

OVITA: Open-Vocabulary Interpretable Trajectory Adaptations

Anurag Maurya¹, Tashmoy Ghosh¹, Anh Nguyen² and Ravi Prakash¹

Abstract—Adapting trajectories to dynamic situations and user preferences is crucial for robot operation in unstructured environments with non-expert users. Natural language enables users to express these adjustments in an interactive manner. We introduce OVITA, an interpretable, open-vocabulary, language-driven framework designed for adapting robot trajectories in dynamic and novel situations based on human instructions. OVITA leverages multiple pre-trained Large Language Models (LLMs) to integrate user commands into trajectories generated by motion planners or those learned through demonstrations. OVITA employs code as an adaptation policy generated by an LLM, enabling users to adjust individual waypoints, thus providing flexible control. Another LLM, which acts as a code explainer, removes the need for expert users, enabling intuitive interactions. The efficacy and significance of the proposed OVITA framework is demonstrated through extensive simulations and real-world environments with diverse tasks involving spatiotemporal variations on heterogeneous robotic platforms such as a KUKA IIWA robot manipulator, Clearpath Jackal ground robot, and CrazyFlie drone.

Index Terms—Motion and Path Planning, Human-Robot Collaboration, Big Data in Robotics and Automation.

I. INTRODUCTION

Robotic systems have increasingly permeated diverse domains, from industrial automation to service robotics, demanding efficient trajectory generation and adaptation techniques. A fundamental challenge in this context lies in enabling robots to generalize in dynamic and unstructured environments. Large Language Models (LLMs) have gained attention for their reasoning abilities, driving their use in robotics, particularly for high-level task planning in embodied AI and human-robot collaboration [1]–[3]. However, LLMs are underutilized in low-level control, often relying on pretrained skills and motion primitives. Methods like VoxPoser [4] generate affordances, cost functions, and 3D cost maps, while Language to Rewards [5] designs reward functions. Recent works [6] has shown LLMs can generate zero-shot dense trajectories, but their outputs remain inferior to human demonstrations and are limited to simple shapes and linear interpolations.

Traditionally, robot trajectories are either planned from scratch using sampling-based methods (e.g. Rapidly-Exploring Random Trees (RRT), RRT* [7] and MPC [8]) or adapted from previously learned demonstrations. While the former lacks online reactivity to uncertain situations, the latter learns

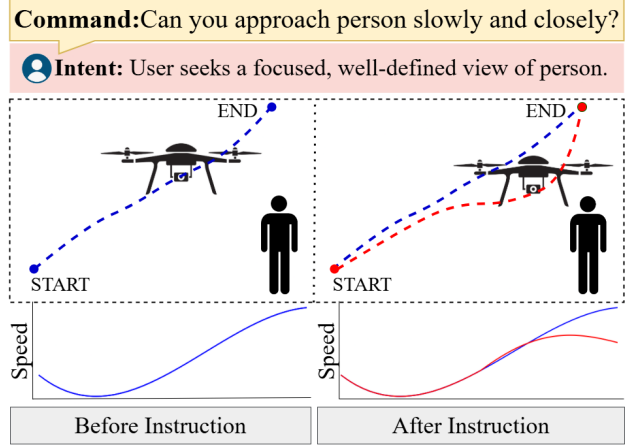


Fig. 1: Traditional planning algorithms often overlook user intent. OVITA enables intuitive adjustments through language commands. The blue trajectory represents the initial path, while the red one reflects sample OVITA output.

adaptive and reactive motion plans through Learning from Demonstrations (LfD) [9] that are inherently robust to uncertainties and changes in the environment. Robot learning from demonstrations allows humans, regardless of expertise, to transfer task knowledge to robots [10], eliminating the need for tedious, error-prone manual programming. However, a key challenge remains in generalizing learned behaviors to new situations. Moreover, a high number of samples is typically needed for each task.

We propose OVITA, a user-friendly robot trajectory adaptation method that leverages LLMs for interactive and interpretable adjustments via natural language. By eliminating complex task parameterization, OVITA unifies trajectory adaptations under a single framework. Natural language instructions enable users to convey intent intuitively, allowing robots to dynamically adjust to task variations.

For example, a planner may generate a trajectory for a drone to surveil a region. However, if the user intends to have a focused and well-defined view of an individual, they can issue a command like, "Can you approach person slowly and closely?" (Fig. 1). In this context, natural language serves as a medium for the user to express specific preferences. Similarly, in object transportation tasks, a predefined trajectory might guide the robot in loading items onto a table. However, if the user needs more time to add additional items, they could issue a command like, "Stop near the table for 10 more timesteps," allowing the robot to adapt dynamically to the revised requirements. This flexibility highlights the value of combining human instructions with trajectory adaptation.

Natural language provides an intuitive medium for users to express their intent, but its open-vocabulary nature introduces

Manuscript received: March, 21, 2025; Revised June, 28, 2025; Accepted August, 13, 2025.

This paper was recommended for publication by Editor Olivier Stasse upon evaluation of the Associate Editor and Reviewers' comments.

¹First Author, Second Author, and Fourth Author are with Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bangalore, India (anuragm1, tashmoyg, ravipr)@iisc.ac.in

²Third Author is with the Department of Computer Science, University of Liverpool, England anh.nguyen@liverpool.ac.uk

Digital Object Identifier (DOI): see top of this page.

unique challenges. Unstructured commands can result in inconsistent interpretations. For example, the instruction "Stay at a larger distance from the cup" may be understood differently by users—one might interpret it as a slight adjustment to only nearby points, while another might see it as requiring a shift to all the waypoints. By offering corrections when adaptations deviate from expectations, users can iteratively refine the robot's actions, leading to more personalized and contextually appropriate adjustments. Transparency and interpretability are equally vital in this adaptation process [11]. Black-box systems that obscure trajectory modifications make it difficult for users to assess whether their instructions have been correctly implemented. On the other hand, clear visualization of adjustments empowers users to evaluate the outcomes, identify discrepancies, and refine their commands.

Starting with a trajectory from a planner or human demonstration, our method adjusts low-level actions using natural language commands. OVITA eliminates the reliance on complex mathematical formulations and prioritizes intuitive, user-driven adjustments. Our method adjusts key waypoints while maintaining trajectory smoothness. It interprets natural language as specific modifications and encodes them into actionable updates using code as an adaptation policy.

OVITA offers the following key contributions:

- It uses pre-trained LLMs to interpret open-vocabulary instructions for spatiotemporal trajectory adaptations in a multi-agent framework.
- It supports exact numerical changes (e.g., "Go left by 0.2"), open-ended commands (e.g., "Move in a spiral"), and multi-step adjustments with feedback.
- The integration of a formal, QP-based optimization module that translates LLM-generated adaptations into smooth, safe, and physically-feasible trajectories.
- Real-world experimental validation of framework on heterogeneous robotics platforms and tasks.

To the best of our knowledge, OVITA is the first interpretable framework for trajectory adaptation with open-vocabulary instruction support. The paper is structured as follows: Section II reviews related literature, Section III outlines the proposed methodology, Section IV presents results and discussions, and Section V covers limitations and conclusions.

II. RELATED WORKS

A. Trajectory adaptation in Robotics

In new or changing scenarios, robot motion plans must adapt to sensory feedback—a key research area in Imitation Learning [12]. By enhancing the policy with task parameterization [13], robots can generalize their learned knowledge to different variations of the same task, allowing robots to adapt to new scenarios. This can be popularly achieved by Dynamic Movement Primitives (DMPs) [14], which uses second-order stable attractor dynamics with adjustable spatial targets and temporal execution. Alternatively, skill generalization has been formulated with ProMPs [15] and KMPs [16]. While ProMP uses basis functions and Gaussian conditioning to adapt trajectories to any waypoints while maintaining temporal and DoF correlations, KMP adapts these trajectories using kernel-based

temporal correlations. By encoding probability distributions of trajectories concerning task-relevant reference frames, TP-GMM (Task-Parameterized Gaussian Mixture Model) enables motion generation through TP-GMM + GMR/GPR [17] with improved out-of-distribution targets. Despite their utility, these methods require task knowledge, parameter tuning, and extensive expert demonstrations, limiting scalability for non-experts. OVITA seamlessly adapts to out-of-distribution targets, modifying waypoints solely through natural language instructions. In addition, we can add a new task and generate extra waypoints originally not present in the original trajectories.

B. Large Language Models for Robotics

Recent advancements in LLMs have facilitated their integration into robotic task planning. The key works in high-level task planning include the following. PaLM-E [18] introduces embodied language models that connect language with real-world sensor data, using multimodal inputs and outputs high-level plan. LLaRP [19] uses an LLM-based reinforcement learning policy to translate text and visual inputs into a plan of predefined actions. RobotGPT [20] uses ChatGPT to iteratively refine code to generate data for training agents. SayCan [21] combines LLM-based task grounding with value functions for action feasibility, while Code-as-Policies [22] pioneered generating interpretable, executable code with predefined motion primitives. Unlike end-to-end models, this approach offers debuggable logic, linking language, and robotic control. To incorporate user preferences into high-level robotic plans, [23] uses a closed-loop strategy with feedback like success detection, scene description, and human interaction. LLMs have also been used to generate low-level trajectories, translating task descriptions into movement sequences [6]. Their approach is limited to simple trajectories and tabletop tasks, while ours adapts complex trajectories to user preferences and supports diverse robot dynamics and morphologies.

C. Trajectory Adaptation with Natural Language

Natural language is a powerful interface for robots, but linking it to low-level motion plans is challenging. Traditionally, representing human-robot interactions through language has required either rigid instructions [24] or complex mathematical methods to model probabilities of actions and targets [25]. LILAC [26] enables online trajectory corrections through natural language inputs in shared autonomy. Recent advances in NLP foundation models like BERT [27] have enabled end-to-end learning with deep embeddings. Prior work has leveraged LLMs for either optimizing cost maps via motion planners [28] or mapping input to output trajectories based on geometric object poses and LLM-based language embeddings [29], [30]. Notably, the failure analysis of LaTTe [30] by ExTraCT [31] highlighted that embedding models often generate similar embeddings for sentences with opposing meanings but with high lexical similarity. Additionally, prior approaches rely on templated, qualitative instructions and lack support for precise corrections (e.g., "Go left by 0.3") or multi-instruction commands. These limitations motivate our development of a robust, language-driven trajectory

adaptation pipeline. OVITA incorporates precise numerical adjustments and multi-instruction commands. Additionally, It handles open-ended directives, such as: “Set the new goal position as the midpoint between the box and sofa, and modify the trajectory smoothly to reach it,” or “Move slightly closer to the punching bag and follow a trapezoidal velocity profile.”. This capability sets our method apart.

The most similar works to ours are LaTTe (Language Trajectory Transformers) [30] and ExTraCT [31]. LaTTe combines language embeddings and spatial representations in a multi-modal, end-to-end framework. However, it requires extensive training data, is non-explainable, and lacks support for complex and open-vocabulary interactions. ExTraCT uses language embeddings to map deformation functions but depends on handcrafted features, limiting its generalizability. In contrast, OVITA is a training-free(zero-shot) method for complex, multi-step numerical adaptations. It requires no fine-tuning, is fully interpretable, and supports open-vocabulary, monologue-based interactions. Additionally, feedback support facilitates iterative refinement.

III. METHODOLOGY

A. Language-based Adaptation Pipeline

Given an initial trajectory $\tau = \{(x_i, y_i, z_i, v_i)\}_{i=1}^N$, consisting of N waypoints, where (x_i, y_i, z_i) represent the coordinates and v_i the corresponding speed. We also consider a user instruction L_{instruct} and environment observations $\{\text{Position}(O_i), \text{Dimension}(O_i), \text{Label}(O_i)\}_{i=1}^m$ for m objects. These observations may optionally include an environmental description E_d . Our goal is to construct a mapping $g : \tau_{\text{initial}} \rightarrow \tau_{\text{modified}}$ that adapts the trajectory according to L_{instruct} . We leverage LLMs to achieve this adaptation. The adaptation process is structured in four phases. prompt grounding, code-policy execution, visualization, and feedback integration.

First Phase: Prompt Grounding

This phase translates user instructions into a grounded prompt (Fig. 2 A) using a task-agnostic template that provides context for LLM decisions and structured code generation. The template includes core components common to all adaptations. 1) *Environment configuration*: A defined coordinate system and a placeholder for environment descriptors to provide both spatial and semantic context. 2) *Trajectory Characteristics*: Rules for smooth transitions and to allow for adding or removing waypoints, etc. 3) *Function Definitions*: The prompt includes functions like `detect_objects()` to retrieve object positions and `get_trajectory()` for trajectory data. 4) *Examples*: Two examples demonstrate the response format for a high-level plan, not the code policy. These examples are kept the same across all tasks.

Prompt Grounding: The user observes the initial trajectory and issues a natural language command. We add environmental context using semantic descriptions, either generated by a vision-language model (VLM) or provided by the user. The VLM describes objects in the scene, including their names, properties (e.g., color), and layout. An object detector then extracts their 3D positions and sizes. (Details in IV-E). The

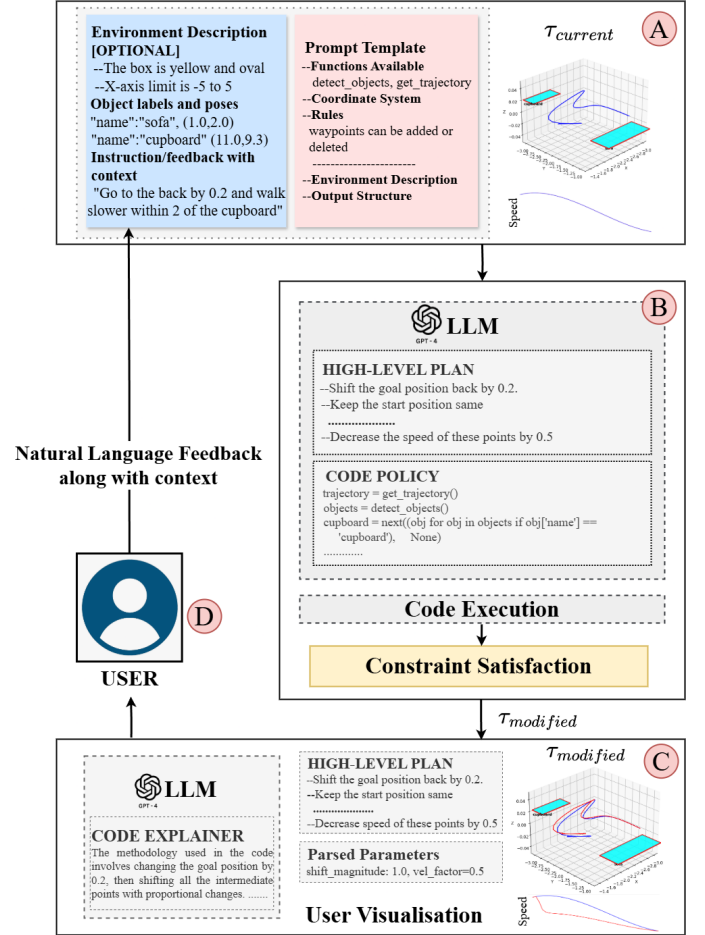


Fig. 2: **OVITA Block Diagram:** A) Prompt grounding from instruction and environment description. B) LLM generates HLP and code, followed by execution and constraint satisfaction. C) Visualization with code explanation. D) User feedback for refinement.

user’s instruction, scene descriptions, and object positions are then structured within the prompt template, resulting in a physically grounded prompt. prompts are provided in the prompts.py file of the codebase.

Second Phase: Code-Policy generation and Execution

In this phase, the LLM processes a unified prompt to generate a dictionary with both a high-level plan and Python code in a single generation step (Fig. 2 B). Empirically, we observed that this method leads to higher consistency compared to separate generations. The high-level plan specifies the precise steps required to align the trajectory with user instructions. This plan guides the LLM in generating Python code, which transforms input trajectory data (position and velocity sequences) into adapted output sequences. The code bridges high-level directives and low-level execution, acting as an adaptation policy (Fig. 3). AST (Abstract Syntax Trees) parsing is employed to refine the generated code, extract key parameters, and remove unnecessary elements. The resulting code is executed in a Python interpreter, producing the adapted trajectory. An optional constraint satisfaction module (CSM)

```

trajectory = get_trajectory() # Tool-call
objects = detect_objects() # Tool-call
cassette = next((obj for obj in objects if obj['name'] == 'cassette'), None)
if cassette:
    cassette_x, cassette_y, cassette_z = cassette['x'], cassette['y'], cassette['z']
    factor = 0.1 # Parameter controlling the strength of proximity
    for point in trajectory:
        # Adaptation to bring trajectory closer to cassette
        vector_to_cassette = {'x': cassette_x - point['x'],
                              'y': cassette_y - point['y'], 'z': cassette_z - point['z']}
        for coordinate in ['x', 'y', 'z']:
            point[coordinate] += vector_to_cassette[coordinate] * factor
        point['velocity'] = 1.0 # Adaptation to move at constant velocity
    modified_trajectory = trajectory

```

Fig. 3: Code as an adaptation policy: the instruction is "Move closer to the cassette while maintaining a constant speed throughout."

validates the trajectory, ensuring the environment and robot-specific constraints (Details in III-B).

Third Phase: User Visualization

In this phase, the user is shown a comparative visualization (trajectory plots) of the adapted and initial trajectories, along with access to the high-level plan, and parsed variables (Fig. 2 C). This enables users to provide well-informed feedback. To mitigate potential interpretation challenges for non-expert users, a code explanation module was integrated. This module utilizes an LLM to generate accessible explanations from the code, high-level plan, and extracted parameters. **Code-explanation prompt** explains code-adaptation policy by: 1) summarizing the methodology used; 2) detailing the function and values of key hyperparameters; and 3) outlining logical assumptions underlying trajectory adjustments.

Fourth Phase: Feedback integration

This phase introduces a crucial feedback loop (Fig. 2 D). Natural language ambiguity and user diversity can cause LLM hallucinations. To address this, natural language feedback is incorporated, enabling iterative corrections. As illustrated in Fig. 4, this feedback loop effectively refines the high-level plan, mitigating LLM hallucinations and enhancing adherence to user preferences. We observed ambiguity in user feedback, as it could target either the initial or current trajectory. To avoid confusion, we implemented a feedback context selector, offering "original" or "current" options. Choosing "Original" appends new feedback to the original instruction, applying the resulting code policy to the original trajectory. Selecting "current" applies the code policy to the previously modified trajectory.

B. Constraint Satisfaction Module

We formulate the trajectory optimization as a convex Quadratic Program (QP) over a sequence of T 3D waypoints and their speed $\mathbf{x} = [\mathbf{x}_0^\top, \mathbf{x}_1^\top, \dots, \mathbf{x}_{T-1}^\top]^\top \in \mathbf{R}^{4T}$. The objective penalizes deviation from the reference trajectory \mathbf{x}^{ref} and encourages smoothness via first-order differences.

$$\min_{\mathbf{x} \in \mathbf{R}^{4T}} \lambda_{\text{dev}} \|\mathbf{x} - \mathbf{x}^{\text{ref}}\|_2^2 + \lambda_{\text{smooth}} \|D_1 \mathbf{x}\|_2^2 \quad (1)$$

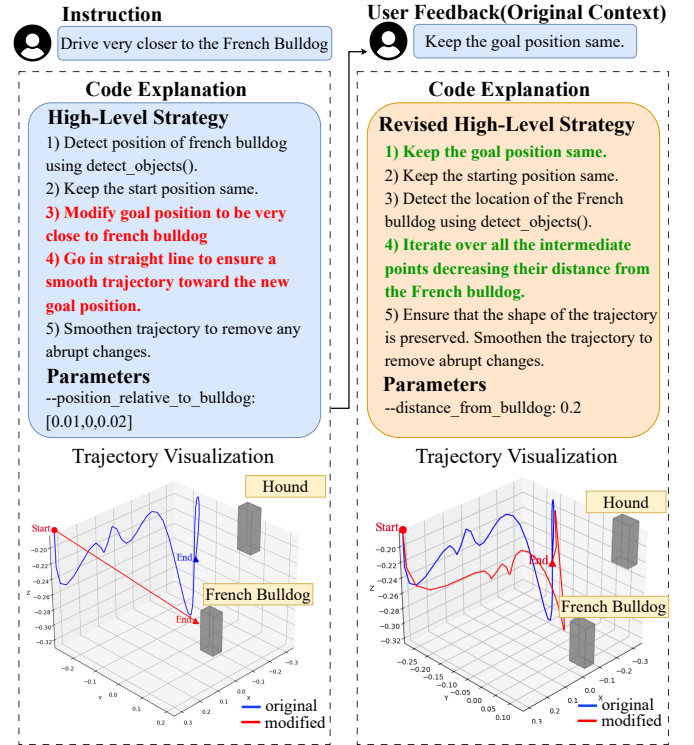


Fig. 4: Visual illustration of our approach to mitigate LLM hallucinations: Interactive trajectory visualization helps identify code errors (red) and enables targeted user feedback for refinement. Code explanations with parameters aid in identifying missteps to produce updated strategies (green)

where $\lambda_{\text{dev}} > 0$ and $\lambda_{\text{smooth}} > 0$ are weights, and D_1 is the block-diagonal first-order difference matrix.

Constraints. We impose the following linear inequality constraints, where G and \mathbf{h} encode workspace and obstacle avoidance terms mentioned in Eq. 3, Eq. 4, Eq. 5 and Eq. 6:

$$G\mathbf{x} \leq \mathbf{h} \quad (2)$$

1) **Workspace constraints:** δ is safety margin parameter.

- *Cuboidal bounds (for drones and ground robots):*

$$x_i^{\min} + \delta \leq \mathbf{x}_t^{(i)} \leq x_i^{\max} - \delta, \quad i \in \{x, y, z\} \quad (3)$$

- *Spherical bounds (for manipulators):*

$$R_{\min}^2 \leq \|\mathbf{x}_t - \mathbf{c}\|_2^2 \leq R_{\max}^2 \quad (4)$$

2) **Obstacle avoidance:** Each obstacle is approximated by a bounding sphere of radius R_{obs} , centered at \mathbf{o} .

$$\|\mathbf{x}_t - \mathbf{o}\|_2^2 \geq R_{\text{obs}}^2 + \delta^2 \quad (5)$$

Spherical workspace and obstacle avoidance constraints are linearized via first-order Taylor expansion around the reference trajectory \mathbf{x}^{ref} . In our pipeline, the reference is the LLM-modified trajectory reflecting user intent. The CSM then refines this path locally for safety, keeping the final trajectory near \mathbf{x}^{ref} . This makes linear constraint approximations both accurate and well-suited.

3) **Velocity constraint:**

$$v_i \leq v_{\max} \quad (6)$$

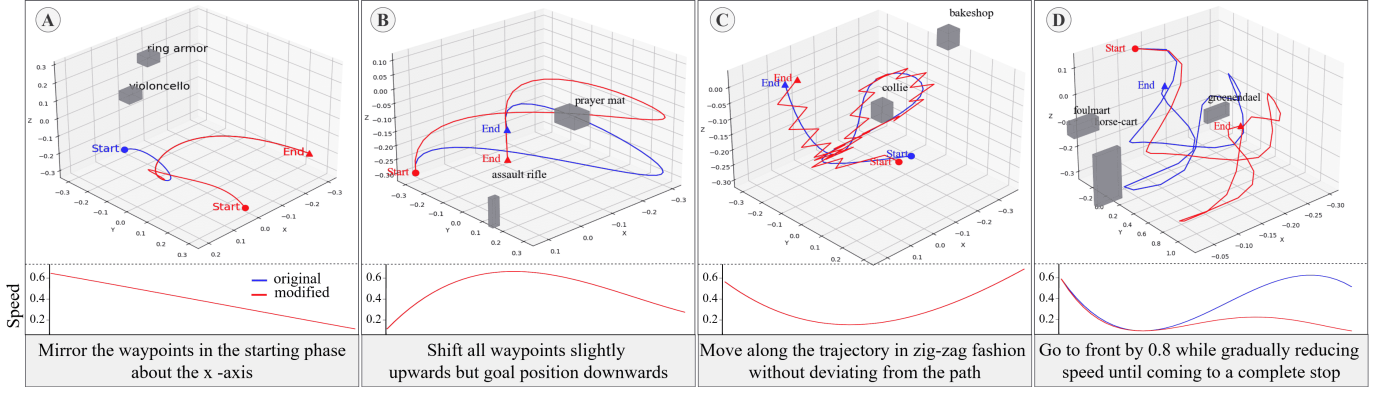


Fig. 5: **Results on open-vocabulary instructions:** The original trajectory is represented in blue, while the modified trajectory using our approach is shown in red.

4) **(Optional) Equality Constraints.** To enforce fixed start and/or goal configurations:

$$Ax = b \quad : \quad x_0 = x_0^{\text{ref}}, \quad x_{T-1} = x_{T-1}^{\text{ref}} \quad (7)$$

QP Formulation We formulate P and Q matrices as

$$P = 2(\lambda_{\text{dev}}I + \lambda_{\text{smooth}}D_1^\top D_1), \quad q = -2\lambda_{\text{dev}}x^{\text{ref}} \quad (8)$$

From Eq 2, Eq 7, and Eq 8, the full QP becomes:

$$\begin{aligned} \min_{x \in \mathbb{R}^{4T}} \quad & \frac{1}{2}x^\top Px + q^\top x \\ \text{s.t.} \quad & Gx \leq h, \quad Ax = b \quad (\text{optional}). \end{aligned}$$

This problem is solved using a standard convex QP solver. λ_{dev} and λ_{smooth} are chosen through Optuna [32] by minimizing the deviation cost. The search employs the Tree-structured Parzen Estimator (TPE) over a logarithmic range 10^{-3} to 10^1 with 50 trials. Our framework’s modularity allows seamless integration of custom constraint modules.

IV. EXPERIMENTS AND RESULTS

This section presents a detailed evaluation of our method. Our experiments address the following key points:

Q1: How can our method handle free-form natural language commands (open-ended, not restricted by predefined syntax or rigid formats) for spatiotemporal manipulation of the waypoints?

Q2: To what extent is user feedback essential in achieving the desired adaptations? Additionally, how does the interpretability of the proposed approach enable an effective understanding of adaptations?

A. Dataset

Each data sample includes an initial trajectory (τ_0), environmental observations (Object Label(O_i), Positions $P(O_i)$, Dimensions $D(O_i)$), an optional linguistic description (E_d), and a natural language instruction (L_{instruct}). Unlike previous methods [29]–[31] that rely on point objects, we represent objects as cuboids for more realistic constraint satisfaction. We use a subset (75 trajectories) of the LaTTe (Language Trajectory Transformer) [30] dataset. The original LaTTe dataset focuses on three instruction types: (1) Cartesian waypoint

changes (e.g., "Go to the left"), (2) speed adjustments (e.g., "Go slower"), and (3) positional shifts relative to objects (e.g., "Drive closer to the sofa"). The dataset has template-based changes over precise numbers and uses single-instruction templates with a fixed vocabulary. To overcome these limitations, we create a subset with open-vocabulary instructions. Specifically, we include complex commands with numerical specifications. (e.g., "Go left by 20"), combined instructions (e.g., "Go to the front by 0.8 while gradually reducing speed until coming to a complete stop"), and long open-ended commands (e.g., "Move along the trajectory in zig-zag fashion without deviating from the path"). Fig. 5 shows examples from the extended subset.

To evaluate our approach to robot deployment under constraints, we curated trajectories from simulated and real-world environments. They include wheeled, aerial, and manipulator robots. The complete dataset includes 230 trajectories: LaTTe subset (75), extended subset (75), and robot subset (80), evaluated across three LLMs (gpt-4o-2024-11-20 [33], claude-3-opus-20240229 [34], and gemini-1.5-pro-002 [35]) with varied feedback.

B. Evaluation Design

Due to the open-vocabulary nature of the instructions and the multiple possible correct adjustments for any given trajectory, designing a comprehensive evaluation metric for all instructions is highly challenging. Metrics like RMSE require a supervised trajectory as a reference, which isn’t feasible for every open-vocabulary instruction. Moreover, as noted by [31], these metrics don’t fully capture the nuances of trajectory adaptations. For instance, two trajectories—one approaching and the other receding from a reference by the same distance—can yield identical RMSE values, yet very different semantic meanings. We resort to a user study to evaluate the results.

We conducted 690 evaluations with 30 participants (15 robotics experts and 15 non-experts). The evaluation consists of two stages. In the first stage, participants receive a trajectory adapted as per dataset instructions. They can refine it with feedback and save the results. In the second stage, a different participant assessed the adapted trajectories, along with feedback and code explanations, to determine their semantic

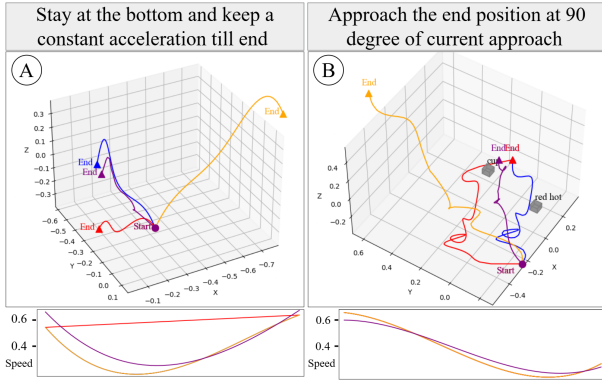


Fig. 6: OVITA, LaTTe, and ExTraCT Trajectory Comparison on extended subset: The initial trajectory is depicted in blue. OVITA, LaTTe, and ExTraCT trajectories are shown in Red, Purple, and Yellow, respectively. Due to overlap, speed plots may not be distinctly visible.

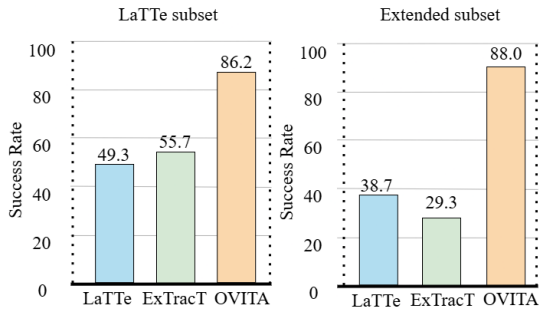


Fig. 7: Performance Comparison between OVITA, LaTTe and ExTraCT on LaTTe subset and extended subset

correctness. This eliminates self-confirmation bias, ensuring an unbiased evaluation. Trajectories were presented in an interactive 3D environment, allowing evaluators to adjust the view. They were assessed on a 5-point Likert scale, from (1) Completely wrong to (5) Completely correct. Participants also rated the usefulness of code interpretations for understanding trajectory adaptation on the same scale (interpretability score). This scale serves as a performance metric for our method, with task success defined as scores of 4 or 5, while scores of 1, 2, or 3 are considered failures.

C. Results and Discussion

Fig.8 illustrates the success rates of trajectories across various dataset subsets, comparing outcomes with and without feedback while Fig. 5 presents the qualitative findings.

Evaluating Q1: Our approach achieved an 81.4% average success rate, excelling on both the extended and Robot subsets (Fig. 8 B), which involve open-vocabulary, freeform, and complex instructions. Notably, the high success rate on the Robot subset confirms our method’s ability to generate constraint-compliant trajectories suited for real-world scenarios. A deeper analysis revealed the following insights:

1) Unlike prior methods focused on transforming existing waypoints, our approach can handle both generation and transformation. For instance, the command “create a spiral in the middle of the trajectory” requires generating new waypoints to form a spiral while maintaining alignment with the original

trajectory (Fig.9). Thus retaining the first half of the trajectory, generating the spiral, and then continuing the original trajectory. 2) Our code-based approach treats waypoints as independent, allowing flexible trajectory modifications. This enables diverse user intents. For example, the instruction “Shift all waypoints slightly upwards but goal position downwards” allows local adjustments to the goal waypoint while globally adapting the other waypoints (Fig.5 B).

Evaluating Q2: We use the Wilcoxon signed-rank test [36] to assess response distribution and report the p-values. Participants noted that the framework better understood their preferences based on feedback ($p < 0.005$), leading to higher success rates across all LLMs (Fig.8 D). Additionally, Participants reported a positive experience with the code-explanation feature ($p < 0.001$), which provided clear insights into trajectory adaptations. This resulted in an impressive average interpretability score of 4.1 out of 5 (Fig.8 E)

Results against Baselines To benchmark against LaTTe and ExTraCT, we evaluated their methods with a similar user study to OVITA. For LaTTe, we employed their best pretrained model with $n_{\text{depth}}=400$ and set the locality factor to 0.5. For ExTraCT, we followed the paper’s settings: a deformation radius of 0.3 and weight w in the range $[0.1, 0.5]$. Participants were not informed about the specific method used to deform the trajectories. Our method outperforms both LaTTe and ExTraCT on both the LaTTe subset ($p < 0.01$) and extended subset ($p < 0.005$ (Fig.7). A qualitative comparison of OVITA vs LaTTe vs ExTraCT can be seen in Fig.6. Note that ExTraCT underperforms compared to LaTTe on the extended subset because it relies on discrete mappings, while LaTTe’s embedding-based architecture may capture parts of complex instructions. The mean inference time, measured over 30 samples, was approximately 12.3 seconds for OVITA, 3.2 seconds for LaTTe, and 0.4 seconds for ExTraCT.

D. Simulation experiments

To evaluate our approach across diverse robot dynamics and environments, we tested it on multiple robots and tasks. Experiments included a Crazyflie drone and a KUKA iiwa LBR 7-DOF robotic arm as shown in Fig.9, demonstrating the framework’s compatibility with various platforms, dynamics, and low-level IK modules. A 3D tabletop environment was created with varying object types and placements. The drone experiments focused on diverse motions in 3D navigation.

E. Real-world experiments

We validated our framework’s executability on real-world robotic systems through experiments using the KUKA iiwa LBR 7-DOF manipulator and the Jackal ground robot, as shown in Fig.10. These trials focused on tasks exploring scenarios such as numerical translation of trajectory points, object-specific trajectory transformation, and multi-step deformation for robot operations. We used a standard CPU/GPU system connected via ROS to execute waypoints. A mounted RGBD camera captures the workspace, and VLM provides an environment description, including object properties (color, shape, orientation, etc.). LangSAM (CLIP (ViT-L/14@336px))

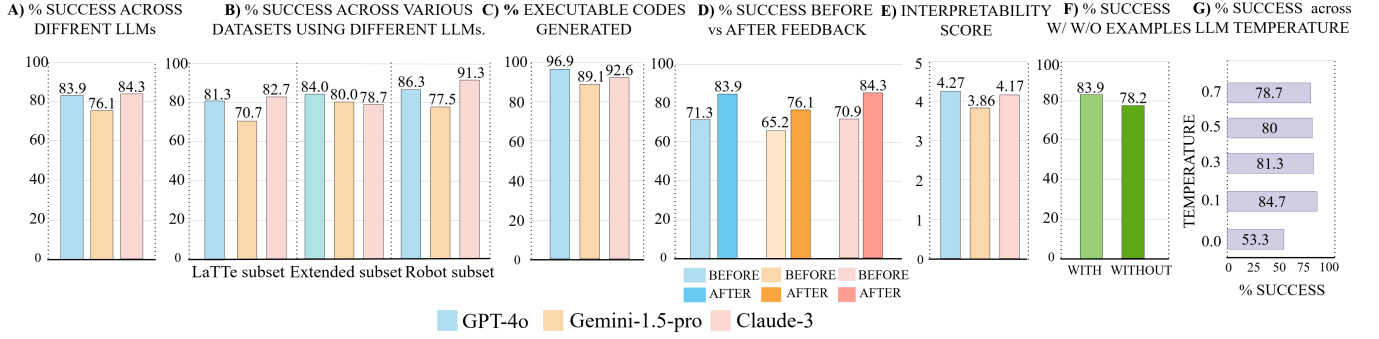


Fig. 8: Evaluation of OVITA: (A) across various LLMs, (B) dataset subsets, (C) executable code generation rates, (D) feedback impact on success, (E) interpretability scores, (F) ablation w/ and w/o examples, and (G) LLM temperature variations

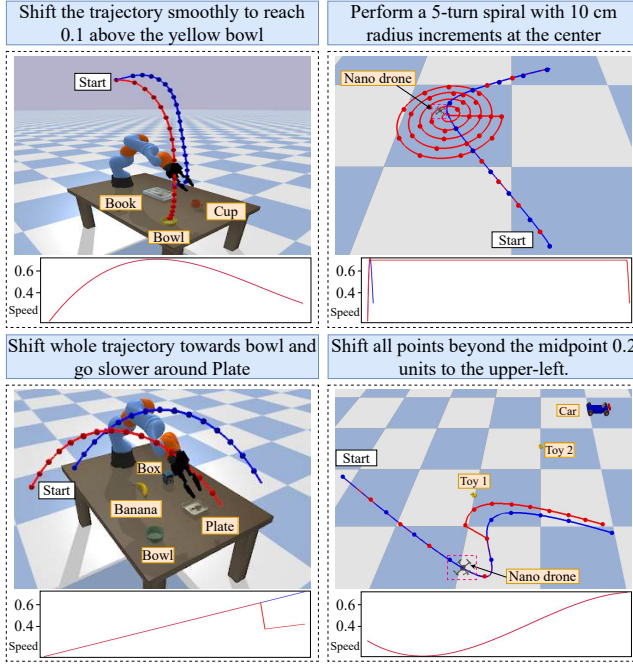


Fig. 9: Simulation results with a robotic arm and drone along with speed profiles. Blue denotes the original trajectory, and red is the modified one.

[37] uses the image and object names to generate masks, which, combined with depth data and intrinsic camera parameters, yield 3D point clouds. From these, object centers, orientations (via PCA), and dimensions are computed in world coordinates.

F. Ablations

We analyze our approach through ablation studies (Fig. 8):

- **LLM Evaluation:** We compare multiple LLMs to identify the best-performing model and assess model dependence. Consistent results across multiple LLMs confirm our approach’s generalizability (Fig. 8 A&B).
- **Impact of Human Feedback:** Incorporating feedback improves success rates across all LLMs, emphasizing its role in performance enhancement (Fig. 8 D).
- **Code Executability:** Our framework generates 92.9% executable code, demonstrating robustness in producing syntactically correct and actionable code. (Fig. 8 C).

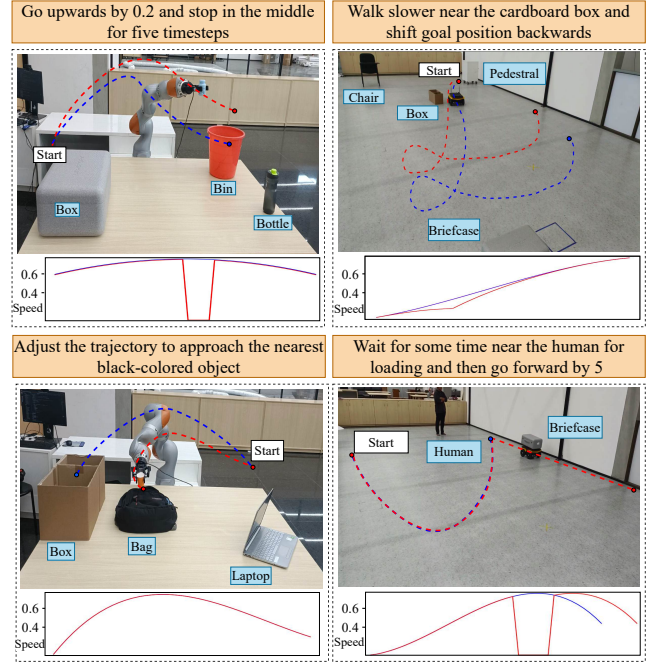


Fig. 10: Real-World Testing on KUKA IIWA and Jackal Robots: Approximate trajectories are shown with original (blue) and modified (red) paths.

- **Interpretability:** High interpretability scores (avg. 4.1) show method transparency and compatibility with both expert and non-expert users. (Fig. 8 E).
- **Impact of example exclusion:** Success rates decline only slightly indicating strong robustness (Fig. 8 F).
- We vary the LLM temperature (0-0.7) and report success rates in Fig. 8 G. Minimal variation shows robustness, with a sharp drop only at temperature 0.

V. CONCLUSION AND LIMITATIONS

OVITA successfully enables intuitive, language-based trajectory adaptation. A closed-loop feedback system ensures quality control and corrects LLM hallucination. The framework can be integrated with any existing waypoint-based planner, making it useful across diverse applications. An interesting application of our approach is natural language-driven trajectory augmentation, improving high-fidelity synthetic data generation.

Our approach, while effective in generating instruction-compliant trajectories, has several limitations. The LLM-generated code may fail due to syntax errors (e.g., incorrect function names), mathematical errors (e.g., division by zero), or deviations from the required function structure. Logical errors, though present, are readily corrected through feedback. While OVITA aligns with user intent, it doesn't guarantee globally optimal paths. Performance depends on precise user input, as vague terms (e.g., "very" or "relatively slower") may be misinterpreted. Additionally, QP-based solvers can fail to find feasible solutions under overly restrictive or conflicting constraints. Future directions includes integrating generation with adaptation, extending to dynamic environments, and incorporating multimodal instructions (e.g., sketching via specialized simulators or AR/VR).

REFERENCES

- [1] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, *et al.*, "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186345, 2024.
- [2] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 2998–3009.
- [3] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.
- [4] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," in *Conference on Robot Learning*. PMLR, 2023, pp. 540–562.
- [5] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, *et al.*, "Language to rewards for robotic skill synthesis," in *Conference on Robot Learning*. PMLR, 2023, pp. 374–404.
- [6] T. Kwon, N. Di Palo, and E. Johns, "Language models as zero-shot trajectory generators," *IEEE Robotics and Automation Letters*, 2024.
- [7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
- [8] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.
- [9] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, no. 1, pp. 297–330, 2020.
- [10] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, J. Kober, *et al.*, "Interactive imitation learning in robotics: A survey," *Foundations and Trends® in Robotics*, vol. 10, no. 1-2, pp. 1–197, 2022.
- [11] S. Anjomshoe, A. Najjar, D. Calvaresi, and K. Främling, "Explainable agents and robots: Results from a systematic literature review," in *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, Montreal, Canada, May 13–17, 2019, 2019, pp. 1078–1088.
- [12] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.
- [13] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.
- [14] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.
- [15] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.
- [16] Y. Huang, L. Roza, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.
- [17] M. R. Montero, G. Franzese, J. Kober, and C. Della Santina, "Learning multi-reference frame skills from demonstration with task-parameterized gaussian processes," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 2832–2839.
- [18] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Thompson, Q. Vuong, T. Yu, *et al.*, "Palm-e: An embodied multimodal language model," in *International Conference on Machine Learning*. PMLR, 2023, pp. 8469–8488.
- [19] A. Szot, M. Schwarzer, H. Agrawal, B. Mazouze, R. Metcalf, W. Talbott, N. Mackraz, R. D. Hjelm, and A. T. Toshev, "Large language models as generalizable policies for embodied tasks," in *The Twelfth International Conference on Learning Representations*, 2023.
- [20] Y. Jin, D. Li, A. Yong, J. Shi, P. Hao, F. Sun, J. Zhang, and B. Fang, "Robotgpt: Robot manipulation learning from chatgpt," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2543–2550, 2024.
- [21] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.
- [22] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.
- [23] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Thompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning*. PMLR, 2023, pp. 1769–1782.
- [24] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 25–55, 2020.
- [25] J. Arkin, D. Park, S. Roy, M. R. Walter, N. Roy, T. M. Howard, and R. Paul, "Multimodal estimation and communication of latent semantic knowledge for robust execution of robot instructions," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1279–1304, 2020.
- [26] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh, "No, to the right: Online language corrections for robotic manipulation via shared autonomy," in *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, 2023, pp. 93–101.
- [27] M. V. Koroteev, "Bert: a review of applications in natural language processing and understanding," *arXiv preprint arXiv:2103.11943*, 2021.
- [28] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox, "Correcting robot plans with natural language feedback," *arXiv preprint arXiv:2204.05186*, 2022.
- [29] A. Buckner, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, and R. Bonatti, "Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 978–984.
- [30] A. Buckner, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, S. Vemprala, and R. Bonatti, "Latte: Language trajectory transformer," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7287–7294.
- [31] J. Yow, N. P. Garg, M. Ramanathan, W. T. Ang, *et al.*, "Extract-explainable trajectory corrections from language inputs using textual description of features," *arXiv preprint arXiv:2401.03701*, 2024.
- [32] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
- [33] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.
- [34] Anthropic, "Claude haiku [large language model]," 2024. [Online]. Available: <https://www.anthropic.com>
- [35] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.
- [36] R. Lowry, *Concepts and Applications of Inferential Statistics*, 2014. [Online]. Available: <http://vassarstats.net/textbook/>
- [37] L. Medeiros, "Langsam: Language segment-anything," <https://github.com/luca-medeiros/lang-segment-anything>, accessed: 2023-10-01.