DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Mechatronics and robotics

Multimodal Trajectory Refiner Using Natural Language Interventions

Arthur Fender Coelho Bucker

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Mechatronics and robotics

Multimodal Trajectory Refiner Using Natural Language Interventions

Multimodaler Trajektorienverfeinerer durch Natürlichsprachliche Eingriffe

Author:	Arthur Fender Coelho Bucker
Supervisor:	Sami Haddadin
Advisor:	Luis Figueredo
Submission Date:	13/10/2022

I confirm that this master's thesis in mechatronics and robotics is my own work and I have documented all sources and material used.

Munich, 13/10/2022 Arthur Fender Coelho Bucker

Acknowledgments

First and foremost, I would like to thank my advisor Luis Figueredo for his guidance, encouragement and friendship. Thank you for closely orienting me during my thesis and for the great discussions that led to this work. In addition to that, thank you for trusting me and giving me the freedom to explore new collaborations and new ideas during the research process.

Furthermore, I am extremely grateful to Rogerio Bonatti for being such an amazing mentor and friend throughout the past few years. I am extremely inspired by his excellence and inventiveness towards the robotics field. His attentive insights guided and motivated me throughout my professional and academic career.

I am grateful to all the members from the Munich Institute of Robotics and Machine Intelligence (MIRMI) for creating such a pleasing work environment and filling my days in the lab with joys and motivation. A special thanks to Luis Figueredo, Sabrina Jegust, Rachele nebbina, Rafael Muchacho, Wenxi Wu, Rhiddi Laha, Amartya Ganguly. I would also like to express my gratitude to my professor Dr. Sami Haddadin for making this institute possible and for providing the resources and thoughtful insights that make researching at MIRMI a great experience.

I also want to thank the team from Microsoft Buisness AI - Research and Science (US). I own a special thanks to Ashish Kapoor, Shuang Ma, Sai Vemprala and Rogerio Bonatti for the invaluable discussions during our research collaboration and development of my thesis.

I am extremely grateful to *Fundação Estudar* Leaders program scholarship for the financial support throughout my masters studies. Moreover, I am thankful for introducing me into such an inspiring and ambitious community of people that always motivates me to dream higher.

I am thankful to the university of São Paulo for providing me the opportunities of studying abroad through the Double Degree program and also for supporting me with the AUCANI merit scholarship.

I am forever grateful for my family. I could not have asked for better parents, Sueli Fender Coelho Bucker and Heitor Bucker, or better sister, Bárbara Fender Coelho Bucker. Thank you all for showing endless support, encouragement, guidance and love throughout my journey. Thank you all for always believing in me, and being with me every step of the way.

Finally, I am extremely thankful for all my friends were by my side during my studies in Germany. Thanks for the laughs, fun and experiences that made my days ever better.

Abstract

Natural language is one of the most intuitive ways to express human intent. However, translating instructions and commands towards robotic motion generation and deployment in the real world is far from being an easy task. The challenge of combining a robot's inherent low-level geometric and kinodynamic constraints with a human's high-level semantic instructions is traditionally solved using task-specific solutions with little generalizability between hardware platforms, often with the use of static sets of target actions and commands. This work instead proposes a flexible language-based framework that allows a user to modify generic robotic trajectories. The proposed method leverages pre-trained language models (BERT and CLIP) to encode the user's intent and target objects directly from a free-form text input and scene images, fuses geometrical features generated by a transformer encoder network, and finally outputs trajectories using a transformer decoder, without the need of priors related to the task or robot information. The system was evaluated in a diverse set of scenarios and robot form factors, such as manipulation, aerial and legged robots. Simulation and real-life experiments demonstrate that the proposed transformer model can successfully follow human intent, modifying the shape and speed of trajectories within multiple environments. Furthermore, the approach was validated trough user studies both in virtual and real-world scenarios. The results show that users significantly prefer the proposed natural language interface over traditional methods such as kinesthetic teaching or cost-function programming.

Kurzfassung

Die natürliche Sprache ist eine der intuitivsten Möglichkeiten, menschliche Absichten auszudrücken. Das Übersetzen von Anweisungen und Befehlen zur Erzeugung von Roboterbewegungen und deren Einsatz in der realen Welt ist jedoch alles andere als eine einfache Aufgabe. Die Herausforderung, die einem Roboter innewohnenden, sich auf niederer Stufe befindlichen geometrischen und kinodynamischen Beschränkungen mit den semantischen Anweisungen des Menschen auf hoher Ebene zu kombinieren, wird traditionell mit aufgabenspezifischen Lösungen überwunden. Diese können kaum zwischen verschiedenen Hardwareplattformen verallgemeinert werden, wobei häufig statische Sets von Zielaktionen und -befehlen Verwendung finden. Stattdessen wird in dieser Arbeit ein flexibles sprachbasiertes Framework vorgeschlagen, das es dem Benutzer ermöglicht, generische Robotertrajektorien zu ändern. Die vorgeschlagene Methode nutzt vortrainierte Sprachmodelle (BERT und CLIP), um die Absicht des Benutzers und die Zielobjekte direkt aus einer Freiform-Texteingabe und Szenenbildern zu kodieren, fusioniert geometrische Merkmale, die von einem Transformator-Encoder-Netzwerk generiert werden, und gibt schließlich Trajektorien unter Verwendung eines Transformator-Decoders aus, ohne dass vorausgehende Konfigurationen in Bezug auf die Aufgabe oder Roboterinformationen erforderlich sind. Das System wurde in einer Vielzahl von Szenarien und Roboterformen, wie z.B. Manipulations-, Luft- und Laufroboter, evaluiert. Simulationen und reale Experimente zeigen, dass das vorgeschlagene Transformatormodell erfolgreich den menschlichen Absichten folgen kann, indem es die Form und Geschwindigkeit von Trajektorien in verschiedenen Umgebungen modifiziert. Darüber hinaus wurde der Ansatz durch Nutzerstudien sowohl in virtuellen als auch in realen Szenarien validiert. Die Ergebnisse zeigen, dass die Benutzer das vorgeschlagene natürlichsprachliche Interface gegenüber traditionellen Methoden wie kinästhetischem Unterrichten oder Kostenfunktionsprogrammierung deutlich bevorzugen.

Contents

1. Introduction

Robots are increasingly working in proximity to humans, sharing living and working spaces. Within this context, it is of high importance for the robotics community to research techniques that allow robots to robustly operate in such environments. Notwithstanding, despite the recent advances in hardware and robot planning and control capabilities, full autonomy will remain out of reach for the foreseeable future [1]. Relying on full autonomy becomes even more challenging when robots are faced with the uncertainty of the real-world, and need to adapt to changes in the environment or a scene that goes off the originally planned script [2]. One way in which this challenge can be addressed is through shared autonomy and by leveraging human context awareness, understanding, and intent of the scene and objects [3]. The teamwork between humans and robots however strongly depends on a paradigm change that integrates multimodal perception and communication and natural human interaction into online robot's action and trajectory adaptation. In other words, one way to address the uncertainties of the human world is to equip robots with capabilities to seamlessly interact with human users and integrate multimodalities into robot actions.

This work focuses on one important facet of human-robot interaction: given a user's intent and a cluttered unstructured environment, how can a robot best generate or online adapt a trajectory in order to respect human preferences and context understanding while tending to geometrical and dynamics constraints in its surroundings?

This Chapter gives an introduction to the topic. The motivation with exemplary application areas is presented in Section 1.1. In Section 1.2, an overview of the proposed method is given and in Section 1.3 the main contributions of this work are stated. Finally, in Section 1.4 the organization of the following Chapters is outlined.

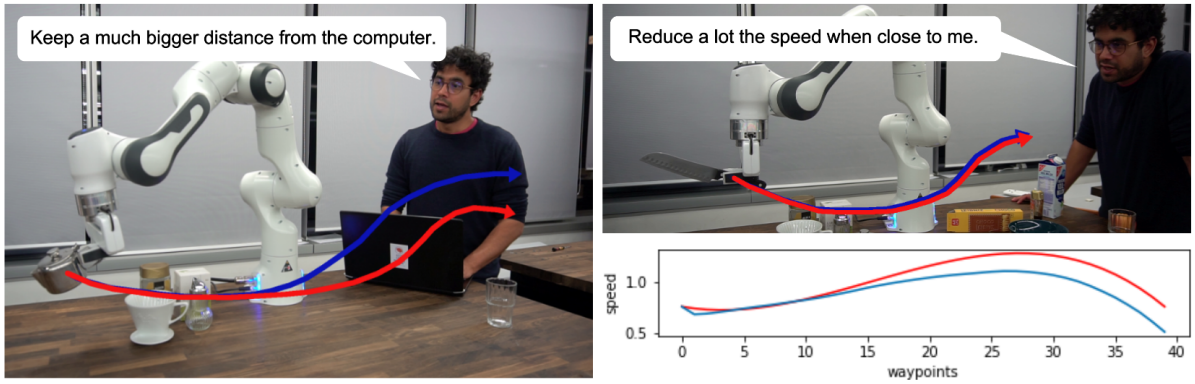

Figure 1.1.: Trajectory reshaping obeying user's intent. The proposed method fuses natural language commands, images of the environment, and geometrical data to generate the modified robot's trajectory. An approximated representation of the original trajectory is shown in red, while the modified one in blue. Full videos in the supplementary material.

1.1. Motivation

Robots of today are still largely pre-programmed for specific tasks and have very limited capability to operate and adapt to new contexts among unstructured human-centered environments. Ideally, in such scenarios, the robot should have the ability to recognize and understand natural language commands in a given context and map them to the task-domain space – where tasks and constraints are largely influenced by context, intent and affordances with objects [3]. This paradigm shift deviates from traditional motion planning, and requires methodologies that are able to integrate multi-modal inputs coming from perception systems (for instance user-provided language commands and robot vision) together with geometrical information to shape robot trajectories towards the desired human intent.

Take for instance the human-robot co-existence and interaction scenario depicted in Figure 1.1, where the robotic arm is deployed to support the user in activities of daily living. In one of the scenarios, the robot is bringing a cup of tea to the human who is working. The tea goes very close to the laptop which – even without colliding with the laptop – raises concerns for the user, diminishing the quality and trust of the interaction. In a human-to-human scenario, one of the partners could simply ask the other user to move away from the

laptop. In this work, the goal is to embed such understanding in the robot. In the left-side image in Figure 1.1, the robot is able to understand a generic voice command and adapt its trajectory accordingly. It is worth noticing that the interaction highlighted involves a grammatically incorrect sentence that aims to reinforce the expressiveness of existing LLMs and their integration into the proposed mapping strategy.

On the right side of Figure 1.1, a second human-robot natural interaction is presented. In this case, the robot is carrying a sharp object, which is perceived as a perilous state from the human's perspective. Hence, given the voice command to reduce the speed, the robot is able to adapt its speed throughout the waypoints that are close to the human – speed is depicted in the bottom right graph.

Both examples in Figure 1.1 highlight the goals of this work, that is, how to seamlessly transform human's multimodal perception and communication in online trajectory adaptation of the robotic system. Our goal here is also to highlight that human-robot co-existence can take place in an increasing number of scenarios and with a growing number of different robotic platforms. Within this context, developing a language interface dissociated from specific robotic form factors, becomes of great relevance. Taking it into account, this work addresses the human robot interactions through the lenses of context and robot-agnostic representations.

1.2. Approach overview

The core of our method lies within natural language understanding, which is the most intuitive way for a user to express their intent. While large pre-trained language models (LLMs) such as BERT [4], GPT3 [5] and Megatron-Turing [6] have revolutionized our ability to perform linguistic tasks in recent years, we have just started to see pioneering works that incorporate large foundation language models with robotics tasks [7, 8, 9, 10]. The use of pre-trained LLMs is extremely beneficial within the robotics context because human-provided annotations are scarce and often costly to obtain. The challenge explored in this thesis then becomes how we can exploit these rich semantic representations and align them with geometrical trajectory data when mapping commands towards changes in robotic behavior.

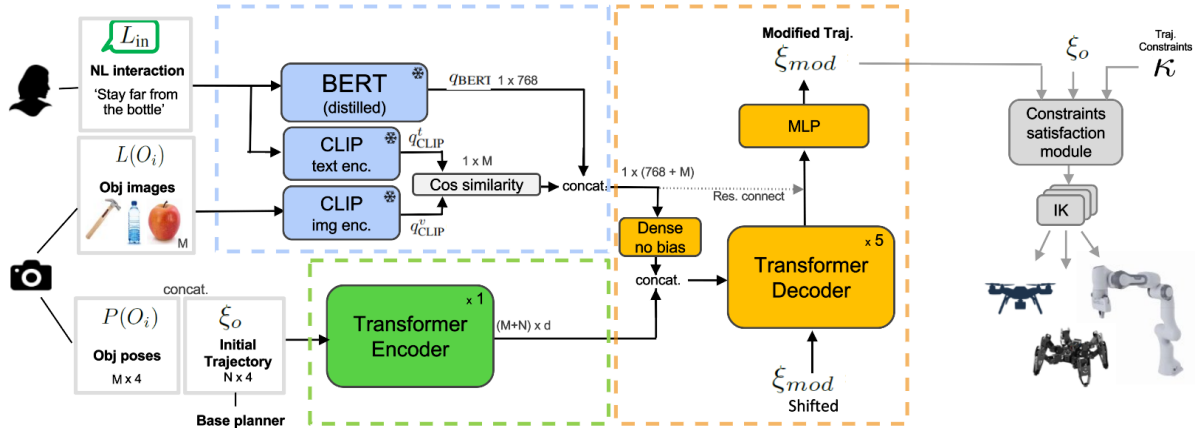

Figure 1.2.: Systems architecture: in blue, the language and contextual encoding module, compose mainly of frozen pre-trained models. In green, the geometrical encoding. In orange, the multimodal transformer decoder.

The method proposed uses an initialization from any geometrical planner (*e.g.* A*, RRT* [11], MPC [12]), which are concerned solely about obstacle avoidance and dynamics constraints, and augments it with semantic objectives.

1.3. Contributions

This thesis takes a step into building large pre-trained foundational models for robotics and shows how such models can create more intuitive and flexible interactions between humans and machines. Specifically, this work proposes a framework that allows a user to reshape a trajectory using language instructions. It can effectively align natural language features with geometrical cues jointly, and perform the goal of trajectory reshaping following human intent.

The main contributions of this work can be summarized as follows:

- **Novel system for semantic trajectory modification:** This work introduces a new approach for a language-based interface for robot behavior modification. Introducing a multimodal attention mechanism for semantic trajectory generation in a context aware manner.

- **Multi-platform evaluation:** Extensively evaluate the approach towards multiple robotics form factors beyond manipulators. We show that the model's outputs are amenable

to different robot dynamics and motion controller in aerial and legged locomotion domains.

- **Quantification of user interaction experience:** Perform diverse user studies to evaluate and compare human robot interfaces within tasks of trajectory modification.

1.3.1. Dissemination

Some of the parts of this thesis have already been published and presented in peer-review conferences and workshops:

Reshaping Robot Trajectories Using Natural Language Commands: A Study of Multi-Modal Data Alignment Using Transformers [13]

- Published at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022);
- Workshop on Collaborative Robots and the Work of the Future[1] and Workshop on Shared Autonomy in Physical Human-Robot Interaction: Adaptability and Trust at the IEEE International Conference on Robotics and Automation (ICRA);
- Northwest Robotics Symposium 2022;
- Project webpage: `https://arthurfenderbucker.github.io/NL_trajectory_reshaper/`
- Supplementary video: `https://www.youtube.com/watch?v=fhS0b3z7aXE`

LATTE: LAnguage Trajectory TransformEr [14]

- Foundation Models for Decision Making Workshop - NeurIPS 2022
- Submitted to: ICRA2023 (pending)
- Codebase available at: `https://github.com/arthurfenderbucker/LaTTe-Language-Trajectory-TransformEr.git`.
- Supplementary video: `https://www.youtube.com/watch?v=yCSZcCJEoPc`

[1] Spotlight contribution

1.4. Thesis Organization

This Chapter has given an introduction, including the motivation behind and a brief overview of the work. The remainder of this work is structured as follows. In Chapter 2, the relevant background is given. It covers the fundamentals regarding transformers in machine learning and foundation models, particularly focusing on large language models (LLMs). The chapter also introduces and discusses the related work in the areas of natural language and transformers in robotics. In Chapter 3, the problem is trajectory adaptation through natural language integration is formalized and the proposed approach based on the use of LLMs and transformers is presented. Chapter 4 presents and discusses experimental evaluation and a user study aimed at addressing the effectiveness of the proposed solution. Finally, the conclusion in Chapter 5 gives a brief overview and assessment of the results and presents possible future research into this area.

2. Background

2.1. Human robot interaction

As robots become more prevalent in environments outside of laboratories and dedicated manufacturing spaces, it is important to offer non-expert users simple ways of communication with machines. Within this context, the main question that rises is how to translate intuitive human behaviors into technical commands?

Human to human interactions are often based on multimodal instructions, combining natural language, gestures, gaze, physical demonstration, and more. The complexity of understanding such instructions increase dramatically if one consider the interdependency of the modalities (for example, associating a speech with a given gesture) or if one consider references to the context or abstract ideas.

Translating such complex interactions towards robotic platforms is an extremely relevant field of study with diverse facets and sub areas. In the past, diverse types of interfaces were introduced to allow humans to express their intent in robotic systems. Figure 2.1 depicts some of this interfaces in the context of trajectory modification. Notwithstanding, considering the goal of providing non-expert users the capability of intuitively coordinating and changing a robot's behavior, this work assumes that a Natural Language (NL) interface is the most natural and easy form of interaction – such assumption is tested and validated trough user studies in section 4.1.3.

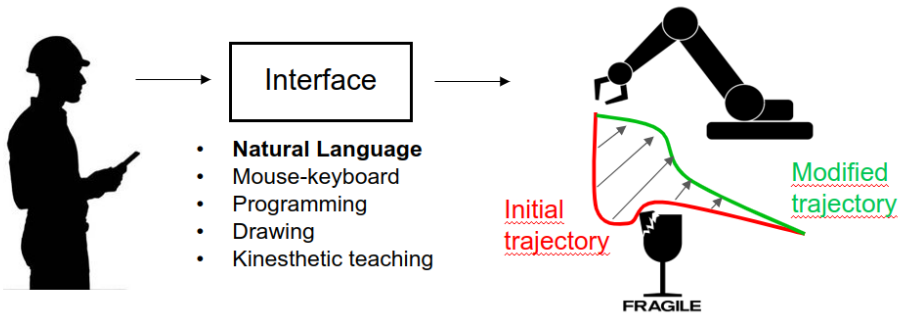

Figure 2.1.: Different types of human robot interfaces for trajectory correction

Therefore, this work lies on the field Natural-language-facilitated human–robot cooperation (NLC) and specifically targets unidirectional instructions. Within this field, as described in [15], solutions commonly fit into these three categories:

- Natural language instruction understanding
- Natural language-based execution plan generation
- Knowledge-world mapping (action/ context labeling)

The method presented in this work fit on the second category, Natural language-based execution plan generation. However, in contradiction to most traditional approaches, that focus on inferring **"what"** sequence of tasks to perform, we target to follow language instructions directed to **"how"** a specific task should be executed.

2.2. Foundational Models

Large language models such as BERT [4], GPT3 [5] and Megatron-Turing [6] have radically improved the quality of machine-generated text, along with our ability to solve natural language processing tasks. Beyond just language, we see a shift in machine learning architectures in multiple domains, as the dominant design paradigm changes from designing task-specific models towards the use of large foundational pre-trained models [16]. Several of these large models already combine multiple data modalities such as text, images, video, depth, and even the temporal dimension [17, 18, 19, 20]. The use of foundational models is appealing

because they are trained on broad datasets over a wide variety of downstream tasks, and therefore provide *general* skills which can be used directly or with minimal fine-tuning to new applications [16].

The field of robotics traditionally uses extremely task and hardware-specific models, which have to be re-trained and even re-designed if there are minor changes in robot dynamics, environment, and operational objectives. This inflexible machine learning approach is ripe for innovation with the use of foundational models [16], in particular when it comes to task specification in ambiguous scenarios (*What should I do?*) and task learning that can generalize across multiple environments (*How should I do it?*). Recent works have just started to explore the use of pre-existing foundational models from language and vision towards robotics [21, 8, 22, 23, 24], and also the development of robotics-specific foundational models [20, 25].

Our work aims to leverage information contained in existing vision-language foundational models to fill the gap in existing tools for human-robot interaction. Even though natural language is the richest form of communication between humans, modeling human-robot interactions using language is challenging because we often require vast amounts of data [26, 22, 27, 28], or classically, force the user to operate within a rigid set of instructions [29, 30]. To tackle these challenges, our framework makes use of two key ideas: first, we employ large pre-trained language models to provide rich user intent representations, and second, we align geometrical trajectory data with natural language jointly with the use of a multi-modal attention mechanism.

2.3. Transformer models

Transformer networks [31] are sequence to sequence models originally introduced to address language processing tasks. Its working principle relies on mainly two parts: an encoder block and a decoder block. The first is responsible to map a given input sequence into an embedding space, with lower dimensionality, that grasps the high-level connection between the elements of the sequence input. The decoder, on the other hand, uses this embedding representation of the input to generate the output sequence in an autoregressive manner. This autoregressive decoder performs the output generation by predicting one element (token) at a

time, and for that taking into account the values of the previously generated output elements and the embedding representation of the input.

The success of this architecture relies mainly on the attention mechanism in the interlayers of the encoder and decoder, and on the fact that it allows high parallelization during the training process. The attention mechanism enables the model to capture long-term dependencies with low computational effort and to focus on more relevant connections within the data. The parallelization benefit translates directly to more efficient use of the current GPUs and memory. Furthermore, it allows training such models with much higher volumes of data.

This model architecture is currently the state of the art for most sequence-to-sequence problems, and it is the foundation of the recent advancements in natural language understanding and temporal-spacial correlations.

2.3.1. BERT model overview

The Bidirectional Encoder Representations from Transformers (BERT) [4] is one of the main transformer-based models for natural language processing. This large language model was trained on pure textual data targeting tasks of question answering and language inference.

By training on a large corpus of textual data, BERT learned a semantic understanding of language and its underlying abstractions and connections. This deep understanding of language can be easily accessed by stacking the pretrained model with much simpler learning models (transfer learning) and training it for specific applications without the need for substantial task-specific architecture modifications.

2.3.2. CLIP model overview

The Contrastive Language Image Pre-training (CLIP) model [7] is a transformer-based architecture trained specifically to correlate images with textual description. This model was trained with a large dataset of 400 million (image, text) pairs collected from the internet and learned to represent both textual and visual information in a joint latent space. Meaning that the same values of the model's embedded space can represent high-level information about images and their appropriated captions. Figure 2.2 depicts this joint latent space.

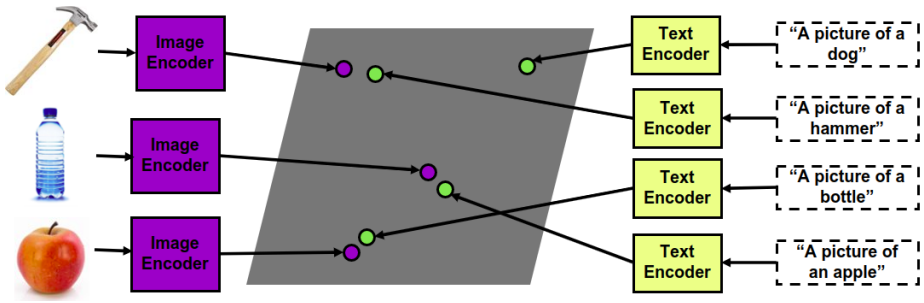

Figure 2.2.: CLIP joint latent space

This characteristic of the clip allows it to be used for a diverse set of problems that require a correlation between language and visual information. Within the context of this work, CLIP becomes extremely handy to perform the correlation between natural language interaction and context perception. In section 3.1.2 with discuss how this model can be applied to find target objects present in the user's interaction.

2.4. Related Work

2.4.1. Natural language and robotics

Equipping robots with natural language models provides an intuitive and straightforward interface to address these challenges through human interaction and decision-making. Classically, modeling human-robot interactions using language is challenging because it forces the user to operate within a rigid set of instructions [32], or requires mathematically complex algorithms to keep track of multiple probability distributions over actions and target objects [29, 30]. There has been an increase in recent works that explore the use of deep models to implicitly keep track of the complex mapping between language and actions, but the downside is that they often require vast amounts of data for training [26, 22, 27, 28].

In the domain of navigation we find literature that investigates the use of multi-modal representations fusing natural language and perception along with planning modules through the use of cost functions or reinforcement learning [21, 8, 22, 23, 24, 16, 33, 34]. In the manipulation domain we also find the work of [8], which uses CLIP [17] embeddings to

combine semantic and spatial information. To this end, it can be often beneficial to use pre-trained multi-modal representations that align visual and language inputs representation such as [35, 36, 37, 38], which often using BERT-style [4] training procedures. Representations are often fine-tuned [39, 40, 41] on the deployment scenario.

2.4.2. Transformers for robotics

Transformers, originally introduced in the language processing [31], quickly proved to be useful in modeling long-range data dependencies other domains. Within the robotics motion planning context, transformers architectures have been directly used for trajectory forecasting [42] and reinforcement learning [43, 44]. A more common use of transformers in robotics has been as feature extraction modules for one or more modalities simultaneously that leverage large-scale pre-trained models [13, 9, 7, 8, 10].

Particularly close to this work is the paper of [10]. It uses pre-trained LLMs to create a semantic cost map that guides a optimization-based motion planner to produce trajectories that satisfy motion constraints provided by a user in free-form text. Similarly, our method also uses LLMs for textual and visual feature extraction, however we use a transformer encoder-decoder pair to align semantic information with geometric cues to recast trajectories.

3. Methods

Though the development of this work, two incremental approaches were created. In this chapter, we present our final and most recent methodology for a 3D language-based trajectory modification [14]. Details of our previous 2D implementation [13] can be found in the supplementary section A.1.

3.1. Approach

Our overall goal is to provide a flexible interface for human-robot interaction within the context of trajectory reshaping that is agnostic to robotic platforms. The user provides a natural language command, and the robot's body or end-effector behavior, which is expressed with a 3D trajectory over time, is expected to be modified accordingly. Our trajectory generation system uses a sequential waypoint prediction model that takes into account multiple data modalities from scene geometry, environment images and the language input, all of which are fed into a transformer encoder-decoder pair.

Beyond the user's semantic intent, we expect the final trajectory to also respect safety and dynamics space-state constraints, which can be achieved by post-processing the model's output into a continuous state space. This last stage allows our same model to be employed by different robot form factors by using the proper inverse kinematics modules.

3.1.1. Problem Definition

Let $\xi_o : [-1, 1] \rightarrow \mathbb{R}^4$ be the original normalized robot trajectory which is composed by a collection of N waypoints and associated velocities $\xi_o = \{(x_1, y_1, z_1, v_1), ..., (x_N, y_N, z_N, v_N))\}$, where x_i, y_i, z_i and v_i are the waypoint coordinates and the velocity at time step i, respectively.

We assume that the original trajectory obeys the system constraints and can be pre-calculated using any desired motion planning algorithm, but falls short of the full task specifications. Let L_{in} be the user's natural language input sent to correct the original trajectory, such as L_{in} = "Go slower when next to the fragile glasses".

Let $\mathcal{O} = \{O_1, ..., O_M\}$ be a collection of M objects in the environment, each with a corresponding position $P(O_i) \in \mathbb{R}^3$ and image $I(O_i)$. Our goal is to learn a function f that maps the original trajectory, user command and obstacles towards a modified trajectory ξ_{mod}, which obeys the user's semantic objectives and is contained in the system feasible domain K:

$$\xi_{mod} = f(\xi_o, L_{\text{in}}, \mathcal{O}) \tag{3.1}$$

Equation 3.1.: ξ_{mod} represents the modified trajectory, ξ_0 the original trajectory, Lin the user's natural language input, and $\mathcal{O}$ the collection of objects

3.1.2. Proposed Network Architecture

We approximate the function f from (3.1) by a parametrized model f_θ, learned directly in a data-driven manner. This mapping is non-trivial since it combines data from multiple distinct modalities, and also contains ambiguities in solution space since there are multiple trajectories that satisfy the user's semantic objective.

Our model architecture is divided into 3 main modules and one constraint satisfaction step. Fig. 1.2 shows the connection between these modules. First, a language and image encoder makes use of distinct pre-trained feature encoders (BERT and CLIP) to generate a embedded representation of the natural language input and to identify the possible objects referred to in the text. Next, a geometry encoder uses object poses and trajectory waypoints as inputs and uses a transformer to learn geometric relations between the original trajectory, speed profiles and the objects in the scene. Finally, a multi-modal transformer decoder combines the embedded outputs of the two prior modules to generate the modified trajectory autoregressively. We discuss each module in detail below:

Language and image encoder:

The use of a large language model creates more flexibility in the natural language interface,

allowing the use of synonyms (shown in Section 4.1.1) and less training data, given that the encoder has already been trained with a massive corpus. We use a pre-trained BERT encoder [4], to produce semantic feature $q_{\text{BERT}}(z|L_{\text{in}})$ from the user's input. In addition, we use the pre-trained text and image encoders from CLIP [17] to extract latent embeddings from both the user's text $q^{t}_{\text{CLIP}}(z|L_{in})$ and the M object images $q^{v}_{\text{CLIP}}(z|I(O))$. We compute the cosine similarity vector s between the visual and textual embeddings in order to identify a possible target object for the user's command. In section 4.1.1 we show that using the object's images for target identification brings equivalent results as the simplified approach with object textual descriptions described in section A.1, since CLIP maps both modalities to a joint latent space. Finally, we concatenate the similarity vector s and the semantic features $q_{\text{BERT}}(z|L_{\text{in}})$ forming what we call semantic embedding q_{S}.

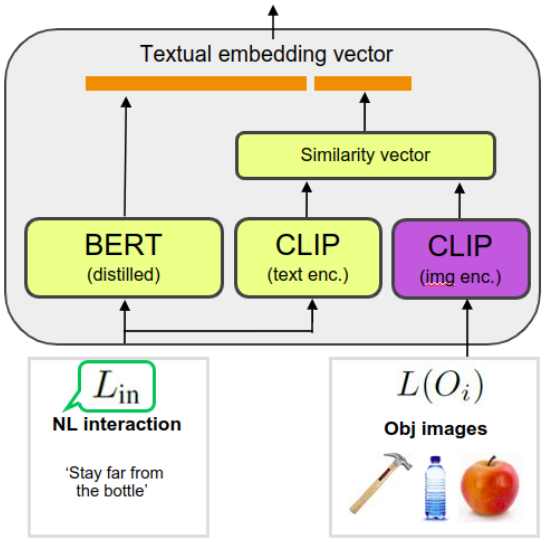

Figure 3.1.: Language and image encoder module

Geometry encoder: The original trajectory ξ_o is composed of points that are low-dimensional tuples $(x_i, y_i, z_i, v_i) \in \mathbb{R}^4$. In order to extract more meaningful information from each waypoint, we follow the example of [42] and apply a linear transform with learnable weights W_{geo} that projects each of these points into a higher dimensional feature space. The poses $P(O_i)$ of each object are also processed with the same linear transform, and padded with zeros for the velocity component.

We then concatenate the sequences of high-dimensional feature vectors from waypoints and objects and use a transformer-based feature encoder T_{enc} to extract geometrical features for each element. The use of a Transformer model is preferred for sequences over recurrent networks because its architecture can intrinsically attend to multiple time steps simultaneously. Conversely, recurrent networks suffer from vanishing gradient issues [42], which negatively affect feature extraction and training stability.

Multi-modal transformer decoder: Feature embeddings from both language and geometry are combined as input to a multi-modal transformer decoder block T_{dec}. This block generates the reshaped trajectory ξ_{mod} sequentially, feeding the last token prediction as input to the next waypoint prediction. This procedure is analogous to common transformer-based approaches for language translation [5, 31], but, in this case, one can reason that the proposed model *translates* trajectories from the original feature space towards a new space that obeys the user's semantic constraints. We use imitation learning to train the model, and employ the Mean Squared Error (MSE) loss between the predicted and ground-truth waypoints.

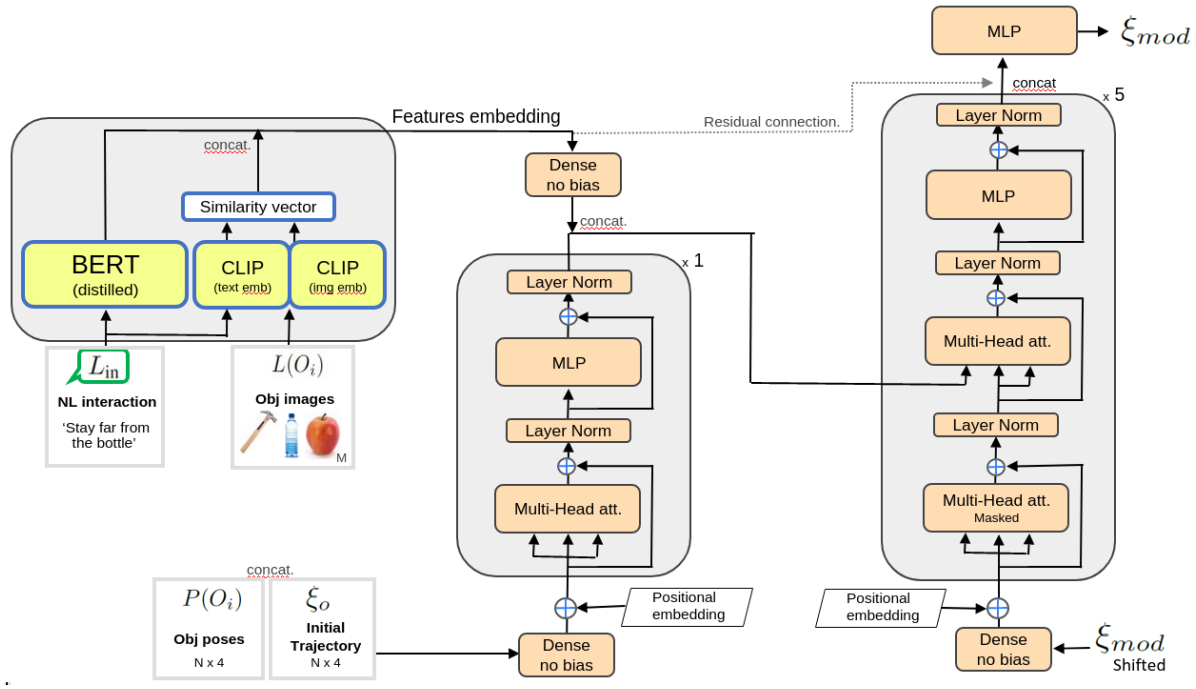

Figure 3.2.: Network full architecture. in the left, the Language and image encoder. In the center the Geometry encoder. In the right, the Multi-modal transformer decoder

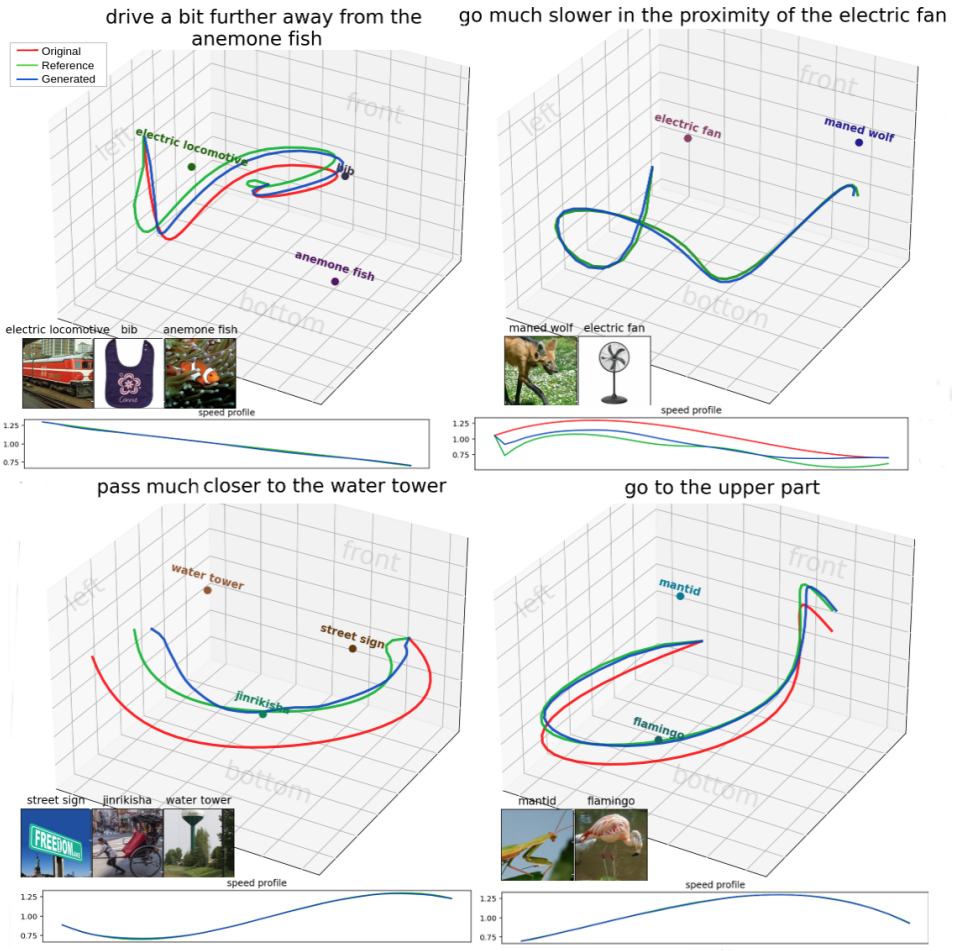

Figure 3.3.: Procedural dataset examples showing the original trajectory (red), ground-truth modifications, and model predictions (blue). Images representing objects are crawled from the web (bottom left), and the speed profile can also be modified (bottom right).

3.1.3. Post-processing and execution

Once a trajectory is generated by our model it needs to be post-processed to allow for the robot's execution. The modules described here allow our method to be agnostic to specific robotics platforms.

Constraint satisfaction: Constraint satisfaction is a complex and open field of study in robotics. In this work we establish two simplifying assumptions regarding our deployment objectives. First, the base motion planner outputs a set of hard constraints K defined in the Cartesian space that define an admissible region for the trajectories. Second, we assume

that the original trajectory is already within in the allowable constraint set. We post-process our model's output trajectory by taking steps starting at the original waypoint towards the direction of new one: $\xi(t) = \xi_o(t) + \alpha(\xi_{mod}(t) - \xi_o(t))$, where $0 < \alpha \leq 1$. If at any step we find that one waypoint reaches an inadmissible region then its position is not further updated. We note that more complex constraint satisfaction algorithms can be developed here, but the simple approached described worked well with our scenarios.

Inverse kinematics: Once the final trajectory is obtained, the user may plug in any inverse kinematics algorithm to obtain final trajectories for higher-dimensional degree of freedom robots. In this work we evaluate our system with manipulators, aerial and legged robots.

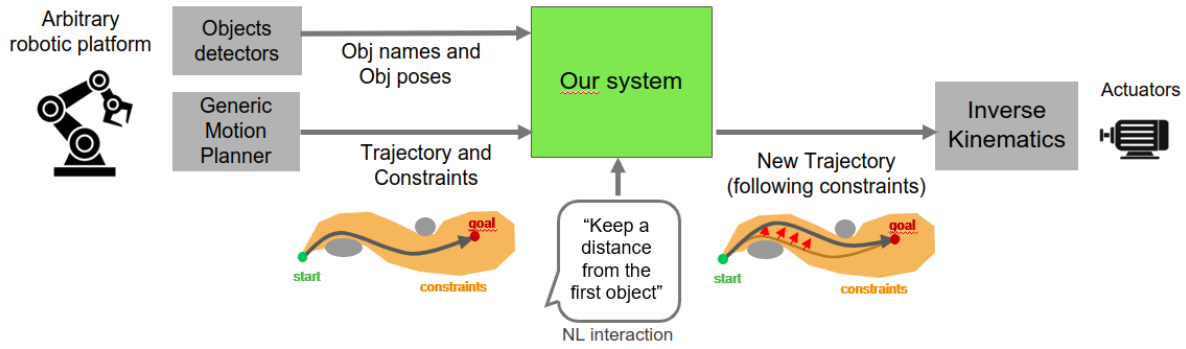

Figure 3.4.: System overview

3.1.4. Synthetic Data Generation

Data collection in the robotics domain can be challenging and expensive, specially when we require alignment between multiple modalities such as language, vision, and geometry. We find different strategies in the robotics literature to deal with these issues, ranging from costly large-scale online user studies for language labeling [45, 46] all the way to procedural data generation using heuristics [27]. Our work relies on purely procedural generation of trajectory-language pairs. We make a key hypothesis that the use of large-scale language models for feature encoding (q_{BERT}, q_{CLIP}) reduces the data requirements in terms of vocabulary diversity. We assume that if we are able to procedurally generate a small but meaningful set of examples with semantically-driven trajectory modifications we can train an effective transformer decoder, given that the BERT and CLIP encoders have already been

trained with large corpuses and are able to handle vocabulary and sentence variations. These assumptions are validated experimentally in Section 4.1.3.

Each data sample is composed of a base trajectory ξ_0, a natural language input L_{in}, a modified trajectory ξ_{mod}, and a set of object $\mathcal{O} = \{O_1, ..., O_M\}$ represented as central poses $P(O)$ and images $I(O)$. ξ_0 is generated by fitting a spline in the Cartesian space through points generated in a random walk. Objects poses are then randomly generated in space, and we sample object names from the Imagenet dataset [47] as their labels, and obtain various images for each one using a crawler over Bing Images using the object name as the web query.

As for the language input L_{in}, we focus on three main trajectory modifications: i) changes in the absolute Cartesian trajectory space (*e.g.* “stay on the left", “go more to the right"), ii) changes in speed (e.g. “go faster", “go slower when next to x"), and iii) positional changes relative to objects (e.g. “walk closer to x", “drive further away from x"). We pick a sample from a vocabulary bank associate each modification type, and calculate a force vector field over the environment using a handcrafted function $F(L_{in}, P(O))$. The field strength may vary depending on additional intensifier words that can be added to the sentences such as “very", “a bit", etc. In the section 4.1.1 we also explore augmenting these language inputs using BART [48], which is a pre-trained paraphrasing model. Finally, we generate the ground-truth trajectory modification by iteratively optimizing the original trajectory along the vector field.

We introduce one additional hyper-parameter in the dataset generation and model training which we name *locality factor*. For the same language prompt, some robotics contexts might require small localized trajectory changes while others might expect long-range modifications. After training, the locality factor allows the user to define their desired range of model influence.

3.2. Trajectory generation

3.2.1. Trajectory model

Each trajectory was initially described as a sequence of massless points in the space connected with virtual linear springs. Furthermore, aiming to constraint the magnitude of the trajectory modifications, each modified waypoint was connected with a linear spring connected to its

original position in the space. Finally, with the goal of keeping the trajectory's overall shape, angular linear spring were added between every consecutive link of waypoints. Figure 3.5 shows the virtual mechanical model assumed for the trajectories.

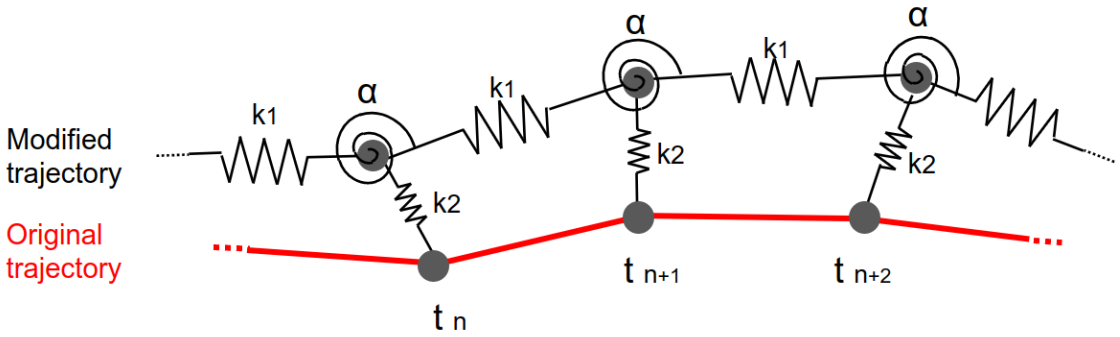

Figure 3.5.: Representation of the mechanical model for the iterative optimization during dataset generation. Where k_1, k_2 and α stands for elastic constants of the ideal springs and t_n, t_{n+1} represent the trajectory's waypoints at each time step.

3.2.2. Iterative trajectory optimization

The original waypoints were iteratively modified by applying the resultant force vector F_{int} computed based on its virtual springs forces and the external forces F_{ext} given by the list of force functions associated to the natural language (NL) interactions. The max number of interaction was set to 1000 and early-stop approach was implemented by checking for insignificant changes of the trajectory (reached stability). The algorithm 1 shows the optimization procedure described.

Algorithm 1 Modified trajectory generation

$it \leftarrow 0$

while $it \leq 1000$ or $err \leq \epsilon$ **do**

  for `f ∈ func_list` **do** ▷ external force vector from each NL interacion

    $F_{ext} = F_{ext} + f(\Xi_{it})$

  end for

  $F_{int} \leftarrow F_{wp\_prev} + F_{wp\_next} + F_{wp} + F_{ang}$ ▷ internal forces from virtual springs

  $delta \leftarrow (F_{int} + F_{ext}) * step$

  $it \leftarrow it + 1$

  $pts \leftarrow pts + delta$ ▷ update waypoints positions

end while

3.2.3. Initial trajectory generation

The initial trajectories were generated by performing a random walk on the 3D space using a random number of steps between 50 and 100. The waypoints of this random walk were then sampled and used as base points for a spline. Similarly, the velocity parameter was independently generated through the same procedure but now on a one dimensional random walk and fed into a 2 dimensional spline, considering the step index and its value.

Algorithm 2 Trajectory Generation

$n_{walk} \leftarrow$ random integer$[50, 100]$

$n_{int} \leftarrow$ random integer$[3, 15]$

$xyz_{walk} \leftarrow$ random_walk_3D(n_{walk}) ▷ n_{walk} steps

$traj_{raw} \leftarrow$ interpolate$(xyz_{walk}, num_key_steps = n_{int})$

$traj \leftarrow$ interpolate$(traj_{raw}, num_key_steps = 40)$ ▷ fixes the traj size for N wp

$vel_{walk} \leftarrow$ random_walk_1D(n_{walk}) ▷ n_{walk} steps

$vel_{raw} \leftarrow$ interpolate$(vel_{walk}, num_key_steps = n_{int})$

$vel \leftarrow$ interpolate$(vel_{raw}, num_key_steps = 40)$ ▷ fixes the traj size for N wp

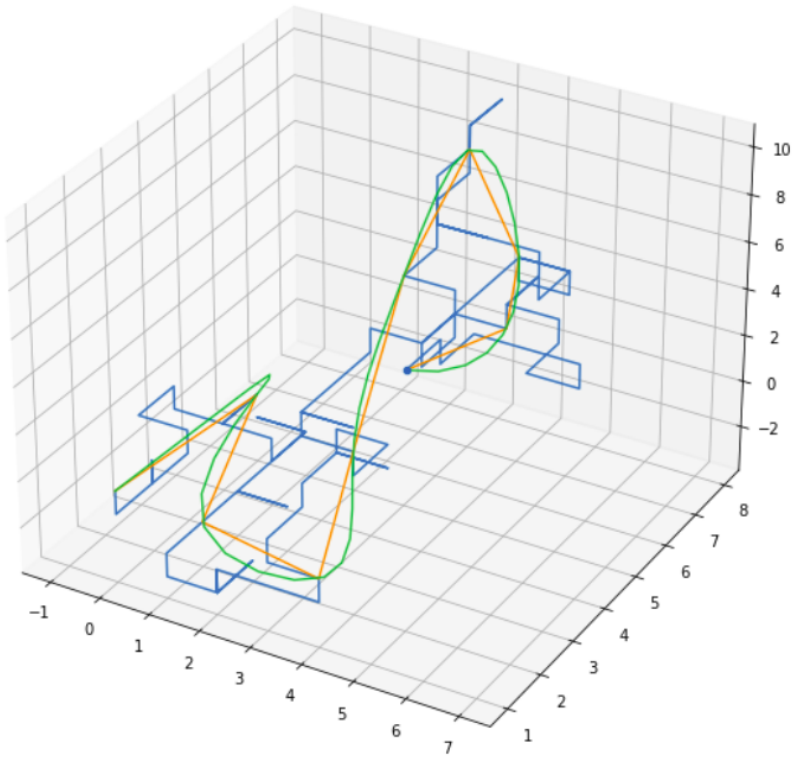

Figure 3.6.: Overlapping steps of the initial trajectory generation for the x,y and z components. In blue, the initial random walk ($n_{walk} = 100$); in orange, the sequence of waypoints used for the first interpolation ($n_{int} = 10$) ; In green the resulting interpolated trajectory ($N = 40$)

3.3. Handcrafted cost functions

Each language input was associated with a handcrafted function F(Lin, P(O)) that outputs a force vector field for the trajectory modification. This work focused on three main forms of interaction with the environment:

- Cartesian changes: referring to changes on the 3 cartesian axis
- Distance changes: modifying the trajectory's distance from objects in the scene.
- Speed changes: altering the velocity of the global trajectory or parts of it in the surrounding of objects.

details of the implementations of the functions and procedural text generation is described below:

3.3.1. Text generation procedure

Language description

The natural language interactions were described by using a directional graph structure, with no loops, where each node was associated with specific functions and parameters. Figure 3.7 shows a graphical example of this structure. This approach allows us to easily create rules and connections between language elements and build alongside an associated function.

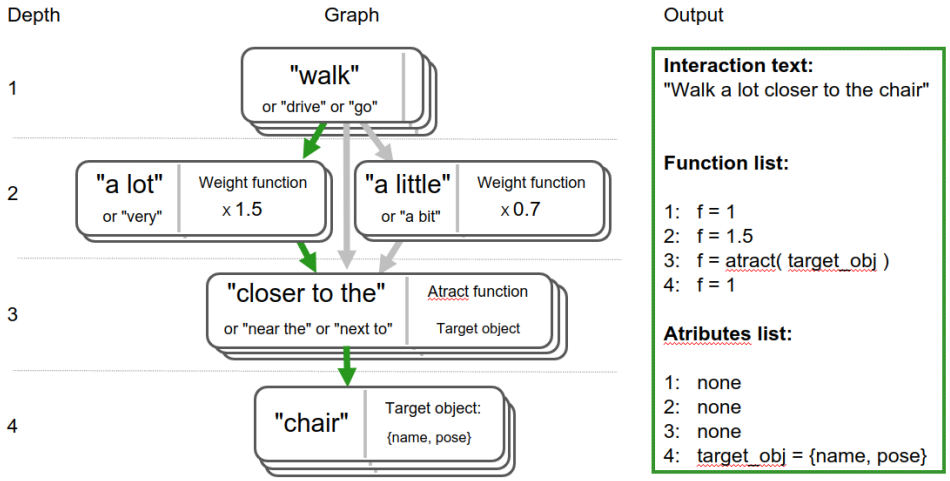

Figure 3.7.: Part of the vocabulary graph and its associated functions and attributes for the *distance changes*. The connection probability between each node is uniformly distributed for each depth of the graph

Some of the textual interactions generated can be seen below:

'pass a lot further away from the Panthera tigris' , 'drive a little closer to the sock' , 'walk a little slower while passing nearby the coffee mug' , 'pass a bit further away from the window screen' , 'stay on the front' , 'pass much closer to the paintbrush' , 'stay on the top' , 'drive very closer to the dial telephone' , 'stay on the bottom part' , 'go a little slower when passing in the proximity of the triceratops' , 'go to the right' , 'walk a bit further away from the

crawfish' , 'walk a bit further away from the tabby' , 'drive a little faster when passing near the bassinet' , 'go very faster when passing in the surrounding of the Lycaon pictus' , 'increase the speed in the proximity of the turnstile' , 'drive a lot closer to the acorn squash' , 'stay on the bottom part' , 'go to the front' , 'walk further away from the passenger car' , 'drive very closer to the kite' , 'stay on the right' , 'stay on the back' , 'keep a smaller distance from the swimming trunks'

3.3.2. Cartesian changes

The outputted vector field is an uniform in the entire task space. Its direction is determined by the cartesian direction word present in the associated text (e.g "...down" $\rightarrow [0, 0, -1]$) and its intensity is scaled by the adjectives present in the text (e.g "very" $\rightarrow$ 1.5, None $\rightarrow$ 1.0, "a bit" $\rightarrow$ 0.7).

Algorithm 3 Cartesian changes

function DIRECTIONAL_FORCE($traj, direction, intensity$)
  $force \leftarrow direction * intesity$
  return $tile(force)$ ▷ same dimensionality as traj
end function

3.3.3. Distance changes

The outputted vector field points out of the referred object in the text. For every point in the task space with a distance greater than a given range (determined by the locality factor), the field intensity is zero, elsewhere within the range, the intensity is constant and depends on the intensity adjectives present.

Algorithm 4 Distance change

function REPEL_ATRACT($traj, direction, sign, taget\_obj$)
  $dist \leftarrow$ distance from wp to target object
  if $dist \leq locality\_factor$ **then**
    $intesity \leftarrow$ cont. force
  else
    $intesity \leftarrow 0$
  end if
  $force \leftarrow direction * intesity * sign$
  return $force$
end function

3.3.4. Speed changes

The speed change is divided into 2 types, global changes (e.g "increase the velocity") and geometrically depended changes (e.g. "go faster in the surroundings of the table"). For the global changes, vector field is constant everywhere for the speed component, with intensity depend on the intensity adjectives. For geometrically depended changes, the field's speed component is constant for the regions within a given range around the referred object and zero elsewhere.

Algorithm 5 Speed change

function SPEED_UP_DOWN($traj, sign, taget\_obj$)
  $dist \leftarrow$ distance from wp to target object
  if $dist \leq locality\_factor$ **then**
    $intesity \leftarrow$ cont. force
  else
    $intesity \leftarrow 0$
  end if
  return $intesity * sign$
end function

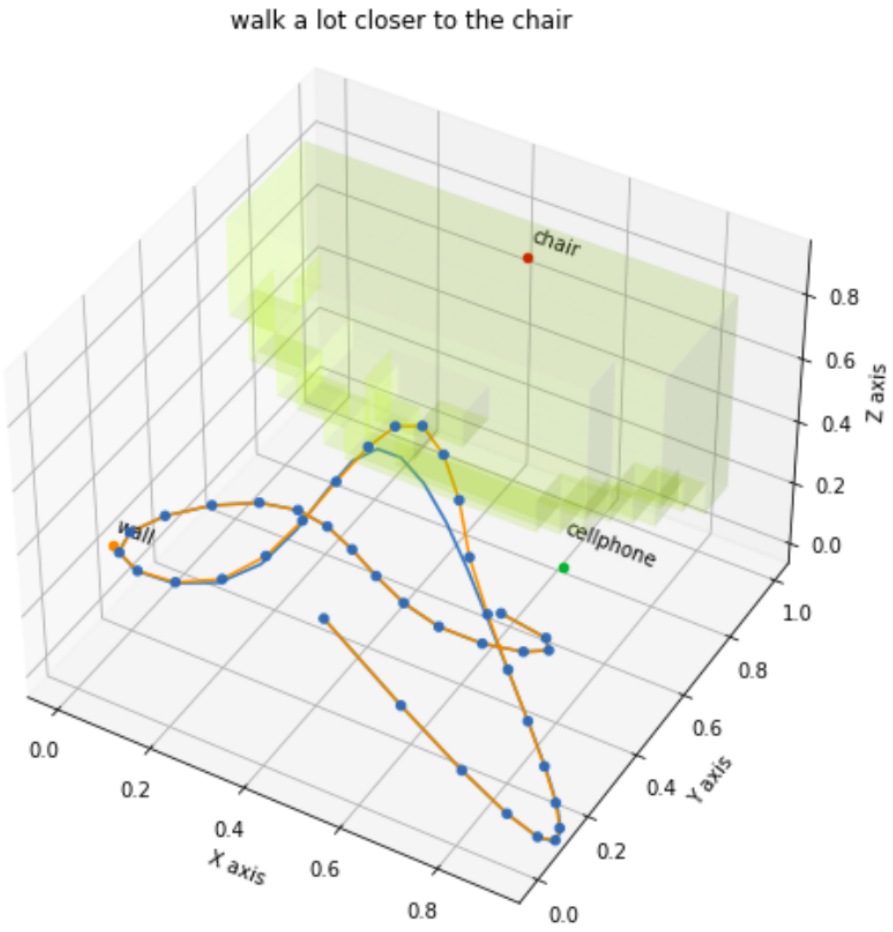

Figure 3.8.: Example of the modified trajectory generation for the input: "walk a lot closer to the chair". The blue line represents the original trajectory; the blue dots, each waypoint under optimization; In orange the final modified trajectory; The yellow volume represents the region of space with an attraction force towards the target object ("chair": red dot)

3.4. Network Training

We trained and evaluated the model described in Section 3.1 over a dataset containing 100k examples of procedurally generated trajectory modification. Among these, we used 70k samples for training, 10k for validation and 20k for testing. We kept both BERT and CLIP encoder weights frozen in other to avoid biasing the models towards our vocabulary, with

$q_{\text{BERT}}(z|L_{\text{in}}) \in^{768}$ and $q^{v}_{\text{CLIP}}(z|I(O)) \in^{512}$. We upscale the dimensionality of each scene object pose from $4 \rightarrow 400$ (depth) using a learned linear matrix, and apply the same procedure to the 40 waypoints from the original trajectory ξ_o. T_{enc} is a 1-block transformer encoder, and T_{dec} is a 5-block transformer. Each transformer has 3 hidden layers with 512 fully-connected neurons with Relu activations, one Layer Normalization, 8 attention heads. Figur3.2 represents graphically the detailed architecture of the model. We use the AdamW [49] optimizer with an initial learning rate $\gamma = 1e-4$, a linear warm-up period of 15 epochs and a learning rate decay of 90% after a plateau of 10 epochs on the validation loss.

We employ the MSE loss (L_{MSE}) both for training and model evaluation.

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m}\left(x_i^j - p_i^j\right)^2 \tag{3.2}$$

Equation 3.2.: where x_i^j respresents the ground truth waypoint vector and p_i^j represents the predicted waypoint. The j-th component in m dimensional state space (m = 4), and the i-th indicates the waypoint index on a trajectory of length N.

We use a Nvidia Tesla V100 GPU with batch size of 16, and train the model for 500 epochs in approximately 2 hours.

Through the use of Azure remote machines multiple models were trained, allowing a quick iteration on the model's architecture and hyperparameter.

Geometric augmentation:

We applied a shift on the geometrical values (trajectory's waypoints and object positions) using random values between -0.2 and 0.2 for each axis (x,y,z). We also performed a scaling operation, multiplying the geometrical values by a random scalar from 0.6 to 1.2. These values were chosen to provide some variation on the waypoints and object positions, but aiming to keep them whit-in the range [1, 1] to allow a direct forecast of using the *tanh* activation function.

4. Results

4.1. Experiments

We conducted several simulated and real-world experiments to validate our methods. Our main goals were to: i) measure the effectiveness of our trajectory modification algorithm in 3D and velocity space, ii) understand the influence of the different architectural components towards the model's success, and iii) validate the applicability of the model to multiple robotic platforms.

4.1.1. Simulation Experiments

We apply our method to several simulated scenarios. First, we show the basic workings of our trajectory adaptation method through qualitative results which can be visualized in Figure 3.3. In this scenario, we use sample objects that were randomly chosen from crawling the web and their corresponding images. Assuming there is an initial trajectory that traverses around these objects, and given language commands indicating how to modify the trajectory (farther/closer to the object, faster/slower in the vicinity of an object), our model predicts trajectories that account for user intent. We show both spatial modifications as well as changes in speed profile in the trajectories output by our model.

Multi-platform evaluation: To validate our framework's ability to adapt to different robot dynamics and environments we designed simulated environments using the CoppeliaSim simulator with Bullet physical engine [50]. While our original training dataset presents itself in a format amenable to end-effector positions within a manipulation context, this new simulator allows us to test our system on distinct robotic platforms, dynamics and base motion controllers.

Specifically, we employ an aerial vehicle and a legged hexapod platform. The drone operates within a 3D global frame of reference and uses PID motion controller for trajectory tracking. In contrast, the hexapod is constrained to 2D movements and uses an open-loop motion controller. As figure 4.1 shows, our approach can successfully modify the base trajectories (red) for different types of natural language inputs. Additional experiments can be seen in the video attachment.

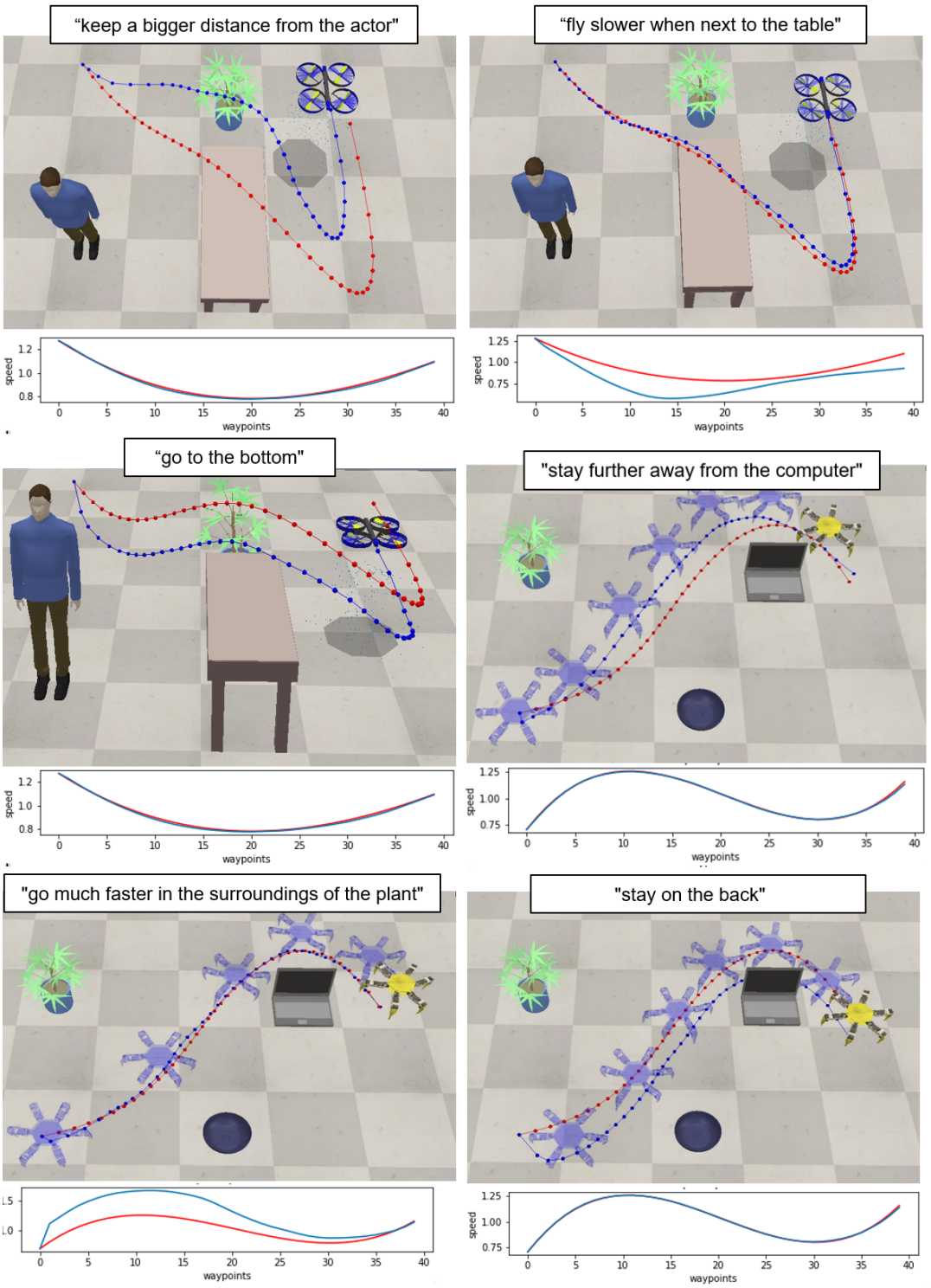

Figure 4.1.: Model deployed with different robot form factors (drone and legged hexapod) for obstacle avoidance, speed refinement and absolute cartesian changes. Original trajectory shown in red, modification in blue, and corresponding speed profiles below each scenario.

Baseline architectures: We compare our proposed multi-modal transformer against architecture variations. Table 4.1 shows the result of a grid search over the number of layers and encoding dimension (depth) of the transformer encoder and decoders. The model with one encode layer, 5 decoder layer and an depth of 400 was chosen to be the reference model for our architecture and further baseline comparisons. We measure performance in terms the similarity between our model's output and the ground-truth trajectory modification in the dataset. Our metrics are MSE (mean squared error), MAE (mean absolute error), DTW (dynamic time warping), and DFD (discrete Frechet distance).

n.enc	n.dec	n.depth	param.	MSE↓	MAE↓	DTW↓	DFD↓
2	3	256	4.95M	0.00306	0.0314	3.1085	0.1346
2	3	400	9.28M	0.00235	0.0273	2.6966	0.1198
2	5	256	6.53M	0.00280	0.0284	2.8455	0.1265
2	5	400	12.7M	0.00238	0.0231	2.4900	0.1152
1	3	256	4.42M	0.00274	0.0272	2.8122	0.1245
1	3	400	8.22M	**0.00224**	**0.0229**	**2.4445**	**0.1130**
1	5	256	6.00M	0.00277	0.0264	2.7527	0.1238
1	5	400	11.2M	0.00234	0.0227	2.4699	0.1138

Table 4.1.: Architecture variations

Table 4.1 provides valuable findings regarding the model architecture. For instance, increasing the number of encoder blocks caused no improvement on the model's performance. Furthermore the model with 3 decoder blocks presented slightly better results than the assumed baseline of 5 decoder block.

In addition to model size, in table 4.2 we compare different architecture structures. The *Naive* approach simply copies the original trajectory. The *No NL input* baseline represents a universal prior of the dataset, with an empty language command. *Ours light* is a more compact version of our model with 1 enc., 3 dec. and depth of 256.

Locality factor: Fig. 4.2 shows the response of our model for different values of the locality factor (LF). This hyper-parameter provides useful information on the range of the desired

Approach	Param.	MSE↓	MAE↓	DTW↓	DFD↓
Naive	-	0.00437	0.02709	3.568	0.1387
No NL input	11.2M	0.04193	0.1663	15.097	0.5674
Ours light	4,42M	0,00274	0,0272	2,8122	0,1245
Ours	11.2M	**0,00234**	**0,02273**	**2,4699**	**0,1138**

Table 4.2.: Baseline architecture comparisons

change change over the trajectory, which can serve as a finer user control besides the language input itself.

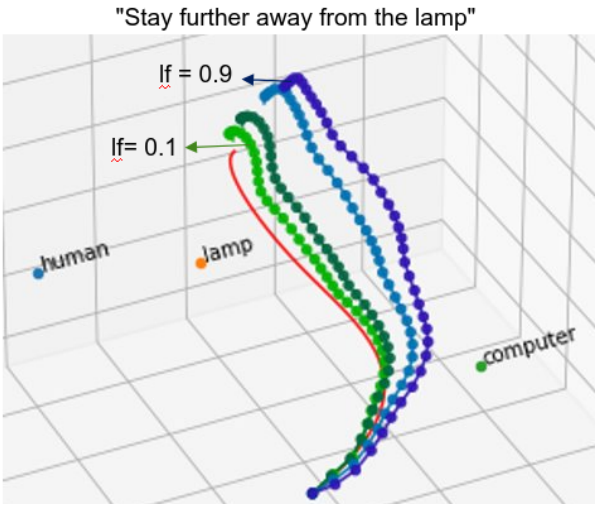

Figure 4.2.: Locality factor influence

Dataset size and augmentations: Table 4.3 shows the effect of increasing the training dataset size in model performance, as well as the effect of applying augmentations in the training data. An increase in the dataset size from 1k to 10k samples significantly improves the validation metrics with minimal challenges besides a longer training time, given that data can be generated procedurally without expensive human annotations. The geometrical augmentation (randomly shifting and scaling operations) shows a modest increase in performance.

	Without geometrical augmentation			
Dataset size	MSE↓	MAE↓	DTW↓	DFD↓
1k	0.02608	0.11063	8.20700	0.46488
10k	0.00243	0.02347	2.47016	0.11683
100k	0.00229	0.02201	2.39301	0.11175
	With geometrical augmentation			
Dataset size	MSE↓	MAE↓	DTW↓	DFD↓
1k	0.01420	0.07590	5.35290	0.35737
10k	0.00248	0.02324	2.50841	0.11593
100k	0.00234	0.02273	2.46992	0.11383

Table 4.3.: Effect of dataset size and geometrical augmentation.

Vocabulary and object diversity:

One key hypothesis assumed true when designing our model architecture was that the use of pre-trained large language models as feature encoders would make our pipeline amenable to a diverse set of natural language inputs, despite the relatively small amount of training examples. To test this hypothesis we compute results using with novel user commands, with vocabulary not present in our training language labels. Fig. 4.3 shows that our simplified 2D model (presented in section A.1) still executes the expected behavior, being able to find the correct semantic meaning despite the new words.

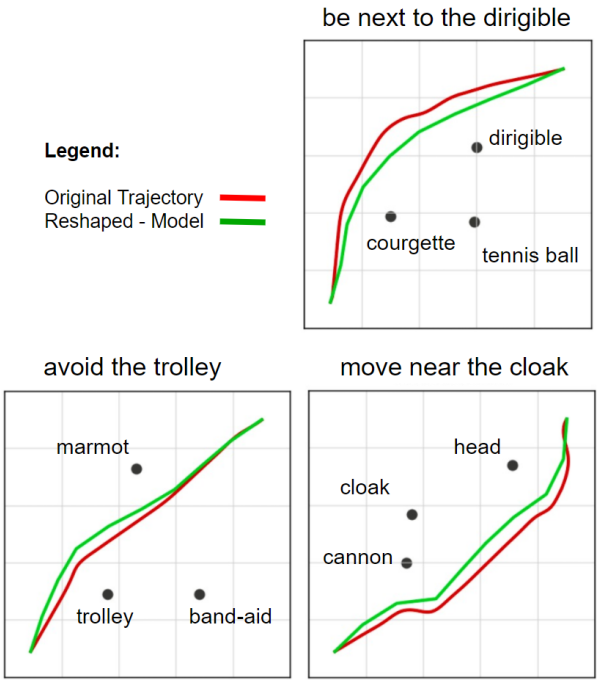

Figure 4.3.: Trajectory reshaping results using novel vocabulary (not seen in the training data) as the user input. Our 2D model is able to correctly execute the desired semantic commands due to the large capacity of the BERT and CLIP text encoders.

4.1.2. Real Robot Experiments with Manipulation

We deployed our model in real-world experiments using a 7-DOF PANDA Arm robot equipped with a claw gripper. An off-the-shelf CPU/GPU setup computes the arm's low-level controller and our model. A RGBD camera (Intel RealSense D435) mounted on the workbench captures images of the obstacle setting, and a YOLOV3[51] object detector extracts bounding boxes of the five most likely objects to be sent to the CLIP encoder. The 3D poses of the objects are inferred using the depth image of each bounding box. Snapshots of the setup and results can be found in figures 1.1 and 4.4. Additional experiments shown in the video attachment.

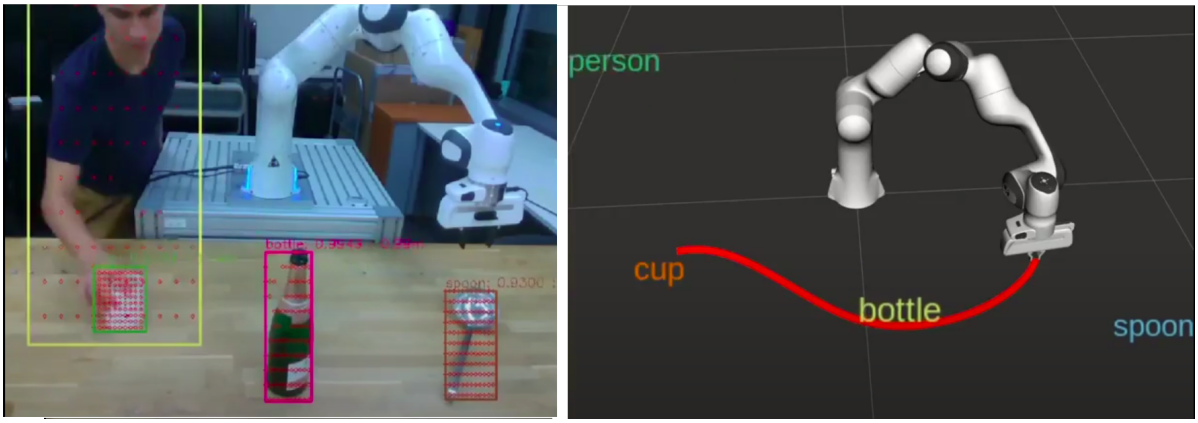

Figure 4.4.: Real life setup. It depicts the online object detection and 3D pose inference of the objects present in the scene. Full videos in the supplementary material.

4.1.3. User study experiments

4.1.3.1. How much the model captures the user intent?

We evaluated the model's performance against baseline architectures in a user study, collecting in total 300 data-points from 10 participants. Each user was asked to evaluate within a 1-5 Likert scale the trajectory changes generated from 5 different approaches considering a given NL interaction. Figure 4.5 summarizes the distribution of answers for each baseline. *"Ground Truth"* represents the procedural dataset used for training. As the chart shows, most users considered that our trajectory modifications in the dataset correctly represented the language commands. A similar pattern emerged from our trained model (*"Ours"*), which yielded high-quality ratings. The *"Ground Fake"* approach shows samples of the dataset with intentionally wrong modifications, opposite to the ground truth, for the means of comparison. Non surprisingly, it is rated with the lowest score. The *"No language"* baseline was also badly evaluated, showing that the model's performance is highly dependent on the language input, and that the model does not memorize bias purely based on the scene context. Finally, the *"Projected 2D"* distribution shows a direct comparison with our simplified approach (section A.1), which produces pure 2D trajectory modifications. Its bad performance motivates the importance of the additions of 3D and velocity space that we incorporate in this updated setup.

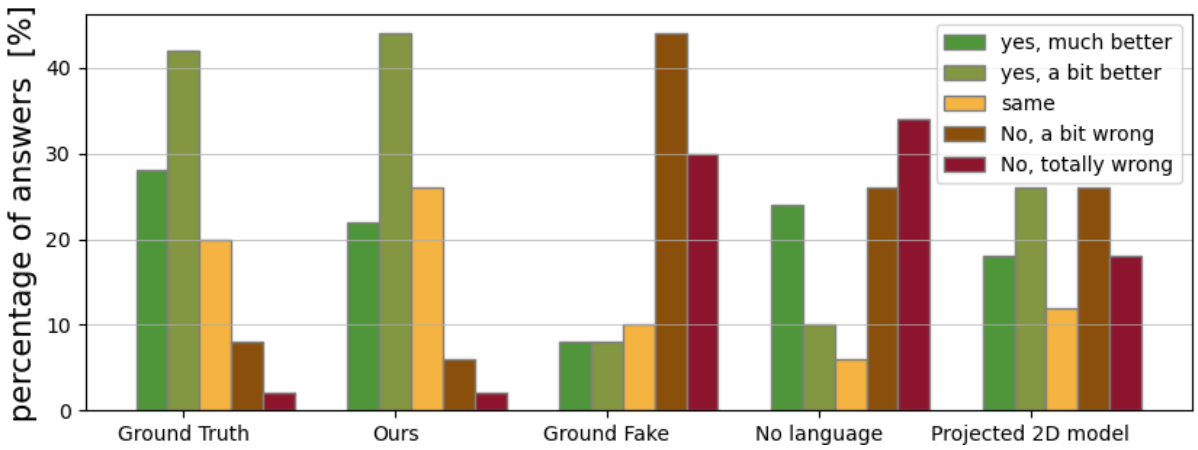

Figure 4.5.: Userstudy distributions of answer for each baseline.

After the initial evaluations, each user was asked to freely interact with 5 trajectories using a text box, and next judge the quality of the generated modifications. 48% of the user inputs presented words never seen by the model during the training process (out of distribution). Even under these challenging conditions the model only failed on 24% of the cases. Table 4.4 compares our model's performance for in and out of distribution settings.

Textual interaction	Better [%]	Same [%]	Worse [%]
In-dataset vocabulary	66.0	26.0	8.0
Free user input	46.0	30.0	24.0

Table 4.4.: Evaluation of out of distribution NL interactions

4.1.3.2. How the NL interface compares with other types of interaction?

We also evaluate our system with real-world experiments, and compare our method with the use of multiple human-robot interfaces. We use a 7-DOF PANDA Arm robot equipped with a claw gripper, and execute tasks on a 1×1m tabletop workspace. A standard desktop computer with an off-the-shelf GPU connected to the robot computes the original trajectories, executes our model, and runs low-level controls for the arm.

At the time of this experiment, our architecture was still restricted to 2D trajectory modifi-

cations, as describe in section A.1. Because of that, we operate the model using 2D planar projections of the original robot trajectories, and respect the original waypoint heights when executing the reshaped motion plans. We use object positions given by markers, but we discuss the use of vision-based localization in section 4.1.2.

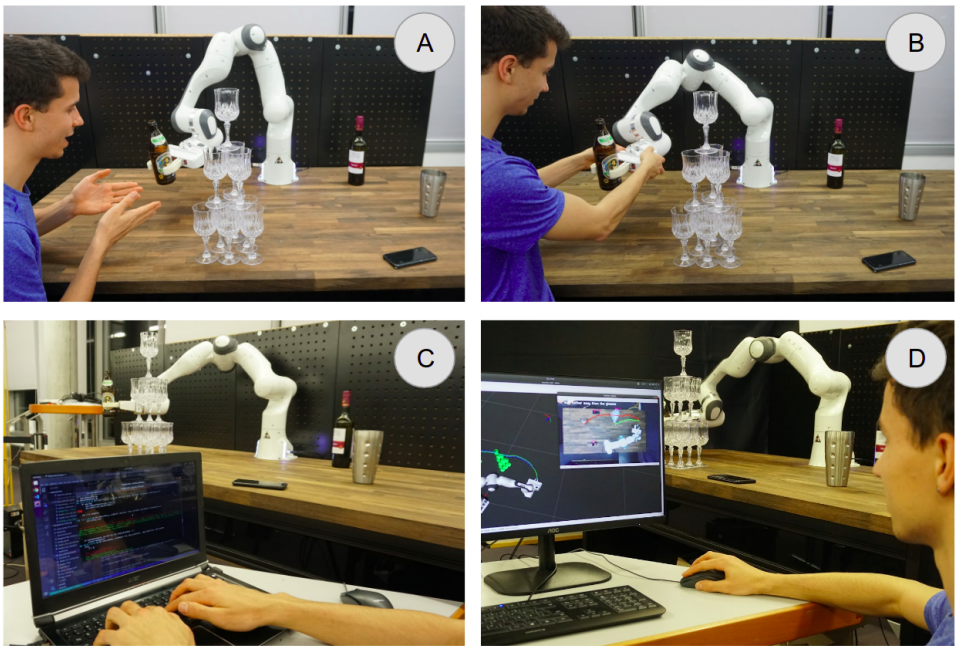

Figure 4.6.: Human-robot interfaces tested in the user study: a) natural language (NL), b) kinesthetic teaching (KT), c) programming via keyboard, and d) trajectory drawing.

The goal of the study is to have the user control a robotic bartender. A traditional motion planning algorithm calculates an initial trajectory to transport a bottle of wine towards a cocktail shaker and pour the liquid inside (we leave the problem of learning how to make fancy drinks for future iterations of this work). This original trajectory comes dangerously close to toppling over a tower of crystal glasses, and the user needs to interact with the robot to make the end-effector trajectory safer. As seen in Fig. 4.6 we test 4 different human-robot interfaces: natural language (NL-ours), kinesthetic teaching (KT), trajectory drawing (Draw), and programming obstacle avoidance weights via a keyboard and mouse (Prog). A top-down view of the experimental platform is seen in Fig. 4.7. All user interactions followed a

study protocol approved by the Technical University of Munich's ethics committee, and we conducted a total of 10 interviews.

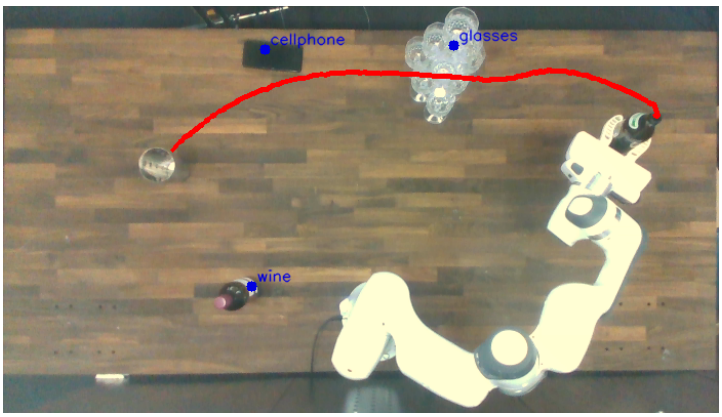

Figure 4.7.: Experimental platform used for the user study. The tabletop contains three objects (a cellphone, a wine bottle and crystal glasses), and the original robot trajectory (in red) passes dangerously close to the tower of glasses. This exact same view was used for the drawing interface.

Quantitative user evaluation: We measured statistics on the number of iterations, success rate and total time taken for users to modify trajectories using the different interfaces. From Table 4.5 we see that the programming interface takes by far the longest for users to master, and requires numerous iterations. In the meanwhile, NL is the fastest option. We see a large number of failures for kinesthetic teaching and drawing because user inputs are often times kinematically infeasible by the robot joints. The natural language method proved to be the most robust, and we found no failure cases during in the study.

Interface	Avg. iterations	Success rate (%)	Avg. Time (s)
NL	**1.33**	**100**	**81**
KT	1.78	56.24	139
Draw	1.89	64.7	120
Prog	4.00	91.66	284

Table 4.5.: Statistics collected over the user study experiment

Qualitative user evaluation: After the experiments we asked users to rate the trajectories produced by different interfaces according to different criteria in a psychometric questionnaire:

1. How satisfied were you with the final robot motion?
2. How easy was refining the robot motion?
3. How safe was the final robot trajectory?
4. How natural was the human-machine interaction?
5. How predictable was the trajectory for you?

Table 4.6 summarizes the responses. We can see that most methods present a similar user satisfaction level except for programming, which was rated lower likely due to the difficulty of interaction. NL was rated as the easiest and most natural method, but at the same time was deemed less predictable than KT and drawing because with these two methods users have direct control over the final trajectory.

Interface	Satisfied	Ease of use	Safety	Natural	Predictable
NL	**90**	**92**	92	**98**	72
KT	**90**	88	88	78	**96**
Draw	88	74	**100**	80	88
Prog	62	58	82	62	48

Table 4.6.: User ratings collected in the user study

4.1.4. Ablation studies

4.1.4.1. What does the multimodal attention layer learn?

Fig. 4.8 displays an attention map that gives us insights into the model's decisions. Attention layers of the geometrical transformer encoder focus mainly on the relation between the trajectory waypoints and the object positions, as well as the connections between neighboring positions. On the other hand, heatmaps from the transformer decoder show intense focus

on the previous two waypoints from the output sequence, and a high attention towards the BERT/CLIP feature vector on the far right.

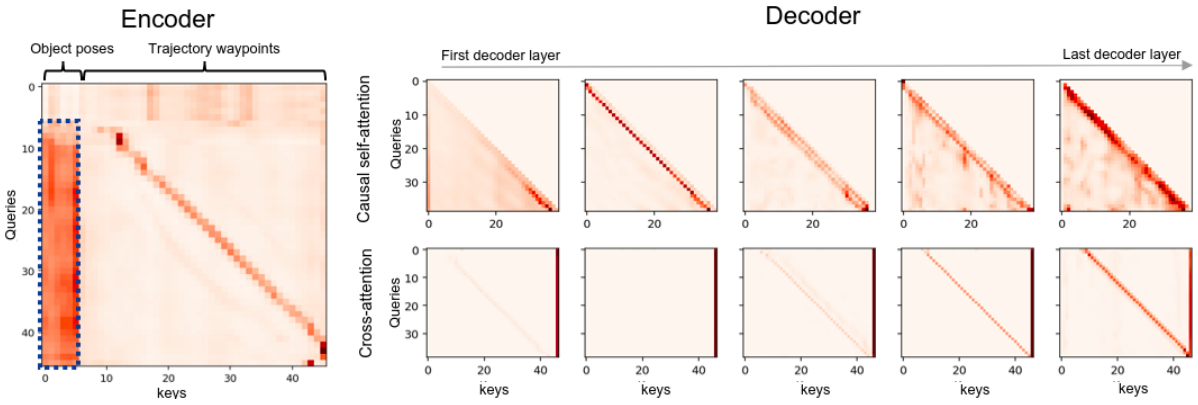

Figure 4.8.: Heatmap of average attention over the test set showing the transformer encoder and decoder layers.

4.1.4.2. Training ablation studies

During the development of the model and training procedure, we performed a few experiments aiming to improve the model's performance. However, the modifications described in this section did not show significant improvement on the model and in many cases decreased the model's accuracy. Despite that, we decided to show and describe the experiments performed for the sake of completeness. The full training procedure used for our final models can be found in the section3.4.

Training loss variation:

Aiming to achieve better convergence and improve the model's inference capabilities, we tested 3 different losses:

- Length Difference and Angle Difference (LDA) loss: As described [52], the LDA loss function should lead to more stable and accurate approximation of the long term dynamical behavior in systems. Following its results, we implemented the LDA loss for our system as described:

  But, in spite of testing multiple learning rates, the loss has shown unstable during training or did not lead to improvements on our system when compared with the MSE loss.

$$L_{LDA} = \frac{1}{N}\sum_{i=1}^{N}\left(k_1 \cdot \frac{\sqrt{(|x_i| - |p_i|)^2}}{|x_i| + |p_i|} + \frac{1}{2} \cdot \left(1 - \frac{x_i \cdot p_i}{|x_i| \cdot |p_i|}\right)\right) \tag{4.1}$$

Equation 4.1.: Where x_i stands for the ground truth waypoint vector and p_i for the predicted waypoint vector. The i-th indicates the waypoint index on a trajectory of length N.

- LDA + MSE loss: Aiming to better stabilize the loss during training, as described in [52], we combined the LDA loss with MSE:

$$L_{LDA\_MSE} = c \cdot L_{LDA} + (1 - c) \cdot L_{MSE} \tag{4.2}$$

Equation 4.2.: Where c stands for the weighting factor between the losses

The combined approach indeed improved the training stability. However, even testing multiple balances between the losses (c) and different learning rates, we were not able to achieve better results than by using just the MSE loss.

- soft-DTW loss: considering that the Dynamic time warping (DTW) was created to quantify the difference between sets of temporal-spatial data, it is extremely suitable to describe similarity between trajectories. [53] proposed a differentiable version of this metric, allowing it to be used to optimize learning models. Motivated by the results achieved by the o work of [54] and [53] we incorporated the metric in our training pipeline. Nevertheless, due to the quadratic complexity ($O(n)$) of soft-DTW and the high number of waypoints on our trajectories, the use of the loss led to impracticable training time, not being suitable for our setup.

Transformer withoutLayer Normalization

Considering that our implementation uses the transformer architecture for a regression task – predicting each new waypoint positions in a continuous state space –, we stated the hypothesis, that by removing the "Layer Normalization" layer inside the transformer encoder and decoder blocks, the prediction accuracy might increase, since the absolute information of the waypoints locations would not be lost in the normalization process. However, against our

expectations, this hypothesis has shown to be false. By removing the "Layer Normalization", the training process became unstable even for much lower learning rates. We assume that this behavior occurred due to exploding gradients during the training process and insufficient regularization in the deeper layer of the network.

Dataset textual augmentation: Aiming to validate possible improvements on the model's performance, we augmented the language inputs using the paraphrasing model BART [48]. Although the model correctly paraphrased a few samples, most of them were kept unchanged or didn't follow our original semantic intent. 80 assume that because of these failure cases that our model didn't perform better with the augmentation. Some selected examples of success and failure cases are presented in section A.3

5. Discussion

5.1. Conclusion and Discussion

This work develops a flexible language-based human-robot interface that allows a user to modify existing robotic trajectories. Our method leverages pre-trained large language and image models (BERT and CLIP) to encode the user's intent and target objects directly from a free- form text input and scene images, fuses geometrical features generated by a transformer encoder network, and outputs trajectories using a transformer decoder.

Our model can modify robot trajectories in 3D and velocity spaces. The output trajectory can be post-processed and applied towards diverse different platforms such as manipulation, aerial vehicles and legged robots. We provide a comprehensive set of simulated and real-world experiments demonstrating the effectiveness of our model and highlighting insights into what the model is learning. We validate our approach on diverse user studies, quantifying the model's its performance against benchmarks and different types of interactions.

5.2. Future directions

The applications of the method presented can extend beyond geometrical modifications of trajectories. In future iterations of this work, we seek to explore additional modalities such as force inputs, as well as the ability of the model to interact with the user over longer time horizons and multiple instruction inputs. We hope that our framework can serve as a building block for a novel paradigm in human-robot collaboration that employ large language models.

Furthermore, the implications of this work could surpass its initial target of human robot interactions. We aim to explore the use of such a natural language interface to facilitate the learning processes of Reinforcement Learning (RL) algorithms. Since it allows humans

to easily intervene and correct generic trajectories, it could be used to guide the RL algorithm's convergence during a semi-supervised training process through sparse meaningful interactions.

5.3. Final remarks

This thesis takes a step into building large pre-trained foundational models for robotics and shows how such models can create more intuitive and flexible interactions between human and machines. We hope that the concept and methods proposed in this work can be used to expand the capabilities of real-world robotics systems in multiple other applications, and allow a more trustful and interactive paradigm in human-robot cooperations.

A. Appendix

A.1. 2D simplified case

As an initial implementation of our language-based trajectory modification system, we addressed the problem in a simplified manner. In this section, we describe this initial 2D approach and depict its main differences comparison to our final method described in the chapter 3.

A.1.1. Simplified problem definition

One typical application for our systems is that of a user re-configuring a robotic arm trajectory that, although already avoids collisions, gets uncomfortably close to particular fragile obstacles in the environment. We design the trajectory generation system with a sequential waypoint prediction decoder, which takes into account multiple data modalities from geometry and language into a transformer network. The modified trajectory should be as close as possible to the original one throughout its length and respect the original start and goal constraints, while obeying the user's semantic intent. Fig.A.1 depicts the expected model behavior in a typical use-case scenario addressed in this simplified case.

Let $\xi_o : [0, 1] \rightarrow \mathbb{R}^2$ be the original robot trajectory, which is composed by a collection of N waypoints $\xi_o = \{(x_1, y_1), ..., (x_N, y_N)\}$. We assume that the original trajectory is a reasonable path from the start to the goal positions (*i.e.* avoids collisions) and can be pre-calculated using any desired motion planning algorithm, but falls short of the full task specifications. Let L_{in} be the user's natural language input sent to correct the original trajectory, such as L_{in} = "Stay away from the wine glass". Let $\mathcal{O} = \{O_1, ..., O_M\}$ be a collection of M objects in the environment, each with a corresponding position $P(O_i) \in \mathbb{R}^2$ and semantic label, such

as $L(O_i) =$ "glass". Our goal is to learn a function f that maps the original trajectory, user command and obstacles towards a modified trajectory ξ_{mod}, which obeys the user's semantic objectives:

$$\xi_{mod} = f(\xi_o, L_{\text{in}}, \mathcal{O}) \tag{A.1}$$

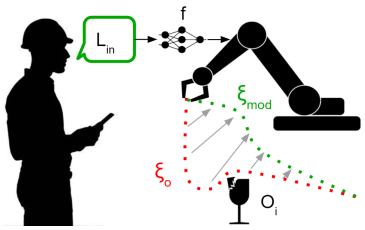

Figure A.1.: Typical use case for trajectory reshaping. The user's natural language command L_{in} is processed by function f to reshape the original robot trajectory relative to the target object O_i.

A.1.2. Context and interaction representation

One of the main differences between this 2D method versus our finial approach is the process of representing language instruction and context. Similarly to our main approach (section 3.1.2), we employed a pre-trained language model encoder, BERT [4], to produce semantic features $q_{\text{BERT}}(z^{\text{in}}|L_{\text{in}})$ from the user's input. The difference relied in the context representation, instead of using the textual and image encoder from CLIP [17], we make use of only the textual encoder to extract latent embeddings from both the user's text and the M object semantic labels ($q_{\text{CLIP}}(z|L)$). Figure 3.1 shows this encoder modification performed to adapt this purely textual approach to incorporate image representations of the context.

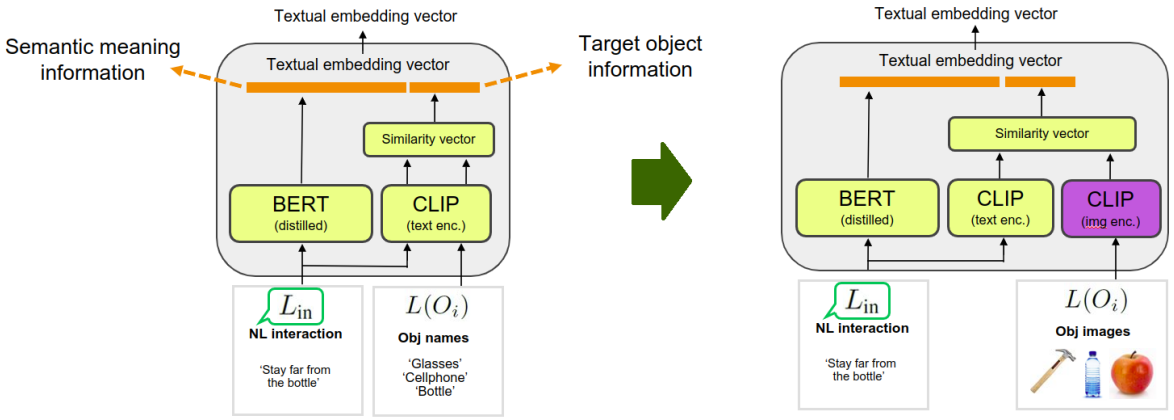

Figure A.2.: Adaptation of the initial language-based encoder into the language and image-base encoder presented in section 3.1

This conversion is extremely facilitated by the fact that the CLIP model was pre-trained to represent text and images in a joint latent space.

A.1.3. Synthetic 2D data generation

For our simplified 2D dataset, also followed a procedural generation approach and for this 2D cases, we targeted mainly 2 types of natural language interaction:

- Distance changes: modifying distance from the original trajectory to a specific desired object
- Cartesian changes: modifying the trajectory towards a specific direction ("front", "back", "left", "right").

In order to generate the natural language interactions, we followed the same procedure describe in subsection 3.3.1.

Trajectory generation:

We employed an A^* planner to generate reasonable initial trajectories ξ_o in randomized environments with different object configurations, and based on a set of pre-determined semantic combinations, we used the CHOMP motion planner [55] to compute ξ_{mod} by modifying weights of different cost functions. Our vocabulary involved different directions relative an object (closer or further away from $\cdot$, to the left/right/front/back of $\cdot$), intensity

changes (a bit/little, much, very), and a thousand object labels sampled from the ImageNet vocabulary. We generated a total of 10,000 trajectory labels. Fig. A.3 displays examples of original and reshaped trajectories.

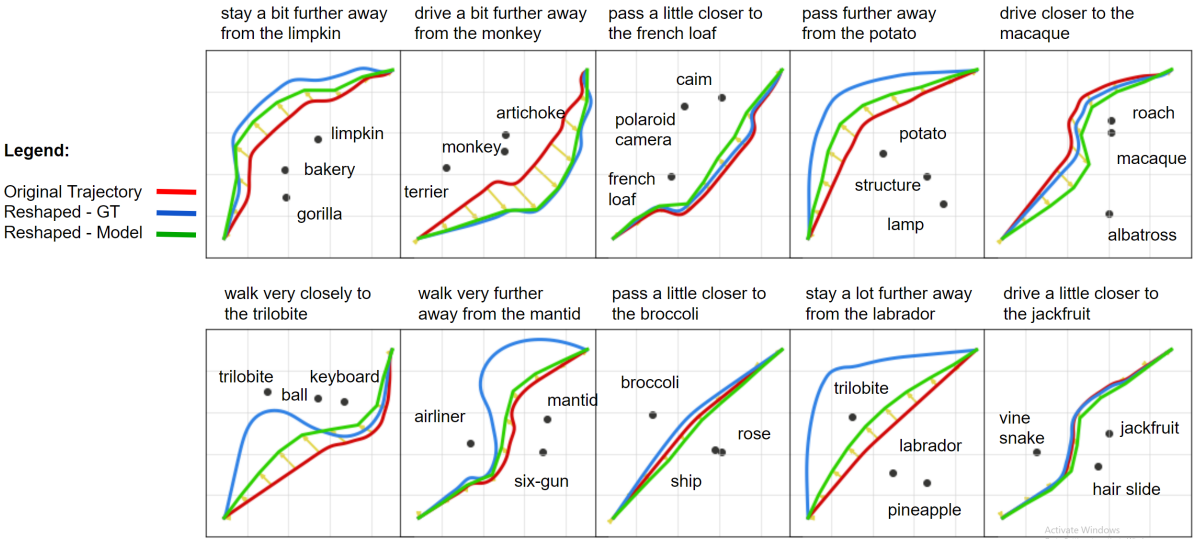

Figure A.3.: Randomly picked planning problems extracted from our validation set. Different colors display the original trajectory (calculated using A^*), the ground-truth reshaped trajectory (calculated using CHOMP), and the reshaped trajectory outputted by our model.

A.2. Additional random samples

Figures A.4 and A.5 display random data samples present in the testset and the respective model predictions using our main approach (Chapter 3). In red, the original trajectory; in green, the dataset ground truth trajectory; in blue, the modified trajectory.

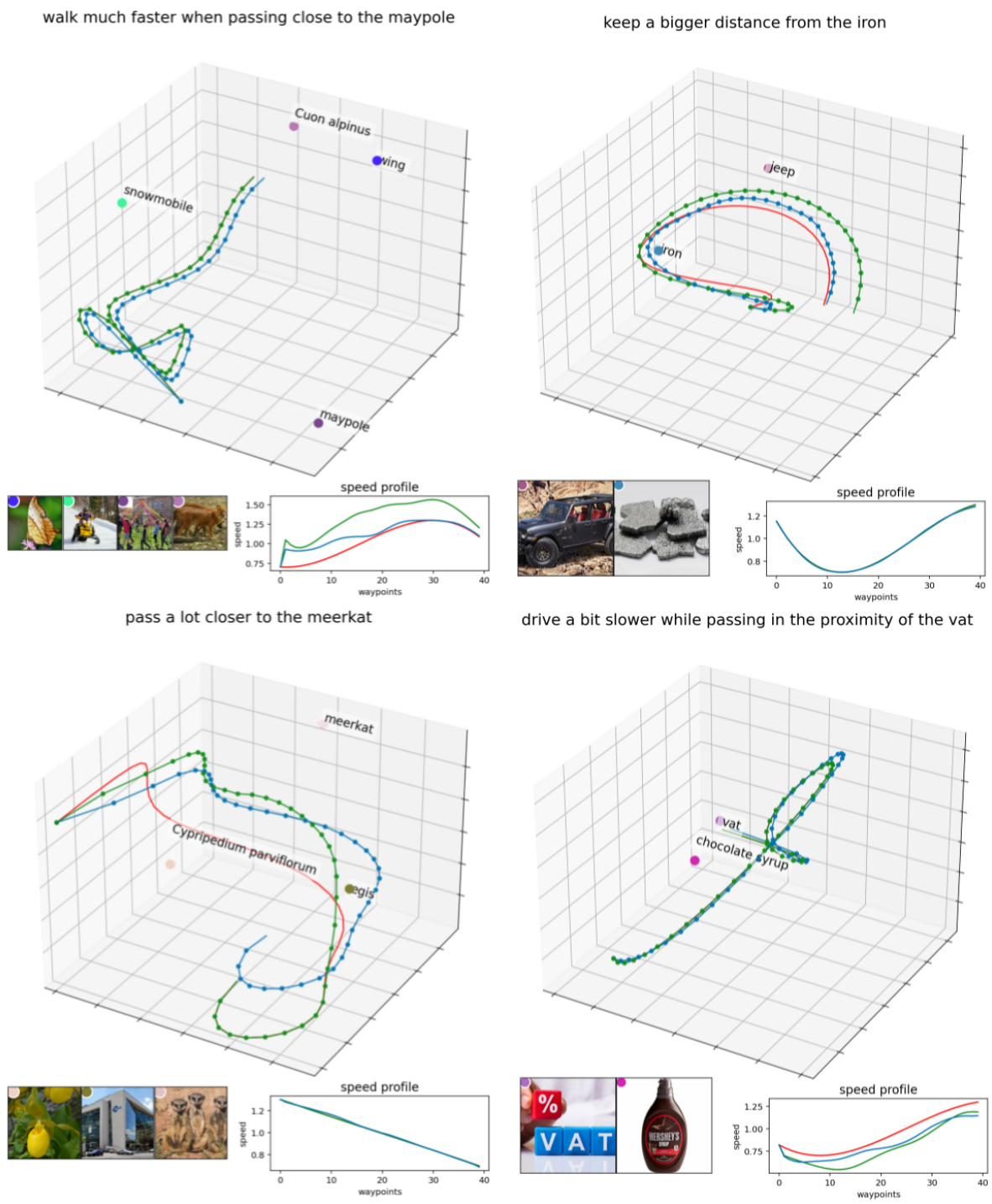

Figure A.4.: Random samples 1

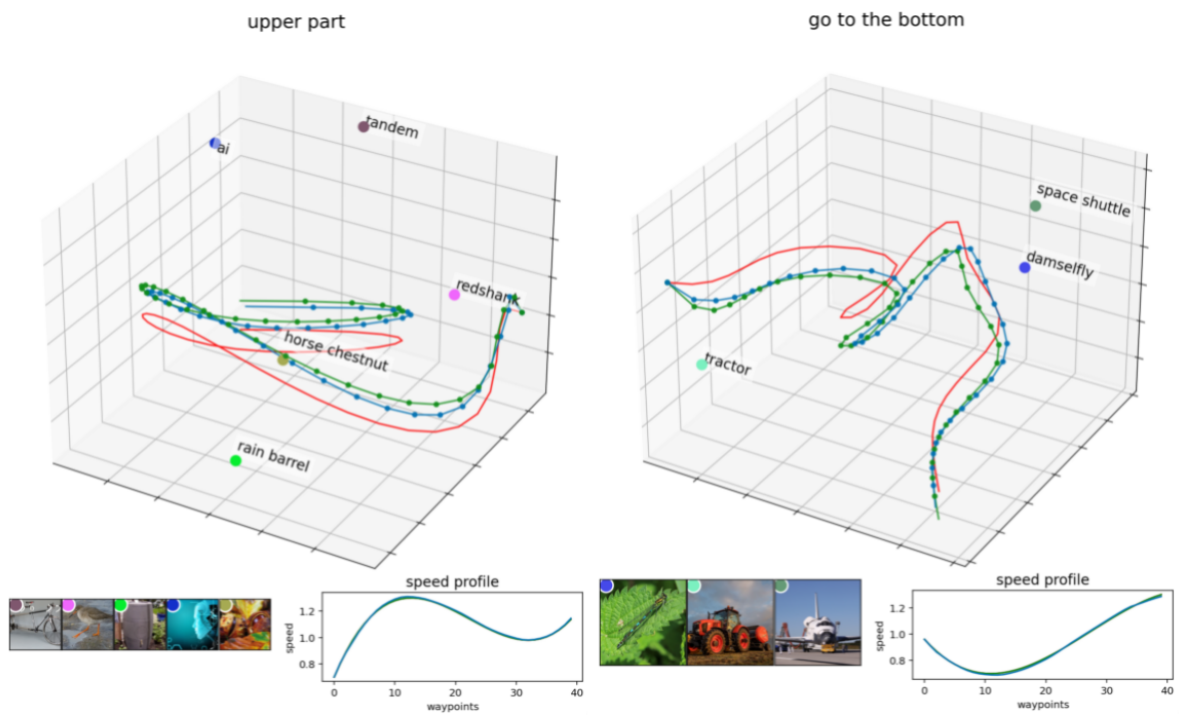

Figure A.5.: Random samples 2

A.3. Textual augmentation samples

Success examples:

Original: pass much further away from the ambulance

Rephrased: Pass far further away from the ambulance.

Original: walk slower when passing nearby the hair drier

Rephrased: When passing nearby the hair drier, walk slower.

Original: drive very further away from the water jug

Rephrased: drive very far away from the water jug.

Original: increase the speed while passing in the surrounding of the chimpanzee

Rephrased: In the surrounding area of the chimpanzee, increase the speed while passing.

Original: drive much faster while passing close to the tiger cat
Rephrased: While passing close to a tiger cat, drive much faster.

Fail examples:

Original:walk a little further away from the sports car
Rephrased: A little distance away from the sports car, walk a little further.

Original: drive a little closer to the ski
Rephrased: drive a little closer to the ski?

Original: walk closer to the vine snake
Rephrased: How do you walk closer to a vine snake?

Original: stay on the front
Rephrased: stay on the front of the line.

A.4. Supplementary videos

- **Reshaping Robot Trajectories Using Natural Language Commands: A Study of Multi-Modal Data Alignment Using Transformers** [13]
  Video: `https://www.youtube.com/watch?v=fhSOb3z7aXE`
- **LATTE: LAnguage Trajectory TransformEr** [14]
  Video `https://www.youtube.com/watch?v=yCSZcCJEoPc`

List of Figures

List of Tables

Bibliography

[1] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug. "Interactive, collaborative robots: Challenges and opportunities". In: *IJCAI International Joint Conference on Artificial Intelligence*. 2018, pp. 18–25. ISBN: 9780999241127. DOI: 10.24963/ijcai.2018/3.

[2] L. M. Hiatt, C. Narber, E. Bekele, S. S. Khemlani, and J. G. Trafton. "Human modeling for human–robot collaboration". In: *International Journal of Robotics Research* (2017), pp. 1–16. ISSN: 17413176. DOI: 10.1177/0278364917690592.

[3] A. Jain, S. Sharma, T. Joachims, and A. Saxena. "Learning preferences for manipulation tasks from online coactive feedback". In: *Int. J. Robotics Res.* 34.10 (2015), pp. 1296–1313. DOI: 10.1177/0278364915581193. URL: https://doi.org/10.1177/0278364915581193.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[6] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, et al. "Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model". In: *arXiv preprint arXiv:2201.11990* (2022).

[7] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song. "CLIP on Wheels: Zero-Shot Object Navigation as Object Localization and Exploration". In: *arXiv preprint arXiv:2203.10421* (2022).

[8] M. Shridhar, L. Manuelli, and D. Fox. "Cliport: What and where pathways for robotic manipulation". In: *Conference on Robot Learning*. PMLR. 2022, pp. 894–906.

[9] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. "Do as i can, not as i say: Grounding language in robotic affordances". In: *arXiv preprint arXiv:2204.01691* (2022).

[10] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox. "Correcting Robot Plans with Natural Language Feedback". In: *arXiv preprint arXiv:2204.05186* (2022).

[11] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[12] C. E. Garcia, D. M. Prett, and M. Morari. "Model predictive control: Theory and practice—A survey". In: *Automatica* 25.3 (1989), pp. 335–348.

[13] A. Bucker, 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: *International Conference on Intelligent Robots and Systems (IROS)* (2022).

[14] A. Bucker, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, and R. Bonatti. "LaTTe: Language Trajectory TransformEr". In: *arXiv preprint arXiv:2208.02918* (2022).

[15] R. Liu and X. Zhang. "A review of methodologies for natural-language-facilitated human–robot cooperation". In: *International Journal of Advanced Robotic Systems* 16.3 (2019), p. 1729881419851402.

[16] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258* (2021).

[17] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[18] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, et al. “Florence: A New Foundation Model for Computer Vision”. In: *arXiv preprint arXiv:2111.11432* (2021).

[19] H. Alwassel, D. Mahajan, B. Korbar, L. Torresani, B. Ghanem, and D. Tran. “Self-supervised learning by cross-modal audio-video clustering”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9758–9770.

[20] S. Ma, S. Vemprala, W. Wang, J. Gupta, Y. Song, D. McDuff, and A. Kapoor. “COMPASS: Contrastive Multimodal Pretraining for Autonomous Systems”. Feb. 2022.

[21] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. “Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents”. In: *arXiv preprint arXiv:2201.07207* (2022).

[22] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould. “A Recurrent Vision-and-Language BERT for Navigation. arXiv 2021”. In: *arXiv preprint arXiv:2011.13922* ().

[23] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. “Concept2robot: Learning manipulation concepts from instructions and human demonstrations”. In: *The International Journal of Robotics Research* 40.12-14 (2021), pp. 1419–1434.

[24] W. Goodwin, S. Vaze, I. Havoutis, and I. Posner. “Semantically Grounded Object Matching for Robust Robotic Scene Rearrangement”. In: *arXiv preprint arXiv:2111.07975* (2021).

[25] A. S. Chen, S. Nair, and C. Finn. “Learning generalizable robotic reward functions from" in-the-wild" human videos”. In: *arXiv preprint arXiv:2103.16817* (2021).

[26] J. Fu, A. Korattikara, S. Levine, and S. Guadarrama. “From language to goals: Inverse reinforcement learning for vision-based instruction following”. In: *arXiv preprint arXiv:1902.07742* (2019).

[27] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. “Language-conditioned imitation learning for robot manipulation tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 13139–13150.

[28] P. Goyal, R. J. Mooney, and S. Niekum. "Zero-shot task adaptation using natural language". In: *arXiv preprint arXiv:2106.02972* (2021).

[29] 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". In: *The International Journal of Robotics Research* 39.10-11 (2020), pp. 1279–1304.

[30] M. R. Walter, S. Patki, A. F. Daniele, E. Fahnestock, F. Duvallet, S. Hemachandra, J. Oh, A. Stentz, N. Roy, and T. M. Howard. "Language Understanding for Field and Service Robots in a Priori Unknown Environments". In: *arXiv preprint arXiv:2105.10396* (2021).

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[32] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek. "Robots that use language". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), pp. 25–55.

[33] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, et al. "Habitat 2.0: Training home assistants to rearrange their habitat". In: *Advances in Neural Information Processing Systems* 34 (2021).

[34] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. "Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3674–3683. DOI: 10.1109/CVPR.2018.00387.

[35] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. "Videobert: A joint model for video and language representation learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7464–7473.

[36] J. Lu, D. Batra, D. Parikh, and S. Lee. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks". In: *Advances in neural information processing systems* 32 (2019).

[37] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao. "Unified vision-language pre-training for image captioning and vqa". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13041–13049.

[38] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai. "Vl-bert: Pre-training of generic visual-linguistic representations". In: *arXiv preprint arXiv:1908.08530* (2019).

[39] W. Hao, C. Li, X. Li, L. Carin, and J. Gao. "Towards Learning a Generic Agent for Vision-and-Language Navigation via Pre-Training". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 13134–13143. DOI: 10.1109/CVPR42600.2020.01315.

[40] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer. "Vision-and-dialog navigation". In: *Conference on Robot Learning*. PMLR. 2020, pp. 394–406.

[41] K. Nguyen and I. Daumé. *Help, Anna! Visual Navigation with Natural Multimodal Assistance via Retrospective Curiosity-Encouraging Imitation Learning*. Sept. 2019.

[42] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso. "Transformer networks for trajectory forecasting". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 10335–10342.

[43] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. "Decision transformer: Reinforcement learning via sequence modeling". In: *Advances in neural information processing systems* 34 (2021).

[44] M. Janner, Q. Li, and S. Levine. "Offline Reinforcement Learning as One Big Sequence Modeling Problem". In: *Advances in neural information processing systems* 34 (2021).

[45] R. Bonatti, A. Bucker, S. Scherer, M. Mukadam, and J. Hodgins. "Batteries, camera, action! Learning a semantic control space for expressive robot cinematography". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 7302–7308.

[46] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. "Roboturk: A crowdsourcing platform for robotic skill learning through imitation". In: *Conference on Robot Learning*. PMLR. 2018, pp. 879–893.

[47] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[48] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *arXiv preprint arXiv:1910.13461* (2019).

[49] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. "Large batch optimization for deep learning: Training bert in 76 minutes". In: *arXiv preprint arXiv:1904.00962* (2019).

[50] E. Rohmer, S. P. N. Singh, and M. Freese. "CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework". In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. www.coppeliarobotics.com. 2013.

[51] J. Redmon and A. Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[52] R. Vortmeyer-Kley, P. Nieters, and G. Pipa. "A trajectory-based loss function to learn missing terms in bifurcating dynamical systems". In: *Scientific reports* 11.1 (2021), pp. 1–13.

[53] M. Cuturi and M. Blondel. "Soft-dtw: a differentiable loss function for time-series". In: *International conference on machine learning*. PMLR. 2017, pp. 894–903.

[54] R. Meattini, A. Bernardini, G. Palli, and C. Melchiorri. "sEMG-Based Minimally Supervised Regression Using Soft-DTW Neural Networks for Robot Hand Grasping Control". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10144–10151.

[55] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. "CHOMP: Gradient optimization techniques for efficient motion planning". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 489–494.