Diffusion Model for Natural Language Guided Trajectory Planning

Utkarsh Aashu Mishra, Ujjwal Gupta, Naveen Balaji Nagarathinam

Abstract

Planning in robotics is accompanied by certain feasibility and safety constraints. In the growing era of large language models, using user-defined natural language prompts is the best way to formulate such constraints. However, such a formulation is complex and requires finding suitable methods to represent and incorporate such instruction into the planning formulation. In this project, we propose to learn a trajectory planner which finds appropriate modifications needed in a given input trajectory to satisfy the language constraints in a given scene of multiple objects. To achieve this, we employ diffusion model-based methodology to formulate a conditional distribution learning problem over the domain of all feasible trajectory modifications. The constraints are represented by foundation models like CLIP and BERT to generalize across a significantly large set of objects and language prompts. The model outputs distributions of all possible modifications to solve for a particular constraint in a defined scene; hence, suitable solutions can be sampled flexibly. Parallely, we also develop a real-world simulation setup to execute planned trajectories directly on a drone platform. Combining both the proposed segments of planning and execution will lead to a much more generalizable and human-robot interaction-friendly trajectory planning and execution strategy.

1. Introduction

In the rapidly evolving field of robotics, the central challenge lies in enabling robots to carry out a multitude of tasks upon receiving high-level instructions from humans, all while ensuring reliable planning and control in complex environments. Addressing this challenge involves considering specific feasibility and safety constraints in planning, which may stem from structural limitations in the environment or user-defined objectives. Traditional task-based planning methods often require incorporating these constraints directly into low-level planning. This process can be daunting, necessitating domain expertise and potentially hindering human interaction. However, recent progress in parsing language commands using large language models [4,19] has paved the way for converting high-level language instructions into low-level planning constraints [1]. Further, the availability of foundation models like Bidirectional Encoder Representations from Transformers (BERT) [10] for structuring natural language and Contrastive Language-Image Pretrainin (CLIP) [18] for correlating visual and textual inputs has made the scope of generalization quite achievable.

Importance of Natural Language Constraints: Recently, there has been a rise in the applications of language-based guidance in robotics, where suitable prompts have been used as task instructions without incorporating them into the low-level planner. In this context, several representation learning models have been proposed to transform high-level language commands into low-level planning constraints implicitly. In the domain of robotics, constraints can be defined in multiple ways. While the terminal constraints specify goal requirements such as "Place the block inside the red bowl" or "Move to the center of the table," [3,20,21] the trajectory constraints specify several aspects of the waypoint planning such as "Fly near the Bottle" or "Move slowly while passing across the tree." [1,5] In a straightforward approach, the easiest way to define such constraints in using natural language. However, finding suitable representations in this context is challenging.

State of the Art and Limitations: To solve the problem of trajectory planning using natural language instructions, an LLM-based planner LATTE [5] was proposed, which converts the instruction prompt and the information about the objects in the scene into an embedding space by using BERT and CLIP respectively. This embedding vector is used to formulate a conditional transformer [24] decoder to output the modified trajectory, given the input trajectory embeddings from the transformer encoder. This architecture was trained end-to-end using an MSE loss between the predicted output trajectory and that obtained by incorporating the constraints explicitly into a planner. While this approach is promising but logically, there can be many such output trajectories that satisfy the user-defined constraint. Also, with their transformer-based approach, adding additional planning constraints during evaluation is difficult. To overcome these challenges, we propose a new way to represent the collection of all possible output trajectories as distributions from which solutions can be sampled according to any additional heuristic during evaluation.

Our Approach: In modeling such distributions condi-

tioned on several scene and task representations, diffusion models have gained significant attention for their applicability in trajectory planning from imitation data [2, 6, 11]. Additionally, these models have been used to generate target poses for language-conditioned object rearrangement tasks [14, 16]. For a given task, such models have successfully learned the underlying distribution of the positive samples. In this project, we simplify finding the output trajectory by modeling the distribution of the suitable modifications required to transform the initial to the desired trajectory. We use CLIP similarity between the task instruction prompt and the objects present in the scene to find the object of interest and use BERT embedding to represent the whole prompt. We train a diffusion model using the dataset samples and use classifier-free guidance [7] to sample from the learned distribution suitable solutions which satisfy the constraints. Further, we also add real-world simulations to accelerate hardware implementation by implementing a trajectory execution pipeline.

With this approach, generalized natural language-guided trajectory planners can be developed. They can be used to find desired trajectories by incorporating any differentiable cost function or classifier with our pre-trained diffusion model. The simulation setting facilitates rapid execution of our setup for real-world problems backed up by the strong task and scene coverage by the foundation models.

2. Baseline and Dataset

In this section, we describe the baseline LLM based trajectory planner with a transformer backbone, LATTE: LAnguage Trajectory TransformEr [5]. This work motivated our project and is the only prior work that solves the problem we are concerned about. In order to address the challenge of integrating a robot's low-level geometric constraints with high-level semantic instructions from humans, the proposed approach allows users to modify general robotic trajectories through free-form text input and scene images.

LATTE utilizes pre-trained language models (BERT [10] and CLIP [18]) to encode the user's intention and target objects directly from free-form text input and scene images. The encoded information is then combined with geometrical features generated by a transformer encoder network based on the input trajectories. Subsequently, modified trajectories are generated using a transformer decoder. The modified trajectories are preprocessed to comply with safety and dynamic space-state restrictions by adjusting the original waypoint towards the new one, but only if the waypoint is in an admissible region.

Importance of CLIP Similarity and BERT embeddings: The foundation model CLIP enables the extraction of latent embeddings from both the user's text instruction and the images of objects present in the scene. The cosine similarity between these visual and textual embeddings represents potential target objects for the user's command. In addition, BERT embeddings of the instruction prompt represent a semantic feature that captures the contextual meaning of the text, enabling the transformer model to better understand and interpret the user's commands and queries.

Dataset: We use the dataset provided by LATTE containing 100k examples of different scenes, language prompts, and modified trajectories. The dataset consists of a base trajectory $\mathbf{x}^{\text{input}}$ comprising of waypoints $p = \begin{bmatrix} x, & y, & z \end{bmatrix}^\top \in \mathbb{R}^3$ in the world frame, along with velocity $v \in \mathbb{R}$, a natural language prompt L_{in} to define the task, a modified trajectory $\mathbf{x}^{\text{output}}$, and a set of objects $O = O_1, ..., O_M$ represented as central poses $P(O)$ and their corresponding images $I(O)$. The information about the objects is used to construct the scene and the language instruction is used to construct the task of our problem. Also, to fit our project's needs, we only consider positions and disregard the velocity component. Further, we also filter this dataset to use samples that only contain three objects in the scene and do not contain any velocity-modulating conditions. After this, we have 17k samples, out of which we train on 15k samples and test on the rest 2k samples. More information about the baseline and dataset is available at: www.github.com/arthurfenderbucker/LaTTe-Language-Trajectory-TransformEr

3. Background: Diffusion Models

Let us consider that we are given samples $\mathbf{x}$ from an unknown distribution $q(\mathbf{x})$; the motivation of diffusion models [8] is to estimate this distribution as a parameterized distribution $p_\theta(\mathbf{x})$. The way a diffusion model learns this distribution is divided into two steps: forward noising and reverse denoising.

Forward Diffusion: This process continuously injects gaussian noise into the samples $\mathbf{x}$ for K diffusion steps following a fixed variance schedule, denoted as $\beta_0, \ldots, \beta_{K-1}$ (standard notation as in [22]). The true samples for this distribution at $k = 0$ are the provided $\mathbf{x}$ and is represented as $\mathbf{x}_0$ Following this, we get a Markov chain such as:

$$q(\mathbf{x}_{1:K}|\mathbf{x}_0) = \Pi_{k=0}^{K-1} q(\mathbf{x}_{k+1}|\mathbf{x}_k)$$

where the individual noising is according to a gaussian normal distribution $q(\mathbf{x}_{k+1}|\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_{k+1}; \sqrt{1-\beta_k}\mathbf{x}_k, \beta_k\mathbf{I})$. Now at every diffusion step, a similar distribution can be formulated for $q(\mathbf{x}_{k+1}|\mathbf{x}_0)$ that is

$$q(\mathbf{x}_{k+1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{k+1}; \sqrt{\alpha_k}\mathbf{x}_0, (1-\alpha_k)\mathbf{I})$$

where $\alpha_k = \Pi_{i=0}^{k}(1-\beta_k)$. This calculation of alpha makes sure that $\alpha_K \approx 0$ and hence $\mathbf{x}_K \sim \mathcal{N}(0, \mathbf{I})$, a normal distribution with 0 mean. The fact that the final samples after the forward diffusion resemble a normal distribution with 0 mean and identity variance, it is easy to start the reverse

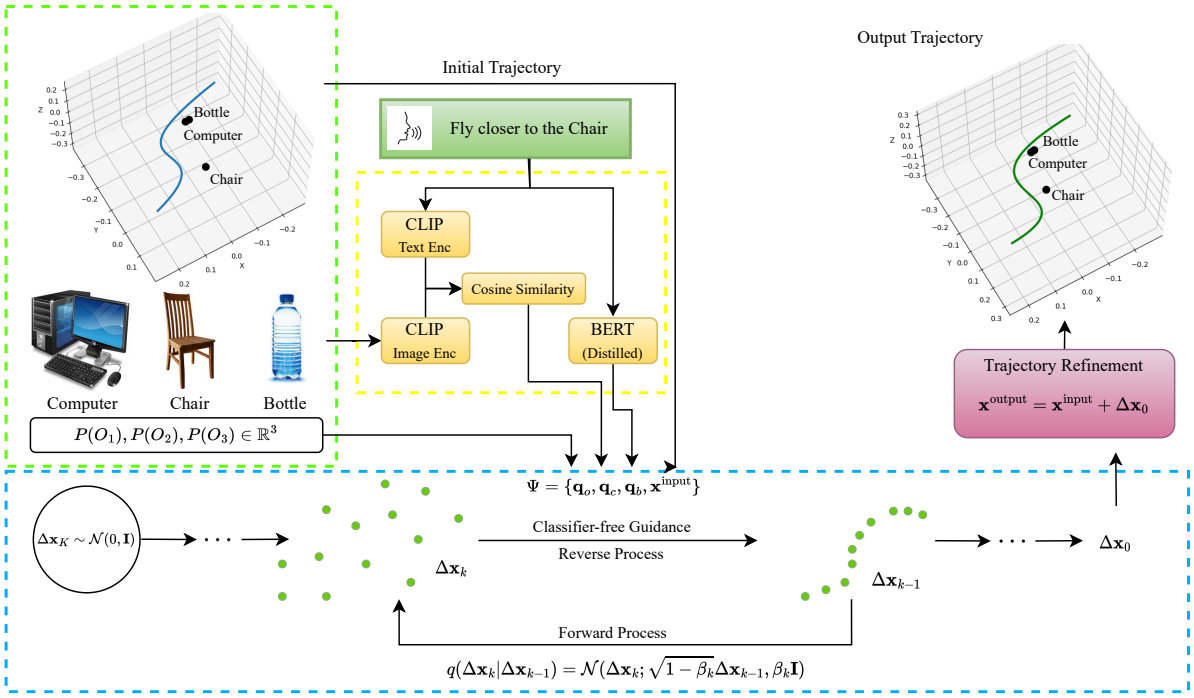

Figure 1. Schematic of our proposed pipeline: The formulation is provided with an initial input trajectory, the scene description as the poses and images of the objects present in the scene, and a natural language instruction. The yellow nodes are frozen and a scene-task embedding is constructed using the inputs. This embedding conditions the diffusion model to generate suitable modifications which are added to the input trajectory to obtain the output trajectory that satisfies the user-defined language constraint.

diffusion process while sampling from a learned diffusion model.

Reverse Diffusion: Keeping the variance schedule fixed, this step learns to reconstruct the diffused samples back to the original distribution and hence tries to formulate gradual denoising by following the formulation $p_\theta(\mathbf{x}_k|\mathbf{x}_{k+1}) = \mathcal{N}(\mathbf{x}_k; \mu_\theta(\mathbf{x}_{k+1}, k+1), \beta_{k+1}\mathbf{I})$, where μ_θ is the mean of this reverse gaussian distribution and it is given by:

$$\mu_\theta(\mathbf{x}_k, k) = \frac{1}{\sqrt{1-\beta_k}}\left(\mathbf{x}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\epsilon_\theta(\mathbf{x}_k, k)\right).$$

In the above formulation, the model $\epsilon_\theta(\mathbf{x}_k, k)$ is popularly coined as the score-function which predicts the perturbations and the noising schedule of the forward process by optimizing the denoising score-matching objective [23]

$$\arg\min_\theta \mathbb{E}_{\mathbf{x}_0 \sim q, \epsilon \sim \mathcal{N}(0,\mathbf{I})}\left[\|\epsilon - \epsilon_\theta(\sqrt{\alpha_k}\mathbf{x}_0 + \sqrt{1-\alpha_k}\epsilon, k)\|^2\right]$$

Theoretically, $\epsilon_\theta(\mathbf{x}_k, k)$ represents the gradients of the log probability of the learned distribution as

$$\nabla_{\mathbf{x}_k} \log p_\theta(\mathbf{x}_k) = -\frac{1}{\sqrt{1-\alpha_k}}\epsilon_\theta(\mathbf{x}_k, k).$$

4. Methodology

This section presents the problem formulation and outlines our approach for trajectory shifting based on the diffusion model. We also discuss the trajectory refinement process.

4.1. Problem Definition

Traditional trajectory planners can easily provide a collision-free trajectory $\mathbf{x}^{\text{input}}$, for given initial and final waypoint, considering the objects present in the scene. Now, given information about the input trajectory and the images $I(O)$ and poses $P(O)$ of the objects, our task is to satisfy a user-defined natural language constraint (L_{in}). We illustrate the given information and language prompt in Figure. 1 (green box and green block respectively). The proposed approach aims towards finding suitable modifications $\Delta\mathbf{x}$ as a diffusion model-based learned distribution (blue box) which when added to $\mathbf{x}^{\text{input}}$ and when passed through robot aware trajectory refinement model (purple block), given an output trajectory $\mathbf{x}^{\text{output}}$, satisfying the desired constraints.

4.2. Trajectory as Distributions

Motivated by the approach shown by LATTE, we aim towards finding the required shifts in the initial trajectory to

satisfy the language-based constraints. While LATTE used transformers to generate the output trajectory sequentially, we plan to learn the necessary deviations from the initial trajectory to get the final trajectory. Such a formulation validates our motivation and allows us to learn a distribution of such deviations using diffusion models. Necessary or sufficient deviations can be flexibly sampled from such a learned distribution [11] $p(\Delta\mathbf{x}|\mathbf{x}^{\text{input}})$, given the input trajectory. In such a case the final output trajectory can be simply formulated as $\mathbf{x}^{\text{output}} = \mathbf{x}^{\text{input}} + \Delta\mathbf{x}$

4.3. Conditional Trajectory Sampling

As discussed before, we work under the same formulation as proposed by LATTE where we have to shift the trajectory in a way that satisfies a given language prompt in the provided scenario consisting of multiple objects. The goal of the diffusion model is to formulate a distribution of appropriate shifts conditioned on the scene and the task information. We use the scene information by the cosine similarity between the CLIP features of images of the objects in the scene and language instructions. We augment this embedding with the BERT embeddings of the task (language) instruction to jointly represent the scene-task embedding vector [5], denoted as Ψ. Hence, we now use the diffusion model to learn the conditional distribution $p(\Delta\mathbf{x}|\mathbf{x}^{\text{input}}, \Psi)$. It is easy to extend the final sampling strategy to any differentiable cost function $\mathcal{C}(\mathbf{x})$ by flexibly formulating the modified distribution as:

$$\Delta\mathbf{x} \sim p(\Delta\mathbf{x}|\mathbf{x}^{\text{input}}, \Psi)h(\mathbf{x}^{\text{input}} + \Delta\mathbf{x})$$

where h is a distribution heuristic given by $h(\mathbf{x}) = \exp(-\mathcal{C}(\mathbf{x}))$ [11]. An exponential distribution helps in formulating the cumulative score of the modified distribution as

$$\epsilon_\theta(\Delta\mathbf{x}) \propto \nabla_{\Delta\mathbf{x}} \log \Big[p(\Delta\mathbf{x}|\mathbf{x}^{\text{input}}, \Psi)h(\mathbf{x}^{\text{input}} + \Delta\mathbf{x})\Big]$$

$$= \nabla_{\Delta\mathbf{x}} \log p(\Delta\mathbf{x}|\mathbf{x}^{\text{input}}, \Psi) + \nabla_{\Delta\mathbf{x}} \log h(\mathbf{x}^{\text{input}} + \Delta\mathbf{x})$$

where

$$\nabla_{\mathbf{x}} \log h(\mathbf{x}) = -\nabla_{\mathbf{x}}\mathcal{C}(\mathbf{x})$$

4.4. Diffusion Model Score Function and Inpainting

We learn a conditional distribution and since we cannot have explicit classifiers to model the conditions, we follow classifier-free guidance [9] formulation to sample from a diffusion model trained by following the score-matching objective [23]. For such a trained diffusion model with a score function $\epsilon_\theta(\Delta\mathbf{x}_k, k, \mathbf{x}^{\text{input}}, \Psi)$, the final score for the diffusion step k is obtained as:

$$\hat{\epsilon}_k = (1 + w)\epsilon_\theta(\Delta\mathbf{x}_k, k, \mathbf{x}^{\text{input}}, \Psi) - w\epsilon_\theta(\Delta\mathbf{x}_k, k, \emptyset, \emptyset)$$

To solve preliminary problems we do not consider additional heuristics and the final reverse diffusion can be done by first sampling $\Delta\mathbf{x}_K \sim \mathcal{N}(0, \mathbf{I})$ and then following the reverse diffusion formulation as explained in Section 4.1.

We take advantage of the inpainting approach [15] while sampling by making sure that the first and the last shifts in $\Delta\mathbf{x}$ are zero and thus the initial and final point of the modified trajectory is the same as the input trajectory.

4.5. Trajectory Refinement

After obtaining the modified trajectory $\mathbf{x}^{\text{output}}$ by sampling from the learned distribution, we refine it using a spline-based technique to ensure smooth and precise flight paths. To achieve this, the given waypoints

$$\mathbf{x} = \{\mathbf{p}_i = [x_i, y_i, z_i], \quad i \in [1, 40]\}$$

are fitted into a spline-based trajectory optimization function characterized by a third-order polynomial as follows with time, $t \in [0, 1]$:

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

The fitted coefficients, a_0, a_1, a_2, and a_3, are then used to obtain the velocity and acceleration profiles as $v(t) = a_1 + 2a_2 t + 3a_3 t^2$ and $a(t) = 2a_2 + 6a_3 t$ respectively. These profiles are then transmitted to the drone's trajectory tracking loop, enabling the drone to adhere to a seamless and accurate trajectory while maintaining the desired speed and stability.

We use the python Scipy [25] interpolation solver to do the polynomial fitting. Such an approach can accommodate a variety of constraints on the system, such as maximum thrust or velocity limits, and can be adjusted to tackle uncertainties or environmental disturbances. Throughout the entire trajectory, the drone's yaw setpoint remains zero. It is worth to be noted that LATTE [5] uses a waypoint-based optimization process without incorporating the velocity and acceleration profiles. Consequently, this limits the ability to enforce thrust and orientation constraints effectively.

4.6. Limitations over the current Dataset

Since we constrain our training procedure to a lower number of samples as compared to LATTE, it is highly likely that the diffusion model overfits to the set of trajectories following a particular language instruction pattern. Further, the constraint representation used by LATTE might not be the best representation for our diffusion model-based approach. With proper fine-tuning, we can achieve better efficiency in learning the underlying conditional distributions. In terms of execution, the waypoints and the corresponding trajectory generation are heavily dependent on the robot it is executed on, thus requiring heavy post-processing. A more generalized framework development can be an interesting future work direction.

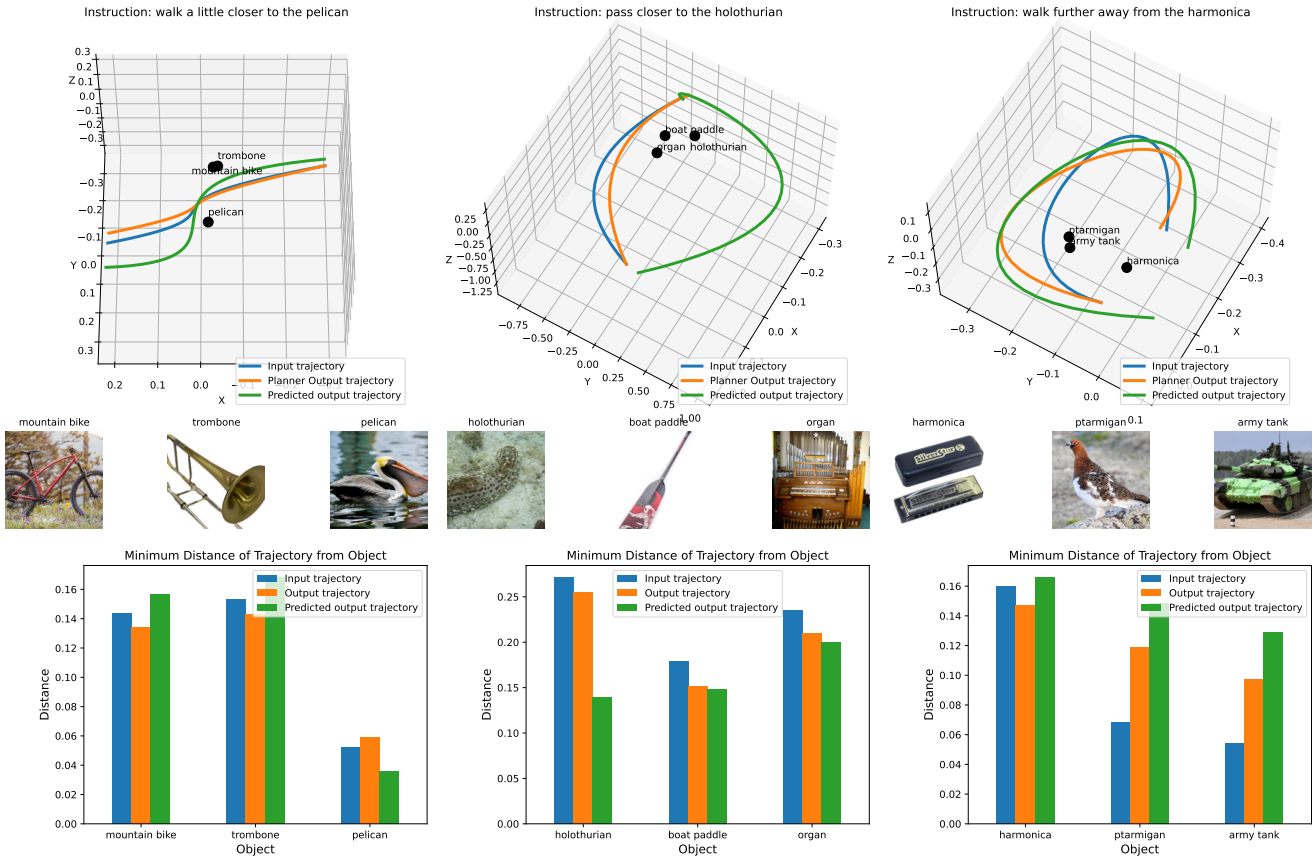

Figure 2. The above figure depicts the performance of the proposed diffusion model-based trajectory modification where the language instruction is mentioned at the top and the objects present in the scene are shown. The 3D-plot depicts the visual performance as compared to the motion planner output trajectory after adding the constraint while the bar plot numerically validates the modified trajectory.

5. Experimental Setup

For each sample in the dataset derived from the one provided by LATTE, we calculated the CLIP embeddings using the 'ViT/L-14' model provided by Open AI, and for the BERT embeddings, we use pre-calculated ones from the provided dataset i.e. from the BERT (distilled) model. Both these are pre-calculated and fixed throughout the training process. The scene-task representation is structured by augmenting the CLIP similarity $q_c \in \mathbb{R}^3$, BERT embeddings $q_b \in \mathbb{R}^{768}$, and the object poses, $q_o \in \mathbb{R}^{3\times3}$. The final conditioning vector consists of the input trajectory $\mathbf{x}^{\text{input}} \in \mathbb{R}^{40\times3}$ and $\Psi = \{q_o, q_c, q_b\}$. All trajectories are flattened and the diffusion model parameter is a one-dimensional vector $\Delta\mathbf{x} \in \mathbb{R}^{120}$. The score function is a standard MLP network following the UNet structure with a linear embedding layer (512), 3 downsampling (512, 256, 128), 2 intermediate (128, 128) and 3 upsampling (128, 256, 512) and a final embedding layer (120). The diffusion step embedding is represented by a sinusoidal pose embedding and augmented with every layer of the score function. The score function is trained to optimize the score matching loss using a MSE loss. All the trainings are done for 200 epochs on a Nvidia RTX A6000 GPU with a batch size of 256 in approximately 2.5 hours. We base our diffusion model formulation based on this public source: www.github.com/UtkarshMishra04/ReorientDiff/tree/code

5.1. Simulation Environment

Our objective is to incorporate the diffusion model for UAV waypoint modifications that can be seamlessly integrated into real-world UAV systems. To this end, PX4-based flight controller units (FCUs) are a popular choice among researchers and industry professionals due to their open-source community. Therefore, to validate our proposed method, we selected Gazebo along with the PX4 autopilot as our UAV simulation platform. It is worth noting that in practice, a PX4-based FCU with ROS is commonly used for deployment on hardware [12, 17], making the PX4-ROS Gazebo simulation a more realis-

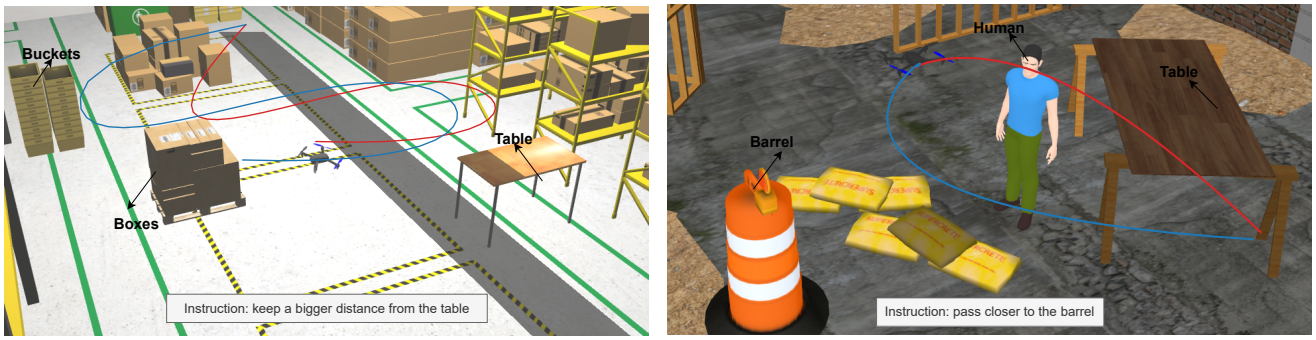

Figure 3. The figure shown above represents the real-world simulation in Gazebo where the modified trajectory (shown in blue) is compared against the original trajectory (shown in red) of the PX4-enabled drone.

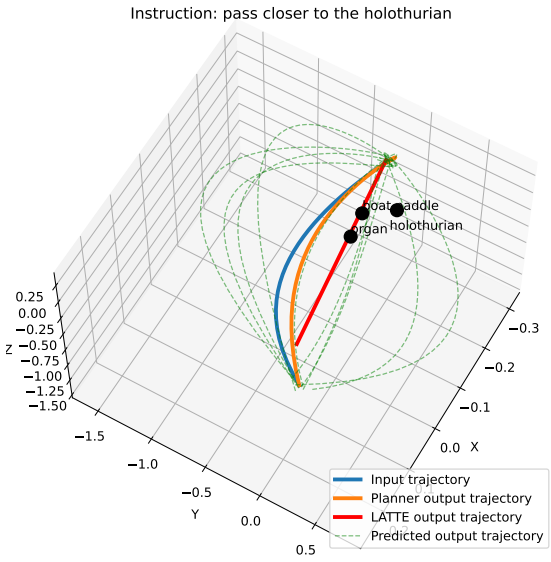

Figure 4. The above figure illustrates the advantage of learning a distribution as compared to learning a single output model. The learned distribution can be used to sample trajectories from which certain other constraints can be satisfied during evaluation.

tic choice for validation. For our simulations in Gazebo, we opted for two different worlds: Amazon's warehouse world called *aws-robomaker-small-warehouse-world*, and a construction site world. These choices were made because they represent realistic scenarios where drones could be deployed and interact with various objects in their environment. We derive our simulation setup from: `www.github.com/aws-robotics/aws-robomaker-small-warehouse-world` and `www.github.com/clearpathrobotics/cpr_gazebo`

5.2. Waypoint Execution Pipeline

In order to facilitate trajectory tracking or waypoint navigation, the PX4 autopilot controller uses a standard PID cascaded control architecture, which allows it to accept various setpoints, including reference positions, velocities, accelerations, yaw, thrust, and body rates. As discussed in the trajectory refinement section, we only provide positions, velocities, and accelerations with zero yaws as setpoints. To achieve this, we set the PX4 autopilot to offboard flight mode and sent the position and yaw setpoints at a rate of 100 Hz through MAVROS, a ROS package that facilitates communication between computers running ROS and PX4-enabled autopilots.

6. Performance Evaluation

We evaluate the performance of our diffusion model by visualizing the quality of the sampled trajectories and numerically analyzing the distance metric to validate that the constraint satisfaction is reflected. We show both these results along with the scene and task description in Figure. 2 and also provide additional results in Figure. 6 in the supplementary.

As LATTE is trained end-to-end to predict the trajectory of the motion planner with the constraints, it is evaluated on the MSE loss between the predicted and the ground truth modified waypoints. This allows us to consider the output trajectory performance equivalent to the LATTE performance, which is depicted as the planar output trajectory in Figure. 2. Hence, the bar plots in Figure. 2 can be used to compare the proposed method's performance with our baseline algorithm. Out of the 2k test samples, the solutions sampled from the learned distribution have been 58% successful in satisfying the user-defined constraints.

The primary advantage of modeling distributions is shown in Figure. 4 where 10 trajectories are sampled from the learned distributions and all of them satisfy the user-defined constraint. It is worth to be noted that these trajectories are robot agnostic and hence, any suitable solution can be sampled favorably based on the robot framework.

Simulation Experiments: Our proposed method is evaluated in two different Gazebo environments, as described in the section on simulation environment. Each environment includes three labeled objects that the drone must navigate

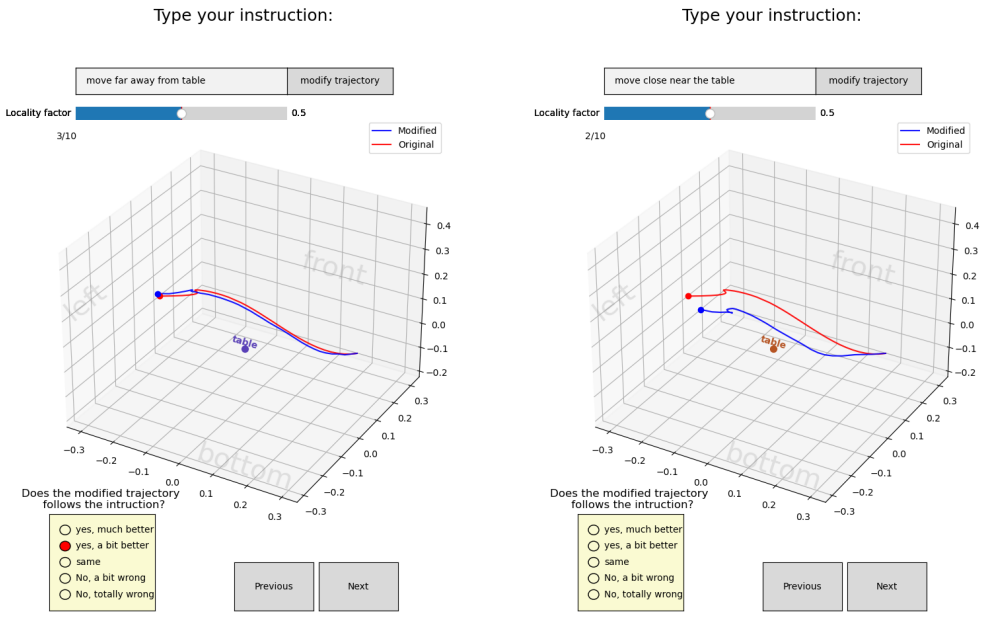

Figure 5. User study GUI setup to visualize the generated trajectories given the natural language prompt

around. Given the initial trajectories that pass through these objects, our model modifies the trajectories based on the user's language commands. As shown in Figure 3, the diffusion model successfully changes the trajectories to align with the user's intent, such as shifting the trajectory away from the table in the first scenario and making the drone pass closer to the construction barrel in the second scenario. It should be noted that the refined trajectories were sent to the drone as setpoints for trajectory tracking. A reference video is provided here: www.youtube.com/watch?v=9XObTJx06WQ

7. User Study

To extend the validation of our proposed methodology, we prepared to use the graphical user interface (GUI) proposed by LATTE [5] model for conducting a user study to analyze the quality of the generated trajectories given the natural language prompt for human-robot interaction. Figure 5 shows the GUI setup where users can input prompts, such as "move far away from object." The proposed setup generates new trajectories in real-time based on user constraints, highlighting the potential for adapting robot behaviors using language models.

The user can observe and compare how language prompts influenced the waypoints generated by the model, and then rate the effectiveness of the approach. Further, the participants can be asked to evaluate trajectory changes generated by five different approaches within a 1-5 Likert scale [13], considering various aspects of human-robot interaction. The results of such a study will provide valuable insights into the practicality and utility of natural language prompts for trajectory generation in real-world scenarios. In future work, we plan to complete such a study.

8. Conclusion

This project introduces a novel approach to robotics trajectory planning using natural language constraints. By employing a diffusion model-based methodology and leveraging CLIP similarity and BERT embeddings, we model the distribution of suitable trajectory modifications. Additionally, we develop a real-world simulation setup for drone execution. The proposed method enables the creation of generalizable, natural language-guided trajectory planners and accommodates various applications by incorporating differentiable cost functions or classifiers. The simulation environment allows easy execution and adaptation for real-world problems, while a new method for representing output trajectories as distributions overcomes existing challenges.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1

[2] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional gen-

erative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022. 2

[3] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020. 1

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[5] Arthur Bucker, Luis Figueredo, Sami Haddadin, Ashish Kapoor, Shuang Ma, Sai Vemprala, and Rogerio Bonatti. Latte: Language trajectory transformer, 2022. 1, 2, 4, 7

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. 2

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 2

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 2

[9] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 4

[10] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, 34(10):1296–1313, 2015. 1, 2

[11] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022. 2, 4

[12] Bhargav Jha, Ujjwal Gupta, Vladimir Turetsky, and Tal Shima. Game theoretic approach for robust trajectory tracking by a multirotor. *IEEE Transactions on Control Systems Technology*, 31(3):1435–1442, 2023. 5

[13] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. Likert scale: Explored and explained. *British journal of applied science & technology*, 7(4):396, 2015. 7

[14] Weiyu Liu, Tucker Hermans, Sonia Chernova, and Chris Paxton. Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects. *arXiv preprint arXiv:2211.04604*, 2022. 2

[15] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022. 4

[16] Utkarsh A Mishra and Yongxin Chen. Reorientdiff: Diffusion model based reorientation for object manipulation. *arXiv preprint arXiv:2303.12700*, 2023. 2

[17] Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022. 5

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1, 2

[19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[20] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022. 1

[21] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022. 1

[22] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 3, 4

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[25] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020. 4

Student Name	Contributed Aspects	Details
Utkarsh Aashu Mishra	Preprocessing scrapped dataset, Implementation, Training, and Evaluation of the Diffusion Model	Changed the CLIP model for pre-processing in the acquired dataset, and regenerated the filtered dataset. Implemented and completed training of the diffusion model for the trajectory modification, and formulated the evaluation pipeline for analyzing and comparing the performance with the ground truth data.
Ujjwal Gupta	Real-World simulation pipeline; Analysis and Execution in Gazebo-ROS	Created the execution pipeline in ROS along with the trajectory refinement module. Implemented the drone controller pipeline.
Naveen Balaji Nagarathinam	Evaluated LATTE model against proposed approach and setup user study	Evaluated the baseline LATTE model and analyzed the LATTE user study to extend it to our setup.

Table 1. Contributions of team members.

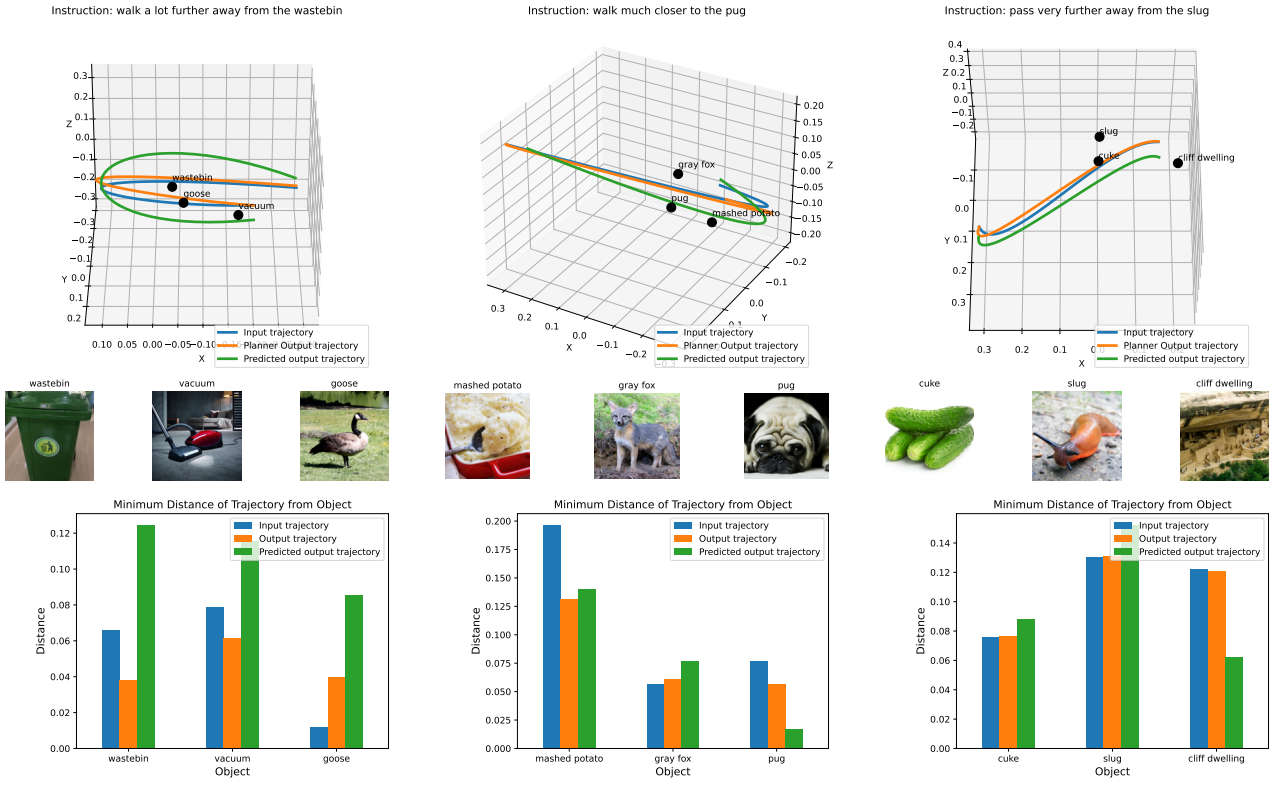

Figure 6. Additional Results for the performance of the proposed methodology visually (3D plot) and numerically(bar chart) as compared to the motion planner output with the added constraints.