language models as operator agents in the space domain," *AIAA SCITECH 2025 Forum*, 2025, p. 1543.

[34] Zucchelli, E. M., Wu, D., Briden, J., Hofmann, C., Rodriguez-Fernandez, V., and Linares, R., "Fine-Tuned Language Models as Space Systems Controllers," *arXiv preprint arXiv:2501.16588*, 2025.

[35] Foutter, M., Gammelli, D., Kruger, J., Foss, E., Bhoj, P., Guffanti, T., D'Amico, S., and Pavone, M., "Space-LLaVA: A Vision-Language Model Adapted to Extraterrestrial Applications," *2025 IEEE Aerospace Conference*, 2025, pp. 1–23. https://doi.org/10.1109/AERO63441.2025.11068757.

[36] Guffanti, T., Gammelli, D., D'Amico, S., and Pavone, M., "Transformers for Trajectory Optimization with Application to Spacecraft Rendezvous," *IEEE Aerospace Conference*, 2024, pp. 1–13. https://doi.org/10.1109/AERO58975.2024.10521334.

[37] Celestini, D., Gammelli, D., Guffanti, T., D'Amico, S., Capello, E., and Pavone, M., "Transformer-based model predictive control: trajectory optimization via sequence modeling," *IEEE Robotics and Automation Letters*, Vol. 9, 2024, pp. 9820–9827. https://doi.org/10.1109/LRA.2024.3466069.

[38] Takubo, Y., Guffanti, T., Gammelli, D., Pavone, M., and D'Amico, S., "Towards Robust Spacecraft Trajectory Optimization via Transformers," *IEEE Aerospace Conference*, 2025.

[39] Celestini, D., Afsharrad, A., Gammelli, D., Guffanti, T., Zardini, G., Lall, S., Capello, E., D'Amico, S., and Pavone, M., "Generalizable spacecraft trajectory generation via multimodal learning with transformers," *2025 American Control Conference (ACC)*, IEEE, 2025, pp. 3558–3565.

[40] Takubo, Y., Gammelli, D., Pavone, M., and D'Amico, S., "Agile Tradespace Exploration for Space Rendezvous Mission Design via Transformers," *IEEE Aerospace Conference*, 2026. (accepted).

[41] Putterman, A. L., Lu, K., Mordatch, I., and Abbeel, P., "Pretraining for Language Conditioned Imitation with Transformers," *Offline Reinforcement Learning Workshop at Neural Information Processing Systems (NeurIPS)*, 2021.

[42] Mao, Y., and Acikmese, B., "SCvx-fast: A superlinearly convergent algorithm for a class of non-convex optimal control problems," *arXiv preprint arXiv:2112.00108*, 2021.

[43] Koenig, A. W., Guffanti, T., and D'Amico, S., "New state transition matrices for spacecraft relative motion in perturbed orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1749–1768.

[44] Kruger, J. J., Guffanti, T., Park, T. H., Murray-Cooper, M., Low, S. Y., Bell, T., D'Amico, S., Roscoe, C. W., and Westphal, J., "Adaptive End-to-End Architecture for Autonomous Spacecraft Navigation and Control During Rendezvous and Proximity Operations," *AIAA SCITECH 2024 Forum*, 2024, p. 0430.

[45] Sanh, V., Debut, L., Chaumond, J., and Wolf, T., "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[46] Kumagai, N., and Oguri, K., "Robust nrho station-keeping planning with maneuver location optimization under operational uncertainties," *AAS/AIAA Astrodynamics Specialist Conference*, 2023.

[47] McInnes, L., Healy, J., and Melville, J., "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," , 2020. URL https://arxiv.org/abs/1802.03426.

[48] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, Vol. 32, 2019.